OPOSSUM: Bridging the Gap between Web Services and the Semantic Web

Eran Toch*, Iris Reinhartz-Berger†, Avigdor Gal*, and Dov Dori*
*Faculty of Industrial Engineering and Management
Technion - Israel Institute of Technology
Technion City, Haifa, 32000 Israel
erant@tx.technion.ac.il, {avigal, dori}@ie.technion.ac.il
†Department of Information Systems
University of Haifa
Carmel Mt., Haifa, 31905 Israel
iris@mis.hevra.haifa.ac.il

I. INTRODUCTION

Web services are distributed and loosely coupled software components, which are accessed through the World Wide Web. Web services form the basis for service-oriented architectures used by enterprises to develop and integrate large-scale information systems. The ongoing adoption of service-oriented architectures raises the need for efficient and precise search and retrieval of Web services. We propose to investigate aspects of Web service retrieval, facing a gap between user specifications, given in some form of a semantic description, and Web service definition, given in a standard interface description such as WSDL [3]. Bridging this gap is becoming more and more urgent as users need to find Web services among increasing numbers of Web services within organizations and on the Web.

The main challenge in service-retrieval is the lack of semantics in their interface description for precise search. Current retrieval solutions can be classified as either *linguistic-based* or *semantics-based*, each of them exhibit different limitations. Linguistic-based solutions are based on textual analysis of actual Web service descriptions, mainly in the form of WSDL documents, which defines the operations and message parameters of the service [3]. This approach is taken, for example, by the Woogle search engine [4]. Linguistic methods (mainly clustering techniques) are used in order to support similarity search of operations by grouping together names of operation's parameters into meaningful concepts. While linguistic approaches dramatically raise the precision and recall of WSDL-described Web services, they do not reach the level of certainty required to support a fully automated solution.

The semantic approach to service-retrieval is based on expanding the description of Web services with formal semantic models, such as OWL-S [1]. These models provide an unambiguous description of service properties by relating them to concepts belonging to *Web ontologies*. In our context, Web ontologies are predefined vocabularies that are accessible through the Web and formally and unambiguously define concepts (and relations between them) [2]. Several works, including [5] and [6], had proposed methods for matching queries and advertisements of semantic Web services. As concepts are well defined with structured ontologies, logic-based proof inference is used to match services. While these methods retrieve services with very high precision and certainty, they require the availability of full semantic models of the service query and advertisements in order to perform the matching. Apart from adding overhead to the task of developing a Web service, these methods require designers to agree on common concepts and meaning. While this is feasible in some application fields, it may not be feasible in others.

In order to address the current limitations of service-retrieval, we had developed $\mathcal{OPOSSUM}$ (Object-PrOcedure-SemanticS Unified Matching). It is a Web-based search engine that uses semantic methods for precise and efficient retrieval of Web services, based on their WSDL descriptions. $\mathcal{OPOSSUM}$ crawls the Web for WSDL descriptions, transforming them into ontological-based models of the Web services. It does that by automatically augmenting the service properties with existing concepts, which are collected from ontologies on the Semantic Web [2]. A novel algorithm [7], is used to match queries with the ontological-based model of Web services. The algorithm relies on probabilistic methods for structural analysis of the associated ontologies in order to improve the recall and precision of the search process. The search engine retrieves dynamically-created services, which are composed of several atomic services. Thus, even if a query is not answered by a single service, it might be answered by combining several services. Furthermore, users can specify behavioral requirements, such as execution order.

Demonstration highlights: To illustrate the features of $\mathcal{OPOSSUM}$ we will consider a set of business processes used by a corporation to handle its book-buying services. Using $\mathcal{OPOSSUM}$'s current index, that includes around **900** Web services and **15** ontologies, we will demonstrate the searching process. The characteristics of ontological-based searching will be presented, and discussed with respect to way ontologies are structured and written.

The rest of this document is constructed as follows. We first discuss the way users interact with the system, namely how users enter queries and search for services. We then give

an overview of the theoretical model behind $\mathcal{OPOSSUM}$. Finally, we describe the architecture of the search engine.

II. SEARCHING FOR WEB SERVICES

Users communicate with the $\mathcal{OPOSSUM}$ search engine using a simple query language, which is based on natural language. A set of query operators enable users to refer to specific properties of desired Web services. For instance, the operator *input*, will return all services that have an input that correlates with the query concept. Other operators include *output*, *description*, *name* etc. Furthermore, $\mathcal{OPOSSUM}$ enable administrators to create and configure new operators, allowing qualitative-based retrieval to be plugged into the search framework (such as *price*, *quality of service* etc). Entering words without specific operators invoke the default operator, which refers to the union of all possible properties.

The syntax of the query language combines propositional conjunction and disjunction connectives, as well as connectives that refer to the behavioral aspects of Web services. The expression *"input:film or input:video-clip and output:price"* returns services that receive either a film or a video-clip concept and return a price. Note that the *"and"* connective can be dropped, as it serves as the default connective. Also, grouping operators into parenthesis is done automatically. Several connectives specify behavioral patterns of services. For instance, the connective *"after"* specifies a composite service which contains the transitive closure of a sequenced set of services that begins with the second service and ends with the first one.

Using the user interface illustrated in Figure 1, users can submit queries and view the results. $\mathcal{OPOSSUM}$ presents a list of results, ranked according to their degree of affinity with the query. The results include the service name, a short description of the service, and a link to the actual WSDL document. $\mathcal{OPOSSUM}$ allows users to access the WSDL document, as well as executing and testing the service from inside the search engine framework.

III. ONTOLOGICAL MODEL FOR WEB SERVICES

Our approach relies on ontologies as a mean for increasing the certainty of matching service functionality. By using data integration and conceptual model techniques, WSDL documents are analyzed and transformed into a generic service model. Following that, the service properties are mapped to concepts that belong to ontologies in different domains. For instance, a corporate book-buying service will be mapped to ontologies that describe e-commerce, finance, and corporate concepts. A Web service is described as a set of operations $WS = \{OP_1, OP_2, ..., OP_n\}$. The operations, rather than the services themselves, are the elements which will be matched against the query. An operation is defined as a tuple $OP = \langle I, O, C, M \rangle$, where I is a set of input messages, and O is a set of output messages. C is a set of concepts, belonging to a set of ontologies. M is a mapping between input and output messages to concepts, which assigns a probability value $p \in [0, 1]$ that signifies the certainty of the mapping.
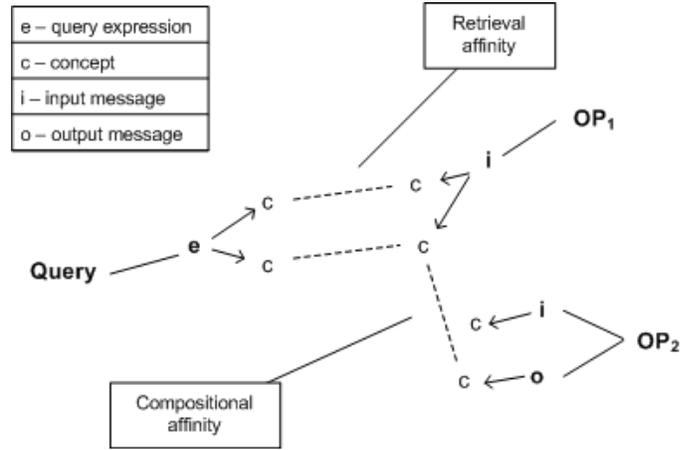

Fig. 2. Ontological Mapping of Web Services

The search query terms will be mapped to ontological concepts in a similar way. We will define a query as a tuple $Query = \langle E, C, M \rangle$, where E is a set of structured query expressions (which are not formally defined in this paper). C and M have the same meaning as in the formal model of Web services described above. Figure 2 depicts the conceptual models of operations and queries. The small-cap c denotes concepts, while the i and o symbolizes the operation's input and output messages, respectively.

The service-retrieval problem is transformed from a mapping problem between a query and a set of services to a mapping problem between a set of query concepts and a set of service concepts (annotated as the *retrieval affinity* in Figure 2). Rather than using only linguistic methods for measuring the affinity between concepts, the matching algorithm analyzes the ontology, using related concepts and the structure of the ontology itself. The results will be ranked according to the accumulated certainty (over the set of query expressions, concepts and messages) of the match between the query and the services.

The basic setting of our research assumes WSDL-based descriptions of Web services, without requiring further annotation of services. One of the consequences of this approach is the need to infer the underlying relations between individual Web services should be inferred in order to support retrieval of composed services. Similarly to the retrieval algorithm, $\mathcal{OPOSSUM}$ uses semantic affinity between concepts as a way to determine relations between Web service operations. Depicted in Figure 2 as the *compositional affinity*, semantic similarity between conceptual representation of the operation's input and output messages, will be uses as a basis for estimating the probability of two operations being composed together.

IV. $\mathcal{OPOSSUM}$'S ARCHITECTURE

In this section, we will provide a high level description of the architecture of the search engine. Figure 3 depicts the main components of $\mathcal{OPOSSUM}$, and the relations between them.

- **Crawler**: The Web crawling (downloading of web services WSDL and ontology documents) is done by several distributed *crawlers*, operating in an offline mode. The

Fig. 1. A screenshot of $\mathcal{OPOSSUM}$'s Search Form and Result List

crawlers download the Web content, parse it and add the content to the designated index.

- **Annotator**: Web services stored in the *Services Index* are analyzed by the *Service Annotator*, which relates their properties to concepts of the *Ontology Index*. It also provides initial ranking of services according to their compatibility with different ontologies. Another type of analysis is carried out by the *Compositor*, which infers possible compositions of Web services.
- **Searcher**: When a user submits a query to a web server, the web server runs the *Searcher* which uses the *Services Index* and *Ontology Index* to answer the query at hand. An on-the-fly query analyzer parses the query and matches it against concepts stored in the *Ontology Index*.
- **User Interface**: There are two ways in which the search engine can be accessed. Humans use the Web-based user interface, while external systems can access the engine through a Web service based API.

REFERENCES

[1] Anupriya Ankolekar, David L. Martin, Zeng Zeng, Jerry R. Hobbs, Katia Sycara, Burstein Burstein, Massimo Paolucci, Ora Lassila, Sheila A. Mcilraith, Srini Narayanan, and Payne Payne. DAML-S: Semantic markup for web services. In *Proceedings of the International Semantic Web Workshop (SWWS)*, pages 411–430, July 13 2001.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[3] E. Christensen, F., G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1. Specification document, W3C, March 2001.

[4] Xin Dong, Alon Y. Halevy, Jayant Madhavan, Ema Nemes, and Jun Zhang. Simlarity search for web services. In *VLDB*, pages 372–383, 2004.
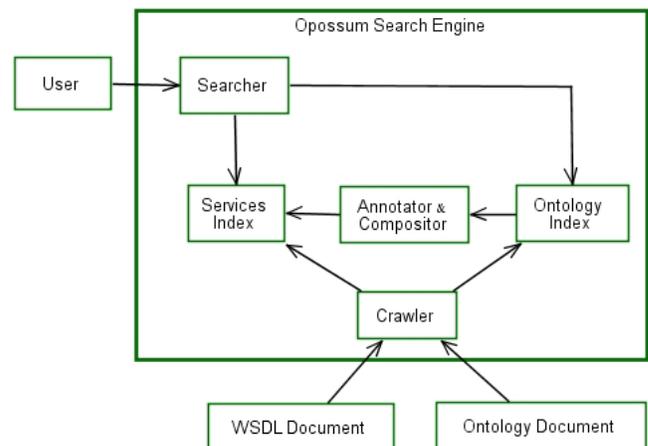
Fig. 3. The Architecture of the $\mathcal{OPOSSUM}$ Search Engine

[5] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara. Semantic matching of web services capabilities. In *International Semantic Web Conference*, pages 333–347, 2002.

[6] Evren Sirin, , James Hendler, and Bijan Parsia. Semi-automatic composition of web services using semantic descriptions. In *Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003*, Angers, France, April 2003.

[7] Eran Toch, Avigdor Gal, and Dov Dori. Automatically grounding semantically-enriched conceptual models to concrete web services. In *ER*, pages 304–319, 2005.