IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP5203

This paper appears in *Managing Modern Organizations Through Information Technology*, Proceedings of the 2005 Information Resources Management Association International Conference, edited by Mehdi Khosrow-Pour.

Modelling XML and Web Services Messages with UML

Samir El-Masri and Bhuvan Unhelkar

School of Computing & Info. Technology, University of Western Sydney, Locked Bag 1797, Penrith South DC NSW 1797, Australia

{s.elmasri, bhuvan@cit.uws.edu.au}

ABSTRACT

Web Services (WS) are rapidly replacing the existing distributed computing technologies such as DCOM, CORBA/IIOP and RMI. Extensible Markup language (XML) Web Services are based on XML and the related technologies such as Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Discovery, Description and Integration (UDDI); each of these technologies requires excellence in modelling, to enhance their quality and maintainability. In this paper, it is suggested that the Unified Modelling Language (UML) be used as a graphical modelling tool to model XML web services and SOAP messages between service providers and client applications.

INTRODUCTION

Web Services represent the next major chapter of online computing that has enabled seamless integration of application services across the Internet. We believe WS is the cornerstone towards building a global distributed information system, in which most of the individual applications will take part. The centre for that global system will obviously be the WS. As there is no place today for a stand-alone computer, in the near future, there will be no place to a stand-alone application. Therefore, building a powerful application whose capability is not limited to local resources will unavoidably require interacting with other partner applications through WS. The Internet has been the revolution in networking and XML WS might be the one in software.

The strengths of WS come from the fact that WS use XML technologies in connecting business applications based on various computers and locations with various languages and platforms. However, XML document and XML Schema are low-level languages and software and system developers in particular analysts and designers need a high-level graphical modelling language to model the system before the programmers start coding. On the other hand UML (Booch, Rumbaugh, & Jacobson, 1999) became the standard modelling language in the modern object oriented software engineering. Therefore it will be a great idea if these two new technologies XML and UML can work together. Recently, some researchers were interested in mapping XML Schema to class diagram (Carlson, 2001) (Routledge, Bird, & Goodchild, 2002) (Malik, 2003). There have been also some works in XML and database mapping by Bohme and Rahm (2001)

In this paper, a simple hospital XML document is given, and then a hospital XML Schema is developed. Using UML, a class diagram for XML Schema is modelled showing the mapping between the class association in the model and the elements relationships in the XML Schema. The mechanism of XML WS have been briefly explained with the related technologies SOAP, WSDL and UDDI in a general sequence diagram which shows how a client application uses and interacts through WS using SOAP messages with provider services. At the end, SOAP and SOAP-RPC messages based on our hospital model and XML document are created. The messaging mechanism is successfully modelled with a sequence diagram.

XML

We believe that the starting point for all-good modelling in Web Services is XML. XML is a generalised markup language or metalanguage that is commonly used to transfer data and information across varying platforms and domains (Carlson, 2001). XML became the standard markup language and this is because it is extensible and it contains data in its document with no information on presentation, unlike HTML, which is inextensible, and contains data with information on how to be presented using HTTP (Hyper Text Transfer Protocol). Therefore, the first thing that needs to be modelled, when modelling Web Services is XML. In this paper, XML and its characteristics are briefly discussed, followed by the usage of UML in modelling XML.

XML is a simplified version of SGML (Standand Generalised Markup Language), on which HTML (HyperText Markup Language) is based. HTML has its own defined tags, which cannot be modified. On the contrary, XML allows the user to choose their own tags and elements depending on their need. HTML is a presentation language viewed by humans via browsers. XML is a data carrier documents and is independent of any presentation. Using a related technology like XSLT (Extensible Style Sheet Transformation), which in turn is XML document, XML document can be viewed with different format via Internet browsers, PDA or mobile phones, or can be transferred to another XML document. Those features make XML represents in the eyes of the big IT companies like Microsoft, IBM and Sun Microsystems a brilliant future as a common language and a medium to exchange data independently of the languages and the platforms used by applications. XML is dramatically and rapidly changing the technology and many believe that XML is the next revolution in the technology. XML can be stored in database as nature XML databases or can be stored in Object-Oriented or relational databases using some adaptation or mapping tools. As relational databases dominate the market and they work very well, many tools and techniques have been developed to connect with the relational databases in both directions that are converting XML documents to the existing database and converting the database to XML documents. The simplicity of XML document makes it the right choice for the Web Services (WS), which will be explained in more details in subsequent sections.

Listing 1 shows an example of XML document, which represents information from the hospital domain, including departments, doctors and patients.

Each XML document must start with the first line <?xml version=”1.0” ?> which shows that it is an XML document and its version as an attribute, it can also contain other attributes. It should contain also one root element, which is, in our case, <HOSPITAL>. The element <HOSPITAL> can contain other elements like <NAME>, <ADDRESS> which in turn contains <STREET>, <CITY> and <STATE> elements. It should be noted that the element <NAME> within the element <HOSPITAL> represents the name of the hospital, but <NAME> in the element <DOCTOR> is the name of the doctor. HREF and ID are used in this document to refer to the address element outside the hospital element. XML needs in most cases a schema to define the structure and the data types in the document.

Listing 1. Hospital XML Document

```
<?xml version="1.0" ?>
< HOSPITAL>
      <NAME>Westmead private</NAME>
      <ADDRESS>
            <STREET> 90 Burford St.</STREET>
            <CITY>Westmead</CITY>
            <STATE>NSW</STATE>
      </ADDRESS>
      <DEPARTMENT NAME="Maternity">
            <HEADDOCTOR>
                  <NAME>Gabriel Saguet</NAME>
                  <SPECIALITY>Gynaecologist</SPECIALITY>
                  <ADDRESS HREF="#address1"/>
            </HEADDOCTOR>
            <PATIENT ID="5">
                  <NAME>Mary Pelorson</NAME>
                  <ROOM LEVEL="3" NUMBER="10"/>
                  <ADDRESS HREF="#address2"/>
            </PATIENT>
            <PATIENT ID="6">
                  …
            </PATIENT>
            .
      </DEPARTMENT>
      .
      <DEPARTMENT NAME="CHILDREN">
            …
      </DEPARTMENT>
</HOSPITAL>
<ADDRESS ID="address1">
      <STREET> 20 George St.</STREET>
      <CITY>Sydney</CITY>
      <STATE>NSW</STATE>
</ADDRESS>
<ADDRESS ID="address2">
      <STREET> 20 George St.</STREET>
      <CITY>Sydney</CITY>
      <STATE>NSW</STATE>
</ADDRESS>
```

XML SCHEMA

In order to be able to recognise the meaning and the data types of every element and sub-elements by any application software used by external clients who are interested in interacting with the hospital system, an XML schema should be created (Walmsley, 2002). Two kind of schema can be used: XML DTD (Data Type Document) or XML Schema with Capital "S" which is in turn an XML document. XML Schema is more flexible and has many advantages over DTD. Listing 2 shows the XML Schema for the hospital document shown in Listing 1.

The first line in Listing 2 represents a namespace xsd referring to its URI. The line 2 and 3 show the complex data type of the hospital that contains name, address type and department type that in turn have their own definitions. More interpretations will be found while XML Schema is modelled in the next section.

MODELLING XML SCHEMA WITH UML

Having discussed the characteristics of XML with the two listings, we now consider how we will be able to model the XML Schema using UML. Figure 1 shows the UML class diagram modelling the XML Schema in listing 2.

In general, XML Schema can be represented as a class or a group of classes called class diagram, while the XML document represents an instance of the class or the class diagram.

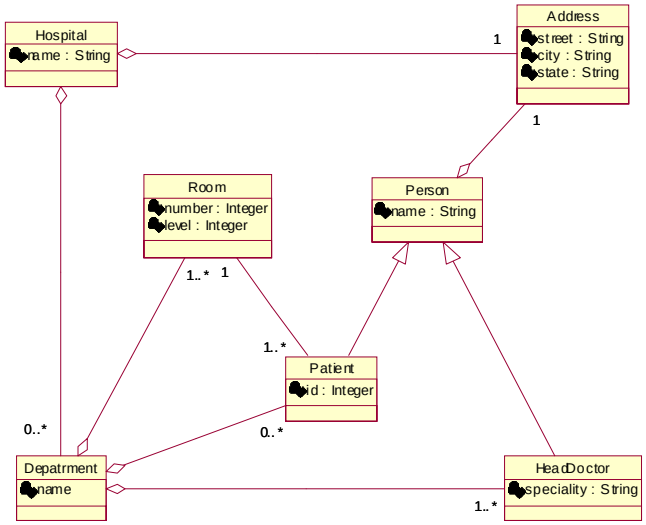
In Figure 1, a hospital class is modelled as aggregating two other classes Address and Department, as in Listing 2; HOSPITAL element contains DEPARTMENT and ADDRESS elements. The association between Room and Patient classes are represented in the Schema by (ref) and this is to distinguish between association and aggregation relationships. Although in Figure 1, Department class aggregates Room class in the model, this relationship cannot be seen in the XML Schema (Listing 2). This is because in this context, it is not of interest to show all the rooms that belong to each department; instead room should be associated with each patient. This would explain that one UML class model could be translated using different XML Schemas to different XML documents depending on the context. The inheritance in Figure 1 is represented in the Schema by special element <xsd: extension base=…>, and the multiplicities are translated by "minOcurrs" and "maxOcurrs" as addi-

Listing 2. Hospital XML Schema

```
<xsd:schema xmlns:xsd="http://www.W3.org/2001/XMLSchema">
<xsd:element name="HOSPITAL" type="HOSPITAL_TYPE"/>
<xsd:complexType name="HOSPITAL_TYPE">
      <xsd:sequence>
            <xsd:element name ="NAME" type="xsd:string"/>
            <xsd:element name="ADDRESS" type="ADDRESS_TYPE" />
            <xsd:element name="DEPARTMENT" type="DEPARTMENT_TYPE"
                              minOccurs="0"     maxOccurs="unbounded"/>
      </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="DEPARTMENT_TYPE">
      <xsd:sequence>
            <xsd:element name="HEADDOCTOR" type="HEADDOCTOR_TYPE"
                              minOccurs="1" maxOccurs="unbounded"/>
            <xsd:element name="PATIENT" type="PATIENT_TYPE"
                              minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="NAME" type="xsd:string"/>
</xsd:complexType>
<xsd:element name="PERSON" type="PERSON_TYPE"/>
<xsd:complexType name="PERSON_TYPE">
      <xsd:sequence>
            <xsd:element name="NAME" type="xsd:string"/>
            <xsd:element name="ADDRESS" type="ADDRESS_TYPE" />
      </xsd:sequence>
</xsd:cmplexType>
<xsd:complexType name="HEADDOCTOR_TYPE">
      <xsd:complexContent >
            <xsd:extension base="PERSON_TYPE">
                  <xsd:sequence>
                        <xsd:element>name="SPECIALITY"   type="xsd:string"/>

                  </xsd:sequence>
            </xsd:extension>
      <xsd:complexContent >
</xsd:complexType>
<xsd:complexType name="PATIENT_TYPE">
      <xsd:complexContent >
            <xsd:extension base="PERSON_TYPE">
                  <xsd:sequence>
                        <xsd:element name="NAME" type="xsd:string"/>
                        <xsd:element ref="ROOM_TYPE"/>
                  </xsd:sequence>
                  <xsd:attribute name="ID" type="xsd:int"/>
            </xsd:extension>
</xsd:complexType>
<xsd:complexType name="ROOM_TYPE">
      <xsd:attribute name="LEVEL" type="xsd:int"/>
      <xsd:attribute name="NUMBER" type="xsd:int"/>
</xsd:complexType>
<xsd:complexType name="ADDRESSE_TYPE" maxOccurs="1" use="required"/>
      <xsd:sequence>
            <xsd:element name="STREET" type="xsd:string"/>
            <xsd:element name="CITY" type="xsd:sting"/>
            <xsd:element name="STATE" type="xsd:sting"/>
      </xsd:sequence>
</xsd:complexType>
```

Figure 1. UML Class Diagram Model for the Hospital XML Schema in Listing 2



tional attributes. The default value for "minOccurs" where it is not mentioned is "1" where "unbounded" value means many. There is also another attribute called "use" with "required" value, this implies that the existence of this element (class) is mandatory. It should be pointed out that the actual XML document could be modelled the same way as in Figure 1 but instead of having class diagram, object diagram will be used. More details about XML and XML Schema can be found in the literature.

WEB SERVICES

The UML modelling of XML is not valuable unless it is used in some applications. The most successful use of XML is its role in the XML WS. Earlier technologies like DCOM, CORBA/IIOP and RMI have been used in integrating applications. These technologies could make two different applications based on different languages and platforms communicate with one another. The problem is that these technologies are complex and expensive to implement and they have to be implemented separately for each application. On the contrary, the XML WS that are based on SOAP messaging are simple and easy to deploy and it can be used by any application. XML WS and SOAP use the ubiquitous Internet Protocol HTTP as transport protocol, which makes it even more popular. The WS are usually published through UDDI. UDDI are similar to white and yellow pages of the telephone directory. Once the service is found, WSDL, which could be a part of UDDI, describes it to the public client.

WSDL describes all the services, which are practically operations or functions available to use, the ports, messages and protocols. In addition it should contain the XML Schema explaining the structure and the data type of the documents. It specifies in detail the parameters you should send in the message heading for the WS and the expected response message if any. In fact WSDL describe the structure of the message that should be sent and received by the client application. WSDL is an XML document, which could be generated automatically by some tools like Microsoft .NET framework. Although service providers could communicate with the client applications directly, there is a strong need to register with WS which make them exposed to more customers and easier to find (Figure 2).

The registration could be free or could cost some money to the providers and the clients. Service providers and clients in most cases should register to the WS.

It is shown in Figure 3, how the user and its application would interact with the service providers through WS. The client application represented by Browser Client sends a message requesting for a specific service.

The WS which plays the role of proxy class in the client application looks for the right services among all the registered service providers, once the services are found, the WS request the services from the

Figure 2. Discovering WS, Developing Application Trough WSDL and Registering by Application Developer

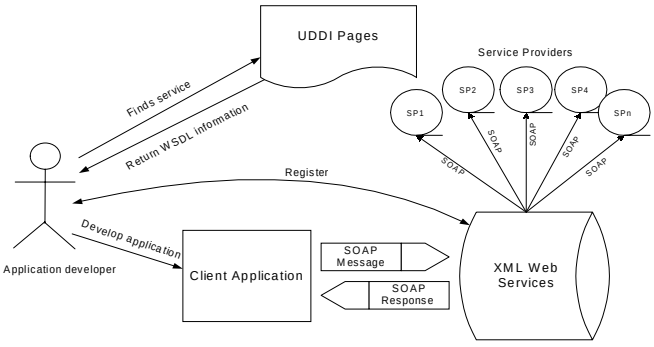
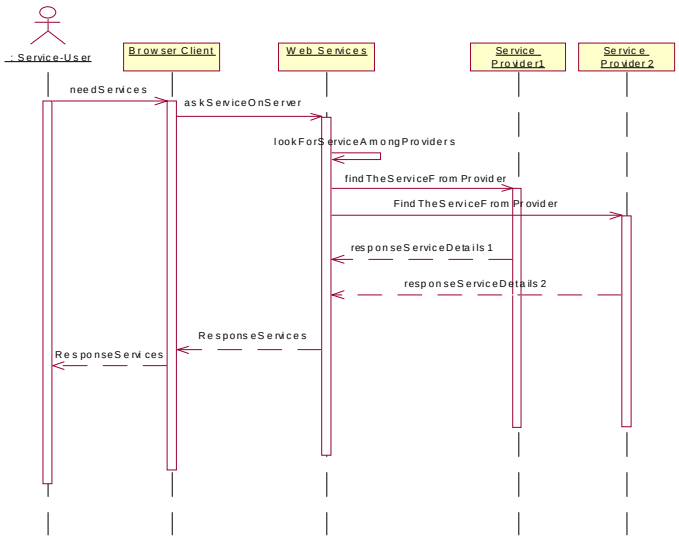


Figure 3. Sequence Diagram for Typical Exchange Messages with Web Services



providers. The providers send back the results to the WS, which in turn pass it on to the client application. It should be noted that all the messages of the interaction between client application, WS and service providers are SOAP message based carried over HTTP. Figure 3 shows only two service providers, as an example however, in a real life application, there could be thousands.

Although several protocols can be used to exchange data with the web services, the most common protocol, which became the standard protocol, is SOAP.

SOAP

As WSDL, SOAP is an XML document that contains an envelope as the root element representing the message and which contains an optional header element and a body element. Listing 3 shows an example of SOAP message that simply sends an XML document to another application or to WS. The document contains in the message body the **name** (Gabriel Saguet) and **speciality** (Gynaecologist) of the head doctor of the maternity department in Westmead Private Hospital (WPH). The department and the hospital names are provided in the message header. This one-way message will be sent with no request for any response.

As it is shown, "soap", "h" and "m" are namespaces with their appropriate URI. The role of this SOAP message is to send an XML document to other parties from our original hospital document, however, in most cases SOAP message can represent a call of a remote method based in WS that will ultimately call another methods from service provider applications as presented in the following section.

SOAP-RPC

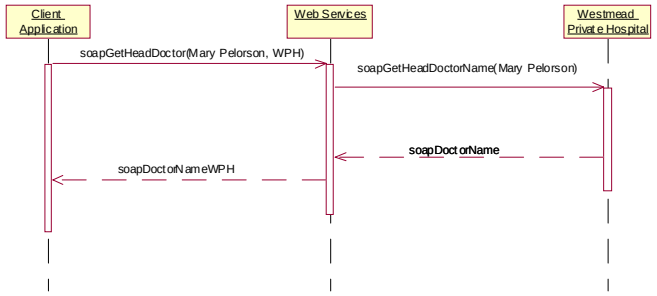
SOAP message becomes more attractive when it is associated with RPC (Remote Procedure Call) especially to those who suffer from the complexity of CORBA and DCOM. A client application can send SOAP message containing a call and parameters for an operation (method) that belongs to some service providers. The WS passes on that call to the operation at the provider system. The operation returns response carried back to the client by WS.

Through UDDI and WSDL, the client application knows about all the services and methods provided by service providers through WS. An example of a specific method that requires parameters and that returns value is presented. The following method *soapgetHeadDoctor()* has two

Listing 3. XML Document SOAP Message

```
<soap:Envelope xmlns:soap="http://schema.xmlsaop.org/soap/envelope/"
      soap:encodingStyle="schemas.xmlsoap.org/soap/encoding">
    <soap:Header>
      <h:hospital xmlns:h="http://www.wmp.com.au/Header">Westmead private
     hospital</h:hospital>
     <h:department
     xmlns:h="http://www.wmp.com.au/Header">Maternity</h:department>
    </soap:Header>
    <soap:Body>
     <m:headDoctor xmlns:m=http://www.wmp.com.au/doctor/>
           <m:NAME>Gabriel Saguet</NAME>
           <m:SPECIALITY>Gynaecologist</SPECIALITY>
     </m:headDoctor>
     </soap:Body>
</soap:Envelope>
```

Figure 4. Sequence Diagram for the Request/Response SOAP Message



parameters "name" and "hospitalname" of "string" type. It also returns a response value of "string" type.

string soapGetHeadDoctor(string name, string hospitalName)

The SOAP-RPC message is shown below in Listing 4.

The body message in Listing 4 calls the method *soapGetHeadDoctor* and provides two arguments that are the patient and the hospital names. The message should includes also the data types of the provided arguments using "xsi:type". This will help the end application where the method resides to understand the argument types.

When the message reaches the service provider application, the method will be called and it will return a response message to the sender application through WS. The response message is shown in Listing 5. It returns the doctor name "Gabriel Saguet" who is the head of the department in which the patient "Mary Pelorson" is treated. The return method is soapDoctorNameWPH().

The request and response message in Listing 4 and 5, are shown in the Sequence diagram in Figure 4.

In the sequence diagram (Figure 4), we can see three objects representing three classes from the Client Application, Web Services and the service provider Westmead Private Hospital. It is considered in this diagram that the three systems work like they are one application. This is where proxy classes are involved; the WS is represented in the client application by proxy class and the service provider (WPH) is represented in the WS by another proxy class. The client application calls the WS proxy class by the method name provided by the SOAP message (*soapGetHeadDoctor()*) and the WS proxy class in turn calls another method (*soapgetHeadDoctorName()*) that is carried by SOAP message, located at the service provider proxy class. The service provider proxy class returns a SOAP message (*soapDoctorName*) carrying response to the last method call to WS proxy class. The proxy class WS takes the return value and sends it as an ultimate SOAP message (*soapDoctorName*) carrying the final response message to the client application.

Listing 4. SOAP-RPC Request Message

```
<soap:Envelope xmlns:soap="http://schema.xmlsaop.org/soap/envelope/"
      soap:encodingStyle="schemas.xmlsoap.org/soap/encoding"
      xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance>
      xmlns:xsd=" http://www.w3.org/1999/XMLSchema">
    <soap:Body>
      <m: soapGetHeadDoctor  xmlns:m=http://www.wmp.com.au/methods/">
           <NAME xsi:type="xsd:string">Mary Pelorson</NAME>
           <HOSPITALNAME xsi:type="xsd:string">Westmead
Private</HOSPITALNAME>
      </m: soapGetHeadDoctor>
    </soap:Body>
</soap:Envelope>
```

Listing 5. SOAP-RPC Response Message

```
<soap:Envelope xmlns:soap="http://schema.xmlsaop.org/soap/envelope/"
      soap:encodingStyle="schemas.xmlsoap.org/soap/encoding"
      xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance>
      xmlns:xsd=" http://www.w3.org/1999/XMLSchema">
    <soap:Body>
      <m:soapDoctorNameWPH  xmlns:m=http://www.wmp.com.au/methods/">
           <NAME xsi:type="xsd:string">Gabriel Saguet</NAME>
      </m: soapDoctorNameWPH >
    </soap:Body>
</soap:Envelope>
```

RESEARCH APPROACH

Currently, our approach to verifying the ability of the UML to model XML Schemas is based on the experimental work happening in our research centre. It is based on the 'action research' approach to verifying the hypothesis, and in our case, we are considering the "ability of the UML to model XML and related Web Services technologies", as the hypothesis. While the research itself has not fully evolved yet, our hope is that by considering the preliminary capability of UML to model XML, we will be able to create a comprehensive approach to modelling quality software applications that integrate various business applications by utilizing the Web Services. This paper is an attempt to stir the thinking in the direction of quality improvement of WS through excellence in UML-based modelling.

CONCLUSION AND FUTURE DIRECTION

In this paper, the capability of the UML to model XML document and XML Schema with class and object diagrams has been presented. The concept and the mechanism of Web Services and how they work in particular with

XML messages and other related technologies like SOAP, WSDL and UDDI have been explained. The Web Services technologies have been generally modelled using successful approach with UML sequence diagram. Since the scope of the paper is the new approach of modelling Web service SOAP messages, a practical SOAP message from the origin hospital XML document that sends data through an XML document without requesting any return response has been written.

The most important contribution in this paper was the successful modelling of a real SOAP-RPC request message and a real SOAP-RPC response message using UML sequence diagram.

Further research will be to refine our UML modelling approaches for XML, Web services and SOAP, and mainly to create new approaches to model WSDL and UDDI. WSDL and UDDI are both XML documents; therefore similar approaches might be used.

REFERENCES

Bohme, T., & Rahm, E. (2001). *Benchmarking XML Database Systems - First Experiences*. Paper presented at the High Performance Transaction Systems Workshop, Pacific grov, California, USA.

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The Unified Modeling Language*. Indianapolis, IN: Addison-Wesley.

Carlson, D. (2001). *Modeling XML Applications with UML, Practical e-Business Applications*. NJ: Addison-Wesley.

Malik, A. (2003). *Design XML schema using UML*, URL htpp://www-106.ibm.com/developerworks/xml/library/x-umlschem/, Accessed 16 June 2004

Routledge, N., Bird, L., & Goodchild, A. (2002). *UML and XML Schema.* Paper presented at the Australasian conference on Database technologies, Melbourne.

Walmsley, P. (2002). *Definitive XML Schema*. NJ: Prentice Hall.