Designing Experiment Agnostic Remote Laboratories

P. Orduña [1], J. García-Zubia [2], Jaime Irurzun [1], E. Sancristobal [3], S. Martín [3], M. Castro [3], D. López-de-Ipiña [2], U. Hernández [2], I. Angulo [2], J. M. González [2]

[1] DeustoTech – Tecnológico Fundación Deusto, Bilbao, Spain
[2] Faculty of Engineering, University of Deusto, Bilbao, Spain
[3] IEECD – Spanish University for Distance Education (UNED) – 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, assisting different types of subjects but commonly bound to a restricted set of requirements. Due to 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. Furthermore, the use of a technology is usually another requirement that any developer aiming to create a new experiment must fulfill. In this paper, we describe several aspects that might be taken into account when designing a Remote Laboratory architecture, describing the XWL Architecture.**

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

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 [1][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, and the XWL Architecture is described.

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 some types of experiments which receive 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

This section describes the architecture used in the WebLab-Deusto project, aiming the previously explained questions.

A. 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 [4] for the database tier, Python is used in the logic tier, and an AJAX script (written with the Google Web Toolkit framework [5]) is used in the presentation tier.

The servers in the logic tier will communicate with the authentication services found in other tier. Currently, these authentication services support three ways to verify the user credentials:

1. Using a username and a password in the MySQL database. This is intended for local users.
2. Through a remote LDAP server. The users of WebLab-Deusto can use the credentials they use in the Universities that support LDAP.
3. Based on the IP address of the client who logs in. This way, a WebLab-Deusto instance can trust in a set of servers of a third-party entity (such as other University) for a set of users. This third-party entity will sign up these users in the WebLab-Deusto instance, without a password.

   Then they can deploy an application in their trusted servers that starts sessions of these users in the WebLab-Deusto instance without a password, and provide the user the session ID. This way, the verification of credentials is delegated to this third-party application (such as a Moodle or .LRN plug-in), which will redirect the user to the WebLab-

Deusto instance transparently. The WebLab-Deusto instance will keep a log of the activities performed by this user so as to provide this information to the third-party application later. This type of authentication fits in architectures aiming to integrate Remote Laboratories with LMS such as [6].

If the servers of this third-party entity became compromised, the attacker would only be able to log in as the set of users signed up by that entity.

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 may need to 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. On it, a light Experiment Server -which is the only server in the WebLab-Deusto Architecture that is aware of the experiment itself- will run.

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), WebLab-Deusto introduces a new tier that hides the Experiment 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.

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 an Experiment Server that will redirect the message to the hardware. While this is fine in big deployments, depending on the experiment it may need to be optimized. If the hardware experiment is controlled though an application run in a computer, then the Experiment Server tier should not exist. If this experiment is found in the same network where the logic servers are, 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.

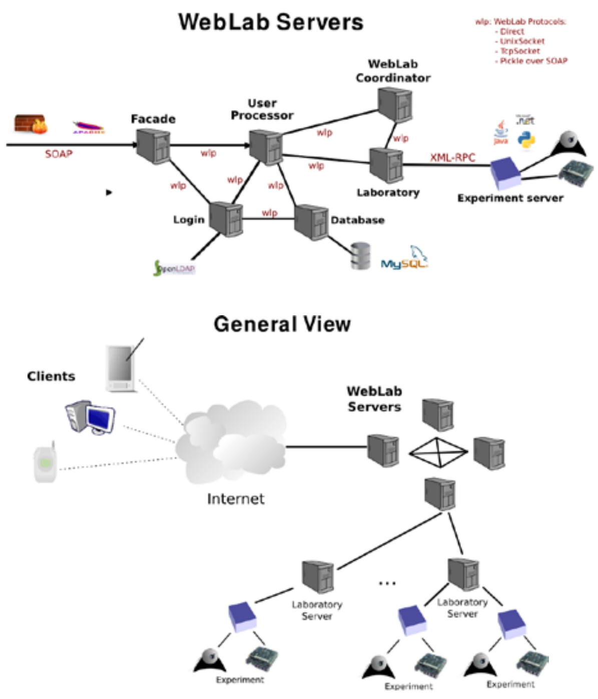


Figure 1: WebLab 3.0 Architecture

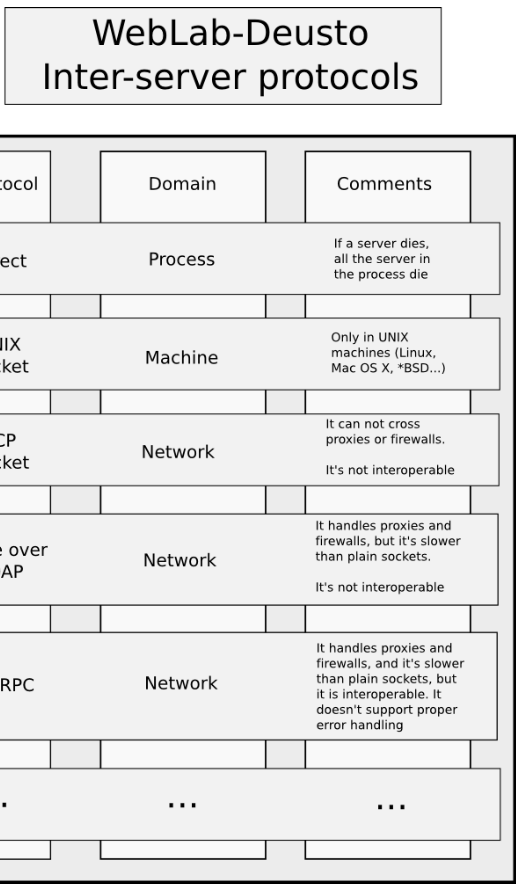
To perform these optimizations, WebLab-Deusto 3.0 communications have been built on top of a pluggable system of protocols. Currently, five protocols have been written, and new protocols could be added. These protocols are Python-dependent SOAP messages, TCP sockets, UNIX sockets, XML-RPC and “Direct”, which calls the method name of the server in the same program instance. The decision of choosing between the different communications systems is handled through a communications broker, parameterized by the system administrator. If a server tries to connect to 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:

- 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 the same machine and it is based on UNIX, it will use a UNIX socket.
- If the two servers are located in different machines but in the same network, or in the same machine in a not-UNIX environment (i.e. Microsoft Windows), it will use TCP sockets.
- If the system administrator defines that the server doesn’t support TCP sockets (i.e. if this server is behind an HTTP Proxy), it will use SOAP with Python-dependent messages.
- In order to communicate with the Experiment Servers, it can optionally use XML-RPC. The advantage of using XML-RPC in WebLab-Deusto is that the server can be implemented in other technology (such as Java or .NET), in opposite to the other communications protocols used.

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 advantages and disadvantages of each protocol are summed up in Figure 2.

The system administrator is responsible for deploying in a secure way. If the system is deployed with a single process running the whole system using “direct”, then if the Experiment Server code fails at process level, it may shut the whole server down. Or, if an attacker manages to exploit a vulnerability in a layer of the system and the Login Server is running with the same privileges, the attacker could access sensitive information such as passwords.

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.



WebLab-Deusto
Inter-server protocols

Protocol	Domain	Comments
Direct	Process	If a server dies, all the server in the process die
UNIX Socket	Machine	Only in UNIX machines (Linux, Mac OS X, *BSD...)
TCP Socket	Network	It can not cross proxies or firewalls. It's not interoperable
Pickle over SOAP	Network	It handles proxies and firewalls, but it's slower than plain sockets. It's not interoperable
XML-RPC	Network	It handles proxies and firewalls, and it's slower than plain sockets, but it is interoperable. It doesn't support proper error handling
...	...	...

Figure 2: WebLab-Deusto Inter-Server Protocols

B. *Adding new experiments: the XWL Architecture*

The aim of the WebLab-Deusto project is to develop a framework as independent as possible of the experiment itself, fulfilling the transversal requirements of Remote Laboratories. The developer of a new Remote Laboratory should be able to see WebLab-Deusto as a tool for developing the desired experiment without implementing the rest of the requirements.

In order to do so, the framework must be flexible in terms of required technologies. If the framework requires Java Applets, and the developer of a new Remote Laboratory does not count with experience in the development of Java Applets, or he wants to reuse legacy code developed using Adobe Flash, the developer will probably not use the framework. The same applies to the server side.

1) Developing the experiment

The approach of WebLab-Deusto for solving this challenge is embracing the XWL (eXtensible WebLab) Architecture. In this Architecture, the developer of the Remote Laboratory can use any web-based client technology that can interact with JavaScript in the client side –which includes both Adobe Flash and Java Applets, among others–, and any technology supporting XML-RPC –which includes any widely used technology– in the server side.

Once the developer has selected which technology is most suitable for the client side, he must write a small application that provides a user interface showing the experiment –basically the inputs and outputs of the experiment–, and that consumes an simple interface for sending commands (strings) to the server and retrieving responses from it (also strings). This interface is provided by the framework implementing the XWL Architecture, so the developer will not lead with communications. The application will be loaded by the framework as soon as the experiment is reserved, so the application will not implement any reservations management mechanism neither any authorization mechanism.

Then, the developer will need to implement the server side application that interacts directly with the hardware. The framework will call XML-RPC methods of this application to interact with it, sending programs and commands and receiving responses. This application will provide a method for starting the experiment, another for disposing its resources, and some methods to interact with the hardware. It does not need to manage reservations, concurrency or user-related information.

Both sides are communicated through the framework implementing the XWL Architecture. The framework does not understand the commands sent and received, since they are simple strings designed by the developer of the remote experiment. So it is responsibility of the developer of the remote experiment to properly serialize and deserialize those commands in both the server and the client side in order to use them.

2) Using the XWL-enabled framework

The framework implementing the XWL Architecture will provide the rest of requirements. It manages the authentication of the user, the communications between client and server, the administration management (including logs of the activities of the users), issues related with scalability and security, and the management of resources. In WebLab-Deusto, these features are described in Fig. 3.

This way, when the user accesses the system through the framework, it checks the credentials and permissions of the user, and returns the available experiments for this user. Then the user chooses what experiment he wants to use, and the system will manage the queues or the required schedules to check that there is actually a free experiment deployed that he can use.

Once the experiment is assigned to the user, in the client side the application developed by the developer of the experiment will appear, inside the XWL client (see Fig. 4). When the user presses buttons or sends files in this application, the application will delegate to the system sending them to the application in the server side. The system will do this as configured by the System Administrator, but the developer of the experiment is not aware of how.

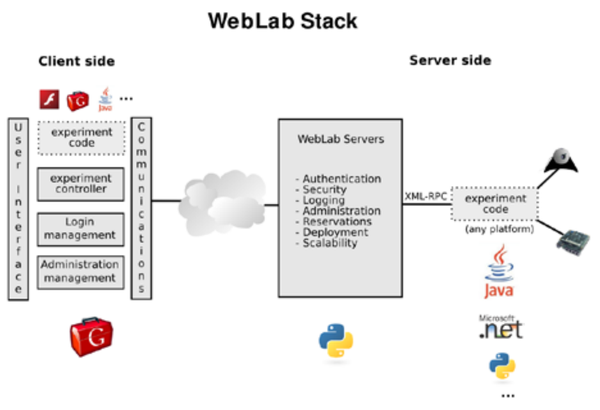


Figure 3: The XWL Architecture: WebLab-Deusto stack

This way, the developer of the experiment delegates most requirements to the system, and gets benefits transparently from all the new features that system implementing the XWL Architecture may add without changing the code.

3) *Further considerations*

The XWL framework should be implemented using AJAX, so the platform does not require any plug-in installation to the end user.

However, if the developer of an experiment selects Java Applets as client side technology, the users of that experiment will need to have the Java plug-in installed. Anyway, the users of the rest of the experiments will not need Java.

When selecting the client side technology, the developer of the Remote Experiment must be aware of this kind of problems, which have already been studied in [2]. While from the point of view of the WebLab-Deusto team AJAX is a proper technology for the Remote Laboratories development, under certain circumstances other technologies might fit better. This is the main rationale for supporting other technologies in the XWL Architecture.

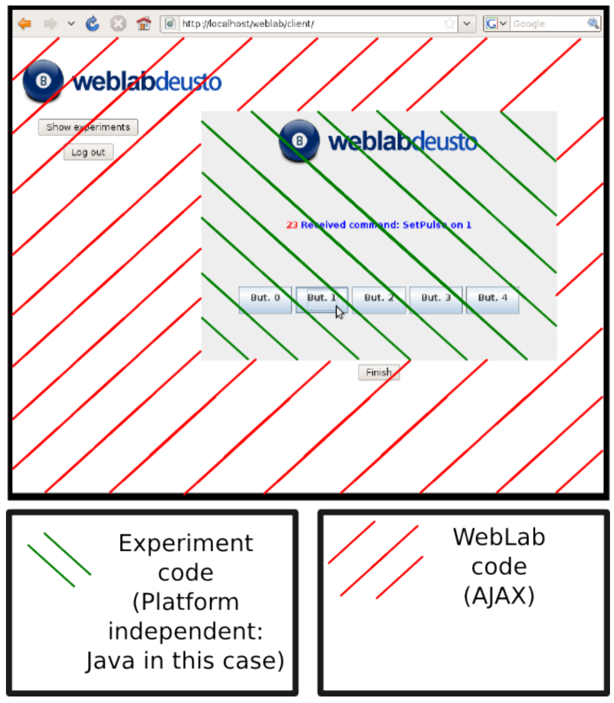

http://localhost/weblab/client/
weblabdeusto
Show experiments
Log out
weblabdeusto
23 Received command: SetPulse on 1
But. 0
But. 1
But. 2
But. 3
But. 4
Finish
Experiment code (Platform independent: Java in this case)
WebLab code (AJAX)

Figure 4: Java Applet implementing a dummy experiment inside the XWL-enabled WebLab-Deusto client

C. *Results*

WebLab-Deusto, implementing the XWL Architecture, 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.

IV. Conclusions and Future Work

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 communicate the Remote Laboratory with Learning Management Systems so as to reuse all the features they provide for online education. We also 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.

Acknowledgment

The authors would like to acknowledge to the Spanish Science and Innovation Ministry for the support in the project TIN2008-06083-C03/TSI “s-Labs – Integración de Servicios Abiertos para Laboratorios Remotos y Virtuales Distribuidos”

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

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

[6] E. Sancristobal, S. Martín, R. Gil, C. Martínez, E. López, N. Oliva, F. Mur, G. Díaz, M. Castro. Development and Interaction between LMS Services and Remote Labs. International Journal of Online Engineering (iJOE), Vol 4, No 3 (2008).

Authors

P. Orduña, is a Computer Science Engineer by the University of Deusto, Bilbao 2007. Nowadays he is a Research Assistant at the Ambient Intelligence department of DeustoTech - Tecnológico Fundación Deusto, and a PhD student of the University of Deusto, and his research is focused on Remote Laboratories. He is the lead software designer and developer of WebLab-Deusto. (e-mail: pablo@ordunya.com).

J. García-Zubia is with University of Deusto, Electronics and Automation Department, Avenida de las Universidades 24, 48007 Bilbao (Spain), is with the University of Deusto, he is Head of Dpt. Of Industrial Electronics, Control Engineering, and Computers Architecture of the Faculty of Engineering. He is the responsible of the remote lab at the University of Deusto (WebLab-DEUSTO: weblab.deusto.es). The WebLab-Deusto has been implemented using web 2.0 techniques (AJAX, SOAP, etc.), this approach is a novelty in Europe. Different works have been published explaining the results and the technology of this weblab and the evolution of WebLab-DEUSTO has been supported by different projects. (e-mail: zubia@eside.deusto.es).

J. Irurzun is a student of the last year of Computer Engineering in the Faculty of Engineering of the University of Deusto. He is with the WebLab-Deusto research group since 2007. (e-mail: jaime.irurzun@gmail.com).

E. Sancristobal, San Cristobal Ruiz, Elio is a Computer Science Engineer by the Salamanca Pontifical University (UPS), Madrid 2002, has a Technical Engineering degree in computer networks (UPS), Madrid 1998. He finished his Doctoral Studies in Electronics Technology by the Electrical and Computer Engineering Department from the Spanish University for Distance Education (UNED), Madrid 2005. He worked for the University Distance Education Institute (IUED). Nowadays is working for the Computer Science Service Centre of the Spanish University for Distance Education (UNED) and is an associate professor Electrical and Computer Engineering Department from UNED. (e-mail: elio@ieec.uned.es).

S. Martín, is Computer Engineer from Carlos III University of Madrid (UC3M), Distributed Applications & Systems speciality, receiving Honour marks in his final project: “Applications Manager based on location by wireless lan". Technical Computer Engineer from the Madrid Polytechnic University (UPM). He is now studying for his PhD in Educative Technologies in the Electrical and Computer Engineering Department (DIEEC) of the Spanish University for Distance Education (UNED). (e-mail: smartin@ieec.uned.es).

M. Castro, Electrical and Computer Engineering educator in the Spanish University for Distance Education (UNED), has an industrial engineering degree from the ETSII (Industrial Engineering School) of the Madrid Polytechnic University (UPM) and a doctoral engineering degree from the same University. He received the Extraordinary Doctoral Award in the UPM and the Viesgo 1988 Award to the Doctoral Thesis improving the Scientific Research about the Industrial Process Electricity Application. He works as researcher, coordinator and director in different projects, ranging from solar system and advanced microprocessor system simulation to telematics and distance learning systems, acting now as and senior technical director. He is now with the UNED (Spanish University for Distance Education) as Professor in the Electronics Technology subject inside the Electrical and Computer Engineering Department as well as he is Director of the Department. (e-mail: mcastro@ieec.uned.es).

D. López de Ipiña is the Director of SmartLab (http://www.smarlab.deusto.es), Principal Investigator in the Mobility Research Lab (http://www.morelab.deusto.es) and Head of the ICT Unit of Tecnológico Fundación Deusto (http://www.tecnologico.deusto.es). He received his PhD from the University of Cambridge, U.K in 2002. His main research areas of interest are pervasive computing middleware, human-environment-interaction and mobile pervasive web for the Internet of Things. (e-mail: dipina@eside.deusto.es).

U. Hernández is with the University of Deusto in the Telecommunications Department at the Faculty of Engineering. He is developer of the research group on web-based laboratories and he is in charge of the remote labs based on instruments control. He is involved too in the deployment in University of Deusto of the VISIR project leaded by the Blekinge Institute of Technology (Ronneby, Sweeden). (e-mail: uhernand@eside.deusto.es).

I. Angulo is with the University of Deusto in the Dpt. of Industrial Electronics, Control Engineering, and Computers Architecture of the Faculty of Engineering . He is the lead hardware designer and developer of WebLab-Deusto. (e-mail: iangulo@eside.deusto.es).

J. M. González is a Computer Engineering student of the University of Deusto. He is involved in the software development of the WebLab-Deusto project (e-mail: jose.gg88@gmail.com).