MOOLs for MOOCs

A first edX scalable implementation

Christophe Salzmann, Denis Gillet
École Polytechnique Fédérale de Lausanne, Switzerland
{christophe.salzmann, denis.gillet} @epfl.ch

Yves Piguet
Calerga sarl, Switzerland
yves.piguet@calerga.com

***Abstract*— Deploying remote laboratories at a large scale is the next challenge in remote experimentation. It is referred as Massive Open Online Labs (MOOLs). Being able to sustain a massive number of users accessing a specific resource concurrently is a complex task. This paper presents technical solutions, currently deployed, to implement such a remote lab in a MOOC. The *control system lab* deployed in edX, where more that 200 students access concurrently a farm of 20 electrical drives, is used as an example. Gamification mechanisms which accommodate even more students are proposed.**

Keywords—Massive Open Online Labs (MOOLs), MOOCs, edX, smart device, LTI, Sysquake

I. INTRODUCTION

Remote experimentation finally took off through various projects and it is now current to see remote labs with a significant number of experimentations available at a distance. The model often chosen to enable access to these labs is *Laboratory as a Service* (LaaS) [1], where the lab is seen as a set of resources that the user can select on demand. The *smart device* model is similar to the LaaS model and tries to describe the lab equipment and its controlling computer as unique entity [2]. The smart device is seen through a set of services that the user can connect to [4]. These models are the corner stone for deploying massive open online labs (MOOLs) but, alone, they just provide a one-to-one (1:1) access: one user access one real equipment at a time. Various solutions are proposed to increase the numbers of concurrent users [4], [5] and a ratio of 5-10:1 is possible. The next step is to be able to handle the massive access, in the range of 50-100:1.

This paper first introduces the smart device model. Then, it proposes various modifications to existing remote lab in order to support the massive aspect of MOOCs. The MOOC and MOOL infrastructure are then described, especially how a smart device client application is integrated in a MOOC as a LTI module and how this application is able to interact with other applications or tools such as simulations. A learning scenario for a MOOC session using the above elements is then presented. Finally, a gamification process that permits to select users from a massive pool is suggested.

II. SMART DEVICE

This section summarizes the Smart Device paradigm that has been presented in [3]. The main differences between existing implementations and Smart Devices are first the full decoupling between the server and the client and, second, the server representation as a set of well-defined services and functionalities that enable interoperability and user-oriented personalization.

The decoupling removes the umbilical cord between the client and the server so that they can live their own separate lives. While in a traditional client-server architecture, the server and client share specifications that are often uniquely used by them, the Smart Device paradigm defines one common set of specifications that is shared by all Smart Devices. This reuse of a common set of specifications and the client-server decoupling alleviates most of the problems developers are facing when the client application needs to be adapted to new OS or platforms. It is also instrumental if the client application is to be integrated in other environments such as learning management systems (LMS), or simply if additional features are added to the server. Furthermore, interoperability with, and reuse of, existing applications and services become possible when labs share common specifications.

Smart Devices mainly provide (Web) services to access sensors and actuators. The Smart Device specifications [3] fully describes the Smart Device from a client point of view by specifying only the interfaces, not the inner behavior of the lab that is left to the lab owner's discretion. The Smart Device specifications are agnostic about the server-side hardware, but re-engineers the software component by adding 'intelligence' to handle complex tasks such as local supervision and validation or peers discovery for load balancing.

Smart devices often rely on WebSockets for data transmission with the client [6]. It is currently the most

978-1-4673-8246-5/16/$31.00 ©2016 IEEE 24-26 February 2016, UNED, Madrid, Spain
2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)

efficient way for transmitting real-time information from/to a Web application. Similarly, the transmitted data are often encoded using the JSON schema [7].

III. INTEGRATION IN MOOCS

Online laboratories are becoming more and more popular. In many cases, an online laboratory experiment consists only in remotely interacting with devices over computer networks without necessarily reflecting the educational objectives that underlie the activities. There is an ongoing effort to standardize the relationship between all the components (Software, hardware and learning environments) in order to ease the design and implementation of pedagogically driven online laboratory activities [8]. This standard aim is to ease the design, the implementation, and the usage of pedagogically oriented online laboratories as smart learning objects and their integration in learning environments and learning object repositories.

Figure 1 illustrates implementation levels for online labs. The low level refers to the online lab as a service (LaaS), which can be personalized at the intermediary level. The intermediary level refers to the online lab as an Open Educational Resource (LaaR), which can be integrated in learning environments, such as edX or Graasp

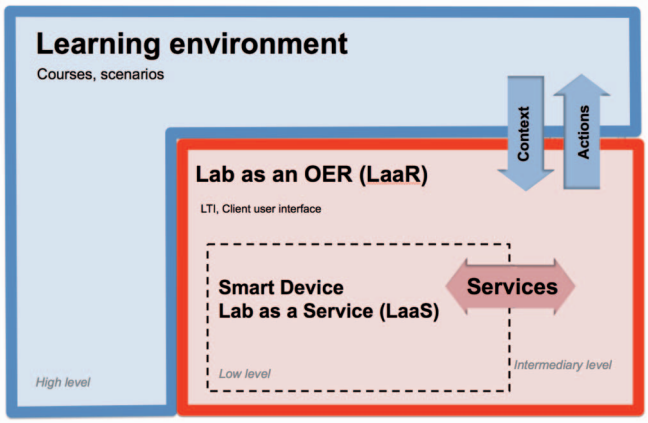

Fig.1. Implementation levels for online labs

The proposed levels fit well with the required MOOLs for MOOCs infrastructure. The MOOC (high Level) is used to deploy a full fledge remote laboratory made of smart devices (low Level) integrated as LTI modules (Intermediary level).

The MOOC infrastructure offers many features to support the students learning process. To take full advantage of the MOOC paradigm, classical courses and/or hands-on laboratories sessions are to be completely reworked. Classical MOOCs consist of videos interleaved with questions and online exercises. Automatic or peer grading is provided to support self-paced user progress.

Supporting a large number of users accessing shareable resources such as video or virtual simulation is "reasonably straightforward" provided that you have the needed servers and bandwidth. To handle the same number of users accessing a critical resource (i.e. not sharable) such as a remote lab is a challenge. To tackle this challenge and efficiently integrate remote lab sessions in MOOCs, the following actions can be combined: the first one is to **rework the classical hand-on laboratory** sessions such that existing experimentation activities are split into shorter parts to ensure that the critical resource, the remote experiment, is not hold for too long by a single user. Typical activities last for 30 seconds to a few minutes. The shortest the activities duration is, the larger the number of users handled per unit of time can be.

The second action is to **rethink how the user interacts with the remote equipment**. Is the distant equipment only to be observed? In this case the measurements and video frames can easily be duplicated at the server side and streamed to each client. If the envisioned scenario involves acting on the equipment, a policy has to be defined if more that one user wants to act at the same time. The smart device specifications propose various mechanisms and policies to implement resource sharing. A controller/observer mechanism with a first-come first-server policy is an efficient way to share a critical resource. Each user is able to observe the equipment concurrently, but only one user at a time can act on the equipment. If two or more users want to act at the same time, the requests are queued. The smart device proposes these mechanisms, but it is up to the client application to take advantage of them. It is especially important that the client application provide full awareness regarding the queuing/acting information to the client. This implies that waiting users are informed about the waiting time, similarly acting users are aware of the remaining time available for the current activity.

The third action is to **duplicate the critical resource** (the equipment) as a farm of remote labs to support more users.

Section VI suggests additional actions to handle a larger number of concurrent users using gamification.

There exist many MOOC platforms. Currently, the two main ones are Coursera and edX. The solution presented in this paper is implemented in edX but could be implemented in other platforms provided that the needed technology is available.

IV. MOOC INFRASTRUCTURE

The complete infrastructure to support MOOL and its integration into a MOOC is composed of the following elements (Fig. 2):

- one or more smart device servers
- an HTML client application running in a browser
- a *.cgi* interface for LTI authentication, database and other services
- an edX server

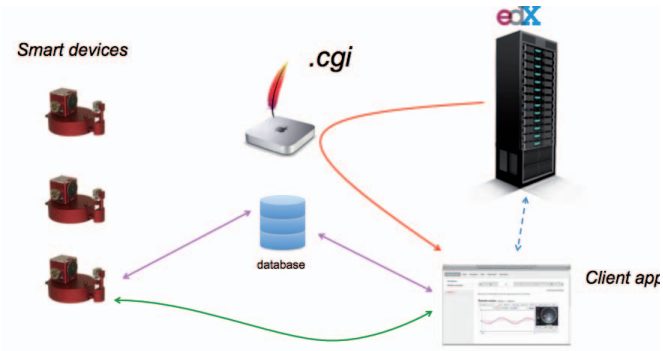

Fig. 2. The infrastructure with the smart devices, the cgi, the edX server and the client application.

A. *Smart device server*

The smart device server consists of the equipment to be controlled [2]. In our example, it is an electrical drive, and the computer that handles *i)* the connection with the equipment through sensors and actuators, *ii)* the local controller, *iii)* the web and WebSocket server and *iv)* the local "intelligence" that ensures that remote users behave adequately. The latter part also interacts with the storage services and the smart device that acts as the load balancer.

B. *Client application*

The client application is a pure HTML5 application that enables the user to fully interact with the smart device (Fig. 3).

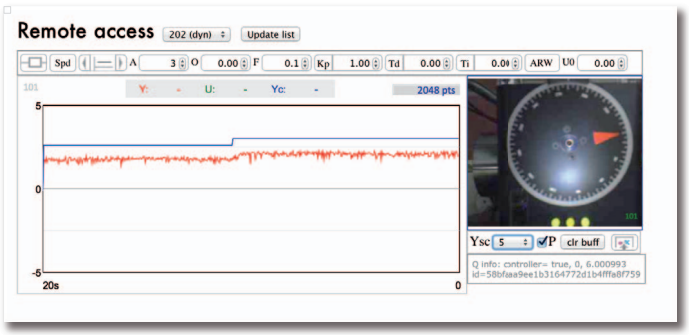

Fig. 3. HTML5 client application.

It provides measurements displayed in an oscilloscope window, real-time video stream, a control strip that allow users to specify the reference or control signal, the control parameters, and the controller structure. The controller/observer mode is supported by this interface. When in controller mode, all elements are accessible, when in observer mode the control strip is greyed out and an indication of the waiting queue size is provided in the "administrative" part. It also provides a mean to save current measurements in the hosting environment. When in observer mode, the client application cannot save measurements and other data. This application can be used in a standalone mode or within a hosting environment such as a MOOC platform or a LMS.

C. *The cgi interface*

The *cgi* interface is the corner stone between edX and the external tools. When an external module is added to a MOOC session as and LTI (external) module, the *cgi* first validates the LTI encoded request containing the edX user ID and other information such as the user role, then it servers the LTI module content that will be integrated as an iFrame in the edX page. In the proposed MOOC the content is either a Web interface to access the Smart Device (Fig. 3) or one of the Sysquake tools for data analysis or controller design (Fig. 5).

The client application is integrated in the hosting environment (edX MOOC) as an LTI module (Fig. 4). The LTI specifications permit to exchange information such as user ID between the hosting environment and the client application. It also ensures that the provided information is not tempered by signing the transmission with a shared secret. LTI carries the advantage of being widely used. This solution proposed for *edX* could also be ported with little effort to other environments supporting LTI such as *Moodle*.

D. *The edX server*

A MOOC can be hosted by the edx.org consortium, alternatively an institution can install its own instance of edX which is open source. It was decided to host the MOOC in our institution for the first run. This enables a tighter control of the MOOC and an agile development. The following iterations of the MOOC will be hosted by edx.org.

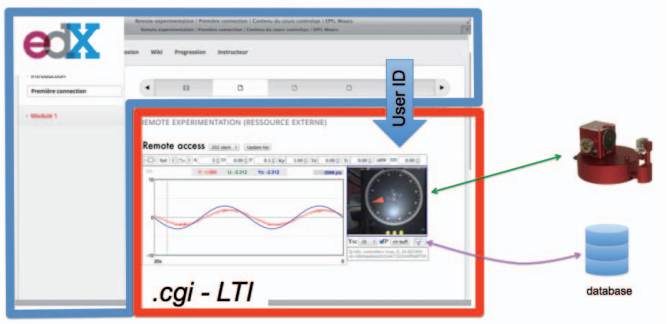

Fig.4. The edX MOOC hosting an external LTI module (served by the .cgi) which host the interface to the smart device. Information such as edX User ID is provided to the LTI module.

E. *Additional services*

1) *Saving*

edX can remember where the student stops the last time s/he was connected, other information such as numerical values that has been entered in edX are also stored. On the other hand the proposed LTI modules are state less and the edX's currently implemented version of LTI does not propose a mechanism to store/retrieve data from/to edX.

A database service has been developed outside of the edX's infrastructure to provide a private storage to each user. The database user identity is provided by edX and pass securely to the LTI tools via the *cgi* interface. This database service is accessible only by the LTI tools. Users can save various kind of information: measurement, model parameters, controller parameters, etc. The saved information is JSON encoded. Metadata are attached to the saved information. These metadata permit LTI tools to filter files to display to the user. At this time, the saving service is more a database than a complete file system. For instance, users cannot retrieve the saved data or upload data located on their computers. An evaluation of students' needs will be conducted to see if additional features are required.

Saving information between steps is a key element to enable continuity in an MOOC session. Section V provides an example of such session.

2) Data processing

Various Web tools are proposed to process data, other tools offer simulations or controller design (Fig. 5). Similarly to the smart device client application, these tools are encapsulated as LTI modules. These interactive tools are generated by Sysquake [6], a simulation engine compatible with the Matlab syntax that generates pure HTML5/JavaScript applications. These lightweight tools run in the client browser and do not require any computation on the server! Since there are encapsulated in an LTI module, these tools also have access to the user private space to store or retrieve information.

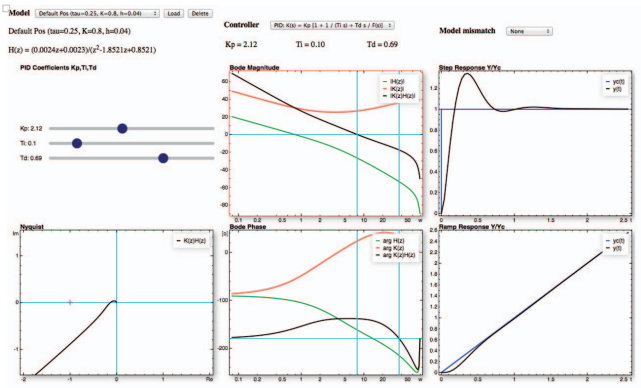

Fig.5. The pure HTML5 simulation tool for PID design generated by Sysquake.

3) Load balancing

The wealth of users is split among the pool of smart devices. For the current MOOC run there are 23 identical smart devices. One of the smart devices is dynamically chosen to act as the load balancer. Each smart device broadcast information about its status and the number of waiting clients. In collaboration with the smart devices, the load balancer guides the client to the smart device server that has the smallest number of clients waiting to act.

V. LEARNING SCENARIO

The proposed infrastructure is currently exploited and validated for a first MOOC fully dedicated to hands-on activities. It is offered at the Swiss Federal Institute of Technology in Lausanne (EPFL) to students of mechanical engineering, micro-engineering, as well as electrical engineering. The video parts of the MOOC describe how to use the various tools (client application, simulation, etc.), the lab infrastructure and the experiments to be performed.

The learning scenario of each session is split into short phases (~5 min each) which follows a common structure: a short video describing the experiment (with a theoretical recall if needed), an set of hands-on exercises using the remote lab clients integrated in edX, data pre or post processing using interactive simulation tools, a set of numerical questions to validate user's finding and another set of open questions based on the user observations to evaluate user's understanding (Fig. 6).

For example, a scenario where the user has to design a controller for a given equipment could have the following sequence:

i) Watch the introduction video, which provides a theoretical recall (ex. Loop shaping) and explain the steps to be performed in the current session.

ii) Perform a step response on the real equipment and save the measurements. During this step a remote connection to the smart device is established.

iii) Process the measurements using the temporal fit tool (generated by Sysquake) to identify key parameters (model transfer function), save the transfer function.

iv) Enter the transfer function parameter in edX for self-validation.

v) Using the identified transfer function, synthetize a controller according to some specifications. With the interactive PID loop-shaping tool (Fig. 5) load the identified transfer function and design your controller. Simulate various cases using the provided tool. Save the controller parameters.

vi) Test and validate the proposed controller on the real equipment through various experimentations.

vii) Add a perturbation on the real equipment, and test than the design controller is robust to these perturbations. For example an additional load (resistance) can be added on demand, alternatively delay can be added to the system.

viii) Answer additional comprehension questions.

In the above scenarios the remote experimentation is accessed in steps *i)* and *v)*. A specific experimentation time is defined for each task to be performed. This parameter, defined by the MOOC authors, facilitates a tight time control, read "set the minimal time", for each experiment. The smaller this time, the larger the number of users per unit of time. Note that the current access policy permits a user to stay connected as the controller until another client connects. The other steps *ii) - iv)* do not required timing since the Sysquake web tools used for analysis and simulation run on the client browser without live connection to the server.

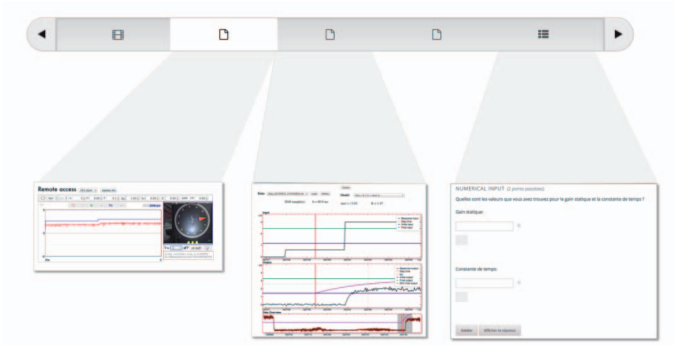

Fig.6. A sequence of steps in edX, represents steps *ii), iii), iv)*.

VI. Massive Access and Gamification

With the proposed actions (short activity time, controller/observer mode, FIFO queuing and equipment duplication) a ratio of about 5-10:1 has been achieved in the control system lab MOOC. Increasing the number of users by another order of magnitude requires the addition of mechanisms to select users who will be able to act on the equipment according to some criteria. This is like implementing a priority queue where selected users will wait less time.

A gamification process can be used to select users. In this controller design scenario presented above, a possible selection criterion for step *v)* can be: *how close are the controller parameters found by the user from the optimal ones?* Users with a correct set of parameters will be entitled to access the real equipment for testing while users with demonstrably bad answers will be re-directed to simulations. This simulation will highlight the problems with the proposed parameters. Similarly, such simulation could exhibit theoretical behaviors that would normally destroy the physical equipment.

The proposed tools and infrastructure, especially the Sysquake javascript simulations, support such gamification scenario.

Traditionally reservation is the preferred method to guaranty access to a share resource at a given time. Such mechanism is cumbersome to manage and to explain to users, in addition it can be very time consuming to implement. While the proposed smart device has the needed mechanism to support reservation, it is not used for the above rationales. Also such mechanism is antagonist to the MOOC idea that user should perform sessions at his/her own will. A substitute mechanism is to inform the user about the best time, i.e. the period at which the waiting time is the smaller, to connect to the equipment on a global scale. The Web client application reports the waiting time for the current smart device as well as the waiting time for the other smart devices in its neighborhood. Only the instantaneous waiting time are provided, providing trends over the week would permits the user to select the most appropriate experimentation time.

VII. Initial Results

Preliminary evaluation has been conducted. The main conclusion is that students agree to wait to access a shared resource if the waiting time is provided. During the initial experimentations the waiting time was less than 6 minutes, which indicated less than 3 users were waiting. If the waiting time information is not provided or incorrect, then students are not willing to wait for a "random" time and complain.

Also, even though students could switch manually to a resource with shorter waiting time, they prefer to remain on the same resource and wait for their turn, this is probably due to the structure of the MOOC where they can perform another tasks while waiting. Switching to another task and coming back to the original remote experimentation client will put the user at the end of the waiting queue, this limitation has been quickly overcome by students who open two tabs on they browser.

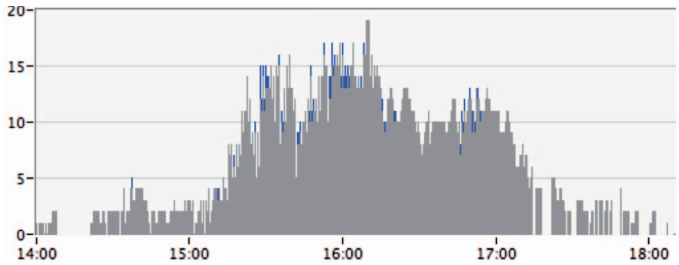

Fig. 7. Severs loads during a planned session (15:15-17:00), 21 smart devices were available.

Figure 7 shows a typical sever load during a planned session with 120 students. There is a total of 21 smart devices. The load balancer assigns free smart devices to new comers. The grey bars represent the number of used smart devices at a given time. The blue bars represent the number of waiting users for a specific smart device. This picture shows that even there are free smart devices (the highest grey bar value is 19) users prefer to wait for a given smart device. Also, the highest blue bar is 3, which represents the number of people waiting for a specific smart device. Outside the planned sessions, the load of the servers was rarely above 5. Figure 7 also shows that the available infrastructure can easily sustain the current planned MOOC sessions with a ratio of 6 to 1.

VIII. Conclusion

In this paper, we presented a solution to deploy remote laboratories within MOOC infrastructures. We first cover the technical aspects of this implementation using smart device paradigm, HTML5 client application, LTI, interactive tools, user storage and load balancer. Then, we presented a way to rework classical hand-on lab session in order to accommodate a new MOOC paradigm where many users access critical resources at the same time. The MOOC infrastructure has been detailed with an example MOOC learning scenario. Finally, gamification processes combined with simulations are proposed to elect users entitled for a direct access to the real equipment. The proposed solution is currently implemented in the *control systems lab* MOOC offered in the university instance of edX to more than 200 bachelor students in mechanical engineering, micro-engineering, as ell as electrical engineering at EPFL. It is the first MOOC at EPFL fully dedicated to hands-on lab activities.

IX. Acknowledgment

The authors acknowledge the support provided by the EPFL for the development of the control system MOOC described in this paper.

References

[1] Tawfik, M., Salzmann, C., Gillet, D., Lowe, D., Saliah-Hassane, H., Sancristobal, E., Castro, M. (2014). Laboratory as a Service (LaaS): A Novel Paradigm for Developing and Implementing Modular Remote Laboratories. *International Journal of Online Engineering*, 10(4), 13-21

[2] C. Salzmann and D. Gillet. Smart Device Paradigm, Standardization for Online Labs. 4th IEEE Global Engineering Education Conference (EDUCON), Berlin, Germany, 2013.

[3] C. Salzmann, S. Govaerts, W. Halimi and D. Gillet. The Smart Device Specification for Remote Labs, in International Journal of Online Engineering, vol. 11, num. 4, p. 20-29, 2015.

[4] C. Salzmann and D. Gillet. From online experiments to smart devices, in International Journal of Online Engineering (iJOE), vol. Vol 4, num. SPECIAL ISSUE: REV2008, p. 50-54, 2008.

[5] Lowe, D., MOOLs: Massive Open Online Laboratories: An Analysis of Scale and Feasibility. *REV2014: 11th International Conference on Remote Engineering and Virtual Instrumentation*, Piscataway, USA: (IEEE) Institute of Electrical and Electronics Engineers.

[6] Websockets, https://tools.ietf.org/html/rfc6455

[7] IEEE-SA P1876, https://ieee-sa.imeetcentral.com/1876public/doc/20128082/w-1876WorkingGroupPublic

[8] http://www.json.org

[9] Sysquake is a product of Calerga, www.calerga.com