A Snapshot of Public Web Services

Jianchun Fan & Subbarao Kambhampati
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406
Email: *{jianchun.fan, rao}@asu.edu*

Abstract

Web Service Technology has been developing rapidly as it provides a flexible application-to-application interaction mechanism. Several ongoing research efforts focus on various aspects of web service technology, including the modeling, specification, discovery, composition and verification of web services. The approaches advocated are often conflicting—based as they are on differing expectations on the current status of web services as well as differing models of their future evolution. One way of deciding the relative relevance of the various research directions is to look at their applicability to the currently available web services. To this end, we took a snapshot of the currently publicly available web services. Our aim is to get an idea of the number, type, complexity and composability of these web services and see if this analysis provides useful information about the near-term fruitful research directions.

1 Introduction

With the rapid development of the internet technology, the World Wide Web is being used more and more in application to application communication beyond the current human-machine interaction. Web Service technology has received much attention in the last few years as it aims to provide flexible machine to machine interaction mechanism over the web. Web Services, or software services accessible via standardized protocols, are viewed as the potential fundamental infrastructure for the future web oriented distributed computation. The academic and industry research efforts have proposed many standards to formalize many aspects of web service technology, including communication, invocation, monitoring, discovery and composition of services [SK03;BF+03;TP04].

There are currently many directions in the frontier research of web service technology. The directions pursued are often conflicting—based as they are on differing expectations on the current status of web services as well as differing models of their future evolution. Some implicitly assume that primarily applications of web services are likely to be on the public web, while others assume that most applications of web services are likely to be in intra-corporate scenarios. The assumptions do affect the research directions pursued. For example, those considering public web services assume that it is infeasible to expect machine-interpretable service descriptions on the public web. They thus tend to focus on discovery and composition given just syntactic (text-based) descriptions [DH04;HK03;AG+03;CS+03]. In contrast, those looking at intra-corporate web services expect complex, access-restricted but well-annotated services, and focus on such as automated or semi-automated composition, verification and monitoring of services [SH+03;MM03;TP04].

One way of deciding the relative relevance of these research directions is thus to investigate to what extent the current ground reality of the web services conforms to the assumptions made. To this end, we decided to take a snapshot of the existing public web services.[1] Our aim is to get an idea of the number, type complexity and composability of these web services and see if this analysis provides useful information about the near-term fruitful research directions. The main contribution of this paper is to describe the results of our study and discuss its implications.[2]

We will start by providing a brief description of the current research directions in web services. We will then describe the methodology we have used to take a snapshot of the public web services. Taking a snapshot itself turned out to be reasonably complex because of the largely unstructured nature of the publicly available

[1] Ideally a snapshot of intra-corporate services is also needed. However, getting a fair sampling of these services is harder than those on the public web.

[2] The snapshot was taken in June, 2004. The collected raw data is available at http://rakaposhi.eas.asu.edu/PublicWebServices.zip

web services. We first describe the details of how we crawled web services from a large number of registries, removed duplicates and validated the services. We then describe how we subjected the resulting set of services to a variety of automated and manual analyses. Finally, we describe the implications and lessons of these analyses for the research in web services.

2 Overview of Current Research Directions in Web Services

Web services are software services distributed on the internet. They are accessible on the internet through the standard web communication protocols such as HTTP. The invocation of a web service is done by platform independent and language neutral message exchange between the client and the server, which makes web services differ from other distributed computation models such as RPC and makes web services a more flexible infrastructure to build web oriented and inter-enterprise applications. This loosely coupled open environment, together with the possible service registration facilities, increase the potential of dynamic combination of existing services together.

Many standards have emerged recently to formalize web services at the level of communication (SOAP), description (WSDL, OWL-S), composition (BPEL4WS, OWL-S) and discovery (UDDI).

- SOAP (Simple Object Access Protocol) specifies the XML serialization for typed data and provides a XML envelop for messages exchanged between client and server. This is the lowest level of service invocation specification [SOAP].
- WSDL (Web Service Definition Language) is a grammar that describes web service as communication endpoints capable of message exchanging. The interfaces of the operations and invocation grounding information are specified in the WSDL files of services as parts of the service profiles to be published in the service registry [WSDL].
- BPEL4WS (Business Process Execution Language for Web Service) is an XML based work flow definition language which describes how individual services are connected to join a business process. BPEL4WS provides rich control structures to combine primitive activities, such as invocation of an individual service, into complicated business logic [BEPEL4WS].
- OWL-S (Ontology Web Language for Services, formerly DAML-S) supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. OWL-S markup of Web services will facilitate the automation of Web service tasks including automated Web service discovery, execution, interoperation, composition and execution monitoring. [OWL-S]
- UDDI (Universal Description, Discovery and Integration) provides a standard way for publishing and discovering information about web service. A UDDI registry provides the facilities for the service providers to advertise their services in some standard industry taxonomy and also for the user to query the desired service profile [UDDI].

Two of the most popular problems in web service technology addressed by both industry and academia are service *discovery* and *composition*. At an abstract level these efforts could be classified into two main trends: one is promoted by the leading industry organizations in which the syntax of the service interfaces are specified in WSDL and the composition is done in a work-flow style language such as BEPL4WS. In this approach UDDI registries are for the service provider to register their services under the predefined industry taxonomy and the registries provide the facilities to search the registered services. The search is mostly keyword search in the name or text description of the services. The underlying assumption of this approach is that the service providers will mark up their service profiles using English descriptions so that they could be easily understood by application developers who try to integrate the service into their applications, and that the service provider will register his service properly in the UDDI registry and provide enough text description so that one can search and locate their services. This however makes automated discovery much harder as English text descriptions are not machine interpretable. This has lead to a slew of research efforts aimed at “extracting” higher level descriptions and service classifications from WSDL descriptions [DH04;HK03].

On the other hand, the semantic web community argues for adding more semantics into the web services so their meaning and functionality are specified in an unambiguous and machine-interpretable way (e.g. OWL-S). The motivation for this is that the composition could be done in an automated or semi-automated way by the software agents that are able to reason on the semantic specifications upon the underlying ontologies [CS+03]. The discovery of such services is more like matching of functionalities and properties beyond pure text keyword search. The underlying assumption of this approach is that there are well-defined domain ontologies and the services are marked up properly with those ontologies. The process of reasoning with the ontologies would then help locate the services with desired functionalities and properties. The main question here is how feasible is it to expect semantically marked up services.

An important way to decide which of the approaches is more relevant will be to have an idea on what type of application will web services support in the near future. There are two diverging views here – some argue that web services will find more use in intra-corporate scenarios. In such cases, it is likely that services will be annotated by the providers using a consistent ontology. Service discovery is likely to be less of a challenge and supporting non-trivial service composition is feasible.

Others see the main role for web services to be on the public web – with multiple services being available to lay users. In this case, the dream of consistent semantic annotation seems less feasible. We are likely to find mostly free text annotations of services, making automated service discovery, and the attendant extraction of semantics from syntactic descriptions, a pressing problem.

While we cannot gauge the use of web services in intra-corporate scenarios, it is possible to take a snapshot of public web services. This is what we do here – in the hope of that it will shed light on what models of evolution of web services are closer to current reality.

3 Snapshot of Current Web Services

By taking a snapshot of the public web services we wanted to see (1) how many public web services were there, (2) how complex they were, (3) how diverse they were and (4) how meaningfully they were documented.

At first glance, getting a snapshot of what services are actually available on the public web would seem easy because we have the UDDI registries promoted by many leading industry organizations. But the truth is, the current UDDI registry system is still evolving and not very mature. There is no mechanism of verification or business model that could enforce the service providers to only register services that are well implemented and ready to be understood and integrated to user applications. In fact, current UDDI registries such as *uddi.ibm.com* allow anybody to register almost anything as a web service entry, and when we looked into the registries, a very large portion of those registered services were either "hello-world"-style simple testing or experimental services or not actual services at all.

Many of the registry entries do not even have a valid WSDL file URL, let alone the actual end point of the services. So obviously the UDDI registries are not good start for us to have a good picture of what services are available online. There are some other major online web service registries though, which do not necessarily conform to UDDI standards and do not yet have very large number of registered entries, but these registries have much higher percentage of services registered that are actually available. We took a comprehensive study on the web and found several largest and most representative web service registries, or directories (see below). The union of the registered services on these registries seems to cover a large portion of all the ones available online and represents their properties and features to a reasonable degree. So we took these registries as the source of the collection of the real web services.[3]

To find out what services are there, we first crawled these service registries, and then processed the data collected to remove the invalid entry and duplicates. Then we performed a text description and documentation based clustering on the collected services, trying to classify the available web services in terms of their functionalities and properties.

3.1 Crawling the Registries

To collect information about the current available services, we wrote several crawlers to fetch the registered information of the web services. The registries we crawled are:

- www.bindingpoint.com
- www.salcentral.com
- www.xmethod.com
- www.webservicex.com
- www.webservicelist.com

These registries usually have the query facilities to do the keyword lookup or category browsing on the registered information. The services registered usually have the information about the names, providers, text descriptions and the URL of the WSDL files. We collected all this information and in addition followed the URL and fetched the WSDL files into our database. Sometimes the URLs do not point to the WSDL files but rather to the introductory html pages of the services, in such cases we followed this kind of link and tried to find the WSDL file URL in the pointed page too. To filter the invalid registry information which is very common in all the registries, we discarded the collected service entries which do not have a valid URL to their WSDL file or to a page that contains a URL of a WSDL file. Here we only look at the string representation of

[3] Admittedly, our snapshot will not cover the web services which are available but not registered in any of the known registries. It would be interesting to develop a focused crawler to gather such services.

the URL to decide if it points to a WSDL file and later we further validate the collected WSDL files.

This kind of simple filtering works well for the crawling. From the above registries we collected 2432 registered services at the time of crawling. After filtering the invalid entries we have 1544 entries with a valid WSDL URL. The collected information, including service name, provider, text description as well as the content of the WSDL files were saved in our local database.

3.2 Removing Invalid Entries and Duplicates

Some of the registered services might not have a valid WSDL file entry, or the WSDL file is not a well-formed xml document, or the WSDL file does not conform to the WSDL standards. We considered such entries as invalid ones. There are also a lot of duplicates among the collected service entries.

To remove the invalid entries we parsed every fetched WSDL file first to see if it is a valid XML document and eliminate the invalid ones from the database. Then for the rest, we performed a simple check of their conformance to the WSDL standards by checking the existence of several necessary WSDL tags inside the file and eliminated the invalid ones. To remove the duplicates, we used the combination of service name and provider's name as the key and checked the duplicates based on the keys. [4]

This step removed all the invalid entries and most of the duplicates. We got 640 valid entries out of the total 1544 entries in the collection. There are some hidden duplicates left in the collection, usually because of typos in the names or slightly different versions of the same service registered in different places.

Next, we performed both a manual and automated clustering to classify these collected web services in terms of their functionalities. The automated one is done to see how effective text categorization techniques are in classifying and subsequently discovering web services.

4 Automated Clustering of the Available Services

Our initial motivation in clustering the services was the belief that proper clustering would help the retrieval of services. Without structured semantics in service descriptions, the keyword based search is the simplest way for the users to specify their requirements. Simple keyword lookup might not show all the potential candidates that could satisfy the user's requirement, but by taking the correlation or similarity among services into consideration, more relevant services can be retrieved. Our hypothesis was that the automatically generated clusters will be able to suggest similar services.

The automated clustering is done based using text clustering techniques. We used information from three parts of the service description:

- the text description provided when they are registered in the registries;
- the documentation fields of services in their WSDL files and
- the documentation fields of individual operations of services in their WSDL files.

We view the union of all these three parts of information of each service as a bag of words to do the clustering.

We began by processing the bags of words to enhance the quality of the later clustering. We started with standard word stemming and stop word elimination. After the first running of the clustering, we found that there are some domain specific stop words in this clustering problem. For example the word "string", "return", "information", "web", "service", etc. appear in many of the service text descriptions, and other words such as all the html tags "font", "h1", etc. are also very common in the registration information of the services. These words are all eliminated during the pre-processing to improve the quality of clustering.

The clustering uses the Hierarchical Agglomerative Cluster (HAC) [SK+00] algorithm and the Jaccard Similarity [RE92] as the distance measure between service descriptions. The collection has as many clusters as the number of services at the beginning and then we continuously merge the closest cluster until there is only one cluster in the root. The result of the HAC clustering is a binary tree and tends to have too many levels in the tree. So after the clustering, we performed a flattening step on the tree. The flattening is done by checking the tightness of each child of a given node. If a child's tightness is less than the distance between the child itself and the node, then all its children will be merged as the children of the (parent) node. The flattening starts from the root to the leaves of the tree. We used average pair wise distance of all the children of a node as the tightness measure of that node. A single iteration of flattening might not give sufficiently good quality of the hierarchy so here we do the flattening repeatedly until

[3] There are some minor issues here. For example some providers might use slightly different service names or provider's names in different registries. For example for the same service, the name registered in one registry might contain space or other special characters but not in another registry; and a provider's name would be registered as XYZ.com in one registry and XYZ in another. All these issues are handled in the duplicate removing step.

the whole computation converges, i.e. there is no change during any iteration of the flattening.

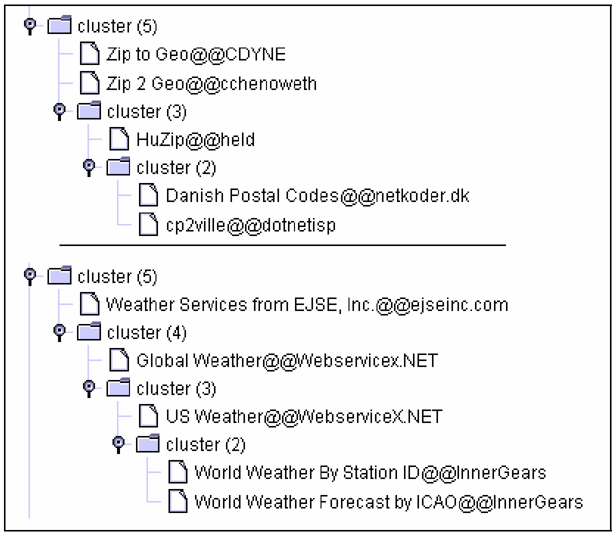
cluster (5)
Zip to Geo@@CDYNE
Zip 2 Geo@@cchenoweth
cluster (3)
HuZip@@held
cluster (2)
Danish Postal Codes@@netkoder.dk
cp2ville@@dotnetisp
cluster (5)
Weather Services from EJSE, Inc.@@ejseinc.com
cluster (4)
Global Weather@@Webservicex.NET
cluster (3)
US Weather@@WebserviceX.NET
cluster (2)
World Weather By Station ID@@InnerGears
World Weather Forecast by ICAO@@InnerGears

Figure 1 Two Subtrees of the Automatic Clustering

The clustering of the collected services works well and it captures most of the functional similarity of the individual services. For example in Figure 1 we have two subtrees of the cluster: one of them contains all services about zip code lookup and the other one contains services about weather forecast.

We also noticed that there is a noticeable amount of noise in the clustering, which usually arises when a service does not have enough information to differentiate itself from others during the clustering. In fact, when we check the registered information of the services which appear to have no connection to other ones, most of them are those which have only very short text description registered and do not have any “documentation” field in their WSDL files. This is not surprising as we find that even human have difficulty assessing the functionality of these services. These will be further analyzed in later sections.

5 Manual Analysis of Types of Web Services Available on Public Web

As stated above, automated text clustering of the services captured much correlation of the services in terms of their functionality, as long as there is enough text description. However that still cannot give us a clear picture of what types of services are there. So we did a manual classification of the web services collected by checking the crawled text description, the WSDL file and the cluster got in the automated clustering. We have the result as the “classes” of these currently available services in Figure 2. We now present some analysis on this.

5.1 Diversity of Services

As we can see, the largest portion of the services (more than 45%) can be classified as data source lookup services, which have the same functionality as the current html form based web pages. Moreover another three large classes, “number conversion”, “sensing” and “data processing” can all be viewed as data source look up in some sense. These four groups together count up to 84% of all the services. In the following we will look at some important lessons learned from this classification.

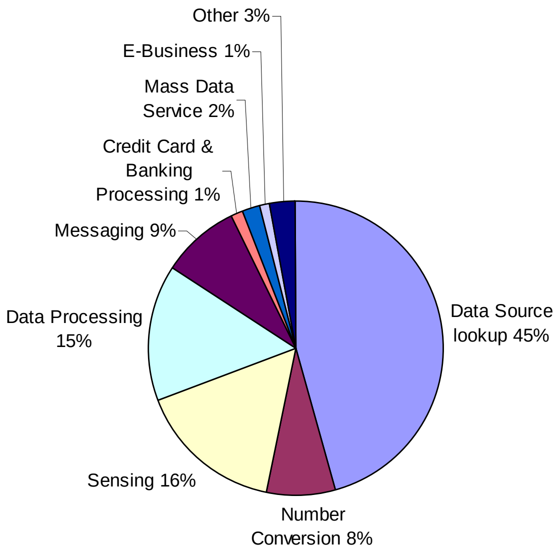

Figure 2 Types of Collected Web Services

The detailed classification of the services is shown in Figure 3.

5.2 (Operation) Complexity of the Web Services

One way of measuring the complexity of the web services is to see how many individual operations are involved in the individual services. We collected the information of the number of operations in each service as the measure of the complexity of the services. Figure 4 shows the distribution of this measure on the whole collection (of the 640 services).

- *Data Source lookup: 291*
    - *Search engine & Database lookup: 137*
    - *Geometry lookup & computation: 82*
    - *DNS and IP lookup, ping, etc. 34*
    - *Dictionary lookup & translation: 24*
    - *Email addresses validation: 6*
    - *Credit card validation: 8*
- *Number conversion: 49*
    - *Unit conversion: 31*
    - *Currency conversion: 18*
- *Sensing: 103*
    - *Time: 7*
    - *Weather: 15*
    - *Traffic: 2*
    - *Flight status: 4*
    - *News, headlines and real time statistics: 36*
    - *Stock quote: 39*
- *Data processing: 95*
    - *Mathematics computation: 37*
    - *Encryption, security: 20*
    - *Financial computation: 23*
    - *Text & document processing: 15*
- *Messaging: 56*
    - *Sending email & instant message: 25*
    - *Sending fax: 10*
    - *Sending SMS message: 21*
- *Credit card & bank account processing: 9*
- *Mass data service: 12*
- *E-Business: 7*
- *Other: 18*

Figure 3 Detailed (Manual) Classification of Collected Web Services

More than 77% of the services have less than 5 operations and more than 36% of them have only one operation. Moreover when we looked into the WSDL files of the services with multiple operations, more often than not the operations do not have interactions among them. Very few services, specifically the less than ten E-business services, have more complicated inter-operation semantics (which is not explicitly defined in the WSDL file). We also tried to find some interesting composition of the services by manually checking the compatibility of the operations among these services, but it turned out that no composition with more than 2 operations could be found in this collection. It seems that at least at the current stage we do not have large numbers of public services which are both very complicated and have the potential to be composed with other services. The motivation to research of the composition of "complicated" web services must come from intra-corporate scenarios.

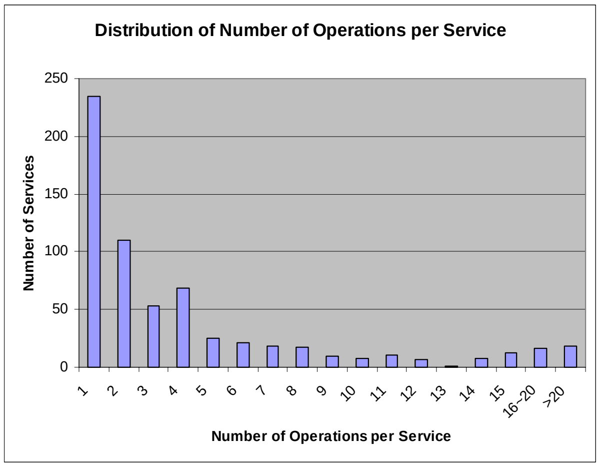

Figure 4 Distribution of Number of Operations per Service

5.3 Quality of the (WSDL) Service Descriptions

From another point of view, given the current available services, if an application developer simply wants to use a service in his application, are those services ready to be used? For a developer to integrate a service into his application a key problem is to understand both semantically and syntactically about the services and the operations they support. The only way for the developers to get the semantics of the services is to read and interpret the text description and the documentation of the services. The amount and accuracy of these textual resources directly determine if the semantics could be interpreted correctly. As stated above, these types of information are used in our clustering and we noticed that sometimes these text resources are not enough for the clustering algorithm to make good classification of the services. One may argue that the current WSDL standards are not machine oriented and the WSDL files are supposed to be consumed by human being. However it is questionable as to whether the service providers are seriously using the WSDL files as the way to convey the correct interpretation to the developers who will use them. To settle this, we performed a statistical analysis on the available services registration information.

We first collected the information of the lengths of the text description of the services (including the registration information and the "documentation" field of the service in their WSDL files) as the measure of amount of information conveyed in the service profiles. Figure 5 shows the distribution of the lengths (in terms of number of words) in the collection of the 640 services.

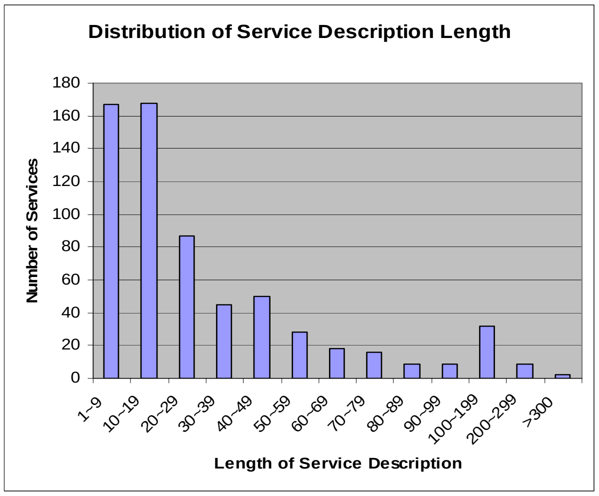

Figure 5 Distribution of Service Description Length

As we can see, most of the services (>80%) have text descriptions less than 50 words and more than 52% of the services have text description with less than 20 words.

Usually a service contains multiple operations and Figure 6 shows the average length of the documentation fields of all the operations in each service in the whole collection of 640 services.

In this collection, nearly 80% of the services have the average documentation for each operation of less than 10 words, nevertheless almost half the services do not have any documentation for any of the operations supported (length = 0). Operations are the key elements in the WSDL files because they describe the interfacing information which directly determines if the operations can be used in the user's applications. It is clearly questionable as to whether the semantics of the operations can be described adequately with less than 10 words.

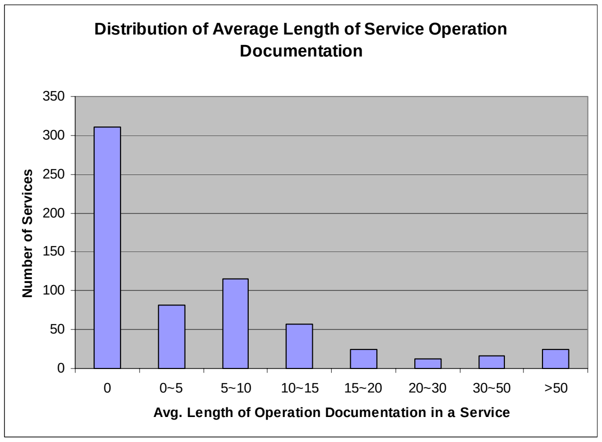

Figure 6 Distribution of the Average Length of Operation Documentation

From Figure 5 and Figure 6 we found that most of the services available online are not well documented. Since in the current model WSDL file and the registration information are the only source for the user to understand the functionality of the services, it is quite doubtful that the current ones available in the public web are ready to be used by the user without further human to human interaction.

6 Implications and Lessons Learned

From the statistics and analysis above we can look again at the current directions of research in web service from their relevance to the current web services available in the public web. A general caveat is in order before we proceed to enumerate the lessons—it is entirely possible that we are in the stone-ages as far as publicly available web-services are concerned; and that the type and complexity of publicly available services will improve significantly in the near future as the infrastructure standards take root. Nevertheless, we believe that it is worthwhile to evaluate the potential fruitfulness of the current research directions in web services from the stand point of the current snap shot of the public web services.

Service Types: One somewhat surprising result of our analysis is that most publicly available services are simply data sources that use SOAP protocols to support data sensing and conversion. Handling such services is just a variation of the standard data integration problem [KK02]. For example, the service composition is nothing but generating a query plan that accesses sources. The "conversation" between such services boils down to accepting queries and returning answers in SOAP format. As a matter of fact, some researchers are working on the dynamic composition of web services with data integration techniques by modeling web services as data sources [TK03]. In contrast, a lot of current work on web service composition, monitoring and execution assumes much more complex web services that have world-changing effects. While it is not surprising that data sources can be seen as services [JG04], what is surprising is how much of the public web services are just data sources!

The preponderance of data source-oriented web services also explains to some extent an apparent paradox in the approaches to service composition that have been advocated. Specifically, although in theory service composition is expected to involve complex plan synthesis (c.f. [TK03;SK03;SH+03]), several projects use composition techniques that are indistinguishable from query plan formation in data integration scenarios. (c.f. [TK03;PF02;KK02]).

Retrieval: A lot of research efforts on web services have concentrated on the service discovery/retrieval issue. The discovery issue is most critical for the publicly available sources. One interesting observation is that, if the text description such as WSDL and UDDI entry is the only source to describe the web services, the simple information retrieval techniques perform well, as shown in our text clustering experiments, as long as such descriptions are reasonably long. If that is the case, the problem of "discovery" itself is not likely to be a challenging one because the general discovery does not seems to be able to achieve more than what the current commercial search engines already do. Nevertheless the performance of service discovery depends on not only the techniques to "discover" but also the quality of the registration information of the registered services themselves, which currently are not guaranteed without a proper business model to enforce and verify the service publishing activities. While an argument can be made that retrieval will be more challenging as web services evolve and become more involved, it is also possible that the same evolution will advance the registry system such that there will be more structured entries on registries making retrieval easier.

Composition: We found that there are very few ways of composing services available online, mainly because of the lack of services and the correlations among them. Most of the current available services can be viewed as data sources with interfaces clearly defined with WSDL. Data sources with proper defined XML interfaces are easier to be integrated compared to current web database integration scenario because the integrators no longer need to screen-scrape the html pages to isolate the real data from the fancy representation. But when it comes to the problem of composition, it does not seem very different from the current data integration problem. Of course we have to admit that in intra-corporate scenarios there may be other types of "complex" web services with data updates, complicated interactions and other run time semantics involved, the composition as well as the verification and monitoring, of such services would be a challenging problem. All we can infer is that composition is not a pressing problem for public web services.

7 Related Work

Despite the amount of research devoted to web services, very little attention has been paid to understanding the nature of currently existing web services. The only exception that we know of is the work by Dong *et. al.* [DH04] which has been done around the same time as our own work. Although there are some similarities between our work and theirs in that both efforts crawl the web to aggregate web services, the overall aims of our projects are different. Their was aimed at supporting automated service discovery, where parameter names appearing in the service descriptions are clustered and the similarity between operations and operation input/output are quantified based on the that clustering. They also try to suggest possible composition of service operations (although, consistent with our results, they too find that there are no cases of composition which involve more than 2 operations). In contrast, we have used our crawl to analyze the existing web services and draw lessons from that analysis about fruitful research directions. In this sense our study is similar in spirit to that of Arnaud Sahuguet's investigation on DTDs in XML applications in the year of 2000 [SA00].

8 Conclusion

Web services are becoming more and more popular in both the industry and academic research. The relevant problems include the modeling, communication, composition, discovery, verification and monitoring of web services. Prior to the research on these problems we have to know what kind of services actually exist and on the other hand from the academic research point of view we have to figure out what is the shortcoming and defects of the current web service model and what problem should be handled as the future direction.

In this report we presented a snapshot of the web services currently available in the public web and discussed the relevance of various research issues of web service technology based on the data and statistics collected. We found that there is a big gap between the frontier research activities and the reality of the web services. We also argued that the problem of discovery is not a very feasible one given the syntactic specification and text description of services as the basis. Second we found that most current services can be viewed as data sources using WSDL to describe the interfaces and the composition of such services does not differ from the problem of data integration problem. The composition of more complex services may well be challenging problem, but the motivating scenarios are not likely to come from current public web services. We also found that the current WSDL standards are used more often for the documentation purpose rather than clearly defining the syntax and semantics of the services which is inadequate to be easily used by the application developers and the research on automated or semi-automated annotation of services would be a challenging topic.

In closing we would like to reiterate that all our observations and conclusions are based on the web services publicly available on the web. It would be interesting to do a similar study on the current status of the intra-corporate web services. Intra-enterprise web services and well controlled collaborative inter-corporation web services could have characteristics that are significantly different from those of the public ones covered in our snapshot. These are also scenarios where machine interpretable annotations may well be feasible, foregrounding the need for more complex composition and conversation frameworks (c.f. [BF+03;SK03;TK04]).

Acknowledgments

This work is supported by the ASU ET-I3 initiative grant ECR A601. We wish to thank Biplav Srivastava for his comments on the initial version of the report. We also thank Ullas Nambiar, Thomas Hernandez and Zaiqing Nie for many discussions and comments during this study.

References

[AG+03] Rama Akkiraju, Richard Goodwin, Prashant Doshi and Sascha Roeder, A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI, Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), August 2003.

[BF+03] Tevfik Bultan, Xiang Fu, Richard Hull, Jianwen Su: Conversation specification: a new approach to design and analysis of e-service composition. WWW 2003: 403-410.

[BPEL4WS] Business Process Execution Language for Web Services, http://www106.ibm.com/developerworks/library/ws-bpel/

[CS+03] Mark Carman, Luciano Serafini and Paolo Traverso, Web Service Composition as Planning, ICAPS 2003 Workshop on Planning for Web Services, June 2003.

[DH04] Xin Dong, Alon Halevy, Jayant Madhavan, Ema Nemes, Jun Zhang, Similarity Search for Web Services, VLDB 2004.

[JG04] Jim Gray, the Next Database Revolution, SIGMOD 2004 Keynotes.

[HK03]. Andreas Hess, Nicholas Kushmerick, Automatically attaching semantic metadata to Web Services, Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), August 2003.

[MM03] Daniel J. Mandell and Sheila A. McIlraith, Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation, Second International Semantic Web Conference (ISWC2003).

[KK02] Craig Knoblock and Subbarao Kambhampati, AAAI 2002 Tutorial on Information Integration on the Web. URL: http://rakaposhi.eas.asu.edu/i3-tut.html

[PF02] Ponnekanti, S. R., and Fox, A. 2002. SWORD: A Developer Toolkit for Web Service Composition. In Proc. of the Eleventh International World Wide Web Conference, Honolulu, HI.

[RE92] Rasmussen E. Clustering algorithms. In: Frakes WB, Baeza-Yates R, editors. Information retrieval : data structures & algorithms. Englewood Cliffs, N.J.: Prentice Hall; 1992. p. 419-442.

[SA00] Arnaud Sahuguet, Everything You Ever Wanted to Know About DTDs, But Were Afraid to Ask, WebDB 2000.

[SH+03] Evren Sirin, James A. Hendler, Bijan Parsia, Semi-automatic Composition of Web Services using Semantic Descriptions, Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003, April 2003.

[SK03] Srivastava, B., and Koehler, J. 2003. Web service composition--current solutions and open problems. ICAPS 2003. Workshop on Planning for Web Services, Trento, Italy.

[SK+00] M. Steinbach, G. Karypis, and V. Kumar, A comparison of document clustering techniques, KDD Workshop on Text Mining, 2000.

[OWL-S] OWL-S 1.0 Release, http://www.daml.org/services/owl-s/1.0/

[SOAP] Simple Object Access Protocol, http://www.w3.org/TR/soap/

[TK03] Snehal Thakkar, Craig A. Knoblock and Jose-Luis Ambite, A view integration approach to dynamic composition of web services, Proceedings of 2003 ICAPS Workshop on Planning for Web Services.

[TP04] Paolo Traverso, Marco Pistore: Automated Composition of Semantic Web Services into Executable Processes. International Semantic Web Conference 2004: 380-394.

[UDDI] The UDDI Technical White Paper. http://www.uddi.org

[WSDL] Web Services Description Language, http://www.w3.org/TR/wsdl