Towards a Distributed Architecture for Remote Laboratories

J. García-Zubia[1], P. Orduña[2] , I. Angulo[1], J. Irurzun[1] and U. Hernández[1]
[1] Faculty of Engineering, University of Deusto, Bilbao, Spain
[2] Fundación Tecnológico Deusto, Bilbao, Spain

***Abstract*—Traditionally, Remote Laboratories have been focused on specific solutions for specific problems. We can find a wide range of Remote Laboratories in the literature [1], assisting very different types of subjects (electronics, robotics, optics, fluids mechanics...), but commonly bound to a restricted set of requirements. Because of this, little attention has been paid on working on a scalable, maintainable, secure, open architecture that addresses the requirements of a wide set of experiments, and that could be open enough to support or adapt itself to new experiments. In this paper, we describe several aspects that might be taken into account when designing a Remote Laboratory architecture, resulted from the iterative development of our Remote Laboratory.**

***Index Terms*—WebLab,Remote-Lab,SOA**

I. Introduction

A Remote Laboratory is composed of a client-side software and a server-side software.

The decisions taken for choosing among different client-side technologies (AJAX, simple HTML, Adobe Flash, Java Applets, Microsoft Silverlight, etc.) are critical [2][3] to allow or not the final users to use the Remote Laboratory on different browsers and different operating systems, as they are critical to enable certain degree of security, accessibility and user interface capabilities (3D, video, sound). These decisions will have an impact on the Remote Laboratory architecture [2] because depending on the client technology used, the server side will be able to interact with the client using different types of protocols (SOAP, REST, JSON, RMI, sockets, etc.).

However, other required factors (i.e. scalability, maintainability of the system, most of the security aspects, quality of service, etc.) of Remote Laboratories are bound to the server design and independent of the client side.

In this paper we illustrate the different requirements of a Remote Laboratory and the approach taken by our Remote Laboratory (WebLab-Deusto) to match these requirements.

II. Defining Requirements of Remote Laboratories

Examining the requirements of Remote Laboratories, there are important aspects that have to be taken into account when designing an architecture for Remote Laboratories. We have summarized these aspects in a set of questions that the designer of a Remote Laboratory architecture might consider, and grouped them into the following six categories:

A. Dependence on the type of experiment

○ How dependent on the nature of the experiment is the architecture?

○ Can experiments be shared with different users at the same time by time-division multiplexing? Most experiments can not be multiplexed (those that have some kind of state -a program in a device, a robot in a position, etc.-), but there other types of experiments that can be multiplexed (those only using digital electronics [3]). Can this technique be used in some experiments in the architecture?

○ Can't the experiments be shared and thus the users need time-based sessions? Does the Remote Laboratory assume a magnitude of time for each session (milliseconds, seconds, minutes, hours...)? Does the experiment have a scheduling system based on this magnitude (i.e. no waiting in the case of milliseconds, waiting in a queue for seconds or minutes, and having to schedule it for hours and days)?

○ How generic are the commands used to interact with the device? Does the Remote Laboratory assume a some types of experiments which receives certain type of information? Does the Remote Laboratory assume a size or frequency for these commands?

○ Does the Remote Laboratory use a video stream (i.e. from a webcam)? Does the Remote Laboratory assume a video quality for it?

B. Scalability

○ How many users does the architecture support in the highest peaks? How affordable is to increase that number?

○ Does the Remote Laboratory scale vertically (adding more resources -memory, CPUs- to a single node, the system supports proportionally more connections)?

○ Does the Remote Laboratory scale horizontally (adding more nodes, the system supports proportionally more connections)? Can the application be distributed along those different nodes? It is easier to add more nodes than to add more resources to a single node, but it is also usually more complex to implement a Remote Laboratory that scales horizontally that one which does not.

C. Maintainability

○ Does the architecture assume a single schema for the integration of the Remote Laboratory?

○ Can it be integrated in the IT Services of different universities? Can it use different schemas for this integration (supporting SSPI, LDAP, different database providers, etc.)? Different entities tend to use different

solutions for storing credentials and personal information of the users. Providing a pluggable authentication system is useful to integrate the Remote Laboratory in different schemas.

○ Does it support an advanced user management -involving multiple types of users with different privileges: users, professors, laboratories administrators, system administrator, etc.-? As the Remote Laboratory user and experiment base grows up, the administration tasks gets more complex. Not supporting different roles centralizes these tasks into a single role -administrator-, consuming time doing tasks that could be done by other maintainers that can not have so many privileges.

For instance, a professor who owns an experiment should be able to check the logs of the use of that experiment, but not the logs of other experiments. If there is no such role, the professor will need to contact the administrator to get those logs in a manual way so no automatic response is obtained, and if there are many experiments the administrator will be wasting too much time in maintainability tasks that could be automated. However, all those professors should not have administrator privileges, and although only certain privileges could be provided to each professor user, the creation of roles and groups can speed up the maintainability of the system.

○ Does it support an advanced log management? Can professors easily know how much do their students use the experiments they handle?

D. Security

○ Does the architecture take into consideration security in its design? It is important to take into account security issues during the whole process of software development, including its design. A vulnerable design can become difficult to secure in the latter stages of the development.

○ Does the Remote Laboratory avoid security flaws in the different modules of the system? Does the Remote Laboratory support secure communication protocols? Does the Remote Laboratory count with systems to avoid code injection (such as SQL/LDAP/XPath injection)? Does the Remote Laboratory store the passwords in a secure way?

○ Have security policies been established in the Remote Laboratory development?

E. Dependence on the protocol

○ Does the architecture assume a certain topology and bases the protocol decisions on that topology?

○ Does the Remote Laboratory assume that the different experiments and the application servers are in the same computer/room/building/city?

○ Does the Remote Laboratory support multiple protocols for different types of experiments, depending on the requirements of these protocols? For instance, an optimized binary protocol is more suitable for experiments which require real time feedback from the device, while it might have problems when dealing with firewalls or proxies. The decision depends on if real time feedback really is such a requirement.

○ Does the Remote Laboratory support multiple protocols depending on the security needed given the topology (using IPC -i.e. UNIX sockets-, or a dedicated network, a university private network, a public network)? Are authentication and encryption considered depending on the type of network?

F. SOA-compliant

○ Does the architecture match the Service Oriented Architecture?

○ Is the Remote Laboratory deployed as a service using a well known transport that can be consumed by other applications such as SOAP, REST or JSON? Or does it only support its own client (i.e. it only supports a web client)?

○ Can other services be built on top of the Remote Laboratory using a public interface?

III. WEBLAB-DEUSTO ARCHITECTURE

The software architecture of our Remote Laboratory has taken into account the previously explained.

A. Software architecture evolution

WebLab-Deusto had previously gone through two main iterations [4]:

- Version 1.0: The first approach of WebLab-Deusto that students actually used. The client was developed as a Jython applet (the user had to install the Java Runtime Environment in order to use the Remote Laboratory), while the server was a single Python application that could manage a single experiment. The communication between client and server was a proprietary socket-based protocol; this way the user could not be behind a HTTP proxy server and problems would arise if the user dealt with firewalls.
- Version 2.0: The second approach of WebLab-Deusto improved the client side of the project. The client side was rewritten using AJAX, and in the server side a new layer was written on top of the previous server so it managed SOAP messages instead of low level socket based messages. The user now could use the experiments even behind a HTTP proxy server and a firewall. Furthermore, since no software installation was required (since the client was purely written in AJAX), the user could use the Remote Laboratory from certain mobile devices [5]. Other enhancements, such as a cross-platform version of the laboratory, auxiliar applications to manage the Remote Laboratory, etc. were achieved in this branch through the different 2.x versions.

B. WebLab-Deusto 3.0 Requirements

As a result of the success of these previous iterations, new requirements came up and required a new Remote Laboratory architecture.

In terms of maintainability, WebLab-Deusto 2.0 still managed a single experiment. This way, in order to provide multiple experiments to the students it was necessary to deploy multiple instances of the Remote Laboratory, and maintain them independently.

In terms of flexibility, the design of WebLab-Deusto 2.0 was tied to the requirements of WebLab-Deusto 1.0, with only two different experiments. But future

deployment plans included a wide range of a dozen different experiments.

The number of students accessing the laboratory has also increased, so scalability had to be taken in the new design. Since the hardware was distributed along different laboratories of the Faculty of Engineering, the Remote Laboratory needs to support a complex deployment that will deal with heterogeneous networks that might include elements such as HTTP proxies, firewalls and untrusted networks between the different servers.

C. *WebLab-Deusto 3.0 Architecture*

In order to match the requirements explained above, the WebLab-Deusto 3.0 is based on a distributed architecture as shown in Fig.1.

1) *Logical architecture*

In this architecture, the clients connect to servers located in a server farm, maintained by the IT services of the University. A multitier architecture is applied, where the presentation tier is found in the client side, and the logic and database tiers are physically placed in this server farm.

The project currently supports MySQL 5 [6] for the database tier, Python is used in the logic tier, and an AJAX script (written with the Google Web Toolkit framework [7]) is used in the presentation tier. The servers in the logic tier will communicate with the authentication services found in other tier, which can currently be implemented using MySQL or LDAP.

Since the hardware is placed in laboratories that can be found in different buildings, the communication between the logic servers and the servers found in the laboratories should be secured, and the number of network addresses reserved by these servers should be minimized. On the other hand, in order to maximize the number of hardware experiments, it is necessary to reduce the cost of the devices that directly interact with the hardware. These devices might be micro servers, that will have direct access to the hardware but they will have limited resources.

In order to combine these cheap devices with the required resources per laboratory (secure communication between the logic server and the laboratory, minimized number of network addresses), we introduce a new tier that hides the micro servers to the logic servers, acting as proxies. This way, the communication between the logic server and these proxies can be secured, and each proxy will handle a number of micro servers that will directly interact with the hardware. The information communicated between the logic server and the proxy server will not contain sensitive information related to the users that are sending the information, but the information must be at least signed to avoid attackers using directly these proxy servers to interact with the hardware without credentials to do so.

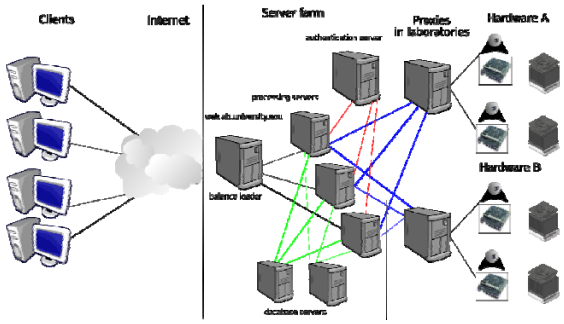

Figure 1: WebLab 3.0 Architecture

2) *Communications in the WebLab 3.0*

The problem with this architecture is that, although it matches the logic requirements, it demands many tiers. When the client sends a message to a device, it must be sent securely to the logic servers, that will redirect the message securely to a proxy in a laboratory that will redirect the message to a microserver that will redirect the message to the hardware. While this is fine in big deployments, depending on the experiment it can be optimized. If the hardware experiment is controlled though an application run in a computer, then the microserver tier should not exist. If this experiment is found in the same network where the logic servers is, the proxies tier should not exist.

Furthermore, depending on the requirements of the experiments and on the configuration of the deployment, the communication between these tiers will require to be optimized, using compressed proprietary protocols through TCP sockets instead of XML formatted data through HTTP.

To perform these optimizations, WebLab-Deusto 3.0 communications have been built on top of a pluggable system of protocols. Currently, only two protocols have been written, but new protocols will be added. These protocols are SOAP and "Direct", which calls the method name of the server in the same program instance. The decision of choosing between the different communication systems is handled through a communications broker. If a server wants to communicate with other server, it provides the WebLab-Deusto address of this server, and the communications broker will check what possible protocols can be used and it will automatically choose the fastest one. For instance, if two servers are located as different object instances in the same process, the "direct" protocol will be used, since it avoids the use of a network. If the two servers are located in different machines in the same network, it will use any network protocol, such as SOAP (the only network protocol implemented in the current version). If these two servers are in the same machine, an IPC protocol (such as UNIX sockets) is used, and if no IPC protocol is found (we don't support any IPC protocol under Microsoft environments) the communications broker would fall back to SOAP.

Because of this protocol-agnostic system, the Remote Laboratory can be configured in a very flexible way, supporting the avoidance of communications between different tiers if they are not necessary.

The communications with the client, though, only support SOAP at this moment, since sockets-based communications would not be directly supported by a

pure AJAX application. Anyway we are working on provide a sockets-based alternative for performance reasons. This sockets-based alternative is only an alternative since relying exclusively on it would introduce problems with HTTP proxies and firewalls.

3) Implementation notes

WebLab-Deusto 3 design is briefly drawn in Fig. 2. The logic software has been built on top of an WebLab-independent layer that can be reused for other projects. This WebLab independent layer supports the dynamically generated communications engine, as it does support some utility modules (for caching, logging, and sessions management). These modules are prepared to support an scalable architecture. For instance, the session management module supports storing the sessions in a MySQL database in order to make it possible to process a user request in a server and the next request in other server.

On top of it, the different servers have been implemented, as well as the hardware managers (called "Micro Servers" in Fig. 2). These hardware managers deserialize the messages created in the client and interact with a shared library of low-level hardware handlers such as "Xilinx Impact" or "Serial Port". The number of protocols, hardware managers and devices is going to be increased in the future.

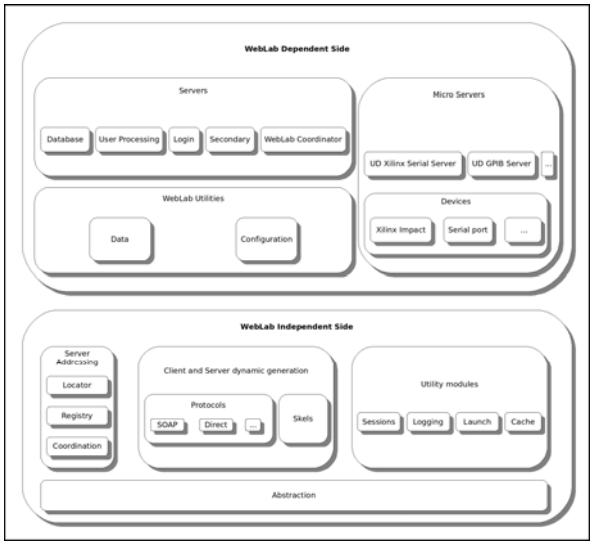

Figure 2: WebLab-Deusto 3.0 Design

D. Results

WebLab-Deusto has been successfully used by students in the University of Deusto since February 2005, along the three different versions. WebLab-Deusto v.3 started being used by students in October 2007, and since then it has been used in five different classes with four different experiments, deployed along three different rooms of the Faculty of Engineering with four different dedicated servers. Another experiment has already been developed and will be used during the next course. More experiments will be added during the next course.

IV. Conclusions and Future Work

Traditionally, Remote Laboratories have been focused on specific solutions for specific problems. This paper has shown several software aspects that should be considered before designing the architecture of a Remote Laboratory, and the benefits of it. Consequently, we have applied these aspects to our own WebLab in order to progress to a scalable, extensible and flexible Remote Laboratory.

For future work we plan to increase the number of experiments and devices, and we plan to add a socket based alternative in the client side. We also plan to add more protocols to the communications engine for performance reasons.

References

[1] C. Gravier, J. Fayolle, B. Bayard, M. Ates and J. Lardon, State of the Art About Remote Laboratories Paradigms - Foundations of Ongoing Mutations. International Journal of Online Engineering, Vol 4, No 1 (2008).

[2] J. Garcia-Zubia, P. Orduña, D. López-de-Ipiña, U. Hernández and I. Trueba. Section III - Remote labs development issues, Remote Laboratories from the Software Engineering point of view. July 2007. ISBN: 978-84-9830-077-2

[3] I. Gustavsson, J. Zackrisson, H. Åkesson, L. Håkansson, I. Claesson. and T. Lagö. Remote Operation and Control of Traditional Laboratory Equipment. International Journal of Online Engineering, Vol 2, No 1 (2006).

[4] J. García-Zubia, D. López-de-Ipiña, P. Orduña. Towards a Canonical Software Architecture for Multi-Device WebLabs. IECON 2005, 31st Annual Conference of the IEEE Industrial Electronics Society, ISBN: 0-7803-9253, pp. 2146-2151, November 2005.

[5] D. López de Ipiña, J. García-Zubia and P. Orduña. Remote Control of Web 2.0-enabled Laboratories from Mobile Devices. 2nd IEEE International Conference on e-Science and Grid Computing, eScience 2006, Amsterdam (Netherlands), December 2006, ISBN 0-7695-2734-5

[6] MySQL, http://www.mysql.com/

[7] Google Web Toolkit, http://code.google.com/webtoolkit/

Authors

J. García-Zubia is with the Faculty of Engineering, University of Deusto, Avda. De las Universidades 24, 48007 Bilbao, Spain (e-mail: zubia@eside.deusto.es).

P. Orduña, is with the Tecnológico Fundación Deusto, Avda. De las Universidades 24, 48007 Bilbao, Spain (e-mail: pablo@ordunya.com).

I. Angulo is with the Faculty of Engineering, University of Deusto, Avda. De las Universidades 24, 48007 Bilbao, Spain (e-mail: iangulo@eside.deusto.es).

J. Irurzun is with the Faculty of Engineering, University of Deusto, Avda. De las Universidades 24, 48007 Bilbao, Spain (e-mail: jaime.irurzun@gmail.com).

U. Hernández is with the Faculty of Engineering, University of Deusto, Avda. De las Universidades 24, 48007 Bilbao, Spain (e-mail: uhernand@eside.deusto.es).