Addressing Software Impact in the Design of Remote Laboratories

Javier García-Zubia, *Member, IEEE*, Pablo Orduña, *Member, IEEE*, Diego López-de-Ipiña, and Gustavo R. Alves

***Abstract*—Remote Laboratories or WebLabs constitute a first-order didactic resource in engineering faculties. However, in many cases, they lack a proper software design, both in the client and server side, which degrades their quality and academic usefulness. This paper presents the main characteristics of a Remote Laboratory, analyzes the software technologies to implement the client and server sides in a WebLab, and correlates these technologies with the characteristics to facilitate the selection of a technology to implement a WebLab. The results obtained suggest the adoption of a Service Oriented Laboratory Architecture-based approach for the design of future Remote Laboratories so that client-agnostic Remote Laboratories and Remote Laboratory composition are enabled. The experience with the real Remote Laboratory, WebLab-Deusto, is also presented.**

***Index Terms*—eLearning, remote laboratories, service-oriented architecture (SOA), web services.**

I. Introduction

THE SHIFTING of paradigms in education from a "faculty-centric" to a "student-centric" teaching approach [1] is in line with the Bologna Declaration and remarks the "learning by doing" using laboratories [2]. Currently, Remote Laboratories or WebLabs have clearly shown their academic usefulness [3]–[5], not only to substitute the real physical laboratories but rather, also to complement and power them, although it has also been pointed out that Remote Laboratories might not always be as good [6]. Anyway, the first Remote Laboratories [7] or WebLabs [8], [9] were organized and promoted by a laboratory or department, but their success has motivated the universities themselves to manage them. This change supposes an acknowledgment of the importance of WebLabs, but it also introduces new challenging requirements (security, accessibility, universality, and so on) which often are disregarded in the original design, but they are essential to constitute truly professional services.

A WebLab has to manage three different objectives: educational, organizational, and technological. Although there are several actors involved in the development and provision of WebLabs, due to its implicit association with education and the institutions that host them, technology plays a central role in all that happens in a WebLab [10]. This paper will be focused on the technological issues and, in particular, on the importance of software in the design process of WebLabs.

A common wrong approach is to design first a prototype which works—it works!—and then worry about adding new features. Unfortunately, this is not a valid approach since oftentimes, there is a need to redo the whole application. This is something bearable by a computer scientist but not so obviously affordable by other type of engineers.

For example, in the remarkable work of [11], more than 100 articles around WebLabs are examined. However, in only one of them [12] is a software given proper importance. All the other works focus on hardware and academic aspects. In the only software-related paper, the importance of adopting cutting-edge Web-related technologies [Web 2.0 and Web Services/ Service Oriented Architecture (SOA)] to produce better quality Remote Laboratories is stressed.

In the same way, the International Journal on Online Engineering (http://www.ijoe.org) was created in 2005, and since then, eight issues have been published in the field of Remote Laboratories and Remote Engineering. Analyzing the 67 papers published, none of them is related with the analysis of the different software strategies that can be used in a Remote Laboratory.

In December 2007, [13] explained how a service-oriented approach can be applied to a Remote Laboratory for robotics.

In conclusion, very few researchers have worked on the software aspects of Remote Laboratories. However, many Remote Laboratory researchers coincide in pointing out the increasing importance of adding universality, accessibility, security, or capability combination capacities to their WebLabs [14]. This is only possible if there is a bigger focus on software, both on the client and server sides of the WebLabs. In consequence, this paper analyzes the relative importance of these two different software sides and also conjectures about the benefits of adopting a Service Oriented Laboratory Architecture (SOLA) approach in the design of future remote WebLabs.

Sections II and III analyze several technologies for implementing the client and the server, respectively, justifying the choice of asynchronous Javascript and XML (AJAX) and Python in each case. This selection is established correlating the technologies with the main characteristics of a WebLab, seeing that some characteristics can only be reached using specific technologies. Section IV describes the SOLA, which is proposed for the development of future WebLabs. In Section V, the experience and the usefulness of the real Remote Laboratory, WebLab-Deusto, are presented.

Manuscript received March 30, 2007; revised June 17, 2009. First published June 30, 2009; current version published November 6, 2009.

J. García-Zubia and D. López-de-Ipiña are with the Faculty of Engineering, University of Deusto, Bilbao, Spain (e-mail: zubia@eside.deusto.es).

P. Orduña is with the DeustoTech (Fundación Deusto), Bilbao, Spain.

G. R. Alves is with the Department of Electrical Engineering, School of Engineering, Polytechnic Institute of Porto (IPP), 4200-072 Porto, Portugal.

Digital Object Identifier 10.1109/TIE.2009.2026368

II. Client Side

The client side in a Remote Laboratory is the software that the user of such laboratory employs. Depending on the experiment, this client software may need to send a file to the Remote Laboratory server, show a real-time video of what is happening in the actual laboratory, present a file with the results of the experiment, allow the interaction and telecontrol with the equipment of an experiment, or provide other functionalities.

An essential feature of such clients should be not to set unnecessary restrictions on the user. Thus, providing all the functionalities through a universal Web application (a Web application that, while being capable to match Remote Laboratories requirements, can be accessed by the users wherever they are, under different hardware and software platforms) may be a much better option than providing a stand-alone application which requires a lot of software to be installed.

This section is organized to select the most suitable client technology to implement a Remote Laboratory. Sections II-A–D present the client technologies, the main characteristics of a Remote Laboratory, and the criteria to choose among the five technologies analyzed: Java applets, Adobe Flash, AJAX, HTML, and ActiveX.

A. Classification of Technologies

A wide range of technologies can be applied to the development of Remote Laboratory clients, from the lightest Web-based ones to the heaviest stand-alone desktop-based ones. Hence, client applications could be classified into two groups:

1) *Desktop clients:* those run in the user's desktop computer;
2) *Web-based clients:* those accessed by a browser in the user's desktop computer.

A desktop application is very flexible and powerful, it can be developed in many languages (C, C++, Java, .NET, Delphi, Python. . .) and over different platforms, and it may have few restrictions. However, those applications are less portable and more intrusive than the Web-based applications, and they are just regular applications that the user launches; many are programmed for one concrete operating system and usually demand an installation process. Anyway, the quality of a desktop application depends on itself. The most remarkable feature of desktop applications is the flexibility they provide; they are usually more flexible and powerful than Web-based applications. Since, in principle, they do not usually have restrictions, the designer can explore some novel possibilities, such as making use of 3-D graphics or integrating them in the user's desktop. This is something Web applications usually cannot provide.

This paper will focus mostly on Web applications, since they provide two more essential features that desktop applications do not offer, i.e., more portability and less intrusiveness. Under this point of view, client-development technologies can be classified into two categories.

1) *Intrusive applications*. Regular desktop applications and some forms of Web-based applications are intrusive, since they require complete-access privileges. For instance, they usually can access the client's hard disk, read any file in the computer, or open as many connections to the outside world as a user can. Anyway, a desktop application built in Java or .NET might be nonintrusive, but they are not common.
2) *Nonintrusive applications*. Those which warranty the user that the application is not going to access any system resources which may damage the hosting machine. This way, the user can safely run the application without worrying about security or privacy because the application will not be able to read the information from any file of the hard disk that the user does not explicitly choose; it will not be able to introduce any kind of virus in the system, and so forth.

The main problem with intrusive applications is security. Applied to a Remote Laboratory developed and hosted by a university, the students will download the client application from the server of the Remote Laboratory, having to trust the following agents:

1) the Remote Laboratory development team;
2) the server where the client software and experiments are hosted has not been tampered with;
3) the network they are using to download the application is secure enough.

If any of these aspects fail, someone may, in fact, be breaking into the students' computers, and perhaps, as a side effect, the university will have some responsibility in that. Consequently, nonintrusive applications are obviously clearly preferred in terms of security.

Considering Fig. 1, it can be said that the more powerful and intrusive a technology is, the less universal it becomes.

In Table I, experts from different universities have ordered ten characteristics: 1 is associated to the main characteristic and 10 to the least. Obviously the priorities are different for each center and even for each researcher, but universality is better considered than power.

In Table I, the opinion of the IT Services of the University of Deusto must be highlighted because if the WebLab is aimed to be offered by the university, it has to fulfill the requirements imposed by the IT Services, as Moodle does, for example. Otherwise, the WebLab will fail to be a professional educational tool.

The characteristics analyzed in Table I are the following.

1) *Cross-platform.* The WebLab can be accessed by all the operating systems (OS): Windows, Linux, Mac OS, etc.
2) *Security.* The WebLab uses HTTP, does not need permissions on the firewalls, only needs the 80 and 443 ports opened, etc.
3) *Web browsers.* The WebLab can be accessed by all the Web browsers: Explorer, Mozilla, Opera, Safari, etc.
4) *Intrusivity.* The user does not give permissions to the WebLab client application: hard-disk access, execution of native code, etc.
5) *Interaction.* The WebLab needs to implement the maximum of interaction with the user.
6) *Installation.* The WebLab runs without any previous installation in the client side: plug-in, JVM, Flash Player, etc.

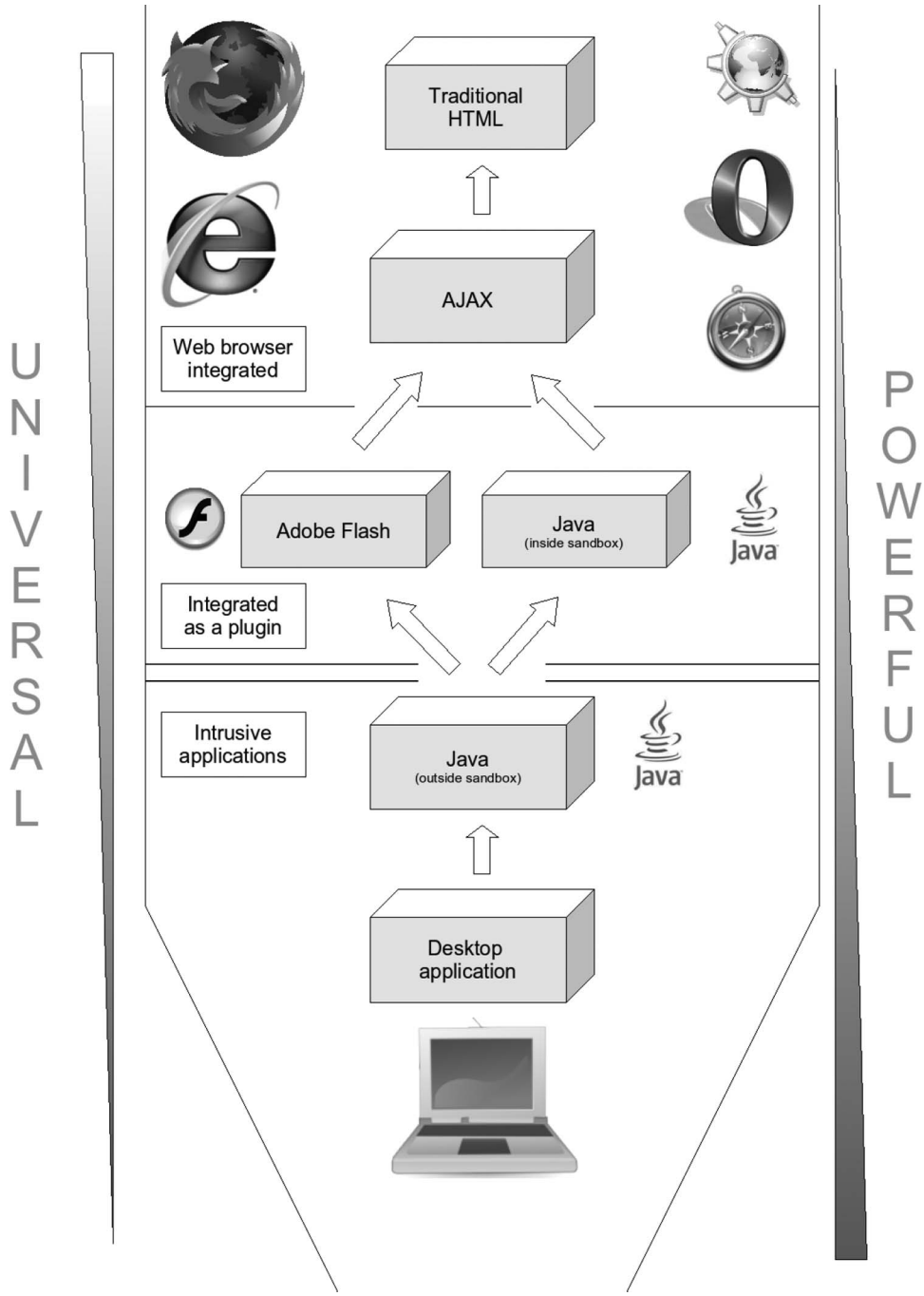


Fig. 1. Technology classification in the client side.

7) *Devices.* The WebLab can be accessed by all the devices: PC, PDA, mobile phone, etc.
8) *Bandwidth.* The WebLab needs the maximum bandwidth efficiency.
9) *Audio and Video.* The WebLab needs the maximum of audio and video power.
10) *Power.* The WebLab is very complex and needs a powerful tool to be implemented.

The rest of this section analyzes the different client technologies available to select one of them. It also justifies why some advanced features should be left out to promote a higher degree of universality.

B. Choosing Communication Technologies

Another problem is choosing the technology that will make the communications between the client and server possible. The main question is that in Remote Laboratories, the client and the server can be located in different networks, trying to cross through firewalls and proxies, and non-HTTP-based protocols might find it impossible to cross them. Inside this group of non-HTTP-based technologies, it is possible to find versatile technologies as CORBA, Java RMI [15], .NET Remoting, or even TCP/IP sockets [16], while Web Services would be placed in the HTTP-based technologies group.

TABLE I
OPINION OF EXPERTS ABOUT CHARACTERISTICS OF REMOTE LABORATORIES(1)

	E(2)	E(3)	E(4)	E(5)	E(6)	E(7)	E(8)	E(9)	Total
Cross-platform	2	4	3	1	2	1	1	1	**15**
Security	1	2	5	3	5	2	2	3	**23**
Web browsers	3	5	4	2	3	3	4	2	**26**
Intrusivity	4	1	8	4	1	10	5	5	**38**
Interaction	6	7	2	7	6	4	3	6	**41**
Installation	5	3	9	6	4	5	7	4	**43**
Devices	10	6	10	5	7	9	6	7	**60**
Bandwidth	8	9	1	8	9	7	9	10	**61**
Audio&Video	9	8	6	9	8	6	8	8	**62**
Power	7	10	7	10	10	8	10	9	**71**

(1) E: Experts
(2) IT Services of the University of Deusto.
(3) *Deusto:* Javier Garcia-Zubia at the University of Deusto (Spain), coordinator of WebLab-Deusto.
(4) *BTH:* Ingvar Gustavsson at Blekinge Institute of Technology (Sweden), coordinator of the VISIR project.
(5) *Artec:* Dieter Müller at Artec-Lab at the University of Bremen, coordinator of (1) MARVEL project.
(6) *ISEP:* Gustavo Alves at Instituto Superior da Engenharia of Porto (Portugal), coordinator of Rex-Net project.
(7) *Genoa:* Andrea Bagnasco at the University of Genoa (Italy), coordinator of isiLAB.
(8) *MIT*: Jesús A. del Álamo is the coordinator of iLAB at the M.I.T. (EE.UU.)
(9) *EPFL*: Denis Gillet is responsible of the Remote Lab at the EPFL.

In fact, some Remote Laboratories have been built on top of CORBA [17], [18]. The problem is that these technologies are restricted to local networks, and its use is not easily applied to the Internet. A designer who uses these non-HTTP-based technologies must assume several security considerations since the deployment of the Remote Laboratory will demand modifications in the firewall configuration.

This is why Web Services are, in general, a more suitable technology for the implementation of Remote Laboratories [19], [14]. The main drawback of Web Services is performance: Non-HTTP-based technologies tend to be faster than Web Services, which might be important in real-time applications.

C. List of Client Technologies

Different client technologies are described next before being analyzed in detail.

Desktop Applications: Desktop applications are mostly intrusive applications because they are based in a particular protocol, security, etc., for example [20]. Thus, these applications cannot be analyzed in general, and they will not be taken into account in the next sections.

ActiveX: Java and ActiveX are probably the most powerful systems in terms of flexibility among the Web-application technologies, but ActiveX only runs under Microsoft Internet Explorer, and its applications are intrusive applications (although they ask the client to confirm for permissions to access system resources). These facts make ActiveX-based applications closer to desktop applications than to pure Web applications.

Anyway, an unarguable fact is that Microsoft Internet Explorer and Microsoft Windows are widely used (76.33% in http://www.thecounter.com, on December 2008), making ActiveX suitable for developing applications (including Remote Laboratories) with a high index of availability.

LabVIEW: A person knowledgeable in LabVIEW can create a remote accessible VI by simply pushing in "Web Publishing Tool." He is not required to understand Web technologies to do so. There is a lot of literature specifically referring to this approach, particularly in the fields of control engineering and electronics [21]–[24].

However, the remote control in LabVIEW is based on Microsoft ActiveX, so, from a technological point of view, LabVIEW is the same as ActiveX.

Java Applets: Java is a well-known technology for designing WebLabs [25]–[29] because it is a powerful platform to develop Rich Internet Applications (RIAs).

To use Java, the client needs to have the Java Runtime Environment (JRE) installed. The good point of the JRE is that it can be installed in many OS, and it can be embedded in multiple Web browsers. The bad point of the JRE is its availability: There are different versions of it, and if the designer develops the Java client (known as Java applets) for JRE 1.5, it will not run in the client's machine if it has JRE 1.4 installed.

Another availability problem is that, since Java applets are not such a popular technology anymore, the user of the application will have to download the JRE and install it before running the application. This can be a real problem if the client is in a restricted computer (such as a cybercafé or probably the computers of the university, where he does not count with administrator's privileges).[1]

An interesting point of Java is that when an applet is running, it runs in a *sandbox*: It is not, by default, an intrusive application. It does not have access to the hard disk, it cannot establish connections to other computers (except for the server which provided the applet), and so on. The problem is that when the experiment requires the user to send a file, the sandbox cannot handle the request as it implies accessing the hard disk. In this situation, the designer has to choose between sending the file in other technology (like basic HTML) or avoiding the sandbox (turning the applet into an intrusive application). Another solution is to develop a mixed application (using both technologies), but although it is possible to call Java applets' methods from Javascript and Javascript functions from Java, this is not usual because, in general, Java is discarded and a more modern technology is used. It is better to choose another technology or, if there are key reasons to use Java, then just escape from the sandbox or sign the Java applet. However, if the WebLab needs an automatic recognition of the applet certificate, the signing organization—the university—must pay for a certification made by a certification authority.

Adobe Flash: Adobe Flash (formerly called Macromedia Flash until December 2005) is now the leading technology for RIAs. The user of an Adobe Flash has to install the Adobe Flash Player, which will interpret byte codes found in files in the SWF format. Once the Adobe Flash Player is installed, the applications made in Adobe Flash will be nonintrusive cross-platform applications with many capabilities: video, real-time video, audio, development in ActionScript, access to Web Services, and even access to files in a nonintrusive way (when accessing a file, the user chooses the file). The potential Adobe

[1]For instance, Steve Jobs in the presentation of iPhone commented "Nobody uses Java applets anymore," January 2007, http://pogue.blogs.nytimes.com/2007/01/13/ultimate_iphone_faqs_list_part_2/.

Flash has for graphics and animations, as well as to access Web Services, providing a nonintrusive approach, makes it suitable for developing Remote Laboratories [30].

The use of Adobe Flash is widely spread, and it is available under many platforms (Microsoft Windows, Linux, Mac OS). Anyway, this availability is relative because today, no version is supported under 64-bits architecture, which is quite a big drawback. Furthermore, version 7 has been the only one supported under Linux until mid-January 2007.

AJAX: AJAX [31] is the combination of several existing Web technologies (XHTML, Javascript, CSS, DOM...) with a new component, i.e., XMLHttpRequest. This component allows calling asynchronously XML Web Services from Javascript. This is why AJAX is an acronym for asynchronous Javascript and XML.

The big point of AJAX is that all the components, except for XMLHttpRequest, are standards that the Web browsers already support. Therefore, if any Web browser implements this new component, AJAX applications will automatically work in that Web browser.

This is a very interesting issue since this makes AJAX the most portable platform, of all the ones explained up to this point, that supports interactivity with the server, even in a Remote Laboratory [32], [33]. There are many implementations of this set of technologies under most platform and architectures since wherever there is a Web browser, AJAX applications are going to run. This way, even Web browsers for mobile devices, such as the Opera mobile Web browser in many mobile devices, the latest versions of Microsoft Internet Explorer for Windows CE, or the new open-source Web browser that Nokia includes in many of their devices, support AJAX. Therefore, with no extra effort at all, AJAX applications will run even in mobile devices [34].

Big companies such as Google or Yahoo started releasing their new advanced Web applications in AJAX, like Google Maps, Google Mail, or Flickr. Since then, many platforms for AJAX development were released, so, AJAX is now being used in many Web applications.

AJAX itself does not provide video or audio capabilities [35]. For small videos with no sound where a slow frame rate may do the job, refreshing an image could be enough, and this way, many Remote Laboratories could be completely based on AJAX, but so far, only WebLab-Deusto has been implemented [14]. Anyway, if the Remote Laboratory needs high-resolution video and audio capabilities, the application must integrate a specific function based in Adobe Flash, for example.

Traditional HTML Applications: Traditional HTML applications are Web applications which only use the classic well-known Web standards such as XHTML, HTML, CSS, etc. It does not have, by default, any capability of interaction with the server, video, or audio. Anyway, if the Web page follows Web standards, it will work under any standard compliant Web browser.

Furthermore, there is much work being placed on Web accessibility (based on Web standards), making it possible to develop an accessible Web application that will allow disabled people to use the Web page [32], [36]—there are laws that regulate the accessibility of some information services; for Spain, see [37] and [38]. This is something quite difficult to do with all the previously mentioned technologies, except for Adobe Flash which has provided, since Flash Player version 6, accessibility functions for developers to use.

JavaFX and Microsoft Silverlight: Since 2007, both Sun Microsystems and Microsoft have developed two new platforms, JavaFX and Microsoft Silverlight, as direct competitors to Adobe Flash for RIA development.

These technologies are too new to be analyzed in this paper, although it can be stated that the marks given to them will be similar to the ones given to Adobe Flash. For example, Microsoft Silverlight has been released from the beginning under both Microsoft Windows and Mac OS X and under different Web browsers.

D. Choosing Technology for the Client

The question to address after explaining these technologies is: Which technology is most appropriate to develop a Remote Laboratory client? A possible answer will always depend upon the criteria favored by the designer placing the question, e.g., a traditional HTML application is better than an AJAX application, an AJAX application is better than an Adobe Flash application, and an Adobe Flash application is better than a Java applet, if he/she favors characteristics such as "availability," "portability," or "accessibility," or rather the opposite answer if he/she favors other characteristics such as "network protocols," "bandwidth efficiency," or "price."

Tables II–V summarize the possibilities of the technologies for designing the client. There are 17 characteristics analyzed in the tables, and they have been selected using the experience gathered in Remote Laboratories since 2001 in the design, development and use of the WebLab of the University of Deusto [14], [20], [34] (http://weblab.deusto.es). WebLab-Deusto has evolved in four versions: desktop application (v0.1), Java-applet-based Web applications (v1.0), and two AJAX-based Web application (v2.0, v3.0). These reflections were discussed in the International Meeting on Professional Remote Laboratories in 2006 [39].

The 17 characteristics have been grouped in four issues: universality, security, power, and development. In the rest of the section, each characteristic is associated with a mark in the range 1–5.

1) Universality: Is the client accessible without any restriction?
   - *Paradigm:* In which grade does the technology match the current paradigm for new rich applications?
   - *Cross platform:* Does the application run under different OS?
   - *Availability:* How often is the technology available in the client system?
   - *Accessibility:* How accessible is the technology for disabled people?
   - *Acceptance by Web browsers:* Is the technology part of the Web browser?
2) Security/Standards: Is the client secure and/or based on standards?

TABLE II
ANALYSIS OF CLIENT-SIDE TECHNOLOGIES IN TERMS OF UNIVERSALITY

Characteristic	Technology					
Paradigm (1)	Java Applets	■	■			
	Adobe Flash	■	■	■	■	
	AJAX	■	■	■	■	■
	HTML	■	■	■	■	
	ActiveX	■				
Cross-platform (2)	Java Applets	■	■	■	■	
	Adobe Flash	■	■	■		
	AJAX	■	■	■	■	■
	HTML	■	■	■	■	■
	ActiveX	■				
Availability (3)	Java Applets	■	■			
	Adobe Flash	■	■	■	■	
	AJAX	■	■	■	■	■
	HTML	■	■	■	■	■
	ActiveX	■	■	■	■	■
Accessibility (4)	Java Applets	■	■			
	Adobe Flash	■	■	■	■	
	AJAX	■	■			
	HTML	■	■	■	■	■
	ActiveX	■	■			
Acceptance by Web Browsers (5)	Java Applets	■				
	Adobe Flash	■				
	AJAX	■	■	■	■	■
	HTML	■	■	■	■	■
	ActiveX	■	■	■	■	■
Universality	**Java Applets**	**11**				
	Adobe Flash	**16**				
	AJAX	**22**				
	HTML	**24**				
	ActiveX	**14**				

1. The use of Java Applets and ActiveX is decreasing in RIA.
2. ActiveX only runs under Windows, Flash is not supported by 64 bit architecture and Sun does not support Java in all the architectures, i.e. PowerPC.
3. Flash Player is now more commonly found than JVM. AJAX and HTML are integrated in the Web Browsers, while ActiveX is integrated only in Internet Explorer, which is not mandatory although it is available in more than 76.33% of the computers.
4. Flash provides some accessibility features, while the rest do not. HTML directly provides support for accessibility.
5. AJAX and HTML are intrinsically implemented by the Web Browser, while Java Applets and Adobe Flash must be installed as a plug-in for the Web Browser. ActiveX is only part of the browser in Microsoft Internet Explorer.

Intrusiveness: Are permissions asked to the user for accessing the hard disk, establishing connections, and so on?

Standardization: In which grade is the technology based on standards?

Installation required: Does the application require software installation such as virtual machines or players?

Network protocols: What network protocols are available in the technology?

3) Power: How powerful can the client become?

Audio and video: How powerful are the audio and video capabilities provided with this technology?

Bandwidth efficiency: How good is the technology in terms of bandwidth efficiency?

Flexibility: Have the technology capabilities for developing applications under different contexts?

Mobile devices: How suitable is the technology for being used in PC, PDA, cellular phones, etc?

4) Development: What facilities does the technology offer for client developments?

TABLE III
ANALYSIS OF CLIENT-SIDE TECHNOLOGIES IN TERMS OF SECURITY

Characteristic	Technology					
Intrusiveness (1)	Java Applets	■	■	■	■	■
	Adobe Flash	■	■	■	■	■
	AJAX	■	■	■	■	■
	HTML	■	■	■	■	■
	ActiveX	■				
Standardization (2)	Java Applets	■	■	■	■	
	Adobe Flash	■	■	■		
	AJAX	■	■	■	■	
	HTML	■	■	■	■	■
	ActiveX	■	■			
Installation required (3)	Java Applets	■	■			
	Adobe Flash	■	■	■		
	AJAX	■	■	■	■	■
	HTML	■	■	■	■	■
	ActiveX	■	■	■	■	■
Network protocols (4)	Java Applets	■	■	■	■	■
	Adobe Flash	■	■	■	■	■
	AJAX	■	■	■	■	
	HTML	■	■			
	ActiveX	■	■	■	■	■
Security/ Standards	Java Applets	**16**				
	Adobe Flash	**16**				
	AJAX	**18**				
	HTML	**17**				
	ActiveX	**13**				

1. In Java Applets, if the developer tries to work out of the sandbox, the application would be intrusive. Otherwise it would not.
2. The format of the files used by Adobe Flash is not publicly available, in contrast to the format of the Java Applet files.
3. Java Applets and Adobe Flash are plugins. While the former needs to install the whole JVM, the latter only needs a thinner runtime. ActiveX does not require any installation and it is available in more than 95% of the computers, but it only runs under Microsoft Windows.
4. AJAX adds basic network capabilities to HTML through the XMLHttpRequest object. Java applets, Adobe Flash and ActiveX can establish binary sockets with the server. Usually, Remote Laboratories implemented using binary sockets do have problems with firewalls and proxies.

Development tools: Are there powerful tools for working with the technology?

Price: What is the cost of the tool for users and developers?

Providers: How independent are the users and developers from a single provider?

Network of developers: How big is the network of developers using the technology?

Fig. 2 shows the marks obtained in Tables II–V.

The following can be deduced by analyzing numerically the results of Fig. 2.

1) AJAX is numerically the most valued technology.
2) Looking at the most important aspects, AJAX is also more valued (see Table VI).
3) If the application needs audio or high quality video, at least Adobe Flash is required.
4) If interaction is required, as usual in Remote Laboratories, traditional HTML must be discarded.
5) Java Applets are similar to Adobe Flash in most of the issues, but they lose in terms of availability.
6) ActiveX is not recommendable for Remote Laboratory development because it does not provide anything useful that the other technologies cannot provide, and it presents problems in terms of availability in different platforms.

TABLE IV
ANALYSIS OF CLIENT-SIDE TECHNOLOGIES IN TERMS OF POWER

Characteristic	Technology					
Audio and video	Java Applets					
	Adobe Flash					
	AJAX					
	HTML					
	ActiveX					
Bandwidth efficiency (1)	Java Applets					
	Adobe Flash					
	AJAX					
	HTML					
	ActiveX					
Flexibility (2)	Java Applets					
	Adobe Flash					
	AJAX					
	HTML					
	ActiveX					
Mobile devices (3)	Java Applets					
	Adobe Flash					
	AJAX					
	HTML					
	ActiveX					
Power	Java Applets	**14**				
	Adobe Flash	**17**				
	AJAX	**12**				
	HTML	**8**				
	ActiveX	**17**				

1. The use of binary sockets might improve the network efficiency, although its use can introduce problems with firewalls and proxies.
2. The capabilities provided by Windows to ActiveX are more powerful and flexible than the ones provided by the JRE or by the Flash Player. The capabilities provided by a Web Browser for AJAX or HTML are even less powerful.
3. Any device with a Web Browser (like the Opera web browser, Nokia OSS Web Browser, etc.) will support both AJAX and HTML Remote Labs. The solution provided by Adobe (Flash Lite) is not suitable for a wide range of mobile devices.

TABLE V
ANALYSIS OF CLIENT-SIDE TECHNOLOGIES IN TERMS OF DEVELOPMENT

Characteristic	Technology					
Development tools (1)	Java Applets					
	Adobe Flash					
	AJAX					
	HTML					
	ActiveX					
Price (2)	Java Applets					
	Adobe Flash					
	AJAX					
	HTML					
	ActiveX					
Providers (3)	Java Applets					
	Adobe Flash					
	AJAX					
	HTML					
	ActiveX					
Network of developers (4)	Java Applets					
	Adobe Flash					
	AJAX					
	HTML					
	ActiveX					
Development	Java Applets	**18**				
	Adobe Flash	**13**				
	AJAX	**20**				
	HTML	**20**				
	ActiveX	**12**				

1. There are many tools for developing RIA with HTML, AJAX and Java Applets. The problem with Flash development is that it is coupled to the editor provided by Adobe.
2. The user does not need to pay for the Adobe Flash player, but the developers will have to pay if using the editor provided by Adobe to create the Remote Laboratory, although there are free alternatives. ActiveX is free for both users and developers, but it requires Microsoft Windows, which is not free.
3. There is only one provider for both Adobe Flash and ActiveX (Adobe and Microsoft), while we can find more providers for Java Applets (Sun Microsystems, IBM, etc.) and even more for AJAX and HTML (Microsoft, Mozilla, Opera, Apple, Nokia, etc.).
4. There is a big network of developers sharing knowledge and resources for each technology.

For a specific WebLab, the designers can select the requirements of their WebLab in Tables II–V and analyze them or perhaps add new characteristics to the table or weigh them up/down. The marks shown in the tables are clear and important, but the conclusion behind them is that the technology selected for the client will establish some irreversible limits in the WebLab.

Table VI shows the comparison between Adobe Flash and AJAX for the development of WebLab-Deusto. The most suitable technology for the WebLab-Deusto requirements is AJAX.

Anyway, among all the technologies considered, the approach that is experiencing a faster growth is, by far, the AJAX approach. More and more, particularly inside the so-called Web 2.0, new Internet applications are using AJAX as the technical engine of the client software. The advantages it provides in terms of availability, independence from a unique provider, fast load speed, and integration inside traditional Web pages, make it very suitable to be seen as the first technology to use when interaction in a Web page is needed. The main drawback of AJAX for Remote Laboratory development is that it does not directly provide audio or high-quality video capabilities, which can be provided by adding an Adobe Flash application or Java applet, which supports this. Since both Adobe Flash and Java applets are interoperable with AJAX, the integration of these technologies in an AJAX application can become trivial. Google Mail, for instance, is a complete AJAX application which supports online conversations, and it uses a little Adobe Flash application for playing sounds each time someone sends a message. Everything in Google Mail, except for these sounds, will work on a Web browser without Adobe Flash.

III. SERVER SIDE

Although a very important part of the Remote Laboratory is the client and the technologies associated to it, the biggest part of the project is, for sure, the server side, but a good design of the server side does not depend so clearly on the technology used. The characteristics associated to the client cannot be applied to the server.

1) If the WebLab uses Web Services for the communication, the client technology will be independent from the technology used in the server. Thus, it is possible to implement the client of a Remote Laboratory in AJAX, Adobe Flash, etc., while the server side is implemented in any server technology.
2) The technology in the client side forces every single user to assume dependencies in terms of plug in, cross platform, Web browser, etc., but the dependencies forced by the technology in the server side only have to be assumed by the system administrator, and their effects do not affect to the final users.
3) The requirements in terms of security are very different between the server and the client. The server must control

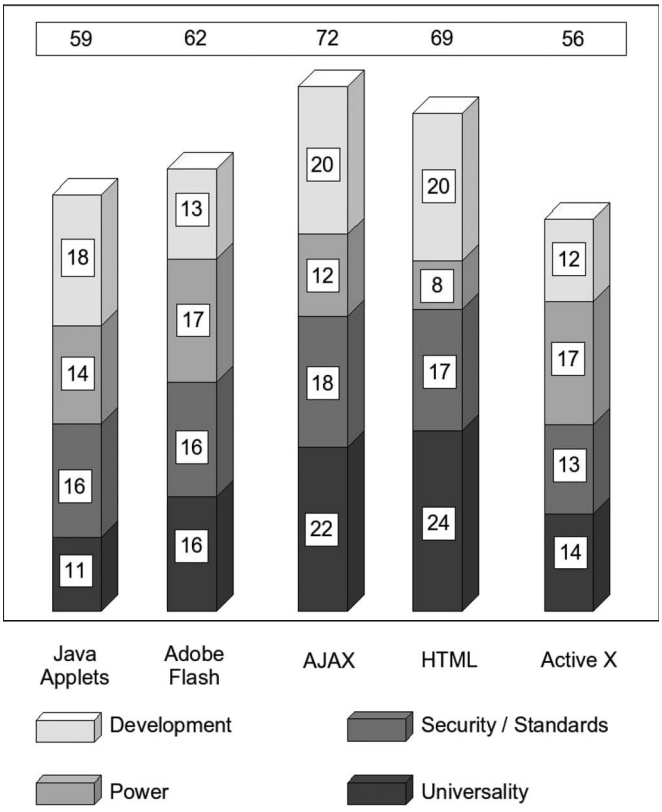


Fig. 2. Comparison between different client technologies.

the authentication, authorization, integrity, and privacy of the communication and must avoid attacks. These issues can be reached with all the technologies. The security of the server depends more on the architecture than in the technology.

4) All the servers are intrinsically intrusive, and this is not a problem because it only affects the servers.
5) In the client side, five main technologies were analyzed, but in the server side, there are many: Python, .NET, Java, Perl, C++, PHP, Visual Basic, Ruby, etc., and all of them can be used to implement a Remote Laboratory with any client technology. Moreover, it is common to integrate different technologies in the server side (Java + Oracle, PHP + Apache + MySQL, etc.), a situation which is not usual in the client side.

Summarizing, universality is mainly related with the client, and security on the server side can be implemented with all the technologies. The analysis will be focused on the power of the development tools.

Table VII summarizes the analysis of the three server technologies: Java, .NET, and Python; the first two are the most popular server technologies—Indrusiak *et al.* [40] use Java, iLab-MIT uses .NET—and Python is a dynamically typed language used for quick prototyping. The main conclusion of Table VII is that all the technologies are suitable for Remote Laboratories. The final decision falls on the research group.

The chosen technology for the WebLab-Deusto development has been Python because: 1) It is a very powerful dynamically typed programming language, which has a strong open-source community in its background; 2) it allows very fast development, being very suitable for rapid prototyping; and 3) it is being used internally in Google, Yahoo, Industrial Light & Magic, NASA, and other important companies, thus, demonstrating its practicality.

TABLE VI
ANALYSIS OF CLIENT-SIDE TECHNOLOGIES FOR WEBLAB-DEUSTO

Adobe Flash		AJAX
■ ■ ■ ■ □	Paradigm	■ ■ ■ ■ ■
■ ■ ■ □ □	Cross-platform	■ ■ ■ ■ ■
■ □ □ □ □	Acceptance by Web Browsers	■ ■ ■ ■ ■
■ ■ ■ ■ ■	Intrusivity	■ ■ ■ ■ ■
■ ■ ■ □ □	Installation required	■ ■ ■ ■ ■
■ ■ ■ ■ ■	Audio and video	□ □ □ ■ ■
■ ■ ■ □ □	Mobile devices	□ ■ ■ ■ ■
■ ■ ■ ■ □	Development tools	■ ■ ■ ■ ■
28	**Marks**	**36**

TABLE VII
ANALYSIS OF SERVER-SIDE TECHNOLOGIES

Characteristic	Technology					
Cross-platform	Python	■	■	■	■	■
	.NET (1)	■	■	□	□	□
	Java	■	■	■	■	□
Development tools	Python	■	■	□	□	□
	.NET	■	■	■	■	■
	Java	■	■	■	■	■
Development speed	Python (2)	■	■	■	■	■
	.NET	■	■	■	□	□
	Java	■	■	■	□	□
Network of developers	Python	■	■	■	■	■
	.NET	■	■	■	■	■
	Java	■	■	■	■	■
Robustness	Python (2)	■	■	■	□	□
	.NET	■	■	■	■	■
	Java	■	■	■	■	■
Web Services libraries	Python	■	■	□	□	□
	.NET	■	■	■	■	■
	Java	■	■	■	■	■
Language functionalities	Python (3)	■	■	■	■	■
	.NET	■	■	■	□	□
	Java	■	■	□	□	□
Price	Python	■	■	■	■	■
	.NET (1)	■	■	□	□	□
	Java (4)	■	■	■	■	□
Marks	Python	**32**				
	.NET	**30**				
	Java	**33**				

(1) If the Remote Laboratory works under Mono, license costs will be decremented and it will be able to be used under different platforms. In this case the mark will be 5.
(2) As Python is a dynamically typed programming language it is oriented to rapid although less robust development.
(3) Python provides high functionalities included in the language itself. The same functionalities must be programmed by the developers in Java or .NET.
(4) It depends on the tools and the framework used.

IV. SOLA

A final important aspect regarding the importance of software for the design of better Remote Laboratories is not just the software technology itself but the paradigm adopted for the design and implementation of a Remote Laboratory. Lately, it has become commonplace adopting an SOA in the design of novel distributed applications [41], as it has been done in related European projects such as SOCRADES

(http://www.socrades.eu/Home/default.html). The main feature of such an approach is that it enables reuse and fosters modularity, composability, componentization, and interoperability, by promoting the cooperation of loosely coupled collections of unrelated Web services. Other remarkable benefits of the SOA approach are its compliance to standards (both common and industry specific) and the capacity of identifying and categorizing services to ease searching and composition of them [42].

WebLabs are good candidates to be designed following the SOA approach. After all, they are not more than a software service whose implementation is based on actual hardware. However, their functionality can easily be abstracted as a set of remotely accessible methods. Thus, it could be beneficial adopting a SOLA, i.e., an adaptation of the commonly known SOA to the Remote Laboratory domain, in the design of future Remote Laboratories. Consequently, in those newly designed laboratories, the functionality offered for a given Remote Laboratory would be seen as a set of Web services. Some of those services, when referring to the same type of functionality, should offer a compatible or identical interface to foster cooperation among different Remote Laboratories. Thus, SOLA is a an emerging standard, refining the SOA concept for standard enterprise services, for connecting distributed Remote Laboratories to the SOA in the enterprise. An important distinctive feature of the SOLA approach is that the default SOA functionality needs to be coupled with event-driven, real-time (strict performance guarantees), and distributed service-scheduling features to enable a feasible cooperation between the distributed functional blocks of Remote Laboratories that may be assembled to compose sophisticated, previously infeasible within a single organization, Remote Laboratory experiments.

Adapting and exporting the functionality of a Remote Laboratory as a set of Web Services would allow developers to design and implement client applications (desktop or Web-based) that combine the functionality of several WebLabs through Web-service composition. Thus, the creation of very sophisticated experiments would be possible by concatenating the outputs of one hardware experiment as inputs of another one, and, consequently, independently of who offered such services as long as clients had access rights to them. Therefore, another important aspect of the SOLA approach apart from the event-driven, real-time, and scheduling demands aforementioned is the need to put in place a security and trust mechanism among the different SOA-aware Remote Laboratory.

In conclusion, the adoption of a SOLA approach would decouple the client and server parts of a Remote Laboratory. Then, the server side would be completely agnostic to the clients consuming its functionality. It would only provide a common Web Services Description Language-Application Programming Interface accessible through a distributed communication standard such as Simple Object Access Protocol (SOAP), which will be used by third-party client applications to mash up the functionality of previously unrelated WebLabs.

V. WebLab-Deusto Experience

The University of Deusto has implemented the WebLab-Deusto, http://weblab.deusto.es, as a Web service using SOAP, AJAX, and Python [13]. Only one other project [32] has been found using an AJAX approach, too. WebLab-Deusto has four different versions:

1) v 0.1 Desktop application implemented in C. 2001;
2) v 1.0 Web application implemented in Java. 2004;
3) v 2.0 Web Application implemented in AJAX. 2005;
4) v 3.0 Web application implemented in AJAX. 2007.

WebLab-Deusto has been used in three subjects of the Faculty of Engineering: Programmable Logic, Electronics Design, and Electronics Instrumentation, by 200 students per year since 2003. The questionnaire of Table VIII shows the acceptance of WebLab-Deusto by the students of Programmable Logic and Electronics Design. The minimum mark is 1, and the maximum is 5.

TABLE VIII
WebLab-Deusto Academic Results

Questions	(1)	(2)	(3)	(4)	(5)	(6)
1. Has WebLab helped you with the subject?	4.6	3.8	3.75	4.1	3.8	3.7
2. Did you feel that you were in a better position by having been in the WebLab group?	4.7	3.9	3.7	3.9	3.7	3.8
3. Do you think it is a good idea if this WebLab experiment is extended to all the students?	4.7	4.2	4.1	4.6	4.1	4.1
4. Is it easy to use?	4.4	3.9	3.9	4.4	3.7	4.2
5. What is the quality of the WebCam like?	3.2	2.7	2.5	2.4	3	3.3
6. Did you feel at ease managing the inputs?	3.7	3.0	3.1	3.1	3.5	3.2
7. What do you think about the time assigned to each connection?	3.7	3.1	2.4	2.7	3.2	4.0
8. What do you think about the inputs/outputs implemented?	3.8	3.4	3.5	3.2	3.4	3.8
9. Being far from the prototype, have you felt you were in control of it?	4.1	3.6	3.7	3.7	3.6	3.7
10. Would you like to use WebLab in other subjects?	4.3	3.9	4.1	4	3.8	3.6
11. What is your global satisfaction with WebLab?	4.7	3.7	4	3.9	3.7	3.6

(1-3) Results in 2004/2005, 2005/2006 and 2006/2007 for the subject "Programmable Logic".
(4-6) Results in 2005/2006, 2006/2007 and 2007/2008 for the subject "Electronics Design".

VI. Conclusion

Using the experience obtained developing WebLab-Deusto since 2001, this paper has analyzed different strategies to develop a WebLab from the software point of view—server and client sides—avoiding specifically the hardware side.

The client technologies can be classified in terms of power and universality. It can be said that the more powerful a technology is, the less universal it becomes. This paper has established that some requirements can only be reached with a specific technology. According to Table I, the universality of a WebLab client is more important than its power. Using the results of Tables II–V, the most ideal technology for Remote Laboratory client development is AJAX, particularly if universality is the goal of the WebLab.

The scenario and the criteria to select the technology for developing the server side is not like those used in the client side. The option that suites better the requirements of

WebLab-Deusto is Python because of its rapid prototyping cycle and open-source nature. However, this fact is not such a clear result as the one considering the client side.

Finally, it is suggested that a good future direction will be to adopt a SOLA approach in the design and development of loosely coupled new WebLabs which are agnostic to the clients accessing them and enable composition for the creation of more sophisticated WebLab experiences.

REFERENCES

[1] J. Biggs, *Teaching for Quality Learning at University*. Maidenhead, U.K.: Open Univ. Press/McGraw-Hill, 2003.

[2] L. Carlson and J. F. Sullivan, "Hands-on engineering: Learning by doing in the integrated teaching and learning program," *Int. J. Eng. Educ.*, vol. 15, no. 1, pp. 20–31, 1999.

[3] N. Ertugrul, "New area in engineering experiments: An integrated and interactive/learning approach, and real-time visualizations," *Int. J. Eng. Educ.*, vol. 14, no. 5, pp. 344–355, 1998.

[4] J. García-Zubia, U. Hernández, D. Ponta, P. Orduña, I. Trueba, and I. Angulo, "WebLab-GPIB at the University of Deusto," in *Proc. REV Remote Eng. Virtual Instrum.*, 2007, 74_paper.pdf.

[5] J. E. Corter, J. V. Nickerson, S. K. Esche, and C. Chassapis, "Remote versus hands-on labs: A comparative study," in *Proc. 34th ASEE/IEEE Frontiers Educ. Conf.*, 2004, pp. F1G17–F1G21.

[6] O. Soysal, "Computer integrated experimentation in electrical engineering education over distance," in *Proc. ASEE Annu. Conf.*, Saint Louis, MO, Jun. 2000.

[7] M. F. Aburdene, E. J. Mastas, and R. Massengale, "A proposal for a remotely shared control systems laboratory," in *Proc. IEEE/ASEE Frontiers Educ. Conf.*, 1991, pp. 589–592.

[8] C. M. Boroni, F. W. Goosey, M. Grinder, R. J. Ross, and P. Wissenbach, "WebLab! A universal and interactive teaching, learning, and laboratory environment for the world wild web," in *Proc. 28th SIGCSE Tech. Symp. Comput. Sci. Educ.*, San Jose, CA, 1997, pp. 199–203.

[9] B. Aktan, C. A. Bohus, L. A. Crowl, and M. H. Shor, "Distance learning applied to control engineering laboratories," *IEEE Trans. Educ.*, vol. 39, no. 3, pp. 320–326, Aug. 1996.

[10] N. A. Hine, G. R. Alves, H.-H. Erbe, D. Müller, J. B. da Mota Alves, C. Pereira, J. M. Ferreira, L. E. Sucar, O. A. Herrera, L. Chiang, and J. G. Zubia, "Institutional factors governing the deployment of remote experiments: Lessons from the REXNET project," in *Proc. REV Remote Eng. Virtual Instrum.*, 2007, 66_paper.pdf.

[11] J. Ma and J. V. Nickerson, "Hands-on, simulated, and remote laboratories: A comparative literature review," *ACM Comput. Surv.*, vol. 38, no. 3, pp. 1–24, 2006.

[12] S. Kolberg and T. A. Fjeldly, "Web Services remote educational laboratories," in *Proc. Int. Conf. Eng. Educ.*, Gainesville, FL, 2004, pp. 1–6.

[13] J. Fernandez, R. Marin, and R. Wirz, "Online competitions: An open space to improve the learning process," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3086–3093, Dec. 2007.

[14] J. García-Zubia, D. López-de-Ipiña, P. Orduña, U. Hernández, and I. Trueba. (2006). Questions and answers for designing useful WebLabs. *Int. J. Online Eng.* [Online]. *II(3)*. Available: www.ijoe.org

[15] R. Marin, P. J. Sanz, P. Nebot, and R. Wirz, "A multimodal interface to control a robot arm via the web: A case study on remote programming," *IEEE Trans. Ind. Electron.*, vol. 52, no. 6, pp. 1506–1520, Dec. 2005.

[16] H. Hassan, C. Dominguez, J. M. Martinez, A. Perles, and J. Albaladejo, "Remote laboratory architecture for the validation of industrial control applications," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3094–3102, Dec. 2007.

[17] F. He, F. Wang, W. Li, X. Han, and J. Liu, "Object request brokers for distributed measurements," *IEEE Comput. Appl. Power*, vol. 14, no. 1, pp. 50–54, Jan. 2001.

[18] E. Guimarães, A. Maffeis, J. Pereira, B. Russo, E. Cardozo, M. Bergerman, and M. F. Magalhães, "REAL: A virtual laboratory for mobile robot experiments," *IEEE Trans. Educ.*, vol. 46, no. 1, pp. 37–42, Feb. 2003.

[19] A. Bagnasco, M. Chirico, A. M. Scapolla, and E. Amodei, "XML data representation for testing automation," in *Proc. IEEE AUTOTESTCON*, 2002, pp. 577–584.

[20] J. García-Zubía and A. del Moral, "Suitability and implementation of a WebLab in engineering," in *Proc. 10th Int. Conf. ETFA*, Catania, Italy, Sep. 2005, vol. II, pp. 49–56.

[21] L. Costas-Pérez, D. Lago, J. Fariña, and J. J. Rodriguez-Andina, "Optimization of an industrial sensor and data acquisition laboratory through time sharing and remote access," *IEEE Trans. Ind. Electron.*, vol. 55, no. 6, pp. 2397–2404, Jun. 2008.

[22] G. R. Alves, J. M. Ferreira, D. Muller, H.-H. Erbe, J. B. M. Alves, C. E. Pereira, E. Sucar, O. Herrera, N. Hine, and L. Chiang, "Remote experimentation network—Yielding an inter-university peer-to-peer e-service," in *Proc. 10th IEEE Int. Conf. ETFA*, Catania, Italy, Sep. 2005, pp. 1023–1030.

[23] W. Hu, G.-P. Liu, D. Rees, and Y. Qiao, "Design and implementation of web-based control laboratory for tests rigs in geographically diverse locations," *IEEE Trans. Ind. Electron.*, vol. 55, no. 6, pp. 2343–2354, Jun. 2008.

[24] B. Hercog, B. Gergic, S. Uran, and K. Jezernik, "A DSP-based remote control laboratory," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3057–3068, Dec. 2007.

[25] F. Davoli, G. Spanò, S. Vignola, and S. Zappatore, "LABNET: Toward remote laboratories with unifed access," *IEEE Trans. Instrum. Meas.*, vol. 55, no. 5, pp. 1551–1558, Oct. 2006.

[26] A. Mittal, C. Gupta, and A. Gupta, "Addressing the bandwidth efficiency, control, and evaluation issues in software remote laboratories," *IEEE Trans. Ind. Electron.*, vol. 55, no. 6, pp. 2326–2333, Jun. 2008.

[27] M. Bertocco, F. Ferraris, C. Offelli, and M. Parvis, "A client server architecture for distributed measurement systems," *IEEE Trans. Instrum. Meas.*, vol. 47, no. 5, pp. 1143–1148, Oct. 1998.

[28] A. Ferrero, S. Salicone, C. Bonora, and M. Parmigiani, "ReMLab: A Java-based remote, didactic measurement laboratory," *IEEE Trans. Instrum. Meas.*, vol. 52, no. 3, pp. 710–715, Jun. 2003.

[29] J. Sánchez, S. Dormido, R. Pastor, and F. Morilla, "A Java/Matlab-based environment for remote control system laboratories: Illustrated with an inverted pendulum," *IEEE Trans. Educ.*, vol. 47, no. 3, pp. 321–329, Aug. 2004.

[30] I. Gustavsson, "A remote access laboratory for electrical circuit experiments," *Int. J. Eng. Educ.*, vol. 19, no. 3, pp. 409–419, 2003.

[31] L. D. Paulson, "Building rich web applications with Ajax," *Computer*, vol. 38, no. 10, pp. 14–17, Oct. 2005.

[32] F. Gobbo and M. Vaccari, "Open standards for higher education in robotics by immersive telelaboratories," *Learn. Technol. Newslett.*, vol. 7, no. 3, pp. 30–32, 2005.

[33] I. M. Atkinson, D. du Boulay, C. Chee, K. Chiu, T. King, D. F. McMullen, R. Quilici, N. G. D. Sim, P. Turner, and M. Wyatt, "CIMA based remote instrument and data access: An extension into the Australian e-Science environment," in *Proc. 2nd IEEE Int. Conf. e-Sci. Grid Comput.*, Amsterdam, The Netherlands, Dec. 2006, p. 125.

[34] D. López-de-Ipiña, J. García-Zubia, and P. Orduña, "Remote control of Web 2.0-enabled laboratories from mobile devices," in *Proc. 2nd IEEE Int. Conf. e-Sci. Grid Comput.*, Dec. 2006, p. 123.

[35] J. Emigh, "New flash player rises in the Web-Video market," *Computer*, vol. 39, no. 2, pp. 14–16, Feb. 2006.

[36] M. Cooper, "Accessibility and usability in complex web based learning applications: Lessons from the PEARL project," in *Proc. Corp., Gov., Health, World Conf. E-Learn. Higher Educ.*, 2002, pp. 1358–1365.

[37] *Boletín Oficial de las Cortes Generales*, no. 68-13, Jul. 3, 2002.

[38] *Boletín Oficial del Estado*, no. 289, Dec. 3, 2003.

[39] L. Gomes and J. Garcia-Zubia, Eds., *Advances on Remote Laboratories and e-Learning Experiences*. Bilbao, Spain: Univ. Deusto, 2007.

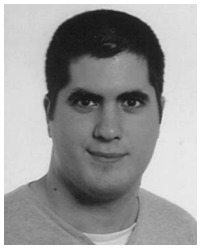
[40] L. S. Indrusiak, M. Glesner, and R. Reis, "On the evolution of remote laboratories for prototyping digital electronic systems," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3069–3077, Dec. 2007.

[41] I. M. Delamer and J. L. M. Lastra, "Service-oriented architecture for distributed publish/subscribe middleware in electronics production," *IEEE Trans. Ind. Informat.*, vol. 2, no. 4, pp. 281–294, Nov. 2006.

[42] J. Pasley, "How BPEL and SOA are changing web services development," *IEEE Internet Comput.*, vol. 9, no. 3, pp. 60–67, May/Jun. 2005.

Javier García-Zubia (M'08) received the B.S. degree in 1987 and the Ph.D. degree in computer engineering from the Faculty of Engineering, University of Deusto, Bilbao, Spain, in 1996.

He is with the Faculty of Engineering, University of Deusto, as an Associate Professor and Head of the Department of Industrial Electronics, Control Engineering, and Computers Architecture. He is responsible for the Remote Laboratory at the University of Deusto.



Pablo Orduña (M'06) is currently working toward the Ph.D. degree at the University of Deusto, Bilbao, Spain.

He is a Research Assistant at the Ambient Intelligence Department of DeustoTech (Fundación Deusto) and also the Lead Software Designer and Developer of WebLab-Deusto.

Diego López-de-Ipiña received the Ph.D. degree from the University of Cambridge, Cambridge, U.K., in 2002.

He is a Principal Researcher with the Software Systems Research Line at the Faculty of Engineering, University of Deusto, Bilbao, Spain. He has directed several research projects involving the adoption of Web 2.0 to novel application areas such as industrial electronics or mobile ubiquitous computing. His main research areas are pervasive computing, Internet of things, semantic-service middleware, and mobile-mediated human–environment interaction.

Gustavo R. Alves received the B.S., M.Sc., and Ph.D. degrees in electrical and computer engineering from the Faculty of Engineering, University of Porto, Porto, Portugal, in 1991, 1995, and 1999, respectively.

Since 1994, he has been an Adjunct Professor with the Department of Electrical Engineering, School of Engineering, Polytechnic Institute of Porto (IPP), Porto, Portugal. His research interests include design for debug and test, reconfigurable systems, and remote experimentation in e-learning contexts.