Enabling mobile access to Remote Laboratories

Pablo Orduña, Javier García-Zubia, Jaime Irurzun, Diego López-de-Ipiña, Luis Rodriguez-Gil
DeustoTech – Deusto Institute of Technology
University of Deusto
Bilbao, Spain
{pablo.orduna,zubia,jirurzun,dipina}@deusto.es, luis.rodriguez@opendeusto.es

Abstract— **Remote Laboratories constitute a first order didactic resource in engineering faculties. Their use from mobile devices to increase the availability of the experiments at the laboratory is a challenge highly coupled to the requirements established by each experiment. This paper will present and compare the main strategies for adapting a Remote Laboratory to mobile devices, as well as the experience of a real Remote Laboratory, WebLab-Deusto, in this adaptation.**

Keywords-component; remote laboratories; m-learning; android, iphone

I. INTRODUCTION

A Remote Laboratory is a software and hardware system that enables students to use real experiments physically located at a university. This way, students can access real experiments 24 hours a day, 7 days a week, even including holidays, from anywhere with access to the Internet. Given that the experiments are real, Remote Laboratories have the opportunity to probe this fact to the students (e.g. with a webcam), so the students don't lose the feeling that they are doing exactly what they would do in a hands-on-lab session.

The adaptation of Remote Laboratories to mobile devices has been studied in the literature [1] for older devices. However, the spread of new advanced mobile platforms such as iPhone or Android among students generates a new challenge and increases the opportunities to exploit these devices in their education. Furthermore, due to the decreasing fees of mobile networks, a student will be able to use or validate experiments and compare results with other students outside the laboratory.

The paper is organized as follows: section II presents the possible strategies that can be used when aiming mobile devices; section III describes how a remote laboratory can be built aiming them, showing the experience of WebLab-Deusto and comparing the existing approaches. Finally, section IV points out the future work and section V sums up the conclusions.

II. STRATEGIES

Two strategies can be used when developing software for mobile devices:

- Web technologies: AJAX, subsets of HTML5 in certain devices, etc.
- Native technologies: iPhone SDK, Android SDK, GTK+/QT on Maemo/MeeGo, LiMo SDK, Compact.NET on Windows Mobile, Symbian SDK, etc.

The main advantage of Web technologies is that they increase the availability of the application among different devices. Furthermore, with the increasing support of HTML5 in browsers, features like geolocation, audio or video become available on mobile devices. However, the flexibility of the Web is limited when compared to native applications, which count with more powerful features. When talking about Remote Laboratories, the correct balance must be established in order to choose the most suitable technology for each experiment.

A. Using mobile technologies

Nowadays, most mobile device manufacturers provide development frameworks on top of which third-party developers can build applications. The added value of this is clear: the functionality of a mobile device becomes flexible, since new applications can be built using the capabilities of that mobile device.

However, the range of development frameworks has become wide: Symbian, Android, iPhone, Windows Mobile, LiMo, BlackBerry or Maemo (now merged with Moglin in MeeGo) are examples of mobile operating systems. The applications available for these operating systems are mainly native applications developed using their own SDK, which is only supported by their operating system. For instance, a native application for Symbian, developed using Symbian C++, will only work on Symbian.

The advantage of using a native technology is that it can use all the resources that the mobile device provides through its SDK. If the mobile device supports so, an application may use 3D graphics, retrieve the user's position, access the accelerometers, take pictures or video from the camera, connect with other devices via bluetooth, interact with files and handle disk storage, access the mobile calendar or contacts, or even play music and videos, while mobile Web browsers usually do not provide these features to Web applications.

978-1-61284-643-9/11/$26.00 ©2011 IEEE April 4 - 6, 2010, Amman, Jordan
2011 IEEE Global Engineering Education Conference (EDUCON) – "Learning Environments and Ecosystems in Engineering Education"

The main question, however, is about the usefulness of all these features for Remote Laboratories:

- 3D graphics might be used to provide an immersive scenario that could improve the interaction with the experiment, as is already done with WonderLand [2] or Second Life [3].
- Access to the accelerometers can improve the usability of the system, especially if the experiment requires some movement.
- Handling disk storage is useful to retrieve results from the experiment and to provide scripts that the remote experiment might require. This would not be supported by iPhone or Android, since the user cannot directly handle files.
- Accessing the user's calendar could assist the user by showing his personal events at the screen where he is offered to book a remote laboratory session.
- The camera can be used to perform Augmented Reality experiments. The student could use a textbook tagged with QR codes to see the real experiment running on top of the textbook in a transparent way.
- Notifying the user that the experiment enqueued is now ready to be used could be useful so the user does not need to check it manually every few seconds.

B. Using Web technologies

It is actually claimed that around 40% of iPhone users browse the Web more often from their iPhone than from their own desktop [4], which means that mobile devices have become a proper way to support the Web.

However, Web applications usually need to be adapted to mobile devices. This adaption requires three changes:

- Provide a proper layout. Developers should think what is actually going to be used from a mobile device, and how the user interface should look like to be easily used in such a small screen. For instance, newspapers tend to provide a vertical panel where each headline is represented in a row with a single sentence, so the user can quickly see which news are more interesting, and click on them. Each row works as a button, so it becomes easy to press it with a touch screen.
- Provide the required contents. Developers should think which contents are going to be migrated to the mobile version. Users might look at the mobile version as a complement to the desktop version, so it becomes normal to remove some features.
- Avoid plug-ins. Many Web applications provide features that are based on plug-ins such as Java applets, Adobe Flash or Microsoft Silverlight. These plug-ins are not available in many devices, and it is difficult that they become available due to the resources required by the plug-in developer to port the plug-in to the wide range of existing mobile platforms.

Identify applicable sponsor/s here. If no sponsors, delete this text box. (sponsors)

Remote Laboratories, however, tend to rely on plug-ins like Java Applets or Adobe Flash, as detailed in [5]. To mention two examples, in MIT iLabs [6] Java has been the preferred client development environment [7], and the BTH VISIR [8] project uses Adobe Flash [9]. While some devices support Adobe Flash (such as those based on Maemo, and certain new versions of Android), relying on a plug-in will exhaustively decrease the number of supported users.

III. Development of a Remote Laboratory for Mobile Devices

Figure 1. shows the layers of a simple Remote Laboratory, which are:

- Client software placed at the student's computer
- Server software placed at the university's servers
- Physical experiment placed behind this server software

Each layer can later be divided into complex components. For instance, the second layer might be composed by several servers deployed at the university, both for security reasons as well as to trust the laboratory experts with the experiments' management. The third layer can also provide high complexity, designing experiments that can handle several users concurrently at hardware level, experiments that self test that everything works even under adverse situations -such as a robot handling its own battery-, or experiments checking that the inputs sent by the user will not damage the devices.

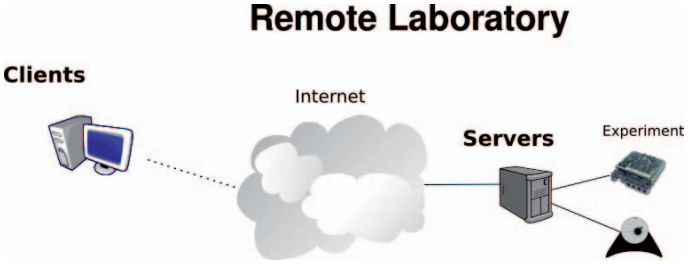

Figure 1. Initial view of a Remote Laboratory

However, the focus of this paper is to detail how to access these experiments from mobile devices, and thus on the first layer.

A. Which type of experiments might be used from mobile devices?

There is a wide range of mobile devices and the experiments will only work on subsets of them. The more powerful the mobile device is, the more suitable will probably become to support an experiment.

However, there are certain experiments that can simply not work on a mobile device. For instance, if an experiment requires the user to use an IDE not available in a mobile device, it will be difficult to use that experiment from a mobile device. There are two approaches to consume this type of experiment:

- Providing the IDE software in a remote machine. The Carinthian University of Applied Sciences counts with an experiment that uses a CPLD. In order to provide the required software to the student, the student directly accesses the remote machine, which has the software already installed. This way, a student can efficiently use the IDE software without having to install it, from anywhere in the Internet. This approach could be applied to the scenarios related to mobile devices, where the student might be able to use a mobile device to interact through a remote desktop with the required IDE. However, even if this approach was applied, it might be still difficult to use it, since the IDE GUIs are designed to be used from a desktop.
- Being a secondary user of the experiment. In the sample use described above, the student works in his own, without collaborating with other students. However, this is not always the case. SecondLab [3] is a SecondLife-based client for a certain experiment powered by WebLab-Deusto. In this experiment, a microbot is remotely programmed by a student from Second Life. Other students can be in the same room in Second Life watching the same results without interacting with the experiment. The same can be applied to the experiments performed by [2]. This master/slave usage of the experiment could be adapted to mobile devices: a student could write a program using an IDE and then send it to the device while other students see what is happening from a mobile device, or even they could interact with the experiment during the session managed by the master student.

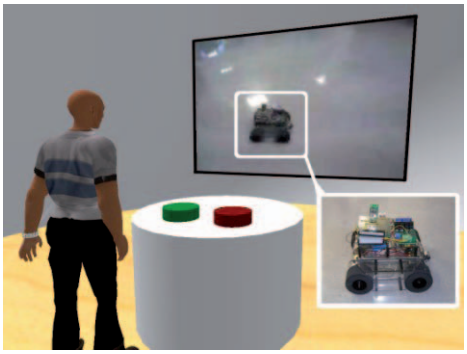

Figure 2. WebLab-Deusto microrobot experiment used from Second Life with SecondLab

B. *Extensible Remote Laboratories*

Extensible Remote Laboratories are a solution to avoid facing from scratch the development of a whole Remote Laboratory when somebody wants to enable remote access to a certain experiment. WebLab-Deusto is an example of this. It is an Open Source (GNU GPL v2 at the moment of this writing; BSD in the next release) web-based Remote Laboratory developed at the University of Deusto.

WebLab-Deusto is not an experiment itself, but a framework on top of which experiments can be developed. Therefore, there are two separate developer roles:

1. Infrastructure Developer
2. Experiment Developer

The first type of developer must be an IT expert that will deal with low level issues, such as scalability, communications, authentication, authorization, session management, scheduling, user tracking, user management, logging, as well as handle complex deployments.

However, the second type of developer should only focus on building the experiment itself. In order to do so, the Experiment Developer is required to build two components:

1. Server side component: it will validate the user's input and then translate it to the devices.
2. Client side component: it will show the User Interface of the experiment. The user will interact with this User Interface, which will send commands to the server side component and afterwords will show the responses.

In order to make the development of these components easy, WebLab-Deusto provides two APIs: one to develop clients and another one to develop servers. These APIs are based on commands: the client will call a method to send a command (which will be a string like "turn switch 1 on"), and the framework will internally send that command to the server, receive the response and finally provide it to the client. The Experiment Developer will handle the serialization and deserialization of the commands.

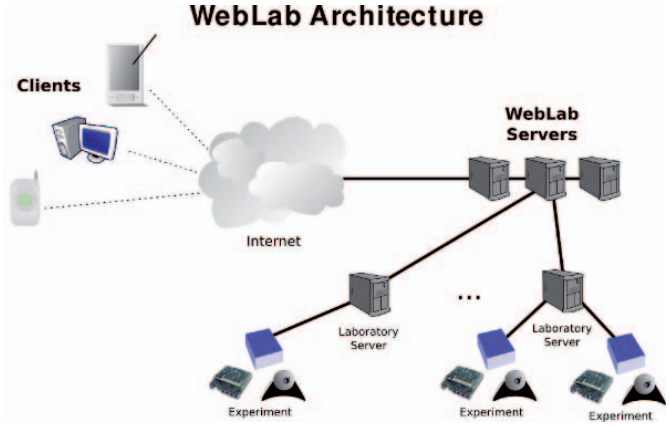

Figure 3. WebLab-Deusto architecture

Since the background of the different Experiment Developers is diverse, as well as the tools they have experience with, WebLab-Deusto supports a range of development frameworks for the development of both the client and the server. In the latest release (3.9.0), WebLab-Deusto provides libraries to develop the server in the following programming languages:

- C
- C++
- Java
- .NET

- LabVIEW
- Python

These libraries have an interface that the developer must implement, with four simple methods:

- Start: a new user is about to use the experiment.
- Send command: the current user is sending a string, and a response is expected.
- Send file: the current user is sending a file, and a response is expected.
- Dispose: the current user has finished using the experiment.

On the client side, WebLab-Deusto provides libraries in three technologies:

- Java Applets
- JavaScript
- Adobe Flash

These libraries have methods for retrieving the current configuration, as well as for sending commands and gathering responses from the experiment server.

Therefore, as detailed in Figure 4. , the framework already handles the common requirements of Remote Laboratories, providing the Experiment Developer with a flexible system on top of which to develop the experiments using simple APIs. These common requirements are [10]:

- Authentication and authorization: the framework handles authentication and authorization.
- Security: the framework wraps the database management and the communications (i.e. the framework can be configured to use SSL without changing the client).
- Logging and user tracking: the framework stores who used every experiment, as well as in which moment and exactly what was done during that session. If a user sends a malicious file, the administrators can later find it, since it has been stored in a different node.
- Reservations: scheduling models are provided by the framework, so without changing the client the administrator can grant access to the experiments to different users with different priorities and time assignements.
- Deployment: the experiments can be placed in different rooms at the university while the core servers are placed in a different place (such as the IT services). The experiments don't need to have a public IP address or any port opened to the Internet, since the framework proxies all the requests.
- Scalability: the core servers can be spread among different machines to balance the load of users.
- Administration: administration tools are provided to manage the rest of the requirements.

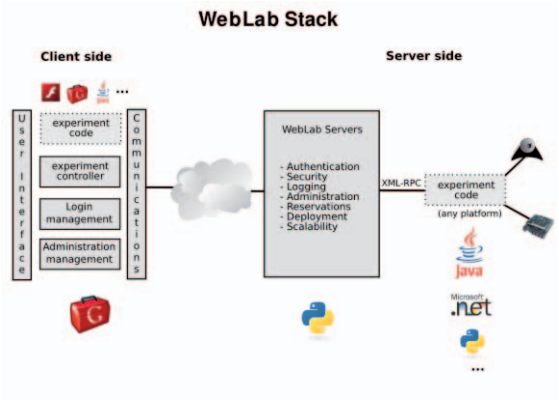

Figure 4. WebLab-Deusto stack

C. *Embracing mobility*

In December 2006, the University of Deusto presented how WebLab-Deusto 2.0 could be accessed from mobile phones such as Symbian devices with the Opera Mobile Web browser [1].

Figure 5. WebLab-Deusto 2.0 running under a Nokia 6630 device with Opera Mobile

During the year after, the Remote Laboratory was rewritten, and by September 2007 the new version (WebLab-Deusto 3.0) was first used by students. The client of WebLab-Deusto 2.0 had been written in plain JavaScript. However, the client of WebLab-Deusto 3.0 [11] was completely written in Google Web Toolkit (as well as in the current 3.9 and next future 4.0).

Google Web Toolkit -GWT- is an open source technology developed by Google that provides an API to be used from the Java programming language. The toolkit is capable of translating that Java code to JavaScript. This way, a developer can use a Java IDE like Eclipse (it actually includes a plug-in for Eclipse), and there write and test Java code, finally compiled into JavaScript. The Java code can only use libraries provided by the GWT API (for instance, the classes for managing files are not available since the result is compiled into JavaScript, which does not support files).

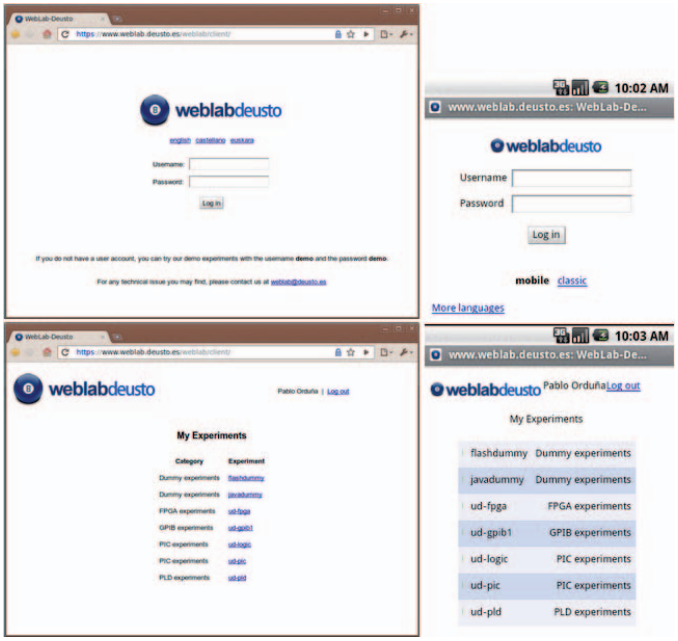

Figure 6. On the left side, the desktop version of the WebLab-Deusto client. On the right side, the mobile version running on an Android 1.6 emulator

The following are remarkable features provided by GWT:

- Fast development cycle: in the development mode, any change to the code will be updated as soon as the Web browser refreshes the Web page. With the Eclipse plug-in, the developer can debug the code as a regular Java application and use JUnit to create unit tests.
- Deferred binding: GWT does not generate a single JavaScript file with all the resulting code, but a set of JavaScript codes. For instance, it compiles a version for each major Web browser, optimizing all the low level calls for each Web browser. It also compiles a version for each supported language. Currently it supports 6 Web browsers and 4 languages, so it generates 24 different versions of the same Web, making this effort completely transparently to the developer.
- Size optimizations: GWT can compile in the "obfuscated mode" if requested, which reduces considerably the size of the generated scripts (given that they do not conserve the original Java names). It also supports a feature called "code splitting", which is that the developer can select parts of the code and explicitly mark them so they will be separated as different JavaScript files and will only be downloaded if a certain conditions match. For instance, each experiment of WebLab-Deusto is compiled into a different file, so the code and the resources of an experiment are not downloaded if the user does not use that experiment. The same applies to the different themes available (such as the "mobile" and the "desktop" ones).
- Cache management: GWT makes possible to include resources (like images or CSS files) in the JavaScript files. With this system, as soon as the current fragment of code is downloaded, it is ready to be rendered, without requiring any more time to download the images. Additionally, since this JavaScript file is cached by the Web browser, the next time the browser will only perform one request to the server to check if the JavaScript file was changed, without having to check if every single image was changed too.

WebLab-Deusto currently provides two versions (actually two themes, which a third-party could reimplement to adapt the appearence to its institution) of the common, experiment independent user interface: a desktop version and a mobile version. The framework enables the Experiment Developer to implement two different user interfaces of its experiment: one for desktop themes and another one for mobile themes. Therefore, it is up to the Experiment Developer to write the client of the experiment in the suitable technology. If the Experiment Developer chooses to use Adobe Flash or Java Applets, and he does not provide an extra implementation for mobile versions, then that experiment will hardly be supported in mobile devices. However, if the Experiment Developer only implements one version in JavaScript or GWT, then it will probably run on mobile devices, although it might not have been properly adapted to smaller screens.

As an example, Figure 7. shows a single experiment (ud-logic) that has been adapted to the mobile version, by using smaller images and reorganizing the layout of the experiment.

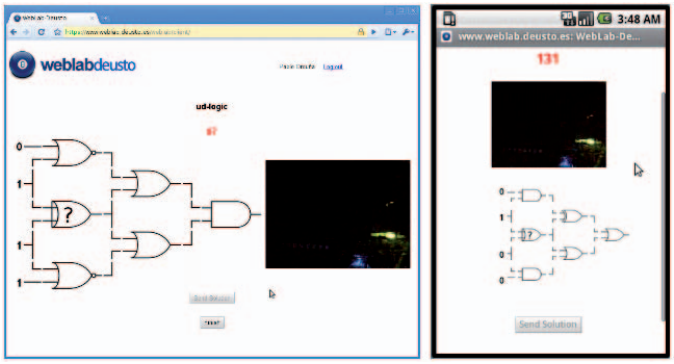

Figure 7. On the left side, the desktop version of the WebLab-Deusto client. On the right side, the mobile version running on an Android 1.6 emulator

D. *Comparing both approaches*

The decision of implementing a native version or Web one mainly depends on whether it is affordable. In general, the Web approach requires adapting the experiment once to mobile screens and then it will work on most modern mobile devices. Depending on the technologies and the design used to develop the client of the Remote Laboratory, providing this Web version might not require much effort. However, if it was built on top of a plug-in, such as Java Applets, Adobe Flash or LabVIEW, it might require to be completely rewritten.

On the other side, providing a different version of the application for each mobile platform is expensive in terms of development and maintenance. Every time a new feature

appears, all the versions should be updated. Google Mail, for instance, provides a mobile Web interface, and Google provides an enhanced version that is integrated with the operating system so as to perform notifications or add the "send through Google Mail" option to the gallery. However, this enhanced version is only available for Android, Blackberry and Symbian, while it relies on the Web version for iPhone, Windows Mobile and other mobile devices. Facebook also provides a mobile Web version supported by most mobile phones, but at the same time it targets a wide range of mobile platforms with native versions, including Symbian, Android, Windows Mobile, iPhone, Palm and Blackberry.

The main difference between both projects is that Facebook requires more integration with the mobile device: it is usually used to share pictures that the user may take from the mobile phone. The geolocation is also important in order to tag these pictures. Notifications are also important in Facebook, and may not be mapped to other protocols, while users of unsupported platforms of Google Mail can use a regular mail client to receive notifications.

Therefore, the decision of implementing the native or the Web version will rely on if the requirements of the experiment are important enough to justify the amount of work required to implement it for the most common mobile platforms used by the students. This decision must be made for every experiment independently.

IV. Future Work

One of the fields in which remote laboratories can best fit with mobile devices is Augmented Reality. At the moment, this combination would be possible only using native mobile technologies (instead of Web technologies). At the moment, the WebLab-Deusto Research Group is designing a system to give the students a modern tool based on this idea. The project will allow a student to see a real experiment performing what he is studying in the text book, just pointing with his mobile phone to a small label printed on his book.

Furthermore, different users may interact with the same experiment in a collaborative way, in a master/slave way or in a fully collaborative way. This could increase the potential of using Remote Laboratories under mobile environments: if a teammate is present in a laptop on the Internet, the technical drawbacks of mobile devices might be blurred. While the area provides interesting results for Remote Laboratories, they have not yet been exploited in mobile devices.

V. Conclusions

Using the experience in the software development of Remote Laboratories and the adoption of Remote Laboratories in mobile devices since 2006, the paper has presented and analyzed two strategies to implement a mobile-enabled Remote Laboratory.

The first strategy uses Web technologies, so existing code from the desktop version may be reused, although some adaption is still required. Using this strategy, the number of target mobile devices is maximized.

The second strategy uses native mobile technologies. While the control of the device is higher, being able to access the camera or drawing 3D graphics, the application must be rewritten and maintained for each target mobile platform.

The adoption of the first strategy in WebLab-Deusto is detailed, and finally both strategies are compared. The use of one or the other depends on the affordable cost of development and, as well as on the requirements of the experiment. This situation is similar to the decision of using lightweight technologies as AJAX or using more powerful but plugin-dependant technologies like Adobe Flash or Java Applets in regular Remote Laboratories, already addressed in the literature [12].

Acknowledgment

The authors would like to acknowledge to the Spanish Science and Innovation Ministry for the support in the project TIN2008-06083-C03/TSI "s-Labs – Integración de Servicios Abiertos para Laboratorios Remotos y Virtuales Distribuidos" and to the Industry Dpt. of the Regional Government of the Basque Country for the support in the project EUSKADI09 "WLPRO".

References

[1] López-de-Ipiña, D., García-Zubia, J., & Orduña P. (2006) Remote Control of Web 2.0-enabled Laboratories from Mobile Devices. Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing 2006 (pp. 123-123).

[2] Scheucher, T. and Bailey, P.H. and Gütl, C. and Harward, V.J. Collaborative virtual 3D Environment for Internet-accessible Physics Experiments. Proceedings of the International Conference of Remote Engineering and Virtual Instrumentation. 2009.

[3] SecondLab: A Remote Laboratory under Second Life. Garcia-Zubia, Javier; Irurzun, Jaime; Angulo, Ignacio;Orduña, Pablo; Ruiz-de-Garibay, Jonathan; Hernández, Unai; Castro, Manuel; Sancristobal, Elio. IEEE Engineering Education 2010 (EDUCON 2010). Madrid, Spain.

[4] Sauer, F. The enterprise apps in your pocket. [Web log post]. Retrieved from http://googlewebtoolkit.blogspot.com/2009/10/enterprise-apps-in-your-pocket.html

[5] Gravier, C., Fayolle, J., Bayard, B., Ates, M., & Lardon, J. (2008). State of the Art About Remote Laboratories Paradigms - Foundations of Ongoing Mutations. International Journal of Online Engineering, 4 (1). Retrieved from: http://online-journals.org/index.php/i-joe/article/view/480

[6] iLabs. (2010). [computer software]. Available from http://ilab.mit.edu/wiki/

[7] Harward, V.J., del Alamo, J.A., Lerman, S.R., Bailey, P.H., Carpenter, J., DeLong, K., Felknor, C., Hardison, J., Harrison, B., Jabbour, I., Long, P.D., Tingting Mao Naamani, L., Northridge, J., Schulz, M., Talavera, D., Varadharajan, C., Shaomin Wang Yehia, K., Zbib, R., Zych, D. (2008). The iLab Shared Architecture: A Web Services Infrastructure to Build Communities of Internet Accessible Laboratories. Proceedings of the IEEE. 96(6). 931-950. doi: 10.1109/JPROC.2008.921607.

[8] VISIR. (2010). [computer software]. Available from http://openlabs.bth.se/

[9] Gustavsson, I., Zackrisson, J., Håkansson, L., Claesson, I., & Lagö, T. (2007). The VISIR project - an Open Source Software Initiative for Distributed Online Laboratories. Proceedings of the Remote Eng. And Virtual Instrumentation Conference 2007.

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

J. M. (2009). Designing Experiment Agnostic Remote Laboratories. Proceedings of the Remote Engineering and Virtual Instrumentation Conference 2009.

[11] WebLab-Deusto (2010) [computer software]. Available from http://www.weblab.deusto.es/

[12] García-Zubia, J., Orduña, P., López de Ipiña, D., & Alves, G. (2009). Addressing Software Impact in the Design of Remote Labs. IEEE Transactions on Industrial Electronics. 56 (12). 4757 - 4767. doi:10.1109/TIE.2009.2026368.