Addressing Software Impact in the Design of Remote Labs

Javier Garcia-Zubia, *Member IEEE*, Pablo Orduña, *Member IEEE*, Diego Lopez-de-Ipiña, 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 work presents the main characteristics of a Remote Lab, analyses 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 SOLA (Service-Oriented Lab Architecture)-based approach for the design of future Remote Labs so that client agnostic remote laboratories and remote laboratory composition are enabled. The experience with the real Remote Lab, WebLab-Deusto, is presented.**

***Index Terms*— eLearning, Remote Labs, Web Services, SOA**

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 Labs or WebLabs have clearly shown their academic usefulness [3] [4] [5], not only to substitute the real physical laboratories but rather to complement and power them, although it has also been pointed that Remote Labs might not always be as good [6]. Anyway, the first Remote Labs [7] or WebLabs [8] [9] were organised and promoted by a laboratory or department, but their success has motivated the universities themselves to manage them. This change supposes an acknowledgement 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, organisational 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]. The 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 again. This is something bearable by a computer scientist but not so obviously affordable by other type of engineers.

For example, in the remarkable work [11] more than 100 articles around WebLabs are examined. However, in only one of them [12] software is 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/SOA, Services Oriented Architecture) to produce better quality remote labs 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 Labs and Remote Engineering. Analysing the 67 papers published, none of them is related with the analysis of the different software strategies that can be used in a Remote Lab.

Recently, in December 2007, [13] explains how a service oriented approach can be applied to a remote lab for robotics.

In conclusion, very few researchers work on the software aspects of remote labs. However, many remote lab 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 side of WebLabs. In consequence, this work analyses the relative importance of these two different software sides and also conjectures about the benefits of adopting a SOLA (Service-Oriented Lab Architectures) approach in the design of future remote WebLabs.

Sections 2 and 3 analyse several technologies for implementing the client and the server, respectively, justifying the choice of AJAX and Python in each case. This selection is established correlating the technologies with the main characteristics of a WebLab, seeing that some characteristics only can be reached using specific technologies. Section 4 describes the SOLA architecture, proposed for the development of future WebLabs. In the section 5, the

experience and the usefulness of the real Remote Lab, WebLab-Deusto, is presented.

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 tele-control 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, it can be accessed by the users wherever they are, under different hardware and software platforms) may be a much better option than providing a standalone application which requires a lot of software to be installed.

This section is organised to select the most suitable client technology to implement a Remote Lab. The four subsections present the client technologies, the main characteristics of a Remote Lab and the criteria to choose among the five technologies analysed: 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 Laboratories clients, from the lightest web-based ones to the heaviest standalone desktop-based ones. Hence, client applications could be classified into two groups:

- *Desktop clients*: those run in the user's desktop computer.
- *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, 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 3D graphics or integrating them in the user's desktop. This is something web applications usually can not provide.

The present work 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:

- *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 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 non-intrusive, but they are not common.
- *Non-intrusive 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:

- The Remote Laboratory development team
- The server where the client software and experiments are hosted has not been tampered with.
- 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, non-intrusive applications are obviously clearly preferred in security terms.

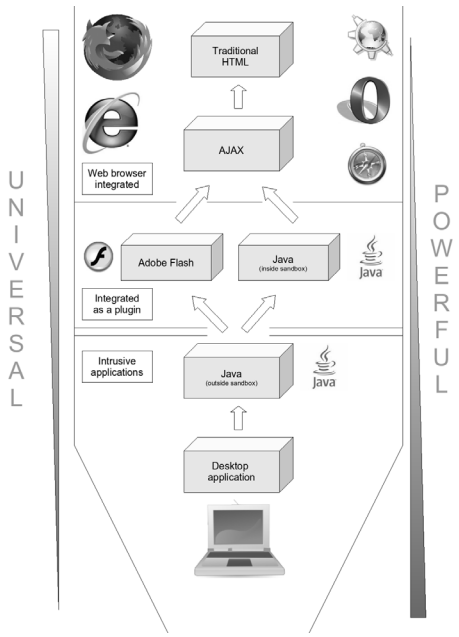

Figure 1. Technologies classification in the client side

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 aims to be offered by the university, it has to fulfil the requirements imposed by the IT services, as Moodle does, for example. In other case the WebLab will fail to be a professional educational tool.

The characteristics analysed in Table 1 are:

- *Cross-platform.* The WebLab can be accessed by all the O.S.: Windows, Linux, Mac OS, etc.
- *Security.* The WebLab uses https, does not need permissions on the firewalls, only needs the 80 & 443 ports opened, etc.
- *Web browsers.* The WebLab can be accessed by all the web browsers: Explorer, Mozilla, Opera, Safari, etc.
- *Intrusivity.* The user does not give permissions to the WebLab client application: hard disk access, execution of native code, etc.
- *Interaction.* The WebLab needs to implement the maximum of interaction with the user.
- Installation. The WebLab runs without any previous installation in the client side: plug-in, JVM, Flash Player, etc.
- *Devices.* The WebLab can be accessed by all the devices: PC, PDA, mobile phone, etc.
- *Bandwith.* The WebLab needs the maximum bandwith efficiency.
- *Audio&Video.* The WebLab needs the maximum of audio & video power.
- *Power.* The WebLab is very complex and needs a powerful tool to be implemented.

TABLE I.
OPINION OF THE EXPERTS ABOUT THE CHARACTERISTICS OF REMOTE LABS (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**
Bandwith	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.

The rest of this section analyses the different client technologies available to select one of them. It also justifies why some advanced features should be left out in order 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.

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 below before being analysed 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 analysed 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 applications 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 in order to do so. There is a lot of literature specifically referred to this approach, especially in the fields of control engineering and electronics [21] [22] [23] [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] [26] [27] [28] [29] because it is a powerful platform to develop Rich Internet Applications (RIA).

In order 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 Operating Systems, 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(1), so 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 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 can not 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 organisation –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 (Rich Internet Applications). The user of an Adobe Flash has to install the Adobe Flash Player, which will interpret byte-code found in files in the SWF format. Once the Adobe Flash Player is installed, the applications made in Adobe Flash will be a non-intrusive 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 non-intrusive way (when accessing a file, the user chooses the file). The potential Adobe Flash has for graphics and animations, as well as to access web services, providing a non-intrusive 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 bit architectures, which is quite a big drawback. Also, 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: XMLHttpRequest. This component allows calling asynchronously XML Web Services from Javascript. This is why AJAX is actually 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. So, 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 the ones explained up to this point that supports interactivity with the server, even in a Remote Lab [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. So, with no extra effort at all, AJAX applications will run even in mobile devices [34].

Big companies 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 now is 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 Lab needs high-resolution video and audio capabilities the application must integrate a specific function based in Adobe Flash, for

[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/

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 placed on web accessibility (based on web standards), making possible to develop an accessible web application that will allow disabled people to use the web page [32] [36] –there are laws to 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 provides, 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 analysed in this study, 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 a 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, and 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, III, IV and V summarize the possibilities of the technologies for designing the client. The characteristics analysed in the tables are seventeen and they have been selected using the experience gathered in Remote Labs 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 seventeen 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 Operating Systems?
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?

TABLE II.
ANALYSIS OF THE 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.

2. Security/Standards: Is the client secure and/or based on standards?
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?

TABLE III.
ANALYSIS OF THE 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.

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 bandwith 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?

TABLE IV.
ANALYSIS OF THE 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.

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

TABLE V.
ANALYSIS OF THE 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.

The Fig. 2 resumes the marks obtained in the Tables II-V.

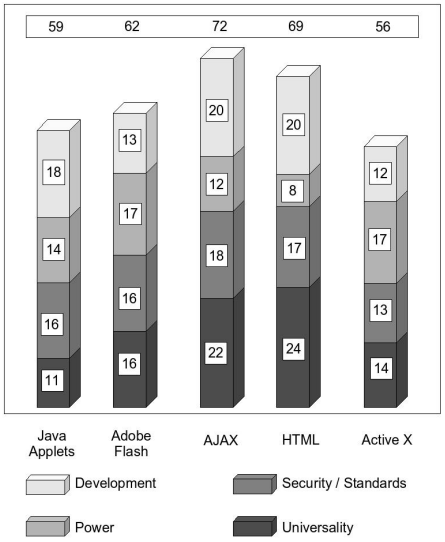

Figure 2. Comparison between different client technologies

Analysing numerically the results of Fig. 2:

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

For a specific WebLab, the designers can select the requirements of their WebLab in the Tables II-V and analyse them, or perhaps add new characteristics to the table or weight 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.

TABLE VI.
ANALYSIS OF THE 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**				

Anyway, among all the technologies considered, the approach that is experiencing a faster growth is, by far, the AJAX approach. More and more, especially 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 Laboratories 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 can not be applied to the server:

- 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.
- 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.
- The requirements in terms of security are very different between the server and the client. The server must control 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.

- All the servers are intrinsically intrusive, and this is not a problem because it only affects to the servers.
- In the client side five main technologies were analysed, 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 Lab with any client technology. Moreover, it is common to integrate different technologies in the server side (Java+Oracle, PHP+Apache+MySQL, etc), what it 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.

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

TABLE VII.
ANALYSIS OF THE 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.

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

IV. SOLA: SERVICE-ORIENTED LAB ARCHITECTURES

A final important aspect regarding the importance of software for the design of better Remote Labs is not just the software technology itself but the paradigm adopted for the design and implementation of a Remote Lab. Lately, it has become commonplace adopting a service-oriented architecture, 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 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 in order 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 Service Oriented Lab Architecture (SOLA), i.e. an adaptation of the commonly known Service Oriented Architecture (SOA) to the Remote Lab domain, in the design of future Remote Labs. Consequently, in those newly designed labs the functionality offered for a given Remote Lab 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 in order to foster cooperation among different Remote Labs. Thus, SOLA is a an emerging standard, refining the SOA concept for standard enterprise services, for connecting distributed Remote Labs to the Service Oriented Architecture (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 in order to enable a feasible cooperation between the distributed functional blocks of Remote Labs that may be assembled to compose sophisticated, and previously infeasible within a single organisation, Remote Lab experiments.

Adapting and exporting the functionality of a Remote Lab 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 able by concatenating the outputs of one hardware experiment as inputs of another one, and so 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 Lab.

In conclusion, the adoption of a SOLA-approach would decouple the client and server parts of a Remote Lab. Then, the server-side would be completely agnostic to the clients consuming its functionality. It would only provide a common WSDL API accessible through a distributed communication standard such as 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:

- v 0.1 Desktop application implemented in C. 2001.
- v 1.0 Web application implemented in Java. 2004.
- v 2.0 Web Application implemented in AJAX. 2005.
- v 3.0 Web application implemented in AJAX. 2007.

WebLab-Deusto is now being used in three subjects of the Faculty of Engineering: Programmable Logic, Electronics Design and Electronics Instrumentation by two hundred 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. Conclusions

Using the experience obtained developing WebLab-Deusto since 2001, the paper has analysed 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. The paper establishes that some requirements can only be reached with a specific technology. According to Table 1, 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 Lab client development is AJAX, specially if universality is the goal of the WebLab.

The scenario and the criteria in order 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 (Service Oriented Lab Architecture)-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] Biggs, J. *Teaching for quality learning at university*, Open University Press/McGraw Hill, 2003.

[2] Carlson, L., Sullivan, J.F. “Hands-on engineering: Learning by doing in the integrated teaching and learning program” *Int. J. Eng. Education, VOL 15, Nº 1,* 1999.

[3] Ertugrul, N. “New area in engineering experiments: An integrated and interactive/learning approach , and real-time visualizations.”, *Int. J. Eng. Education, VOL 14, Nº 5*, 1998.

[4] Garcia-Zubia, J. et al. "WebLab-GPIB at the University of Deusto”, *Proc. REV 2007 Remote Engineering and Virtual Instrumentation*, ISBN: 978-3-89958-278-9, 2007.

[5] Corter, J.E. et al “Remote versus hands-on labs: A comparative study” *34th ASEE/IEEE Frontiers in Education Conference*, 2004.

[6] Soysal, O. "Computer integrated experimentation in electrical engineering education over distance" *Proceedings of ASEE 2000 Annual Conference*, Saint Louis, MO, June 2000.

[7] Aburdene, M.F. et al. “A proposal for a remotely shared control systems laboratory” *IEEE/ASEE 1991 Frontiers in Education Conference*, 1991.

[8] Ross, R.J. et al. “WebLab! A universal and interactive teaching, learning, and laboratory environment for the world wild web”. *Proc. 28th SIGCSE Technical Symposium on Computer Science Education*, San Jose (EE.UU.), 1997.

[9] Atkan, B. et al “Distance learning applied to control engineering laboratories” *IEEE Trans. Education,* VOL 39, Nº 3, 1996.

[10] Hine, N et al. "Institutional factors governing the deplyment of remote experiments: lessons from the REXNET project”, *Proc. REV 2007*

Remote Engineering and Virtual Instrumentation, ISBN: 978-3-89958-278-9, 2007.
[11] Ma, J. and Nickerson, J.V., "Hands-on, simulated, and remote laboratories: A comparative literature review", *ACM Computing Surveys*, Vol. 38, Nº 3, 2006
[12] Kolberg, S. y Fjeldly, T.A., "Web Services remote educational laboratories", *Proceedings of the International Conference on Engineering Education,* Gainesville, FL 1-6, 2004.
[13] Fernandez, J.; Marin, R.; Wirz, R. "Online Competitions: An Open Space to Improve the Learning Process" *IEEE Trans. on Industrial Electronics, VOL: 54 , Issue: 6,* Dec. 2007.
[14] Garcia-Zubia et al, "Questions and answers for designing useful WebLabs", *International Journal of Online Engineering, VOL II, Nº 3,* ISSN: 1861-2121, www.ijoe.org, Austria, 2006.
[15] Marin, R. et al. "A multimodal interface to control a robot arm via the web: a case study on remote programming" *IEEE Trans. on Industrial Electronics, VOL: 52 , Issue: 6,* Dec. 2005.
[16] Hassan, H.; Dominguez, C.; Martinez, J.M.; Perles, A., Albadalejo, J. "Remote Laboratory Architecture for the Validation of Industrial Control Applications" *IEEE Trans. on Industrial Electronics, VOL: 54 , Issue: 6,* Dec. 2007.
[17] He, F. et al "Object request brokers for distributed measurements", *IEEE Comput. Appl. Power,* VOL 14, Nº 1, Jan. 2001.
[18] Guimaraes, E. et al "REAL: a virtual laboratory for mobile robot experiments" , *IEEE Transactions on Education, VOL 46, Nº 1*, Feb. 2003.
[19] Bagnasco, A.; Chirico, M.; Scapolla, A.M. and Amodei, E. "XML data representation for testing automation", *IEEE AUTOTESTCON Proceedings*, 2002.
[20] García-Zubía, J. et al. "Suitability and implementation of a WebLab in engineering", *10th International Conference on Emerging Technologies and Factory Automation, ETFA 2005*, ISBN: 0-7803-9402-X, Vol II, pp: 49-56, Catania (Italia), Sep 2005.
[21] Costas-Pérez, L. et al "Optimization of an industrial sensor and data acquisition laboratory through time sharing and remote access" *IEEE Trans. on Industrial Electronics, VOL: 55 , Issue: 6,* June 2008.
[22] Alves, G. et al. "Remote Experimentation Network - Yielding an Inter-University Peer-to-Peer e-Service" *10th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA'05*, Catania, Italy, Sept. 2005
[23] Hu, W. et al "Design and implementation of Web-Based control laboratory for tests rigs in geographically diverse locations" *IEEE Trans. on Industrial Electronics, VOL: 55 , Issue: 6,* June 2008.
[24] Hercog, B.; Gergic, B.; Uran, S.; Jezernik, K. "A DSP-Based Remote Control Laboratory" *IEEE Trans. on Industrial Electronics, VOL: 54 , Issue: 6,* Dec. 2007.
[25] Davoli, F. et al "LABNET: Toward remote laboratories with unifed access", *IEEE Trans. on Instrumentation and Measurement,* VOL 55, NO 5, October 2006.
[26] Mittal, A.; Gupta, Ch. and Gupta, A. "Addressing the bandwith efficiency, control, and evaluation issues in software remote laboratories" *IEEE Trans. on Industrial Electronics, VOL: 55 , Issue: 6,* June 2008.
[27] Bertocco, M. et al "A client server architecture for distributed measurement systems" *IEEE Trans. Instrum. Meas.*, VOL 47, Nº 15, Oct. 1998.
[28] Ferrero, A. et al "ReMLab: A Java-based remote, didactic measurement laboratory", *IEEE Instrum. Meas.* , VOL 52, Nº 3, June 2003.
[29] Sánchez, J. et al, "A Java/Matlab-Based environment for remote control system laboratories: Illustrated with an inverted pendulum", *IEEE Transactions on Education, Vol. 47, Nº 3,* pp. 321-329, 2004.
[30] Gustavsson, I. "A remote access laboratory for electrical circuit experiments" *Int. J. Eng. Education, VOL 19, Nº 3*, 2003.
[31] Paulson, L.D. "Building rich web applications with Ajax," *Computer (IEEE Computer Society), vol. 38, no. 10*, Oct. 2005.
[32] Gobbo, F. and Vaccari, M. "Open standards for higher education in robotics by immersive telelaboratories" *Learning Technology Newsletter (IEEE Computer Society)*, *VOL 7, Nº 3*, 2005.
[33] Atkinson, I.M. et al. "CIMA based remote instrument and data access: An extension into the australian e-Science environment" *2nd IEEE International Conference on e-Science and Grid Computing*, Netherlands, Dec 2006.
[34] Lopez-de-Ipiña, D., Garcia-Zubia, J. and Orduña, P. " Remote Control of Web 2.0-enabled Laboratories from Mobile Devices" *2nd IEEE International Conference on e-Science and Grid Computing, eScience 2006*, Dec. 2006.
[35] Emigh, J. "New flash player rises in the Web-Video market," *IEEE Computer Society*, *vol. 39, no. 2,* Feb. 2006.
[36] Cooper, M. "Accessibility and usability in complex web based learning applications: Lessons from the PEARL project" *Proceedings of the Corporations, Government, Health, and World Conference on E-Learning in Higher Education*. 2002.
[37] Boletín Oficial de las Cortes Generales, 2002, Num. 68-13, 3 de julio de 2002.
[38] Boletín Oficial del Estado, 2003, Num. 289, 3 de diciembre de 2003.
[39] Gomes, L. and Garcia-Zubia, J. eds. *Advances on remote laboratories and e-learning experience*s, Ed. University of Deusto, ISBN: 978-84-9830-077-2, 2007.
[40] Indrusiak, L.S.; Glesner, M.; Reis, R. "On the Evolution of Remote Laboratories for Prototyping Digital Electronic Systems" *IEEE Trans. on Industrial Electronics, VOL: 54 , Issue: 6,* Dec. 2007.
[41] Delamer, I.M. and Lastra, J.L.M. "Service-Oriented architecture for distributed publish/subscribe middleware in electronics production" *IEEE Trans. on Industrial Electronics, VOL: 2 , 4,* Nov. 2006.
[42] Pasley, J. "How BPEL and SOA are changing web services development," *IEEE Internet Computing, VOL 09, N°. 3*, May/Jun, 2005.

Javier García-Zubia graduated in 1987, and received the PhD. degree in computer engineering in 1996 from the Faculty of Engineering of the University of Deusto. He is Associate Professor and Head of Dpt. of Industrial Electronics, Control Engineering, and Computers Architecture of the Faculty of Engineering of the University of Deusto. He is the responsible of the remote lab at the University of Deusto (http://weblab.deusto.es).

Pablo Orduña is a Research Assistant at Ambient Intelligence department of DeustoTech 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.

Diego López-de-Ipiña is principal researcher of the Software Systems Research Line at the Faculty of Engineering of the University of Deusto. He received his PhD from the University of Cambridge, U.K in 2002. His main research areas are pervasive computing, internet of things, semantic service middleware, and mobile-mediated human-environment interaction. 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.

Gustavo R. Alves graduated in 1991, and received the M.Sc., and Ph.D. degrees in electrical and computer engineering in 1995, and 1999, respectively, from the Faculty of Engineering, University of Porto, Portugal. He is an Adjunct Professor in the Department of Electrical Engineering of the School of Engineering of the Polytechnic Institute of Porto (IPP), since 1994. His research interests include design for debug & test, reconfigurable systems, and remote experimentation in e-learning contexts.