Requestor Friendly Web Services

Ravi Konuru and Nirmal Mukhi

IBM Research,
19 Skyline Drive, Hawthorne, NY 10591, USA
Email: {rkonuru, nmukhi}@us.ibm.com

Abstract. Web service providers rely on the Web Service Description Language (WSDL) as the way to communicate information about an available service to a service requestor. This description or meta-data of the service is used by a service requestor to inspect the available interfaces and to access the service. In this paper, we argue that publishing a WSDL with a functional description of a service alone is not requestor friendly, i.e., it does not allow the requestor any flexibility in improving the end-to-end responsiveness and customize the Web Service behavior. We present some scenarios to back this argument and also outline a spectrum of solution approaches.

1 Introduction

Web services efforts, to a large extent, are being driven by the industry to solve business problems. Business needs demand seamless integration of information within and across enterprises to improve operational efficiency in terms of time and resources used. Information flow and updates must happen in a composite context of security, transactionality, processes, workflow, etc. Efforts are underway to specify and create new generation of systems based on Web services standards that abstract these concepts and incorporate security, transactions, orchestration and choreography, grid computing capabilities, business documents and processes, simplified integration and mapping to existing middleware systems and application assets. In other words, the Web services framework is defining a unified architecture and software infrastructure that can support the transformation of existing IT assets into services which businesses can use and integrate to perform effectively and efficiently in a heterogeneous, dynamic, distributed, multi-domain environment.

Service metadata is a core concept in the Web services framework. This metadata is used by service providers to generate partial service implementations and configure supporting middleware and by service requestors to generate the necessary client paraphernalia to communicate with a service. Traditional distributed object systems such as CORBA [1] rely on interface descriptions as being sufficient descriptions of a distributed object. Web services framework implementations have consciously or unconsciously adopted this notion for services and as a result rely on a WSDL [2] document as being an adequate rendition of the service for requestors to use.

The primary purpose and contribution of this paper is to motivate the need for giving service requestors access to a more complete metadata description of a Web service, covering aspects beyond the interfaces and protocols. This can allow service requestors to reason about a Web service and make smarter decisions when it comes to using it, resulting in improved responsiveness and customization. We describe various scenarios where detailed service metadata can provide superior value. Finally, we outline our thoughts on a spectrum of approaches to address this issue.

2 Scenarios

Currently, given a WSDL description, a client side tool or infrastructure is able to either generate a proxy or act as a universal proxy that can communicate with the web service. The advantage of this approach it simplifies service development by completely factoring out the infrastructure available at a service requestor's end from the service provider. This method of using a web service from a client must always be supported. However, the argument here is that it this isolation between the provider and the requestor comes with certain costs:

Since the proxy has no knowledge of the service beyond the service description, every call to the web service must necessarily perform a round trip to the provider. As a result, the proxy cannot leverage local resources (locally known Web services that provides better quality of service, available CPU cycles, etc.) when acting on behalf of the service.

Since all requestors are provided with the same coarse-grained view of the service, there is no direct support for customization of the service based on an individual requestor's execution context and its requirements.

In this section, we present scenarios where giving the service requestor prominence in the definition of the service metadata as well as service infrastructure can result in new levels of functionality and performance.

2.1 Responsiveness

Consider a web service that is being used to interactively query and update a catalog. The query interface is via an interactive forms interface where the user can enter input and select various criteria. A form submit translates to conceptually a new query or update on the catalog. In addition to the types of individual inputs, the catalog update and query governed by several rules that express the relationship among the inputs. For example, one rule might be if the value for the material input is Gore-tex then the product-type-list must contain one or more values from the set {Clothing, Shoes}. As another example, another rule might be that the value entered for a Suede jacket must always be greater than that of a Denim jacket of the same size. In other words, there is some validation that needs to be performed before the request is processed and it cannot be handled in the context of the basic type system.

With current Web services infrastructure, this validation can potentially result in several wasted roundtrips to the server, increasing load on the server and frustration levels at the requestor's end. There should be a standard mechanism by which a web service can provide either extra semantics about its data model or provide validation logic that can be used by the requestor to reduce round trips to the server.

The main point to take away from this scenario is that the issue of providing responsiveness exists in today's web application domain and is being addressed in a variety of ways (which we cover in the next section of this paper) all assuming some functional capability of the client endpoint. With web services as a normalization concept, there is a need to address the problem at a level above the technologies and languages and provide the right abstractions and mappings/bindings to a widely agreed upon set of technologies and lower-level standards.

While the above scenario describes a user-facing application, it can also be mapped to automated application-to-application interactions. The Web services standards already build on XML schema to define data types. This allows the requestor end to validate requests for prior to sending them on the wire. Providing a more complete model of the data, with additional semantics such as in XFORMs is another step in that direction.

2.2 Customization

Consider the scenario where an application is interacting with a Call Center Web service, that itself uses multiple services to perform its function: a customer lookup service, a mailer service, a spell checker service, and a problem report service. By default the spell checker service used is the Webster spell checker service. When the requestor creates in a problem statement in a particular technical domain, the requestor application would like intelligent prompts and spell checks. In the case where the service composition details are completely hidden from the client, the Webster service returns a highlighted text via the composing Call Center service indicating what it thinks are incorrect words. However, if the Call Center service exposed aspects of its composition in a well-defined manner along with the defaults, it would enable the requestor application to dynamically indicate which particular spell checker service to use based on the current problem. In fact, the new spell checker might be a native application that is part of the software infrastructure available at the requestor's end that can offer better response and integrates better with the eventual application.

This scenario has some aspects that are indirectly related to efforts in the industry related to Web service composition, coordination and orchestration (BPEL4WS [3], WS-C, W3C Choreography, …) and in particular to the notion of abstract definition of web service interfaces, relationships, flow and binding them to implementations at a later time. However, this scenario has a much simpler need, the ability for a service to simply export its dependent services and default bindings in a standard manner and the ability for a client to over-ride the default bindings.

3 Related Work

There is an enormous body of work in both research and industry that is relevant to support requestor friendliness as described in this paper, some of which we describe below. Research in distributed object systems that supports mobility and disconnected operation is another source towards a generic solution. The attempt of this section is not to be complete, but the main point to take away is that there is a lot of work that we can leverage and attempt to normalize at the level of web services without any bias towards a particular set of technologies.

The problem of frequent round tripping to the server is recognized in the web application domain where programmers resort to using JavaScript to perform not only just validation but also several other functions such as sorting and simple computations that do not go back to the server. XFORMs specification [4] also addresses this problem supporting the definition of data models and corresponding constraints. An XFORMs compliant browser, on receiving an XFORM document, can perform a great deal of validation including at the level of instance values without requiring any code from the server or round trips to the server.

Caching of data on the requestor's end is a well known approach to improving responsiveness of distributed applications. Mowbray and Malveau [5] illustrate the use of such smart client stubs in their "Fine Grained Framework".

WSRP [6] proposes a way for user-facing Web services to be plugged into portals. It also allows expression of cache policies so that presentation data for the service can be stored locally on the client. Our approach advocates a similar expression of suitable policies and other metadata, going beyond the realm of presentation data alone.

The BPEL4WS specification [3] defines *abstract compositions* of Web services. It allows the actual service instances used in a composition to be configured separately, possibly at runtime. This plays directly into our requirement for a Web service to expose its dependent services in a standard manner.

4 Towards a Solution

A key challenge in proposing requestor-friendly services and designing a supporting software infrastructure is that we need to be able to leverage existing open standards, in keeping with the philosophy behind the definition of the Web services platform. So much as XML Schema has become part of the Web services vocabulary, we can leverage higher level data models such as XFORMs. Another challenge is to be able to support existing methods of accessing Web services, while at the same time taking advantage of more metadata if that is available.

The abstract approach that we are experimenting with is based on the classic MVC paradigm with a little twist. Specifically, in this approach, a Web Service conceptually consists of one or more of the following elements:

- A data model that represents the business data of the service along with all its constraints.

- A presentation model, separated from the data model since the policies associated with its use are often unique.
- Execution model that operates on the data and can provide information about constraints on interaction with the service. The execution model can be a platform independent description such as WSDL and BPEL4WS documents or may correspond to java byte code.
- A connection model or a plug-in model that specifies the services/service interfaces that the service depends on and the additional requirements imposed on larger compositions due to the presence of such dependencies.

The basic idea is that we provide a means for a Web service to export the above models and in addition what is required on the requestor to interpret/execute those models. Then a model-aware requestor in conjunction with the service can make decisions the amount of processing to be performed on the requestor versus the provider end. All this has to be done in a secure and seamless manner especially when code deployment is involved on the requestor's end.

In the first phase of this work we have begun to design a model-aware service invocation framework based on WSIF [7] and hope that it eventually results in the definition of a more requestor-friendly metadata stack for Web services..

5 Summary

We motivated the need to extend Web services vocabulary and usage patterns to better support and exploit service requestors. Several solutions are possible but we believe that it can be completely transparent to application code by encapsulating this new function within a generic requestor framework. An application is not required to use this new functionality; existing application independent proxy generators can still be used. This work will leverage similar research in web applications and traditional distributed systems, and augment Web service descriptions and policies with existing standard ways of defining data and execution constraints wherever possible.

Acknowledgements

We thank Francisco Curbera and Sanjiva Weerawarana for their comments on this paper.

References

1. CORBA (Common Object Request Broker Architecture) 3.0, published on the World Wide Web by OMG, July 2002, http://cgi.omg.org/docs/formal/02-06-33.pdf
2. Christensen, E., Curbera, F., Meredith, G. and Weerawarana., S. Web Services Description Language (WSDL) 1.1. W3C, Note 15, 2001, www.w3.org/TR/wsdl

3. Andrews, T., Curbera, F. et. Al. Business Process Execution Language for Web Services (BPEL4WS) Version 1.1, ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf
4. Dubinko, M., Klotz, L. et. al. XFORMS 1.0, published on the World Wide Web by W3C, http://www.w3.org/TR/xforms/
5. Mowbray, T. and Malveau, R., "CORBA Design Patterns", published by John Wiley and Sons.
6. Diaz, A., F., Peter, WS-RP (Web Services for Remote Portlets) published on the World Wide Web by IBM, January 2002, http://www-106.ibm.com/developerworks/web/library/ws-wsrp/?dwzone=web
7. Duftler, M., Mukhi, N. et. al. Web Services Invocation Framework (WSIF), OOPSLA Workshop on Object Oriented Web Services, October 2001.