Web System for Storing and Visualizing Web Objects in XML

José M. Hernández-Salinas, Carlos R. Jaimez-González, Betzabet García-Mendoza
Departamento de Tecnologías de la Información
Universidad Autónoma Metropolitana, Unidad Cuajimalpa
Mexico City, Mexico
Email: {cjaimez, bgmendoza}@cua.uam.mx

***Abstract*—Web Objects in XML (WOX) is a framework for creating object-based distributed applications, it supports the interoperability among different object-oriented programming languages, it uses XML as the format representation for objects, and it uses HTTP as its transport protocol. This paper presents a web system that was developed to complement the functionality of WOX, which allows the storage of distributed objects and the visualization of their state and methods. The web system has a repository, in which objects can be visualized and their methods can be executed through a web interface where the user can provide values for each of their parameters.**

Keywords—Web Objects in XML; visualization of objects, distributed objects, object storage

I. Introduction

The web system presented in this paper complements the Web Objects in XML (WOX) framework [1], which is used to create distributed object-based applications. WOX allows building distributed systems, it uses XML as a representation for the objects and messages exchanged [2], and it provides synchronous and asynchronous communication between clients and servers [3]. WOX has special features taken from two paradigms used to build distributed systems: the object-based paradigm and the web-based paradigm. WOX can store Java, C#, Python and PHP objects, which can be generated by local or distributed applications.

The rest of the paper is organized as follows. Related work is presented in section 2 with a comparative analysis of different systems that have a similar purpose to the web system described in this paper. Section 3 provides an introduction to the WOX framework. Section 4 describes the functionality of the web system, its structure and interface. Section 5 shows the web system in operation, in particular the storage of objects and the repository for visualizing objects are presented. Finally, section 6 presents the conclusions and future work.

II. Related work

This section describes the features and operation of some systems that are similar to the web system presented in this paper. The systems analyzed are the following: CORBAWeb [4], SopView+ [5], PESTO [6], CORBA Object Browser [7] and Apache Axis2 [8]. A comparison of these systems is presented at the end of the section, together with a brief description of the features that were taken into consideration.

A. CORBAWeb

It is a system [4] that acts as a gateway between the web and the Common Object Request Broker Architecture (CORBA). This system is known as a generic object browser, since its idea is to allow clients to view and invoke methods on any local or remote CORBA object, through a web browser. With this system a client can navigate through CORBA object links using dynamically generated URLs for each remote object.

CORBAWeb works through a web browser, in which a user can access and invoke methods on remote objects that reside in a server, for this, HTML forms are automatically generated from the Interface Definition Language (IDL), which allows the invocation of methods of any CORBA object. CORBAWeb receives the user's actions and translates them, accesses the remote object required to invoke the object's method, gets the result, and finally returns an HTML document containing the results of the invocation.

B. SOPView+

It is a project [5] of the Seoul National University, South Korea, developed in a UNIX environment, it uses the Motif widget tool to provide a graphical interface. This project aims to create an object browser and viewer, mainly for querying and managing object-oriented databases. With this system it is possible to explore the database in order to locate a desired object, retrieve its information and view it graphically. In addition, this tool visualizes the objects in a hierarchical way and allows navigation in large databases by changing the base object, which is an object that becomes the main node through which the navigation begins.

SOPView+ allows users to change the base object while searching for objects in the databases; in order to do this, an anchor is placed on the object. This makes it possible for users to explore objects in a large database more easily.

C. PESTO

This system [6] was created from the GARLIC project [9], whose objective is to build an information system capable of integrating data that resides in different database systems. The data can be queried through a language similar to SQL, which has been extended to include object-oriented features. The GARLIC project aims to provide a novel interface that offers the query and navigation of objects, called the Portable Explorer of Structured Objects (PESTO), which is the joint work between

IBM and the University of Wisconsin - Madison, where they work on the development of an interface for users.

PESTO offers its own interface in which it is possible to navigate over objects, it is designed to explore object databases and offers the possibility of making complex queries of objects in databases, through a language similar to SQL thanks to the query in place paradigm. This tool provides users with an interface that allows interactive navigation and consultation of the contents of GARLIC databases. Like SOPView+, PESTO allows users to move back and forth in the database; it allows to query and navigate through objects, which are displayed in a reference hierarchy that can be represented by connecting nodes of objects through links.

D. CORBA Object Browser

The purpose of this system [7] is to access CORBA objects directly from a web browser using a URI scheme; it allows browsing and invoking CORBA objects in the same way that users browse the Internet. In this tool it is possible to view and execute the methods of a specific object from a web browser; however the user has to use a prototype browser called HotJava Web Browser, which is no longer available.

The advantage of the HotJava Web Browser is that it had a mechanism to access CORBA objects that run on a secure Object Request Broker (ORB), through the CORBA Object Browser. Accessing secure objects through the browser required authentication with the remote ORB and also having a secure communication.

E. Apache Axis2

It is a web services engine [8] developed by Apache Software Foundation, which is interoperable, implemented in C++ and Java, in which interoperable and distributed applications can be created. Like WOX, Apache Axis2 is an open source, XML-based framework that uses SOAP for message exchange. Although Apache Axis2 works with objects, it does not save the state of the objects, so its methods are only invoked as if they were static methods. A particular feature of this tool is that although it offers the execution of methods on objects, it does not have an interface for its users, since it is necessary to call the objects through their URL.

F. Comparison Table

Table 1 shows a comparison of the systems analyzed in this section: S1) CORBAWeb, S2) SOPView+, S3) PESTO, S4) CORBA Object Browser, and S5) Apache Axis2. A tick indicates that the system has the feature, while a cross indicates that the tool does not have it. The features considered for the comparison are the following: open source, which refers to software that is made freely available and may be redistributed and modified; interoperability, which means that the system is capable of communicating on different platforms or programming languages; based on objects, which indicates that the system supports the use of remote objects; web services, which means that the system supports web services; use of XML, which indicates that the system uses XML as a means of communication between the client and the server; visualization of objects, which indicates that the system allows the visualization of objects graphically; visualization of attributes, which means that the system allows the visualization of the attributes that each object contains; execution of methods, which refers to the fact that the execution of methods belonging to an object is allowed through a web browser; web interface, which indicates that the system provides a web interface in which the user can view the methods of the remote objects; database management, which means that the system is able to navigate through object databases or repositories.

TABLE I. COMPARISON OF FEATURES OF THE ANALYZED SYSTEMS

Features	**S1**	**S2**	**S3**	**S4**	**S5**
Open source	✓	✗	✗	✓	✓
Interoperability	✓	✗	✗	✓	✓
Based on objects	✓	✓	✓	✓	✗
Web services	✓	✗	✗	✗	✓
Use of XML	✗	✗	✗	✗	✓
Visualization of objects	✓	✓	✓	✓	✗
Visualization of attributes	✗	✓	✓	✓	✗
Execution of methods	✓	✗	✗	✓	✓
Web interface	✓	✗	✗	✓	✗
Database management	✗	✓	✓	✗	✗

III. WEB OBJECT IN XML

This section provides a brief introduction to the WOX framework, which combines features of distributed object-based systems and distributed web-based systems. Some of the features of this framework are described.

WOX uses URLs to uniquely identify remote objects, following the principles of the Representation State Transfer (REST) architecture [10]. This is an important feature because all objects are uniquely identified by their URL and can be accessed from anywhere on the web, either through a web browser or programmatically.

WOX uses an efficient serializer, called WOX serializer [2], which is the basis of the framework for serializing objects, requests and responses exchanged between clients and servers. This serializer is an independent XML-based library, which is capable of serializing Java, C#, PHP and Python objects to XML and vice versa. One of its main features is the generation of standard XML for objects, which is independent of the programming language and allows interoperability between different object-oriented programming languages. Applications written in these programming languages can interoperate.

WOX has a set of standard and special operations that are applied on local and remote objects. These operations include requesting remote references, invoking static methods (web service calls), invoking instance methods, destroying objects, requesting copies, duplicating objects, updating and uploading objects, invoking asynchronous methods, among others. Some of these operations are described in [1]. The mechanism used by WOX in a method invocation is illustrated in Figure 1.

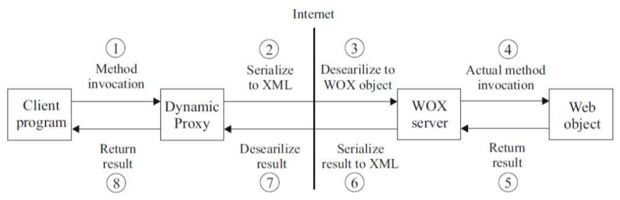


Fig. 1. Mechanism of remote method invocation in WOX.

The series of steps carried out in the invocation of a method in WOX is as follows: 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 way as if it invokes a method on a local object); 2) the WOX dynamic proxy takes the request, serializes it to XML and sends it over the network to the WOX server; 3) the WOX server takes the request and deserializes 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 it is returned to the client, either the actual result or a reference to it (the result is stored in the server in case a reference has been sent); 7) the WOX dynamic proxy receives the result and deserializes 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 point of view of the WOX client program, it only performs the invocation of the method and gets the result back transparently. 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 deserializing it. The following sections present the web system that allows to store and visualize WOX objects through a web browser.

IV. Analysis and Design

This section describes the functionality of the developed web system, its structure and interface.

A. Functionality

The functionality of the web system is described below, through a series of actions that can be carried out in it.

Access to a specific WOX object over the network. The web system allows its users to access a WOX object stored in the system from a unique URL generated by the same system. With this URL, users can directly access the object without having to enter the repository and search for their desired object among all the existing objects in the same repository.

Visualization of objects graphically. Users can visualize, through an interface, the attributes of a given WOX object. Two views are provided: the first is a table that represents the object and within it all its attributes are displayed regardless of its data type; the second is a view of the XML code that represents the object, which is displayed in a format that allows the user to understand each tag of the XML code.

Visualization of methods of any WOX object. Users can view all the methods that belong to a particular object, regardless of the parameters that each method requires to be invoked. Each method has a space to display the response that it returns when it is invoked.

Execution of any method belonging to the class of a WOX object. Through the web system it is possible to invoke any method of a WOX object; there is also a parameter verification to prevent the user from entering erroneous parameter values, this way it is ensured a correct invocation of a method. Furthermore, the system can return a response from the invocation of a method regardless of its data type. It is important to mention that for the web system to be able to execute a method on an object, it must have the class to which that object belongs.

Storage of WOX objects on the server. Users are able to upload WOX objects to the server for later manipulation, either as a parameter for the invocation of an object or simply to store it in the repository.

B. Structure

The web system consists of a dynamic web interface that allows users to access WOX objects hosted in a repository. The user can visualize each existing object in the repository in a graphical way, in addition to being allowed to execute the methods of each object through a web browser.

Figure 2 shows the navigation map, which represents the structure of visualization used in the web system.

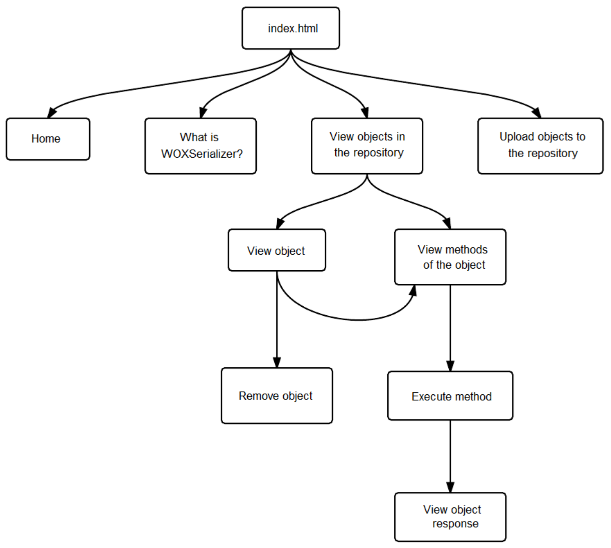
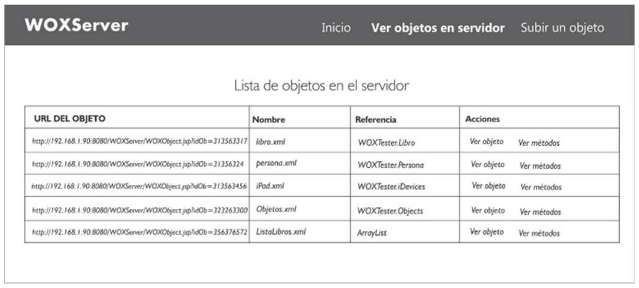


Fig. 2. Navigation map of the web system.

C. Interface Prototypes

Figure 3 shows the interface prototype for the list of objects stored in the server's repository, in which the following fields are displayed: object URL, which is a unique URL displayed to access directly a specific object existing in the repository; name, which displays the name of the XML file that represents the WOX object; reference, which shows the class to which each object belongs; actions, which provides two options to the user, the first option is to view the object's attributes graphically and the second option is to directly view and access the methods that the object contains.

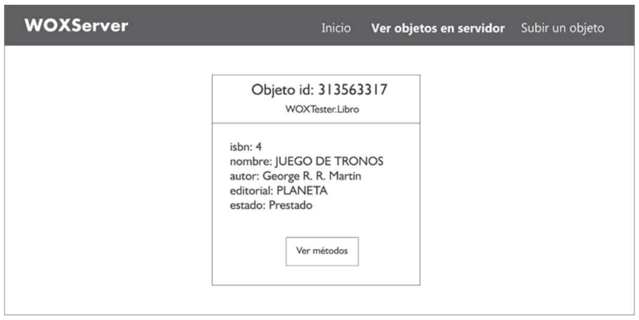


Lista de objetos en el servidor

URL DEL OBJETO	Nombre	Referencia	Acciones	
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=313563317	libro.xml	WOXTester.Libro	Ver objeto	Ver métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=31356324	persona.xml	WOXTester.Persona	Ver objeto	Ver métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=313563456	iPad.xml	WOXTester.iDevices	Ver objeto	Ver métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=323263300	Objetos.xml	WOXTester.Objects	Ver objeto	Ver métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=356376572	ListaLibros.xml	ArrayList	Ver objeto	Ver métodos

Fig. 3. Interface prototype for the list of objects stored in the repository.

Figure 4 shows the interface prototype that shows the graphical representation of a WOX object. The prototype includes a table with a header that contains the ID that makes the object unique and its class; in the body of the table is contained the object's attributes and there is a button to access the object's methods. In this example the object id is 313563317, and the attributes of the object are the following: isbn, name, author, editorial, and status.

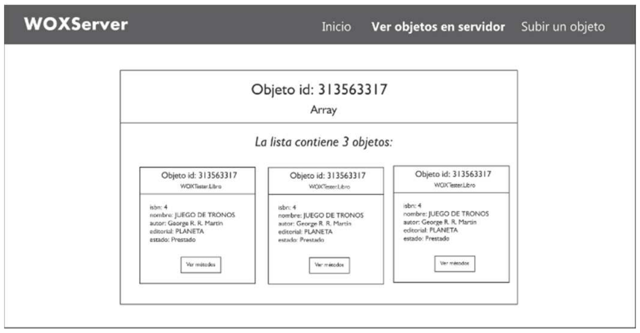


Objeto id: 313563317
WOXTester.Libro

isbn: 4
nombre: JUEGO DE TRONOS
autor: George R. R. Martin
editorial: PLANETA
estado: Prestado

Fig. 4. Interface prototype for the visualization of a WOX object.

Similarly, in Figure 5, the interface prototype shows the graphical representation of an object that can contain one or more objects within its attributes. In the case shown, it is an object of type array that contains three objects; the methods of each object can be accessed through its methods button. In this example there are three objects in the list; each object has its own attributes and values: isbn, name, author, editorial, and status of the object.



Objeto id: 313563317
Array

La lista contiene 3 objetos:

Objeto id: 313563317
WOXTester.Libro
isbn: 4
nombre: JUEGO DE TRONOS
autor: George R. R. Martin
editorial: PLANETA
estado: Prestado

Objeto id: 313563317
WOXTester.Libro
isbn: 4
nombre: JUEGO DE TRONOS
autor: George R. R. Martin
editorial: PLANETA
estado: Prestado

Objeto id: 313563317
WOXTester.Libro
isbn: 4
nombre: JUEGO DE TRONOS
autor: George R. R. Martin
editorial: PLANETA
estado: Prestado

Fig. 5. Interface prototype for the visualization of a list of WOX objects.

D. Interface

The home page of the web system is shown in Figure 3, where the menu has the following four options: 1) Home; 2) What is WOX serializer?; 3) View objects on the server; and 4) Upload object to the server.

Fig. 6. Home page of the web system.

V. Web System in Operation

This section presents the web system in operation. It describes how objects are stored in the web system and how they are visualized through the repository.

A. Storage of Objects

Figure 7 shows the operation of the web system to store WOX objects, through a series of steps that are carried out between client and server.

Step 1, the serialization of an object with the WOX serializer, for which any of the programming languages supported by WOX [11] can be used.

Step 2, the user proceeds to upload the object to the server by selecting the option Upload an object to the server from the main menu of the web system.

Step 3, the server is in charge of storing the object in the directory called UploadTemp of the web system repository.

Step 4, the system proceeds to verify the object; it is deserialized to verify that it does not contain errors or that it already exists and depending on the result.

Step 5.1, the object is moved to the trash if it had errors.

Step 5.2, the object is registered in the repository in case it is an object without errors.

Step 6, the user is informed of the result of storing the WOX object in the repository.

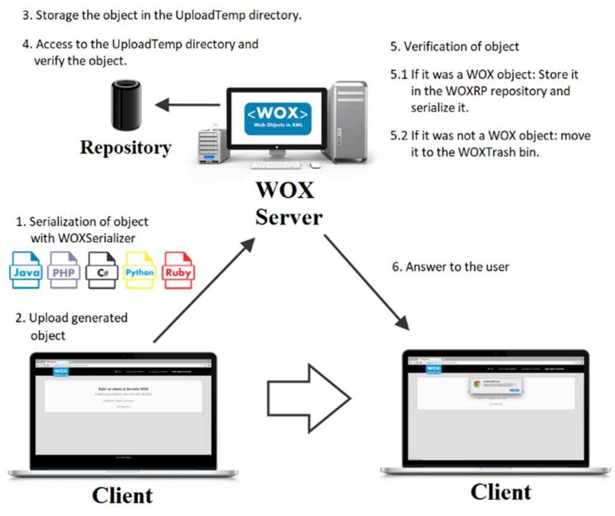

3. Storage the object in the UploadTemp directory.
4. Access to the UploadTemp directory and verify the object.
5. Verification of object
5.1 If it was a WOX object: Store it in the WOXRP repository and serialize it.
5.2 If it was not a WOX object: move it to the WOXTrash bin.
Repository
WOX Server
1. Serialization of object with WOXSerializer
Java
PHP
C#
Python
Ruby
2. Upload generated object
6. Answer to the user
Client
Client

Fig. 7. Storing a WOX object.

B. *Visualization of Objects*

Figure 8 shows the operation of the repository for visualizing WOX objects, through a series of steps that are carried out between client and server.

Step 1, the user accesses the repository by selecting the option View objects in the server from the main menu of the web system.

Step 2, the WOXRP.xml object is deserialized, which is a list of objects of the WOXObject class that contains the information of all the objects stored in the repository.

Step 3, once the WOXRP.xml object has been deserialized, the information of the registered objects can be accessed, so a table is generated that will contain the URL and class of each registered object.

Step 4, each registered object has two options, one to view the object and another to access its methods.

Step 5, the generated table is shown, so that the user can choose an object to be visualized.

In the interface of the web system to access the object repository, the user must select the option View objects in the server, which will take the user to the page shown in the screenshot of Figure 9.

The object repository has four columns: the URL of the object in the server, the name of the object, the remote reference to the object, and the actions that can be executed on the object (View object and View its methods). The View Object hyperlink displays the graphical representation of the object and its representation in XML, as shown in Figure 10.

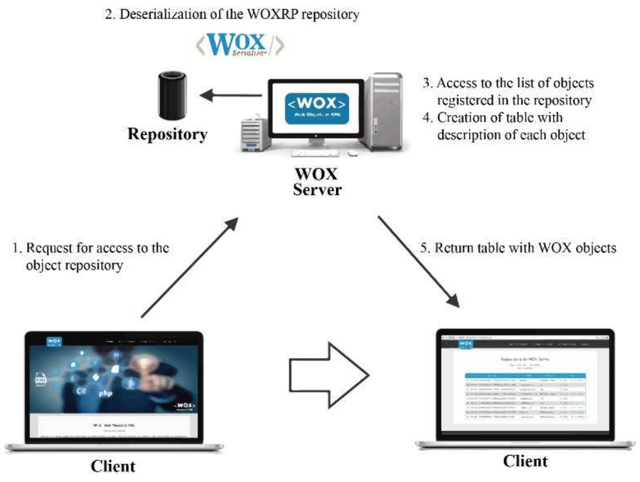
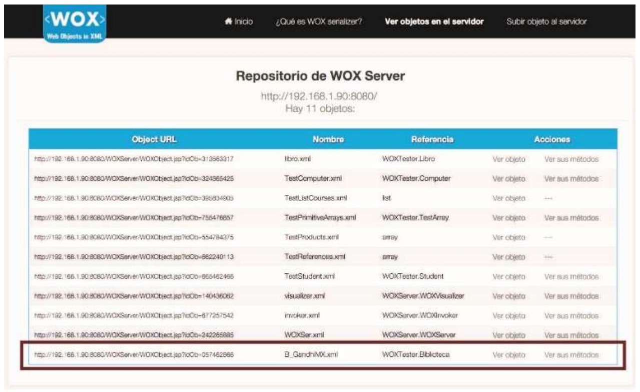

2. Deserialization of the WOXRP repository
WOX Serializer
Repository
WOX Server
3. Access to the list of objects registered in the repository
4. Creation of table with description of each object
1. Request for access to the object repository
5. Return table with WOX objects
Client
Client

Fig. 8. Access to the repository for visualization of objects.

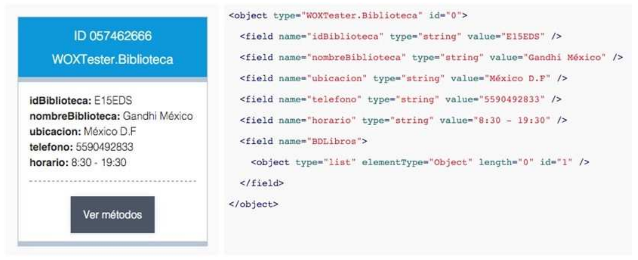


Repositorio de WOX Server

http://192.168.1.90:8080/
Hay 11 objetos:

Object URL	Nombre	Referencia	Acciones	
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=313563317	libro.xml	WOXTester.Libro	Ver objeto	Ver sus métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=324565425	TestComputer.xml	WOXTester.Computer	Ver objeto	Ver sus métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=395834905	TestListCourses.xml	list	Ver objeto	---
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=755476857	TestPrimitiveArrays.xml	WOXTester.TestArray	Ver objeto	Ver sus métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=554784375	TestProducts.xml	array	Ver objeto	---
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=662240113	TestReferences.xml	array	Ver objeto	---
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=665462466	TestStudent.xml	WOXTester.Student	Ver objeto	Ver sus métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=140436062	visualizer.xml	WOXServer.WOXVisualizer	Ver objeto	Ver sus métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=677257542	invoker.xml	WOXServer.WOXInvoker	Ver objeto	Ver sus métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=242265885	WOXSer.xml	WOXServer.WOXServer	Ver objeto	Ver sus métodos
http://192.168.1.90:8080/WOXServer/WOXObject.jsp?idOb=057462666	B_GandhiMX.xml	WOXTester.Biblioteca	Ver objeto	Ver sus métodos

Fig. 9. Access to the repository for visualization of objects.


ID 057462666
WOXTester.Biblioteca

idBiblioteca: E15EDS
nombreBiblioteca: Gandhi México
ubicacion: México D.F
telefono: 5590492833
horario: 8:30 - 19:30

```
<object type="WOXTester.Biblioteca" id="0">
  <field name="idBiblioteca" type="string" value="E15EDS" />
  <field name="nombreBiblioteca" type="string" value="Gandhi México" />
  <field name="ubicacion" type="string" value="México D.F" />
  <field name="telefono" type="string" value="5590492833" />
  <field name="horario" type="string" value="8:30 - 19:30" />
  <field name="BDLibros">
    <object type="list" elementType="Object" length="0" id="1" />
  </field>
</object>
```

Fig. 10. Visualization of an object graphically and in XML.

VI. Conclusions and Future Work

This paper presented a web system that allows you to store objects, as well as visualize their state and methods. This web system is a complement of Web Objects in XML (WOX), which is a framework for programming distributed object-based applications.

The web system met the objectives set at the beginning of its development, since it was possible to successfully design and implement the storage of WOX objects and provide an object repository from which it is possible to visualize, through a web browser, each object that has been created in it, both graphically and in its XML representation.

Future work is necessary to complete the implementation for displaying the methods of any WOX object found in the repository, as well as the execution of methods of a WOX object through an interface, where the values for each of its parameters are provided.

REFERENCES

[1] C. R. Jaimez-González, S. M. Lucas, “Implementing a State-Based Application Using Web Objects in XML”, in: Meersman R., Tari Z. (eds) On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS. OTM 2007. Lecture Notes in Computer Science, vol. 4803, pp. 577-594, 2007, Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-76848-7_40

[2] C. R. Jaimez-González, S. M. Lucas, E. López-Ornelas, “Easy XML Serialization of C# and Java Objects”, Balisage: The Markup Conference 2011, Montréal, Canada, August 2011, in Proceedings of Balisage: The Markup Conference 2011, Balisage Series on Markup Technologies, vol. 7. https://doi.org/10.4242/BalisageVol7.Jaimez01

[3] C. R. Jaimez-González, W. A. Luna-Ramírez, S. M. Lucas, “A Web Tool for Monitoring HTTP Asynchronous Method Invocations”, in Proceedings of the IEEE International Conference for Internet Technology and Secured Transactions, pp. 127-132, London, December 2012, https://ieeexplore.ieee.org/document/6470883

[4] P. Merle, C. Gransart, J. Geib, “CorbaWeb: A Generic Object Navigator”, http://www.lifl.fr/~merle/papers/96_WWW5/paper/Overview.html

[5] S. Chang, H. Kim, “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.

[6] M. Carey, L. Haas, V. Maganty, J. Williams, “PESTO: An Integrated Query/Browser for Object Databases”, in Proceedings of the 22th International Conference on Very Large Data Bases, Mumbai, India, 1996.

[7] G. Kumar, P. Jalote, “A Browser Front End for CORBA Objects”, in 10th International World Wide Web Conference, 2001.

[8] Apache Software Foundation. Web Services - Apache Axis. http://ws.apache.org/axis/

[9] M. Tork, M. Arya, L. Haas, M. Carey, W. Cody, R. Fagin, P. Schwarz, J. Thomas, E. Wimmers, “The Garlic Project”, in Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, New York, 1996.

[10] R. Fielding, “Architectural Styles and Design of Network-Based Software Architectures”, PhD thesis, USA, 2000.

[11] C. R. Jaimez-González, S. M. Lucas, “Web Objects in XML (WOX): Efficient and easy XML serialization of Java and C# objects”, http://woxserializer.sourceforge.net/