Remote Experimentation Mashup

Vargas Héctor*, Salzmann Christophe**
Denis Gillet, Sebastian Dormido***

**National University for Distance Education (UNED), Madrid, Spain (e-mail: hvargas@bec.uned.es)*
***Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland (e-mail: christophe.salzmann@epfl.ch)*

Abstract: eLogbook is a new Web 2.0 social software framework developed in the Swiss Federal Institute of Technology in Lausanne (EPFL, Switzerland) to provide collaborative Web services in an open academic environment for higher education. Several cross platform Web-based technologies compose the internal architecture of this tool which is intended to online learning communities and community of practices with high social interaction demand. In this paper, we describe the necessary requirements to integrate external Web-based applications for remote experimentation into **eLogbook**. The integration of a virtual and remote control laboratory from the National University for Distance Education (UNED, Spain) is presented as a case of study.

Keywords: eLogbook, social software, control education, remote and virtual laboratories, personal learning environment (PLE), collaborative learning, online learning communities.

1. INTRODUCTION

This paper describes how external Web applications can be mashed up into a social software, namely eLogbook, with a specific focus on integrating external remote experimentation applications. **eLogbook** is a Web 2.0 personal learning environment developed at the Swiss Federal Institute of Technology in Lausanne. The integration of new Web tools not specifically designed for eLogbook is a challenge that is tackled using a combination of mashup technology, agents and smart devices. The paper is organized as follow: first, Section 2 presents current remote experimentation web solutions and their objectives. Then, Section 3 introduces eLogbook and its concept and model. Section 4 states the needed requirements for the integrations (mashup) of external web applications (agents) into eLogbook. The mashup of a new agent from the UNED Remote Laboratory is presented as a case study in Section 5.

2. REMOTE EXPERIMENTATION WEB APPLICATIONS

Remote experimentations have been widely available for more than a decade (Salzmann *et all.*, 2007). The advent of the Web promotes a wider adoption of remote experimentation since Web browsers are the ideal mean to ubiquitously execute the client application. The client application can typically be a pure html/javascript application or can require a dedicated plugin such as Flash, Java, ActiveX, etc.

Remote laboratories can be of many kinds; we focus on remote laboratories where the users mainly access physical equipment for remote experimentation purpose. Remote experimentation is typically introduced to complement hands-on laboratory sessions in traditional higher education settings, to avoid travelling to the training centres in distance learning or to offer live demonstrations in classroom sessions.

The objective of a remote experimentation solution is to make the student interaction with the distant system as close as possible as the actual work on the real equipment. In other words, the best possible feedback has to be provided to a user action so that the drawbacks inherent to the distance between the user and the physical equipment are minimized.

A typical remote experimentation session consists in performing real-time measurements on the distant system. These measurements can be streamed as data points or video feed or both. The user often has the possibility to save these measurements either on his desktop or in a provided shared space. The user can act on the distant system by changing its parameter remotely. This implies the implementation of careful safety measures and authentication to ensure access to authorized person only (Salzmann *et all.*, 2007).

At the EPFL, eLogbook is integrated in the automatic control hands-on laboratories activities offered to master students from Electrical, Mechanical and Micro Engineering. After an introduction session where the purpose of the laboratory sessions and the usage of the platform are introduced, students can work either locally or remotely to perform the three experimentation sessions dedicated to the identification and control of one of the 22 proposed electrical drives. eLogbook offers the possibility to share/save measurements, documents and other material between its internal share space and remote experimentation agents. eLogbook also offers the possibility of structuring tasks and learning by defining sub-activities.

Preprint submitted to 8th IFAC Symposium on Advances in Control Education.
Received May 15, 2009.

At UNED, remote experimentation has been used in the last years to give response to the demand from students who have difficulty to regularly attend the classes in the academic centres. Being a distance learning university, the introduction of this kind of education methodology has been a key piece to cover needs from students who must conjugate work and academic life. These laboratories are offered to master students in engineering school. Students are offered the possibility to first have a direct contact with the available systems (a *direct current servo-motor*, a *control system temperature* and a *coupled threetank system*) and then, they can complete their work remotely by making use of the web-based experimentation system provided by university. During experimental sessions, they can save data measurements and parameters, which they will use later to write their reports.

3. ELOGBOOK

eLogbook is a Web 2.0 social software application developed in the Automatic Control Laboratory from EPFL (Switzerland). The public of eLogbook are online learning communities and communities of practices which share an open framework for discussing, learning, collaboration and for exchanging ideas and knowledge about different topics. In the context of remote experimentation eLogbook is used as a personal learning environment (PLE) where users access various tools to collaborate and organize their work on the distant system. Students, teaching assistants and educators are considered as members of a learning community with predefined roles. In addition to the standard share repository feature, the proposed tools permit to interact, measure, analyse, design and test solutions.

The eLogbook is based on the 3As models where **A**ctors generate **A**sset within a given **A**ctivities. This 3As model is described in detail in (Gillet *et all*., 2008) and is summarized hereafter.

An **actor** is any entity capable of initiating an event in the collaborative environment. It can be a person, a Web service, a software agent or even an online physical device. In that sense, the proposed definition of an actor is broader that the traditional definition of social software users or community members. An actor is an artefact for doing/producing something, i.e. an agent or an instrument corresponding to "who?" or "which?". In the remote experimentation framework, actors are members of a group carrying out a laboratory experiment and the associated remote experimentation software represented as non-human actors or agents.

An **asset** is any kind of resource produced by or shared between community actors. The proposed definition goes beyond the typical digital assets like rich-text documents or multimedia resources. It can also include as example discussion threads or wiki pages. An asset is an artefact done/produced by an actor somehow, i.e. a product corresponding to "what?". In the remote experimentation framework, an asset can be measurement or configuration file, students' reports, laboratory protocols or instructions.

An **activity** is the formalization of a common objective to be achieved by actors such as discussing topics or completing tasks. An activity is an artefact for doing/ for producing something, i.e a purpose corresponding to "why?". In the remote experimentation framework, an activity can be a laboratory session.

The **eLogbook** has been design from the ground up to support external tools and applications. The 3As model that is implemented in eLogbook is general enough to conceptually mashup (combine) any Web applications into the 3As model (Bogdanov *et all*., 2008).

In the case of remote experimentation, agents are used to map the remote experimentation functionalities into eLogbook (Salzmann *et all.*, 2008a). An agent is a non-human actor. From an eLogbook point of view agents are actors since that can initiate events in the collaborative environment. These actions can be generated by the agent itself (alarm generation) or on the behalf of the user using the agent (saving in eLogbook measurements performed on a remote system). Agents can have multiple forms and various degrees of "intelligence".

The "intelligence" placed on the remote system can be limited and thus the agent needs to perform numerous tasks or the remote system can be a smart device (Salzmann *et all.*, 2008b) that can directly be integrated or mashed up with eLogbook.

The eLogbook Graphical User Interface (GUI) consists of one central area where the information related to the selected entity is displayed (Figure 1). There are three surrounding areas, one for each 'A' (left: Actor, top: Activity, right: Asset) that display a list of the related entities. The user navigates within eLogbook by selecting the appropriate entities. When an entity is selected via a mouse click, its information is displayed in the center area and the surrounding lists are dynamically updated to reflect their new relationship to the center entity. This selection of a focal entity and the display of the related ones in the surrounding areas make the eLogbook GUI contextual.

Fig 1. eLogbook contextual Graphical User Interface (GUI)

eLogbook promotes both asynchronous and synchronous interaction. By introducing agent, eLogbook supports live interaction through specific synchronous activities called sessions. The proposed live interaction enables actors to

Preprint submitted to 8th IFAC Symposium on Advances in Control Education.
Received May 15, 2009.

collaborate with other actors or between actors and agents in a flexible approach.

eLogbook is designed as a mashup container, which mean that it can handle the result of a mashup between two or more applications. At this time the mashup integration requires manual intervention, but it has internally all the needed elements to enable a seamlessly integration.

4. REQUIREMENTS FOR THE INTEGRATION OF EXTERNAL AGENTS

The minimal requirement for the integration of external agents is their ability to communicate using either a standard Web protocol (HTTP, RSS, etc) or a using lower level protocol (UDP, TCP) with the help a Web plug-in. A simple agent could for example provide a remote temperature, with the eLogbook providing the needed UI to display the received information. The considered remote experimentations setups have more advanced feature and thus the requirements are more stringent. These remote experimentation setups are often implemented as smart devices (Salzmann *et all,* 2008b) and thus minimal mashup efforts are required. These requirements are: *i)* authentication, *ii)* GUI and *iii)* bidirectional communication. While many remote experimentation setups relie on a specific Web plug-in (java, Flash) to overcome the current real-time limitations new specifications may soon permit a pure HTML/Javascript solutions (W3C, 2009).

4.1 Authentication

The remote experimentation agent may need to authenticate to work on the behalf of a given user. This authentication enables the user to remotely control the setup, i.e. change its parameters and also enable to save the performed measurements on a shared space for example. The remote experimentation agent needs to be able to adapt to the authentication mechanism provided by the host environment. This operation may require the help of a third party. Ideally the user only authenticates once and all the agents running within the same session (browser) know about the authentication, this mechanism is generally implemented with the help of cookie. Similarly when the user logs out, all the agents working on the behalf of the logged user are disconnected.

4.2 Graphical User Interface

While the common elements such as user (actor) or document (asset) can directly be mashed up with the hosting environment, it is difficult to standardize specific aspects of the remote experimentation GUI without drastically limiting its functionalities. Thus it is the agent responsibility to provide a GUI for these elements. Graphical interfaces are often implemented with the help of a Web plug-in. For example, the GUI presented here after are implemented in java (see Figure 6). Similarly to data communication new Web standards may soon provide the needed elements to design a pure HTML/Javascript solution.

4.3 Bidirectional data transmission

Remote experimentation setups often propose more than one way to interact with the outside world. This is especially true when the remote experimentation setup is implemented as a smart device. For example the smart device presented in (Salzmann *et all,* 2008b) is able to stream in real-time live measurements and video feed but it can also send an email with the latest measurements. It also uses Web Services (XMP-RPC) to respond to requests about its status and in case of failure it is able to send SMS alerts. The bidirectional data exchange process often require a fine control to guaranty the information delivery in a timely manner, this is especially true for live data streaming. At this time such control can only be implemented with the help of Web plug-in to access protocols (TCP, UDP) which are currently not accessible from HTML/Javascript.

4.4 Sessions in eLogbook

To get a full access to all services provided by eLogbook users of any type (either *agents* or *human actors*) must initialize a *session* on the environment. Once logged in, *actors* and *agents* can interact with each other and with their own environment around a specific *activity*. During these interactions, they can generate/tag/rate *assets*, which later can be used by a group of *actors* for discussion. Web 2.0 social software seeks to encourage these interactions among entities by providing tools and services that stimulate the contribution and collaboration among users. Figure 2 illustrates these interactions through a simple *entities relationship scheme*.

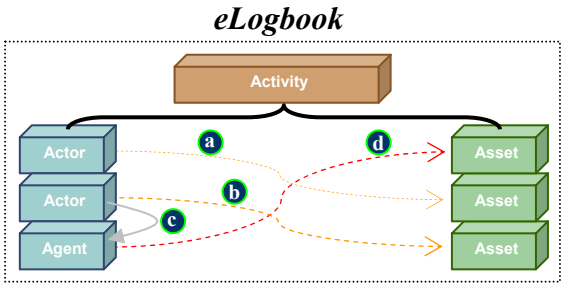


Fig. 2. Relationships among entities of eLogbook.

5. CASE OF STUDY: THE MASHUP OF A NEW AGENT FROM UNED REMOTE LABORATORY TO ELOGBOOK

In a collaboration framework between EPFL and the Department of Computer Science and Automatic Control at UNED (Spain), the possibility to mashup applications from other universities into eLogbook has been explored. The chosen application is the remote experimentation of a didactical setup. This existing application can already be remotely controlled with the help of a custom based GUI written in Java.

eLogbook already includes external applications or tools as agents. There are two tools designed to perform data analysis and one agent to connect to one of the 22 electrical drives

Preprint submitted to 8th IFAC Symposium on Advances in Control Education.
Received May 15, 2009.

available at the EPFL control laboratory. These tools and eLogbook were designed by the same team and thus the mashup operation was almost straightforward. The mashup validation scenario implies the integration of an application that was not directly conceived for eLogbook.

The next sections present the UNED setup, a *Direct Current Motor*, its interface and its mashup process to integrate the UNED remote experiment into eLogbook.

5.1 DC Motor from UNED Remote-Lab

The *Direct Current Motor* is one of the classical experiments in automatic control laboratories. It allows studying the dynamic behaviour in *speed* and *position* of a motor which is fed by a *direct current source*.

A common actuator in control systems is the *DC Motor* and therefore, although is a simple and straightforward example, it requires special attention given its educational contribution (Vargas *et al.*, 2008). Figure 3 shows the didactical equipment and its hardware components.

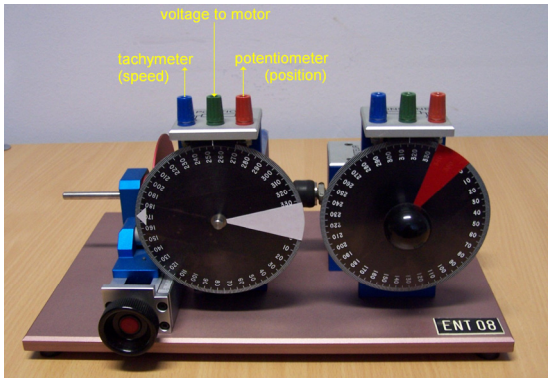


Fig. 3. The *DC Motor* didactical equipment located in the UNED Remote-Lab.

The didactical setup is an electromechanical process intended to perform the *angular position* or *speed* control of a load using the direct current supplied to the motor as a manipulated variable.

The charge is composed of an inertia (steel disk tight to the motor's axis) and a viscous friction due to a magnetic brake. The engine is fed through a unity gain amplifier capable of delivering the power required to the motor.

The current on-line *DC Motor* plant available at UNED will serve as the test infrastructure to explore the integration of external applications into eLogbook. Figure 4 depicts the global architecture of the remote laboratory. The client and server applications have been programmed using Easy Java Simulations (*Ejs* by short) (Esquembre, 2004) and LabVIEW™ (LabVIEW, 2009), respectively.

The server-side application performs the data acquisition and closed-loop control of the motor using LabVIEW. On the other side, the client application is a Java applet (developed using *Ejs*) running in a Web browser. Both sides implement the transmission layer needed to exchange data with each other (in this case, TCP protocol has been chosen). *Visual feedback* of the distant equipment is also provided to the client application with the help of an IP camera pointing to the real equipment.

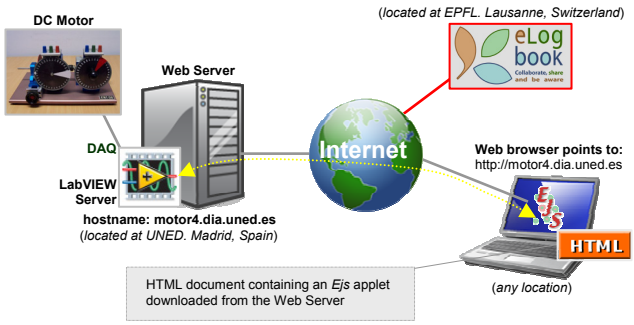


Fig. 4. Global architecture of the *DC Motor Remote-Lab*.

Through the applet graphical user interface, students can observe the effect in the dynamic behaviour of the motor during the remote manipulation of the system. They can also save data register (parameters and measurements) of the carried out experiments for later analysis.

5.2 Integration into eLogbook

Figure 5 shows the *activity* (or *space*) named "**EJS**" which has been created into eLogbook to deploy the integration of the *DC Motor remote-lab*. The centre part of the environment displays a brief description of the *activity* goals. A group of four participants compose this new *space*. The first two are human actors (authors of this paper) while the other two are remote experimentation agents coming from both the laboratories of UNED and EPFL, respectively. Both agents can be accessed by clicking on the agent image.

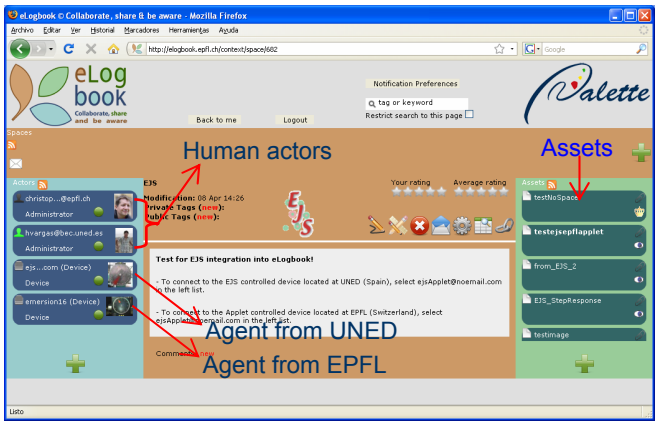
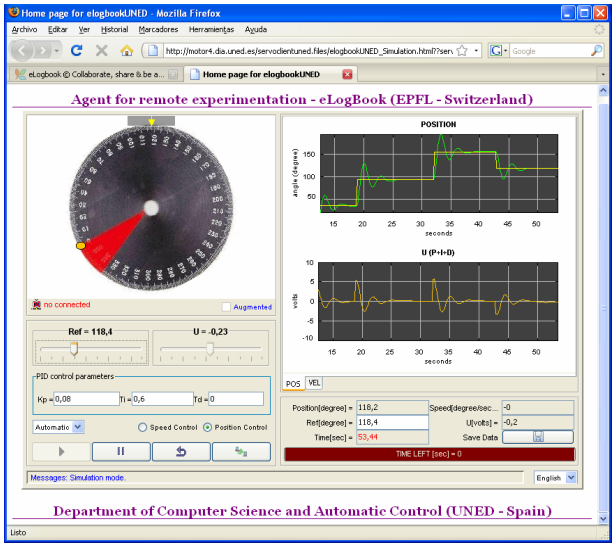


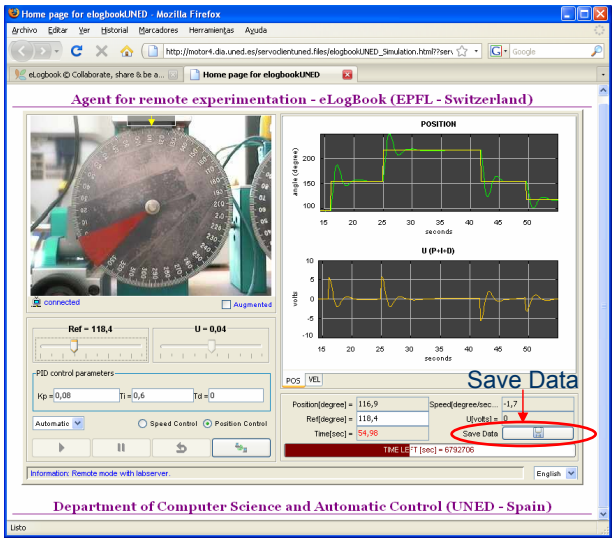
Fig. 5. EJS *space* into eLogbook. Two human actors and two experimentation agents compose the *activity*.

Figure 6 shows the *Ejs applet* deployed when accessing the UNED agent. The GUI of the system presents two operation modes: *simulation* and *remote*. When running in simulation mode (Figure 6.a), the view of the application is represented by a scheme of the real process whose visual aspect changes

according to the temporal evolution of a mathematical model. On the other hand, when the interface works in remote mode (Figure 6.b), the interface connects to the real equipment located in the laboratory of the university (in Madrid, Spain) to exchange data. At the same time, the graphical representation of the process is replaced by video images coming from the IP camera that looks to the steel disk tight to the motor's axis.



(a) Simulation mode.



(b) Remote mode.

Fig. 6. Graphical User Interface of the DC Motor remote experimentation agent into eLogbook.

Furthermore, the interface incorporates two key features for integrating new remote experimentation agents into eLogbook framework: *i)* The inclusion of the Java code needed to recognize the *session* of the user logged in and, *ii)* the use of the *Web-services* provided by the environment to save data as *assets* during a session.

These communication tasks can be easily implemented in a Java program by making use of the API provided in a package called "eLogbookcomm.jar". *Ejs* developers can import this library in their programs to get communicated with eLogbook.

On the one hand, to retrieve and keep information about user's sessions, the API provides a utility class called *UserDetails.java*. This class acts as a container of information relative to an *actor* into eLogbook. When the *Ejs applet* is downloaded from the environment, this initialization page is executed and *user's details* are retrieved from URL parameters and kept up during a session. Thus, the *Ejs* application knows how and where to send a new *asset*.

Standard *Web Services* are used to send measurements made on the remote experimentation to eLogbook where they are saved by as *assets*.

The Java communication API also disposes of an interface that makes easier the building and the sending of new *assets*. Table 1 lists the methods provided by this interface. As mentioned above, these methods are published as *Web-services* by eLogbook and Java programmers can use them to interact with the environment.

Table 1. Methods of the ELogbookInterface.java class

public UserDetails getUserDetailsFromELogbook(UserDetails ud); >> Returns more relevant user's details from eLogbook.
public List getAssetList(UserDetails ud); >> Returns a list of assets associated to a user.
public List getParamsAssetList(UserDetails ud, String tag); >> Returns a list of assets associated to an user regarding a given tag.
public String getParamsSet(int assetID, UserDetails ud); >> Returns a list of parameters associated to an asset.
public List getSpacesList(int deviceID, UserDetails ud); >> Returns a list of spaces associated to a user and a device specified.
public List getTagsList(UserDetails ud); >> Returns the list the tags associated to a user.
public int createAsset(String name, String note, AttachType attachment, String[] tags, int spaceID, UserDetails ud); >> Builds the asset to be sent to eLogbook.

The first method to invoke from an eLogbook-enabled Java program is the *getUserDetailsFromELogbook()* method. This method gets additional information about the user logged into eLogbook. The other methods help in the management and the building of a new asset, based on the information stored in the *UserDetails* object. The methods to get lists of *assets*, *spaces* and *tags* can be used to propose different inputs information to the *createAsset()* method, which is finally used to compose the *asset* according to the input arguments chosen and then send it to eLogbook.

In order to make this task transparent for the users of the *DC Motor* remote experimentation agent**,** a new dialog window was added to the *Ejs applet* (see Figure 7). This auxiliary window is launched when pushing the *Save Data* button located in the right lower part of the main application. With the help of this interface, users can select the type of *asset* to save, provide the *name*, *tags* and *notes* associated to this *asset* and choose its destination repository. When typing the name of a new asset, tag or at choosing a destination space to save the asset, a pop-up list proposes a set of possible inputs for the field. Internally, these functionalities are provided by their respective *getXxxxList()* methods.

Preprint submitted to 8th IFAC Symposium on Advances in Control Education.
Received May 15, 2009.

Figure 7 shows the setting up for sending a new asset named "*asset_ejs*" in which the "**EJS**" *space* has been chosen from a list of available spaces for the user logged in.

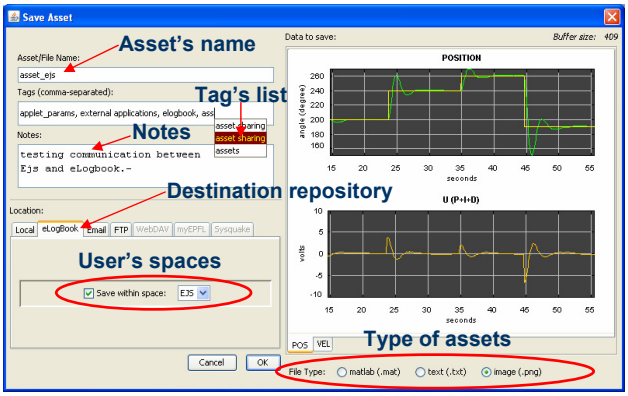


Fig. 7. Interface between the *UNED remote experimentation agent* and the *assets zone* on eLogbook.

Three types of data fragments can be saved as assets on eLogbook:

1. Matlab-enabled data files (*.mat),
2. Generic text data files (*.txt) and,
3. Image files displaying the scope area (*.png).

The data to be saved correspond to the existing *Ejs applet* internal buffer content, this at the time the *Save Data* button is pushed. An image representing the data to be saved is previewed in the right side of Figure 7. Once all the required parameters are provided, the measurements packaged as an asset can be sent to eLogbook by pushing the *OK* button. Upon the asset reception eLogbook updates its left column in order to reflect the new addition. The members of the current activity can then add comments, tags and rate the created asset. These additional Meta data are then indexed and can be sorted and/or searched.

Users can also perform additional processing on the saved measurements by opening the assets in another integrated application. Currently eLogbook propose two applications to perform time and frequency domain data analysis.

6. CONCLUSIONS

In this paper we described the mashup of external Web applications into a social software, namely eLogbook**,** with a specific focus on integrating external remote experimentation applications. eLogbook is a Web 2.0 personal learning environment developed at the EPFL that can act as a mashup container. The integration of new Web tools not specifically designed for eLogbook is a challenge that is tackled using a combination of mashup technology, agents and smart devices. The proposed requirements for mashup are validated with the integration of a java application developed at UNED Remote-Lab (Spain) to remotely control a DC Motor. Once the mashup operation completed, the assets generated by the UNED remote experimentation agent can seamlessly be used by any other eLogbook agents, tools or users.

The proposed approach enable educators to take advantage of social software not only to support collaborative learning in a more flexible way corresponding to the natural interaction schemes of the students, but also to provide access to remote experiments or any smart devices in a generic way. In such a way, social software becomes an artefact to link the Internet of People with the Internet of Things.

REFERENCES

Salzmann, Ch., and Gillet, D. (2007), Challenges in Remote Laboratory Sustainability. *International Conference on Engineering Education – ICEE 2007*, Coimbra Portugal

Gillet D., El Helou S., Yu C.M. and Salzmann Ch., (2008) Turning Web 2.0 Social Software into Versatile Collaborative Learning Solutions, *The First International Conference on Advances in Computer-Human Interaction (ACHI'08)*, Sainte Luce, Martinique, February 10-15

Bogdanov E., Salzmann Ch., El Helou S., and Gillet D., (2008), Social Software Modeling and Mashup based on Actors, Activities and Assets, *Workshop on Mash-Up Personal Learning Environments (MUPPLE'08) at the 3rd European Conference on Technology Enhanced Learning (EC-TEL08)*, Maastricht, The Netherlands, September 17.

Salzmann Ch., Yu C.M., El Helou S., and Gillet D., (2008), Live Interaction in Social Software with Application in Collaborative Learning, *3nd International Conference on Interactive Mobile and Computer Aided Learning,* Amman, Jordan, April 16-18.

Salzmann, Ch., and Gillet, D. (2008b). From online experiments to smart devices. *International Journal of Online Engineering*, vol.4, pp. 50-54.

W3C 2009, HTML 5 specifications, http://dev.w3.org/html5/spec/Overview.html

Esquembre, F. (2004). Easy Java Simulations: A software tool to create scientific simulations in Java. *Computer Physics Communications*, vol. 156 (2), pp. 199-204.

Vargas, H., Sánchez, J., Duro, N., Dormido, R., Dormido-Canto, S., Farías, G., Dormido, S., Esquembre, F., Salzmann, Ch., and Gillet, D. (2008). A systematic two-layer approach to develop Web-based experimentation environments for control engineering education. *Intelligent Automation & Soft Computing*, vol. 14, pp. 505-514.

LabVIEW (2009). NI Website. http://www.ni.com/labview

Preprint submitted to 8th IFAC Symposium on Advances in Control Education.
Received May 15, 2009.