Comprehensive Collaborative Web-based Experimentation: Integration of Teleoperation and Simulation for Supporting Active Learning in Higher Education

Francisco Esquembre [a,*], Christophe Salzmann [b], Denis Gillet [b], Yassin Rekik [b], & Sebastián Dormido [c]

[a] *Departamento de Matemáticas, Universidad de Murcia, Spain*
[b] *School of Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland*
[c] *Dep. Informática y Automática, Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain*

Abstract

Virtual and real experimentation play an important role in natural sciences and engineering education. While many simulation packages and laboratory resources are available online, their effective integration into learning environments remains problematic. First, the development of such complex interactive resources is time consuming. Second, their reusability is limited by the development overhead necessary to adapt them to different educational needs. Finally, interoperability with other resources cannot easily be implemented. This paper presents the approach chosen to integrate both simulation and teleoperation Java applets within the *eMersion* environment dedicated to collaborative Web-based experimentation. The simulation applets were developed using the *Easy Java Simulations* authoring tool. The teleoperation applets provided remote connection to real laboratory resources. These resources complement each other from a learning point of view. The proposed approach can help educators facing the challenge of integrating Web services or components provided by various academic institutions into comprehensive learning frameworks.

Keywords: Authoring, Active Learning, Collaborative Learning, Remote Laboratory, Simulation

1. Introduction

1.1. Web-based experimentation

Experimentation is a core concept in natural sciences and engineering. Experimentation is carried out for testing (analysis), modeling, or prototyping (synthesis or design) purposes. Modeling relies simultaneously on data collection in laboratory, on parameter estimation and on data or model validation by simulation. Testing and prototyping can be carried out either in a laboratory or by simulation depending on the objectives and the availability of models. The acquisition and the reinforcement of both laboratory and simulation competences is

[*] Corresponding author. Tel. + 34 968 363534; fax: +34 968 364182.
E-mail address: fem@um.es

hence a key aspect generally enforced in higher education curricula. Thus, to successfully enable distance or flexible learning in natural sciences or engineering education, or simply to facilitate live demonstration in classrooms, remote access to laboratory and simulation resources should be provided in an efficient and integrated way. The educational usage of such resources via the Internet is defined as Web-based experimentation. For the sake of simplicity, but without any loose of generality, only engineering education is considered in the rest of this paper.

The pioneer environments supporting Web-based experimentation in engineering education appeared about 10 years ago [Henry 96]. Over the years, they became more functionally comprehensive and educationally effective [Cooper 05]. Today the most advanced solutions combine features such as simulation, teleoperation, annotation, collaboration and/or tutoring support [Schmid 99, Ko 01, Böhne 04, Fakas 05]. While being already quite comprehensive, most of these environments integrate only components developed or available within a given institution, as the integration of resources provided by third parties is a very demanding task.

1.2. Difficulties in authoring and deploying Web-based experimentation resources

Educators are generally reluctant to extensively deploy and propose Web-based experimentation resources, as they have still to rely on experts for their development. The consequence of this subcontracting is typically a weak control of the functionalities, quite often a partial matching with their educational needs, and moreover a non appropriation of the resources leading to inadequate support and maintenance.

To overcome these difficulties, authoring solutions enabling the educators, and even the students, to developing Web-based experimentation resources themselves have been proposed. They can be classified in two groups. The first group of solutions relies on professional packages the educators already use for their professional activities, such as Matlab for simulation or LabVIEW for instrumentation. Their Internet functionalities, the Matlab Web Server [Valera 05] and the LabVIEW Remote Panels solution [Watson 04] respectively, ease in some ways deployment for educational purposes, but lake convenient features for a real integration in comprehensive learning environments. In addition, they are proprietary and have associated license costs. The second group of solutions relies on domain-oriented frameworks build upon standard Web development packages [Rojiani 00, Sage 03, Cox 03]. The Easy Java Simulations authoring tool (EJS) detailed in this paper belongs to this second group.

One should also add that typical Web-based experimentation environments differ intrinsically from common Learning Management Systems (LMS), as the resources they provide are not learning content as such, but either spaces and tools for hands-on knowledge discovery, know-how acquisition, and reinforcement. As a consequence, Web-based experimentation environments should be service-oriented and explicitly designed with the following features:

- Simulation and/or teleoperation services, including those required to accessing the experimentation resources and to manipulating and operating them. Experimentation resources can be real physical devices, virtual devices, and/or simulation tools.
- Collaboration services, including synchronous solutions such as chat, video-conferencing and screen sharing, as well as asynchronous solutions such as Wiki, shared calendars, common spaces for resources exchange and support for collaborative editing.
- Operational support, such as data and activities planning and logging, as well as continuity of interaction and awareness features to enable respectively the retrieval of the experimental context between different learning modalities and the evaluation of the progress within the learning community.

For effective Web-based experimentation, these features should not only be available, they should be tightly integrated and intuitively usable.

The eMersion environment detailed in this paper is a Web-based experimentation environment providing most of the above features, except the synchronous collaboration ones which are envisioned for future versions.

1.3. Inter-institutional Integration

Providing a comprehensive Web-based experimentation solution, including authoring and deploying simulation and teleoperation resources, cannot be handled easily by a single academic institution, as well as by a commercial company [Weblab 06] due to the complexity and the variety of the resources to integrate. To provide students with advanced experimentation resources despite this fact, the École Polytechnique Fédérale de Lausanne (EPFL) and the University of Murcia have combined their effort and adapted their own solutions, respectively the eMersion environment and the Easy Java Simulations authoring tool, to come up with a common integrated solution. This initiative has been established in the framework of the ProLEARN European Network of Excellence (http://www.prolearn-project.org).

This paper describes the results of this collaboration as follows. Section 2 presents the eMersion environment and its principal services from an authoring point of view, including a short summary of the requirements for deploying Web-based experimentation resources, an overview of the services provided by eMersion, as well as its core *eJournal* component. Section 3 describes EJS, a general description of its objectives, the authoring procedure to create both the model and the view (the graphical interface) of a simulation, and the procedure to deploy the simulations built with it. Section 4 describes the integration work performed jointly by EPFL and University of Murcia to provide a comprehensive environment for Web-based experimentation. The first part of this Section introduces the *eJournal* API provided to integrate and communicate with new services and tools. The second part describes how EJS was adapted to create simulations that are compatible with this API. Finally, the third part presents the resulting architecture and Web-based experimentation of both a real and a virtual inverted pendulum as a practical example. The paper ends with concluding remarks and perspectives.

2. *eMersion* Web-based Experimentation Environment

2.1. Requirements for the deployment of Web-based experimentation solutions

In addition to the services dedicated to students' hands-on learning activities listed in Section 1.2, additional features dedicated to educators and tutors are necessary to support Web-based experimentation, including:

- Tutoring and supervision services, i.e. all services that help tutors to supervise and control the students' work. This includes, for example, the visualization of students' work progress, the evaluation results or examination support.
- Authoring and deployment services, i.e. all the services that educators needs to, first, define and manage hands-on activity scenarios (authoring), and second, deploy the environment for the end users (deployment). Authoring tools allow educators to define links to theory and lecture notes, tasks and deliverables, deadlines, and all support resources. Deployment tools focus on rendering, configuration, and personalization of the resources. These services also support the integration of external components through APIs.
- Management services: this category involves all the classical management services such as resources access management, users and groups management or documents managements.

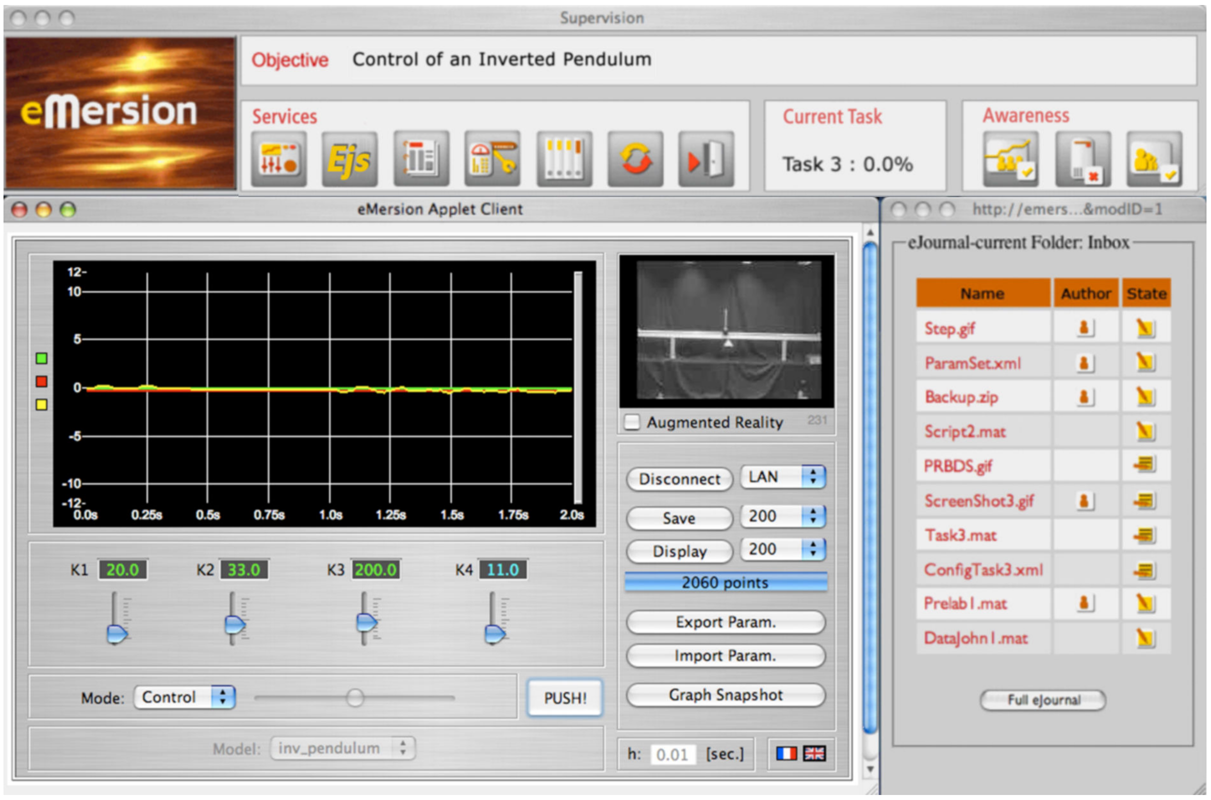
These services are not only important for Web-based experimentation; they are also required services for most e-Learning applications. The challenge is, once again, to couple them with the simulation and/or teleoperation services. The *eMersion* environment described below is the answer to this challenge developed at the EPFL.

2.2. The eMersion environment and its services

The *eMersion* environment (http://eMersion.epfl.ch) is a learning environment dedicated to Web-based experimentation. It is developed and deployed at the EPFL since 2000 to support interactive simulation and hands-on laboratory activities in engineering education. The current version is also used as a technical framework by partners of the ProLEARN network of excellence for integrating online experiments in e-Learning and e-Work contexts. The *eMersion* environment has been developed to answer the major requirements for the deployment of online experiments mentioned in the previous section. The features and the usage of *eMersion* from a student point of view have been described in various papers [Gillet *et al.* 03a, Gillet *et al.* 05]. After a short summary of these aspects to provide the reader with the whole vision, the authoring features of *eMersion* are described.

As a Web-based experimentation environment (Fig. 1), *eMersion* provides the students with the necessary components to successfully complete group assignments typically referred as hands-on learning modules. The main components of the user interface, called the cockpit, are the supervision panel (top part in Fig. 1), the experimentation protocol, the experimentation console (left part in Fig. 1), the analysis toolkit and the *eJournal* (right part in Fig. 1). The experimentation protocol describes the successive stages to carry out and the deliverables to provide for a given module. The experimentation console enables trial-and-error interactive learning by remote control of real laboratory resources. The analysis toolkit allows the processing of experimental data. Finally, the *eJournal* serves as an experimental data repository and as a share and collaboration space used by the members of a group in charge of completing the various tasks associated with a given module.

It is important to underline that the *eMersion* environment can be used with a simple Web browser either for distance experimentation carried out from home or for local experimentation carried out on campus. In the flexible learning context considered at the EPFL [Gillet 03b], the students can freely combine distance and local experimentation activities to complete an assignment. In this framework, the *eJournal* functionalities happened to be the elements that bring the necessary added value to support the real appropriation and effective use of the *eMersion* environment by the students [Fakas et al. 05].



Objective Control of an Inverted Pendulum

Current Task Task 3 : 0.0%

K1 20.0 K2 33.0 K3 200.0 K4 11.0

Mode: Control

Model: inv_pendulum

LAN

Save 200

Display 200

2060 points

h: 0.01 [sec.]

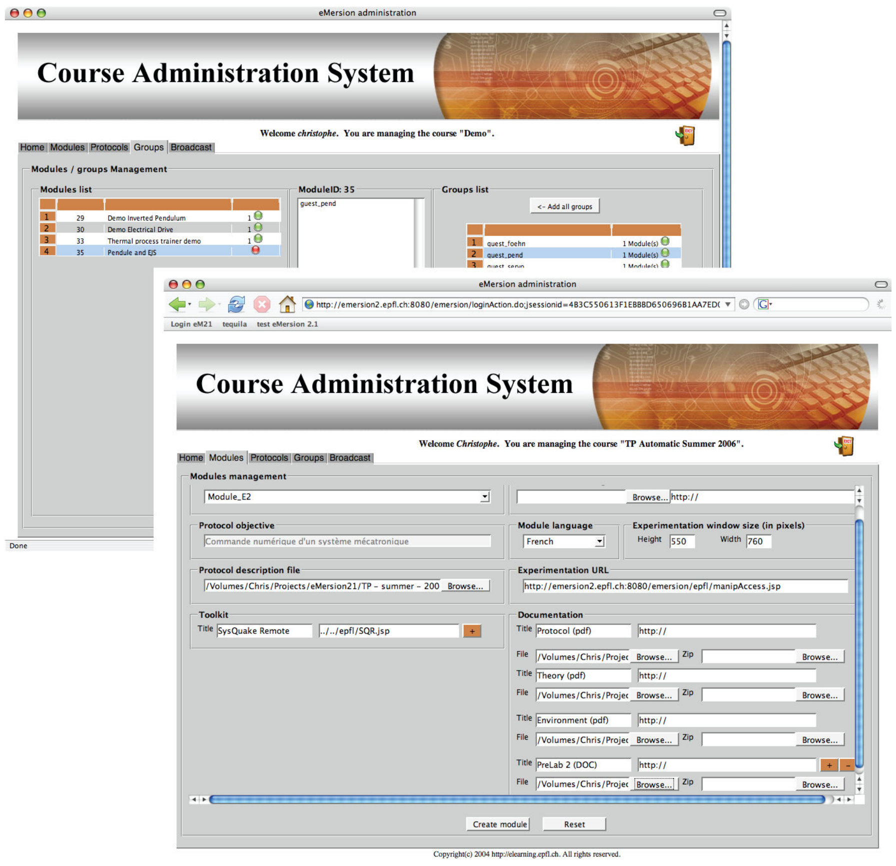
eJournal-current Folder: Inbox

Name	Author	State
Step.gif	●	▲
ParamSet.xml	●	▲
Backup.zip	●	▲
Script2.mat		▲
PRBDS.gif		▤
ScreenShot3.gif	●	▤
Task3.mat		▤
ConfigTask3.xml		▤
Prelab1.mat	●	▲
DataJohn1.mat		▲

Fig. 1. User interface of the *eMersion* environment.

2.3. Authoring support provided by eMersion

In order to support educators in deploying experimentation resources (simulation or teleoperation components), the *eMersion* environment provides them with simple Web-based authoring tools (Fig. 2). The authoring process leading to the deployment of a new cockpit can be divided into four main steps: defining the protocol, defining the resources that will be included in the cockpit, generating the cockpit, and publishing the cockpit to the end users (students).



Course Administration System

Welcome *christophe*. You are managing the course "Demo".

Modules / groups Management

Modules list

1	29	Demo Inverted Pendulum	1 ●
2	30	Demo Electrical Drive	1 ●
3	33	Thermal process trainer demo	1 ●
4	35	Pendule and EJS	●

ModuleID: 35

guest_pend

Groups list

1	guest_foehn	1 Module(s) ●
2	guest_pend	1 Module(s) ●
3	guest_servo	1 Module(s) ●

http://emersion2.epfl.ch:8080/emersion/loginAction.do;jsessionid=4B3C550613F1EBBBD650696B1AA7ED(

Course Administration System

Welcome *Christophe*. You are managing the course "TP Automatic Summer 2006".

Modules management

Module_E2

http://

Protocol objective: Commande numérique d'un système mécatronique

Module language: French

Experimentation window size (in pixels): Height 550 Width 760

Protocol description file: /Volumes/Chris/Projects/eMersion21/TP – summer – 200

Experimentation URL: http://emersion2.epfl.ch:8080/emersion/epfl/manipAccess.jsp

Toolkit

Title SysQuake Remote ../../epfl/SQR.jsp

Documentation

Title Protocol (pdf) http://
File /Volumes/Chris/Projec Zip

Title Theory (pdf) http://
File /Volumes/Chris/Projec Zip

Title Environment (pdf) http://
File /Volumes/Chris/Projec Zip

Title PreLab 2 (DOC) http://
File /Volumes/Chris/Projec Zip

Copyright(c) 2004 http://elearning.epfl.ch. All rights reserved.

Fig. 2. Course management tool and the Module authoring tool.

2.4. *The eJournal and its fragments*

The *eJournal* has been designed as an extended electronic version of the traditional laboratory journal. The *eJournal* plays an important role at both the social level (collaboration among users) and the system one (exchange between components).

The social level features of the *eJournal* and their assessments have been presented in Nguyen *et al.* 05; they are just listed here for the sake of clarity. The *eJournal* is a collaboration space with a user interface similar to an email client shared by a group of students. The stored 'messages', called fragments, can be experimental data, configuration parameters, graphs, documents, etc. All fragments are typed and categorized based on their sources, their content, and possibly the associated task from the protocol. The *eJournal* provides various services to manage the fragments. By tracking the fragments creation history, progress awareness about the group and the class progresses are also generated in real-time and displayed in the cockpit.

The system level features of the *eJournal* enable the import and export of fragments between different Web components. This pivotal role of the *eJournal* allows the *eMersion* environment to integrate new services and components without major development and adaptation. This concept of exchange between Web components is detailed in section 4.

3. Easy Java Simulations Authoring Tool

3.1. Graphical authoring tool for Java-based simulation

The Easy Java Simulations authoring tool (EJS) is a free, open-source solution designed to help educators and scientists create discrete computer simulations in Java. A discrete computer simulation, or simply a computer simulation, is a program that tries to reproduce a natural phenomenon through the visualization of the different states that it can have. Each of these states is described by a set of variables that change in time due to the iteration of a given algorithm.

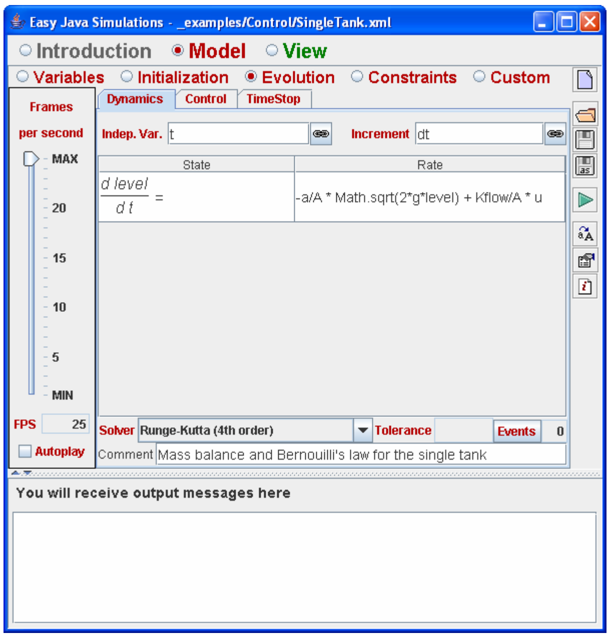
EJS provides specialized interfaces and utilities that allow educators to act as a high-level provider of the information required to create the simulation. The educators can concentrate on both the scientific part of the simulation (the model) and on its pedagogical one (how to visualize phenomena and what interaction capabilities to offer to students). To specify the model of the simulation, the educators fill a simple, yet powerful, predefined scheme provided by the tool. To design the view for the simulation, they use a set of ready-to-use graphical elements suited both for the visualization of scientific data and processes, and for the students' interaction with the model.

The technical aspects of the authoring process are all handled by the authoring tool, from generating the source code to compiling and running the simulation. EJS generates an independent, high performance, Internet-aware, final product in the form of a Java application or applet, complemented with embedding HTML pages [Esquembre 04]. The choice of Java as development language is justified in terms of its wide acceptance by the Internet community, and of its availability on most platforms.

3.2.Creation of models using EJS

EJS structures the model into four main blocks: the definition of variables, the initialization, the evolution, and the definition of constraints. The tool provides a dedicated panel for each of these blocks. For variables, it provides a simple table where the user specifies names, types, and initial values for all the variables that describe the system under study. If the initialization of the model requires some additional computations based on the given values, the tool provides an editor where the author writes the Java code that performs these computations. The gain of using EJS is that only the algorithms for the computations have to be coded. EJS wraps this code into complete Java methods and takes care of calling them at the appropriate stage at run time. This considerably lowers the required technical skills needed to create a Java program.

Similarly, the tool offers dedicated editors for the creation of the evolution, that is, the code that updates the value of the variables of the model as time passes. Here, the author is provided with two types of editors, one for plain Java code, and a second one specialized in systems of ordinary differential equations (Fig. 3). This sophisticated editor helps entering the differential equations in a way much similar to how teachers write them in the classroom. It then automatically generates the Java code that solves the equations using most popular numerical integration algorithms. The editor also supports advanced features such as derivatives of vectors and event handling.



Indep. Var. t

Increment dt

$\frac{d\ level}{d\ t}$ = -a/A * Math.sqrt(2*g*level) + Kflow/A * u

Solver Runge-Kutta (4th order)

Tolerance

Comment Mass balance and Bernouilli's law for the single tank

You will receive output messages here

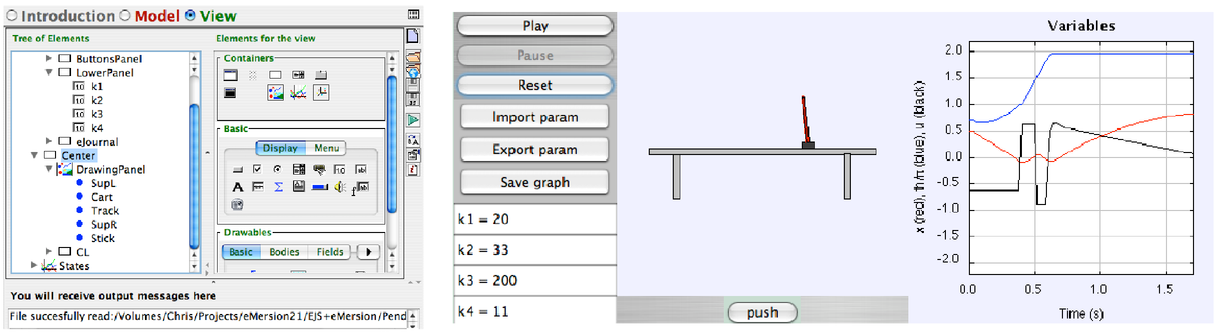
Fig. 3. User interface of Easy Java Simulations. The editor for differential equations is displayed.

Finally, a panel to define constraints is included. It allows the user to specify fixed relations between variables that always hold. Constraints help the computer to keep the correct values of interdependent variables whenever the user interacts with the simulation to change any individual variable.

This set of four panels provides a simple structure that both novices and experts can easily use to specify their models. The range of models that can be created with the help of the tools provided is very wide. However, for the cases in which more advanced programming is required, an extra panel allows the author to create more sophisticated code, including custom Java methods and access to user-provided external libraries. Details can be found in the EJS user's manual [Esquembre 05].

3.3.Building graphical user interfaces

Building the graphical interface for the simulation is probably the most computer-specific part of creating a simulation. Modern simulations, especially those required for teaching, demand advanced graphic visualizations of the underlying scientific processes and a high level of interactivity. The teacher is indeed interested in making a good pedagogical design of the interface, but not in the low-level knowledge of the standard Java libraries required to actually build it. For this reason, EJS offers a set of built-in graphical elements (based on the Open Source Physics library [Christian 06]) which are presented in a click-and-use way. With a few mouse clicks, the author can easily create a tree-like structure of graphical elements that will make the view. Figure 4 shows a typical view.



You will receive output messages here

File succesfully read:/Volumes/Chris/Projects/eMersion21/EJS+eMersion/Pend

Fig. 4. Panel of EJS for the creation of the view (left) and a typical 2D animation for the simulation of an inverted pendulum. Most elements in the view are interactive.

All view elements have internal fields called properties that the author can customize to make the element look and behave in a particular way. These properties can be given constant values (such as a fixed color, or position), but can also be linked to model variables. This linking, which is done in a most natural way, is what makes the simulation really dynamic and interactive. Any change in the value of a variable of the model is then automatically transmitted to the view, which updates its aspect or behavior accordingly. Reciprocally, if the user interacts with the simulation's view, the change is automatically reported to the model, modifying the variables linked to the particular view element affected by the interaction. Finally, so-called action properties allow view elements to execute pieces of Java code. This increases the interaction capabilities of the simulation by providing a way for the student to control it.

3.4 Deploying the simulation applets

Once the educator has provided this information, the description of the model and the design of the view, EJS takes care of the remaining operations. It generates all the required source code, compiles the simulation, compresses it into a JAR file, and runs it. In particular, the tool takes care of implementing all sophisticated technicalities (such as multithreading, to name one). It also creates a set of HTML pages that can be used to quickly publish the simulation on a Web server. If the teacher wants to add more descriptive narrative, EJS provides a simple editor that allows adding extra HTML pages.

The result is a simulation that can be freely distributed, independently of the authoring tool used to create it. Educators only need to copy the generated files, together with a library directory, to a Web server or CDROM. The whole process is quite effective, both in terms of the quality of the resulting simulation and of time and effort required to build it. The author can concentrate practically all of her time in scientific and pedagogical issues and not in computer specific (less interesting) tasks.

4. Integration of Easy Java Simulations and the *eMersion* Environment

4.1. API for communication with eJournal

In order to facilitate the integration of external components and services into the *eMersion* environment and as mentioned in Section 2.3, the *eJournal* has been designed as a pivotal component, which can communicate with other components thanks to a data exchange mechanism. This mechanism is based on a data homogenization and transformation process.

The *eJournal* supplies a set of interfaces to handle the requests generated by the different components integrated into the Web-based experimentation environment. Through these interfaces (classes), the *eJournal* is able to directly process java objects.

The communication channel between the *eJournal* and a Web component (applet) is established either directly or via a helper application to overcome the Java applet security limitations. A servlet is continuously listening for *eJournal* remote access. When receiving a request in the form of a java object, the *eJournal* extracts the type of data received, determines how to transform the request data and which target component should be invoked. For example, the remote experimentation applet (Fig. 1) generates a fragment containing the last 2048 measured points as a binary file. The created fragment will be automatically directed by the *eJournal* to the analysis tool, namely Sysquake Remote, for display and further data processing. Sysquake Remote from Calerga (www.calerga.com) is an Apache module that provides access to a powerful mathematical engine via a Web interface. Table 1 summarizes the fragment types, the associated classes and they respective target components.

This mechanism facilitates significantly the interaction process. Data are passed smoothly and transparently from one component to another. To reduce the amount of transmitted information, some fragments are automatically compressed internally. The exchange can be endogenous, i.e. initiated by a component; or exogenous, i.e. initiated by a user (typically by a single mouse click). The need to use external applications for passing data between components is reduced. For example, a student does not need to save measurements data on her local hard disk and then sent them by email as an attached file to share them with peers.

Fragment type	Associated class	Target component	Comment
binary	*class DatasFile*	.mat -> Sysquake Remote others -> file	also used for external fragment
graphic	*class GraphicsFile*	browser or file	supported formats are GIF and JPEG
text	*class TextFile*	browser or file	plain text or XML text
parameter (for applet)	*class ParamsFileXML*	applet or browser	specific to the remote experimentation applet. Parameters are XML encoded
Sysquake	*class DatasFile*	Sysquake Remote	specific case of binary fragments with a .mat extension
external	*class DatasFile*	file	files are save in native binary format
request	*class Request*	request made by applets	supported requests are: *list, taskNames, fragment, parameter set, snapshot,binary, text, sysquake, external, all*

Table 1. Fragment types, the associated classes and their respective target components

Every request to the *eJournal* is acknowledged with a reply, or an exception in case of error. The reply can be a requested object (fragment/list of fragments) or an informational message.

4.2. Extension of EJS to communicate with the eJournal

To enable the integration of Easy Java Simulations within the *eMersion* environment, EJS has been endowed with new Java methods to communicate with the *eJournal*. EJS can also now generate a HTML page with the URL of the *eMersion* server that will serve the simulation. The simulation applet generated automatically recognizes at run-time that it is being run from an *eMersion* server and includes the needed communication library to interact with the *eJournal*.

The new EJS methods include taking a snapshot of any of the simulation windows, saving and loading all or part of the simulation variables, and producing text or XML reports. When the simulation runs as an application, this input and output routines use the hard disk as support media. However, when running as part of an *eMersion* cockpit, all the input and output procedures are automatically redirected to the *eJournal*.

Because the educators have designed the simulation themselves or can easily modify it, thanks to the open nature of EJS, they can include buttons or other controls in the view that will create the *eMersion* fragments of interest for each particular pedagogical situation.

4.3. A practical example

At EPFL, students enrolled in the automatic control course have to take measurements on a laboratory-scale inverted pendulum, estimate model parameters using these measurements, then design a controller, test it by simulation and finally implement the final solution on the real system. Hence, there are successive comings and goings between simulation and implementation, or between virtual experimentation and real (remote) experimentation. As a consequence, tight integration between the simulation and the implementation tools is necessary. Such a methodology is common in control implementation and has been enforced in education recently [Dormido 04]. The components proposed to the students to carry out the described assignment now integrate an EJS simulation applet and a real teleoperation one. The controller parameters have to be shared through the *eJournal* to facilitate the comparison between the simulation and the implementation results.

The simulation illustrated in Figure 4 works autonomously but can also communicate with the *eJournal* with the help of two buttons. The Export param button save the controller parameters to the *eJournal* by dumping the controller gains to a file that is included as a new *eJournal* fragment. The button action, triggered by a mouse click, consists of a single line of code that calls one of the provided EJS methods. In the same way, Import param button imports the parameters from the *eJournal*. With these buttons the user can test a set of parameters in the simulation using the EJS applet, export these parameters to the *eJournal,* and import them within the remote experimentation applet, to test them on the real setup. These operations are performed with just a few mouse clicks.

To complete the integration, EJS options have to be edited to provide the URL of the directory that will contain this HTML file in the *eMersion* server. Then, EJS can generate the additional HTML file required.

To deploy this simulation through an *eMersion* cockpit, the author just needs to copy the generated files (and the EJS library) to the server, and specify the HTML file created by EJS as the URL for the experimentation resource.

5. Concluding remarks

Web-based experimentation differs from more traditional e-Learning solutions in the sense that it is more service- than content-oriented. What the students do with the resources is more important than the resources themselves.

The key services sustaining Web-based experimentation include remote accesses to real laboratory resources, their simulations, analysis tools and share spaces. These services have to be fully integrated and be able to exchange data in real-time to effectively support typical collaborative hands-on learning activities.

This paper shows that the share space existing in the *eMersion* environment, namely the *eJournal* (and its APIs), is the pivotal component for enabling seamless services integration.

The integration of heterogeneous components has been illustrated with the example of an EJS applet enabling the simulation of the real inverted pendulum within *eMersion*.

In fact, the *eMersion* environment and its *eJournal* can be seen as a container and a mechanism for educators to combine and to enable communication between heterogeneous and distributed laboratory resources, Web services or Web components. As nowadays a large academic community is making real and virtual laboratory resources available online, the proposed integration scheme makes easier to share and to reuse solutions developed in different institutions, reducing in that way the barrier to broader e-Learning deployment.

The proposed solution is an attempt towards stateful Web services integration, which can be handled by educators without the intervention of IT specialists. Standalone e-Learning solutions can easily become reusable components just by adding a proper interface to the *eJournal* that eventually acts simultaneously as an exchange platform between users and between components. The easiness of use of EJS facilitates non-programming teachers the creation of their own simulations.

Next generations of Grid middleware should also provide similar features in a near future. They should also ease the use of more effective real-time communication protocols, orchestration and authentication schemes. Their standardization and deployment unfortunately still rely on the intervention of IT specialists [Saliah-Hassane 05]. The *eMersion* environment will be upgraded continuously to integrate and make available to the academic community the latest technologies in Web, Grid and mobile services.

Acknowledgements

The elements presented in this paper result from various e-Learning projects and activities carried out with the support of the Board of the Swiss Federal Institutes of Technology and of the European Union in the fifth and sixth framework programs. The authors thank Anh Vu Nguyen Noc who contributes to the design, implementation and validation of the *eJournal* as part of his PhD thesis.

References

Böhne, A., Faltin, N., & Wagner, B. (2004). Synchronous Tele-Tutorial Support in a Remote Laboratory for Process Control. INNOVATIONS 2004: World Innovations in Engineering Education and Research, edited by Win Aung (et al.), pp. 317-329.

Christian, W. (2006). Open Source Physics: A User's Guide with Examples. Benjamin Cummings Publisher, ISBN: 0-8053-7759-X.

Cooper, M. (2005). Remote laboratories in teaching and learning – issues impinging on widespread adoption in science and engineering education. *International Journal of Online Engineering (http://www.ijoe.org/ojs/)*, *1*(1), pp. 1-7.

Cox A. J., Belloni M., Christian W., and Dancy M. H. (2003), Teaching Thermodynamics with Physlets® in Introductory Physics, *Physics Education* 38, 433.

Dormido, S. (2004). Control Learning: Present and Future. Annual Reviews in Control 28, 115-136.

Esquembre, F. (2004). Easy Java Simulations: a software tool to create scientific simulations in Java. Comp. Phys. Comm. 156, 199-204.

Esquembre, F. (2005). Easy Java Simulation user's manual. http://fem.um.es/Ejs/.

Fakas, G. J., Nguyen, A. V., & Gillet, D. (2005). The Electronic Laboratory Journal: A Collaborative and Cooperative Learning Environment for Web-Based Experimentation, *Computer Supported Cooperative Work*, *14*, 189-216.

Gillet, D., Nguyen, A. V., & Rekik, Y. (2005). Collaborative Web-based Experimentation in Flexible Engineering Education, *IEEE Transactions on Education*, Special Issue on Web-based Instruction, Vol. 48, No. 4, pp. 696-704.

Gillet, D., Geoffroy, F., Zeramdini, K., Nguyen, A. V., Rekik, Y., & Piguet, Y. (2003a). The Cockpit: An Effective Metaphor for Web-based Experimentation in Engineering Education, *International Journal of Engineering Education,* "Special Issue on Distance Controlled Laboratories and Learning Systems", *19*(3), 389-397.

Gillet, D. (2003b), Towards Flexible Learning in Engineering Education, in the book "Innovations - 2003: World Innovations in Engineering Education and Research", pp. 95-102, Published by iNEER in Cooperation with Begell House Publishers.

Henry, J. (1996). Controls Laboratory Teaching via the World Wide Web. *Proceedings of the ASEE Annual Conference,* Washington, USA, 1996.

Ko, C. C., Chen, B. M., Chen, J., Zhuang, Y., & and Tan, K. C. (2001). Development of a Web-based laboratory for control experiments on a coupled tank apparatus, IEEE Transactions on Education, *44*(1), pp. 76-86.

Nguyen A.V., Rekik Y. A., & Gillet D., (2005). A framework for sustaining the continuity of interaction in Web-based learning environment for engineering education. In proceedings of the 17th ED-MEDIA conference, 2005, Montreal, Canada.

Rojiani, K. B., Y. Y. Kim, and R. K. Kapania (2000), *Web-Based Java Applets for Teaching Engineering Mechanics*, Proceedings, ASEE 2000 Annual Conf., Session 2620, June 18-21, 2000, St. Louis, Mo, USA.

Sage, D., Unser, M. (2003). Teaching Image-Processing Programming in Java. *IEEE Signal Processing Magazine*, *20*(6), pp. 43-52.

Saliah-Hassane, H., Benslimane, D., De La Teja, I., Fattouh, B., Do, L.K., Paquette, G., Saad, M., Villardier, L., and Yan, Y. (2005). A General Framework for Web Services and Grid-Based technologies for Online Laboratories. Proceedings of the 2nd iNEER Conference for Engineering Education and Research (ICEER), March 1-5, Tainan, Taiwan.

Schmid, Ch. (1999). A Remote Laboratory Using Virtual reality on the Web, *Simulation*, Special issue: Web-Based Simulation, *73*(1), pp. 13-21.

Valera, A., Diez, J. L., Vallés, M., & Albertos, P. (2005). Virtual and remote control laboratory development. *IEEE Control*

Systems Magazine, February, pp. 35-39.

Watson, J. L., Bibel, G., Ebeling, K., Erjavec, J., Salehfar, H., & Zahui, M. (2004). On-line Laboratories for Undergraduate Distance Engineering Students. *34th ASEE/IEEE Frontiers in Education Conference*, October 20 - 23, 2004, Savannah, GA.

Weblab (2006). http://www.controlab.com, visited February 2006.

Vitae

Francisco Esquembre received the Ph.D. degree in Mathematics in June 1991, from the University of Murcia, Spain, where he works since 1986, holding a permanent job as Assistant Professor since 1994. His academic expertise includes Differential Equations, Dynamical Systems and Numerical Analysis. Francisco is the author of Easy Java Simulations. He teaches currently at the University of Murcia and his research includes computer assisted teaching and learning as well as simulation of scientific processes for didactical purposes.

Christophe Salzmann is a Senior Research Associate at the École Polytechnique Fédérale de Lausanne (EPFL). He received his MS degree in computer Science from the University of Florida in 1999 and his PhD degree from the EPFL in 2005. His research interests include new Web technologies, real-time control, real-time interaction over the Internet with an emphasis on Quality of Service and bandwidth adaptation.

Denis Gillet is MER (Associate Professor) at the École Polytechnique Fédérale de Lausanne (EPFL). He received the Ph.D. degree in Control Systems in 1995 from EPFL. His research interests include optimal and hierarchical control systems, distributed e-learning systems, sustainable interaction systems, and real-time Internet services. Dr. Gillet received the 2001 iNEER (International Network for Engineering Education and Research) Recognition Award for Innovations and Accomplishments in Distance and Flexible Learning Methodologies for Engineering Education.

Yassin Rekik received the Ph.D. degree in computer science from the École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 2001. He is a Senior Research Associate at EPFL and Professor at the University of Applied Sciences, Neuchâtel, Switzerland. His research interests include Web-based learning, mobile learning, collaborative and group-oriented learning, and online experimentation and laboratory activities. Dr. Rekik is currently involved in several national and international initiatives and projects, in particular, the European Network of Excellence ProLEARN.

S. Dormido S. Dormido received his Physics degree from Madrid Complutense University (1968) and his Ph.D. from Country Vasc University (1971). In 1981, he was appointed Full Professor of Control Engineering at UNED Faculty of Sciences. He has supervised 25 PhD Thesis and co-authored more than 200 conference papers and 150 journal papers. Since 2002 is President of the Spanish Association of Automatic Control, CEA-IFAC. His scientific activity includes various topics from the control engineering field: computer control of industrial processes, model-based predictive control, robust control, modeling and simulation of hybrid systems and control education with special emphasis on remote and virtual labs.