Semantic Web Service Offer Discovery

Jacek Kopecký

Digital Enterprise Research Institute (DERI)
Innsbruck, Austria
jacek.kopecky@deri.at

Abstract. Semantic Web Services are a research effort to automate the usage of Web services, a necessary component for the Semantic Web. For a number of reasons, static detailed and complete semantic service description is not feasible, the client software cannot select the best service offer for a given user goal only by using the static service descriptions. Therefore the client interacts automatically with the discovered Web services to discover the offers, in other words to find the information necessary to select the best offer that will fulfill the user's goal. Semantic Web Service offer discovery is the focus of our work.

1 Introduction

The Semantic Web is not only an extension of the current Web with more semantic descriptions of data; it also needs to integrate services that can be used automatically by the computer on behalf of its user. A major technology for publishing services on the Web is the so-called Web services. Based on WWW standards HTTP and XML, Web services are gaining significant adoption in areas of application integration, wide-scale distributed computing, and business-to-business cooperation. Still, many tasks commonly performed in service-oriented systems remain manual (performed by a human operator).

In order to make Web services part of the Semantic Web, the research area of Semantic Web Services (SWS) aims to increase the level of automation of some of these tasks, e.g. discovering the available services and composing them to provide more complex functionalities. SWS automation is supported by machine-processible *semantic* Web service descriptions. Current state of the art in non-semantic service description is WSDL[1], which can describe the messages accepted and produced by a Web service, and the simple message exchanges (called operations) and all the necessary networking details. In effect, WSDL specifies a limited syntactical contract that the service adheres to. We call it *syntactical* because it only constrains the lexical form of the messages. The meaning of the operations and messages is usually expressed in textual documentation or sometimes only implied in the names of the message elements. Semantic descriptions capture the important aspects of this meaning, generally in a formal language based on logics.

SWS descriptions are processed by a semantic execution environment (SEE, for instance WSMX [4]). A user can submit a concrete *goal* to the SEE, which then finds and

[1] Web Service Description Language, `http://w3.org/TR/wsdl20`

uses the appropriate Web services to accomplish the goal. SWS research focuses mainly on how the SEE "finds the appropriate Web service(s)", so to speak, as illustrated in Figure 1 with the first three SEE tasks. More concretely, SWS research focuses chiefly on the description languages and the discovery mechanisms that use SWS descriptions to match them against user goals.

In the figure, meant to be illustrative of the situation, rather than a real-world scenario, the user decides to lead a healthier life and wants to buy 2kg of fruit. The SEE will first discover any services that sell fruit (discarding the service that sells potatoes), then it will filter depending on the user's constraints and requirements (the user doesn't like peeling oranges), ranks the resulting services according to the user's preferences (the user is a student and so prefers the cheaper options) and selects the one service that is invoked in the end. At any stage in the process, the user can be allowed to confirm the results.

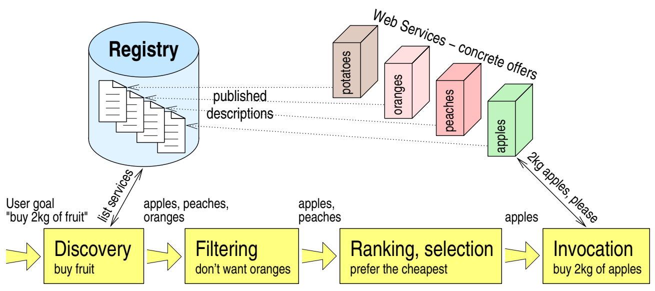

Fig. 1. Semantic Web Services automation tasks

We've chosen the simple fruit-shops example here to illustrate the situation in easily accessible terms, however, the general situation applies to any kinds of service: the existing services will publish their descriptions in a registry; then the SEE will *discover* the applicable services, *filter* and *rank* them according to any constraints and preferences, and *invoke* the selected service to achieve the user's goal.

The steps described above rely solely on the published Web service descriptions to find the best service that matches the user's goal. Alas, outside limited scenarios, it is not feasible to put all the relevant information in the service description, due to reasons detailed later in this paper. For instance, a grocery shop service would not list all the kinds of fruit they currently sell along with their up-to-date prices; instead, such a service would be described as "selling groceries". This limits the scope of discovery based on static descriptions and introduces the need for an additional step, where the SEE will contact the discovered Web services (or their providers) to find out more about the service's concrete offerings.

This additional step is called **offer discovery** (as opposed to service discovery). The objective of this step is to establish whether the service can fulfill the user's concrete goal and under what conditions. In our fruit shopping example, the SEE checks whether a grocery shop service carries any fruits, what sorts of fruits are available and at what prices, as shown in Figure 2. SWS offer discovery is the focus of our work.

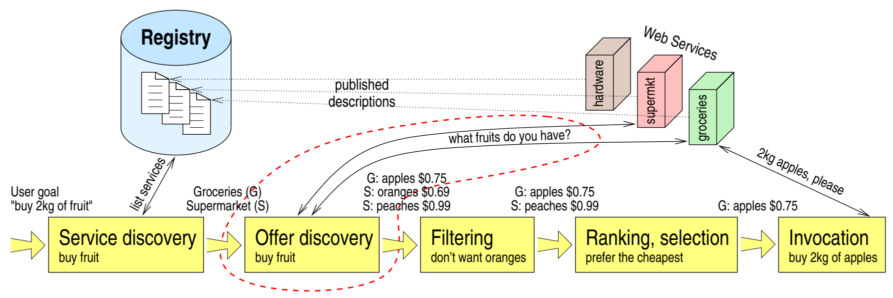

Fig. 2. Semantic Web Service offer discovery in SEE tasks

This paper is structured as follows: in Section 2 we define SWS offer discovery, relate it to other SEE tasks and provide detailed motivation for why this step is necessary. Section 3 presents related work, both within Web services and in earlier research areas. In Section 4, we sketch the envisioned solution. Section 5 describes our expected research and evaluation methodology, and Section 6 contains concluding remarks.

2 Semantic Web Service Offer Discovery

The best way to define offer discovery is by describing the problems that it aims to solve. First, let us review the distinguishable functions of a semantic execution environment (SEE, cf. [7]). The following steps are traditionally executed after a user submits their goal "buy 2kg of fruit", as shown in Figure 1.

1. **Discovery**[2] — using published descriptions, find all the available Web services that may sell fruits (the services may be more generic, like a supermarket with all kinds of products, or more specific, like an owner of a cherry-tree orchard, who naturally only offers cherries).
2. **Filtering** — filter out services that do not fit the user's constraints (for instance a service that sells oranges, because the user does not like them).
3. **Ranking, selection** — rank the remaining offers based on the user's preferences, for instance by price. The best-ranked service may be automatically selected, or the ranking may be presented to the user.
4. **Invocation** — use the selected service to achieve the goal (in our case, purchase the fruit).

On top of these steps, SWS literature often mentions *mediation* and *composition*. Mediation occurs everywhere where the SEE encounters heterogeneities, for instance different units of scalar values. Composition puts together multiple services if a single one cannot accomplish the user's goal. Neither of these tasks is particularly relevant to offer discovery, even though they may interact.

[2] Sometimes, the term *discovery* is used to mean all the steps leading from a user's goal to a service that can fulfill it, i.e. everything but invocation. We choose a narrower definition of discovery which only does matchmaking on the available service descriptons.

The task sequence above is fully adequate when the service descriptions carry all the data relevant for the goal of the user. In a grid environment, a user might need processing services and storage services, and the descriptions will contain such classifications. In our fruit-buying scenario, the services need to advertise in their descriptions that they sell fruit (and what kinds of fruit) and at what price (for ranking).

In restricted environments and in simple cases, it is not a problem to publish all the relevant information about a Web service in its semantic description. However, complete "semantic-heavy" description becomes unwieldy for real-world services on the open Web:

- for a larger online store, the full product catalogue would make a Web service description impossible or impractical to process, considering current reasoning performance;
- similarly to the above case, updating a catalogue in a service registry upon every inventory or pricing change would lead to heavy bandwidth use in the registries;
- a full description of service offers could even reveal trade secrets, for instance a bank service description would have to detail all loan approval procedures.

While reasoning performance may improve, and registry updates can be optimized, trade secrets will not go away. For banks, the loan approval process with all its considerations is part of what makes some banks successful and others bankrupt.

In our experience, the complexity of semantic-heavy service descriptions is certainly a practical barrier to SWS adoption within the Web services industry. The completeness requirement of the semantic-heavy approach, including possibly sensitive details about all the offers, is at least a perceived barrier as well. In other words, service discovery based solely on complete static semantic descriptions is of limited usefulness. On the other hand, less detailed "semantic-light" descriptions (for instance, Amazon would be described as selling books, movie DVDs, music CDs and DVDs etc.) limit the SWS automation to simple matchmaking and ranking.

Therefore, we split the task of finding the most appropriate offer from all the available Web services into static *service discovery* followed by dynamic *offer discovery*. The static service discovery uses coarse-grained semantic service descriptions to find services that potentially match the user's goal, and the dynamic offer discovery uses the semantic description of the service interface to automatically find appropriate offers. Offer discovery can be seen as information retrieval or as negotiation, as discussed in Section 3. With offer discovery, the set of steps can be rephrased as follows:

1. Service discovery — find all the available Web services that may be able to fulfill the user's goal (i.e. discard those which, based on their description, cannot fulfill the user's goal).
2. Offer discovery — by interaction with the discovered services, find all their offers relevant for the goal.
3. Filtering — filter out offers that do not fit the user's constraints.
4. Ranking, selection — rank the remaining offers based on the user's preferences, and select one to be invoked.
5. Invocation — use the selected service.

Semantic offer discovery aims to be able to communicate with any Web service and find information about offers relevant to the user's goal. For communicating with the Web services, the offer discovery engine needs a description of the service interface (what operations it contains that can be used to gather offer information) and a description of the exchanged data, to understand the offers and be able to compare it with the goal. In other words, offer discovery needs different semantic description than that for service discovery; service discovery needs to know what the service offers, whereas offer discovery needs to know how to talk to the service to get the information.

Our main hypothesis is that *the semantic description necessary for automated offer discovery is significantly easier to create and manage (and more acceptable) than the complete semantic description of all the offers*. A further hypothesis is that *the process of offer discovery is more efficient or on par with the processes of managing the complete descripton and reasoning with it in service discovery*. Assuming these hypotheses hold, offer discovery would complement service discovery in a two-stage process that is more efficient than service discovery with complete semantic descriptions. In Section 5, we show how we expect to evaluate these hypotheses.

3 Related Work

Semantic Web service offer discovery, as defined in the preceding section, is related to earlier research in automated negotiation, and query processing/information gathering.

The term *negotiation* has been used for different purposes in a variety of computer science fields, e.g. electronic commerce, grid computing, distributed artificial intelligence and multi-agent systems. In electronic commerce, Beam and Segev [2] define negotiation as "the process by which two or more parties multilaterally bargain resources for mutual intended gain". There are several different types of negotiations in e-commerce: auctions (multiple buyers bid for price), double auctions (both buyers and sellers bid for price, e.g. stock exchanges), one-to-one bargaining, and even catalogue provision (price fixed by seller). Offer discovery is similar to catalogue provision (offer discovery accesses the relevant parts of the offer catalogue), but it could be extended in the direction of bargaining as well.

Research in query answering and information retrieval has dealt, among others, with using multiple information sources to gather the requested (or relevant) information (cf. [5]), based on a user query. In Semantic Web services, a user goal can be seen as a form of query, and the discovered services (or their individual operations) as information sources. The particular problem in SWS offer discovery is the description of the services and their operations so that information retrieval techniques would be applicable.

Apart from related work described above, we know of only one published attempt that involves dynamic offer discovery in Semantic Web Services: Zaremba *et al.* [8,9] talk about a so-called "contracting interface" with a described choreography. In their case, the SEE client follows the predefined choreography to find out the concrete price offered by a discovered Web service. The contracting interface can be likened to a prescribed negotation protocol.

Offer discovery is not a problem specific to SWS. However, earlier efforts on similar automation (e.g. in multi-agent systems) have generally presumed a controlled environment with a predefined set of interaction protocols for various tasks; for instance, a marketplace would dictate a bargaining and auctioning protocol, and a distributed query answering system would presume a single common query language and protocol. Such approaches are applicable to Web services, however, such bargaining/auctioning protocols or common query languages would need to be standardized and adopted both by most service providers. Our SWS offer discovery mechanism, together with any necessary semantic annotations mechanisms, is a novel approach because it aims to be generic, independent of the domain of the service offers; potentially, it could serve as one input to the mentioned standardization process.

4 Envisioned Solution

Our work on Semantic Web Service offer discovery is in a very early stage. In this section, we show the components necessary for SWS offer discovery, and their envisioned implementations. SWS offer discovery needs the following components:

1. semantic annotations of Web services, their operations and messages;
2. evaluation of what operations can be invoked with the available data, and what operations are likely to return *relevant* offer data;
3. invocation of the selected operations with the appropriate goal data, translated into the appropriate XML messages.

The first component deals with the semantic description of the Web service operations and their inputs and outputs, also known as *SWS grounding* (cf. [6]). As part of activities related to SWS grounding, the author has been involved with W3C standardization of the WSDL 2.0 Recommendation[3], and chaired the W3C Working Group that created the Semantic Annotations for WSDL and XML Schema (SAWSDL) specification[4] While these efforts have little research value, the familiarity with the Web and Web Services specifications should help make the semantic annotation mechanisms more practical.

One major piece of useful existing annotation is WSDL 2.0 *operation safety*[5]. Safety annotation allows a client to distinguish operations that are *safe* to be used automatically and opportunistically. For instance on the Web, every information retrieval (HTTP GET) is safe. Operation safety, being part of the WSDL 2.0 standard, is a semantic annotation that should be quite acceptable to the industry.

The second component, therefore, can select all *safe* operations and operations otherwise annotated as useful for information retrieval, if necessary. For instance, such operations could allow browsing the catalogue of an online store or finding out the price and further details for a given product. When we have selected the suitable operations, we can use information gathering and planning techniques to choose the operations that

[3] http://w3.org/TR/wsdl20

[4] http://w3.org/TR/sawsdl , soon to become a W3C Recommendation.

[5] See the definition of *safe interactions* in [1].

can return relevant information about the offers pertinent to the user's goal. We do not have a firm definition of *relevant* at the moment.

The third component actually invokes the chosen operations. Its role is to ground the goal data (presumably in a Semantic Web language, e.g. RDF) to the XML messages expected by the operations, and interpret the returned XML messages as semantic data about offers.

5 Research and Evaluation Methodology

The planned research can be split into the following sequence of high-level steps:

– Investigate information retrieval and distributed query answering methods, especially with respect to how a SWS goal can be transformed into an information retrieval query;
– find out what semantic descriptions are necessary to enable applying the above method(s) to SWS, especially whether Web operation safety is applicable;
– implement a prototype offer discovery engine, evaluate against static semantic-heavy service discovery and potential other competing approaches.

The expected contribution of our work is an efficient approach to automatic offer discovery that complements service discovery based on static descriptions. The efficiency of the approach is evaluated in two dimensions: volume and complexity of the necessary semantic description, and the performance and scalability of the discovery process, as described below. The offer discovery engine will form a part of a semantic execution environment (SEE). The use cases for a SEE include the common scenarios of electronic shopping and travel scheduling[6], but also existing real-world applications such as e-Government Emergency Planning, as described in [3].

The major benefit of the presence of offer discovery in a SEE is that the semantic descriptions of Web services need not be complete and detailed (e.g. the whole catalogue of an online store). This guides the first evaluation criterion: *the semantic description necessary to enable automated offer discovery must be significantly simpler than a complete and detailed static semantic description of the service.* We will do concrete comparisons of reasoning performance, and user tests on the relative complexity of creating and managing the full static description vs. the descriptions necessary for automatic offer discovery.

Apart from the complexity of the semantic descriptions, offer discovery requires interaction with the Web services, whereas static service discovery based on complete semantic-heavy descriptions requires that the service provider updates the description on every change. Therefore, the second evaluation criterion is: *the reasoning and networked interaction during offer discovery should have better performance and scalability than the reasoning and network interactions for service description updates*. Performance can be compared on specific test cases, and scalability needs to evaluate how the approaches can deal with many services and many service offers. Further, the

[6] See http://sws-challenge.org/

comparison combines reasoning tasks with network interactions; therefore we need to evaluate different settings of reasoning power vs. networking setup.

Finally, we will also compare our approach against the "contracting interface" approach from [9]. We can formulate our third evaluation criterion: *the semantic description necessary to enable automated offer discovery should be simpler and more flexible than the semantic description of the contracting interface(s).* Again we talk about simplicity of description, which we can test with users, but we are adding "flexibility" here, because at this time we suspect that a prescribed contracting choreography may be limiting, as it expects a particular level of detail of information described in the user goal. For instance, if a user goal is to buy 2kg of fruit, a choreography for checking the price of concrete products may fail because the user hasn't specified a concrete fruit. Therefore in terms of flexibility, we can compare the correctness and completeness (a.k.a. precision and recall) of offer discovery with a "contracting interface" and our approach that makes use of any available information gathering operations.

6 Conclusions and Future Work

The Semantic Web is an effort towards further automation in the World Wide Web. In addition to semantic descriptions of the available data, the Semantic Web needs to incorporate services. In order for the computer to use the services automatically on behalf of its user, the software needs to be able to discover and invoke the available services. Semantic Web Services are an effort to provide the necessary semantic descriptions of Web services so that such automation can be achieved.

Since Web service discovery cannot always be based on complete and detailed semantic description, it needs to be complemented with automatic offer discovery. We intend to research an approach to SWS offer discovery that will significantly simplify the needed semantic descriptions and thus help ease the adoption of SWS technologies in the industry.

References

1. Architecture of the World Wide Web. Recommendation, W3C (December 2004), Available at, `http://www.w3.org/TR/webarch/`
2. Beam, C., Segev, A.: Automated negotiations: A survey of the state of the art. Wirtschaftsinformatik 39(3), 263–268 (1997)
3. Davies, R.: Summative report on potential applications of SWS in eGovernment, Deliverable D9.16, project DIP (FP6 - 507483) (2006), available at `http://dip.semanticweb.org/documents/D916SummativeRpt_potentialAppssSWS_EGov_final.pdf`
4. Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C.: WSMX – A Semantic Service-Oriented Architecture. International Conference on Web Services (ICWS 2005) (July 2005)
5. Knoblock, C.A.: Planning, executing, sensing, and replanning for information gathering. In: Proc. of the 14th Int'l Joint Conference on Artificial Intelligence, pp. 1686–1693 (1995)
6. Kopecký, J., Roman, D., Moran, M., Fensel, D.: Semantic Web Services Grounding. In: Proc. of the Int'l Conference on Internet and Web Applications and Services (ICIW 2006) (February 2006)

7. Vitvar, T., Mocan, A., Kerrigan, M., Zaremba, M., Zaremba, M., Moran, M., Cimpian, E., Haselwanter, T., Fensel, D.: Semantically-enabled service oriented architecture: concepts, technology and application. Service Oriented Computing and Applications 2(2) (2007)
8. Vitvar, T., Zaremba, M., Moran, M.: Dynamic service discovery through meta-interactions with service providers. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC. LNCS, vol. 4519, pp. 84–98. Springer, Heidelberg (2007)
9. Zaremba, M., Vitvar, T., Moran, M., Hasselwanter, T.: WSMX Discovery for SWS Challenge. SWS Challenge Workshop, Athens, Georgia, USA (November 2006)