A systematic two-layer approach to develop Web-based experimentation environments for control engineering education

H. Vargas, J. Sánchez*, N. Duro, R. Dormido, S. Dormido-Canto, G. Farias, S. Dormido
Department of Computer Sciences and Automatic Control, UNED, c/. Juan del Rosal 16, 28040 Madrid, Spain
F. Esquembre
University of Murcia, 30100 Campus Universitario de Espinardo, Murcia, Spain
Ch. Salzmann, D. Gillet
Laboratoire d'Automatique, EPFL, CH-1015 Lausanne, Switzerland

Abstract – This paper introduces the systematic approach currently used by the Computer Sciences and Automatic Control Department of the Spanish University for Distance Education (UNED) to develop Web-based laboratories for the distance learning of topics with high technical contents such as *control engineering*. This approach differentiates two layers in the construction of Web-based laboratories: the *experimentation layer* and the *e-learning layer*. For the experimentation layer, *LabVIEW* and data acquisition boards from National Instruments are used to create the server-side applications, and *Easy Java Simulations* for the client-side interfaces. For the e-learning layer, the *eMersion* environment is used to support the required flexible educational scheme. This paper describes the programming techniques implemented and the design considerations that justify these particular choices. As an illustrative case-study, an example of development of a web-based application is discussed, in which an electrical drive servo-motor is introduced as a convenient setup to practice with motion control applications. Other significant examples of remote experiments developed by the authors are also reported.

Keywords – Remote laboratory, distance learning, control engineering education, web-based experimentation, collaborative learning.

* Corresponding author: phone +34 91 398 7146, fax: +34 91 398 7690
E-mail address: jsanchez@dia.uned.es (J. Sánchez)

1. INTRODUCTION

New technological advances in communication networks are steadily being introduced to improve both the quality of service and the available bandwidth of Internet to accommodate the increasing world-wide demand of resources. The availability and capabilities of these new communication facilities, combined with the fast development of Web technologies and the generalisation of off-the-self components for data acquisition and control of industrial processes, now enable students to avoid attendance to traditional laboratories. Students are encouraged to use tele-presence systems, transforming the remote experimentation from a subject of research into a reality accessible to everybody [1]. Control Engineering is one of the technical areas in which the impact of new technologies to develop novel approaches to distance learning is most noticeable [2]. A great number of research groups, universities, and academic institutions all over the world are currently implementing virtual and remote control laboratories with one objective in common: to enhance traditional teaching processes by taking advantage of modern technologies [3].

Unfortunately, these developments are often isolated efforts of independent groups that use different programming techniques, different software tools, and different hardware facilities. Based on these premises, it would be very convenient for the control education community to establish a common methodology, procedure, or approach to describe, implement, and interconnect each of the parts that compose a Web-based laboratory. This approach should include a Web-based experimentation platform that supports both the student-teacher interaction and the collaborative processes among students in a way similar to how it takes place in the traditional classroom. This paper

gives some recommendations and insights about how to create such virtual and remote control laboratory applications, and how to make them more accessible to students.

The paper is organized as follows. Section 2 describes the *experimentation layer,* which includes the fundamental aspects required to create virtual and remote control laboratories using *LabVIEW* and *Easy Java Simulations*. Section 3 presents the *eMersion*-based *e-learning layer,* which implements the complementary Web-resources that users need to make a correct use of the experimental application. Section 4 lists all the steps needed to develop a virtual and remote control laboratory completely integrated in *eMersion* by using a servo-motor as didactical setup. Section 5 provides two additional examples of remote experiments developed by the authors. Finally, some conclusions and considerations about further work are given.

2. THE EXPERIMENTATION LAYER

Excluding specific efforts from a few universities [4]-[6], there is no unified methodology or procedure to design, develop, and publish virtual and remote laboratories on the Internet. However, analysing the growing number of publications (journals and conference proceedings) related to Web-based laboratories, it is possible to distinguish a set of basic features that are frequently used as methodological steps for designing an integral environment for remote, pedagogical-oriented experimentation. The first feature corresponds to the design of an experimentation layer. This layer includes the creation of the interactive client-side interfaces and the server-side applications that run the real-time control loop and the transmission components for exchanging information with the remote user interface. The second main feature is the

design of an e-learning layer necessary to deploy the Web-based resources over the Internet.

In our approach, the experimentation layer has been developed following the usual client-server architecture [7]. The choice of this architecture is due to its great flexibility, performance, and adaptability to situations in which either the continuity of the interaction, the orientation to the connection, or the quality of services play a fundamental role. To illustrate more in-depth the experimentation layer, the next two subsections describe some design considerations and programming techniques for developing Web-based control experimental applications.

2.1. Building the Server-Side

Figure 1 shows the standard structure of a feedback loop in control systems theory. The operations performed in a feedback loop follow a standard sequence. The operations labelled (a) and (b) in Figure 1 are, respectively, the measurement of the analogue output signal from the process and its conversion to digital format. Once the conversion is completed (b), this value is compared to the set point value (c) to obtain the error signal. With this information the control algorithm computes the control signal (d) that is then converted to its analogue representation (e) and applied to the physical process. This sequence of operations is known as the *control task* that must be carried out at a fixed time-interval called the *sampling period* that is chosen in accordance with the dynamics of the physical process. This fixed sampling interval or cycle time defines implementation constraints when using traditional control analysis and design techniques where a constant sampling period is often assumed [8]. The notion of *real-time control* is based on the above implementation constraints where the computer operations rely on a deterministic time occurrence. The implementation of this type of

applications in control engineering education and, especially, in virtual and remote experimentation across the Internet is commonly made using *LabVIEW* software [9].

Using LabVIEW for remote experimentation.

Because of the hard constraints in its sampling period (reading from sensors, executing the control algorithm, and writing to actuators), the execution of the control task must take place as regularly as possible. Using LabVIEW, such regularity is achieved by using threads and *DAQ* board interrupts. However, to run a local version of the control task through the Internet, some modifications are required.

In particular, it is necessary to add the LabVIEW code that handles the communication with the remote client and with the video camera that provides the adequate visual feedback of the behaviour of the plant in the laboratory. Figure 2 shows a general scheme of this idea. The three boxes on the server-side represent each of the tasks required. Since the control task must keep the constraints in the cycle time, it is necessary to ensure its determinism in the execution by assigning to it the highest priority. The communication and video tasks are run in parallel to the control task on different threads with lower execution priorities. Finally, it is also necessary to perform the methods that share data among tasks respecting the determinism of the control task.

LabVIEW shows very adequate here, since it provides simple multi-thread programming. Since *LabVIEW 7.1, Timed Loop* blocks allow the author to program multiple threads in a single virtual instrument (VI). These threads can run at different sampling periods and with different priorities.

Figure 3 shows a typical *LabVIEW* diagram to implement the architecture of Figure 2 for a virtual and remote control laboratory application. The three loops in the block diagram run concurrently to perform the main tasks in the server-side: control, communication, and video capturing.

The control task is a time-critical activity running at a sampling period of 20 ms with a priority greater than the other two threads. The *AI One PT* block reads the analogue input signal from the sensor, its output is compared to the set-point input of the *PID* block and the result is fed into the *AO One PT* block that sends the resulting value to the actuator, thus completing the control task. The data structure composed by the set point value, the PID control parameters, the command to the actuator, and other variables is known as the *control vector,* and is sent from the communication task to the control task through *RT FIFO* queues blocks. These variables are produced when users interact with the client interface. On the other hand, the data array formed by the values sent to the actuator, the measurement from the sensor, the current time, and other variables is known as the *state vector* and is transferred from the control task to the communication task through *RT FIFO* queues blocks.

The video task is a non-time-critical activity since the loss of video frames is generally acceptable for the user. For most applications, sending five images per second is enough to get an adequate visual feedback of the remote system.

Finally, the communication task concatenates the current measurements (state vector) and the video frame (video vector) and then uses the *Comm WRITE* block to send the information to the client. In parallel, the control vector is received through the *Comm*

READ block from the clients and passed to the control task using the *RT FIFO* queues. The TCP protocol is used in both implementations because it guarantees packet delivery and bandwidth adaptation, albeit at the cost of potentially added transmission delay [10], [11]. However, for unloaded links such the one found in a LAN, the transmission delay added by TCP can be neglected. An alternative is the use of the UDP protocol which provides a better control of the transmission delay. However, UDP neither has guaranteed packet delivery mechanism nor bandwidth adaptation mechanism, thus it is left to the solution designer to implement these features.

2.2. Building the client-side interfaces

Easy Java Simulations (*Ejs*) [12] has been chosen to develop the client interfaces. *Ejs* is a free software tool designed to create scientific simulations in Java, which is being used successfully to develop virtual and remote control laboratories in several university departments [13]-[21]. The use of Java is justified by its wide acceptance by the Internet community, and due to the large number of supported platforms.

Simulations in *Ejs* are structured in two parts; the *model* and the *view* (see Fig. 4). The model describes the behaviour of the system using variables, ordinary differential equations, and Java code. The view provides the visual and graphical aspects of the simulation. Both parts are closely interconnected since the evolution of the model state affects the view and the interaction of the user with the view affects the model.

The following considerations are important when creating the experimentation interfaces with *Ejs*:

- A Java method to send and receive data packets to/from the server-side must be implemented to control and obtain information from the plant.
- Some data types in Java are different to data types in *LabVIEW*. Therefore, a Java class to convert those *LabVIEW* data types into Java data types is required.
- The same user interface should allow the user to run both the real system remotely or a local (virtual) simulation of it. When the interface is running in simulation mode, the variables associated to the system state are updated according to the evolution of a mathematical model of the process (written in the *Ejs* built-in ODE editor). However, in remote mode, these variables are reflects the real plant measurements.
- The graphical user interface must be simple and intuitive. An interface with too many sliders, buttons, graphical elements, scopes, and auxiliary windows incorrectly distributed would puzzle users and thus decrease their motivation [22].
- Providing a mean to remotely generate disturbances on the real system allows users to test the robustness of its controller design.

Creating an application with *Ejs* requires only a minimum knowledge of Java. *Ejs* provides extensive scaffolding to create the model of the system by defining the variables that describe the system state and specifying the constitutive equations that govern how this state changes with time. In addition *Ejs* includes a library of ready-to-use graphical elements that can be used to build sophisticated and interactive views by simple drag and drop (See *View* panel in Fig. 4). The graphical properties of the elements used in the view can be linked to the model variables, producing a bi-directional flow of information between view and model: any change of a model variable is automatically displayed in the view, and, reciprocally, any user interaction

with the view produces an automatic modification of the value of the corresponding model variable.

Once the developer has defined the model of the process (including the Java code to exchange information with the server-side in remote mode) and the view of the interactive user interface, *Ejs* generates the complete Java source code, compiles it, packages the resulting object files into a compressed file, and generates HTML pages containing the narrative and the applet. The user can then directly run the applet and/or publish it over the Internet. The servo-motor example of Section 4 illustrates all the stages required for the creation of a client-side application using *Ejs*.

More information about *Easy Java Simulation*, the last release of this software tool and complete user's manuals can be freely downloaded from *http://www.um.es/fem/Ejs/*.

3. USING EMERSION AS THE E-LEARNING LAYER

This section describes *eMersion*, a collaborative environment developed at the Ecole Polytechnique Fédérale of Lausanne (EPFL) [23]. This tool supports all the fundamental aspects related to the Web-based collaborative works existing among students and instructors. From a pedagogical perspective, *eMersion* is a very convenient environment for distance learning in engineering disciplines because it incorporates all the elements needed to complete an experimental session emulating the work carried out in the traditional laboratory [24], [25].

3.1. The eMersion environment

The Department of Computer Sciences and Automatic Control at UNED is using the *eMersion* environment to support the experiments carried out by its control engineering students across Internet.

Figure 5 shows the functional structure of the collaborative environment. The system is composed of four parts: the *eJournal*, the *experimentation console*, the *on-line documentation*, and the *navigation bar*. The eJournal module provides a shared space to facilitate the communication and collaboration among students and instructors during the learning process. In this shared space students can store, retrieve, and exchange their experimental results and documents. Documents and files stored in the *eJournal* are called *fragments*.

The experimentation console module contains the Java applet developed with *Ejs*. Using it, students can perform their laboratory assignments in a similar way to a traditional laboratory by accessing to the real laboratory resources through a local network (remote laboratory) or by simulating the physical phenomenon using a mathematical model (virtual laboratory). The on-line documentation module provides a portfolio with theory, assignments, task protocols, and user's manuals of the environment plus any additional information needed to successfully complete an experiment. The protocol describes the tasks to be performed by students during the laboratory session. These tasks are of two kinds: *prelabs* and *labs*. The first ones are assignments that students must carry out in simulation mode to gain knowledge about the real process. Only when the prelabs are mastered, will students obtain remote access

to the plant located in the laboratory. Through the navigation bar the user can access the different modules described above. Aditionally, the *eMersion* environment provides tools to simplify the organization of work-groups and offer asynchronous messaging, news, and e-mail notification services.

3.2. Integrating Ejs applications in eMersion

Ejs offers several predefined Java built-in methods to ease the integration of client-side applications in eMersion. These methods allow *Ejs* applications to exchange fragments of data and snapshots of the graphical scopes with the e-Journal.

Figure 6 shows a part of the structure of the *Ejs* view corresponding to the servo motor application discussed in the next section. The left part on the main window displays the tree-like structure of view elements that make the graphical interface. The right-hand side of this window contains the individual graphical elements that *Ejs* offers to developers to elaborate the views: graphical containers, panels, buttons, text-fields, labels, menus, 2D and 3D drawing elements, control symbols, etc. These elements are selected by dragging and dropping them to the view tree. The Figure shows three auxiliary windows with the Java code executed when the user selects the menu options labelled *SaveGraph*, *StartRecord*, and *StopRecord*. This Java code uses some *Ejs* built-methods that send information to *eMersion*. The functionality of these methods is the following:

_saveImage: Takes a snapshot of the mixed-signal scopes available in the view. Images are stored in GIF format.

_saveText: Records a text file (`.mat` extension) with the temporal evolution of the experiment. Once the experience is completed, the user can load the data in the Matlab workspace for analysis.

_saveState: Stores the system state in a binary file (`.dat` extension).

_readState: Reads a file previously recorded with *_saveState*.

All the data files produced with these predefined methods are automatically stored in the e-Journal, so the *Ejs* developers do not need to include extra Java code to exchange information between both environments. A set of technical examples about how to use these methods in *Ejs* applications is available in the last *Ejs* distributions.

4. A SERVO MOTOR PROTOTYPE

As an illustration of these ideas, the development of a complete virtual and remote control laboratory of an electrical drive servo motor (*eElab* for short) is presented in this section. The *eELab* equipment is located in the Automatic Control Laboratory of the Department of Computer Sciences and Automatic Control at UNED. The prototype is part of a set of didactical setups specially selected to be used in remote experimentation environments. Other prototypes available are a quadruple-tank, a three-tank and a two-tank system, and a heat-flow apparatus. Currently all these prototypes can be manipulated via Internet by accessing on-line courses on control engineering available in the *eMersion* environment installed on the UNED servers. These courses are based on electronic documents, multimedia presentations, interactive simulators, and virtual/remote laboratories.

4.1. The server-side application for the eElab

Table 1 contains a summary of the technical features of the hardware and software elements used to develop the *eELab* server application. Figure 7 shows a detailed model of the servo motor developed by Schorderet Technics [26]. In this scheme, the angular speed of the motor is modelled as a first order system and the angular position is obtained by integration of the speed. *Kv* and *Kp* factors are constants used to scale the physical values of the generator and position sensor in the output values read by the I/O card. Position and speed are controlled by a DC voltage (-5 to +5 volts) applied to the motor through an amplifier.

Considering the time constant of the motor *tau* (0.253 sec), a sampling period of 20 ms is chosen to satisfy the sampling theorem. To guarantee the sampling period accuracy in the control task, a timer found in the DAQ board is used. This timer conveniently permits to synchronize the AD and the DA conversion.

The control algorithm is a standard PID controller with set-point weighting, anti-windup mechanism, and filtering of the derivative action [8]. Figure 8 shows a block diagram of the controller and the code of the discrete version of the PID programmed in LabVIEW.

Within the video task (Fig. 9), images are acquired using the built-in firewire camera. These images are read, compressed in JPEG format, and passed-on to the communication task at 10 frames per second, a value that provides the distant user with an acceptable visual feedback of the real process. These images are then decompressed and rendered within the client *Ejs* application. The communication task performs the data exchange between the client and the server. At the server side, the data produced

by the control task (state vector) and the video task (JPEG image) are concatenated and sent to the client-side using the *Comm WRITE* block VI. The data sent from the clients (control vector) are read using the *Comm READ* block VI and transferred to the control task.

4.2. The client-side of the eElab

Figure 10 presents the client-side interface (experimentation console) developed in *Ejs* to operate in virtual mode (using the model described in the previous section) or in remote mode with the *eELab*. The window is divided into two parts. The left part contains an image of the motor and a control panel used to define different system parameters. The virtual representation has been developed copying the frontal view of the real motor. Hence, any variation of the system state during the simulation mode will be displayed as a rotational movement of the disk. However when the user works in remote mode, this virtual representation will be replaced by the video images sent from the server. When the application is working in remote mode, the *Augmented* option can be enabled. In this configuration a virtual representation of the motor is overlaid on top of the video image. When the available bandwidth is not large enough, the virtual representation can serve as a substitute of the video stream.

The control panel contains two subpanels accessible by tabs. The first tab, named *Control* (visible in Fig.10), lets the user choose the type of experience (virtual or remote), switch between position or speed control, and change the system dynamics (for instance, modifying the set point or introducing disturbances by varying the brake). The four buttons located at the bottom of the control panel (*Play*, *Pause*, *Reset* and *Connect*) allow specifying how the system must work. The *Connect* button starts the remote mode

by connecting to the real process running at the university laboratory. By default, the application begins working in simulation mode. The second tab presents sliders and buttons to tune the PID controller. Finally, a status bar below the control panel displays messages related to the state of the connection between client- and server-sides.

The right part of the main window contains two mixed-signal scopes and numerical fields that show the evolution of the process variables. The status bar located at the bottom indicates the time remaining for the remote session. In the upper part there are three menus labelled *eJournal*, *Control,* and *Language*. The *eJournal* menu provides commands to take a snapshot of the evolution of the system in GIF format and to save a record of the value of its variables in plain text format. The *Control* menu allows switching between manual (the voltage applied to the motor is driven by the user) and automatic control mode (the voltage to the motor is calculated automatically by the PID controller).

4.3. The eELab control laboratory integrated in eMersion

Figure 10 shows the general view of the *eELab* remote control laboratory completely integrated in *eMersion*. The upper frame displays the navigation window, which allows users to access the different modules of the experimentation environment.

The *Objective* area contains a textual description of the topic covered in the laboratory. The *Navigation* area contains a group of icons that grant users access to the different modules described in Section 3. The window on the right contains the *eJournal*. Data fragments produced from the user interface will automatically be stored in the *eJournal*. These fragments can later be used by students to generate reports.

5. OTHER EXAMPLES OF REMOTE EXPERIMENTATION

This section describes two other applications developed using the design considerations described in this paper. These Web-based applications have been developed to present specific PID control concepts, as the temperature control of a heat-flow system, and level control for a set of interactive tanks. These two laboratory experiments provides the same capabilities as the eLab and can both be remotely controlled or used in simulation mode.

5.1. The heat-flow web-based laboratory

The temperature control system (Heat-Flow) has been developed by Quanser Consulting (Figure 11) [27]. This system presents features to study concepts of temperature flow control system with transport lag and identification techniques.

The setup consists of a duct with the following components: a heating element and a blower located at both ends of the structure, and three temperature sensors *S1*, *S2* and *S3* along the duct. The power delivered to the heater and the fan speed can be controlled using the analogue signals *Vh* and *Vb*. Fan speed is measured using a tachometer that produces an analogue signal *Vt* which can be used to design speed controllers. A model is required to build the virtual laboratory simulation. Its transfer function is:

$$G(s) = \frac{C_n(s)}{V_q(s)} = \frac{K_p(1+sT_3)}{(1+sT_1)(1+sT_2)} e^{-Ls}$$

$C_n(s)$: Air temperature at sensor n.

$V_q(s)$: Voltage applied to the heating element.

where steady-state gain *Kp*, lags *T1*, *T2*, *T3,* and delay *L* depend on which of the three sensors is used to close the temperature control loop.

Figure 12 shows the main window of the experimentation console when running in virtual mode. To reduce the student learning curve the experimentation consoles in the proposed Web-based laboratories are always very similar.

In the case of the heat-flow console, at the top of the window there is a 3D representation of the setup whose colour varies according to the air temperature inside the duct. At the bottom of the left panel, there are control elements that allows the user to choose the type of experience (virtual or remote), switch the control mode (open or closed loop), and a set of sliders and buttons to change the operating conditions (for instance, changing the set point or introducing some disturbance by varying the heater voltage *Vh*).

Figure 13 shows the experimentation console when users work with the didactical setup instead of with the heat-flow simulation. In this case, the 3D view of the heat-flow is replaced for a real-time video image of the apparatus. Since the air heating process can not be appreciated in the image, users can choose to overlap the video and the virtual 3D visualization in order to appreciate the change in air temperature.

5.2. The three-tank laboratory

The three-tank setup has been proposed as a benchmark system for different purposes, for instance, test system for fault detection, identification and multivariable control, and

reconfigurable control. To develop this laboratory, the DTS200 three-tank system manufactured by Amira GmbH [29] was selected (Figure 14). The plant consists of three cylinders *T1*, *T2,* and *T3* with cross section *A*. These cylinders are connected serially with each other through pipes of cross section *Sn*. On the left side of the tank *T2* there is the single outflow valve, which has a circular cross section *Sn*. The output-flowing liquid is collected in a reservoir located under the three tanks. This reservoir supplies pumps 1 and 2 with liquid. These pumps represent the input flows of tanks *T1* and *T2*. In the global control system, the pump flow rates correspond to the input signals of the process, and the levels of *T1* and *T2* are the output signals. All of them can be used for control purposes.

As in previous cases, the experimentation console of this laboratory was first developed using *Ejs* and the resulting simulation was then integrated into the eMersion environment. Figure 15 shows two views of the console. The design of the views is very similar to the laboratories for temperature control and for the electrical drive. In this laboratory, users have again the option to overlap to the video a 2D representation of the setup and compare and contrast the results of the simulation and of the real system.

Students operate the system in a very intuitive way. For instance, they can interactively change the set points for tanks T1 and T2 by dragging two level arrows up and down to set the desired levels in the controlled tanks. The opening and closing of the different valves, leaks and pumps in the system can also be operated using different sliders.

In this laboratory, the control objective is to regulate the level of the liquid in tanks *T1* and *T2*. A schematic graph of a possible multi-loop control strategy for this process is

shown in Figure 16.a, where it is assumed that *C1* and *C2* are the controllers. The control strategy implemented in this laboratory is a decentralized PID controller (one for each loop). A detailed block diagram of this structure is given in Figure 16.b, where *K* is the proportional gain, Ti the integral time, *Td* the derivation time, *N* is a parameter to limit the high frequency gain ($N \in [2, 20]$) and *b* is the set-point weighting parameter ($b \in [0, 1]$). By properly modifying the parameters of the PID controller, other variants, such as PI, PD, or P, are easily obtained.

Manual control is also possible in this laboratory. Using this control, students can clearly observe that the system is non linear, since the flow of the system is a function of each of the tank heights.

6. CONCLUSIONS

Virtual and remote experimentation for engineering education can be considered as a mature technology. However, the process of transforming a classical control experiment located in a conventional laboratory room into an interactive Web-based laboratory is not an easy task. This paper provides a systematic approach for the fast development of remote laboratories using three tools: Easy Java Simulations, LabVIEW and eMersion. The proposed component approach eases the development of online experimentation environments. It also provides an effective scheme to switch between simulation and actual operation of real systems, which is a key feature for hands-on learning activities in engineering education.

The proposed approach has been implemented at the UNED for deploying virtual and remote laboratory experiments that are throughout described in this paper. Thanks to

this new facility, the distance-learning students enrolled in automatic control courses do not have any more to travel and stay in Madrid to complete their laboratory assignments.

ACKNOWLEDGEMENTS

This work has been supported by the Spanish CICYT under grant DPI2004-01804.

REFERENCES

[1] Dormido S. Control learning: Present and future. Annual Control Reviews 2004; 28: 115-136.

[2] Magin M, Kanapathipillai S. Engineering students understanding of the role of experimentation. European Journal of Engineering Education 2008; 25(4):351-358.

[3] Gillet D, Nguyen A, Rekik Y. Collaborative Web-Based Experimentation in Flexible Engineering Education. IEEE Transactions on Education 2005; 48(4):696-704.

[4] Pastor R, Sánchez J, Dormido S. A XML-based for the Development of Web-based Laboratories focused on Control Systems Education. International Journal of Engineering Education 2003; 19(3):445-454.

[5] Casini M, Leva A, Schiavo F. AIRES: a Standard for Web-based Remote Experiments. Proceedings of the 16th IFAC World Congress, Prague, Czech Republic, July, 2005.

[6] Pastor R, Martín C, Sánchez J, Dormido S. Development of a XML-based Lab for Remote Control Experiments on a Servo Motor. International Journal of Electrical Engineering Education 2005; 42(2):173-184.

[7] Ko CC, Chen BM, Chen J. Creating Web-based Laboratories. 1st ed. London: Springer; 2004.

[8] Astrom K, Hagglund T. Advanced PID Control. ISA - The Instrumentation, Systems, and Automation Society: Research Triangle Park. NC; 2006.

[9] Salzmann Ch, Gillet D, Huguenin P. Introduction to Real-time Control using LabVIEW^{TM} with an Application to Distance Learning. International Journal of Engineering Education 2000; 16(3):255-272.

[10] Lim D. A Laboratory Course in Real-Time Software for the Control of Dynamic Systems. IEEE Transactions on Education 2006, 49(3):346-354.

[11] Salzmann Ch, Gillet D, Müllhaupt P. Real-Time Interaction over the Internet: Model for QoS Adaptation. Proceedings of the 16th IFAC World Congress, Prague, Czech Republic, July 2005.

[12] Esquembre F. Easy Java Simulations: A software tool to create scientific simulations in Java. Comp. Phys. Comm. 2004; 156:199-204.

[13] Sánchez J, Dormido S, Esquembre F. The learning of control concepts using interactive tools. Computer Applications in Engineering Education 2004;13(1):84-98.

[14] Dormido S, Martín C, Pastor R, Sánchez, Esquembre F. Magnetic Levitation System: A Virtual Lab in Easy Java Simulation. Proceedings of the American Control Conference (ACC'04), Boston, EEUU, pp. 3215-3220.

[15] Sánchez J, Esquembre F, Martín C, Dormido S, Dormido-Canto S, Canto RD, Pastor R. Easy Java Simulations: An Open-Source Tool to Develop Interactive Virtual Laboratories Using Matlab/Simulink. International Journal of Engineering Education 2005; 21(5):798-813.

[16] Buccieri D, Sánchez J, Dormido S, Mullhaupt P, Bonvin D. Interactive 3D Simulation of Flat Systems: The SpiderCrane as a Case Study. Proceedings of the

44th IEEE Conference on Decision and Control and European Control Conference (CDC/ECC 2005), Sevilla, Spain.

[17] Duro N, Dormido R, Vargas H, Dormido S, Sánchez J, Pastor R. The Three-Tank System: A Remote and Virtual Control Laboratory using Easy Java Simulations. Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference (CDC/ECC 2005), Sevilla, Spain.

[18] Hernández AM, Mañanas MA, Costa-Castelló R. RespiLab: A virtual laboratory for the analysis of human respiratory control system. Proceedings of the 7th IFAC Symposium on Advances in Control Education (ACE 2006), Madrid, Spain.

[19] Nguyen V, Cécile N, Konrad S. Simulator of an industrial wastewater treatment process in Easy Java. Proceedings of the 7th IFAC Symposium on Advances in Control Education (ACE 2006), Madrid, Spain.

[20] Mago A, Medina W, Fermín L, Grieco J, Fernández-López G, Cappelletto J. Control systems simulator for wheeled robots using Easy Java Simulations. Proceedings of the 7th IFAC Symposium on Advances in Control Education (ACE 2006), Madrid, Spain.

[21] Visioli A, Pasini F. A virtual laboratory for the learning of process controllers design. 7th IFAC Symposium on Advances in Control Education (ACE 2006), Madrid, Spain.

[22] Gillet D, Fakas G, Rekik Y, Zeramdini K, Geoffroy F, Ursulet S.The Cockpit: An Effective Metaphor for Remote Experimentation in Engineering Education. International Journal of Engineering Education 2003; 19(3): 389-397.

[23] Emersion environment by EPFL home page. Available at http://emersion.epfl.ch

[24] Gillet D, Nguyen Ngoc AV, Rekik Y. Collaborative Web-Based Experimentation in Flexible Engineering Education. IEEE Transaction on Education 2005; 48(4).

[25] Fakas GJ, Nguyen AV, Gillet D. The Electronic Laboratory Journal: A Collaborative and Cooperative Learning Environment for Web-Based Experimentation. Computer Supported Cooperative Work 2005; 14:189-216.

[26] SCHORDERET Technics home page. Available at http://www.schorderet-technics.ch

[27] Quanser Consulting Inc home page. Available at http://www.quanser.com

[28] GmbH home page. Available at http://www.amira.de

Biographies

H. Vargas received his degree in electronic engineering from Chile de la Frontera de Temuco University in 2001. He is working at UNED Department of Computer Sciences and Automatic Control since 2005. His current research interests are related with the design of new systems for control education, virtual labs and the use of internet in education.

J. Sánchez received his computer sciences degree from Madrid Polytechnic University in 1994 and his PhD in sciences from UNED (Universidad Nacional de Educación a Distancia) in 2001. Since 1993 he has been working at UNED Department of Computer Sciences and Automatic Control as an assistant professor. His current research interests are the design of new systems for control education, virtual labs, telepresence, and the use of internet in education.

F. Esquembre received the PhD degree in mathematics in June 1991 from the University of Murcia, Spain, where he has worked since 1986, holding a permanent job as assistant professor since 1994. His academic expertise includes differential equations, dynamical system, and numerical analysis. He currently teaches at the University of Murcia, and his research includes computer assisted teaching and learning as well as simulation of scientific processes for didactical purposes.

N. Duro received her physics degree from Madrid Complutense University in 1995 and her PhD in sciences from UNED (Universidad Nacional de Educación a Distancia) in 2002. Since 1997 she has been working at UNED Department of Computer Sciences and Automatic Control as an assistant professor. Her current research interests are control process, modelling and simulation of continuous processes and the design of new systems for control education.

R. Dormido received her physics degree from Madrid Complutense University in 1995 and her PhD in sciences from UNED (Universidad Nacional de Educación a Distancia) in 2001. Since 1995 she has been working at UNED Department of Computer Sciences and Automatic Control as an assistant professor. Her current research interests are the robust control, the modelling and simulation of continuous processes, and the design of systems for control education.

S. Dormido-Canto received his MS degree in electronic engineering from Madrid Pontificia de Comillas University in 1994 and his PhD in sciences from UNED (Universidad Nacional de Educación a Distancia) in 2001. He joined at UNED Department of Computer Sciences and Automatic Control as an assistant professor in 1994. His current research interests are related with the analysis and design of control systems via intranet or internet, high performance interconnection networks for cluster of workstations and optimal control.

G. Farias received his computer sciences degree from Chile de la Frontera de Temuco University in 2001. He is working at UNED Department of Computer Sciences and Automatic Control since 2005. His current research interests include simulation and control of dynamic systems as well as virtual and remote labs.

D. Gillet received the Diploma in Electrical Engineering from the Swiss Federal Institute of Technology in Lausanne (EPFL) in 1988, and the Ph.D. degree in Information Systems also from the EPFL in 1995. During 1992 he was appointed as Research Fellow at the Information Systems Laboratory of Stanford University in the United States. He is currently Maître d'enseignement et de recherche (Associate Professor) at the Systems Engineering Institute at EPFL, where he leads a multi-disciplinary research group. His current research interests include new learning technologies, sustainable interaction and collaboration systems, real-time Internet services, as well as nanopositioning systems. Dr. Gillet is Associate Editor of the International Journal of Technology Enhanced Learning. During the academic year 2005-2006 he was on sabbatical leave as Visiting Scholar at the Faculty of Engineering, Chinese University of Hong Kong.

S. Dormido holds a degree in physics from the Complutense University in Madrid in 1968 and a PhD from University of the Basque Country (1971). In 1981, he was appointed professor of control engineering at the Universidad Nacional de Educación a Distancia. His scientific activities include computer control of industrial processes, model-based predictive control, robust control, and model and simulation of continuous processes. He has authored and co-authored mode than 150 technical papers in international journals and conferences. Since 2002, he has been president of the Spanish Association of Automatic Control CEA-IFAC, where he promotes academic and industrial relations.

Figure captions

Fig. 1. Feedback control scheme.

Fig. 2. General architecture to distributed control applications using LabVIEW and *Ejs*.

Fig. 3. Loops running concurrently in the LabVIEW server-side application.

Fig. 4. Structure of a client-side application created in Easy Java Simulations.

Fig. 5. Components of the e-learning layer for remote experimentation.

Fig. 6. Integration of *Ejs* applications in the eMersion environment.

Fig. 7. Model of the electrical drive servo motor by Schorderet Technics.

Fig. 8. PID controller structure. (a) Continuous PID controller. (b) Discrete PID

controller programmed with *LabVIEW*.

Fig. 9. LabVIEW data flow in the server-side for the *eELab* remote experiment.

Fig. 10. The eELab integrated in the eMersion environment.

Fig.11. The heat-flow setup by Quanser.

Fig. 12. Graphical user interface developed with the 3D elements of the *Ejs* graphical library. The colour of the setup changes according to the temperature of the inner air. It can be considered as a kind of augmented reality technique to reinforce to the user the feeling of physical presence.

Fig. 13. Experimentation console when the real didactical setup is used for experimentation instead of the model.

Fig. 14. View and structure of the three-tank system.

Fig. 15. Two views of the remote lab for level control of a three-tank system.

Fig. 16. Control strategy for the three-tank system.

Tables

Table 1: Hardware and software components to develop the *eELab* server side