Examining Usage Protocols for Service Discovery

Rimon Mikhaiel and Eleni Stroulia

Computing Science Department, University of Alberta, Edmonton AB, T6G 2H1, Canada
`{rimon, stroulia}@cs.ualberta.ca`

Abstract. To date, research on web-service discovery has followed the tradition of signature matching based on the interface description captured in WSDL. WSDL specifications, however, can be information poor, with basic data types, and unintuitive identifiers for data, messages and operations. The nature of the usage of the WSDL component in the context of a BPEL composition can be an extremely useful source of information in the context of service discovery. In this paper, we discuss our method for service discovery based on both interface and usage matching, exploiting the information captured in the WSDL and BPEL specifications. Our approach views both WSDL and BPEL as hierarchical structures and uses tree alignment to compare them. We illustrate our method with an example scenario.

Keywords: service discovery, WSDL matching, BPEL matching, process matching.

1 Motivation and Background

Service discovery is an essential task in the process of developing service-oriented applications. In a typical service-discovery scenario, the service requester has specific expectations about the candidate service. In general, there are three types of desiderata for a service: it has (a) to be *capable* of performing a certain task, i.e., maintain a shopping cart, (b) to expose a particular *interface*, i.e., provide view, add-product and remove-product, and (c) to *behave* in a certain manner, i.e., ignore any request for product removals if no product additions have been performed yet. Such expectations motivate and guide the developers' searches through web-services repositories, as they try to discover and select the service that best matches their needs.

When not all these dimensions are considered in concert, the precision of the discovery process tends to be limited. For example, applying only a capability matching method between a service and a request does not guarantee that they are interoperable. Additionally, even when the interface of a service matches a request's interface, it is not necessarily clear how exactly its data and operations match the requestor's specification, especially when the parameter types are simple. Furthermore, in some cases, interface matching may get confused because the operation signatures are not significantly distinct.

In this paper, we propose an integrated service matching approach that is based on both interface and behavioral matching. For example, a WSDL [5] (interface) specification of a complex web service may be associated with a BPEL [1] (behavioral) specification that provides semantic information about the usage protocol of the provided operations and the overall usage patterns of the data types. In the proposed integrated matching approach, WSDL components are matched not only in a syntactic manner, but also in a way that complies with their BPEL's usage protocols.

Our method involves two steps: First, relevant information from the WSDL and BPEL descriptions of the desired and candidate services is parsed and represented in a special-purpose tree representation. Next, the tree representing the desired service is compared against the trees representing each of the candidate services to select the most similar one and to precisely identify how the data and operations offered by the candidate map to the requestor’s needs in a way that respects their usage protocols.

The rest of this paper is organized as follows. Section 2 reviews related research on service discovery. Section 3 discusses a case study illustrating the insufficiency of WSDL for effective service discovery and the relevance of BPEL and usage-protocol information to the task. Section 4 presents our service-matching method based on tree alignment. Section 5 shows how the presented method resolves the issues raised in Section 3. Finally, Section 6 concludes with the essential ideas of our work.

2 Related Research

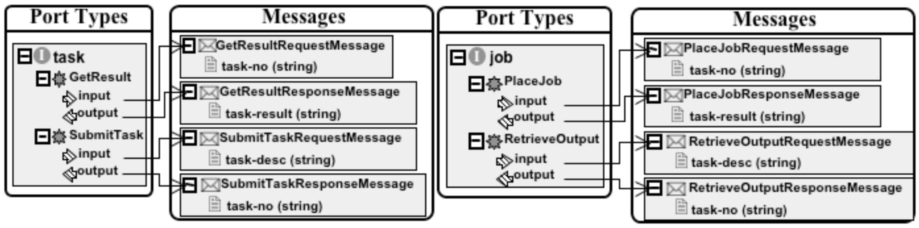
According to the literature, there are three categories of service discovery approaches. *Capability matching* examines whether a published service can satisfy the requested needs. UDDI matching is an example of such an approach. However, such capability matching is not enough to establish a real interaction; it does not specify the ontology of the involved data types nor the operations applicable for this service. *Interface matching* is based on signature matching between the published operations and the requested ones [8], [9], [10], [11]. However, this approach suffers from two drawbacks: first, interface matching does not guarantee a successful interaction because interfaces do not usually specify the usage conditions of the operations involved. Hence, an improper usage of the published operations may lead to an interaction failure. Second, interface matching may easily get confused when the services specifications are not distinctive; because it relies on documentation and operation signatures, when the data types are simple and there is not much documentation there is not enough information on the basis of which to assess (dis)similarity. Finally, *behavior matching* examines whether both the requested and provided service behave according to a similar interaction protocol. This approach is only concerned about either control-structure matching (like Petri nets [4], and π-calculus [7]), or message-flow matching (like WSDL [2] [3], and BPEL finite state machines [12]).

Our method integrates ideas from both interface and behavior matching. In this paper, we demonstrate that such integration is essential not only to enhance the matching quality but also to resolve ambiguities that may occur when each approach is applied separately. Additionally, this method requires a capability-matching step to be performed to maintain a set of candidate services, out of which this method selects the best inter-operable service.

3 The Service-Discovery Problem in an Example

In general, when looking for a service, a developer has in mind both the signatures of the operations desired and some behavioral scenarios, in which the candidate service is expected to participate. Discovery based simply on WSDL matching is concerned with the matching of the operations desired and provided. However, given a candidate service, there usually exist multiple likely mappings between the desired operations and the ones provided by the candidate service. Unambiguously selecting one of these mappings is often impossible, when neither the syntactic types nor the identifiers and documentation are distinctive.

For example, consider the case illustrated in Fig. 1, when the provided service performs an asynchronous task. The consumer is expected to first submit the task, to obtain a receipt, and then to return (after a while) asking for the results associated to that receipt; if, at that time, the task is completed the results are returned to the consumer otherwise an exception is returned. Fig. 1 illustrates the WSDL specifications of a provided and a requested service, both involving such a task-submission service. However, it is not clear how the two published operations can be mapped to the consumer's expected ones; both accept a string parameter and return a string parameter. Applying WSDL matching only cannot unambiguously select a mapping of their operations.



(a) A published WSDL description **(b) A requested WSDL description**

Fig. 1. A visual representation of two WSDL descriptions showing each operation, connected to its associated *input* and *output* messages. Each message description includes its associated *parameter name* and *typ*e.

4 The Method

The specific research question that our method is designed to address is: "how should service components be matched in a way that respects both their interface description and usage protocols?". Its underlying intuition is that both these aspects of a service description provide information that can be useful when considering a candidate service as a potential match for a requested one. More specifically, ambiguities in operations' mapping can be resolved by examining the behaviors in which these operations are involved. Consequently, our method is based on a two-level comparison of the corresponding service aspects, where a specialized tree-alignment approach is adopted for each level. First, the BPEL behavioral specification of the provided service is compared against the behaviors expected of the requested service. Second, the WSDL operations of the desired and the candidate services are compared, using a cost function that is based on the results of the first step.

4.1 Web-Service Matching as Tree-Edit Distance

Both BPEL and WSDL descriptions are expressed in terms of XML documents that can be represented as ordered labeled trees, where the attributes of the XML elements are represented as their children.

For each of the two steps in the service aspects' comparison, we adopted the SPRC [6] tree-alignment algorithm, developed in the context of our work in RNA structure alignment. This algorithm is based on the Zhang-Shasha's tree-edit distance [13] algorithm, which calculates the minimum edit distance between two trees given a cost function for different edit operations (e.g. change, deletion, and insertion). In addition, SPRC uses an affine–cost policy and reports all the alignment solutions that are associated with the calculated tree edit distance. Additionally, in a post-processing step, SPRC applies a simplicity-based filter to discard the more unlikely solutions from the solution set produced by the SPRC tree-alignment phase.

4.2 A Usage-Aware Cost Function

The usage similarity between two components is the similarity of how they are referenced by higher-order descriptions. For example, the result of BPEL matching should be considered while performing a WSDL operation-matching step. If the BPEL references to two operations are similar, then the cost of mapping them in the WSDL-matching step is reduced, proportionally to their *degree of mapping* (DOM). The degree of mapping between two operations is defined as double the number of mappings between the references two the two operations divided by the total sum of their references. Incorporating the DOM ratio of a BPEL-matching step into the WSDL-matching step effectively incorporates the usage similarity of the components references with their signature matching.

5 The Example Revisited

Fig. 2(a) shows the BPEL description of the published service where the service provider describes how that service is supposed to be used, while Fig. 2(c) shows the expected scenario from the consumer point of view. Additionally, Fig. 2(b) and Fig. 2(d) show the tree representation of the published and requested services, respectively.

The result of aligning the BPEL descriptions of Fig. 2(a) and Fig. 2(c) is shown in Fig. 2(b) and Fig. 2(d): the receive action named "receive task" associated with the operation named `SubmitTask` is mapped to the receive action named "place a job" associated with the operation named `PlaceJob`. Additionally, the send action named "Send task-no" is mapped to "Reply with Job ID". Hence, the two (out of two) references to the operation `SubmitTask` are mapped to two (out of two) references to the operation `PlaceJob`, which leads to a DOM ratio of (2*2)/(2+2)=1. Similarly, Fig. 2(b) and Fig. 2(d) show that 3 (out of 4) references to the operation `GetResult` are mapped to the three (out of three) references to the operation `RetrieveOutput` and vice versa, which also results in DOM ration (2*3)/(4+3)=0.86. Therefore, the cost function for the subsequent alignment step is advised to reduce the mapping cost for both (`SubmitTask`, `PlaceJob`) and (`GetResult`, `RetrieveOutput`) by 100% and 86%, respectively. Thus, there is no longer any ambiguity for how the operations and messages of the two WSDL specifications should be mapped.

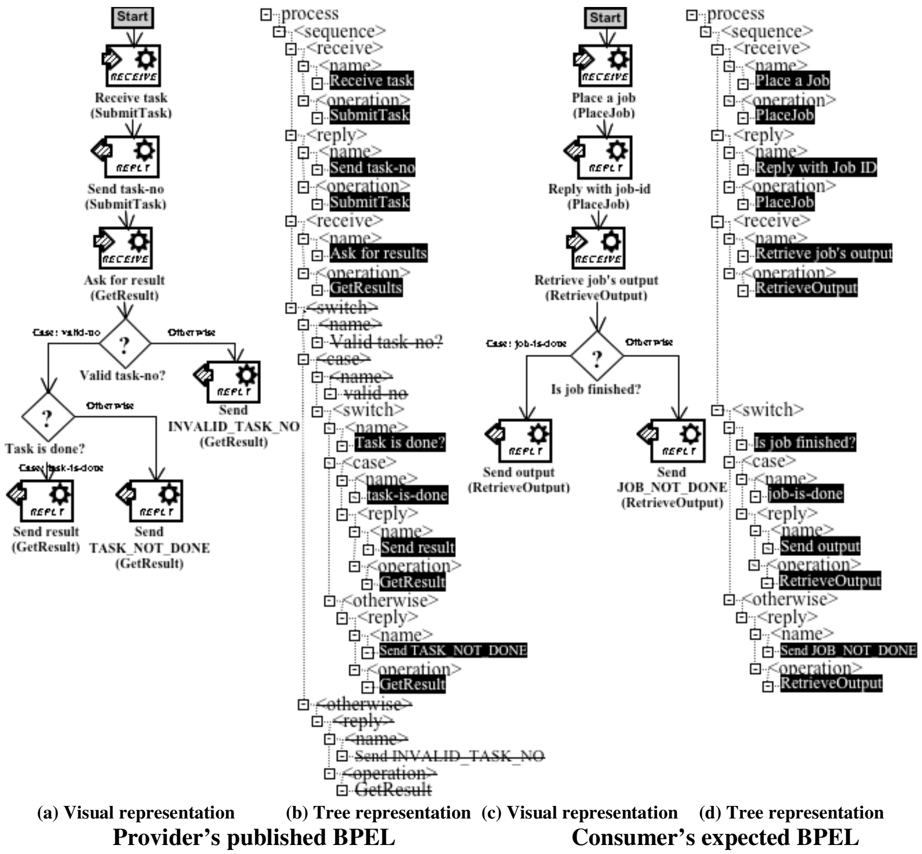


Fig. 2. BPEL descriptions of the published and requested services. Figures (a) and (c) are visual representations showing two types of actions *receive* and *send* actions; each of them is accompanied by the associated operation name. Figures (b) and (d) are annotated tree representations, e.g. a strike labels refer to deleted nodes, while highlighted labels refer to changed nodes.

It is interesting to note that the BPEL description of the published service imposes the pre-condition that a call to `SubmitTask` is to be invoked before a call to `GetResult`. Similarly, according to the requester BPEL description, the consumer expects to invoke an operation that has a semantic like `PlaceJob` before invoking an operation that has the semantic of `RetrieveOutput`. This is exactly the information that our method is designed to respect.

6 Conclusions

In this paper, we discussed our approach for resolving ambiguities in the mapping of published service elements to those of the requested service, by examining the usage of these elements in the context of BPEL-specified behavioral specifications of the service in action. The basic intuition underlying our work is to match operations usage references in order to get a measure of how a certain published operation is used similarly to a requested one. Our service discovery method is based on tree-alignment of the two corresponding specifications (BPEL and WSDL), with results from the BPEL matching step feeding into the WSDL matching step. It is designed to match the semantics implicitly embedded in the BPEL description in order to enhance the quality of the WSDL signature matching, and to resolve any possible ambiguities. Although BPEL specifications of services may not yet be widely used, their potential usefulness, as outlined in this paper, should motivate service providers to develop BPEL descriptions for their services, in order to improve the potential quality of automated discovery methods.

Acknowledgements

This research was supported by ICORE and the Alberta IBM CAS.

References

[1] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., and Weerawarana, S. “Business Process Execution Language for Web Services.” version 1.1, 2003, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[2] Blanchet, W., Elio, R., Stroulia, E. “Conversation Errors in Web Service Coordination: Run-time Detection and Repair”. Proc. International Conference on Web Intelligence (WI), 2005.

[3] Blanchet, W., Elio, R., Stroulia, E. “Supporting Adaptive Web-Service Orchestration with an Agent Conversation Framework”. Proceedings of the third IEEE International Conference on Web Services (ICWS), 2005.

[4] Brockmans, S., Ehrig, M., Koschmider, A., Oberweis, A., and Studer, R., “Semantic Alignment of Business Processes”. In 8th International Conference on Enterprise Information Systems, 2006.

[5] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., “The web services description language WSDL.” http://www.w3.org/TR/wsdl

[6] Mikhaiel, R., Lin, G., Stroulia, E., “Simplicity in RNA Secondary Structure Alignment: Towards biologically plausible alignments”. In Post Proceedings of the IEEE 6th Symposium on Bioinformatics & Bioengineering (BIBE 2006), October 16 - 18, 2006

[7] Milner, R., Parrow, J., and Walker, D., “A Calculus of Mobile Processes, Part I+II”. Journal of Information and Computation, September 1992, 1--87.

[8] Payne, T.R., Paolucci, M., and Sycara, K., “Advertising and Matching DAML-S Service Descriptions”. Semantic Web Working Symposium (SWWS), 2001.

[9] Syeda, T., Shah, G., Akkiraju, R., Ivan, A., and Goodwin, R., “Searching Service Repositories by Combining Semantic and Ontological Matching”. ICWS, 2005, 13--20.

[10] Stroulia, E., and Wang, Y., “Structural and Semantic Matching for Assessing Web-Service Similarity”, Int. Journal of Cooperative Information Systems, 14(4):407-437, June 2005.

[11] Wang, Y., Stroulia, E. “Flexible Interface Matching for Web-Service Discovery”. 4th International Conference on Web Information Systems Engineering, 2003, pp. 147-156, IEEE Press.

[12] Wombacher, A., Fankhauser, P., and Neuhold, E. “Transforming BPEL into Annotated Deterministic Finite State Automata for Service Discovery.” ICWS, 2004, 316--323.

[13] Zhang, K., Stgatman, R., and Shasha, D. “Simple fast algorithm for the editing distance between trees and related problems.” SIAM Journal on Computing, 18(6), 1989, 1245--1262.