Development of a Robust and Flexible Weblab Framework based on AJAX and Design Patterns

Ariadne A. Cruz, Fábio A. L. Gomes, Fabbryccio A. C. M. Cardoso,
Ernesto B. Martin and Dalton S. Arantes

ComL@b - A KyaTera Lab on Digital Communications
Department of Communications
School of Electrical and Computer Engineering - UNICAMP
(ariadne, adrianol, cardoso, martin, dalton@decom.fee.unicamp.br)

Abstract

The objective of this paper is to present a generic and extensible access framework architecture for WebLab integration. In this framework each WebLab becomes accessible by means of a preinstalled plug-in. This modular approach makes it possible to add, remove or modify a plug-in, and its corresponding WebLab, without framework recompilation.

Index Terms: *WebLab Framework, Design Pattern, AJAX, Plug-ins.*

1. Introduction

The demand for efficient Internet applications and Web Services [1] has increased continuously. A WebLab is a typical example of a Web Service in which the goal is to provide remote access to laboratory experiments. The development of a WebLab integration platform based on framework provides a fast and low cost implementation, resulting in an efficient base structure for interconnecting remote equipments, sensors or any kind of electronic device.

At present there are a large number of proposals for WebLabs in FAPESP's KyaTera network [2], and the framework discussed in this paper aims at an efficient integration of these WebLabs. Common access to geographically distributed resources is of fundamental importance to foster collaborative work in key areas of science. The objective of the framework presented here is to offer to developers a basic infrastructure of services, such as a client/server communication, persistent services and facilities to develop user interfaces. Our aim is to contribute to e-Science development in Brazil and Latin America.

The framework is based on state-of-the-art methodologies and technologies, such as **DIP** (Dependency Inversion Principle) and **EO** (Extension Object) design patterns, **AJAX** (Asynchronous Javascript And XML) paradigm and **JSON-RPC** (JavaScript Object Notation-Remote Procedure Call) communication protocol. DIP and EO are design patterns that are used for creating plug-in architectures. AJAX is a Web development technique for creating Rich Internet Applications (RIA) and JSON-RPC is a specification that allows a client/server communication.

2. Motivation

The WebLab integration platform described in this paper is being developed according to the following principles:

2.1. Extensibility

The diversity of WebLabs that are being developed in the KyaTera project requires that the framework should be able to supply different kinds of services to different kinds of users. For engineers, WebLabs should be able to provide remote access to oscilloscopes, spectrum analyzers, vector signal generators and other equipments for signal analysis, visualization, control, and so forth. For statisticians, for example, data analysis services should be available. Since an infinite number of services can be envisioned for the near future, a flexible approach is mandatory for our framework. The adopted solution uses a design pattern plug-in (combination of DIP and EO patterns), which will be described in the following sections.

Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)
0-7695-2833-3/07 $20.00 © 2007 IEEE

2.2. Robustness

Due to the large number of services that will be provided by WebLabs, it is important to guarantee that each addition, removal or modification of a service will not damage any system functionality. Once again, the plug-in pattern offers a good isolation level among the services (plug-ins) and also among the framework and services.

2.3. Usability

Traditional Web applications make use of synchronous communication between server and client. Looking at this paradigm, the client (browser) sends a request to the server and waits for the answer. In the meantime the Webpage stays blocked, disabling the normal user work flow. Moreover, traditional Web applications have poor interfaces, lacking many useful resources which restrict the development of more sophisticated and faster interfaces. As a solution to these problems, user interfaces are developed using AJAX paradigm (see Section 4.1).

3. Related Works

3.1. Eclipse Platform

The Eclipse Platform [3] is designed for building integrated development environments. It is built on a mechanism for discovering, integrating and running plug-in modules.

Any plug-in is free to define new extension points and to provide new APIs for other plug-ins. Some qualified plug-ins can extend the functionality of other plug-ins as well as extend the kernel. This provides flexibility to create more complex configurations. By careful analysis of the Eclipse Platform, we have observed that the DIP and EO patterns (see Sections 5.1 and 5.2) would be an interesting solution for the implementation of plug-ins, since they provide the possibility of inserting, removing and updating the plug-ins without the need of recompilation of the framework's core.

3.2. REAL Virtual Laboratory

The REAL Virtual Laboratory [4][5], which can be accessed by internet, was implemented with a telematic service. The main objective of this proposition is to provide the researchers and students, located anywhere, access to an XR4000 autonomous mobile robot, so that they can test and validate the robot control and navigation methods without spending large amounts of money on an actual robot.

To use REAL's platform, the user needs a computation environment that supports a Web navigator using Java Virtual Machine 2 (eventually for a Java plug-in) with capability to execute applets. The support infra-structure for the services is automatically installed in the user environment every time it is requested. In our case, and since our proposition aims at a minimal installation, AJAX technology has been chosen for developing the pages, since there is no need for the user to download the plug-ins or runtimes (for example, Java, Flash, ActiveX, etc.). This approach still enables, however, the possibility to combine the interaction of the user and the requests to the server in a more integrated manner, allowing the design of interfaces much more similar to desktop programs without losing the advantages of a Web application.

3.3 Google Platform

After Google started to develop some new applications with AJAX, public attention was drawn towards this new technology. Some of the major products Google has introduced over the last year by using AJAX model are the Google Groups, Google Suggests, Google Maps and Google Mail [6].

Our platform is similar to Google's Platform in many aspects, especially in the user-friendly interface that favors usability inherent in the AJAX technology. With this technology the possibilities for the Web development increase significantly. For example, a greater interaction with the user is possible with AJAX, since the start of the interface agents is independent of the user actions. Note, however, that in none of the services offered by the Google Platform the concept of plug-ins has been implemented.

4 Innovative Technologies

Our proposed framework has been developed and implemented using innovative technologies, such as AJAX and JSON-RPC, which will be described next.

4.1 AJAX

Although AJAX [7] is not a new technology, it incorporates a standard-based presentation using XHTML and CSS, dynamic display and interaction using the DOM (Document Object Model), data interchange and manipulation using XML and XSLT, asynchronous data retrieval using XMLHttpRequest and JavaScript binding everything together [8].

These technologies already exist to emphasize asynchronous communication between client (browser) and

server (PHP, Java, Perl, etc). They provide to the client a continuous usage of the application while data are being uploaded or downloaded in background. In Figure 1 we compare the traditional model for Web applications with the modern AJAX model.

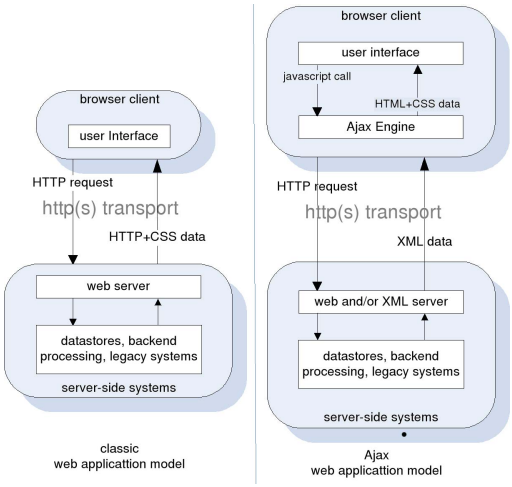

Figure 1. The traditional model for Web applications compared to the modern Ajax model.

Another advantage of AJAX is the fact that the client needs only to download the essential data, in contrast to synchronous requests of the traditional Web applications, in which the data is entirely reloaded in each request. This allows the development of Web applications with much more usability.

4.2 JSON-RPC

JSON-RPC is a stateless and light-weight RPC protocol for inter-networking applications over HTTP. It uses JSON as the data format for all remote procedure call, including all application data carried in parameters [9].

JSON is an interesting alternative to XML/XSLT in AJAX, since it is a native JavaScript format and does not need a JavaScript XML parser in the browser. It is a very simple markup language, with a smaller number of tags and schemas when compared to XML.

The JSON-RPC communication protocol consists in a client establishing a connection with a service and invoking one or more procedures provided by that service.

5 Design Patterns

Design Patterns are not an invention as much as they are a refactoring of existing concepts [10]. The objective is to improve quality software by providing reusability, maintenance, extensibility and short development time [11][12][13].

The description of the patterns is independent of the programming language or details of implementation and it can be used in all types of applications. However, they don't exist in the form of software components; they need to be implemented every time they are used. The patterns provide a way of implementing a "good" project and are intensively used in framework development.

Dependency Injection Pattern (DIP), **Extension Object** (EO), **Data Access Object** (DAO) and **Model View** (MV) are the main patterns that are used in our proposed architecture, as described in the following sections.

5.1 DIP

The objective of DIP pattern is to remove the dependencies control from inside the classes. It simply receives the references instead of instancing each object internally (Figure 2).

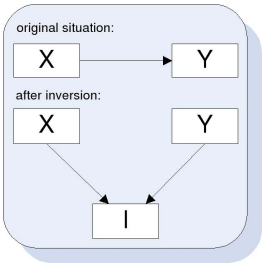

Figure 2. Breaking a dependency with Inversion of Control.

5.2 EO

Known as Extension Object, this pattern prevents bloating of interfaces and breaking of client code when developers add or modify functionality to existing components. Multiple extensions can be attached to the same component, each defining a contract between the component and its clients [14].

5.3 DAO

It offers to a common interface direct access to the data and hides the characteristics of a specific implementation. It defines an interface that can be implemented for each new source of data, allowing the substitution of an implementation for another. Moreover, it manages one connection with

the source of data to get and to store the data. This allows, for example, utilization of a pool of connections managed by the DAO and transparent to the other parts of the system [15].

5.4 MV

This pattern is a simplified version of the MVC (Model-View-Controller) pattern, being a specific variant of the Observer. It has a more streamlined approach, combining the View and Controller functionalities of the MVC paradigm into the View. The application classes may be classified in Model and View [16].

The Model manages the application's data. It responds to queries from the Views regarding state and updates its state when requested to do so by the Views. It notifies the Views when state of the data has changed.

The View presents a view of the model data. It responds to user input, instructing the model to update its data accordingly. On notification of changes to the model data, it retrieves the new model state and renders a view of the latest state of the data. This pattern is used on the client side of the framework developed in this work, as shown in Figure 4.

6 Framework Architecture

In this section we describe the framework technology. The core application is implemented in Java and is divided in four modules; the IU (Interface User) Services, Plug-In Manager, Persistency Service and Model, as in Figure 3.

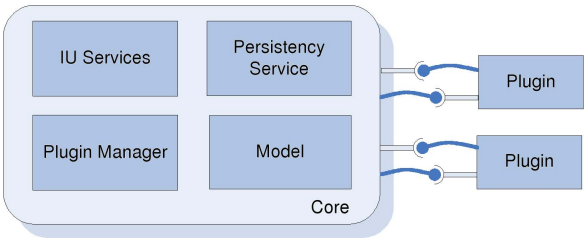

Figure 3. Core Application.

6.1 IU Services

The IU Services is an AJAX application that provides advanced functionalities to the user. This layer is responsible to supply services to the plug-ins and build its own interface.

6.2 Plug-in Manager

This is responsible for the control over all installed plug-ins, and is capable to include new plug-ins without recompiling the core application. This architecture is based on the DIP and EO patterns, as described in Sections 5.1 and 5.2, respectively.

6.3 Persistency Services

This layer provides the ability to store objects in a relational database. However, the use of the DAO pattern described in Section 5.3 allows the use of other storage mechanisms, such as object oriented database (OODB), XML files, among other choices.

6.4 Model

The Model layer provides types for the data structure of the element store and is responsible for the communication between JavaScript client and Server, as in Figure 4. This communication is activated through JSON-RPC protocol (described in Section 4.2).

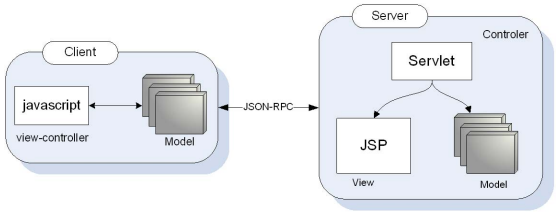

Figure 4. Communication between JavaScript client and server.

7 Implementation Aspects

The current implementation of this framework was based on Backbase [17], which is an AJAX framework. Some services provided by the framework have been developed in Labview, while others have been developed in Java, showing the independency of the platform with respect to the integrated services.

7.1 Backbase Framework

Backbase is an Amsterdam-based company that provided one of the first commercial AJAX frameworks. A key element of the Backbase framework is the Backbase Presentation Client. This is a standard-based engine written

in Javascript that run in the Web browser. It can be programmed via a declarative user interface language called BXML. BXML offers library of UI controls, a mechanism for attaching actions to them, as well as facilities for asynchronous connection with the server.

There were many motivations for using Backbase in the present version of our framework. For example, it hides the complexity of developing AJAX applications, hides the incompatibilities between different Web browsers and platforms, hides the client/server communication complexities and achieves rich interactivity and portability for end users. Besides, the development is simplified by the excellent documentation and the flexibility to work in both Java and .Net.

7.2 Velocity Engine

Velocity [18] is a Template Engine developed in Java. It is a set of classes and not a different program, in other language. One of its greater utilities is in Web applications development, in which the Java code is totally separate from the HMTL code, which increases application's modularity and ease of maintenance.

For the Web development, the Velocity templates implement the View layer to the MVC pattern, that is, the visual layer. It has easy syntax, is fast and has caching configurations to make it still faster. Due to these advantages and to the fact that it is an open source engine, our framework has been developed using Velocity.

7.3 Labview Platform

Labview [19] is a development environment from National Instruments. Its main application area is on measurement and automation. Its programming language is based on dataflow, which is quite suitable for data acquisition and manipulation.

Labview also provides a Web interface which facilitates the integration of this kind of application in our proposed framework. Some applications using Labview have been developed especially for integration to our platform, but they get all the advantages provided by the framework without any need for framework recompilation or extra programming effort besides those in the Labview language.

8 Integration of Different Services

In this section we introduce three applications that we have integrated via the framework, consisting in a digital multimeter, an advanced Digital TV system and a cryptography application. They all have been implemented by other developers in our research group at *ComL@b*.

8.1 A Remote Multimeter

As part of our research effort on remote equipment monitoring and control, we have developed a Labview interface that controls a **Digital Multimeter Wavetek Metermann 235**. This equipment, besides all the functionalities of a Conventional Multimeter, includes a RS232 interface which allows the Digital Multimeter to be connected to a computer and to a sensor that reads temperature in real time. Monitoring temperature is very important in an industrial environment, but here we are mostly interested in the integration aspects of different services in our framework, for e-Science development.

8.2 An Advanced Digital TV System

A computational platform has been developed in Labview which allows the remote monitoring, control and configuration of different equipments connected to an Advanced Digital TV System developed at *ComL@b*. The set of equipments include a **Spectrum Analyzer** for real time spectral estimation of a digital TV channel; a **Digital Oscilloscope** for analysis and visualization of the signal constellation of the digital TV signal; a **Vector Signal Generator** for clock control and for synchronization of the transmission and reception systems; and finally a **Digital Frequencymeter** for clock frequency monitoring.

This platform has been integrated to our framework and today is part of our services available at *ComL@b*. We are now working in order to make this Digital TV experiment available to other university labs in Brazil, by means of the RNP/GIGA and KyaTera high-speed networks.

8.3 A Cryptography Service

This platform has been integrated to our framework and today is part of our services available at ComL@b. We are now working in order to make this Digital TV experiment available to other university labs in Brazil, by means of the RNP/GIGA and KyaTera high-speed networks.

We implemented the Message Digest Method, which is a compact digital signature for an arbitrarily long stream of binary data, which takes a long string of any length as input and produces a fixed length string as output. Two algorithms were implemented, the **SHA-1** and the **MD5**.

9 Conclusions

In this work we described the principles, methodologies and technologies used in a WebLab platform development. We also commented on a number of services that were implemented for validation and future use in collaborative work. This platform will be available for WebLab integration in FAPESP's KyaTera network, allowing the remote access to many engineering equipments, sensors, electronic devices and software platforms, which are essential to e-Science development in Latin America. Since it is not possible to predict all of the possible services, the adopted solution was based on plug-ins, which makes it possible to insert, remove and edit services without framework recompilation. With the use of plug-ins the system framework becomes much more extensible, robust and flexible. These characteristics were obtained by adopting many design patterns, especially DIP and EO patterns, which describe architecture conception techniques based on plug-ins.

Another important point considered in the framework development was the poor usability of the traditional Web applications. For this reason, we have chosen to use AJAX paradigm, which allows the development of rich interfaces. These interfaces have some resources that look like desktop resources, but here the user has no obligation to install some infrastructure software, like Java Web start or Applets solutions, which require Java execution environment installation.

Acknowledgements: This work has been supported by the TIDIA-KyaTera FAPESP Project (Grant Number 03/08264-5). We also thank CAPES for a Master of Science scholarship for the first author.

References

[1] Booth, D.; Haas, H.; McCabe, F.; Newcomer, E.; Champion, M.; Ferris, C. and Orchard, D. Web Service Architecture. Available from http://www.w3c.org/TR/ws-arch, February 2004.

[2] KyaTera's Web Pages. http://www.kyatera.fapesp.br.

[3] Eclipse Platform Technical Overview. Technical Report, IBM, www.eclipse.org/whitepapers/eclipse-overview.pdf, July 2001.

[4] Guimarães, E. G.; Maffeis, A. T.; Pereira, J. L.; Russo, B. G.; Cardozo, E. and Magalhães, M. F. REAL: A Virtual Laboratory for Mobile Robot Experiments. *IEEE Transactions on Education*, pages 46(1):37–42, February 2003.

[5] Guimarães, E. G. *Um Modelo de Componentes para Aplicações Temáticas e Ubíquas (in portuguese)*. PhD thesis, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas, SP, Brazil, 2004.

[6] Google Code. Available from http://code.google.com.

[7] Garrett, J. *AJAX: A new apprach to web applications*. Adaptive Path, 2005.

[8] D. Crane. *AJAX in Action*. Manning Publication, 2005.

[9] JSON-RPC. Available from http://json-rpc.org.

[10] Fowler, M. *Refatoração: aperfeiçoando o projeto de código existente (in portuguese)*. Bookman, 2004.

[11] Kuchana, P. *Software Architecture Design Pattern in Java*. Auerbach Publishers Inc, 2004.

[12] Metsker, S. J. *Design Pattern Java Workbook*. Addison-Wesley, 2002.

[13] Buschmann, F. et al. *Pattern-Oriented Software Architecture - A System of Pattern*. John Wiley & Sons, 1996.

[14] Gamma, E. and Beck, K. *Contributing to Eclipse Principles, Patterns, and Plug-ins*. Addison Wesley Professional, 2004.

[15] Crupi, J. et al. *Core J2Ee Patterns*. Prentice Hall PTR, 2003.

[16] Gamma, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[17] Backbase. Available from http://www.backbase.com.

[18] Velocity Engine. Available from http://velocity.apache.org.

[19] Labview Platform. Available from http://www.ni.com/labview.