World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:11, 2007

A Case Study: Experiences with Building an Online Exhibition System using Web Services

Atakan Kurt, Arzu Naiboğlu

Abstract**—**We present an implementation of an Online Exhibition System (OES) web service(s) that reflects our experiences with using web service development packages and software process models. The system provides major functionality that exists in similar packages. While developing such a complex web service, we gained insightful experience (i) in the traditional software development processes: waterfall model and evolutionary development and their fitness to web services development, (ii) in the fitness and effectiveness of a major web services development kit·

Keywords**—**Web Services, Online Exhibition System, Software Engineering, Waterfall Model, e-business.

I. Introduction

WITH the advent of XML [1] in the Internet, and with the recent developments of protocols and methodologies in network programming, distributed computing has taken a new shape called service oriented computing [2], or web services [3]. Web services is a new computing model where a function that performs a certain task is developed, publicized and used by three different parties, as opposed to the traditional computing model where the developer, the user and the publicizer are usually the same person Obviously web services revolutionize the way the computing is performed on the networked environment, as most applications are now distributed or client/server applications. With more mobile and wireless devices coming to market, web services are becoming more important.

Though it seems that web services are appropriate for small applications or task, we believe that it will be unavoidable in near future that mid and large size applications will be offered as complex sets of web services as part of the distributed computing transformation. Therefore there is a need for development methods and supporting technology for complex web services. In this paper we present the experiences with implementing a web service(s) for an Online Exhibition System. We reflect upon our experience with respect to software process methodology and the experience with the tools used in the web service development.

It seems the traditional theory and methodology of the software design and development lacks to address the problems introduced by the complex nature of the web services computing model. We can expect that the web services model will have a major impact on the software engineering field. Object Oriented Analysis and Design with UML revolutionized the Software Engineering practice. Similarly web services are deeply changing the way the software is designed for the distributed environment.

The experiences with the web services development tools are important as companies are rushing new tools or the new versions of existing RAD tools with web service support to the market. It appears that the technology will be the driving force for the theory in the case of web services. We observed that the tools have solid but limited functionality and far from the meeting the challenges of the intricate web services computing model.

The paper is organized as follows. In Section 2 we summarize the OES functionality and architecture. The architecture of Java API for XML-Based RPC (JAX-RPC) used for the implementation is presented in Section 3. We briefly touch upon some important points on the implementation of OES in Section 4. Various issues surrounding our web service implementation experience specifically and other issues regarding web services in general are discussed in Section 5. Concluding remarks are given in Section 6.

II. System Architecture

Online exhibitions are used in e-business for the online exhibition of goods and services, or in e-museum type applications like [4, 5] to create exhibitions of artworks organized into galleries. The OES allows the system administrator to maintain multiple online exhibitions by artists (which are also the exhibition administrators).

The overall architecture of the classical online exhibition systems is shown in Figure 1. The traditional system consists of system administrator module, exhibition administrator module and an exhibition module. The system administrator module has functionality to create exhibition administrators, reporting/monitoring capability. Exhibition administrator module manages multiple exhibitions and monitor visitor activity. An online exhibition is a timed event created in the exhibition module. Online exhibitions consist of galleries. A gallery is an online room filled with artwork. An artwork is presented to the visitor using multimedia and textual data.

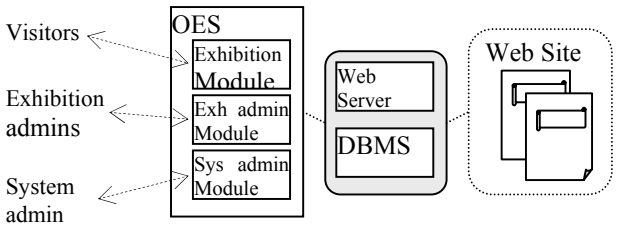

Fig. 1 The Traditional OES Architecture

Manuscript received on 3-30-2005

A.K. is with Fatih University, istanbul, Turkey (+90 212 889-0810 ext 1043, akurt@fatih.edu.tr) A.N. is with Alfabim inc. (anaiboglu@alfabim.com.tr)

In the web services computing model, shown in Fig. 2, the OES is a set of web service available to web site owners (the consumers of web services) who wish to use this service instead of implementing a new one or installing an existing package on the market. The system is designed so that multiple web sites can use this service to offer online exhibitions for their clients. The consumer site contains its own business logic and an *OES client application* that mainly consists of method calls to web services available in the *OES Web Services*. The calls go through the local web server to the OES web server through JAX-RPC [6, 7] using SOAP [8, 9] on top of HTTP explained in the next section.

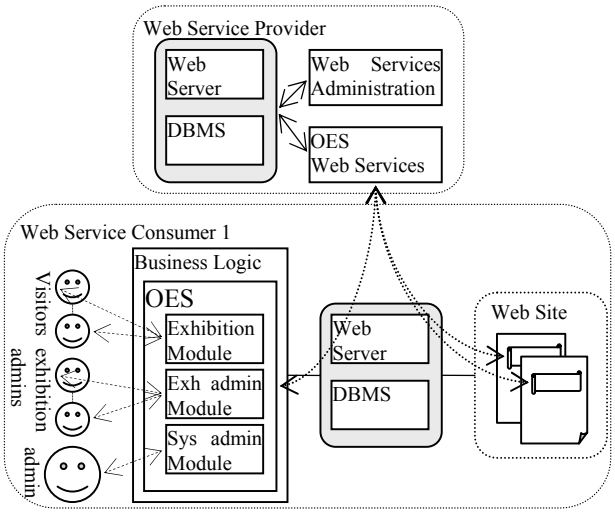

Fig. 2 The OES Web Services Architecture

III. JAX-RPC Architecture

JAX-RPC is a Java API for invoking web services through SOAP using XML. It allows Java clients to make SOAP based Remote Procedure Calls (RPC) to non-Java platforms as well. The complexity of SOAP messages are concealed in the JAX-RPC, because SOAP messages are not required to be explicitly coded to make an RPC call. The call is simply coded using java API. The JAX-RPC converts the RPC to SOAP and transports it to the server/client and server/client converts the SOAP and processes it.

The JAX-RPC Web Service Architecture is shown plainly in Fig. 3. Specific web services are described in *endpoints*. Web Services Description Language (WSDL) [3, 10] documents on the server and the client contains detailed technical information about endpoints. In JAX-RPC, invocations are passed to endpoints. Endpoints are implemented as servlets. All classes, interfaces and other files on clients and server used by JAX-RPC are called *artifacts*. *Stubs*, *ties*, *serializers* and *deserializers* are required artifacts for client-server endpoints communication. Stubs and ties are classes representing service endpoints on the client and the server respectively. A JAX-RPC client invokes a remote method on a service endpoint as though the method were local similar to Remote Method Invocation (RMI) [11] in Java. The corresponding stub class converts this call to a SOAP message and sends it to the endpoint on the server side. The call is handled by a tie on the server side. A tie is a server-side analog to a stub. The tie converts the SOAP message to the right method call. A stub represents a remote object locally on the client. A tie plays a similar role on the server. Serializers and deserializers are classes handling the type conversions from/to Java and XML on both ends.

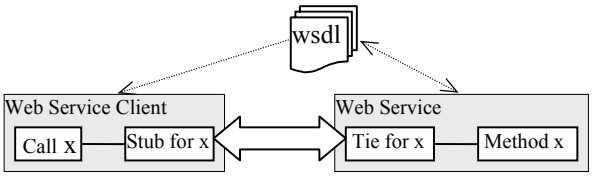

Fig. 3 The JAX-RPC Architecture

IV. Implementation

We implemented the online exhibition system in Java using MySQL database. We used Sun's Web Services Developer Pack (WSDP 1.1) [13] on Windows XP platform. System was written in Java (Java 2 SDK 1.4.1) using JAX-RPC which uses SOAP 1.1 and HTTP 1.1. A screen shot of the system implemented using the OES web services is shown in Fig. 4 (details skipped).

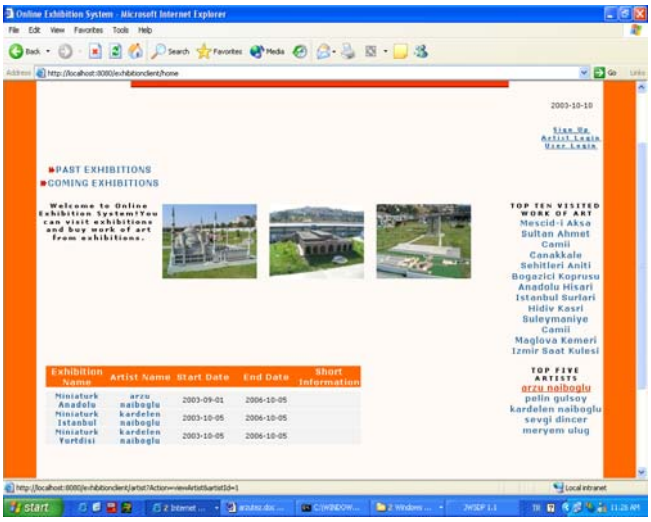

Fig. 4 The Main page of the application

The client application made up of JSP scripts produces pure XML, since the web services are designed to return XML messages. We defined a DTD to define the grammar for the exhibition system for web pages and various administrative reports. The DTD is used for defining a standard vocabulary. All web services produce appropriate XML outputs as described in the DTD not given here for lack of space. The XML messages returned by the web service are converted to HTML by XSLT using an XSL file. The XSLT is performed by the web client such as MS Internet Explorer 5.0 or higher. As a result the web service coding is simplified. Separating the formatting from the data using different versions of the XSL file allows us to create more than one GUI on the client side.

Hence user can dynamically select different GUI without any hassles. Developing a multi-language GUI is also possible with XSLT on the client side in a similar manner. For a better exhibition system, it is possible to create output in VRML and SVG formats through XSLT to create a *virtual* 3D exhibition system.

V. Discussions

We discuss some issues surrounding web service design and development with our experience in this project. One of our objectives was to have a first hand experience with the implementation of a web service. We wished to build a complete application with multiple modules using only web services. We wanted to see the software engineering issues involved in the design and implementation of an application with complex web services. We had a chance to examine a few alternative tools for implementing web services including IBM's Web Services toolkit, now evolved into Emerging Technologies Toolkit (ETTK) [12], Sun's Java WSDP [13], and Microsoft's .NET Web Services [14] among others. We see that some tools are quite handy, while others are cumbersome. Some are easy to use, while others require a good deal of study to understand the architecture.

Waterfall Model and Web Services

Web services are used either as a part of an application like using a car rental web service in an airline reservation system web site or developing a complete application involving multiple modules. In both cases a web service is required to deliver a complete and coherent functionality. The design and development a web service(s) may require using software processes such as waterfall model [15], or evolutionary development using prototyping [15]. How the software processes are affected by web service computing model is an issue that requires consideration. We used waterfall model for the project.

We think that there could be two ways to use waterfall model in web service development as shown in Fig. 5, (Even though the waterfall model is considered an underdog, it is still used widely. We are aware that there have been studies in inadequacy of waterfall model for web services.) First method, shown in the upper part of the figure, requires applying appropriate measures in all steps of the process model. In the requirement analysis phase, issues involving who will provide what web services in what manners etc, needs to be addressed and documented explicitly. In the detailed design (also called object design if Object Oriented paradigm is used), web services will be shown as classes or methods in the *(system and object) design documents*. If the project is a web service client application, then the web service is already implemented by someone else and it will simply be reused by the application. If the project is the design of the web service itself, then the system will be designed in most flexible way to allow others to create different versions of client applications using the web service.

The second method of combining web service design into waterfall model would be to develop the application as a traditional application up to the *implementation phase*. A new phase, called *web services design*, involving the intricacies of web service is employed before the implementation phase. In the web service design phase, the client/server architecture of the web service with all classes/methods, protocols and transports are decided and documented. This second method seems more appropriate if an existing traditional system with all documentations at hand is to be converted to web services architecture, or both a traditional version and web service version of an application are to be developed. We used the second method in our system, since we set out to replicate an existing system.

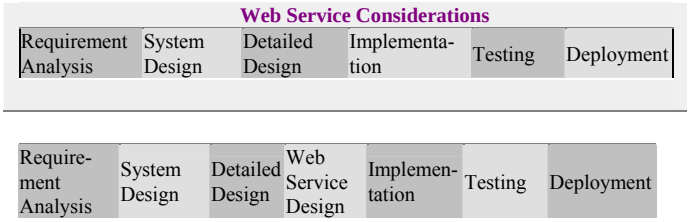

Fig. 5 Waterfall model involving web services

It looks the evolutionary software process model could be easily used with/for web services development as well. Since web services are small invocable web methods, writing and testing methods one-by-one, or in small groups (modules) is a suitable process for the evolutionary model. The web services can be implemented in small coherent increments for the small modules of a big application. It is possible to do throw-away prototyping by creating small prototype applications using the web services developed at the beginning and by adding more web services to create bigger versions at each iteration and arriving a better prototype application at each iteration until a final version is reached.

Web Service Design versus Client Application Design

Let *web service design* be the process of planning a web service itself. A web service can be implemented and tested using stubs without actually creating a client application. A *web service client application* could be a standalone application that integrates one or more web service into its existing code, or it may consist of web service calls only (a *pure* web service client application). Designing a web service itself is different in some aspects from designing the web service client application:

- Web service client application is designed for one company or a web site, whereas a web service is usually designed for many client applications. The issues of multi-users, multi-databases have to be carefully studied. Methodologies and technology at the application design level should be developed to address these issues in web services development. For example a DBMS could be enhanced with tools to either convert a single-company database, to multiple company databases with all stored procedures, or to create new multiple copies of a single-company database to support web pages. Similarly a class library written for a single-company should be either be converted to multi-company library or multiple copies of the library should be generated to support web services for multiple clients.

- Client applications can be created and modified in different ways using the same web service. For example the admin module may or may not include statistics functionality in one client application, while it can have it in other client application. However once described in WSDL documents, the interfaces of methods can't be modified. When creating a complex set of web services, there needs to be more documentation about how all the little web services behave together to create client applications. The data dependencies, control dependencies, deployment dependencies and such issues should be easily and precisely expressible in some form.
- The subjects of deployment, pricing, accounting, auditing, security and such administrative task has to be provided to the administrator of the web service provider by the commercial tools available. These tools should also allow web server client application or the administrator of that application to access similar facilities.

Web Service Integration

Developing a pure web service client application may involve integrating multiple web services from multiple providers. For example in an application authentication, accounting, storage facilities could be provided by different web services. There are issues related to Object Oriented Analysis and Design [16] in developing such applications:

- A pure client application makes full reuse of existing web services. No significant coding, except a skeleton application that invokes the web services.
- Issues involving disparities between the different web services such as data types, time zones, character sets, file formats etc. should be resolved.
- Issues with respect to dependability (used here in software engineering terms) regarding a web service that uses other web services in turn, or a web service that repackages an existing web service.

CASE Tools for Web Service Development

There are Computer Aided Software Engineering (CASE) tools used for developing web services and web service client applications such as Web Services Toolkit with Web Sphere Application Development Environment, Sun's Web Services Development Kid, Microsoft's .NET environment. Many RAD packages usually include one or more tools for web service development. A programming language, a DBMS and a web server are preconfigured in most system, keeping in mind that web services are platform independent.

The tools specify how a web service is created from the existing methods written in a high level language in essence. The WSDL documents, SOAP messaging and UDDI registration are usually automated by the tools. However considerable differences exist between different tools regarding the number of steps involved in creating, modifying, deploying and testing a web service. We observed that in some cases development and testing are slowed down considerably. It appears these tools are in their infancy with respect to the problems mentioned above in this section.

VI. Conclusions

An online exhibition management system using web services computing model is presented. We believe the system demonstrates the followings (i) An existing online application can be fully implemented as a web service or a set of web services (ii) With web services, a flexibility in creating different versions of the same application is possible, i.e., the final applications can be organized in different ways that is not possible or is difficult with the non-web service version. (iii) Most software engineering techniques including Object Oriented Paradigm that apply a traditional application development also apply to the development of web services. However there are basic differences between the traditional distributed application and web services architecture. There are different ways of integrating web services design and development into existing software engineering methodology. (iv) There is a degree of complexity associated with web services model that makes it difficult for developers to understand architecture and to use the web services development tools.

Topics such as web service design, development, integration, security, pricing, accounting, benchmarking, etc. remains among important problems that need further study.

References

[1] W3 Consorium , "The Extensible Markup Language", www.w3.org/xml
[2] E. Thomas, "Service-Oriented Architecture", Prentice Hall, ISBN :0131428985, 2004.
[3] E. Newcomer, "Understanding Web Services: XML, WSDL, SOAP, and UDDI", Addison-Wesley, ISBN: 0201750813, 2002.
[4] Smithsonian Art Museum, http://americanart.si.edu/index2.cfm.
[5] Palmer Museum of Art, http://www.psu.edu/dept/palmermuseum/
[6] Sun Micro Systems, "Java API for XML-Based RPC (JAX-RPC)", http://java.sun.com/xml/jaxrpc/index.jsp
[7] I. Singh et all, "Designing Web Services with the J2EE(TM) 1.4 ", Addison-Wesley Professional, ISBN: 0321205219, 2004.
[8] J. Snell et all, "Programming Web Services with SOAP", O'Reilly, ISBN: 0596000952, 2001.
[9] W3 Consorium, "Simple Object Access Protocol (SOAP)," http://www.w3.org/2000/xp/Group/
[10] W3 Consorium, "Web Services Description Language (WSDL), Version 2.0, Part1: Core Language", http://www.w3.org/TR/wsdl20/
[11] Sun Micro System, "Java RMI Specification", http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html
[12] IBM Alpha works, "Enerping Technologies Toolkit" http://www. alphaworks.ibm.com/tech/webservicestoolkit
[13] Sun Micro Systems, "Java Web Services Developer Pack (WSDP)," http://java.sun.com/webservices/jwsdp/index.jsp
[14] A. Ferrara and M. MacDonald, "Programming .NET Web Services", O'Reilly, ISBN: 0596002505, 2002.
[15] I. Sommerville, "Software Engineering (6th Edtion)", Addison-Wesley, ISBN: 020139815X, 2000.
[16] G. Booch, "Object-Oriented Analysis and Design with Applications (2nd Edition)", Pearson, isbn 0805353402, 1993.