

A Query Paradigm for Web Services *

Mourad Ouzzani Athman Bouguettaya

Department of Computer Science

Virginia Tech

Falls Church, VA, USA

Abstract

The widespread adoption of XML standards has prompted an intense activity to address Web services research issues. The ability to efficiently access and share Web services is a critical step towards the full deployment of the new on-line economy. In this paper, we present a new query paradigm over Web services. Queries are resolved by combining the invocations of several Web service operations. A three-level query scheme allows the expression of database-like queries and their transformation into service execution plans bearing on actual Web service operations. We also introduce a hierarchical matching process between operations allowing the expansion of the solution space to similar and partial answers.

Keywords – Web Service, Queries, Semantic Web.

1 Introduction

The ever increasing amount of accessible information has made quality information access and use an laborious task. The main impediment relates to manipulating *semantics* on the Web so that information can be automatically and “meaningfully” processed. The envisioned *Semantic Web* is a paradigm shift to fulfill this goal [2]. In this paper, we are interested in the emerging concept of *Web services* [6]. We believe that *Web services* will have a propelling role in the achievement of the *Semantic Web*. A *Web service* is a set of functionalities that can be programmatically accessed and manipulated through the Web. Examples of Web services include air fares, flights status, weather forecast, stock quotes, car dealerships, government services, etc. Web services are gaining momentum as the potential *silver bullet* for tomorrow’s Web. This powerful concept is progressively taking root because of the convergence of business and government efforts to make the Web the place of choice for almost all types of human activities.

Enabling the *Service Web* requires the development

of techniques to address various challenging issues. Efficiently accessing and sharing Web services is critical for the full deployment of the new Web economy. An area of research that we are investigating is to offer *database-like* querying facilities where Web services are treated as *first class objects* that can be manipulated as if they were pieces of data.. Users submit queries that are resolved by combining invocations of various Web services. The challenge is then to devise the different alternatives of Web services combinations that answer those queries. The rationale is that a large portion of information on the Web is “hiding” behind Web services and value-added information can be obtained through their combination.

In this paper, we present the main concepts of a new *query paradigm* for Web services. To the best of our knowledge our approach is the first to provide querying facilities over Web services in such a novel way. In the following, we outline some of the motivations behind our approach for querying Web services.

- The proposed query paradigm enables powerful database-like querying over Web services.
- A large number of Web services offering “similar” functionalities are competing. However, they differ in the way these functionalities are offered and the different conditions to use them (e.g., *QoS*).
- For a given query, satisfactory answers may not be necessarily the exact answers. Users may be satisfied by “alternative” answers or “partial” answers. This opens more avenues for the optimization process.

2 Web Services Querying

Our goal is to enable queries to be answered by combining the invocations of Web services operations. Queries would undergo a sequence of transformation that leads to what we call a *service execution plan*. A *service execution plan* specifies among other things: (i) invocations of specific Web service operations in a certain order (this order could be statically defined at compile-time or dynamically devised at run-time), (ii)

*This research is supported by the National Science Foundation under grant 9983249-EIA and by a grant from the Commonwealth Information Security Center (CISC).

data flow operations to route input and output data amongst the query, the Web service operations, and other components, and (iii) data manipulation operations to build the results to be sent to the user. In this section, we elaborate on the different concepts and techniques to support queries over Web services.

2.1 Web Service Query Paradigm

We focus on querying the information flow carried out by the combined execution of several Web service operations. For that purpose, we propose a multi-level query paradigm where queries go through several transformations that lead to the *service execution plan*. Users are presented with “objects” to express and submit queries. Those objects are linked to the actual Web services through a set of Web service-like operations that are typical to a given application domain. These are *virtual* operations that links objects used to express queries and the actual Web services. More precisely, we define a three-level query paradigm:

- *Query level* – Contains a set of relations to allow users to submit queries. They are defined through mapping rules over the virtual operations.
- *Virtual level* – Contains Web service-like operations typically offered in a particular application domain. They are obtained based on expertise from that domain.
- *Concrete level* – Represents the space of Web services offered on the Web. These are potential candidates to answer queries.

2.2 Virtual Operations Representation

For each virtual operation, we need to describe attributes necessary for discovery, matching, and invocation. In particular, the discovery and matching processes depend tightly on semantic attributes. For that purpose, we assume that business partners in a particular application domain would agree on a *shared ontology*. That ontology would be used to describe semantic attributes of virtual and concrete operations.

On the other hand, for any input of a virtual operation, it is necessary to set its value to be able invoke that operation. As queries may contain conditions with inequalities, it should be possible to expand such inequalities into a set of equalities that provide appropriate bindings to the corresponding variables. Thus, we specify, as part of the description of virtual operations, all possible values of some input variables.

virtual operations are represented by a 5-tuple $Vop = (In, Out, Domains, Category, Function)$ where In is a set of pairs consisting of input variables and their types, Out a set of pair containing

output variables and their types (variables’ *types* are defined using XML schema types), $Domains$ a set of pair $(x, range)$ where x is an enumerable variable and $range$ is the set of all potential values for x , $Category$ describes the domain of interest of the operation (e.g., business, travel, weather, stock market, etc.), and $Function$ describes the business function offered by the operation (e.g., information, booking, listing). $Category$ and $Function$ capture the semantics of that operation. They are part of the *shared ontology*. We assume that Web services’ descriptions contain those attributes for their operations.

3 Deploying the Web Service Query Paradigm

The deployment of our multi-level query paradigm requires the definition of mapping rules between the query and virtual levels. We also present a matching approach that allows finding similar or partial answers to a query. We finally outline our *QoS*-based query optimization. More details on query optimization can be found in [5].

3.1 Mapping Rules

Relations are represented as conjunctive queries over virtual operations. Let \mathcal{R} be the set of relations defined at the query level and \mathcal{VOP} the set of virtual operations. For any relation $R_i \in \mathcal{R}$,

$$R_i(x_1, x_2, \dots, x_n) : - \bigwedge_j Vop_j(y_{j_1}, \dots, y_{j_m})$$

where x_i are the attributes of R_i , $Vop_j \in \mathcal{VOP}$, and y_{j_i} are input and output variables of the corresponding operation. This definition means that to get R_i ’s tuples, we need to invoke the different operations Vop_j . Query results will be then gathered from the different outputs of the virtual operations. In addition this definition does not dictate any order on the invocations of the different Vop_j s. That order depends on the current query being processed.

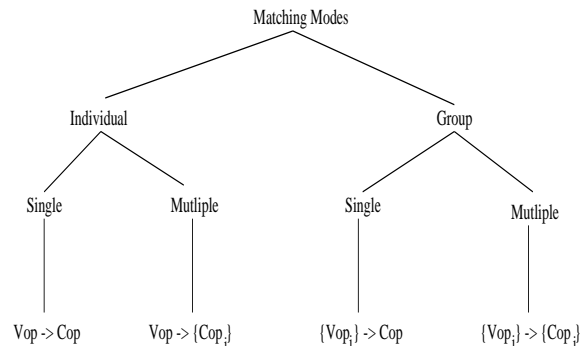


Figure 1: Hierarchical Operations Matching

3.2 Operations Matching

After locating potentially relevant Web services from available Web service registries, operations are matched against virtual operations. It is not always possible to find an exact match for a given virtual operation. In addition, the same functionality may be offered in various ways by different Web services. Finally, in such settings, users may be inclined to accept similar or close answers to their queries. For that reasons, we propose a hierarchical matching scheme as depicted in Figure 1. Matching could be applied for an *individual* virtual operation or a *group* of virtual operation (clustered based on some criteria). In both cases, the matching may concerns *single* concrete operation or *multiple* concrete operations. This defines four different matching modes: *Individual/Single*, *Individual/Multiple*, *Group/Single*, and *Group/Multiple*. For each matching mode, several alternatives may be obtained by varying the way that attributes from virtual and concrete operations are related to each others. This is conducted from both syntactic and semantic perspectives. In this paper, we focus on three alternatives for each of the first two matching modes, namely *Individual/Single* and *Individual/Multiple*.

Concrete Web services descriptions provide the necessary information about their operations, i.e., *name*, *In*, *Out*, *Category*, and *Function*. The different matching modes and their alternatives are defined as follows.

- *Individual/Single Mode* - Let *vop* and *cop* be two virtual and concrete operations respectively. *vop* is matched to *cop* using one of the following alternatives:
 - *Exact match* - The concrete operation matches the virtual operation in every aspect. Two operations *match exactly* if they have the same sets of input and output variables, and they have the same *Category* and *Type* attributes.
 - *Overlap match* - Looks for operations that offer *close* functionalities to that of the virtual operation. Two operations *Vop* and *Cop match by overlap* if they have the same sets of input and output variables, and their *Category* and *Type* attributes overlap.
 - *Partial match* - Corresponds to the case where input and output attributes of the two operations do not necessarily coincide. Two operations *Vop* and *Cop match partially* if *Out(Cop)* is a subset of *Out(Vop)* or *In(Cop)* is a subset of *In(Vop)*, and their *Category* and *Type* attributes are the same or overlap.

- *Individual/Multiple Mode* - In this mode, the virtual operation is matched to a combination of several concrete operations. This means that we can find a set of operations which combined invocations deliver the same functionality as the virtual operation. Let *Vop* be a virtual operation and $\{Cop_i\}$ a set of concrete operations. We define for this mode, three alternatives similar to the previous mode.

- *Exact composite match* - A virtual operation *Vop match exactly* a set of $\{Cop_i\}$ if $In(Vop) = \cup In(Cop_i)$, $Out(Vop) = \cup Out(Cop_i)$, and all *Cop_i*s have the same *Category* and *Type* attributes as *Vop*.
- *Overlap composite match* - A virtual operation *Vop match by overlap* a set of $\{Cop_i\}$ if $In(Vop) = \cup In(Cop_i)$, $Out(Vop) = \cup Out(Cop_i)$, and for some or all concrete operations *Cop_i*s their *Category* and *Type* attributes overlap with those of *Vop*.
- *Partial composite match* - A virtual operation *Vop match by overlap* a set of $\{Cop_i\}$ if *In(Vop)* is a subset of *In(Cop_i)* or *Out(Vop)* is a subset of *Out(Cop_i)*, and for some or all concrete operations *Cop_i*s their *Category* and *Type* attributes either are equal or overlap with those of *Vop*.

3.3 Quality of Service and Rating

QoS is defined through a number of parameters supplied by the Web service providers. We consider the following *QoS* parameters [7, 3]: *Latency* (the average time it takes for an operation to return results after its invocation), *Fees* (the dollar amount to pay to invoke operations), *Availability* (the probability that the Web service is present and ready to be invoked), *Accessibility* (the degree of being capable of serving a request), and *Reliability* (the degree of being capable of maintaining the service and service quality).

QoS as advertised by a Web service may not be always fulfilled. Also, the Web service may need to change some *QoS* values. For an accurate use of *QoS* parameters, we adjust advertised values depending on the behavior of the Web service. *QoS* adjustment is achieved by monitoring Web services and comparing the *delivered QoS* with the *promised QoS*. This comparison allows to rate Web services accordingly and use that rating in the optimization process. The *promised QoS (pQoS)* is the value of the *QoS* parameters advertised by the service provider. The *delivered QoS (dQoS)* is the actual value of the *QoS* parameters.

3.4 Query Processing

A query is subject to several transformations leading to a *service execution plan*. This plan is obtained through the following phases:

- **Query unfolding** – Each relation is unfolded to virtual operations using the corresponding mapping rule. We obtain a new query consisting of virtual operations and conditions on their attributes. For conditions with inequalities, the corresponding input variables are expanded into all their possible values based on their ranges.
- **Operations matching** – For each virtual operation, the system looks for relevant Web services through service registries. Descriptions of returned Web service are then searched for operations that match (using the hierarchical matching) the virtual operation.
- **Service execution plan generation and optimization** – Located operations from the matching phase are selected and combined together based on the defined objective function and feasibility. *Feasibility* means that the obtained plan can be actually executed. The query engine mainly checks that input variables required by any operation to be invoked are bound.

4 Related Work

A major distinction between our query paradigm and service composition [4] is that we do not intend to build new Web services out of existing ones. Rather, our goal is to provide database-like query facilities over Web services. Users and applications get information by submitting queries that are resolved by combining access to different Web services. Little work has been done on querying Web services. An example is *Active XML* [1] which enables the querying of XML documents embedded with Web services invocations. A major difference with our work is that we view Web services as first class objects while Active XML uses Web services to “update” XML documents and the querying targets those XML documents. Additionally, *Active XML* does not support any type of optimization.

A major difference between queries in databases and our approach lies in the manipulated objects. The first class objects in the proposed approach are *services* while *data* is the first class object in the existing database techniques. In our approach, the optimization focus is on several *QoS* parameters related to the behavior of the Web services while in most existing techniques, optimization concerns the efficiency of the query execution (e.g. response time). Our research

goes beyond database integration on the Web. We believe that while databases integration is still relevant in some specific domains, providing query facilities over Web services

5 Conclusion

In this paper, we presented a new query paradigm for querying Web services. To the best of our knowledge, the proposed paradigm is the first to provide database-like querying over Web services in such a novel way. Queries are expressed over the *query level* and transformed into a combination of *virtual operations* invocations. This transformation uses *mapping rules* defined between relations at the query level and virtual operations. The virtual operations are then matched against operations from Web services at the *concrete level* based on hierarchical matching scheme. The work presented in this paper is part of our ongoing project to define the architectural components of a *Web Services Management System (WSMS)*. The aim of a *WSMS* is to do for Web services what DBMSs have done for data. Users will no longer need to think in terms of *data* but rather *services*.

References

- [1] S. Abiteboul, O. Benjelloun, and T. Milo. Web services and data integration. In *International Conference on Web Information Systems Engineering*, 2002.
- [2] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
- [3] A. Mani and A. Nagarajan. *Improving the performance of your Web services*. <http://www-106.ibm.com/developerworks/library/ws-quality.html>, January 2002.
- [4] B. Medjahed, A. Rezgui, A. Bouguettaya, and M. Ouzzani. Infrastructure for E-Government Web Services. *IEEE Internet Computing*, 7(1), January/February 2003.
- [5] M. Ouzzani, A. Bouguettaya, B. Medjahed, and A. Rezgui. Efficient Querying of Web Services. *Submitted*, April 2003.
- [6] S. Tsur, S. Abiteboul, R. Agrawal, U. Dayal, J. Klein, and G. Weikum. Are Web Services the Next Revolution in e-Commerce? (Panel). In *VLDB Conference*, September 2001.
- [7] S. Vinoski. Service Discovery 101. *IEEE Internet Computing*, 7(1), Jan/Feb 2003.