Web Service Discovery with Implicit QoS Filtering

Natallia Kokash

DIT - University of Trento, Via Sommarive, 14, 38050 Trento, Italy
email: natallia.kokash@dit.unitn.it

Abstract. Web Service (WS) discovery is a critical problem hindering web service technology proliferation. The current solution, based on catalog-style browsing, provides no control over the quality of registered services. Application of matching techniques for WS retrieval is still under investigation. The objective of this work is the design of a framework to improve WS discovery. Our approach is based on application of distributed recommendation system to provide Quality of Service (QoS) information and on testing of retrieval methods on service specifications.

1 Introduction

WS paradigm is a promising model of software technology, based on loosely coupled, distributed and independent services operating via the web infrastructure. To overcome platform and language dependence, services are described using Web Service Description Language (WSDL). Standardized XML-based interfaces help performing service reuse. Service descriptions are cataloged in Universal Discovery, Description and Integration (UDDI) registries. Although there exists a stack of standards to regulate the communication of processes and automated tools to convert legacy applications into web services, WS technology is still not widely used. One of the reasons is the lack of means to support WS discovery, i.e., the identification of existing WSs that can be used by new web applications. This problem is rather extensive and admits various interpretations [4]. Under automated discovery, a requester agent performs service search and evaluates the results. Currently UDDI registries are the dominating technological basis for WS discovery. They allow business compliance and reuse, and in perspective they could provide control over data and facilitate WS lifecycle management. But existing registries are still small and mostly private, there is no control over provided information, qualitative characteristics of WS and ability of quality-based retrieval. The discovery supported by UDDI API is inaccurate as services retrieved may be inadequate due to low precision and low recall. Such mentioned disadvantages determine our objective: we propose a framework for efficient WS discovery that provides clients with QoS information and reduces the probability of failure by analyzing statistics of previous service invocations.

The rest of the paper is organized as follows. In Section 2, we discuss different aspects of the problem and present the existing work relevant for our research.

Section 3 describes our approach. In Section 4, we draw some conclusions and outline future work.

2 Background

We can judge how well a web service satisfies a client's goal using various types of matching. *Signature matching* considers only function types without regarding their behavior. *Specification matching* is a way to compare two software components, based on descriptions of their behaviors. Components can be compared with various degrees of accuracy (exact and relaxed matching). However, this approach requires formal pre/postcondition specifications. Hausmann et al. [5] examine the application of such methods for WS discovery. WS matching is related to the automatic schema matching which is a basic problem in many application domains like data integration or semantic query processing. Rahm et al. [13] provide a good survey in this area. In *syntactic matching* we look for the similarities into data using syntax driven techniques. In *semantic matching* the key intuition is the mapping of meanings. For example, *surname*, *family name*, *cognomen* and *last name* represent the same concept. One of the relevant proposals in the field of WS discovery is to use an extension of UDDI that contains WSDL specifications [7]. In this way, dynamic retrieval through common terminology and shared meaning is enabled. WSDL does not provide any special semantic specifications but it contains the `<documentation>` element with service documentation and elements with natural language descriptions of operations and data types. Identifiers of messages, operations and data types are meaningful, and XML syntax allows to capture the domain specific relations. *Semantic Web services*, i.e., web services empowered with formal ontologies [16], revealed new perspectives in service automatic discovery, composition and execution monitoring (e.g., [10]). Since this solution is based on predefined ontologies, it is not flexible and contradicts the dynamic nature of web-based interaction.

From the WS discovery perspective, the major criterion of WS retrieval system evaluation is the relevance of found services with respect to the end user. The most popular quality evaluation measures of Information Retrieval (IR) engines are *precision* (a measure of the usefulness) and *recall* (a measure of the completeness). They have a fixed range and are easy to compare across different queries and engines, but they do not account for the quality of ranking. For a machine learning algorithm we need an effective single number measure. An *average precision* that combines precision, relevance ranking, and overall recall is an ideal measure for our task where good ranking is extremely relevant due to invocation cost of the services.

Assuring the WS quality is an especially critical task since we do not know service exact specifications, developing and testing methodologies. An invocation of a troublesome service can affect all the system. A potential client should be aware of the service behavior before he/she/it starts to exploit it. Among the basic QoS factors are service performance (throughput, response time, latency, transaction time), availability, accessibility, reliability, scalability, exception handling, execution cost, reputation, regulatory, accuracy, integrity, interoperability, security (authentication, authorization, confidentiality, traceability, data encryption, non-repudiation), privacy, network-based factors (network delay, delay variation, packet loss), etc. [12]. Service reputation is an especially interesting property from this list. Kalepu et al. [8] address the problem of subjective perception of reputation due to the lack of performance history. Our proposed solution allows to infer WS rankings from pure statistical data about service invocations based on particular criteria for each client.

An important issue is the way in which a client asks for a service. Currently used by UDDI API keyword-based queries are easy to express but obviously not sufficient. Using WSDL specifications as a query appears more promising. In the case when a client wants to substitute one web service with another without affecting the observable behavior of the system, he/she does not need to write a request. But in general, WSDL specification is redundant and does not contain QoS requirements. Ran [12] suggests SOAP-based requests. In the requirements for WS discovery competitions [2] requests are described in XML format. A similar style can be used to express client's preferences about QoS characteristics and testing samples.

In IR approaches to WS discovery a query consists of keywords, which are matched against the stored descriptions in the UDDI. The Boolean model suffered from either lots or very few returned results. Traditional models like Vector-space model seem to be quite effective. Latent Semantic Indexing (LSI), the prevailing method for small document collections, was applied on the UDDI to capture the semantic associations between services [14]. A suite of algorithms for similarity assessment between two WSDL descriptions were developed [15]. They are based on IR and component matching methods. The WordNet database is applied for semantic analysis of service documentation. According to those experimental results, the methods are neither precise nor robust. Dong et al. [3] describe an approach based on term associations analysis. In [6], WS discovery and composition based on syntactic matching is suggested. Authors propose to build indexes on operations and part names to speed up the matching process. Oh et al. [11] describe the transformation of WS discovery and composition issues into a graph search problem. Web services are compared depending on syntactic matching of input and output parameters whereas data types are ignored.

Retrieval is successful when the information need is satisfied. Therefore, we need to be provided with the user's judgement of the relevance of the found data. Maximillien et al. [9] describe an agent-based system where agents act as proxies to collect information and build a reputation of semantic web services. In [17] trust management mechanism is added to evaluate the credibility of the user reports when predicting service quality. By nature, most of us are not inclined to give feedback. The solution is suggested by a recommendation system that uses implicit feedback from the user by means of analyzing his/her behavior. Birukov et al. [1] present such a system to produce recommendations for web search. In our task, agents can control actions like user search requests, test invocations of web services, bindings, successful service responses or failures, etc.

3 A Framework for WS Discovery

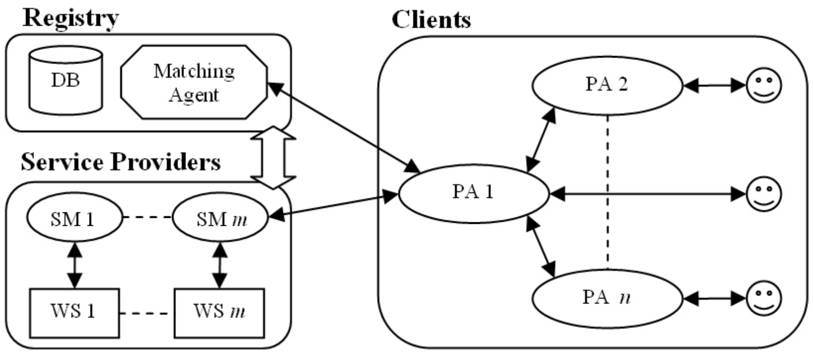

Fig. 1. Web service discovery model. *Personal Agent (PA)* accepts requests from its *client* and sends them to *Registry. Matching Agent* provides lists of relevant services and agents that searched for them. *PA* asks agents and *Service Mediators (SMs)* for information, they provide it. *PA* ranks the services and invokes the best one. Both *PA* and *SM* store execution statistics.

In Fig. 1 a basic architecture of our proposed framework is presented. Each client or group of local clients with similar preferences has a dedicated agent which task is to process all activities pertaining to web services (communication with registries, bindings, requests, responses). The *personal agent* accepts a request from its client (human or application), redirects the description of the goal to the *matching agent* which manages the *registry.* The *matching agent* searches for the services that can conceivably satisfy the client's goal. It keeps history of requests, and provides the requester with the information about the agents that searched for the similar services before. Given a list of previous clients, that is the last k agents to which the *matching agent* recommended the service, the *personal agent* can ask them for recommendations. Then two scenarios are possible: a first one, in which the agents that invoked the services provide the requester with their own rankings, and an alternative one, in which they render to it parts of their own history leaving to the requester the task of processing it. The latter approach tries to overcome the problem of specific client's preferences that cannot be provided by external QoS evaluation systems. The *personal agent* can ask first the agents that gave useful statistics before, and only if they do not have necessary information, to apply more extensive interrogation. An open issue is preventing malfeasant agents from affecting the service reputation. If web services use mediators that collect statistics about all their invocations, the *personal agent* may have more reliable data by comparing parameters provided by agents and the *service mediator.* Another advantage is that centralized data is more convenient for evaluation of such QoS factors as availability, reliability, capacity, robustness.

The *matching agent* provides the *personal agent* with the matching score of its request and returned web services. If this score is low, the relevance of the services is dubious and alternative approaches are needed to check if they correspond with the client's request. Recording of invocation inputs and outputs can help to test the service before usage by new clients. A client sends testing samples to his/her/its agent, and the agent checks if they are among the data it has. If it is so we may strongly believe that the service fits the client's goal and will work properly. But this approach conflicts with security and privacy issues, and may be time and space consuming.

The *personal agent* ranks the web services by combining the scores of matching, recommendations and tests (if they were provided). Before service invocation it asks the client for confirmation. Finally, the best-fit web service is invoked, QoS parameters are measured, and the results are saved. Then, the agent can ask its user if it should always invoke this service for the same query. Such an explicit user feedback can be used to reconstruct the trust policy to other agents or rating algorithm but it is not necessary. If a user keeps invoking a web service, he/she is likely to be satisfied with the service provided. Therefore, repeated usage should have a high weight for QoS evaluation.

4 Conclusions and Future Work

In this paper we have presented a new WS discovery approach. Some subproblems, namely service matching methods, discovery system evaluation and request formats have been discussed. We have proposed a framework for WS selection based on QoS properties collected implicitly by a distributed agent-based system.

We intend to apply the recommendation system presented in [1] for WS discovery. To achieve our goal we need to implement a matching algorithm, and enrich the current functionality of agents with the ability to mediate WS invocations and measure QoS factors. A flexible ranking policy and learning algorithms for agents are also planned as part of our future work.

In [15] pairs of web services are compared by matching their data types, messages and operations. With respect to these approaches we are going to build indexes for provided operations, use semantic matching to eliminate the mismatched operations and apply relaxed structure matching for the filtered subset. We are planning to elaborate the results by trying alternative IR models (LSI, Probabilistic Models, etc.) on WSDL specifications. Another interesting point is that the *matching agent* can learn from experience. It can match each new query across services and across previous requests to them as well.

Our architecture is similar to [9] in a sense that agents control overall communication process between clients and each web service. Several problems are stated in the paper that are also relevant for our task. Among them is the risk tolerance of the user: the invocation of a new service matching the client's goal could be regarded as higher risk than a more mature one. Another issue is that several complementary web services can be found to satisfy the goal. Our current work does not aim at tackling the problem of WS composition but potentially some state-of-the-art approaches can be exploited in order to increase system recall by considering composite services. An application of recommendation system for web services spawns a good number of interesting issues, typical for recommendation systems and novel as well. One of them is a new-system cold-start problem, when there are no user ratings of web services (monitored service invocations). In this case, the initial recommendation can be constructed using Google's rating of the provider. For a new-service cold-start problem agents' ratings of the services by the same provider can be used.

References

1. Birukov, A., Blanzieri, E., Giorgini, P.: "Implicit: An agent-based recommendation system for web search", Proceedings of the 4th International Conference on Autonomous Agents and Multi-Agent Systems, 2005, pp. 618–624.
2. Blake, M. et al.: "The EEE-05 Challenge: A New Web Service Discovery and Composition Competition", IEEE Computer Society, 2005, pp. 780–781.
3. Dong, X.L. et al.: "Similarity search for web services", Proceedings of VLDB, 2004.
4. Garofalakis, J., Panagis, Y., Sakkopoulos, E., Tsakalidis, A.: "Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?", International Workshop on Web Engineering, 2004.
5. Hausmann, J.H., Heckel, R., Lohmann, M.: "Model-based Discovery of Web Services", Proceedings of the IEEE International Conference on Web Services, 2004.
6. Huang, Sh., Wang, X., Zhou, A.: "Efficient Web Service Composition Based on Syntactical Matching", IEEE Computer Society, 2005, pp. 782–783.
7. Jeckle, M., Zengler, B.: "Active UDDI - an Extension to UDDI for Dynamic and Fault-Tolerant Service Invocation", Web and Database-Related Workshops on Web, Web-Services and Database Systems, LNCS, 2003, pp. 91–99.
8. Kalepu, S., Krishnaswamy, S., Loke, S.-W.: "Reputation = f(User Ranking, Compliance, Verity)", Proceedings of the IEEE International Conference on Web Services, 2004.
9. Maximilien, E.M., Singh, M.P.: "Conceptual Model of Web Service Reputation", SIGMOD Record, Vol. 31, No. 4, Dec. 2002, pp. 36-41.
10. Medjahed, B., Bouguettaya, A.: "A Multilevel Composability Model for Semantic Web Services", IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 7, 2005, pp. 954–968.
11. Oh, S.-Ch., Lee, D., Kumara, S.: "Flexible Web Services Discovery and Composition using SATPlan and A* Algorithms", In Modeling Decisions for Artificial Intelligence Conference, 2005.
12. Ran, Sh.: "A Model for Web Services Discovery With QoS", ACM SIGecom Exchanges, Vol. 4, No. 1, 2003, pp. 1–10.
13. Rahm, E., Bernstein, P.: "A Survey of Approaches to Automatic Schema Matching", VLDB Journal, vol. 10, No. 4, 2001, pp. 334–350.
14. Sajjanhar, A., Hou, J., Zhang, Y.: "Algorithm for Web Services Matching", In Proceedings APWeb 2004, LNCS 3007, Springer Verlag, 2004, pp. 665–670.
15. Stroulia, E., Wang, Y.: "Structural and semantic matching for accessing web-service similarity", International Journal of Cooperative Information Systems, Vol. 14, No. 4, 2005, pp. 407–437.
16. The OWL Services Coalition. OWL-S specification version 1.0, November 2003. http://www.daml.org/services/.
17. Vu, Le-Hung. et al.: "Towards P2P-based semantic web service discovery with QoS supports", Workshop on Business Processes and Services, 2005.