WEB SERVICE BASED MULTI-CHANNEL APPLICATION

Paulo Alexandre Neves, Octávio Pereira, João Silva, Paulo Ramalho
Escola Superior de Tecnologia de Castelo Branco
Avenida do Empresário, Castelo Branco, Portugal
pneves@est.ipcb.pt, octavio@est.ipcb.pt, kappy@est.ipcb.pt, pramalho@est.ipcb.pt

ABSTRACT

The paper presents an application using a purpose-created web service that is consumed in a multi-channel environment. The service clients span over the common web interface, a JAVA-enabled GSM 2.5G mobile phone, and a Microsoft® Windows® application. The web service provides basic message passing functionality.

KEYWORDS

Web services, J2ME & J2SE, .NET, mobile systems, Enterprise Application Integration, functional integration

1. INTRODUCTION

The main idea from which this project was born was to develop a message passing middleware, so that any application could be integrated anywhere in a point to point architecture (as a first approach). Even so, an Enterprise Application Integration concept of functional integration was applied [1]. A form of integration of increasing popularity nowadays is the one that appeals to the web services [2]. Supported by standards vulgarly used in the Internet, such as the HTTP protocol [3] and XML language [4] [5], they are faced as a form of functionality integration, distributed, platform independent, and also independent of the language of the system that gives the service was developed, and of the supplier. For its characteristics, the web services are looked at by some authors as an ideal solution for EAI [6].

Although the description refers to a point to point integration case, the developed web service API (*Application Programming Interface*) could be an integrant part of a more complex architecture, where it could be invoked by several heterogeneous systems, through different channels. Such systems could still be supported by different technologies of communication middleware ("Software in the middle of …"[1]). The state-of-the-art architecture which leverages this assumption is known as hub-and-spoke.

The control of the information flow in this kind of structure is achieved by the central integration server (Hub), supported by a middleware technology. The server manages the harmonious communication between the peripheral applications that surround the hub, the Spokes [7]. The links between the hub and the spokes can represent any different middleware, which can integrate different applications. This architecture supports the "publish and subscribe" policy for the service, in a loosely-coupled way. Publishers produce messages on a particular subject or topic name, and subscribers receive those messages by registering either explicitly or through some broader subscription involving wildcards. If the subscription matches a published topic, the application receives the message; otherwise, the application is never notified. As such, information is pushed to applications instead of those applications having to pull or request it [8]. The technology selection falls to web services, which use is located in the rising edge of its hype cycle on the market (see Figure 1).

As a result, the web services technology was adopted to create a multi-channel application. The idea behind this application is to have multiple users sending messages in a one-to-all architecture. The paper contributes to the adoption of web services as a functionality integration tool, and the multi-channel aspect of the application presents client flexibility. In section 2 the paper introduces a short overview of web service technology. The application is then presented, with its various implementations and finally, a conclusion is taken and future work is presented.

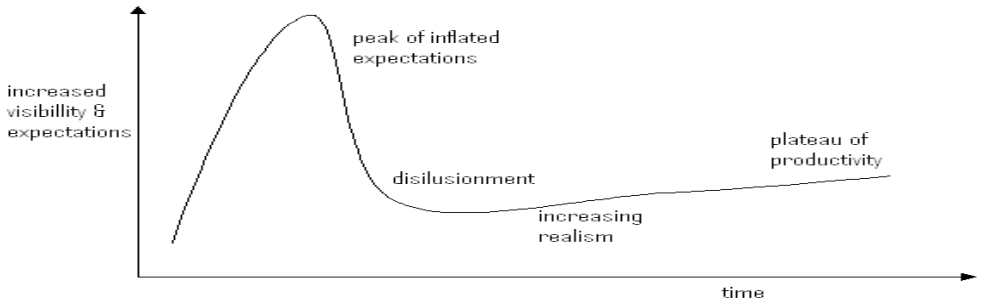

Figure 1 – Hype Cycle of New Technologies. (source: Gartner Group) [9]

2. AN OVERVIEW OF THE WEB SERVICE TECHNOLOGY

A way of functionality integration, with a substantial increase in use, is the one that is based on web services. One of the main ideas behind this recent technology is to provide a means for the communication between different components of an application, using different hardware and software platforms. With IPV6, every "intelligent" device will support the HTTP protocol, which further increases the interest for this technology. The web services are firewall-friendly, with the use of the http transport protocol that, by definition, uses port 80. Since every computer that uses the Web need this port open, the information flows. As a result, integration between different applications can also be achieved, provided the means to use it.

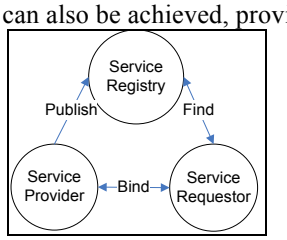

Figure 2 - The web service's architecture [10]

The web services are based in three basic concepts: Find and register a service, transport mechanism to reach a service, and the input and output parameters of the requested service. As a result the web service presents three basic entities that are represented in Figure 2: the Service Registry that is a register of services, the Service Provider that is the supplier of the service and the Service Requester, the service client. Web services can configure architecture of single services, most known as Service Oriented Architecture. Each web service is a closed component, which can be requested in a distributed way by other application [11].

The web services rely on several protocols. The UDDI [12] allows the discovering of a service: Microsoft®, IBM® and Sun® provide UDDI servers that are periodically synchronized. The WSDL document defines an XML schema that describes the service to allow a client to consume it. SOAP is a XML based protocol document that allows communication between applications using HTTP, and can be understand by any type of application. The web service is "called" by a client using a SOAP request and the web service answer is sent back to the client using a SOAP response. As a result, both applications (client and server) need tools to process a XML document, parse it, and withdraw the information from it.

3. APPLICATION DEVELOPMENT

The application (DEMOV – DEvelopment of an application in a Multi-channel enVironment) is a .NET web service, based on .NET running on a IIS server. Two methods were defined: send and receive. Two possible implementations (API's) found for the SOAP protocol were kSoap [13] and wSoap [14]. Both presented a XML parser and connection capabilities, with the wSoap presenting the smallest size.

Two methods were defined: send and receive. The Web Service accepts SOAP calls and answers with a SOAP response. The wSoap and kSoap API's are responsible for the implementation of the SOAP protocol and also present connection capabilities. Figure 3 presents the application architecture.

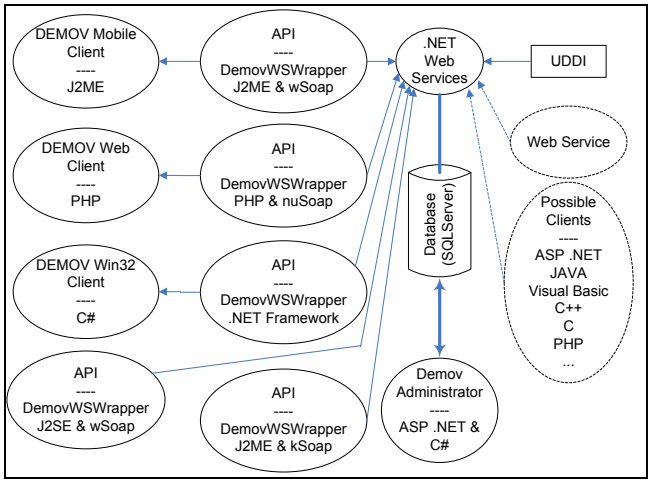

Figure 3 - DEMOV application architecture

3.1. Server and Web/Win32 Client

The server side relies on a Microsoft® SQL Server 2000 to store the messages and authentication of users. Each user has its own login name and password and privileges (administrator or normal user). However, due to memory limitations imposed by the mobile device (64KB – *Nokia 6610*), the authentication information is sent in text. In the Web Client two different implementations exist, based on the user rights. A normal user can only send and see the other user's sent messages, while the administrator can manage the messages stored on the server and create new users that may have normal or administrator rights too. Three types of messages are defined according to the "age": recent messages, that are less than one week old, old messages that have more than one week but less than 2 weeks, and finally, very old messages, that are more than 2 weeks old. A traffic light colour coding was used to clearly identify the several types. This information is only visible in the administration page. The Web Service was built in C# and ASP .NET, using the wSoap API and is registered in a UDDI server, which allows everyone to see and use it. To accomplish this, one just has to search for "Demov". The C# language was used as the "Code Behind" part, and ASP .NET was used to create the page layout.

To enforce the multi-channel side of the application a browser independent application was created. This application runs on a Windows® platform with Win32 support and was developed in .NET. This application has the same functionality as the Mobile Client, presenting a friendlier user interface.

3.2. Mobile Client

The Mobile client relies on J2ME, with the Connected Limited Device Configuration 1.0 specification and the Mobile Information Device Profile. The target device was the Nokia 6610. However, the application should run on any Series 40 device with MIDP 1.0 [15]. The hardware development platform was chosen having in mind market issues, even having in mind the recent drop in Nokia's market share [16]. Having in mind memory limitations of the device, the wSoap API was used. The Mobile Client also has 19KB of message storage space, with the possibility to use the T9 [17] to write the message. At this level every user has the same privileges, since administration is only available on the Web application.

Figure 4 presents the look and feel of the Mobile Client in three different scenarios (from left to right): the main menu, the writing of a new message and the received messages. The received messages are shown with the name of the emitter user. A different symbol is used for the already read messages and the unread ones. The definitions part of application defines the current user (login and password) and defines the web service namespace.

Figure 4 - Screenshots of the Mobile Client. Main Menu, New Message, Received Messages

4. CONCLUSION AND FUTURE WORK

A multi-channel application was developed using a state-of-the-art integration technology: the web services. The application architecture provides functionality integration through the SOAP protocol; which can call any remote method relying on a web service. The message passing application itself provides low functionality; however the functional architecture allows the development of more complex applications.

In terms of future work, the construction of an API to replace the SOAP implementation, in order to shrink the amount of memory needed, and the implementation of the web service under the J2SE platform. With the implementation under a different software platform, performance issues can also be studied. ~

Regarding the developed code, a generic message passing middleware is an objective to reach: most kind of data flow between two applications can be encapsulated through the developed methods on the web service.

REFERENCES

[1] - Linthicum, D., “Enterprise Application Integration”, Addison-Wesley, Massachusetts, (1999)

[2] - Joseph Daniel Procopio, under orientation of Walter D. Potter, “A Three.Tier Design For Allowing Thin Clients Using SML Lahered In SOAP To Access Legacy Applications”, PhD Thesis, 2002

[3] - http://www.w3.org/MarkUp/

[4] - http://www.w3.org/XML/

[5] - Cecilia Mascolo et al., 2002, “XML Technologies and Software Engineering”, Proceedings of the 23rd International Conference on Software Engineering, Toronto, Ontario, Canada

[6] - Samtani, G. e Sadhwani D., “EAI and Web Services-Easier Enterprise Application Integration?”, www.webservicesarchitect.com , 17, 10 (2001)

[7] - Dhenin, C., “Panorama EAI: Les architectures en concurrence”, Journal du Net - http://solutions.journaldunet.com/0202/020211_comparo_eai_archi.shtml, 11, 02 (2002)

[8] - Paul Timberlake, 2002, http://www.ebizq.net/topics/messaging_middleware/features/1798.html

[9] - Massimo Pezzini, Gartner Group, 2001, EAI and Web Sevices, Centro Cultural de Belém

[10] - Mark Colan, A Technical Overview of Web Services, http://ibm.com/developerworks/speakers/colan

[11] - http://www.service-architecture.com/

[12] - www.uddi.org

[13] - www.ksoap.org

[14] – www.wingfoot.com

[15] - www.forum.nokia.com

[16] - http://tek.sapo.pt/4N0/474792.html

[17] - www.tegic.com