Preprint of the paper published in the 9th IFAC Symposium on Advances in Control Education, 2012

Synchronous Collaboration with Virtual and Remote Labs in Moodle

Carlos A. Jara*[1], Ruben Heradio Gil†[2], Luis de la Torre‡[3], Jose Sanchez§[3], Sebastian Dormido¶[3], Fernando Torres‖[1], and Francisco A. Candelas**[1]

[1]Physics, Systems Engineering and Signal Theory Department, University of Alicante, Spain
[2]Software Engineering and Computer Systems Department, Spanish Open University, Madrid, Spain
[3]Computer Science and Automatic Department, Spanish Open University, Madrid, Spain

Abstract

This work presents the tools and procedure developed by the authors to create synchronous collaborative virtual and remote laboratories within a web course produced with a learning management system. Thanks to them, an instructor can prepare a virtual or remote laboratories and easily add them to its online web course, automatically obtaining collaborative features. Also, any member (either a student or a teacher), as long as s/he is enrolled in the course, can use the system to invite other enrolled users to a real-time collaborative experimental session with any of the laboratories that belong to that course. The extension tools presented in this work are based in two different free and open source software programs: Moodle as the learning management system that offers the web environment and Easy Java Simulations for creating the virtual and remote laboratories with collaborative support.

1 Introduction

According to the constructivist learning theory ([9]), people generate knowledge and meaning (i) when they share their ideas and experiences and (ii) from the interaction between them. In this sense, nowadays Learning Management Systems (LMSs) include communication channels to allow user-to-user interaction in web courses. On the other hand, experiential learning is the process of making meaning from direct experience ([5]). This approach is specially important for scientific and technical courses, where experimentation is a key issue in the learning process. Virtual and

*carlos.jara@ua.es
†rheradio@issi.uned.es
‡ldelatorre@dia.uned.es
§jsanchez@dia.uned.es
¶sdormido@dia.uned.es
‖fernando.torres@ua.es
**francisco.candelas@ua.es

Remote Laboratories (VRLs) appeared to cover this necessity in distance education and also to serve as a didactical complement for traditional presential courses.

Even though constructivist web learning environments and VRLs already exist, there is still a lack of real-time interaction between users in both tools. According to the moment when the student–teacher (or student–student) interaction takes place, collaborative systems can be divided in two groups: asynchronous and synchronous ([1]). The first ones allow data exchange in flexible timetables and remote access to web–based course materials to carry out activities in an asynchronous way. They use collaborative tools such as e–mail or forums for on–line communication. However, this type of communication can cause feelings of isolation in the student and hence reduces his/her motivation ([2]). In addition, students do not receive instant feedback from their questions and cannot talk in real-time about results obtained in the learning activities. These limitations have been solved by applying synchronous technologies ([6]).

It is from the intersection of the three previous ideas (constructivism, experimental practice and real time interaction) that the concept of synchronous collaborative VRLs deployed into LMSs is born.

Two valuable software applications for e–learning and VRL development are Moodle and Easy Java Simulations (EJS). Moodle ([8]) is a widespread used LMS that supports constructivist learning, offering its users communication and interaction facilities. EJS ([3]) is a tool specifically created for designing and developing interactive virtual labs; that is, the simulation and the graphical user interface. Moreover, EJS not only has been successfully used and improved by many research groups to create virtual labs, but also the Graphical User Interfaces (GUIs) of their remote counterparts ([10]).

This paper describes an extension for Moodle and EJS that we have developed to provide the following features:

1. Synchronous collaborative support to any VRL developed with EJS; i.e., thanks to our extension, any existing VRL written in EJS can be automatically converted into a collaborative lab with no cost.

2. Deployment support of synchronous collaborative VRLs into Moodle.

The remainder of this paper is structured as follows. Section 2 briefly describes Moodle and EJS, justifying the suitability of such tools to deploy and develop VRLs. Sections 3 and 4 present our Moodle and EJS extensions, respectively. Finally, Section 5 offers some conclusions regarding this work.

2 Tools for creating and publishing VRLs

Section 2.1 justifies why Moodle is an ideal tool to publish on the Internet the VRLs. Afterwards, Section 2.2 introduces EJS, summing up its suitability for VRL development.

Preprint of the paper published in the 9th IFAC Symposium on Advances in Control Education, 2012

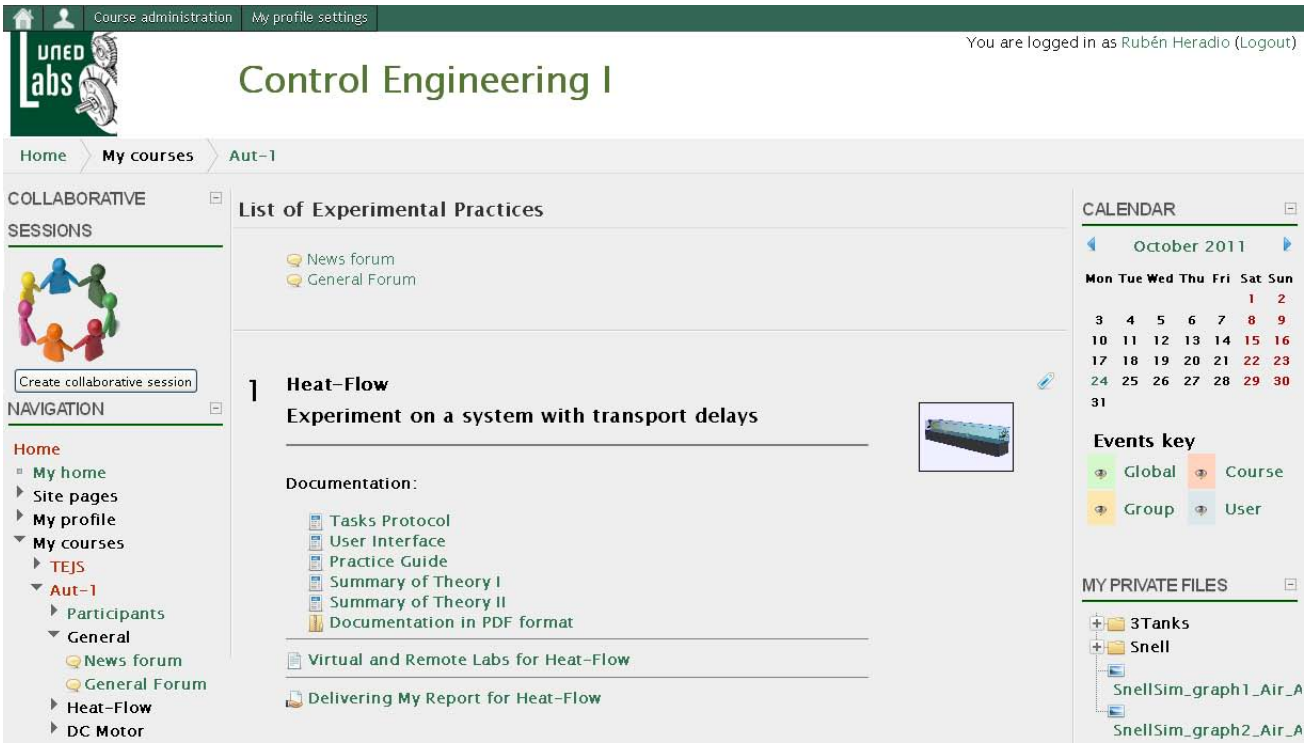

Figure 1: A Control Engineering Course in Moodle.

2.1 Moodle

VRLs do not provide by themselves all the convenient resources for distance education with all the implications that this methodology involves. Specifically, students must carry out their practical activities in an autonomous way and therefore, complementary web–based resources to the VRLs should be included. For this reason, for each VRL there should be available, not only a description of the phenomena under study and of the didactical setup of the experiment for remote experimentation, but also the task protocol students must follow to achieve the proposed goals. Moreover, a lab report must be prepared by the students with the data collected during the simulated and real experimentation that the instructors will correct. Moodle is a widespread LMS that may be customized to support such features. Figure 1 shows an example of a Moodle course we have developed for a Control Engineering subject.

Regarding our proposal of synchronous collaborative labs, Moodle provides two built-in blocks which are very helpful for user communication: *Online Users* and *Messages*. Figures 2 and 3 show how do they look like. By means of the first block, users can see other connected users in order to know who can they invite for a collaborative experimentation session. Thanks to the second one, users can text each other while working together with a virtual or remote lab (e.g., Figure 3 shows that the "Admin user" is working with user "Luis de la Torre'" in a collaborative session with a virtual experiment and has just received an instant message from him). In addition, Figure 3 shows the VRL corresponding to a heat-flow experiment ([11]) developed with EJS and deployed into Moodle with the EJSApp plugin we have created.

Figure 2: *Online Users* block.

2.2 EJS

EJS is a freeware, open-source tool developed in Java, specially designed for the creation of discrete computer simulations. The architecture of EJS derives from the *Model-View-Control* (MVC) paradigm, whose philosophy is that interactive simulations must be composed of three parts:

1. The *model* describes the process under study in terms of (i) variables, which hold the different possible states of the process, and (ii) relationships between these variables, expressed by computer algorithms.

2. The *control* defines certain actions that a user can perform on the simulation.

3. The *view* provides a graphical representation (either realistic or schematic) of the process states; i.e., the GUI of the simulation.

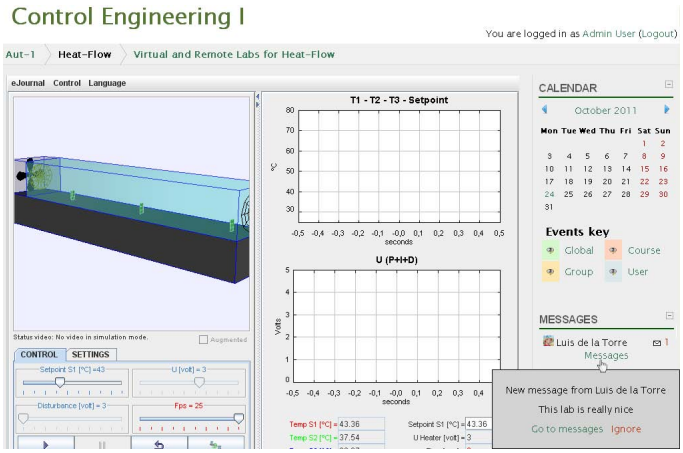

Figure 3: *Messages* block.

Figure4 shows the EJS user interface. EJS makes things even simpler eliminating the "control" element of the MVC paradigm and embedding one part in the view and the other one in the model.

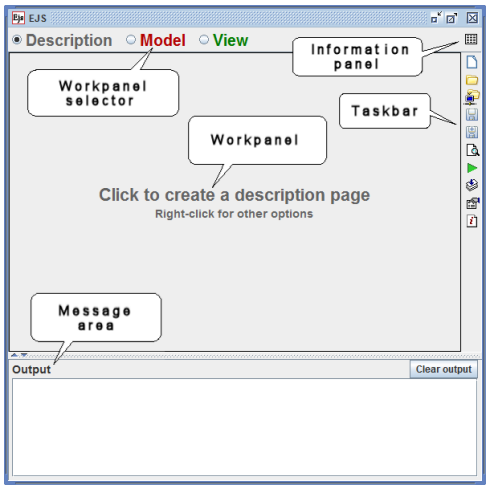

Figure 4: EJS user interface.

Applications are created in two steps: (i) the building of the model to simulate by using the built-in simulation mechanism of EJS, and (ii) the construction of the view to show the model state and its reactions to the changes made by users.

From a practical viewpoint, developers can create advanced interactive applets using EJS since it provides a simplified program structure, custom model tools (such as an advanced differential equation editor), and drag–and–drop view elements that let developers work at a high level of abstraction, thus speeding up the creation process. Developers input the qualified information on the simulation that only a human can provide (e.g., math equations, the initial model state,

the GUI's design...) and the program takes care of all the computer related aspects of creating a finished, independent Java applet.

Although EJS was originally designed for developing interactive simulations in physics, it has been augmented it to help create web–accessible laboratories in control engineering education. For this reason, recent releases of EJS support connections with external applications, such as LabView, Matlab/Simulink, SciLab, and SysQuake ([12]). Hence, EJS not only is useful to create virtual labs, but also the GUIs of their remote counterparts.

3 Extending Moodle

In order to support the one–click deployment of VRLs into Moodle, we have developed the EJSApp plugin presented in Section 3.1. Another plugin, named EJSAppCollabSession, that extends Moodle to support synchronous collaborative sessions of VRLs is described in Section 3.2.

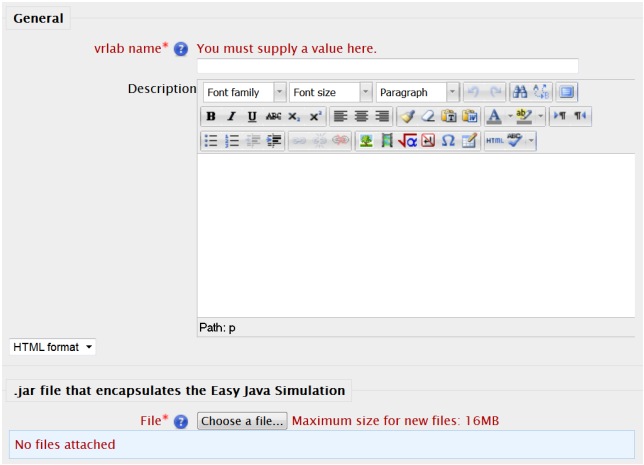

Figure 5: Adding a VRL to Moodle

3.1 EJSApp: deploying VRLs into Moodle

Module[1] EJSApp supports:

1. *Deploying VRLs written in EJS*[2]. Figure 5 shows the form EJSApp provides to add a VRL to Moodle. EJSApp uses the new Moodle 2 feature *File Picker*, enabling VRLs to be uploaded not only from the user computer but also from a variety of repositories such as Dropbox, Alfresco...

[1]There are different types of plugin in Moodle ([7]). Whereas EJSAppCollabSession is an *activity module*, EJSApp is a *block*.

[2]EJSApp supports the integration into Moodle of any kind of VRL developed with EJS (i.e., either collaborative or non–collaborative VRLs).

2. *Controlling user access to the deployed labs.* Figure 6 shows the form EJSApp provides to set the VRL availability (start and end dates when the VRL will be accessible to the students, minimum grade students need to get in other activities to have access to the VRL...)

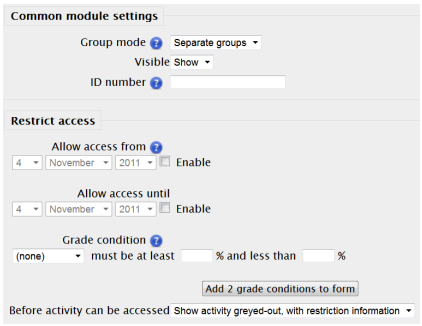

Figure 6: Controlling user access to a deployed lab

3. *Backup and restore.* EJSApp provides maintenance facilities for VRLs, packaging them into Moodle course backups.

3.2 EJSAppCollabSession: synchronous collaborative VRLs for Moodle

A fundamental issue in a synchronous collaborative system is the *floor control* ([4]). This term points out how the system components share the computational resources. The main objective of our proposal is to offer a shared VRL that can be controlled in real–time by different members of a virtual class (e.g., students and teacher) and to be able to share the same experiments like in a traditional classroom. In our case, the shared VRL is composed of an applet generated with EJS. Hence, there are two main kinds of components to coordinate: one session director's applet and some invited user's applets. The session director is responsible for starting, monitoring and closing a collaborative session. Thanks to our EJS extension, the session director's applet manages in real–time the virtual class and synchronizes all the invited user's applets. She has a list of invited user's connected in the virtual session and can disconnect any invited user's at any moment. In order to have a suitable floor control, connected invited user's applets are locked and they cannot interact with the shared VRL in a first moment. They are only allow to see in real-time what the session director is doing in the shared application. This way, the collaborative session avoids collisions of events which can cause unwanted and incoherent results. One example of this problem could be that the real equipment of the VRL becomes uncontrollable because of unsuitable user interactions.

In the following lines, the behavior of the EJSAppCollabSession block is illustrated:

1. From the session director point of view, a collaborative session is composed of the following steps:

   (a) A session is created by clicking the button "Create collaborative session" on the EJSAppCollabSession block (Figure 7, also seen at the left menu of Figure 1).

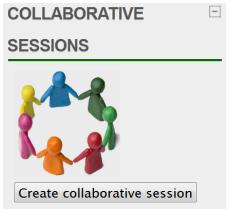

Figure 7: Starting a synchronous collaborative session

(b) From all the collaborative VRLs that have been uploaded to Moodle, using the EJSApp plugin, the session director selects which one is going to be used in the session (Figure 8). It should be noted that one VRL may be shared by several collaborative sessions simultaneously.

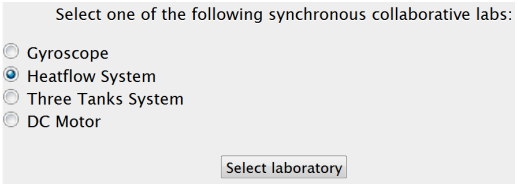

Figure 8: Selecting the collaborative VRL to be used within the session

(c) The session director selects the potential[3] participants to the session (Figure 9). When the "Invite participants" button is clicked, they will be notified with (i) an e-mail and (ii) a Moodle message (i.e., using the *Messages* block introduced in Section 2.1).

Figure 9: Selecting the session participants

(d) The VRL is accessed in collaborative mode, i.e., the session director's applet manages the virtual class and synchronizes all the invited user's applets.

[3] i.e., the selected participants may or may not decide to participate into the session.

(e) The collaborative session is finished by clicking the "Close collaborative session" on the EJSAppCollabSession block (Figure 10).

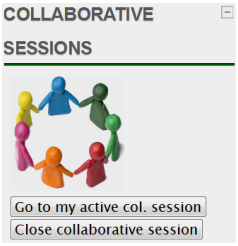

Figure 10: Closing a collaborative session

2. From an invited user point of view, a collaborative session is composed of the following steps:

(a) The user clicks on the button "Participate as an invited student" on the EJSAppCollabSession block (Figure 11). To prevent misuses of EJSAppCollabSession, its graphical interface changes to show just the valid choices available to a given situation (see Figures 7, 10 and 11). So, the "Participate as an invited student" button is only visible because the user has been invited to, at least, one collaborative session.

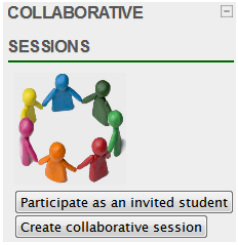

Figure 11: Accepting an invitation to a synchronous collaborative session

(b) From all the received invitations, the user selects in which session s/he wants to participate (Figure 12). Note that a course member can be invited to several collaborative sessions, but s/he can only participate in one of them at the same time.

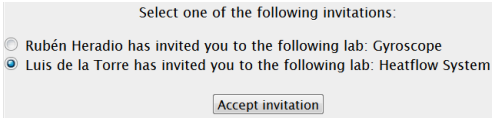

Figure 12: Selecting one of the available invitations

(c) The VRL is accessed in collaborative mode.

(d) The user stops participating in the session when (i) s/he decides to leave it or (ii) when the session director closes it. In the former case, the user is free to enroll either to that session again or to any of the current available invitations.

4 Extending EJS

Firstly, Section 4.1 sums up the collaborative features that our EJS extension adds to VRLs. Afterwards, Section 4.2 describes how to convert any VRL into a collaborative one.

4.1 Collaborative features of EJS VRLs

Our EJS extension provides the session director with the control panel shown in Figure 13. It includes a list of the invited users connected to the collaborative session (e.g., Figure 13 shows that "Luis de la Torre" is the session director and, "Rubén Heradio" and "Jose Sanchez" are the invited users). Using such list, the session director can perform the following tasks:

1. Disconnecting any invited user at any moment.
2. *Assigning the chalk* to an invited user. With this feature, the session director gives permission to control the shared VRL to a specific invited user, by selecting him from the list. The chalk only enables a student to manage the VRL, but not the collaborative session.

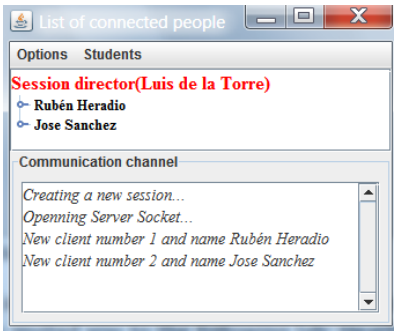

Figure 13: Control panel of the collaborative session

Figure 14 depicts the communication framework that underlays the collaborative sessions. When a session begins, users just interact with the Moodle server by downloading the applet that implements the VRL they are going to use in the session (see dashed lines in Figure 14). On the other hand, users participating in a session interact each other through a server–less collaboration over TCP and UDP protocols (see solid lines in Figure 14). Thus, the communication framework we propose supports multiple simultaneous sessions without overloading the Moodle server.

Invited users' applets are connected directly to the session director's applet in a peer-to-peer (P2P) centralized overlay network. In contrast with server-based approaches, our e-learning system

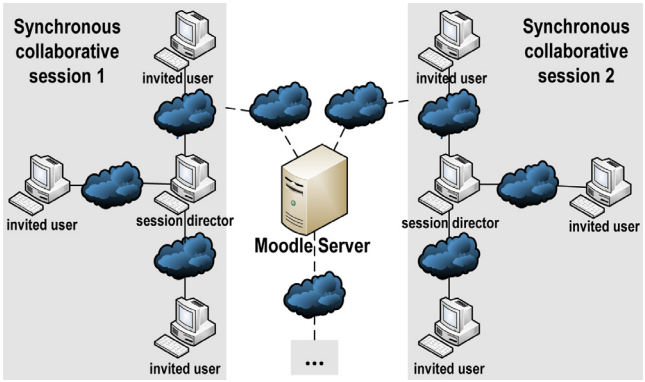

Figure 14: A network of collaborative sessions

is focused in a server-less architecture. This communication method avoids delays caused by the server processing in the data flow because the communication engine is embedded in the Java applets downloaded by the users. In addition, the number of network connections can be substantially decreased because the session director's applet can manage the session, the floor control, and the data exchange having higher control over the invited user applets. As stated, the P2P network is centralized around the session director's applet. This last application is the central node of the collaborative class and contains a multithread communication module which manages the synchronization of all the applets that compose the shared VRL. Invited users' applets are connected to the central node over TCP and UDP sockets performing a centralized network.

To synchronize in real-time all the applets connected to the virtual class a method based in Java object tokens ([4]) is used. Java object tokens are small update messages which contain a String object that defines the action to be performed by other applets of the same session. The small amount of sent information optimizes network utilization and reduces the connection delay.

Since all the applets must be in the same state at any time, it is necessary to synchronize them. The developed communication framework provides a transport service suitable for all update data: a TCP–based channel for reliable messages and an UDP–based channel for fast messages. The TCP channel is used to update all the variables of the application because the transmission of the values needs the reliability of an ACK-based protocol. The UDP channel is used to transmit the small changes in the user-interface and this requires to be quickly updated in the rest of the applets.

4.2 Converting VRLs into collaborative ones

Thanks to our EJS extension, adding collaborative capabilities to an applet is most simple. Whatever the applet looks like and whatever it simulates or control remotely, users just need to follow the next steps:

1. Open the *Information Panel* (Figure 4) and select the *Run Options* tab in EJS to open the dialog window shown in Figure 15.

2. Select the *Add support for collaborative applets* option. This option embeds into the applet the communication engine described in Section 4.1.

3. Select the *Add support for Moodle* option. This enables the application to communicate with the plugin EJSAppCollabSession described in Section 3.2.

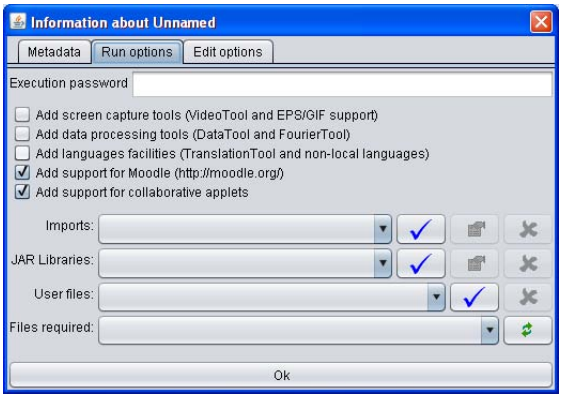

Figure 15: Creating a collaborative VRL for Moodle

5 Conclusion

In this work, we have presented a way for easily developing and integrating EJS collaborative virtual and remote laboratories into a Moodle course. This is done thanks to the extensions developed by the authors: 1) the collaborative features now offered by EJS and 2) a Moodle extension to take care about the collaborative sessions management. The result is a web environment in which users can invite other users to perform laboratory activities in common, share ideas, solve problems in team and ask and answer each other doubts regarding the experiments.

Finally, it is important to highligh that although this methodology has been applied in this work to a control engineering laboratory, it can be used with any type of laboratory (and so, it is also suitable for science courses, for example), as long as it is created with EJS and integrated into a Moodle course.

References

[1] G. Bafoustou and G. Mentzas. Review and functional classification of collaborative systems. *International Journal on Information Management*, 22:281–305, 2002.

[2] M.N.K. Boulos, A.D. Taylor, and A. Breton. A synchronous communication experiment within an online distance learning program: A case study. *Telemedicine journal and e-health*, 11(5):583–93, 2005.

[3] W. Christian and F. Esquembre. Modeling physics with easy java simulations. *The Physics Teacher*, 45:475–480, 2007.

[4] Hans-Peter Dommel and J. J. Garcia-Luna. Floor control for multimedia conferencing and collaboration. *Multimedia Systems*, 5:23–38, January 1997.

[5] C. M. Itin. Reasserting the philosophy of experiential education as a vehicle for change in the 21st century. *The Journal of Experimental Education*, 22:91–98, 1999.

[6] O. Marjanovic. Learning and teaching in a synchronous collaborative environment. *Journal of Computer Assisted Learning*, 15:129–138, 1999.

[7] Jonathan Moore and Michael Churchward. *Moodle 1.9 Extension Development*. Packt Publishing, 2010.

[8] William Rice. *Moodle 2.0 E-Learning Course Development*. Packt Publishing, 2011.

[9] D. Satterly. *Piaget and Education*. The Oxford Companion to the Mind - Oxford University Press, 1987.

[10] H. Vargas, J. Sanchez, C.A. Jara, F.A. Candelas, F. Torres, and S. Dormido. A network of automatic control web-based laboratories. *IEEE Transactions on Learning Technologies*, 4:197–208, 2011.

[11] Hector Vargas, Gonzalo Farias, Sebastian Dormido, Jose Sanchez, Raquel Dormido-Canto, Natividad Duro, Sebastian Dormido-Canto, and Francisco Esquembre. Web–based learning resources for automation technicians vocational training: Illustrated with a heat–flow and a liquid level. In *Advances in Control Education*, Madrid, Spain, 2006.

[12] Héctor Vargas, José Sánchez-Moreno, Sebastián Dormido, Christophe Salzmann, Denis Gillet, and Francisco Esquembre. Web-enabled remote scientific environments. *Computing in Science and Engg.*, 11:36–46, May 2009.