A Simple Web Interface for Inspecting, Navigating, and Invoking Methods on Java and C# Objects

Carlos R. Jaimez-González

Departamento de Tecnologías de la Información,
Universidad Autónoma Metropolitana, Unidad Cuajimalpa,
Av. Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa, C.P. 05300, México, D.F.
cjaimez@correo.cua.uam.mx

Abstract. In recent years there has been an increase in the development of different applications that rely on web browsers as their main interface to users and developers. This paper introduces a simple interface that allows inspecting and navigating Java and C# objects through any XML-aware web browser. The web browser interface also allows displaying and executing all the methods that belong to a specific Java or C# object that has been persisted on the system. The web browser interface is part of a framework for distributed object programming in Java and C#, called Web Objects in XML (WOX).

Keywords. Web Browser Interface, Java Objects, C# Objects, Object Navigation, Method Invocation, Web Objects in XML.

1 Introduction

Recently, there has been an increase in the development of different applications that rely on web browsers as their main interface to users and developers. This paper introduces a simple interface that allows inspecting and navigating Java and C# objects through any XML-aware web browser. The web browser interface also allows displaying and executing all the methods that belong to a specific Java or C# object that has been persisted on the system. The web browser interface is simple to use.

The web browser interface presented in this paper is part of a framework created for distributed object programming, called Web Objects in XML (WOX) [1], which allows building distributed systems; it uses XML as the representation for the objects and messages interchanged [2]; and provides both synchronous and asynchronous communication between clients and servers [3]. The WOX framework has special features; some of them taken from two paradigms used to construct distributed systems: the object-based paradigm and the web-based paradigm, where Java and C# objects can be persisted by distributed or local applications.

The rest of the paper is organized as follows. Related work is presented in section 2. An overview of the WOX framework is given in section 3. Section 4 describes how to inspect objects through the web browser interface. Section 5 provides an explanation on how to execute methods on specific objects that have been persisted on the

Advances in Computer Science
Research in computing Science 81, 2014 pp. 133-143

system. Section 6 discusses how to navigate objects through the web browser interface. Finally, conclusions and future work are provided in section 7.

2 Related Work

This section describes some related work used to browse objects. Although there are existing tools to browse objects, none of them is incorporated to a framework for distributed object programming, such as the one presented in this paper.

The Portable Explorer of Structured Objects (PESTO) [4], is an integrated user interface that supports browsing and querying of object databases. It allows users to navigate in a hypertext-like fashion, following the relationships that exist among objects. In addition, PESTO allows users to formulate object queries through an integrated query paradigm that presents querying as a natural extension of browsing, call it query-in-place. This interface can be configured to different object databases.

In [5] the authors propose an interactional operator dedicated to navigation through time, which would allow visualizing a snapshot of a collection of objects at a given instant, or detecting and examining changes within object states. They also study how this operator can be integrated with the two main types of interactions given in visual object database browsers: navigation within a collection of objects, and navigation between objects via their relationships, where users can explore and navigate the states of a set of related objects.

The authors of [6] worked more on the definition of a language rather than the development of a tool for navigation. They created XQBE (XQuery By Example), a visual query language for expressing a large subset of XQuery in a visual form, inspired by QBE, a relational language initially proposed as an alternative to SQL, which is supported by Microsoft Access. According to the hierarchical nature of XML, XQBE's main graphical elements are trees. One or more trees denote the documents assumed as query input, and one tree denotes the document produced by the query. Similar to QBE, trees are annotated so as to express selection predicates, joins, and the passing of information from the input trees to the output tree.

SOPView+ [7] is an object browser that supports navigation of a large database by changing the base object. The base object is an object which is a basis for navigation; forward navigation is provided for the reference paths ahead of the base object and backward navigation for the ones behind it. SOPView+ allows users to change the base object along the reference hierarchy among a number of database objects; this makes it possible for them to explore a large database until they find objects of their interest on the limited screen space, solving the screen real estate problem.

The CORBA Object Browser [8] is a web browser that can be used to directly invoke methods on CORBA Objects using a specifically designed URI scheme. The URI for an object not only identifies the object but may also optionally include the name of the method to be invoked on the object and the parameters required. It has been implemented by extending the HotJava browser. The browser has a mechanism for supporting access to CORBA Objects that run on a secure ORB through the

CORBA Object Browser. Accessing secure objects through the browser requires the authentication with the remote ORB and may also need secure communication.

3 Web Objects in XML

This section provides a brief overview of the WOX framework, which combines features of distributed object-based systems and distributed web-based systems. Some of the features of this framework are presented in the following paragraphs.

WOX uses URLs to identify uniquely remote objects, following the principles of the Representational State Transfer [9]. This is an important feature because all objects are uniquely identified by their URL and can be accessed anywhere on the Web, either through a Web browser or programmatically.

This framework uses an efficient and easy-to-use serializer, called the WOX serializer [2], which is the base of the framework to serialize objects, requests, and responses exchanged between clients and servers. This serializer is a stand-alone library based on XML, which is able to serialize Java and C# objects to XML and back again. One of its main features is the generation of standard XML for objects, which is language independent and allows reaching interoperability between different object-oriented programming languages. At the moment, applications written in the Java and C# programming languages can interoperate.

WOX has a set of standard and special operations supported on remote and local objects. These operations include the request for remote references, static method invocations (web service calls), instance method invocations, destruction of objects, request for copies, duplication of objects, update of objects, upload of objects, and asynchronous method invocations. Some of the operations are described in [1]. The mechanism used by WOX in a method invocation is shown in Figure 1.

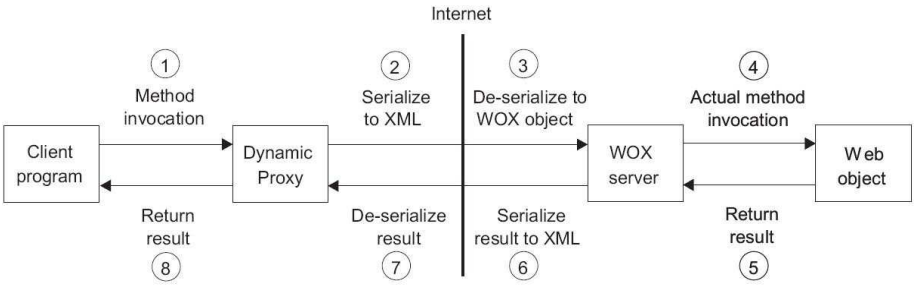

Fig. 1. A method invocation in WOX.

The detailed steps carried out in a method invocation in WOX are the following: 1) The WOX client program invokes a method on a remote reference (the way in which the client invokes a method on a remote reference is exactly the same as that on a local object, as far as the client program is concerned); 2) The WOX dynamic proxy takes the request, serializes it to XML, and sends it through the network to the WOX server; 3) The WOX server takes the request and de-serializes it to a WOX object; 4)

The WOX server loads the object and executes the method on it; 5) The result of the method invocation is returned to the WOX server; 6) The WOX server serializes the result to XML and either the real result or a reference to it is sent back to the client. The result is saved in the server in case a reference is sent back; 7) The WOX dynamic proxy receives the result and de-serializes it to the appropriate object (real object or remote reference); 8) The WOX dynamic proxy returns the result to the WOX client program.

From the WOX client program's point of view it just makes the method invocation and gets the result back in a transparent way. The WOX client libraries carry out the process of serializing the request and sending it to the WOX server; as well as receiving the result of the method invocation and de-serializing it.

The following sections introduce the web browser interface, which allows the inspection of objects through an XML-aware web browser, the execution of methods on objects, and the visualization of them with three different modes of operation: xml, html, and image. It can also navigate remote objects, which allows a client program retrieving child objects of a given root object, through their XML representation.

4 Inspecting Objects

Every Java or C# object that is persisted in the system, it is represented in XML, and can be inspected through a web browser by typing the URL that is assigned to it automatically by the system. Figure 2 shows a web browser with an example of the XML representation for an object of the *Lecturer* class. In order to inspect an object, the user types the object URL in the address bar of a web browser; for example: *http://carlosj:8080/WOXServer/WOXObject.jsp?id=622058786,* which is a URL representing an object with *id=622058786* on the system, which corresponds to an instance of the *Lecturer* class.

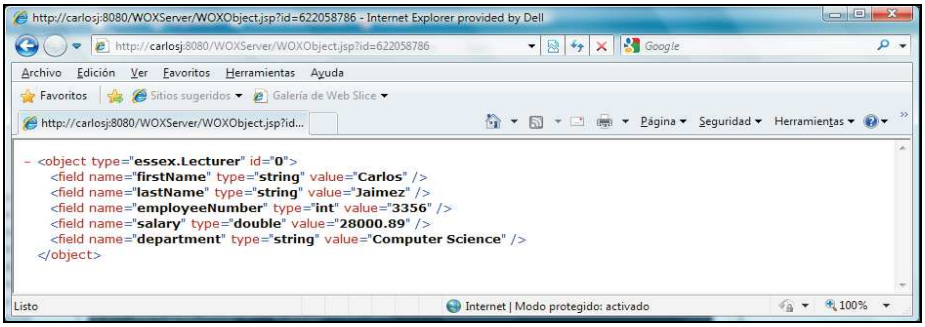

Fig. 2. Inspecting a Lecturer object through a web browser.

The XML representation of the *Lecturer* object is easy to understand, clean and compact. Every object is represented by an *object* XML element; and every field of an object is represented by a *field* XML element, with *name*, *type* and *value* attributes. The *Lecturer* object in Figure 1 has five attributes: *firstName* with *string* type;

lastName with *string* type; *employeeNumber* with *int* type; *salary* with *double* type; and *department* with *string* type. Figure 3 illustrates a diagram with the *Lecturer* class and an instance of it, which is represented in XML previously.

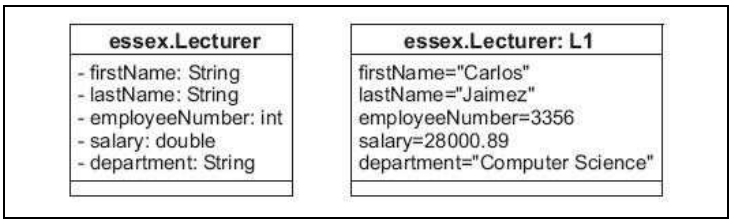

Fig. 3. The Lecturer class and an instance of it.

Alternatively, objects can be inspected through the server database, which holds all the objects that have been persisted on the system. Figure 4 shows an example of a server database, which can be accessed through the following URL: *http://carlosj:8080/WOXServer/WOXObjects.jsp*.

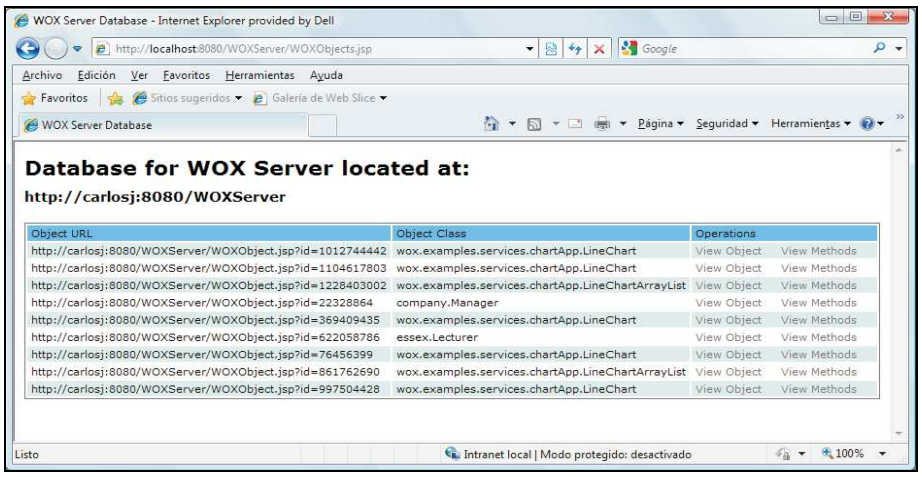

Fig. 4. A server database showing the objects persisted on the system.

The server database provides a table with three columns: the first column shows the URL of the object; the second column provides its implementation class; and the third column shows two hyperlinks. The *View Object* hyperlink can be used to request the specific object through its URL, and shows its XML representation on the web browser, as it was illustrated in Figure 1. The *View Methods* hyperlink is used to show the web browser interface with all the possible methods that can be invoked on the specific object. The invocation of methods is discussed in the following section.

The server database is an interface provided by the system, which allows access to persisted objects. This database can be seen as a directory for all the persisted objects,

where the user can request the object to inspect its XML representation or execute any of its methods available through the web browser interface.

5 Invoking Methods on Objects

Since every object can be identified uniquely through its URL, it is also possible to invoke methods on persistent objects through a web browser. As an example, the URL below shows an invocation of the *getDepartment* method, on an object of the *Lecturer* class, identified by the *622058786* object id: *http://carlosj:8080/WOXServer/WOXInvoke.jsp?objectId=622058786&method=getDepartment*.

The result of the method invocation is returned to the web browser as XML, which is the default mode of operation for the system, but it can also be returned as HTML (by specifying *mode=html* in the query string), in which case only the *string* would be returned. There is a special case in which the system can also return an image (by specifying *mode=image* in the query string), when the return type of a method is an array of bytes that represents an image.

Using a web browser, the system is also capable of invoking methods with parameters of primitive data types, but not with parameters of other data types, like user-defined classes. This way of operation through the web browser is similar to the way in which Apache Axis [10] allows invoking methods of classes. The main differences are that Axis, which is SOAP based [11], does not have the concept of an object, thus the methods are invoked as if they were static methods. Another important difference in this mode of operation is that Axis does not support package-qualified classes. In this respect, the system enables the invocation of methods on any web object (instance methods), and methods of any package-qualified class (static methods).

Alternatively, the system provides the browser interface with all the possible methods to invoke on a specific web object. This is similar to the browser front end for CORBA objects described in [8]. The web browser interface of the system can be accessed by typing the URL presented below, where it will present all the methods available for invocation on that object (in our example, it is a *Lecturer* object): *http://carlosj:8080/WOXServer/WOXInvoke.jsp?objectId=622058786*. Figure 5 shows this browser interface with three of the methods available for a *Lecturer* object. The web browser interface can also be accessed from the server database shown in Figure 4, by clicking the *View Methods* hyperlink of the desired object.

When clicking the *Invoke Method* button of the desired method, a query string is built internally with all the information needed for the method invocation. The request is sent to the system; the system will execute the method requested and generate an answer via an XML message with the result of the method invocation chosen; and finally this answer will be returned to the web browser (the return type of the method invocation is also specified in the web browser interface). The process of executing the method on the desired object is transparent to the user, since it is only needed to click the *Invoke Method* button to request the execution of the specific method, and wait for the answer.

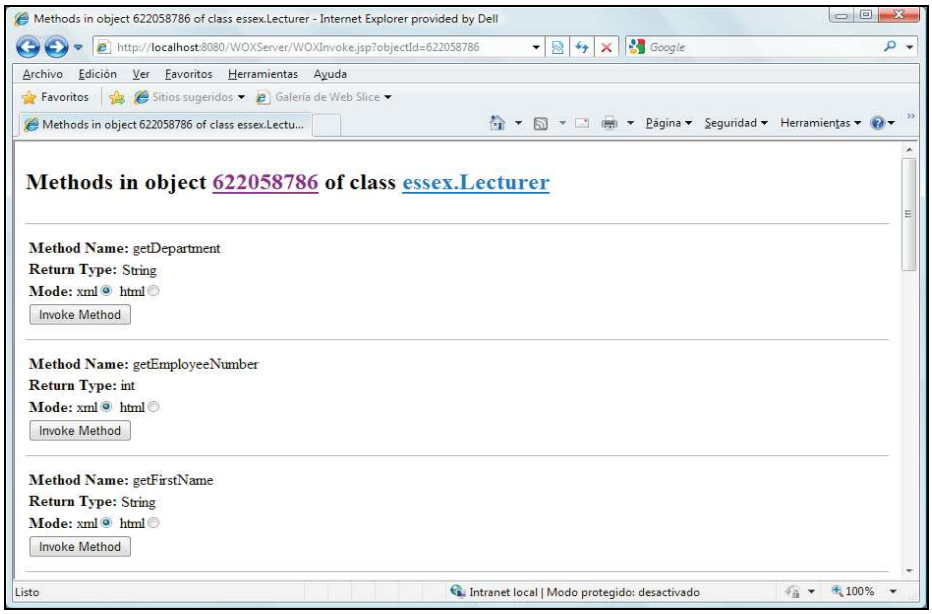

Fig. 5. Some methods available for a Lecturer object.

When the method to be invoked takes some parameters, the web browser interface provides the appropriate input boxes to enter the values to be sent with the method invocation. Figure 6 shows some other methods of a *Lecturer* object, which takes some parameters. It should be noticed that the input boxes shown in the web browser interface are generated automatically by the system, according to the data types of the parameters. Also notice that every parameter is named according to the order in which they appear in the method signature: *param1*, *param2*, etc.

In order to invoke the *setDepartment* method shown in Figure 6, the user must provide the *param1* parameter in the input box shown, which is of type *string*. When clicking the *Invoke Method* button of the *setDepartment* method, a query string is built with all the information needed for the method invocation on the specific object, which is the following: *http://carlosj:8080/WOXServer/WOXInvoke.jsp?objectId= 622058786&method=getDepartment&param1=Finance*.

The web browser interface provided by the system is a very convenient way of discovering and invoking methods on specific objects without needing a Java or C# client program.

6 Navigating Objects

The navigation of objects in the system allows a client program retrieving child objects of a persisted object. This is an important feature of this framework, which is very useful to retrieve only the child objects needed, instead of the entire object. This can also be carried out through a web browser. In this section we will use a different

object, since we want to show how to retrieve and navigate through an object which has child objects.

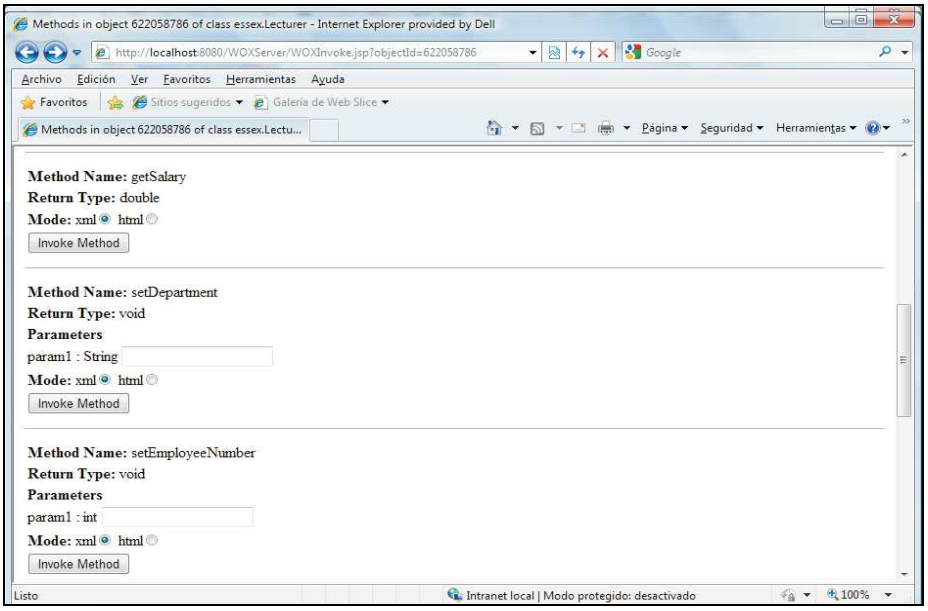

Fig. 6. Some methods with parameters available for a Lecturer object.

6.1 Retrieving an Object

This subsection shows how to retrieve an object. Assuming that the object to be retrieved can be accessed through the following URL: *http://carlosj:8080/WOXServer/WOXObject.jsp?objectId=622058786*. The XML representation of the object retrieved, which is a list of *Course* objects, is shown in Figure 7. It can be observed that the root object has three child objects identified by their *id* attributes with values 1, 2, and 3. The root object is identified by the *id* attribute with the value of 0.

6.2 Navigating an Object

The inner or child objects of a root object can be retrieved independently by specifying its *id* attribute in the URL of the specific object. For example, the following URL retrieves the *Course* object identified by *id="1"*: *http://carlosj:8080/WOXServer/WOXObject.jsp?objectId=622058786&id=1*.

The XML representation of the *Course* object after it has been retrieved is shown in Figure 8. It can be observed that only the child object specified has been retrieved. This is accomplished partly because of the XML representation of the objects, by using identifiers for every child object.

```
<object type="list" elementType="Object" length="3" id="0">
   <object type="Course" id="1">
      <field name="code" type="int" value="6756" />
      <field name="name" type="string" value="XML Technologies" />
      <field name="term" type="int" value="3" />
   </object>
   <object type="Course" id="2">
      <field name="code" type="int" value="9865" />
      <field name="name" type="string" value="O-O Programming" />
      <field name="term" type="int" value="2" />
   </object>
   <object type="Course" id="3">
      <field name="code" type="int" value="1134" />
      <field name="name" type="string" value="Web Programming" />
      <field name="term" type="int" value="2" />
   </object>
</object>
```

Fig. 7. XML representation of a list of Course objects.

Using this important feature, it is possible to retrieve only parts of a persisted object. This feature can be used through an XML-aware web browser where the object can be seen, but it can also be used programmatically from a Java or C# client application, using the framework for distributed object programming.

```
<object type="Course" id="1">
   <field name="code" type="int" value="6756" />
   <field name="name" type="string" value="XML Technologies" />
   <field name="term" type="int" value="3" />
</object>
```

Fig. 8. XML representation of a list of Course objects.

7 Conclusions and Future Work

This paper presented a simple web browser interface to inspect, navigate and invoke methods on Java and C# objects, which are persisted in the system. The web browser interface is part of Web Objects in XML (WOX), an interoperable framework for distributed object programming. It has been shown how objects can be inspected through a web browser; how to discover the methods to be invoked on them; and finally, how to navigate through objects in order to retrieve specific child objects. One of the key advantages of the web browser interface is that a user can inspect, navigate and execute methods on objects though any XML-aware web browser without needing any Java or C# program. The interface is intuitive and easy to use.

Further work is needed in order to allow the invocation of methods with parameters representing objects of user-defined classes, because at the moment parameters of methods can only be of primitive data types. Additional work is needed in this respect to provide an interface to upload this type of parameters. The fact of representing objects in XML is a great advantage, since they can be stored in simple XML files, which facilitates their management.

References

1. Jaimez-González, C., and Lucas, S., Implementing a State-based Application Using Web Objects in XML, In Proceedings of the 9th International Symposium on Distributed Objects, Middleware, and Applications (DOA 2007), Lecture Notes in Computer Science, Volume 4803/2007, pp. 577-594, Vilamoura, Algarve, Portugal, 25-30 November 2007.
2. Jaimez-González, C., Lucas, S., and López-Ornelas, E., Easy XML Serialization of C# and Java Objects. In Proceedings of the Balisage: The Markup Conference 2011, Balisage Series on Markup Technologies, Volume 7 (2011), doi:10.4242/BalisageVol7.Jaimez01, Montréal, Canada, 2-5 August 2011.
3. Jaimez-González, C., Lucas, S., and Luna-Ramírez, W., A Web Tool for Monitoring HTTP Asynchronous Method Invocations, In Proceedings of the 7th IEEE International Conference for Internet Technology and Secured Transactions (ICITST-2012), ISBN 978-1-908320-08-7, pp. 127-132, London, UK, 10-12 December 2012.
4. Carey, M., Haas, L., Maganty, V., Williams, J., PESTO: An Integrated Query/Browser for Object Databases, In Proceedings of the 22th International Conference on Very Large Data Bases, Mumbai, India, 3-6 September 1996.
5. Dumas, M., Daasi, C., Fauvet, M., Nigay, L., Pointwise Temporal Object Database Browsing, In Proceedings of the International Symposium on Objects and Databases, Sophia Antipolis, France, June 2000.
6. Braga, D., Capi, A., Ceri, S., XQBE (XQ uery B y E xample): A visual interface to the standard XML query language, ACM Transactions on Database Systems, 01/2005; 30(2):398-443, 2005.
7. Chang, S., Kim, H., SOPView+: an object browser which supports navigating database by changing base object, In Proceedings of the 21st International Conference on Computer Software and Applications Conference (COMPSAC 97), 1997.
8. Gupta D. Kumar, A. and P. Jalote. A browser front end for corba objects. In 10th International World Wide Web Conference, 2001.
9. Fielding, R., Architectural Styles and the Design of Network-based Software Architectures. Available at http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm, PhD thesis, USA, 2000. Last accessed in September 2014.
10. Apache Software Foundation. Web services - axis. Available at: http://ws.apache.org/axis/. Last access in September 2014.
11. World Wide Web Consortium. Latest soap version. Available at: http://www.w3.org/tr/soap/. Last access in September 2014.