Restructuring WSDL

A Technique for Effective Service Similarity Detection

Doug Martin, James R. Cordy ● Queen's University ● Kingston, Ontario
Leho Nigul, Joanna Ng ● IBM CAS ● Toronto, Ontario

Introduction

A challenge for the Smart Internet will be the automated tagging of equivalent or similar services, both in terms of domain semantics and service protocols, in support of efficient discovery and selection of relevant alternative services for the current matters of concern. Code similarity detection is an established technique that can be brought to bear on this problem if service descriptions can be partitioned into appropriate units for comparison. Unfortunately, specifications written in Web Service Description Language (WSDL) are poorly structured for this purpose, with relevant information for each service operation scattered widely over WSDL service descriptions. In this work we describe a first step in leveraging code similarity techniques to identify and tag similarities in WSDL services. Using source transformation techniques, we describe a method for reorganizing WSDL descriptions such that they are both more human readable and better suited to analysis by similarity detection tools.

Methodology

HotelReservationExample.wsdl

```
<?xml version="1.0"?>
<definitions name="HotelReservationService"
             targetNamespace="http://myhotel.com/reservationservice.wsdl"
             xmlns:tns1="http://myhotel.com/reservationservice.xsd"
             xmlns:xsd1="http://myhotel.com/reservationservice.xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
        <schema targetNamespace="http://myhotel.com/reservationservice.xsd"
                xmlns="http://www.w3.org/2000/10/XMLSchema">
            <element name="GetAvailableRoomsRequest">
                <complexType>
                    <sequence/>
                </complexType>
            </element>
            <element name="GetAvailableRoomsResponse">
                <complexType>
                    <sequence>
                        <element name="room" minOccurs="0" maxOccurs="unbounded" type="tns1:Room"/>
                    </sequence>
                </complexType>
            </element>
            <element name="ReserveRoomRequest">
                <complexType>
                    <sequence>
                        <element name="payment" type="tns1:Payment"/>
                        <element name="room" type="tns1:Room"/>
                    </sequence>
                </complexType>
            </element>
            <element name="ReserveRoomResponse">
                <complexType>
                    <sequence />
                </complexType>
            </element>
            <element name="RoomNotAvailableException">
                <complexType>
                    <sequence />
                </complexType>
            </element>
            <complexType name="Room">
                <sequence>
                    <element name="roomID" type="xsd:int"/>
                    <element name="numBeds" type="xsd:int"/>
                    <element name="isSmoking" type="xsd:boolean"/>
                </sequence>
            </complexType>
            <complexType name="Payment">
                <sequence>
                    <element name="ccNumber" type="xsd:int"/>
                    <element name="cardHolder" type="xsd:string"/>
                    <element name="expiryDate" type="xsd:date"/>
                    <element name="PIN" nillable="true" type="xsd:int"/>
                </sequence>
            </complexType>
        </schema>
    </types>

    <message name="GetAvailableRoomsRequest">
        <part name="body" element="xsd1:GetAvailableRoomsRequest"/>
    </message>

    <message name="GetAvailableRoomsResponse">
        <part name="body" element="xsd1:GetAvailableRoomsResponse"/>
    </message>

    <message name="ReserveRoomRequest">
        <part name="body" element="xsd1:ReserveRoomRequest"/>
    </message>

    <message name="ReserveRoomResponse">
        <part name="body" element="xsd1:ReserveRoomResponse"/>
    </message>

    <message name="RoomNotAvailableException">
        <part name="body" element="xsd1:RoomNotAvailableException"/>
    </message>

    <portType name="HotelReservationServicePortType">
        <operation name="GetAvailableRooms">
            <input message="tns1:GetAvailableRoomsRequest"/>
            <output message="tns1:GetAvailableRoomsResponse"/>
        </operation>
        <operation name="ReserveRoom">
            <input message="tns1:ReserveRoomRequest"/>
            <output message="tns1:ReserveRoomResponse"/>
            <fault message="tns1:RoomNotAvailableException"/>
        </operation>
    </portType>

    <binding name="HotelReservationServiceSoapBinding" type="tns:HotelReservationServicePortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetAvailableRooms">
            <soap:operation soapAction="http://myhotel.com/GetAvailableRooms"/>
            <input name="GetAvailableRoomsRequest">
                <soap:body use="literal"/>
            </input>
            <output name="GetAvailableRoomsResponse">
                <soap:body use="literal"/>
            </output>
        </operation>
        <operation name="ReserveRoom">
            <soap:operation soapAction="http://myhotel.com/ReserveRoom"/>
            <input name="ReserveRoomRequest">
                <soap:body use="literal"/>
            </input>
            <output name="ReserveRoomResponse">
                <soap:body use="literal"/>
            </output>
            <fault name="RoomNotAvailableException">
                <soap:body use="literal"/>
            </fault>
        </operation>
    </binding>

    <service name="HotelReservationService">
        <documentation>A hotel reservation service</documentation>
        <port name="HotelReservationServicePort" binding="tns:HotelReservationServiceSoapBinding">
            <soap:address location="http://myhotel.com/reservationservice"/>
        </port>
    </service>

</definitions>
```

Restructured List of Operations

```
<source file="HotelReservationExample.wsdl" startline="77" endline="80">
    <operation name="GetAvailableRooms" >
        <input message="tns1:GetAvailableRoomsRequest">
            <message name="GetAvailableRoomsRequest">
                <part name="body" element="xsd1:GetAvailableRoomsRequest">
                    <element name="GetAvailableRoomsRequest"/>
                </part>
            </message>
        </input>
        <output message="tns1:GetAvailableRoomsResponse">
            <message name="GetAvailableRoomsResponse">
                <part name="body" element="xsd1:GetAvailableRoomsResponse">
                    <element name="GetAvailableRoomsResponse">
                        <element name="rooms" minOccurs="0" maxOccurs="unbounded" type="tns1:Room">
                            <element name="roomID" type="xsd:int"/>
                            <element name="numBeds" type="xsd:int"/>
                            <element name="isSmoking" type="xsd:boolean"/>
                        </element>
                    </element>
                </part>
            </message>
        </output>
    </operation>
</source>

<source file="HotelReservationExample.wsdl" startline="81" endline="85">
    <operation name="ReserveRoom" >
        <input message="tns1:ReserveRoomRequest">
            <message name="ReserveRoomRequest">
                <part name="body" element="xsd1:ReserveRoomRequest">
                    <element name="ReserveRoomRequest">
                        <element name="payment" type="tns1:Payment"/>
                        <element name="ccNumber" type="xsd:int"/>
                        <element name="cardHolder" type="xsd:string"/>
                        <element name="expiryDate" type="xsd:date"/>
                        <element name="PIN" nillable="true" type="xsd:int"/>
                    </element>
                    <element name="room" type="tns1:Room">
                        <element name="roomID" type="xsd:int"/>
                        <element name="numBeds" type="xsd:int"/>
                        <element name="isSmoking" type="xsd:boolean"/>
                    </element>
                </element>
            </part>
        </message>
        <output message="tns1:ReserveRoomResponse">
            <message name="ReserveRoomResponse">
                <part name="body" element="xsd1:ReserveRoomResponse">
                    <element name="ReserveRoomResponse"/>
                </part>
            </message>
        </output>
        <fault message="tns1:RoomNotAvailableException">
            <message name="RoomNotAvailableException">
                <part name="body" element="xsd1:RoomNotAvailableException">
                    <element name="RoomNotAvailableException"/>
                </part>
            </message>
        </fault>
    </operation>
</source>
```

1. Extract the operations from the `<portTypes>` section which contain the operation's input, output, and faults. This will be the basic skeleton.
2. Find the `<message>` associated with each `<input>`, `<output>`, and `<fault>` and inject it into the corresponding tag.
3. Find the `<element>` associated with each `<message>` and inject it into the tag.
4. For each `<element>` in the newly constructed part, find the type declaration of its type, if one exists. Then take the elements of that type and insert them into the parent element. The same process is repeated recursively for all the elements of the type, until all elements in the `<part>` are native XML types (eg. string, int, etc.).

Conclusion

Early results from similarity detection tools show promise. When comparing operations from WSDL descriptions with no consolidation, the results were sparse and unusable. However, after our restructuring many interesting groups of similar operations were identified (e.g., operations related to employee information, shipping information, customer information, and so on).

WSDL descriptions of web services pose problems for finding similarities. We have developed a way to make descriptions both more human readable and amenable to code similarity techniques.