A Simple Format for the Specification of Service Integrator Assumptions
Version 1.0

Cu D. Nguyen, Alessandro Marchetto, Paolo Tonella
Software Engineering Research Unit
Fondazione Bruno Kessler
Trento, Italy
cunduy,marchetto,tonella@fbk.eu

January 15, 2011

Abstract

Service Oriented Architecture (SOA) help building systems that integrate services from separate service providers from different business domains. Such systems, also called as service compositions, can have capability of discovering interoperable services and binding them at runtime, thus, bringing greate flexibility to service integrator. However, adopting SOA takes away the ability of the service integrator to control and observe the services that are developed by other parties. As a consequence, he has to make assumptions about the types of services that the service composition under development accesses and will access to. In practice, these assumptions are often specified as validating code regarding service invocations, or as invariant assertions spreading in the service composition code. This makes the maintenance of evolution of the assumptions difficult and expensive.

We propose a format for service assumption specification. This format allows to specify the integrator assumptions regarding accessed services in a single place, helping the maintenance of the assumption specifications. Moreover, the assumption specification document regarding a service can be used directly to validate data received from the service, this helps removing validation code from those that do business logic. As a result, one can obtain cleaner composition code and more complete and easier to maintain assumption specification and validation.

1 Introduction

As Webservices standards are becoming more and more mature, the adoption of Webservices is growing fast in many areas, from business integration and social media, to end-user applications. There are more and more service providers who offer Webservices facilities, e.g. Web API, REST[5] or SOAP[1] Webservices, allowing service consumers to easily access, integrate, and share their business data. At the time of writing this paper, according to the site Programmable

Web[1], there are about 2,500 APIs available and about 5,500 *Mashup* systems that exploit those APIs. These numbers are increasing constantly.

Moreover, webservice providers have to evolve their services quickly to meet technology evolution and business changes. For instance, eBay released 8 versions[2] of their finding service within each year, from 2009 to 2010. Similarly, Amazon released 12 versions[3] of their *Elastic Compute Cloud* service per year. Changes are new business features, bug fixes, and performance improvements. As a result, service integrators, who use webservices in their systems, face frequently a critical decision either to update their service composition system to the new versions of the services they are exploiting, or staying with the old ones knowing that the old services might have issues or risks. Compatibility checking is required.

During the course of integrating services from different providers into a system, a service integrator often makes assumptions about technical details regarding the data format that the system will receive from the services. We call these as *service assumptions* from now on. For example, the integrator might expect a list of phone numbers, separated by commas, from the service, when searching for a phone number; or he expects the price of an item to be greater or equal to zero. Such technical information is often specified in informal documents related to the services or is an implicit shared understanding between service providers and integrators. In practice, some of the assumptions are implemented scattering in the code as validating code or assertions, some are neglected.

To support the service integrators in ensuring compatibility and making update decision, we propose to specify service assumptions explicitly in a single formal document and use them as criteria: violating such assumptions implies that the current system might have problem with the new services, exploiting them may require modifications and regression testing; while if no violation is observed, the system is compatible with (and should access to) the new services.

As the first attempt, we focus on the specification of assumptions regarding XML message responses from Webservices. There are other types of assumptions, e.g. those related to interaction protocols. However, we consider them as future extensions.

This technical report describes an XML format for the specification of service assumptions with respect to XML messages (SOAP, REST, any kind of messages in XML format). We discuss technical details of the specification and how to apply it. The benefits of specifying service assumptions are three folds:

1. Using explicit service assumption specifications to check data received from services, this helps the integrator detecting and dealing with bad data properly.

2. Putting together scattered service assumptions (and their related validating code elsewhere in the system) in a single place, reducing maintenance cost and making it easier to check for completeness.

3. Specifying service assumptions in a high-level language, e.g. XML, separately from code, this allows to evolve them easily at runtime, no code

[1]See `http://www.programmableweb.com`
[2]`http://developer.ebay.com/DevZone/finding/ReleaseNotes.html`
[3]`http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=86`

modification and recompilation is needed.

In order to understand our format for service assumption specification, the following background knowledge is required: XML, XML Schema[3], XPath[4].

2 Assumption Specification

An XML message is an XML document that contains XML elements that can be reached by using XPath[4]. Service assumptions are specified to pose restrictions on XML elements. An assumption is composed of an XPath expression that links to XML elements and restrictions that the service integrator expects those elements to obey:

```
  assumption {
   xpath : XPath expression
   restrictions
 }
```

We adopt the restriction specification in XML Schema [3] to the specify restrictions in service assumptions. The motivations are twofold. First, service integrators are familiar with XML Schema, so no or less effort is required to lean how to specify assumption restrictions. Second, XML Schema is widely used, the expressiveness of its restriction specification is sufficient for many real usages. Moreover, this adoption allows assumption specifications to be combined with service interface specification, mostly in WSDL[2] or XSD. Tools that generate service binding code can take these combination to generate better code for data validation.

Service assumptions for each type/family of XML responses are put together in a XML file. The detailed structure of these XML files is specified in XSD as follows.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.fbk.eu/ServiceAssumption" xmlns:tns="http://www.fbk.eu/ServiceAssumption"
    elementFormDefault="qualified">

    <xs:element name="ServiceAssumptions">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="onElement" type="tns:onElementType"
                    maxOccurs="unbounded" />
            </xs:sequence>
            <xs:attribute name="author" type="xs:string" />
            <xs:attribute name="version" type="xs:string" />
            <xs:attribute name="createdDate" type="xs:string" />
            <xs:attribute name="appliedTo" type="xs:string" />
            <!-- Service link that these assumptions should be applied to  -->
        </xs:complexType>
    </xs:element>

    <xs:complexType name="onElementType">
        <xs:choice>
            <xs:element name="restriction" type="tns:complexRestrictionType"></xs:element>
            <xs:element name="constraint" type="tns:constraintType"></xs:element>
        </xs:choice>
        <xs:attribute name="name" type="xs:string" use="required" />
        <xs:attribute name="xpath" type="xs:string" use="required" />
    </xs:complexType>

    <xs:complexType name="constraintType">
        <xs:sequence>
            <xs:element name="expr" type="xs:string" maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="lang" type="tns:constraintLanguage" />
    </xs:complexType>

    <xs:simpleType name="constraintLanguage">
        <xs:restriction base="xs:string">
            <xs:enumeration value="OCL" />
            <xs:enumeration value="XCL" />
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="complexRestrictionType">
        <xs:sequence>
            <xs:group ref="tns:simpleRestrictionModel" />
        </xs:sequence>
        <xs:attribute name="base" type="xs:QName" use="required" />
    </xs:complexType>

    <xs:group name="simpleRestrictionModel">
        <xs:sequence>
            <xs:group ref="tns:facets" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:group>

    <xs:group name="facets">
        <xs:choice>
            <xs:element ref="tns:minExclusive" />
            <xs:element ref="tns:minInclusive" />
            <xs:element ref="tns:maxExclusive" />
            <xs:element ref="tns:maxInclusive" />
            <xs:element ref="tns:totalDigits" />
            <xs:element ref="tns:fractionDigits" />
            <xs:element ref="tns:length" />
            <xs:element ref="tns:minLength" />
            <xs:element ref="tns:maxLength" />
            <xs:element ref="tns:enumeration" />
            <xs:element ref="tns:whiteSpace" />
            <xs:element ref="tns:pattern" />
        </xs:choice>
    </xs:group>

    <xs:element name="totalDigits" id="totalDigits">
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="tns:numFacet">
                    <xs:attribute name="value"
```

```
type="xs:positiveInteger"
                        use="required" />
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="fractionDigits" id="fractionDigits"
type="tns:numFacet">

    </xs:element>

    <xs:complexType name="facet">
        <xs:attribute name="value" use="required" />
    </xs:complexType>

    <xs:element name="minExclusive" id="minExclusive"
type="tns:facet">
    </xs:element>

    <xs:element name="minInclusive" id="minInclusive"
type="tns:facet">
    </xs:element>

    <xs:element name="maxExclusive" id="maxExclusive"
type="tns:facet">
    </xs:element>

    <xs:element name="maxInclusive" id="maxInclusive"
type="tns:facet">
    </xs:element>

    <xs:complexType name="numFacet">
        <xs:complexContent>
            <xs:restriction base="tns:facet">
                <xs:attribute name="value"
type="xs:nonNegativeInteger"
                    use="required" />
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="noFixedFacet">
        <xs:complexContent>
```

```xml
      <xs:restriction base="tns:facet">
        <xs:attribute name="fixed" use="prohibited" />
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>


  <xs:element name="length" id="length" type="tns:numFacet">

  </xs:element>
  <xs:element name="minLength" id="minLength" type="tns:numFacet">

  </xs:element>
  <xs:element name="maxLength" id="maxLength" type="tns:numFacet">

  </xs:element>

  <xs:element name="enumeration" id="enumeration" type="tns:noFixedFacet">

  </xs:element>

  <xs:element name="whiteSpace" id="whiteSpace">

    <xs:complexType>
      <xs:complexContent>
        <xs:restriction base="tns:facet">
          <xs:attribute name="value" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:NMTOKEN">
                <xs:enumeration value="preserve" />
                <xs:enumeration value="replace" />
                <xs:enumeration value="collapse" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
    </xs:element>

    <xs:element name="pattern" id="pattern">
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="tns:noFixedFacet">
                    <xs:attribute name="value" type="xs:string" use="required" />
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

3 Application Example

The following listing shows an example of service assumption specification. Assumptions are specified for the message responses of eBay Finding Webservice API, method findItemsByKeywords(...). The WSDL of the service is available from this link: `http://developer.ebay.com/webservices/finding/latest/FindingService.wsdl`, accessed Janurary 2011.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ServiceAssumptions xmlns="http://www.fbk.eu/ServiceAssumption"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.fbk.eu/ServiceAssumption
ServiceAssumption.xsd ">
    <onElement xpath="//findItemsByKeywordsResponse/searchResult/
item/itemId" name="title">
        <restriction base="string">
            <minLength value="5"/>
            <maxLength value="256"/>
        </restriction>
    </onElement>
    <onElement xpath="//findItemsByKeywordsResponse/searchResult/
item/title" name="title">
        <restriction base="string">
            <whiteSpace value="collapse"/>
        </restriction>
    </onElement>
    <onElement xpath="//searchResult/item/primaryCategory/
categoryId" name="categoryId">
        <restriction base="integer">
            <maxInclusive value="99999"/>
            <minInclusive value="0"/>
        </restriction>
    </onElement>
    <onElement xpath="//searchResult/item/paymentMethod"
name="paymentMethod">
        <restriction base="string">
            <enumeration value="AmEx"/>
            <enumeration value="CashInPerson"/>
            <enumeration value="CashOnPickup"/>
            <enumeration value="CCAccepted"/>
            <enumeration value="COD"/>
            <enumeration value="CODPrePayDelivery"/>
            <enumeration value="CustomCode"/>
            <enumeration value="Discover"/>
            <enumeration value="ELV"/>
            <enumeration value="Escrow"/>
            <enumeration value="LoanCheck"/>
            <enumeration value="MOCC"/>
            <enumeration value="Moneybookers"/>
            <enumeration value="MoneyXferAccepted"/>
            <enumeration value="MoneyXferAcceptedInCheckout"/>
            <enumeration value="None"/>
            <enumeration value="Other"/>
            <enumeration value="OtherOnlinePayments"/>
            <enumeration value="PaisaPayAccepted"/>
            <enumeration value="PaisaPayEscrowEMI"/>
            <enumeration value="Paymate"/>
            <enumeration value="PaymentSeeDescription"/>
            <enumeration value="PayPal"/>
            <enumeration value="PersonalCheck"/>
            <enumeration value="PostalTransfer"/>
            <enumeration value="PrePayDelivery"/>
            <enumeration value="ProPay"/>
            <enumeration value="VisaMC"/>
        </restriction>
    </onElement>
    <onElement xpath="//searchResult/item/sellingStatus/timeLeft" name="timeLeft">
        <restriction base="string">
            <pattern value="P[0-9]{2}DT[1-2]?[0-9]H[1-5]?[0-9]M[1-5]?[0-9]S"/>
        </restriction>
    </onElement>
    <onElement xpath="//searchResult/item/shippingInfo/shippingType" name="shippingType">
        <restriction base="string">
            <enumeration value="Calculated"/>
            <enumeration value="CalculatedDomesticFlatInternational"/>
            <enumeration value="Flat"/>
            <enumeration value="FlatDomesticCalculatedInternational"/>
            <enumeration value="Free"/>
            <enumeration value="FreePickup"/>
            <enumeration value="Freight"/>
            <enumeration value="FreightFlat"/>
            <enumeration value="NotSpecified"/>
        </restriction>
    </onElement>
</ServiceAssumptions>

```

4 Conclusion

We propose a simple format for the specification of service assumptions. Service assumptions specify what a service integrator expects from the services that he/she is exploiting, they contains restrictions on the XML elements of the responses from the services. These restrictions can be different from those speficied in service interface (WSDL), or are complementary parts that are specified informally in the documents related to the services.

The benefits of specifying service assumptions are three folds. First, using explicit service assumption specifications to check data received from services, this helps the integrator detecting and dealing with bad data properly. Second, putting together scattered service assumptions (and their related validating code elsewhere in the system) in a single place helps reducing maintenance cost and making it easier to check for completeness. And finally, specifying service assumptions in a high-level language, e.g. XML, separately from code, this allows to evolve them easily at runtime, no code modification and recompilation is needed.

References

[1] Simple object access protocol (soap). http://www.w3.org/standards/techs/soap. Accessed December 2010.

[2] Web service description language (wsdl). http://www.w3.org/TR/wsdl20. Accessed December 2010.

[3] Xml schema. http://www.w3.org/XML/Schema. Accessed December 2010.

[4] Xml path language (xpath). http://www.w3.org/TR/xpath/, November 1999. Accessed December 2010.

[5] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, UNIVERSITY OF CALIFORNIA, IRVINE, 2000. Chapter 5 - Representational State Transfer (REST).