Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 48-29 (2015) 140–145

A new generation of online laboratories for teaching automatic control [★]

J. Chacón [*] M. Guinaldo [*] J. Sánchez [*] S. Dormido [*]

[*] *Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain (e-mail: jchacon@bec.uned.es, {mguinaldo, jsanchez, sdormido}@dia.uned.es).*

Abstract: Traditionally, online control education labs are designed and implemented with the purpose of teaching on a specific topic: controller tuning, stability, etc. These tools usually have a monolithic design: there is a strong binding between the model and the control. Usually, there is a built-in controller that may have a set of modifiable parameters, but the controller itself cannot be changed. Also, it cannot be easily translated to other system, for example to compare the performance. In other words, the system subject of study is not very configurable, which unnecessarily constrains the possibilities of these tools. Though this approach may be adequate for some situations, it would be interesting to have more flexible and versatile online laboratories, for instance, in Master courses, where students are expected to design and test advanced control laws. This paper proposes a new paradigm to develop such tools. An element has been deployed and allows the dynamic execution of scripts written in different programming languages, such as Java, Javascript, or Matlab. The final aim is to provide students with a tool to promote an active participation in the practice, writing their own controller code and freely experimenting with the system. The paradigm is illustrated with some examples of these new generation online laboratories.

Keywords: Control education, online laboratory, web-based experimentation, educational aids, level control.

1. INTRODUCTION

The fact that digital media such as simulations, videos, or web-based labs can positively impact students knowledge has been accepted by the education community, and it is a widespread practice in distance education (Kozma, 1994). Web-based labs make possible to illustrate scientific phenomena that require costly or difficult-to-assemble equipment and should ideally consist of two different and complementary parts:

- *Virtual Labs* provide computer based simulations which offer similar views and ways of work to their traditional counterparts (Wannous and Nakano, 2010). Nowadays, simulations have evolved into interactive graphical user interfaces where students can manipulate the experiment parameters and explore its evolution.
- *Remote Labs* use real plants and physical devices which are teleoperated in real time (Guimaraes et al., 2011). Remote experimentation through the Internet has been available for more than a decade and its interest has never diminished over the years (Salzmann and Gillet, 2007).

The effort of developing virtual and remote laboratories (VRLs) from the scratch is huge, and in this regards, Easy Java/Javascript Simulations (EjsS) (Esquembre, 2004) helps to overcome this problem. EjsS is a free authoring tool written in Java that helps non-programmers create interactive simulations in Java or Javascript. The architecture of EjsS is based on the model-view-controller (MVC) paradigm, in which three parts can be differentiated: The model, the view, and the controller. The *model* describes the process under study in terms of variables (states of a plant, for example) and relationships between these variables, expressed by computer algorithms. The *view* provides the graphical represensentation of the system, that is, the graphical user interface (GUI) of the simulation. Finally, the *controller* responds to the user interactions and translates them into requests for the model.

When this scheme has been applied to automatic control labs, the two main elements of a control loop, i.e., the model of the plant and the controller, have been unavoidably coupled in the model part of the MVC paradigm. In this way, the algorithm that is executed in reality by the controller belongs to this set of generic relationships between variables described above, and there is no a neat separation between the plant and the controller, and moreover, this is not accessible to the user.

This monolithic structure of virtual labs causes that their reach is limited, since students can manipulate certain parameters of the controller but they cannot access to the core definition of it. Though the *traditional* VRLs have been working satisfactorily in our institution for over a

[★] This work has been funded by the National Plan Projects DPI2011-27818-C02-02 and DPI2012-31303 of the Spanish Ministry of Science and Innovation and FEDER funds.

decade (Sánchez et al., 2002; Dormido et al., 2008), they result limited when used by postgraduate students, which are expected to design and test advanced control laws. This need has impulsed a new generation of virtual labs that decouples the system and the controller, and has the following advantages: Increases the flexibility of these simulation tools, the student can actively participate in the development of the lab practice, it allows the reuse of controllers already implemented in different programming languages, or to test the same controller in different plants without any additional effort.

Thus, this paper presents the architecture as well as the technical details of this second generation of online laboratories. An element of EjsS, which is a sort of Java library provided by a friendly GUI for its configuration, has been developed. From a lower level viewpoint, it allows the dynamic execution of scripts written in different programming languages such as Java, Javascript, or Matlab. From the high level perspective, the controller structure (and not only the values of the parameters) can be change in execution time.

The structure of this paper is as follows: Section 2 gives an overview of the current state of the online laboratories. Section 3 proposes a new paradigm to develop more flexible and versatile online laboratories, and Section 4 covers the implementation of a virtual laboratory that follows the proposed paradigm. Finally, Section 5 presents the conclusions.

2. OVERVIEW

In the literature, different approaches oriented to developing remote laboratories can be found. In Stefanovic et al. (2011), authors present a remote laboratory exclusively created by using the LabVIEW platform (National Instruments, 2015). Although LabVIEW VIs[1] can be easily made ready for Internet delivery, a LabVIEW Runtime Engine must be installed on the client side. This last step is not recommended when creating remote labs since installing software plugins sometimes can become hard for final users. For this reason, LabVIEW platform is commonly used only for creating the server side of a remote lab (other software options for the server side can be Matlab (Farias et al., 2010), Simulink (Fabregas et al., 2011), C++ (Costa et al., 2010), Scicos (Magyar and Zakova, 2012), etc.).

On the other hand, Java applets and Flash applications have been the most popular web technologies for developing the client interface for remote labs. In Hernandez et al. (2008), a virtual laboratory for the analysis and study of the human respiratory system was created. In this example, an applet was developed by using EjsS. Two other interesting examples of remote labs for pedagogical purposes were presented in Sánchez et al. (2002). In these articles authors present a set of web-based laboratories for teaching automatic control concepts where Java applets to access remotely the training services were used as well. Similarly, Flash applications have found some applications in virtual and remote laboratories design Barrett et al. (2003); Goffart (2007). Unlike Java, Flash has been less used by developers for designing web-based labs mainly for license payment issues. Another example can be found in Besada-Portas et al. (2012), where a remote control laboratory is built using EjsS applets and twincat programmable logic controllers.

Virtual Labs (VLs) are nothing but computer applications. Thus, it is easy and cheap for their developers to share them with the community. This is the case of the ComPADRE-Open Source Physics Christian et al. (2011) repository[2], with hundreds and hundreds of Java and Javascript simulations, all of them created with EjsS, or the PhET Wieman et al. (2008) webpage[3], also with hundreds of simulations based on Java, Flash and HTML5. Remote Labs (RLs), however, require real equipment to run. This means that RLs are: 1) more expensive to create and maintain and 2) the maintenance to ensure their correct operation requires time and attention.

The architecture of previous remote labs in UNED were usually based on three software tools: EjsS, LabVIEW (Laboratory Virtual Instrument engineering Workbench) and the JIL server (Chacón et al., 2014; Chacón et al., 2014; Guinaldo et al., 2013; de la Torre et al., 2011). With this approach, an EjsS application implements the user interface in the client side, allowing students to interact with the plant, carry out experiments and obtain experimental data. In the server PC, there is a LabVIEW VI (Virtual Instrument) which is plant-dependent and implements a local control and some safety measures to avoid damaging the plant in case of accident or malicious users. This server PC needs to be running the JIL server, which offers an interface for the EjsS application in the client side to interact with the LabVIEW VI. This architecture has worked very well over the last years, and nowadays there are many RLs in use based on this solution. The main successful idea of *JIL Server* was the introduction of a middleware built over the local control software, that adds an interoperability layer, and reduces the coupling between the client and the server implementation.

But, apart from the software tools used to develop the laboratory, all the cited examples have a non-flexible design, in the sense that there is a reduced set of experiences that can be carried out with the system, and these experiments are pre-designed and incorporated in the lab. Focusing on control learning tools, a control system can be decomposed at least into two differentiated components: the *process* and the *controller*. The design of both components, process and controller, are determined at the building process of the remote or virtual laboratory. They are fixed and cannot be changed by the user, but there are a small set of parameters affecting the system behaviour which can be tuned to accomplish with some goals, such as control performance or any other requirement considered adequate by the educator. In other cases, the student can choose between different pre-implemented controllers, and compare the behaviour for different cases.

Anyway, the use of the tool is limited to the scope the designer has in mind at the deployment stage, and it is

[1] "LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters", *National Instruments* (*http://www.ni.com/white-paper/7001/en*)

[2] http://www.compadre.org/osp

[3] https://phet.colorado.edu/