A Framework for Web Service Discovery: service's reuse, quality, evolution and user's data handling

Uddam CHUKMOL
LIRIS Laboratory, CNRS – INSA de Lyon
Blaise Pascal Building, 7 Jean Capelle Avenue, 69621 Villeurbanne Cedex
France
uddam.chukmol@liris.cnrs.fr

ABSTRACT
With proliferation of published Web services, the task of finding relevant ones for the developers of service oriented application becomes more and more difficult. Several existing tools or mechanisms allow this discovery; however, those approaches often skip different elements such as service's quality, reuse, evolution and users' comment making the discovery result feebly relevant to requesters' need and prevent the requesters from using up-to-date and available web services efficiently. In this research work, we present a framework for web service discovery taking into account the reuse of web service search result through caching technique, the quality of service through qualitative test result, the web service's evolution through version track technique and we provide a novel scheme of discovery using user's annotating information.

Keywords
Web service discovery, web service version evolution tracking, user's annotation, quality of web service, web service caching

1. INTRODUCTION
Web services are autonomous software components widely used in various service oriented applications according to their platform-independent nature. Web services are described before being published and they can be discovered and finally invoked via Web infrastructure by using the stack of known standards such as SOAP, WSDL and UDDI [1]. Despite their visible advantage and accessibility, the rapid growing number of published Web services prevents users or requesters (ordinary users or software developers) from finding easily and efficiently the services relevant to their specific needs. Discovery process can be realized either manually or automatically in the application design phase by software designers or in the execution phase by different software agents [2]. Nonetheless, this discovery can be done semi-automatically by combining the automatically found services and the choices provided by human users to refine final result. Thus, Web service discovery is undeniably considered as a vital phase in the process of service oriented application development.

Proceedings of the Second SIGMOD PhD Workshop on Innovative Database Research (IDAR 2008), June 13, 2008, Vancouver, Canada.

Several algorithms seek to deal with Web service discovery in different ways: from computing textual description similarity based on Information Retrieval techniques up to using complex logical formalism, particularly description logics, to rewrite queries and develop inference engines accordingly without setting aside the structural matching, ontology mapping and semantic processing (description logics, RDF, OWL, etc.) in their proposals. Except from focusing on the discovery of Web services itself, many other important features such as service reuse, service quality, and service evolution are often ignored. Users' comments or opinions on a found service are not completely or even poorly considered by the discovery system either, though they are a very interesting source of information capable of enriching the proved-to-be insufficient Web services' textual description [24]. Some other drawbacks found in most of the actual existing approaches represent the motivation and challenges (mostly seen from the users' perspective) of our study. They can be highlighted as following:

Challenge 1:
Web services found after a discovery process are poorly considered as a pertinent source of information or result to be reused in a future similar search (similar query).

Scenario 1:
Bob would like to include in his application a Web Service realizing a weather forecast given the name of a city. He uses a query with "weather forecast" as key word to search for the set of web services that would be relevant to his request. The Web service discovery system provides him a set of found services as result. Next time, Alice uses the same discovery system and the same keyword in her query to request for the same service. Assuming that there is no change in the source of the discovery tool (set of Web services available for search), the query of Alice is unavoidably computed once again by the system before providing the exact result comparing to what the system proposed to Bob.

Challenge 2:
Even when a Web service is found at the end of the discovery process, none can assure that the service is operational or obsolete.

Scenario 2:

The discovery system proposes to Alice a number of Web services relevant to her request. Alice would like to check if a chosen service is really usable in her application. She would prefer that the system propose a testing interface allowing her to make sure that the service she decides to utilize will operate correctly after being integrated into her application.

Challenge 3:

A Web service evolves just like other software components do in terms of version. Web service's users need particularly to be informed of any change made by the service provider in order to maintain their application up to date and operational.

Scenario 3:

The "weather forecast" Web service is provided by Jane. Bob uses this service inside his application. Jane is working on another better release of "weather forecast" and finally releases the actual version of "weather forecast" and still keeps the old version in her repository during a limited period of time. By searching in the discovery system using the same keyword again, Bob may realize that there are two versions of "weather forecast" made publicly accessible by Jane and that the latest version will be more appropriate to his application.

Challenge 4:

A new requester may learn more about a service if it is enriched by various comments from past users.

Scenario 4:

Bob describes his personal views on the "weather forecast" Web service provided by Jane and makes his comment publicly accessible after using successfully the mentioned service in his application. Ted is a new user of the Web service discovery system that Bob has used ever since. Ted searches in the system for a Web service related to "weather". The Web service provided by Jane is present in the result set output by the system. The comment of Bob on "weather forecast" Web service will support Ted's decision to choose it.

Challenge 5:

An only entry point to use the discovery system is not good enough to make it provide interesting and highly relevant result.

Scenario 5:

Ted would like to express his query other ways than using key word to discover a set of pertinent services. He may combine his key word based and his input-output based queries given to the system and get a better relevant set of Web services than the one output by the system when Ted provides uniquely a key word based query.

Aware of the exclusions mentioned above and perceived in the existing tools and mechanisms for Web service discovery, we will try to propose a all-in-one framework for Web service discovery handling the reuse of discovery result via the following skills: caching mechanism, the quality of service through qualitative systematic test, test by users, and the web service's evolution through version track technique. We provide as well, a novel scheme of discovery using user's annotating information.

This paper is organized as follows. Section 2 discusses related works and the current existing algorithms proposed in the field of Web service discovery. Section 3 presents and describes our proposal and finally Section 4 concludes and offers the future work perspectives.

2. RELATED WORK

As Web Services discovery is an important and difficult task in the development cycle of service oriented application, many algorithms, tools and mechanisms are proposed to solve for the best this problem. Woogle [3] makes use of a clustering technique by grouping the parameter's name of service operations into semantically significant concepts. These concepts are then used to compare the similarity between Web services' operations. Using this mechanism makes Woogle hard to deal with complex data types presented structurally as either a tree or a graph. [4] synthesizes the issues of UDDI related to the discovery based on catalog browsing and tries to solve the Web service searching by combining the textual description similarity of Web services with their semantic structural similarity. The textual description similarity is computed based on Information Retrieval technique through the usage of WordNet[1] thesaurus and the semantic structural similarity is finally dependent on the similarity of data types used in WSDL files. This approach may lead to erroneous semantic structural similarity when the structures of the data types in comparison have numerous identical sub-structures. [6] and [7] define the pair-wise similarity of Web services as the WordNet metrics distance between their textual descriptions. By simply neglecting the structural aspect of WSDL files, this idea may not provide exact final similarity of Web Services. WSXplorer [5], inspired by [4], proposes a Web Service discovery tool taking a textual description as input. This approach processes not only the textual similarity based on Vector Space Model technique of Information Retrieval domain but also the structural similarity by introducing a modified *tree edit distance* method in their comparison module. However, Web services' textual descriptions are not enriched by WordNet before and even during their similarity computing; and this approach does not after all handle the graph structure of WSDL files.

In the same attempt to make Web services discovery more efficient, in semantic Web community, certain approaches turn to the usage of Description Logics. [12], [13], [14], [15] and [23] use basically Description Logics to express their queries and develop inferring mechanisms accordingly to compute the similarity between Web services and a given query. Despite the interesting advantages of Description Logics, expressing a query in this formalism is still not an intuitive and easy task for an ordinary user or even for a software developer. Moreover, we do not see many of Description Logics based Web Service discovery tools made available to large public.

Instead of providing only one type of input query, [8] is an inspiring approach that offers two kinds of query: key word based and WSDL file based (i.e. requesters can look up for Web services by inputting key word or a WSDL file). Even though, this tool can provide interesting ranked result after the look up process, requesters can not go any further to make sure that a chosen service is not obsolete (i.e. there is no possibility for users

[1] http://wordnet.princeton.edu/

to test online the functionality of a Web service in the discovery result set). WSCE [9] provides a Web service discovery tool that, given an input keyword, proposes a set of similar Web services. This tool, by crawling different Universal Business Registries and Web service portals, collects the Web service's instances and stores them in a centralized registry. However, requesters desiring to test a discovered Web service or track its evolution (e.g. version) have no possibility to do so. [19] keeps the users' habit of information search by using conventional Web search engines (e.g. Google, Yahoo, etc) to discover published Web services. Due to the natural characteristic of WSDL files which is particularly partially text based, unlike HTML documents, a great effort will have to be done in incorporating WSDL files' content into HTML documents. Different types of meta-data will have to be added into HTML documents as well in order to facilitate indexing process done by Web search engines. Through this analytical study on the existing proposals related to Web service discovery, we can determine some conclusive remarks: **a)** Information Retrieval techniques are commonly used, **b)** Slight number of tools proposes multiple entry points to discover Web services (several kind of queries can be employed on single system), **c)** Testing a Web service online is rarely possible, **d)** Users' information or comments are frequently ignored, **e)** Users are not informed about the versioning or evolution of found Web services.

Thus, we come up with a schematic proposal, thoroughly described in section 3, of a novel platform for Web Service discovery including: **a)** The usage of multiple types of queries and strategies of result combination helping to leverage the quality of discovery output, **b)** The processing of users' participative annotation or tagging on Web services, **c)** The interface for online "aliveness" test on a found Web service and **d)** The detection and processing of changes undergone by Web services.

3. OUR PROPOSAL

In this section we present in detail our proposed framework (see Figure 1, 2, 3 and 4) that contributes in solving different issues discussed in the Section 1. This framework is composed of different modules and their roles are described in depth within this section.

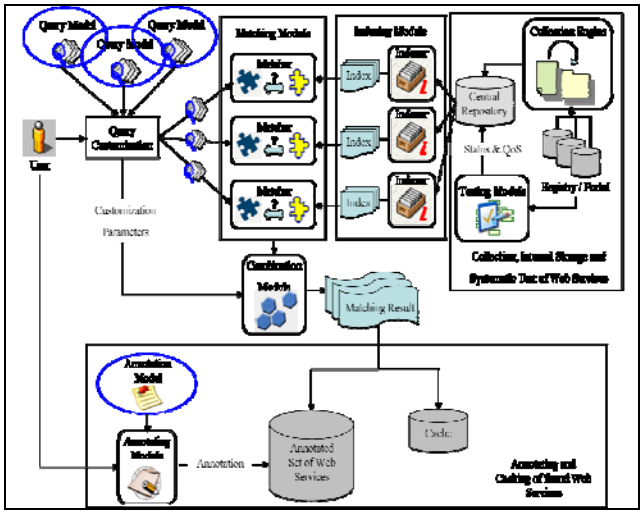

Figure 1: Overall view of our platform

We attempt also to provide a system allowing semi-automatic Web service discovery through a simple Web interface. A user or requester does not need to install anything, except a Web browser at their side allowing them to query and test online a Web service resulting from their request.

3.1 Collection, Internal storage and systematic test of Web services

This important part of our tool is done regularly in background and consists of collecting different Web services from various providers and storing them in a single and accessible source. [10] and [11] have proven that most of the existing Web discovery mechanisms do not utilize UDDI as the only source for the process but also and more frequently different registries or customized portals containing WSDL files. Inspired equally by [9], [17] and [22], the "Collection Engine" (see Figure 1) crawls the Web and identifies WSDL or WSDL-like files (there can be possibilities to find WSDL contents inside non-WSDL files e.g. XML or HTML files), validates them before inputting them into the central repository. If a WSDL file is valid, its copy and its location (URL) will be saved into the central repository. Additionally, our platform offers a possibility to the set of stored WSDL files to be enriched with the working status of a Web service (binary information indicating that the service is operational or not) and several other information related to the quality of Web service such as response time, availability, documentation, etc. The working status and the mentioned qualitative information are injected automatically into the central repository by the "Testing Module" (see Figure 1). This module is in charge of analyzing the content of each stored WSDL file and prepares automatically the test scenarios and data accordingly.

The systematic test's output will be used to update the correspondent WSDL instances in the central repository. By doing so, we can assure that the source employed in our discovery system is populated by several external sources making the information's richness in our tool better than using a single source of Web service provision. The output of "Testing Module" associated with each entry of our central repository offers valuable information supporting requesters' decision in choosing a discovered Web service.

3.2 Indexing Module

The content in central repository is indexed by different *indexers* (see Figure 1). Each *indexer* provides an index representing the entire entries in the central repository. Each index is used by a *matcher* in the "Matching Module" (see Figure 1). Every entry of each individual index has a reference corresponding to a Web service stored in the central repository. This indexing module is designed to be extensible because several *indexers* implementing various indexing techniques can be incorporated and used within this "Indexing Module". Proceeding this way, we believe that even though there is a common source used by the discovery process, it will be better optimized by the *indexers* to be employed more efficiently by a specific *matcher* according to a query expressed by the users.

3.3 Matching Module

This module hosts several *matchers* (see Figure 1). Each matcher implements a matching algorithm treating a kind of query

and using an index (defined in the previous section). These *matchers* are core components of our system because they compute the similarity between the query expressed by a user and the set of Web services in the corresponding index. The output of each *matcher* is an ordered set of Web services in descending order of similarity value with the input query (the higher the value is, the more exact the similarity becomes). Thus, a result of each matching process is a discovery's one.

3.4 Query customization and Combination module

We offer many entry points to use our discovery system via different types of query. Each query conforms to a query model (e.g. keyword based, input – output based, WSDL instance based, etc). Users can express different queries through the guided model proposed by our system. They can then customize the way their queries will be processed by the system. Parameters provided by users are used by the "Combination Module" (see Figure 1) to compute the final result of the discovery process. In this context of combination strategy, [16] proposes 3 kinds of combination techniques that can be done to the matching result sets: a) *mixed*: the result of query q_i is fused with the one of query q_j by weighted sum, b) *cascade:* the result of query q_i provided by the matching m_i is filtered by a query q_j using the matching m_j and c) *switching:* user decides to use the same query q_i but different matching techniques m_i and m_j to assess the result. In addition to these presented combination techniques, we also offer and include in our platform other Boolean-like combination operators such as: **a)** AND (intersection between the result provided by using a query q_i and the one provided by using query q_j), **b)** OR (union between the result of a query q_i and the one of a query q_j), **c)** MINUS (difference between the result of a query q_i and the one of a query q_j), etc.

We believe that such customization mechanism will allow users to have an enriched and more relevant result set of Web services proposed by our system rather than using a unique way to look up for Web services.

3.5 Annotation based discovery and Caching

This part is the cornerstone of our work and it defines our system as a participation platform giving consideration to users' comment on what they have discovered using our system. In effect, in this era of Web 2.0 [28], users do not only retrieve information but also share and participate in the content publication on the Web. We can see this kind of model in [25], [26] or [27] and users form a certain kind of community to share and participate in their published content therein. We therefore take into account the user's dimension in our platform as well. We, however, analyzed [18], [20] and [21] before proposing this search by annotation mechanism. In [18], each user has to install a component at his/her side in order to use the system. Every user's action is communicated to and logged in the remote client component. This approach uses *Implicit Culture* framework and helps a new user to discover Web services through a recommendation system based on past users' experiences. Despite this interesting technique, the transaction between client site and remote site may cause system overload if the number of sessions increases. [20] and [21] use machine learning techniques to leverage the discovery result given users' preference. We can remark some handicap in these approaches related to the number of users' examples to feed the systems before they return interesting results.

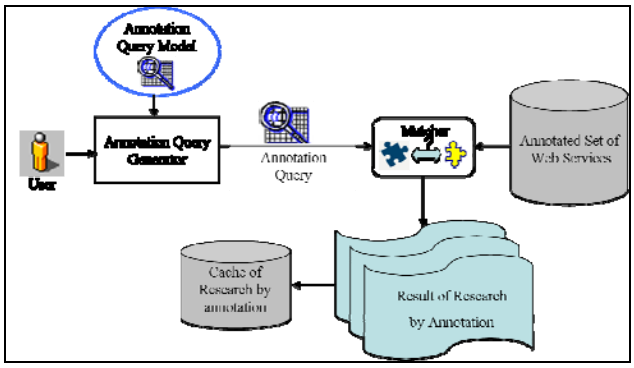

Figure 2: Search by annotation and result caching

Moreover, if users are not guided to express their preferences and the learning rules are not correctly defined, we are not able to assure that the output of such systems is sufficiently relevant.

Therefore, in our system, after a discovery process mentioned in the previous sections, users are provided possibility to simply tag with a cloud of words or structurally annotate a number of Web services in the whole result set (see Figure 2). The initial textual description of the found services will not be modified by this tagging or annotation. Before doing so, users are recommended to test (see section 3.6 and Figure 3) the services plausibly relevant to their initial need to better understand the services' effective functionalities. We opt for the *collaborative tagging* and the tagged information or annotation will be enclosed to the correspondent Web service and made publicly available in association with that Web service.

We consider in this case a Web service as a resource (e.g. similarly to photo on Flickr [26] or URL on del.icio.us [25]) on which we can add tags and annotations (a Web service can be tagged by many users). Considering various tagging styles perceived in Web 2.0 environment [28], we propose three types of tagging / annotation to users. They can use: **a)** keyword tags (a set of simple or composed words separated by a delimiter), **b)** free text tags allowing users to comment freely on a Web service by a free text in the form of sentence or paragraph and **c)** ***structural guided tags*** allowing users to tag a service using free text to fill in different information fields (organized according to a predefined structure) proposed by the discovery system. Users will be guided by the system to complete the tagging information inside a form.

In order to deal with these three forms of tags efficiently, we propose a matching module (see Figure 2) which embeds relatively information retrieval techniques (for the fact that we work with natural language tags) capable of providing interesting result via textual search. We currently conduct a study to solve this issue.

Through usage of this mechanism, the tagged or annotated set of Web services will be enlarged and other users will be able to search among the mentioned set for relevant Web services using simple keyword, free text or structured query (the construction of this kind of query will be guided by our system) and they will be

more and more supported in terms of Web service selection by consulting others' past discovery experiences.

Additionally, in order to reuse the past discoveries' result, the system saves regularly, after each search, the query and the output result in the cache. Further, other users' queries will then be passed through the cache first before going to the "Matching Module". If there's a result related to the same query, the result will be displayed directly without any cost of matching computing done by the system. A Web service will be eliminated from the cache if it is undergone any change.

3.6 User test

After every discovery process, either through ordinary query or annotation, the system will provide an online test interface (after analyzing the output WSDL file), allowing users (guided by the system) to test the "aliveness" of every service appeared in the result set. Users will have to enter necessary input data and submit the test. A test report will be displayed back to users at the end of the testing (see Figure 3).

The concepts and techniques used in the testing process within software engineering domain will be studied in detail and adapted to the Web service testing context.

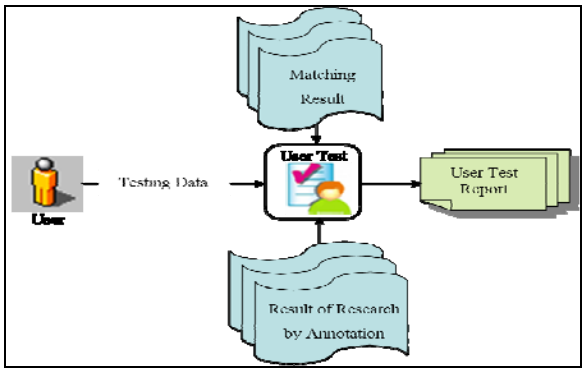

Figure 3: Web service's "aliveness" test by users

3.7 Evolution tracking

This module aims at providing users the working and up-to-date Web services only. Nevertheless, users can still be informed about the change made by the service providers to correspondent services.

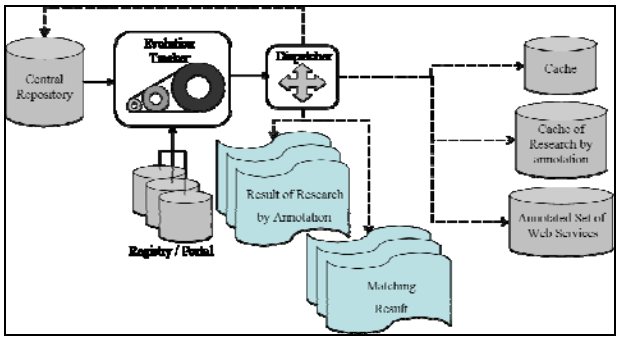

Figure 4: Evolution tracking

"Evolution tracker" (see Figure 4) is a background running process comparing regularly the centrally stored Web services with their original instances in different registries or portals. Any physical (lexical, syntactical, or structural but same functionality) or behavioural (functional modification) change detected by this module is transferred to the "Dispatcher" (see Figure 4) to trigger the necessary change in different storage entities such as cache, matching result, result of search by annotation, etc. The change action to execute immediately on storage entities is replacing the old version of identified Web services with the new ones and informing the users the detected evolution information.

4. CONCLUSION AND FUTURE WORK

We presented in this paper a novel platform aiming at making Web service discovery process more efficient in terms of relevancy and providing sufficient means to identify the quality of a service before being chosen to integrate in any service oriented application. We incorporate the participative characteristic of Web 2.0 in our system considering users' data in terms of tagging or annotation and propose a new scheme in our discovery system by using this information. However, we do not set aside different proposals in the Semantic Web Service community such as SAWSDL[2] or OWL-S[3] and consider them as another alternative mechanism for annotating Web services. We take into account the detection of evolution undergone by stored services and inform requesters of any change perceived.

In order to have a qualitative evaluation of our approach, we are currently working on our first prototype, in which the following elements will be studied, defined and implemented: **a)** Query models for ordinary discovery and search by annotation, **b)** Indexing technique, **c)** Matching algorithm and matching algorithm for search by annotation, **d)** Evolution tracker and dispatcher, **e)** Automatic qualitative test, **f)** User test report model, **g)** Collection engine, **h)** Various strategies tackling the combination of discovery result – query combination customization – customization parameters automatic tuning mechanism, and **i)** Central repository and cache's internal storage structure.

The study and implementation of the annotation or tag based discovery component is in progress. We expect 6 month effort for this key idea realization.

5. ACKNOWLEDGMENTS

Special thanks to my advisors: Dr. Nabila BENHARKAT (nabila.benharkat@liris.cnrs.fr) and Pr. Youssef AMGHAR (youssef.amghar@liris.cnrs.fr) for their patience, constructive advices, attention and efficient supervision.

6. REFERENCES

[1] M. P. Papazoglou and D. Georgakopoulos. Service Oriented Computing. Communication of ACM, Volume 46, Number 10, October 2003. pp. 25 – 28.

[2] J. D. Garofalakis, Y. Panagis, E. Sakkopoulos and A. K. Tsakalidis. Contemporary Web Service Discovery Mechanisms. Journal of Web Engineering, Volume 5, Number 3, September 2006. pp. 265 – 290.

[2] http://www.w3.org/TR/sawsdl/

[3] http://www.w3.org/Submission/OWL-S/

[3] X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes and J. Zhang. Similarity Search for Web Services. In the proceeding of International Conference on Very Large Data Bases (VLDB), 2004, August 31st – September 3rd, Toronto, Canada. pp. 372 – 383.

[4] Y. Wang and E. Stroulia. Semantic Structure Matching for Assessing Web Service Similarity. In the proceeding of 2nd International Conference on Service Oriented Computing, ICSOC 2003, December 15th – 18th, Trento, Italy. pp. 194 – 207.

[5] Y. Hao, Y. Zhang and J. Cao. WSXplorer: Searching for desired Web services. In the proceeding of 19th International Conference on Advanced Information System Engineering, CaiSE 2007, June 2007, Trondheim, Norway. pp. 173 – 187.

[6] J. Wu and Z. Wu. Similarity based Web service Matchmaking. In the proceeding of 2005 IEEE International Conference on Service Computing 2005, SCC 2005, July 11th – 15th, Florida, USA. pp. 287 – 294.

[7] Z. Zhuang, P. J. Mitra, A. Jaiswal. Corpus based Web service Matchmaking. In the proceeding of the 20th National Conference on Artificial Intelligence 2005, AAAI 2005, July 9th – 13th, Pennsylvania, USA. 6 pages.

[8] K. H. Lee, M. Y. Lee, Y. Y. Hwang and K. C. Lee. A Framework for XML based Web services retrieval with Ranking. In the proceeding of IEEE International Conference on Multimedia and Ubiquitous Engineering 2007, MUE 2007, April 26th – 28th, Seoul, South Korea. pp. 773 – 778.

[9] E. Al-Masri and Q. H. Mahmoud. WSCE: A Crawler Engine for large-scale Discovery of Web Services. In the proceeding of IEEE International Conference on Web Services 2007, ICWS 2007, July 9th – 13th, Utah, USA. pp. 1104 – 1111.

[10] D. Bachlechner, K. Siorpaes, D. Fensel et I. Toma. Web Service Discovery – A reality check. Technical Report, January 17, 2006. DERI – Digital Entreprise Research Institute.

[11] J. Becker, K. Bachhaus, H. L. Grob, T. Hoeren, S. Klein, H. Kuchen, U. Müller-Funk, U. W. Thonemann et G. Vossen. Web service discovery – Reality Check 2.0. Working papers No. 5, ERCIS – European Research Center for Information Systems. ISSN 1614-7448.

[12] B. Benatallah, M. S. Hacid, A. Leger, C. Rey and F. Toumani. On Automating Web Services Discovery. In VLDB Journal 14(1). pp: 84 - 96, 2005.

[13] M. Stollberg, U. Keller, H. Lausen, S. Heymans. Two-phase Web Service Discovery based on Rich Functional Description. In the proceedings of 4th European Semantic Web Conference (ESWC'07), June 3rd - 7th 2007, Innsbruck, Austria. pp: 99 – 113.

[14] M. Kifer, R. Lara, A. Polleres, C. Zhao, U. Keller, H. Lausen, D. Fensel. A Logical Framework for Web Service Discovery. In the proceeding of 3rd International Semantic Web Conference (ISWC'04). 16 pages. November 8th 2004. Hiroshima, Japan.

[15] R. Lara, M. Angel Corella et P. Castells. A flexible model for web service discovery. In the proceeding of 1st International Workshop on Semantic Matching and Resource Retrieval – Issues and perspectives. September 11th 2006, Seoul, South Korea.

[16] N. Kokash, W. J. Van den Heuvel and V. D'Andrea. Leveraging Web Services Discovery with Customizable Hybrid Matching. In the proceeding of ICSOC 2006. LNCS 4294, 2006. pp. 522 – 528.

[17] C. Platzer and S. Dustdar. A Vector Space Search Engine for Web Services. In the proceeding of the 3rd European Conference on Web Services, ECOWS 2005. November 14th – 16th, Vräxj, Sweden. 9 pages.

[18] N. Kokash, A. Birukou and V. D'Andrea. Web Service Discovery Based on Past User Experience. In the proceeding of 10th International Conference on Business Information Systems, BIS 2007, April 25th – 27th, Poznan, Poland. pp. 95 – 107.

[19] H. Song, D. Cheng, A. Messer and S. Kalasapur. Web Service Discovery Using General-Purpose Search Engines. In the proceeding of 2007 IEEE International Conference of Web Services, ICWS 2007, July 9th – 13th, Utah, USA. pp. 265 – 271.

[20] L. Kovacs, A. Micsik and P. Pallinger. Handling User Preference and Added Value in Discovery of Semantic Web Services. In the proceeding of 2007 IEEE International Conference of Web Services, ICWS 2007, July 9th – 13th, Utah, USA. pp. 225 – 232.

[21] Y. Zhou, L. Zhang, L. Zhang, B. Xie and H. Mei. User Feedback-Based Refinement for Web Services Retrieval using Multiple Instance Learning. In the proceeding of 2006 IEEE International Conference of Web Services, ICWS 2006, September 18th – 22nd, Chicago, USA. pp. 471 – 478.

[22] C. Atkinson, P. Bostan, O. Hummel and Dietmar Stoll. A Practical Approach to Web Service Discovery and Retrieval. In the proceeding of 2007 IEEE International Conference of Web Services, ICWS 2007, July 9th – 13th, Utah, USA. pp. 241 – 248.

[23] S. Agarwal and R. Studer. Automatic Matchmaking of Web Services. In the proceeding of 2006 IEEE International Conference of Web Services, ICWS 2006, September 18th – 22nd, Chicago, USA. pp. 45 – 54.

[24] J. Fan and S. Kambhampati.: A Snapshot of Public Web Services. ACM SIGMOD Record, Volume 34, Number 1, March 2005. pp. 24 – 32.

[25] Delicious Web Site: http://del.icio.us (accessed on 18th March 2008)

[26] Flickr Web Site: http://www.flickr.com (accessed on 18th March 2008)

[27] Facebook Web Site: http://www.facebook.com (accessed on 18th March 2008)

[28] What is Web 2.0?: http:// www.oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html (accessed on 18th March 2008)