Integration View of Web Labs and Learning Management Systems

Elio Sancristobal, Manuel Castro
Electrical and Computer Engineering Department
UNED - Spanish University for Distance Education
Madrid, Spain
elio@ieec.uned.es, mcastro@ieec.uned.es

Judson Harward, Philip Baley, Kimberly DeLong, James Hardison
Center for Educational Computing Initiatives
MIT, Cambridge, Massachusetts, USA
jud@mit.edu, pbailey@mit.edu, kirky@mit.edu, hardison@mit.edu

***Abstract*—The integration of Learning Management Systems and specific learning support applications known as Web Labs (remote and virtual laboratories) are the target of a new wave of service-oriented applications devoted to improving on-line learning experiences.**

Nowadays these solution works in a separet way therefore the students teachers, administration must log in different systems, the are not reusing services, etc.

For these an other reasion in this paper we are focus in two topics. In one hand we describe a technique to present a web lab through a browser delivered by an LMS as a part of SCORM standard packaging. In other hand we describe a service-oriented architecture which allow integrating multiple LMSs (Moodle, .LRN, Claroline, etc.) with iLabs and multiples web an remote labs to supply the full functionality needed by educators

Keywords- e-learning; Learning management system; remote Labs, virtual labs, iLabarchitecture, e-learning standards; Web services.

I. INTRODUCTION

This paper discusses the need for merging several e-learning solutions into one. At present there are a great number of universities that are using blended learning or distance learning in parallel with traditional learning. In the case of distance learning, it is necessary to change and apply distance learning methods so that students achieve both theoretical and practical knowledge. To achieve this double goal, there are two new solutions particularly designed for distance learning [1]:

- A learning management system (LMS) is a software program that enables the display of theoretical content in an organized and controlled way. LMSs offer a set of features and services: user administration, e-learning standards (SCORM, IMS-QTI), content packing, etc.
- A web lab is a program that allows students to execute experiments remotely using a PC and an Internet connection. There are several ways to implement a web lab:
    - Software Lab. They are simulation programs and are executed locally on the student's computer. There is no collaborative work, and students do not work with real instruments or hardware.
    - Virtual Web Lab. These are simulation programs that use web resources. They permit students to collaborate during the execution of experiments.
    - Remote Lab. A remote lab allows the student to manipulate real instruments over the Internet during the run of an experiment.

Many universities are developing their own virtual and remote labs, but these efforts lack a unity of design, involve much custom development and present integration issues. There is little to no reuse of software between these efforts; each is developed from scratch. As one solution for this problem, the Massachusetts Institute of Technology has implemented the iLab Shared Architecture (ISA) [2-3] to facilitate the rapid development of new web labs and to provide a mechanism so that students from one university can use experiments and hardware instruments published from another.

While the ISA solves many problems, it does not offer the standard features supplied by learning management systems (e.g., chat, forums, learning modules). If you want these features in a current iLabs, you must program them into each lab's software. In this article we define a common architecture and middleware for adopting these typical LMS services as e-learning standards. We also illustrate the merger of theoretical and practical learning in a particular solution.

Thus, in this paper we will focus on two topics:

- We describe a technique to present a web lab through a browser delivered by an LMS as a part of SCORM standard packaging. The LMS will provide the web lab's communication, administration, and authentication tools. Of course, the lab can invoke the SCORM API so that the teacher can monitor the student's progress. When implemented as SCORM standard packages, web labs can be deployed in different LMSs such as Moodle, .LRN, Claroline, Sakai, etc.
- While the ISA provides an excellent management infrastructure for online labs, we argue that we need a service-oriented fusion of this architecture with general LMS services, compatible with multiple LMSs (Moodle, .LRN, Claroline, etc.) to supply the full functionality needed by educators.

978-1-4244-6571-2/10/$26.00 ©2010 IEEE

II. LEARNING MANAGEMENT SYSTEMS (LMS)

A LMS is a software program that allows displaying theoretical content in an organized and controlled way. To do this, the most of LMS are designed using a common architecture, Fig. 1. that allows adding, deleting or modifying new functionalities.

The main elements in this architecture are:

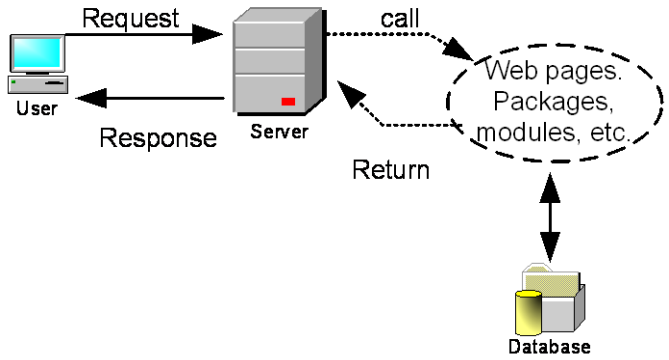

Figure 1. Architecture of a LMS.

- Database. This stores the information that the services are going to need and the information that will be displayed to user (administrators, teachers, students, etc.). Depend on the LMS that you are going to install and use, you will be able to work with mysql, oracle, postgres, etc.
- Modules, blocks, packages, etc. The structure of this modules or packages depends on the LMS. So how the programming language that you should use. For instance, if you are working in Moodle [4] you must use PHP to program. To sum up, these modules contain the logical of LMS services

  It is very important to mention that there are open source LMS as Moodle, DotLRN [5], Sakai [6], Claroline [7], etc. So that, a programmer could add new modules or modify standard services that are include in the standard installation of LMS.
- Web Server. All the LMS are installed along with a web server. This allows responding the user requests though Internet. Also, at the same that the others elements mentioned above, depending on the LMS with you are working, you have to install TOMCAT, APACHE, etc.

Once, we have known how a basic LMS architecture is. We are going to enumerate several of most important features and services that a LMS offers:

- Administration. It must be able to manage user registrations, roles, assign tutors, user payments, etc
- Content packing. It organizes the content in a hierarchical structure and sets up a mechanism to swap content between different learning management systems. To do it, it's usually used the specification IMS content packaging or the specification SCORM (Shareable Courseware Object Reference Model).
- Synchronous and asynchronous Communication Tools. It must allow collaborative work. So that they can share information, opinions and experiences.
- Knowledge evaluation. The tutors and teachers must be able to evaluate the student's progress. Also the students can do test where they can see their progress. To do it, it's possible to use the specification IMS QTI (Question and Test Interoperability).
- Tracking user. This feature should provide information user with teacher about that difficulties and problems have been found in the course by the students for the course term.

So, we have a tool that offers a set of features and services to display theoretical content in an organized and controlled way. As well as using e-learning standards like SCORM, IMS-QTI, etc.

III. SOFTWARE, WEB, REMOTE LAB AND ILAB

In many distance learning o blended learning courses, besides displaying theoretical knowledge by LMSs, is needed that the students acquire skills and practical knowledge. This and other reasons (the students are be able to carry out their experiment 24 hour by day and 365 day in a year, etc.) have given risen to design and create software, web and remote labs [8].

In this section we are going a brief description each one of these labs and one solution called iLab that was developed by the Massachusetts Institute of Technology to facilitate the rapid development of new web labs and to provide a mechanism so that students from one university can use experiments and hardware instruments published from another.

- Software Labs

  They are based on software programs that are being executed in the student's computer, Fig. 2.

  The student's computer must have the hardware and software requirements and an Internet connection it is not required.

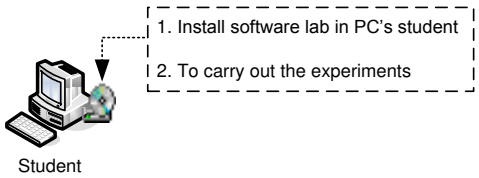

Figure 2. Software Lab.

Some of the main problems of theses labs are:

- Lack of collaborative tools
- Version problems, students have a lot of version of same labs. To solve it, the software labs allow the students to update the software using an Internet connection, Fig. 3.

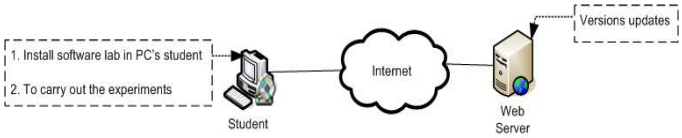

Figure 3. Software Lab with Internet Connection.

- Web Lab. These are simulation programs that use web resources. They permit students to collaborate during the execution of experiments, Fig. 4.

  The main problem is that students don´t manipulate real instruments to carry out his experiments.

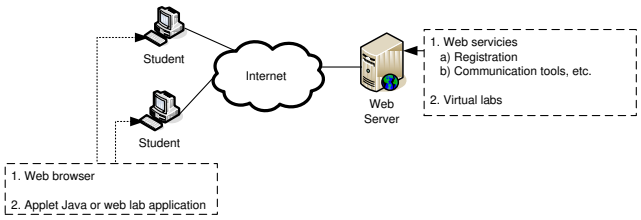

Figure 4. Web Lab.

- Remote Lab. These are simulation programs that use web resources. They permit students to collaborate during the execution of experiments, Fig 5.

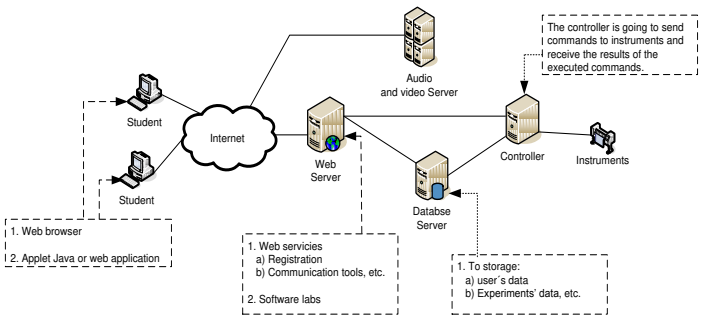

Figure 5. Web Lab.

Once we have described briefly what are software, web and remote labs. We are going to talk about the main problem that appeared. Many universities and organizations started developing their own virtual and remote labs, but these efforts lack a unity of design, involve much custom development and present integration issues. There is little to no reuse of software between these efforts; each is developed from scratch. For this reason, the Massachusetts Institute of Technology implemented iLab Shared Architecture (ISA) to facilitate the rapid development of new web labs and to provide a mechanism so that students from one university can use experiments and hardware instruments published from another.

To do this MIT divide the experiments according to the type interaction between user and lab. As a result of this, MIT has designed two architectures:

- Architecture based on batched experiments, Fig. 6.

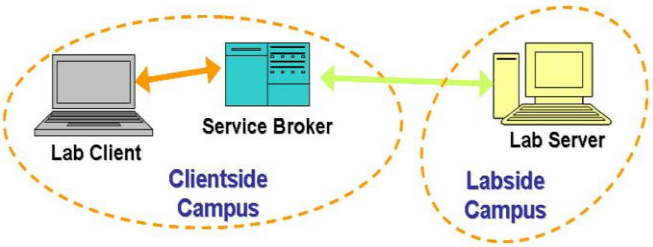

Figure 6. Topology of a batched experiment based on the iLab shared.

- Architecture based on interactive experiments where user and labs must establish a direct communication, Fig. 7.

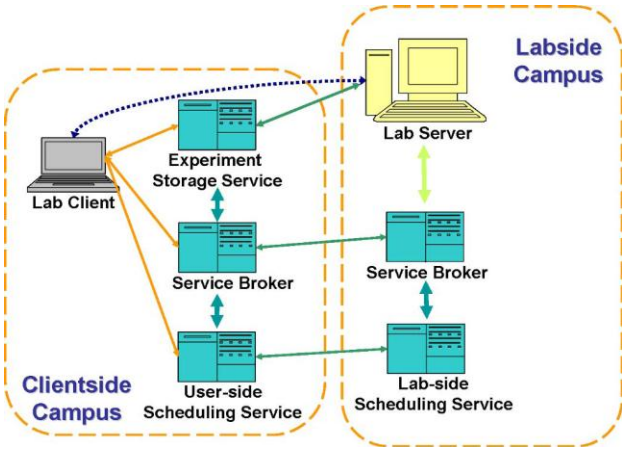

Figure 7. Topology of a interactive experiment based on the iLab shared.

So, ISA supports batch and interactive experiments. But, as well as, ISA allows being installed in every university or organization and therefore a student of one university could log in the system and using the web and remote labs from other universities.

While the ISA solves many problems, it does not offer the standard features supplied by learning management systems (e.g., chat, forums, learning modules). If you want these features in a current iLabs, you must program them into each lab's software.

So, imaging two universities, both of them want to create the same electronic remote laboratory and as well as theses universities want the laboratory provide one set of features, as: authentication, forums, chat, etc. To do this both of these universities must facing with the following challenges and disadvantages:

- Everyone should:
  - Defining and design one architecture. And probably the chosen architecture will be different.

- Programming and implementing the lab. Depend on the chosen architecture will be used different programming languages.

This provoked that the user from different universities needed different ways to log in and work with the system and therefore it is impossible to share labs. Due to this, appears ISA.

- Also, everyone should create the services that the laboratory needs as: forums, chats, storage area, etc. This is a problem, because by every lab we have to create, we must programme the same services one and other time. And therefore , the universities are wasting their time, personal and efforts in doing the same thing all the time, Fig 8.

  To solve this and other problems as the integrating between both solutions (LMS and web, remote labs) we are defining and developing a common architecture and middleware for that the labs could use the LMS services and e-learning standards, Fig 9. And therefore, reuse services.

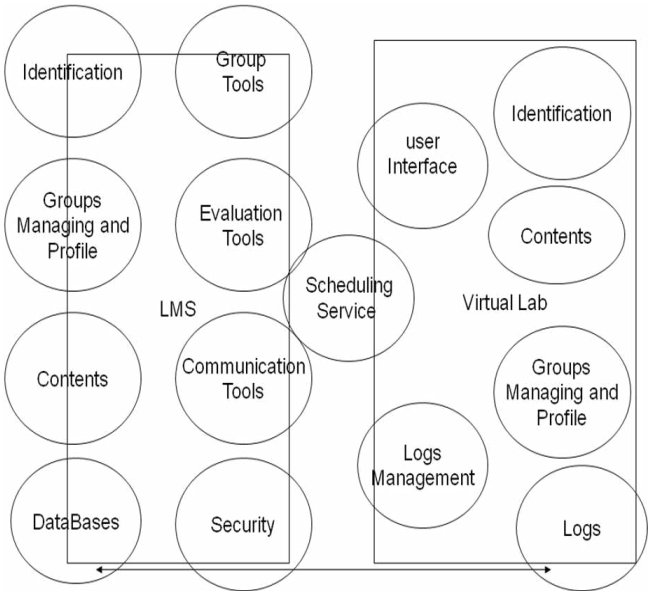

Figure 8. Duplicating services.

In the next sections also illustrate the merger of theoretical and practical learning in a particular solution. Thus, in this paper we will focus on two topics:

- We describe a technique to present a web lab through a browser delivered by an LMS as a part of SCORM standard packaging. The LMS will provide the web lab's communication, administration, and authentication tools. Of course, the lab can invoke the SCORM API so that the teacher can monitor the student's progress. When implemented as SCORM standard packages, web labs can be deployed in different LMSs such as Moodle, .LRN, Claroline, Sakai, etc.
- While the ISA provides an excellent management infrastructure for online labs, we argue that we need a service-oriented fusion of this architecture with general LMS services, compatible with multiple LMSs (Moodle, .LRN, Claroline, etc.) to supply the full functionality needed by educators.

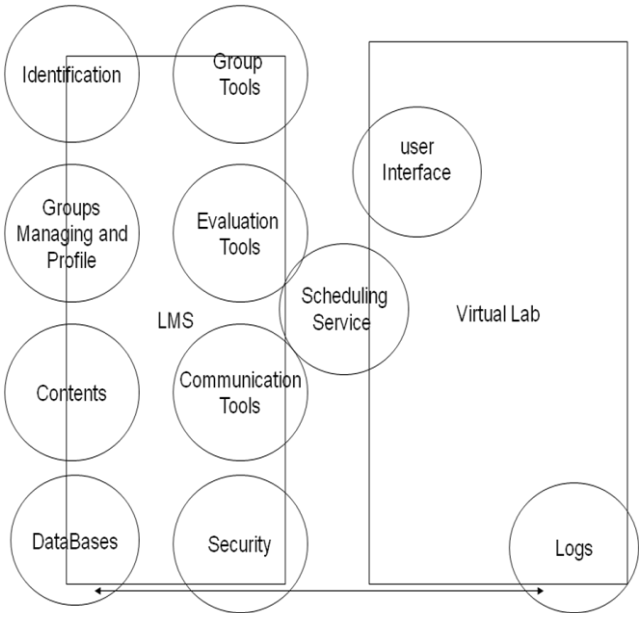

Figure 9. Reusing services.

IV. SCORM AND LMS

A Great number of Learning Management Systems support several e-learning standards as: IMS_QTI, IMS-LD, SCORM, etc. Every one of these has a different utility, for instance IMS-QTI is a specification for a metalanguage which enables the modeling of learning processes. In this section we are going to focus on e-learning standard called Sharable Content Object Reference Model (SCORM) [9-10].

SCORM content packaging provides a consistent form for describing content structures, learning content, the metadata that describes the various components of the content structures and sequencing and navigation rules.

This facilitates searching and discovering content packages and their resources.

How it is mentioned above SCORM documentation version 2004 describes a set of features of a content package as:

- Content Packaging: describes the SCORM components used to build a learning experience from learning resources. So, a package SCORM is composed of assets, sharable content objects (SCOs), activities, a content organization and content aggregations, Fig. 10.

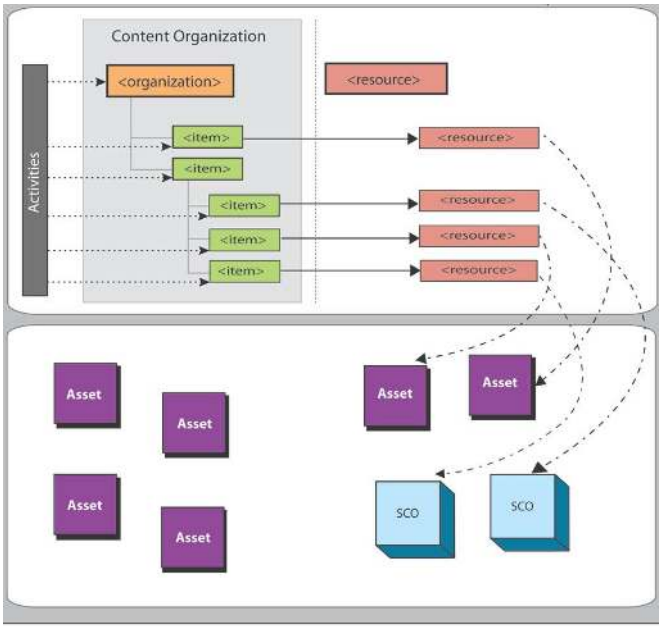

Figure 10. Content aggregation.

- Assets are an electronic representation of media, such as text, images, sound, assessment objects or any other piece of data that can be rendered by a Web client and presented to a learner.
- A SCO is a collection of one or more assets. The only difference between a SCO and an asset is that the SCO communicates with an LMS using the Institute for Electrical and Electronics Engineers (IEEE) ECMAScript.
- A learning activity may provide a learning resource (SCO or asset) to the learner or it may be composed of several sub-activities.
- A content organization is a representation or map that defines the intended use of the content through structured units of instruction (activities).
- Content aggregation can be used to describe the action or process of composing a set of functionally related content objects so that the set can be applied in a learning experience.

These packages are saved in a zip file. This file contains among other files, a description of package in XML, named imsmanifest.xml and the physical file of resources called

- The SCORM Run-Time Environment, Fig. 11. In this "book" is described:
  - Content launch process. LMS must load the SCORM when the user perform a request.
  - Standardized communication between content and LMSs. To do this, it uses an API
  - Standardized data model elements used for passing information relevant to the learner's experience with the content.

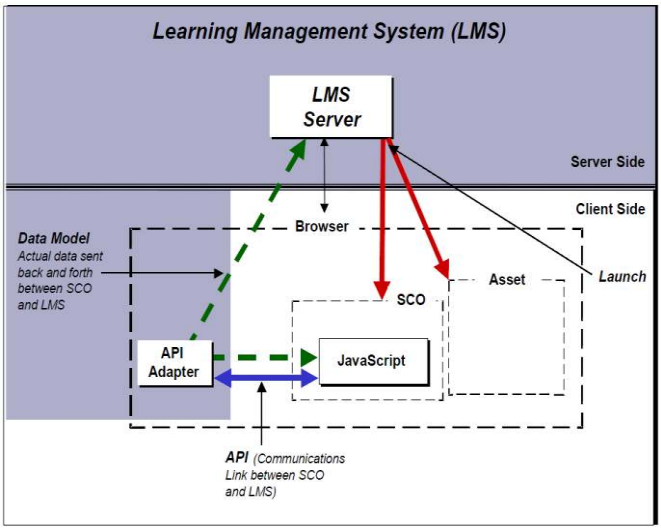

Figure 11. Scorm Run-time eviroment.

- Sequencing and Navigation: Descriptions and requirements for defining sequencing and navigation information.

  Nowadays there are several versions of SCORM v1.1, 1.2, 2004 and not all this version supports sequencing and navigation.

So, if a teacher or e-learning designer creates a SCORM package using these specifications then he could install it in all the LMS that support that SCORM version, reusing content and services.

V. PACKING WEB LAB IN SCORM

In the previous section we have read how a SCORM is composed and how we can communicate this SCORM with the LMS using an API. Now we are going to explain how we could use a web lab into a SCORM and therefore how we could reusing the LMS services and install this package in different LMS that are SCORM compliant.

Nowadays you don´t need program to create a SCORM package, only if you want to use the API for communicating LMS and SCORM. In this case you have to include in SCOs one o several javascript file, for instance:

- APIWrapper.js whose purpose is in wrapping the calls to the API is to provide a consistent means of finding the LMS API implementation within the window hierarchy and to validate that the data being exchanged via the API conforms to the defined CMI data types.
- SCOFunctions.js contains functions encapsulate actions that are taken when the user navigates between SCOs, or exits the Lesson.

Once you include these files in an html page or other SCO. You could use the functions that they have implemented to communicate with the API implemented in the LMS, Fig. 12.

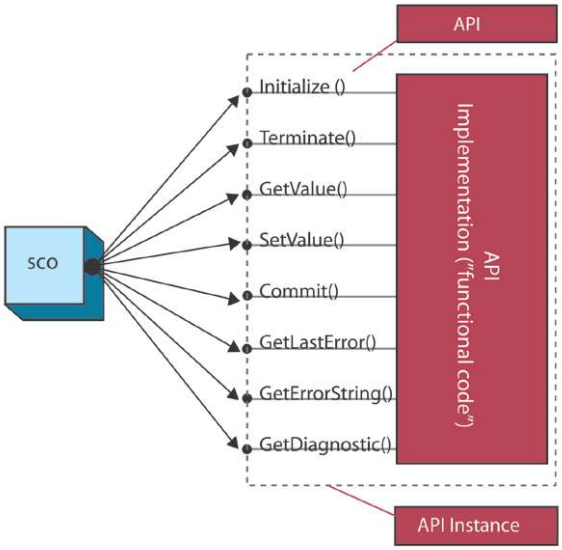

Figure 12. API.

It is very important to tell that when you create a web o remote labs you can use programming languages that don´t allow embedding JavaScript code, in this case we can integrate this lab in and *iframe* of a web page and in this page adds API functions to establish a communication with the LMS. The main problem is that the lab couldn´t communicate directly with the LMS. At this moment there are several projects to implement this API with web services.

So, the first thing that we have to do is to collect the resources that are going to composed the SCORM (html pages, images, etc.) and include the javascript files, that implement the API functions such as initializing the connection, exchanging information between SCO and LMS and finishing the connection, in the SCO that we want to communicate with the LMS. Later we have to establish an organization of every one of the resources to compose aggregation content.

Nowadays there is a great number of tools (free or commercial) allowing creating SCORM packages in a graphical way. So the teacher o learning designer, who don´t know about XML, can create their own SCORM packages without problems. Some of these free tools are:

- Reload, Reusable eLearning Object Authoring and Delivery.
- Couselab, is a free program but is not open source.
- eXe OPEN SOURCE SCORM Development Package

For this small example of weblab SCORM we have used Reload [11] and have include a web page named pag1.htm that include an *iframe* whit a URL of web lab in, Fig. 13. Of course you can include a web page hierarchy where you explain how the web lab works, the experiments that user can carry out, etc.

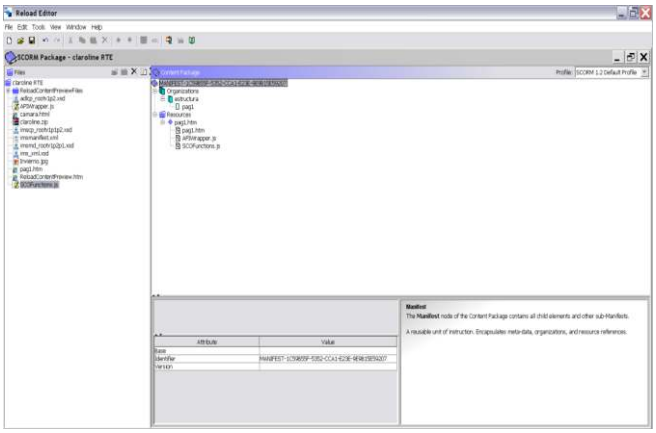

Figure 13. SCORM package created with Reload.

Once you save it, you could install this package in every e-learning platform that supports the same SCORM version that you have created. In the figure 14, we can see the result of installing this SCORM in Moodle.

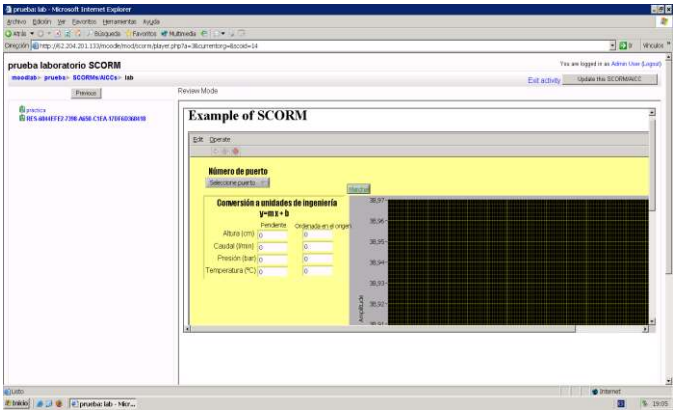

Figure 14. SCORM installed in Moodle.

Besides of creating an SCORM with a weblab, information about the experiment, etc and can use it in different LMSs (reusing). Also the user could use the LMS services as: Chats, Forums, Storage area, etc without having to be written a code line by lab programmer.

We have mentioned the problem in which code of web lab doesn´t allow embedding JavaScript and therefore there is no direct communication between web lab and LMS. To solve this and other problem as:

- What happen if the URL of web lab changes
- If we have several identical web labs and one this less busy than other (load balancing)
- If we simply want that web lab send information in a directly way
- Etc.

To solve this and other problems, we are working in a middleware for merging and managing both LMSs and web, remote labs in one.

VI. NEW MIDDLEWARE

In his section we are going to focus on a middleware and architecture to manage and integrate both solutions in one, Fig 15.

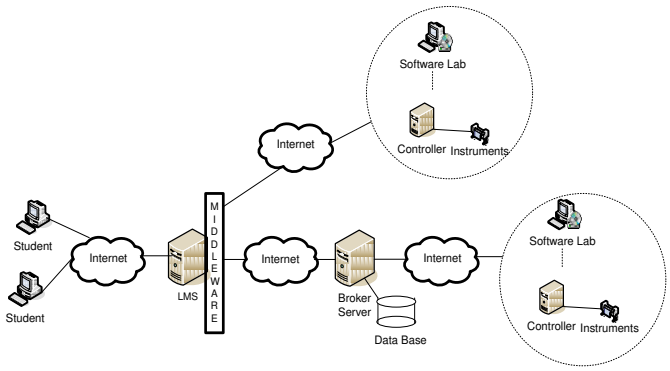

Figure 15. Integration LMS, iLabs and Virtual Labs.

To design this architecture we have to consider several ideas that have been mentioned in the previous sections:

1. The most of the LMS is composed by data base, a logical programming structure (modules, blocks, packages, etc.) and Web servers. So, we can design a module or package that use a database where is stored information about the laboratories, experiments, the pair lab-course, etc.

   Therefore if we design a module in an open source LMS as Moodle, this module could be installed in every server that contains an instance of Moodle. And the same if we create a package in .LRN or in other open source LMS as claroline, sakai, etc.
2. The iLab Shared Architecture (ISA) to facilitate the rapid development of new web labs and to provide a mechanism so that students from one university can use experiments and hardware instruments published from another.
3. There are other laboratories that only need be connected with a URL, with some information through of URL or other type of connection.
4. Both LMS as Web and remote labs need be displayed through Internet and by the way they use Web servers.

If we consider these ideas, the first thing that we should decide was what type of communication through Internet allows establishing a good communication among different systems and as well as provide several features as: scalability, loose coupling, etc. One of the best solutions that fulfil with these requirements is Service Oriented Architecture (SOA), Fig. 16.

As you can see in the figure 16, SOA [12-15]is based on services, to avoid the duplicating the services, the services providers publish information about the web service. The web service clients can search for the web services directory, if there are any web services that carry out the actions that the client need. If it is found the client bin and invoke the web service that is located in the provider.

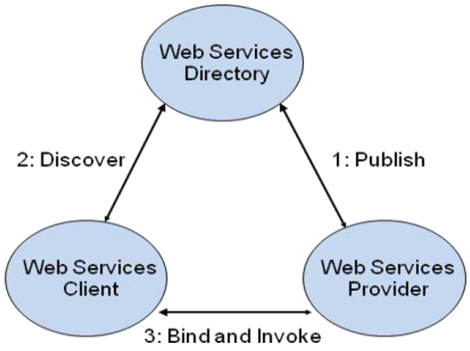

Figure 16. SOA.

All this process used several standards as:

- Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.
- Discovery and Integration (UDDI) is a directory service where providers can register and clients search for Web services.
- Simple Object Access Protocol (SOAP) is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.

One part of SOA is the infrastructure that allows you to use services in a productive system. This is usually called the enterprise service bus (ESB), Fig. 17. The responsibilities of ESB involve:

- Providing connectivity
- Data transformation
- (intelligent) routing
- Dealing with security
- Dealing with reliability
- Service management
- Monitoring and logging

Therefore, this middleware establishes a way to communicate heterogeneous systems (such as old systems and new systems), provides a set of features as routing, security, etc. And it is based on Standards as WSDL [16], UDDI [17], SOAP [18], etc.

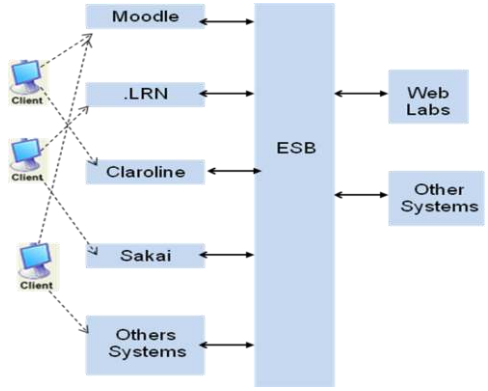

Figure 17. ESB.

VII. LMS (MODULE, PACKAGES BLOCKS, ETC.)

One of the first steps to define and design the architecture mentioned in the last section is to create LMS architecture. So, we have read that an open source LMS is composed by database, a logical programming structure (packages, modules, blocks, etc.) and web server. If we communicate and programme this element we can create an architecture based on services, Fig. 18.

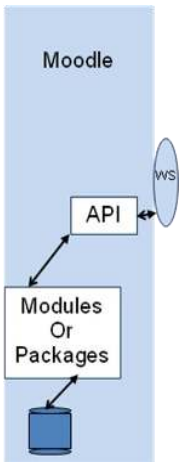

Figure 18. Middleware to connect a client with open source LMS as Moodle

At this point, we are talking about the creating of one .LRN package and a Moodle module to connect web and remote labs.

A. *.LRN package:*

We have defined and programmed a package for every .LRN administrator can create an area where connect a web or remote labs. To do this area we have created a package with the followed elements:

- A set of table associates to .LRN database (oracle or postgres) to store the created laboratory, the experiments that you are going to do, the way of connection between .LRN and web or remote lab, etc.
- A Logical programming (using Tool Command Language TCL) to exchange information between user and LMS, etc.
- And a user interface (ADP or HTML files).

Once the package has been installed in the .LRN the administrator could create all the areas that we need. This area is composed by, Fig. 19:

- A navigation menu where we can find services as:
  - Calendar
  - Asynchronous communication (forums, etc.)
  - Synchronous communication (chats, etc.)
  - Experiment area where are stored user manual, texts about the experiment, etc.
  - And area to display the remote labs and where the students can be work with it.

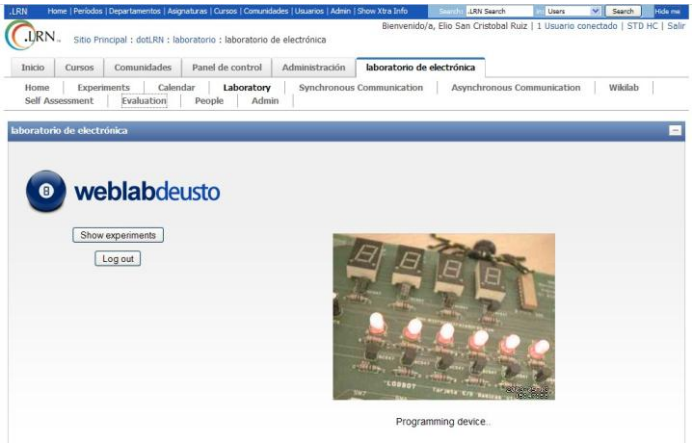

Figure 19. .LRN package.

B. *Moodle Module:*

To create a lab module in its first version we have carry out the next step:

1- We have created a set of table and have related them with Moodle database. To create this table we have used a XML file called intall.xml that uses Moddle to allow generating these tables with independence of the type of database that you install in Moodle (mysql, oracle, etc.).

2- We also modify the administration site for allowing that the administrator is able to add, modify and delete the labs or server broker (in case of iLab) and the way of connection of these, Fig. 20.

3- Later we have created te PHP file that provide of tle logical of module, as:

   a. Add a laboratory to a course.
   b. To Call connection services
   c. To store information in the tables of module
   d. Etc.

As a result of this a teacher could add every lab which has been inserted, using Administration site, in a course, Fig 21.

If we click on the created link then the students log in the web lab directly. Of course this is a first version so that we connect with web labs that require the username and password through URL. We are working in designing and implementation the web services into LMS.

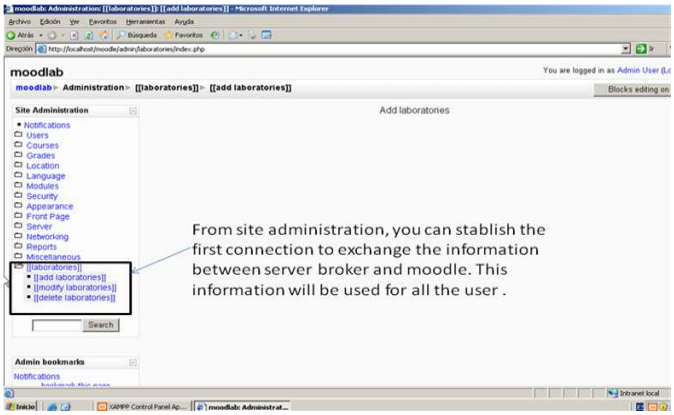

Figure 20. .Adding option to Administration site of Moodle.

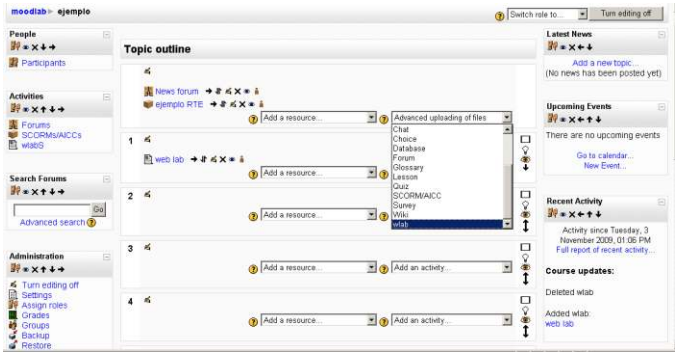

Figure 21. .Adding a web lab in a Moodle course.

VIII. FUTURE WORKS

This is a strong first step to connect LMS with iLabs and remote labs, and obtain shared labs, reuse of services and merge of these two solutions. But, of course we have to work in several important aspects:

- Design and implement web services in the open source LMS as Moodle or .LRN. For example scheduling service.
- Allowing LMSs and iLabs and web labs to use a single sign on
- Designing and implementing a enterprise service bus and the connector that systems need.
- Establish a common way to connect LMSs and iLabs and remote labs.
- Also we would like to work in the idea to describe web labs, so we can define an standard like WSDL or UDDI to search for web labs and bind with provider.

IX. CONCLUSIONS

We are working in a middleware and architecture that allow connect different systems and at the same time offer:

- Data transformation
- (intelligent) routing
- Dealing with security
- Dealing with reliability
- Service management
- Monitoring and logging

X. 8 ACKNOWLEDGEMENT

The authors would like to acknowledge to the Spanish Science and Innovation Ministry for the support of the project TIN2008-06083-C03/TSI "s-Labs – Integración de Servicios Abiertos para Laboratorios Remotos y Virtuales Distribuidos"

REFERENCES

[1] E. Sancristobal, S. Martín, R. Gil, E. López, G. Díaz, E. Ruiz, M. Castro, and J. Peire, Integrating and Reusing OF Virtual Labs in Open Source LMS. REV 2008 International Conference Dusseldorf.

[2] J. L. Hardison, K. DeLong, P. H. Bailey, and V. J. Harward. Deploying Interactive Remote Labs Using the iLab Shared Architecture. FIE 2008.

[3] iLab, MIT http://icampus.mit.edu/iLabs/ (November, 2009).

[4] Moodle http://moodle.org/development/ (November, 2009).

[5] DotLRN http://dotlrn.org/ (November, 2009).

[6] Sakai http://sakaiproject.org/ (November, 2009).

[7] Claroline http://www.claroline.net/ (November, 2009).

[8] C.C. Ko Chen, Creating Web-based Laboratories. Ed. Springer, 2004.

[9] http://www.adlnet.org/Pages/Default.aspx (November, 2009) .

[10] http://www.adlnet.gov/Technologies/scorm/SCORMSDocuments/2004%204th%20Edition/Documentation.aspx (November, 2009).

[11] http://www.reload.ac.uk/ (November, 2009).

[12] D. Nickull; D. Hinchcliffe; J. Governor, Web 2.0 Architectures, 1st Edition O'Reilly Media, Inc., 2009.

[13] N. M. Josuttis. SOA in Practice: The Art of Distributed System Design (Theory in Practice). O'Reilly, 2007.

[14] D. A Chappell. Enterprise Service Bus. O'Reilly MediaReleased, June 2004.

[15] T. Erl. Service-Oriented Architecture (SOA):Concepts, Techno-logy , and Design. Pearsdon 2005.

[16] http://www.w3.org/TR/wsdl (November, 2009).

[17] http://www.oasis-open.org/committees/uddi-spec/faq.php (November, 2009).

[18] http://www.w3.org/TR/soap/ (November, 2009).