Web Services Need Consistency

Giacomo Piccinelli
Hewlett-Packard Laboratories
Stoke Gifford Park
Bristol BS34 8QZ, UK
giacomo_piccinelli@hp.com

Anthony Finkelstein & Christian Nentwich
Dept. of Computer Science
University College London
Gower Street, London WC1E 6BT, UK
{a.finkelstein|c.nentwich}@cs.ucl.ac.uk

Abstract

Web Services provide a powerful access channel to business capabilities. Inside the company, resources can be shared and costs reduced. Outside the company, the interaction with customers and suppliers can be streamlined and business opportunities extended. A major issue with the current Web Service model is the problem of composition and the difficulties that arise as Web Services are assembled, typically in a bottom-up manner. We briefly discuss this issue, and propose a lightweight solution based on automated consistency checking of Web Service interfaces.

Introduction

Since their appearance in 1998, the main expectations for Web Services were for an evolutionary approach to the accessibility of business assets. Companies could already count on considerable data-processing capabilities. The challenge was to make available such capabilities within the company, as well as to business partners.

Web Services leverage the experience built in technical communities ranging from distributed systems to software engineering. For example, significant influences come from the RM-ODP (Reference Model for Open Distributed Processing). Technical and business requirements are distilled into a simple but effective model, which applies uniformly to different types of resources. The main roles in the Web Service model are service provider, service consumer, and service registry. The main artefacts are services and service descriptions. The operational model is based on the find-bind-use approach. The consumer looks for the provider of a given service in the registry. If the search is successful, consumer and provider establish a connection and the service delivery process can be performed. From a technical perspective, open standards are instrumental for the success of Web Services. Initiatives such as UDDI (Universal Description, Discovery and Integration) and the W3C Web Service Activity are noticeable examples. A more complete overview of the Web Service landscape can be found in [2].

Considering the current stage of evolution for model, standards, and technology, the Web Service initiative reveals a solid bottom-up approach. The emphasis has been on connectivity and access to resources, which is the foundation for the development of concrete business solutions. The downside of the bottom-up approach is that the gap between the business and the technical notion of services is still open. A business service is concerned with the end-to end delivery of a meaningful business outcome, a much coarser grain than that of the typical Web Service. There is a significant amount of work and infrastructure required on top of the Web Service layer in order to create a business solution.

In this paper, we propose a lightweight approach to the reconciliation of business services and Web Services. The approach is based on checking the conformance of, or consistency between, a set of Web Service interfaces and a business service policy. These business service policies constrain the valid compositions of Web Services. To perform the checking we use `xlinkit` [1], a consistency management framework. We first outline the lifecycle of a Web Service, and discuss consistency issues with respect to business services. We then describe a framework for consistency management, which is based on `xlinkit`.

Web Service Lifecycle

The current scope of the Web Service model influences significantly the lifecycle of a Web Service. As a very rough approximation, one Web Service can be compared to one method in more traditional software contexts. A service provider is represented by a set of ports (one port for each Web Service) that the service consumer connects to in order to request a service instance (invoke the method). The interaction between provider and consumer substantiates in one request-response exchange of information. Concepts such as the identity of a service instance or the aggregation of multiple services into higher-level entities (e.g. the equivalent of objects for methods) are not currently supported by the more established part of the Web Service model.

The lifecycle of a Web Service starts with the definition of a capability with a suitable granularity. Examples of Web Service used inside a company are the printing of a document, specific queries to databases (e.g. list of available places on a training course), and the notification of specific events (e.g. process completion). Examples of Web Service used by a company for external interaction are the submission of an order, the notification of payment, and the request of product information. The business capabilities are already implemented by the IT system of a company. The creation of the Web Service layer mainly consists of the specification and publishing of the service content. The service specification is formalised using WSDL (Web Service Description Language). Using the

analogy with methods, a WSDL description captures the signature for a Web Service. The description of the Web Service is then categorised and added to the information base of a UDDI registry. Assuming the back-end integration is in place, the Web Service is ready for use.

The Need for Consistency

Web Services provide an effective access route to individual business capabilities. Still, one of the issues that solution developers are left with is the gap between individual capabilities and a complete business service. Even simple business services usually require the coordinated usage of a number of different capabilities. For example, submitting orders is only one aspect in the overall service logic involved in a direct-sale service. Moreover, the interaction involved in a business service usually goes well beyond one request-response exchange. In the direct-sale example, the flow of communication may start with a product enquiry to end with a payment confirmation. The business conversation is driven by different roles at different stages.

The requirements for a business service translate into requirements for the implementation infrastructure, which includes the Web Service layer. A coherent view of the various components involved in the implementation is essential for the realisation of the overall service. The Web Service model brings technological homogeneity into the picture but does not address this coherence.

A Lightweight Approach

Our proposal for the management of consistency among a dynamic set of Web Services finds concrete instantiation in the use of the `xlinkit` framework.

`xlinkit` is a framework for checking the consistency of distributed, heterogeneous documents. It comprises a language, based on first order logic, for expressing constraints between such documents, a document management mechanism and an engine that checks the documents against the constraints. A full description of `xlinkit`, including a formal specification of its semantics, its scalability and an evaluation can by found in [1]. An interactive tutorial is available at http://www.xlinkit.com The infrastructure of `xlinkit` is especially geared towards the processing of XML-encoded documents, and the language provides specific capabilities for expressing direct relations between documents.

We propose to apply `xlinkit` to the WSDL documents describing the Web Services related to a business service. The `xlinkit` rule language will be used to describe business service policies that must be adhered to by the Web Services from which the business service is constituted.

This is best seen by way of a simple example. A business service policy might be “every time a message deals with special offers a transaction ID has to be used”. Such a policy is required to support auditing of special price deals.

The following are data type definitions in a WSDL file(s):

```
<types>
 <schema targetNamespace="http:// freightmixer.com/specialoffers.xsd"
xmlns="http://www.w3.org/1999/XMLSchema">
 <element name=“SpecialOffersAvailableForDestination">
      <complexType> <all>
        <element name=“departureCity" type="string"/>
        <element name=“destinationCity" type="string"/>
        <element name=“deliveryMonth" type="string"/>
        <element name=“loadWeight" type=“integer"/>
        <element name=“transactionUniqueIdentifier"  type=“integer"/>
    </all> </complexType>
 </element>

 <element name=“SpecialOffers">
      <complexType> <all>
         <element name="price" type="float"/>
          <element name=“transactionUniqueIdentifier"  type=“integer"/>
       </all> </complexType>
 </element>
</schema>
</types>
```

The following is the part of the WSDL file(s) where the messages are defined.

```
<message name=“RequestForSpecialOffersInput">
    <part name="body" element="xsd1:
SpecialOffersAvailableForDestination "/>
</message>

<message name=“RequestForSpecialOffersOutput">
    <part name="body" element="xsd1: SpecialOffer"/>
</message>
```

The policy can be defined simply:

```
<forall var="m"
in="./freightmixer/wsdl/message[contains(@name,'SpecialOffer')]">
  <forall var="y" in="./freightmixer/wsdl /types/xsd:schema/xsd:element">
    <implies>
      <equal op1="substring-after($m/part/@element,':')"
              op2="$y/@name"/>
      <exists var="e" in="$y/xsd:complexType/*/xsd:element">
        <equal op1="$e/@name" op2="'transactionUniqueIdentifier'"/>
      </exists>
    </implies>
  </forall>
</forall>
```

The evaluation of `xlinkit` rules by the check engine generates hyperlinks that link consistent and inconsistent elements as defined by the rule. If the policy above were violated, it would link together the message element and the xsd:schema element that is referenced by the message. The user could infer that the schema element needs a subelement transactionUniqueIdentifier in order to be used with this message.

Conclusions

We believe that the expressive, declarative and formally defined `xlinkit` rules are well suited to expressing business service policies by way of constraints across WSDL descriptions. In this style policies are specified once, and they can be automatically applied to evolving sets of Web Services related to one or more business services. The semantic domain for a policy can range from business-level requirements (e.g. always send the customer a contact reference) to more technical requirements (e.g. concerning the use of appropriate XML schemas). The `xlinkit` framework is lightweight and itself readily deployable in as a Web Service. The implicit compositional constraints seem to be well aligned with bottom up construction of business services.

The approach outlined above must however, be treated with some caution. It has only been briefly sketched and is in need of systematic evaluation. Such an evaluation forms the next step in our research programme.

References

[1] Nentwich C., Capra L., Emmerich, W. and Finkelstein, A. *"xlinkit: a Consistency Checking and Smart Link Generation Service"* ACM Transactions on Internet Technology, 2(2), May 2002, pp. 151-185.

[2] Piccinelli G., Emmerich W., Zirpins C., and Schütt K. *"Web service interfaces for inter-organisational business processes: an infrastructure for automated reconciliation"* Conference on Enterprise Distributed Object Computing (EDOC), Lausanne, Switzerland, 2002.