Proceedings of the 10th WSEAS International Conference on COMPUTERS, Vouliagmeni, Athens, Greece, July 13-15, 2006 (pp944-947)

Universal Web Application Server

JAVIER PARRA-FUENTE
MARTA FERNÁNDEZ-ALARCÓN
LUIS JOYANES-AGUILAR

Computer Languages and Systems Department
Pontifical University of Salamanca - Madrid campus
Paseo Juan XXIII, 3. 28040 Madrid
Spain

Abstract: Two platforms are the most used to develop Web Application -Sun Microsystem's J2EE Platform and Microsoft's .NET platform-. These platforms are incompatible each other, so if you need to migrate from one platform to another, you will have to rewrite the application code to adapt it to the new platform, or if you want to use a functionality of the Web application, you will have to use a client written with the same technology. The current trend of Web development of software is based on the interoperability of applications that are composed by modules written in different languages and even are written to be deployed in different platforms, but each of these modules are deployed in servers which work only with a certain technology, for example a J2EE Server or a. NET Server.

In this work, we introduce an Universal Web Application Server, in which is possible to deploy applications composed by modules written with different technologies: such as J2EE, .NET, etc, and that are able to communicate each other and interoperate directly without SOAP conversions or without establishing remote communications, like the Web services.

Key-Words: Application Server, J2EE, .NET, Interoperability, Web Services, SOAP, HTTP.

1 Introduction

Nowadays the most used platforms on the Web applications development are J2EE and .NET. Software companies develop several projects at the same time, which are implemented with different technologies and, on the other hand, software programmers try to reuse the most software code as possible. For that reason, Web applications developers might be able to reuse existing software written with any technologies.

These days, it does not exist a server that allows deploying on it different modules written with different technologies, but only the modules written with the same technology can be deployed in the same server. So you need so many different servers as different technologies you use (J2EE, .NET and so on).

In the present work we propose an Universal Web Application Server model, which makes possible to deploy Web applications developed with any Web technology.

In section 2, we analyze the current lacks of the application servers and the solutions to solve them. In sections 3 and 4, we present the requirements and the architecture of the Universal Web Application Server, which solve the lacks what was said before. Related and future work, are described in sections 5 and 6 respectively. Finally, the conclusions of this paper are presented in section 7.

2 Current Lacks in Application Servers

The main lack of the current application servers is the limitation of platform, because a server can only deploy software written with a certain technology.

There are several solutions to achieve interoperability among different platforms: technology independent generic solutions, like the use of Web Services, or solutions implemented for a specific technology, like J2EE Connector Architecture (JCA).

Web services are programmable components that use SOAP.[1] as access protocol, regardless their client and component technology (a drawback in DCOM.[2]), regardless the language in which both communication ends are written (a drawback in RMI.[3]); SOAP generally uses HTTP over the port 80 for request/response, thus crossing corporate firewalls (a drawback in CORBA.[4] or DCOM) and

it also facilitates the interoperability of applications that work with different technologies. The use of Web services to implement applications with interoperability between its modules implies to establish HTTP communications whenever we invoke a service, which will slow down the execution of the application.

On the other hand, JCA [5][6] is Java technology that allows the compatibility among different J2EE servers, and even with .NET components deployed in EIS (*Enterprise Information Servers*) servers. Often, it does not exist a complete compatibility between J2EE servers of different manufacturers. JCA offers a standard to which the suppliers of application servers can use to any EJB will be able to communicate with their application server, allowing a higher interaction. As far as the interoperability between J2EE and .NET, JCA offers a mechanism to allow the interoperabitity between Sun's EJBs and Microsoft's COM components. At the moment there are several J2EE servers with JCA support, and in addition, several companies have developed specific connectors for COM, allowing the use of transactions, security and connection pools for COM. This solution does not solve the need of using remote communication to communicate modules implemented with different technologies within a same application, but it is a very specific solution for certain technologies, and it forces to maintain the different modules in different servers, with the maintenance problems that it supposes.

3 Universal Web Application Server Requirements

In order to solve the above mentioned lacks of the current application servers, we propose the development of an Universal Web Application Server that allows to deploy, in the same server, applications composed by modules implemented with different technologies. The main characteristics of this server are:

- *Multilanguage and multiplatform server.* The Universal Web Application Server will be able to deploy and run applications implemented with different languages and independently of the platform on which it will run, so we will be able to run on the same server modules implemented with different technologies, like J2EE, .NET and so on.
- *Language-independent code representation.*. The server will use a language-independent representation based on XML, what will allow the running of modules written in different languages together.
- *Transformation from compiled code into language-independent code.* The server must be able to transform the original compiled code into language-independent code during the phase of the application deployment in the server. That common representation for all languages will allow the direct communication among the application modules implemented in several languages.
- *Semantic storage of the application deployment information.* It will be necessary to have a repository, in which the deployed modules of the different applications will be published. The application modules could discover another modules or functionalities deployed in the same server and published in this repository.
- *Resource location.* The applications deployed in the Universal Web Application Server and transformed into a language-independent representation must be able to process the HTTP requests. The *locator* must look for the application and resource to be run into the semantic storage and give it to the run engine of the server.
- *Running platform translation.* The programming language-independent application code will have to be transformed into native code of the end platform to be run.
- *Direct interoperability.* The deployed applications modules in the same server will be able to communicate each other directly, even if they are implemented with different technologies, without SOAP translations or HTTP communications.

4 Universal Web Application Server Architecture

After analyzing the requirements that the Universal Web Application Server has to achieve, we introduce the server architecture:

- *Multiplatform code parser.* There will be a compiled code parser of each language in which could be written the Web modules deployable in the server, which will transform the compiled code into language-independent code in a document based on XML.
- *Application repository.* Storage where is collected semantic information about the applications and its modules deployed in the server.

- *Application locator*. This module locates Web resources deployed in the server using the application repository.
- *Multilanguage interpreter*. Translator that turns the language-independent code into native code of the execution platform.

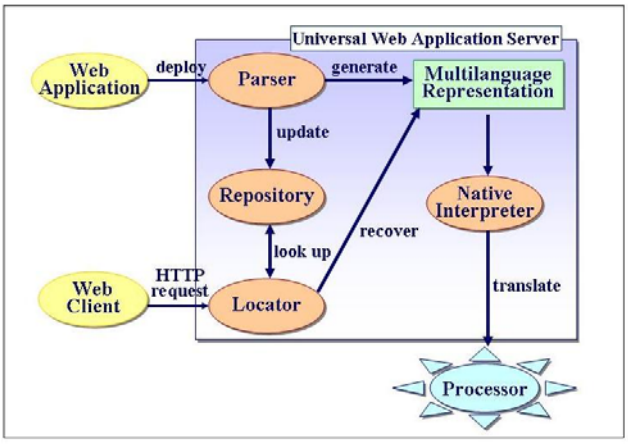

Figure 1. Universal Web Application Server Arquitecture.

Figure 1 shows the existing relationships between the different modules of the server architecture. The Web application will be deployed in the server and the *parser* will analyze the code and it will transform it into a multilanguage code representation document based on XML. The parser also updates the *repository* with the deployed application data. This repository will be used by the *locator* when the server receives a HTTP request from the Web client to run a resource of a specific Web application. Once the multilanguage code of the resource is located, it will be transformed into native code by the *interpreter* and it will be run by the processor.

5 Related Work

The Universal Web Application Server has its theoretical base in the multiplatform RAWS server [7]. This server deploys Web services and turned them into reflective and adaptable Web services automatically, in which is possible to modify dynamically the structure (business method's name, adding a parameter to the method, etc.) and the behaviour (changing the code that runs a business method) of the Web service.

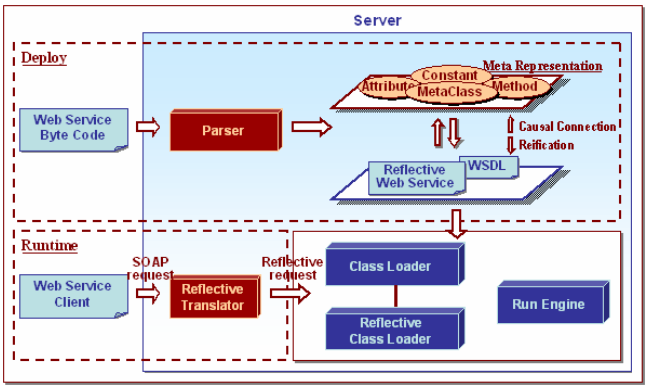

Figure 2. Multiplatform RAWS Architecture.

Figure 2 shows the multiplatform RAWS Server architecture [8] in which a *parser* analyzes the Web service Byte Code and it deploys it with a reflective architecture, which has a meta-representation with the structure and behaviour of the Web service. The meta-representation holds all the necessary reflective methods to transform the structure and the behaviour of the Web service. On the other hand, the *reflective translator* translates the SOAP requests into runnable requests in the reflective architecture.

6 Future Work

At the moment there are four opened research lines related to the Universal Web Application Server:

- *Multiparadigm server*. The current server recognises J2EE code and .NET code. This code is multiplatform object oriented. The aim of this work is to develop a multiparadigm representation that allows the run of Web applications independently of the programming paradigm in which they are implemented.
- *Reflective server*. The applications to be deployed in this new server version will be auto-adaptable, so that it is not necessary to change the code, to compile it or to deploy it again, in order to transform their structure or behaviour.
- *Web component server*. In that research line is being modified the server in order to transform automatically the deployed Web application functionalities into business methods of Web services just to make them remotely accessible through SOAP/HTTP requests.
- *Semantic server*. The server will hold a semantic representation of the content deployed on it, so that the applications can know and locate the functionalities that are in the server in order to incorporate them into their Web applications.

7 Conclusions

In this paper we have introduced a Universal Web Application Server that is able to deploy and run applications implemented in different languages and with different technologies. This server presents multiple advantages compared with the conventional application servers, some advantages are the following:

- *Universal server*. It provides a server to be able to deploy on it applications, independently the language in which it is written. It will not be necessary to administer, to have licenses, to know the operation, etc of different servers, because all the Web application will be deployed in this server.
- *Direct interoperability.* The Web modules of the same application implemented in different languages will be able to communicate each other without SOAP communications.
- *Language-independent representation.* The applications are represented independently of the implementation language used, what allows the code generation directly from the analysis and design phases, without having to know multiple programming languages.

References

[1]Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.J, Frystyk, H.: *Simple Object Access Protocol Version 1.2*, W3C Recommendation. World Wide Web Consortium (2003).

[2]Thai, T.L.: *Learning Dcom*. O'Reilly (1999).

[3]Grosso, W.: *Java RMI*. O'Reilly (2001).

[4]Bolton, F.: *Pure CORBA*. SAMS (2001).

[5]Hansen, M., Mamorski, P.: *Integrating EJB and COM using the J2EE Connector Architecture*. (2001) http://www.ebizq.net/topics/dev_tools/features/1603.html

[6]http://java.sun.com/j2ee/connector/

[7]Parra-Fuente, J., Sánchez-Alonso, S., Sanjuán-Martinez, O., Joyanes-Aguilar, L., *RAWS: Reflective Engineering for Web Services*, The 2004 IEEE Internacional Conference on Web Services (ICWS'2004), pp: 488-495, IEEE Computer Society, 2004.

[8]Parra-Fuente, J., Sánchez-Alonso, S., Sanjuán-Martinez, O., Joyanes-Aguilar, L., *RAWS Architecture: Reflective and Adaptable Web Service Model*, International Journal of Web Services Research (JWSR), vol. 2, issue 1, pp. 36-53; Idea Group, 2005.