Integration of Online Laboratories-LMS via SCORM

Ildefonso Ruano-Ruano, Juan Gómez-Ortega, Javier Gámez García, Elisabeth Estévez-Estévez
Group of Robotics, Automation and Computer Vision (GRAV)
University of Jaén
Jaén, Spain
Email: {alonso, juango, jggarcia, eestevez}@ujaen.es

***Abstract*—Web-based laboratories have been used as valuable resources to support teaching for many years. The methodology and operation of laboratories over the Internet have changed considerably since the early implementations. In fact, one of the last steps taken related to this issue has been the integration of laboratories and Learning Management Systems (LMS). In this paper we present methods that facilitate the development of virtual and/or remote labs software to interact with Learning Management Systems using Shareable Content Object Reference Model (SCORM), the most important standard for e-learning content. A virtual lab model is developed to show how the methods are used in order to integrate the virtual lab software with the LMS via SCORM.**

Keywords-component; Learning Management System; Online Laboratories; SCORM; EJS

I. INTRODUCTION

Lab work is unquestionable needed for many subjects in higher education, especially those related to engineering [1]. There are many reasons to incorporate online laboratories into higher education. Among them we can highlight the cost savings, a more extended use of scarce resources, the possibility of sharing equipment with other organizations, an improved security for users and a better availability of resources in time and space.

Virtual/Remote labs (VRL) can be understood as a way of e-learning. The first prototypes of remote laboratories for education were custom made using specific purpose servers and protocols [2]. After that, the idea of using the web to support the experiments was introduced [3] and new approaches emerged to develop and provide access to online laboratories that were supported by specific platforms and architectures [4] [5]. The e-learning has evolved due to the development of tools and standards that facilitate interoperability of content. Today Learning Management System (LMS) is a basic tool for students in e-learning. It contains most, if not all, of the content and tools that are used in teaching. Therefore, the logical evolution of online labs is to achieve integration with LMS, like many other projects that have been carried out in this direction [6].

The integration of online labs and LMS offers several advantages for developers as well as for students. Mainly, LMS controls the access of users to the website and ensures that all teaching resources are offered in the same environment.

LMS has positioned itself as an essential integrative tool that provides all the e-learning resources in the same environment; while Shareable Content Object Reference Model (SCORM) has proved to be a set of standards and specifications for e-learning that enable content sharing between LMS [7]. SCORM was created for developers to generate learning content by making it durable, accessible, reusable and interoperable with LMS. In order to achieve this objective, SCORM provides guidelines for launching and managing the content, a common mechanism for the contents to communicate with the LMS, and a predefined language/vocabulary that forms the basis of this communication.

The first online laboratories, integrated in LMS, tried to use laboratory models that were already developed using their particular technologies, ignoring the *de facto* standard for e-learning content, SCORM, because it was integrated them by developing LMS specific modules and plug-in that catered to the particular characteristics of individual laboratory [8]. Although they got promising and powerful results [9], this proposal was not directly reusable in other labs.

More recently, a new procedure was developed. It includes online laboratories in SCORM modules that allow partial integration with the LMS. It offered a web portal that worked as a repository of laboratories that were created by embedding laboratory software in a SCORM format [10]. This meant that no technological requirements were made to develop the laboratory, so any technology was valid. However, this system besides complicated the communications required to carry out an experiment (real plant, SCORM repository, the LMS and the PC) and, above all, did not offer a complete interaction between the laboratory software and the LMS.

In this paper we present a set of functions/methods for developers of VRL and where goal is that their labs interact directly with an LMS. Using these functions the VRL can communicate with the LMS and achieve a complete integration.

This set of tools is incorporated into the online laboratory software which should be generated as an applet. The applet is included in an html file inside a SCORM format packaging. Finally the generated SCORM content can be used by learners in an LMS to work in the online laboratory fully integrated.

The embodiment shown in this paper makes an encapsulation of the functions provided to the SCORM content

by the LMS (using an instance of Application Program Interface, API). It allows them to handle the data model of the SCORM Run-Time Environment (RTE) [9]. This way the details of SCORM-LMS communications are transparent to developers. As an example of validation, a virtual laboratory has been developed using the Easy Java Simulation (EJS) tool [11], which in turn uses the Java programming language to communicate with the LMS. This example is exported as a Java applet that, once embedded within a SCORM format, must be imported as a learning content in an LMS.

The rest of the paper is organized as follows: in section II, the main features of SCORM and its 3 components are explained, emphasizing the RTE and its basic operating principle. In section III, the performance of the developed functions is explained. In section IV, a virtual lab example is shown, which was developed at EJS. Once it is integrated into a SCORM module interacts with ILIAS, which is the LMS, or a virtual teaching platform, used at the University of Jaén (UJA). Finally, section V shows the conclusions of the paper and presents future actions.

II. SCORM APPLICATION TO ONLINE LABS

The SCORM standards and specifications are the *de facto* e-learning standard. The set of products that adopt SCORM includes most of the LMS market [12]. SCORM is documented and maintained by the Advanced Distributed Learning Initiative (ADL), but it's derived from work done by various industry and technology organizations, including the IMS Global Learning Consortium (IMC), the Aviation Industry CBT Committee (AICC) and the Electrical and Electronics Engineers Learning Technology Standards Committee (IEEE LTSC). SCORM provides a framework that applies to e-learning content in an LMS in order to define its encapsulation, launching and data exchange. With this objective three sub-specifications have been defined:

- Content Packaging section (Content Aggregation Model, CAM) [9]. It specifies how content should be packaged and described. It's based mainly on XML (eXtensible Markup Language).
- The sequencing section (Sequencing and Navigation, SN) [9]. It specifies how the learner can navigate between parts of the SCORM content (*SCO* o Sharable Content Object). It's defined by a set of rules and attributes written in XML.
- The Run-Time section (RTE) [9]. It specifies how content should be launched and how it communicates with the LMS. It's based mainly on ECMAScript (JavaScript).

In order to perform this work three sub-specifications have been used, paying more attention to the RTE sub-specification.

In the SCORM context there are two types of objects: assets and *SCOs*. The assets are not able to communicate with an LMS, SCORM only requires that an LMS launches the assets to the learners by using the HTTP protocol. Examples of assets are a video, an image, a text file, a sound file, an html page or any other piece of content. Within a SCORM module, a *SCO* is the smallest logical unit of information that a LMS can deliver to the learners via an LMS and can communicate with the LMS. A *SCO* is also defined as the only component of a SCORM that uses the SCORM API, which is the communication mechanism for retrieving and storing data between the LMS and the *SCO* (e.g. score, time limits, objectives, etc.) and for displaying the conceptual communication state between the *SCO*s and the LMS (e.g., initialized, terminated and/or in error conditions). The information that can exchange a *SCO* and an LMS is defined by the predetermined and standardized data model. This data model is used to define the data model elements that the *SCO* and the LMS must know. The LMS must maintain the data model in the same way independently of LMS used.

When a LMS launches an active SCORM content (*SCO*), it must be done it in a web browser window that is a dependent window of the LMS window that exposes the API instance as an object of the Document Object Model (DOM). The instance of the API must be provided by the LMS. A LMS can only launch and register a *SCO* at once (per learner). The *SCO* must search recursively in parent window hierarchy until it finds the instance of the API, once achieved, it can initiate a communication session with the LMS.

The SCORM API consists basically of the definition of only 8 functions that can be used by a *SCO*. These functions can be structured into 3 categories:

- Session Functions (Initialize() and Terminate()). Used to set the start and the end of a communication session.
- Data Transfer Functions (GetValue(), SetValue() and Commit()). They are used to exchange SCORM RTE model data between the LMS and the *SCO*.
- Support Functions (GestLastError(), GetErrorString() and GetDiagnostic()). They are used to support communications between the *SCO* and the LMS.

Once a LMS launches a *SCO*, the *SCO* can initiate a communication session with the LMS to store/retrieve information to/from the RTE data model. This is achieved by making calls to JavaScript functions that contain the instance of SCORM API that the LMS facilitates, where the *SCO* is located. The process may seem easy, as there are only 8 functions; however, it can be complicated because the data model for SCORM RTE has a fairly significant number of data and subsets of elements in many of them.

III. INTEGRATING ONLINE LABS AND LMSS VIA SCORM

The aim of this work is to facilitate the integration of online laboratories, virtual or remote, with the LMS. This is why, we tried to find a solution that is as universal and extensible as possible, and so we decided to use SCORM to support the laboratory software. The questions to be answered now are: How can we include an online laboratory in SCORM?, How can we achieve communication between the laboratory and the LMS?.

The answer to the first question is easy; SCORM is an open format that is offered as html page, which is fully compatible with the majority of VRL that usually are offered to students through web sites. This has been carried out successfully in

previous works [10], offering the ability to access a repository of virtual and remote laboratories that are embedded in a SCORM format.

However, the answer to the second question has not been found yet, because the laboratories included in this way do not have any chance of communicate with the LMS model for data exchange. To answer the second question some changes need to be made in laboratory software in order to access the API JavaScript functions of SCORM module where the laboratory is embedded.

This can be achieved by facilitating its use, by developing a set of native functions to the development technology of the lab in order to conceal the problem of access to the SCORM RTE data model. This way the online laboratories developers can use these functions to integrate their development with the LMS easily, and they will not need to know in detail the features of SCORM RTE.

For the development of online laboratory we have used java, which is one of the most used programming languages. We have also used the netscape.javascript.* package to access the DOM of the html page where the laboratory applet is included and can invoke the JavaScript functions of the SCORM API as recommended by Oracle (the company responsible for Java) [13].

We have developed functions to access to the data value of the RTE data model that holds the LMS and also change it in cases where it is permitted. The functions that have been created can be classified in two types: those that can retrieve RTE data model values and incorporate to the software of the laboratory, and those that can modify some value of the RTE data model depending on the work done by the student when using the lab software.

Some of the RTE model data that have been consulted with these functions are the learner name that the LMS keeps in its database, the API version and the status of completion of the *SCO* in which is located the lab applet, so they can be displayed in a window of the applet. Thus, for each of these consultations, we have created a specific function in order to facilitate its use for the future developers of the laboratories who will not need to know the details of this model or of the communications that take place.

We have also created functions that modify data values of the RTE model and create new elements when it's possible, like some data sets that are associated with the comments. New comments can be created from the online laboratory (Java applet) in order to be stored in the LMS, these comments may include a) data strings written by the learner or automatically generated by the lab software, b) the timestamp (date and time) and c) the location where they were created.

Figure 1 illustrates, in an UML sequence diagram, the involved entities and the methods for providing support to virtual laboratories with LMS via SCORM offered resources. First, the learner is logged in to the LMS platform. Once the main webpage is loaded, after, the learner selects the exercise which is in a particular SCORM module. As has been said before, every SCORM is formed by a set of *SCO*s (sco_i). These latters could include an Applet and, if necessary, can include the load of the APIWrapper JavaScript.

On the other hand, the RTE data model, stores information related to the complete SCORM, but also, *SCO*'s dependent data. Figure 1 illustrates the most complex example, i.e. *SCO*s composed by applets and JavaScript functions. Therefore, sco_i requires to load the JavaScript and the corresponding applet. Between these two entities the Learner session is started and initialized. In this case, the applet prints the LMS user name. To do this, the rteGetLearnerName() has been carried out. This function invokes doGetValue wrapper function, which encapsulates the GetValue() (provided by SCORM API). RTE data model content depends on the goal of the practice. This is expressed in a loop with a set of RTE data read/write.

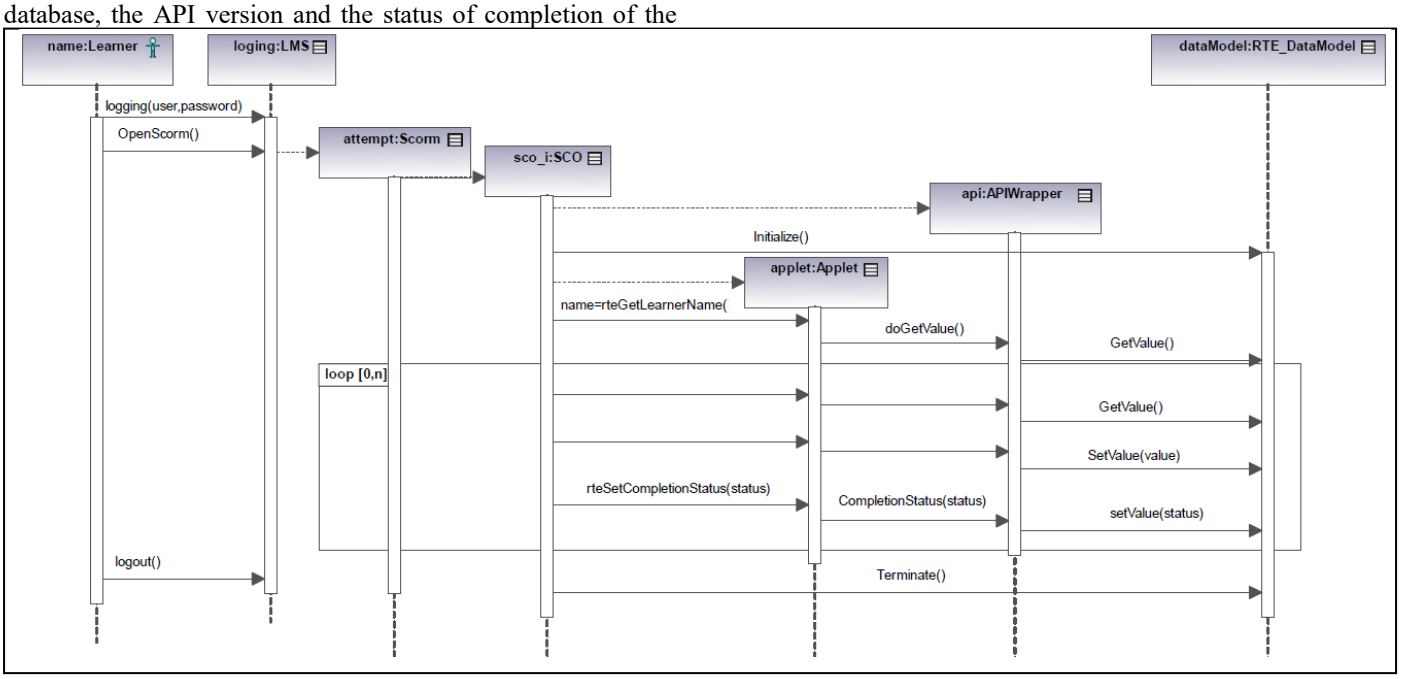

Figure 1. UML Sequence diagram example with the involved entities and the methods.

IV. CASE OF STUDY: CAR'S SUSPENSION SYSTEM

A virtual laboratory has been developed with EJS to create a case study that helps to validate the proposal. We have used the EJS software, because it holds highly recommended features, among which are: an extensive worldwide use, its ease to handle, the fact that it is generated under free license and the ability to generate the laboratory software as Java applet format. It could have been a remote laboratory because communications that exist in a laboratory model do not matter when using the developed functions –it can be used with all types of online laboratories, virtual or remote. The developed virtual lab is a performance simulation of a steering damper in a car in which there is the possibility to work under different conditions. In the Java code of the laboratory we use the developed functions in order to interact with an LMS. Once the lab is exported from EJS as Java applet, it's included in a web page which is the main resource of a *SCO*. The easiest and recommended way to create a SCORM content package is by using a template. For this case, we used an example of basic SCORM package downloaded from the ADL website. It contains a structure with HTML files that have been modified to include a virtual lab. This modification has basically consisted in including the laboratory applet that must have been previously exported from EJS, within an html page of the downloaded SCORM template. The downloaded SCORM module includes a JavaScript help file (API wrapper) that helps in using the SCORM API. The developed Java functions have taken into account the wrapper API to simplify their creation.

Figure 2 represents the structure of SCORM module, which contains the java Applet of the virtual Lab in the *SCO2*. The dashed lines show the interactions between *SCOs* and LMS.

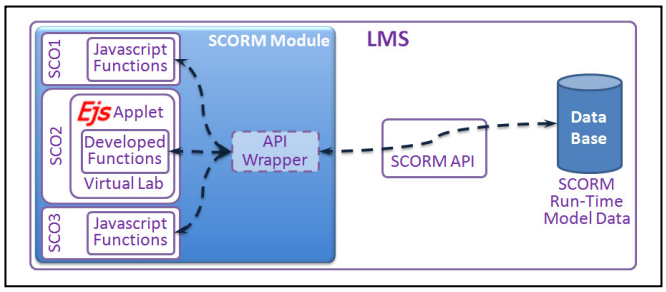

Figure 2. Structure of the SCORM module and *SCO*s-LMS interactions.

As can be seen above, the SCORM package includes three *SCO*. In *SCO1* some basic instructions have been included for the understanding and management of the laboratory. The student should read and understand this content before accessing the virtual laboratory. In the *SCO2* the virtual laboratory is included with practical instructions to be performed. And finally, in the *SCO3*, tests questions have been included in order to assess of acquired knowledge. This structure has been chosen because it is the most appropriate when implementing online teaching laboratory.

Once all the changes in the SCORM template have been done, the structure SCORM is compressed in zip format and it is imported from the ILIAS virtual teaching platform of the UJA into a publicly available folder that offers free access [14].

The functions used in this example to interact with LMS allow the obtaining of the following data from the LMS: the name and identifier of the learner who accessed the SCORM module in the LMS, the API version that is running (used by the LMS), the completion status of the lab (if completed)[1], the laboratory success status (if passed)[1], the achieved score[1] and the comments sent by the student about the lab at LMS[1].

On the other hand other functions have been developed for adding information to the LMS: information regarding the date and time of events, information regarding comments of learners that remain persistently in the LMS (for future enquiries from the online lab in subsequent sessions and attempts), information regarding the score achieved by the student in an attempt to overcome the laboratory (to accomplish an evaluation of it), information regarding the completion of the experiment.

A. *Description of the Lab Exercise*

We used a simple example that shows the work done in a simple way. The developed virtual laboratory shows an operation of a car's suspension, which allows the student to modify their parameters K: spring constant, and B: damping constant. It also allows the user to select different operating scenarios. The system equation is:

$$m\ddot{y} = -mg - K(y - u - L0) - B(\dot{y} - \dot{u})$$

Where "m" is the mass of the body (one quarter of weight of the vehicle), "y" is the distance from the mass to the reference point, "u" is the distance from the road to the reference point, "L0" is the length of the spring at rest and "g" is the acceleration of gravity. The lab applet is initially displayed in a window that is embedded in an HTML page. Figure 3 shows a screenshot of it (applet of the virtual lab in the *SCO2*). Figure 3A shows the main window of the java applet and figure 3B shows the graphics popup windows.

At the top it shows a panel that displays information retrieved by the LMS applet at the time of its release (which can be updated at any time) and an area in which the student that is working on it can save comments he or she deems appropriate in the LMS, to retrieve them when they want to (to do it they must open a pop up window by selecting the check box from the same panel). In the centre there are some moving images that help students to recognize the scene and performance that is achieved in the car's suspension, while at the bottom there are three sub-panels from which can be configured: top sub-panel (Parameters of the different scenes that can be set for the operation of the suspension system), middle sub-panel (Select of the system usage scenes) and bottom sub-panel (Contains buttons for starting and pausing the simulation, a text box to change the configuration constants of the suspension system, and a check box to control the launching of a popup window with graphical evolution of the variables).

V. CONCLUSIONS AND FUTURE ACTIONS

The integration of online laboratories in the LMS is a process which offers many benefits to learners and teachers. These advantages include the ability to offer all the contents in

[1] The laboratory software must store this information in the LMS before

the same environment where access control can be done. Another advantage is that it's also possible to customize the use of laboratories in terms of users, and to program their overcoming and evaluation. All of them are desirable features in the development and integration of VRL.

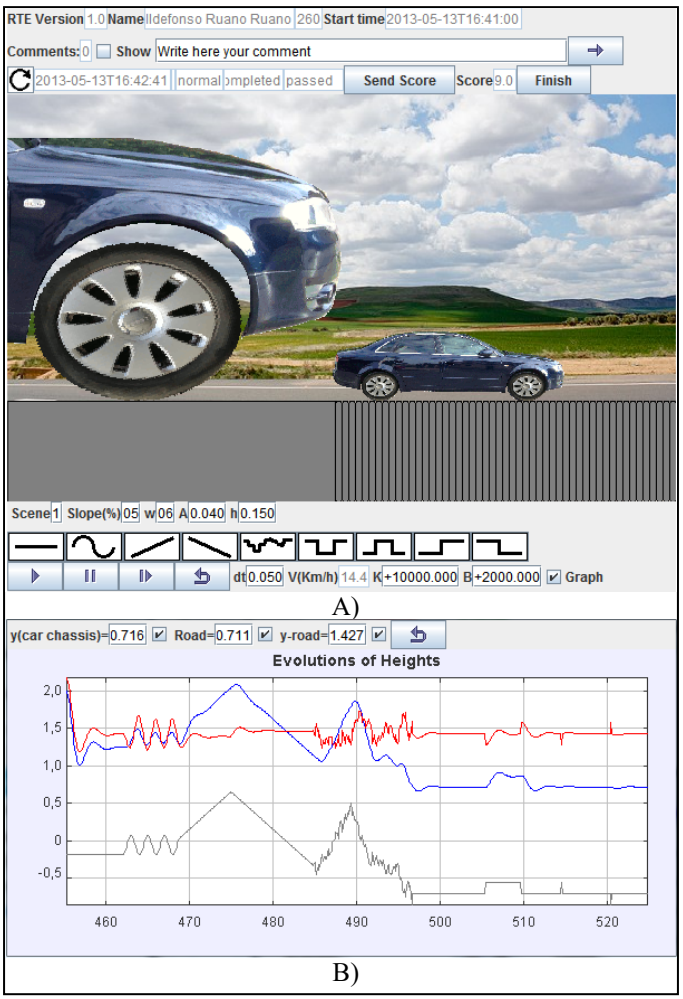

Figure 3. A) Screen capture of the virtual lab EJS applet in the SCORM packet B) Graphics popup window.

In this paper we have provided resources that encourage this integration. The performed functions are useful tools for VRL Java developers. Using these functions, they can create online laboratories that can be virtually integrated with all LMS of the market. This tool facilitates the incorporation of information from the LMS to online lab, enriching it. It also facilitates the learner to save persistent data to the LMS about their laboratory practices that they can retrieve later. This information can help learners overcome the tests which have been provided. Another action that can be programmed is an update of the practices that have been established in the LMS and the score achieved, which will remain in the LMS so that the tutor can check what has been the result of the work done by the learner. The use of these functions facilitates the integration of online laboratories (remote or virtual) that are developed in the Java language by its inclusion as an applet in a SCORM package, which can be used in most of the LMS.

To show the validity of this work, we have developed an EJS virtual laboratory example that integrates with the ILIAS LMS of the UJA and uses some of these functions to show some of its potential. A free access copy of this example can be tested [14]. We are working on developing a Java package that will include objects and methods related with the SCORM RTE. This Java packet will be available for free download.

ACKNOWLEDGMENT

This work was partially supported by the projects DPI2011-27284, TEP2009-5363 and AGR-6429

REFERENCES

[1] L. Feisel, & A. Rosa, “The role of the laboratory in undergraduate engineering education”. Journal of Engineering Education, 94(1), 2005, pp.121-130.

[2] Burgin Aktan, C.A. Bohus, L.A. Crowl, M.H. Shor. "Distance learning applied to control engineering laboratories". IEEE Transactions on Education 39 (3) 1996, pp.320-326, DOI:10.1109/13.538754.

[3] J. Henry, “Controls laboratory teaching via the World Wide Web” ASEE Annual Meeting, Washington, D.C., June, 1996.

[4] S Dormido, H Vargas, J Sánchez, R Dormido, N Duro, F Morilla. "Developing and implementing virtual and remote labs for control education: The UNED pilot experience". 17th IFAC World Congress 2008, pp. 8159-8164

[5] V.J. Harward, J.A. del Alamo, S.R. Lerman, P.H. Bailey, J. Carpenter, K. DeLong, C. Felknor, J. Hardison, B. Harrison, "The iLab shared architecture: A web services infrastructure to build communities of internet accessible laboratories" Proceedings of the IEEE,Vol:96, Issue:6, 2008, pp.931–950, DOI: 10.1109/JPROC.2008.921607

[6] E. San Cristobal, M. Castro, J. Hardward, P. Baley, K. DeLong, J. Hardison, "Integration view of web labs and learning management systems". IEEE EDUCON Education Engineering- The future of Global Learning Engineering Education 81, 2010, pp.1409-1417

[7] Advanced Distributed Learning, “SCORM 2004 4th Edition Specification”. 2012, [Online]. Available: http://www.adlnet.gov/wp-content/uploads/2011/07/SCORM_2004_4ED_v1_1_Doc_Suite.zip

[8] E. Gutiérrez, M.A. Trenas, J. Ramos, F. Corbera, S. Romero, "A new moodle module supporting automatic verification of VHDL-based assignments". Computers & Education 54(2), 2010, pp.562-577

[9] L. de la Torre, J. P. Sánchez, R. Heradio, C. Carreras, M. Yuste, J. Sánchez, S. Dormido, "UNEDLABS: An example of EJS labs integration into moodle", World Conference on Physics Education, Istambul, Turkey, 2012.

[10] V. Mateos, A. Gallardo, T. Richter, "LiLa booking system: Architecture and conceptual model of a rig booking system for on-line laboratories". International Journal of Online Engineering (iJOE),2011, 7 (4)

[11] F. Esquembre, "Using Easy Java Simulations to create scientific simulations in Java". EUROCON 2003. Computer as a Tool. The IEEE Region 8 Vol:1, 2003, pp.:20-23

[12] Advanced Distributed Learning (2013). “SCORM adopters”. [Online]. Available: http://www.adlnet.org/wp-content/uploads/2012/01/SCORMAdoptersLocked.xlsx

[13] Oracle (2013). "Invoking JavaScript code from an applet". [Online]. Available: http://docs.oracle.com/javase/tutorial/deployment/applet/invokingJavaScriptFromApplet.html

[14] I. Ruano-Ruano (2013) “SMC2013, SCORM module with an example of virtual lab integrated with ILIAS LMS”. [Online]. Available: http://dv.ujaen.es/docencia/goto_docencia_sahs_428996.html