LiLa Booking System: Architecture and Conceptual Model of a Rig Booking System for On-Line Laboratories

doi:10.3991/ijxx.vxnx.xxx (Please do not delete this line)

V. Mateos[1], A. Gallardo[2], T. Richter[2], L. Bellido[1], P. Debicki and Víctor Villagrá[1]
[1] Technical University of Madrid, Madrid, Spain
[2] University of Stuttgart, Stuttgart, Germany

Abstract—**Many educational institutions acknowledged the importance of providing online-access to student laboratories. To optimize the use of the scarce and expensive resources such laboratories depend on, it is advisable to establish and setup a booking system that helps to administer access to them.**

This paper reports on the architecture and conceptual model of the rig booking system designed for the LiLa Portal, a web portal that makes virtual and remote experiments available on the Internet. The design of the booking system is based on a requirements analysis carried out by the EC funded LiLa project in cooperation with international partners from the Global Online Lab Consortium, GoLC.

Index Terms—**Remote Experiments; Reservation System; design for experiments**

I. INTRODUCTION

A. LiLa

In the context of the project LiLa ("Library of Labs"), many virtual and remote experiments of the contributing partners spread out over Europe are made available through Learning Management Systems (LMS) and an internet portal, the so-called "LiLa Portal." The portal will also collect feedback from the students to check their knowledge and learning success, and to check whether the experiments fit their needs. [1]

From a technical point of view, the LiLa Portal is a repository of virtual and remote experiments, available as electronic content packages that can be accessed and executed from the Internet requiring a Web browser only. These content packages can also be downloaded and reused in other LMSs such as moodle or Ilias.

Remote experiments make use of limited resources, and pose the challenge of controlling the access to them to avoid conflicts and to maximize their availability.

B. LiLa Booking System

The LiLa Booking System provides the functionality to schedule the access to the remote experiments—or more accurately, to their rigs, as we will see later—with the aim to accommodate as many students as possible and to help them organize their activities in the portal.

C. Document structure

This document builds upon [2] but aims to be self contained, thus some sections from the original paper have been quoted here.

The rest of this document is structured as follows: In section II, we give an overview on related work in the area of experiments booking systems. In section III, we introduce the conceptual model for LiLa and its booking system. In section IV, we provide technical details about the system architecture and its components. In section V, we describe how each user will typically use the system depending on her role. Section VI finally concludes and proposes future work in this area.

II. RELATED WORK

The concept of a booking system for remote laboratories is not new. Other projects have designed and implemented already some solutions that address the contended access to share resources. Some of these solutions are extensions to the functionality of selected LMSs and cannot be reused independently from them – hence, they are not suitable for LiLa purposes. For example, the MARVEL project [3] and its successor EDIPE [4] include a booking system for moodle [5]; WebLab [6] of the Slovak University of Technology in Bratislava is also a module for moodle; and ReLEEP, developed by the Qatar University also uses a moodle specific solution extending the "Meeting Room Booking System" (MRBS) [7] to accommodate their needs.

Other solutions build on proprietary software, such as iLabs [8], which is open source itself, but requires a Windows Server machine and uses the Microsoft SQL Server and Visual Studio.NET; or NETLab [9], developed by the Indian Institute of Technology, Kharagpur, that relies on proprietary software licensed products from Oracle, National Instruments and Agilent. In [10], the authors present a "Web Service-based MetaScheduling Service" to coordinate the allocation of time-slots with local resources in Grid-Computing environments developed in the context of the VIOLA project [11]. Solutions that are based on proprietary software are, however, unsuitable for LiLa purposes as well as they conflict with our license policy.

There is even an alternative approach adopted by the Kumamoto University, Japan, consisting of a time scheduling system compatible with various LMSs that focuses on the generation of courses timetables [12].

All these solutions are not easy to adapt, or cannot be reused at all, in the context of the LiLa project where the distinction of user roles, to be introduced below, is important, open source is a must, and re-usability and flexibility one of its main goals.

III. CONCEPTUAL MODEL

Due to the international nature of the project, the LiLa Portal and its booking system (from now on, "LiLa") have been designed to minimize the overhead of administering its users and to maximize the student access to the experiments. With this in mind, LiLa is built upon three different portal user roles – content providers, teachers and students – and five key elements: experiments, rigs, reservation for teachers, reservation for students and tickets.

This separation of roles and the support of Shibboleth® [13] (a federative authentication and authorization mechanism), saves us from the need of a central administrator (that traditionally had to assign rights to each user) and allows for portability of experiments to external LMSs.

The following sections go into these concepts in greater depth.

A. Experiments

Within LiLa, experiments are software that can be accessed via the LiLa Portal and can be executed in a Web browser. We classify experiments into virtual and remote experiments, the former not requiring additional hardware (for these, there is typically no need of a booking system), and the later accessing limited remote hardware.

B. Rigs

Rigs are the remote hardware that remote experiments operate on: the scare resources, the expensive physical set-ups, the bulky and noisy machines, the fragile robots...

Several experiments could operate on the same rig, but the reciprocal, one experiment requires more than one rig, has not been considered because it is a relatively rare case in education and we did not want to increase the overall complexity of the system unnecessarily.

To allow this flexibility, the LiLa Booking System has been designed to administer rigs, instead of experiments. If an experiment requires a rig, this dependency must be provided when uploading the experiment into the LiLa Portal.

LiLa requires for every specific hardware setup that is to be administered a rig name and a description. The rig name becomes its identifier within LiLa. It is the user's responsibility to assign meaningful names for the rigs and to avoid names collisions.

C. Roles

Before defining reservations and tickets, it is necessary to dig into the LiLa defined user roles. These user roles are not a new concept in reservations systems. We can find them also, for example, in some airline's computerized reservation systems [14]: the content providers in Lila act as vendors who provide their services; the teachers act as representatives for the vendors; and the students act as costumers.

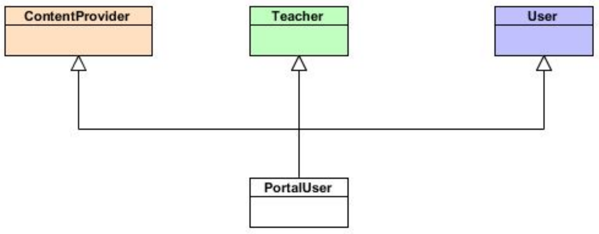


Figure 1. A LiLa Portal user can play any of these roles. In this figure, "user" stands for "student" (a "student" is an "experiment user").

Content Providers are users of the Lila Portal authenticated by the Shibboleth IdP of a LiLa-friendly institution—having a trusted Shibboleth account is important for auditing reasons—who have uploaded at least one experiment to the portal. They must indicate when an experiment makes use of a rig to be able to administer its availability.

Content Providers do not need to care about fine-grained time-slots distribution among students in the same way as the airlines vendors do not care who a travel agency's representative sells a ticket to. Content providers create "*reservations for teachers*" (each with an associated reservation code) to delegate the administration of the experiments during defined periods to teachers.

Teachers are users of LiLa, also authenticated by a Shibboleth IdP, that "hold" one or more "*reservations for teachers*"—hold in the sense that they know its reservation code. They have rights to create "*reservations for students*" (each with a corresponding reservation code) that further refine when the experiments will be available for their students.

Teachers have the main task of finding and getting learning resources for their students where such learning resources are here understood to be the experiments made available by LiLa.

Students are users of LiLa experiments (hosted in LiLa or external LMSs). To run an experiment that requires booking, a student must have a named account—anonymous users are not allowed to use these experiments—and must book a time-slot for its rig.

D. Reservations

To limit access to the experiments, the Booking System relies on the administrative elements "reservations for teachers," "reservations for students" and "tickets," depending on whose privileges they restrict.

Reservations for teachers are administrative elements protected with a secret "reservation code" that content providers create to allocate rigs during selected time-slots for teachers. (See Figure 2.)

If teachers want their students to run selected remote experiments, they must negotiate with the content provider which time-slots and under which conditions the corresponding rigs should be blocked.

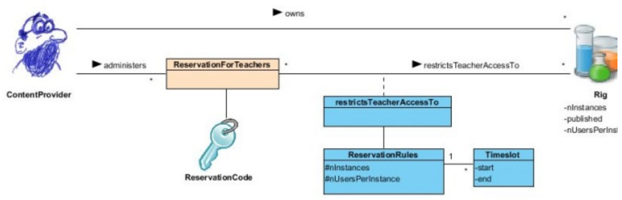


Figure 2. The content provider creates and administers reservations for teachers.

Reservations for students are the administrative elements protected with a secret "reservation code" that the teachers create to allocate rig time-slots for students. (See Figure 3.)

If a teacher that already has a "reservation for teachers" want their students to use a corresponding experiment, they must further refine the time-slots blocked by the tcontent provider for teachers' exclusive use.

This mechanism allows easy groups administration. For example, if a teacher wants to create two different groups of students of a selected rig, one using it on Monday and the other using it on Tuesday, the teacher would create two "reservations for students" with two different reservation codes; and then distribute one code to one of the groups and the other to the other group. Distribution of such reservation codes happens outside of the system, e.g. by writing them on a blackboard.

E. Tickets

Tickets allow students to make reservations within time-slots reserved for their student group, and hence to retrieve *stamped tickets* from the booking system. The system creates a ticket for a rig when the student enters a valid reservation code. The ticket thus consists of the tuple *(student identifier, reservation code)*. Once holding a valid ticked, the LiLa Booking System will provide her with the list of time-slots during which the rig is still available for use. When the student selects one time-slot, the LiLa Booking System creates a "stamped ticket" for the student, i.e. adds the time-slot as third object to the tuple.

Stamped tickets, finally, grant students access to rigs in the time-slot selected by the student. (See Figure 4.).

Even though students never physically *see* this ticket, it is still a system resource administrated internally by the booking server. Very much like a bus ticket, a ticket obtained by a student is valid for a single ride, which is redeemed by entering the bus – here selecting a time-slot for the experiment. It is then invalid for a second ride – here reserving a second time-slot. Only until after performing the experiment, the student is allowed to re-use this ticket.

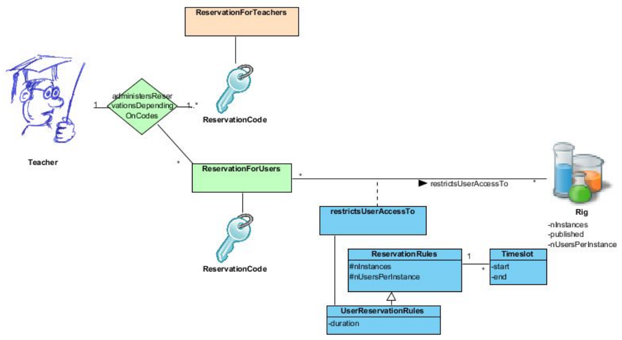


Figure 3. The teacher creates reservation for students (in this figure, "reservation for users") with a protecting reservation code to allocate resources for the students.

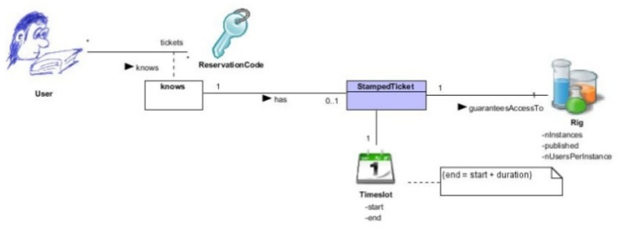


Figure 4. The student ("User" in this figure) must enter a student reservation code to get a ticket, and can get a stamped ticket that grants her access to a rig during a time-slot.

IV. TECHNICAL PERSPECTIVE

A. SCORM, SCOs and LLOs

LiLa strives for maximal portability and reusability by conforming to existing standards whenever possible. The means to make experiments available in LiLa is based upon the ADL's reference model "SCORM" [15]. In this model, the smallest runnable unit is called SCO ("Sharable Content Object"):

"A SCO is a collection of one or more assets [electronic resources that can be rendered by a Web client] that represent a single launchable learning resource […]. A SCO represents the lowest level of granularity of a learning resource that is tracked by an LMS […]." [16], pp. CAD-2-4.

For the physical exchange of learning content, SCORM defines the concept of "content package" that strictly adheres to the IMS Content Packaging Specification. A content package contains two major components: an XML document describing structure and resources of the

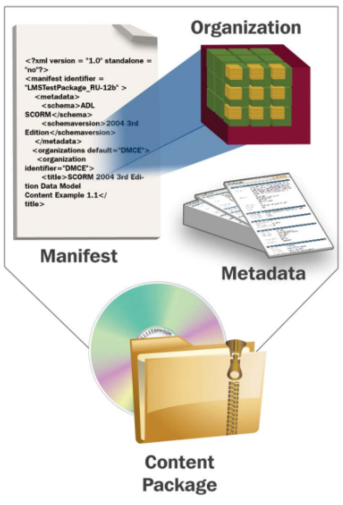
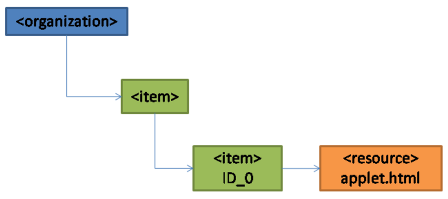

```
<?xml version = "1.0" standalone =
"no"?>
<manifest identifier =
"LMSTestPackage_RU-12b" >
   <metadata>
      <schema>ADL
SCORM</schema>
      <schemaversion>2004 3rd
Edition</schemaversion>
   </metadata>
   <organizations default="DMCE">
    <organization
identifier="DMCE">
      <title>SCORM 2004 3rd Edi-
tion Data Model
Content Example 1.1</
title>
```

Figure 5. A content package contains a manifest file – that lists all resources (SCOs and assets) in the package and its structure diagram ("organization")--, and all physical SCO and asset files for the content package.



```
<manifest>
    <metadata>…</metadata>
    <organizations>
        <organization>
            <item identifier="ID-0"
             identifierref="RES-0" />
        </organization>
    </organizations>

    <resources>
        <resource identifier="RES-0">
            <file href="applet.html" />
            <file href="lib/applet.jar" />
        </resource>
    </resources>
</manifest>
```

Figure 6. Conceptual content structure of an LLO. In this example, the LLO defines "applet.html" as the "entry page" to the experiment. The experiment includes an applet to perform its functionality (for instance, remote-control a robot).

package called the manifest file (imsmanifest.xml); and the content making up the content package. SCORM recommends that the content packages be created as Package Interchange Files (PIF) [17] conformant with RFC 1951 [18], and mandates that the archive format be PKZip v2.04g (.zip).

Specifically, experiments in LiLa are available via LLOs ("LiLa Learning Objects"). These are special content packages with a few imposed restrictions as defined in [19], like the following requirements relevant to the booking system:

- An LLO must contain an HTML file that renders the content of the laboratory.
- An LLO may only render a single web page, namely the page showing or providing the experiment.
- If an LLO requires a booking system, the LLO metadata shall additionally contain the URL of the corresponding booking system server.

B. LiLa, SCORM and ECMAScript (JavaScript)

To allow the communication between experiments and LiLa, LiLa adheres to the SCORM reference model. In SCORM terminology, the content (experiments) communicates with the software that controls its execution and delivery (the LiLa Portal, or an LMS hosting LiLa experiments) by using the "IEEE 1484.11.2-2003 Standard for Learning Technology – ECMAScript Application Programming Interface for Content to Runtime Services Communication." [20] (For more details, see chap. 3 in [16].) ECMAScript is also better known under the name *JavaScript.*

Technically, it means that the LiLa Portal must partially implement the API for content to run-time service (RTS) communication. The experiments can rely on this API being available, and can use it to query, for instance, the name of the user of the experiment.

C. LiLa Booking System and JavaScript

To implement the access control mechanism, LiLa embeds a JavaScript code fragment to every uploaded experiment that requires the booking system. This code becomes an integral part of the experiment (it runs in the students' Web browsers) and is responsible for checking against a RESTful [21] server that the user of an experiment has a valid stamped ticket and that she is accessing the experiment at her reserved time-slot. If this is not the case, the code will render an informative message and will redirect the user to an interface (hosted in LiLa) to allow her making a new reservation.

The decision of implementing the access control check with JavaScript code that executes in the client side (that is, in the students' Web browsers) has two important consequences:

- Experiments exported from LiLa into a different LMS will still be subjected to the LiLa Booking System access control
- Students with appropriate programming skills could locally edit the JavaScript code to their advantage and overpass the access control.

The LiLa Booking System is being provided though only as a helpful tool to ease the administration of rigs availability. The implementation of an all-purpose, universal and robust booking system mechanism surpasses the goals and resources of the LiLa project. It should furthermore be noted that one of the goals of the architecture was to depend solely on the interfaces guaranteed by SCORM, and these are based on client-side execution of JavaScript.

Nevertheless, it is not as bad as it seems, because an experiment could still be designed to enforce the checking of a valid stamped ticket if this checking is implemented in the laboratory remote side, as we will see later in section IV.D.d). Such a mechanism has been implemented, for example, for controlling access to the WebLabs at the University of Cambridge.

As content providers typically do not want to implement the necessary JavaScript code themselves, the client-side code to check against the booking system is automatically embedded into the LLO upon uploading it to the LiLa portal. In the simplest possible scenario, content providers would rather implement LLOs that directly render the experiment in the web browser, and all access checking is left to LiLa. No further check would be done, and no further infrastructure at the content provider side would have to be provided. An additional check to the booking system on the server side is hence only necessary in case additional security considerations seem appropriate.

Another issue that must be considered is that this design decision restricts the execution of experiments to JavaScript-capable devices. However, JavaScript is also required when accessing standard SCORM content, so this is not a new restriction.

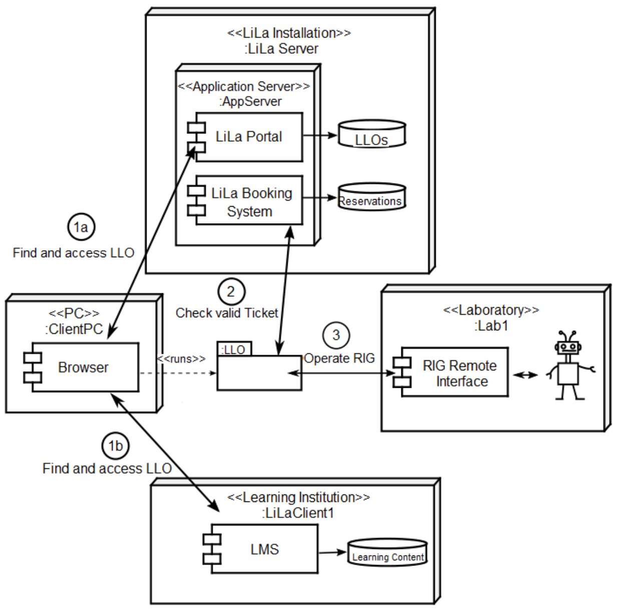


Figure 7. A student can find and access an experiment through the LiLa Portal, or through any LMS where LiLa experiments have previously been uploaded (1a or 1b). The student's Web browser will download the experiment and execute the booking system code to check access permissions (2). If the student has a valid ticket, she can execute the experiment (3).

D. LiLa Architecture

The diagram in Figure 7. depicts very briefly the communication flow during the execution of an experiment. For simplicity, the authentication dialog has been omitted from this overview diagram.

a) Client side: Web browser

In LiLa, students run the remote experiments from within their JavaScript capable Web browsers. When the student accesses an experiment protected by the booking system, the Web browser will be first redirected to request authentication. Once authenticated, it will load the experiment content and execute its associated JavaScript. To operate a remote rig, the browser will need to communicate with an API running in the remote server that interfaces with the rig. The definition of this API is beyond the scope of LiLa; typically, Flash or Java based interfaces implement this API, sometimes LabView plugins are deployed as well. Thus, quite unlike other projects, LiLa *does not* define the interface to the rig.

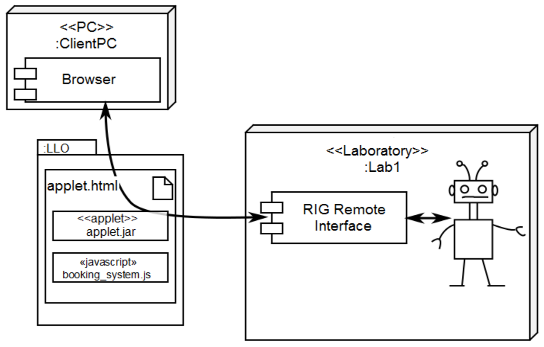


Figure 8. The arrows represent the communication between the browser and the rig

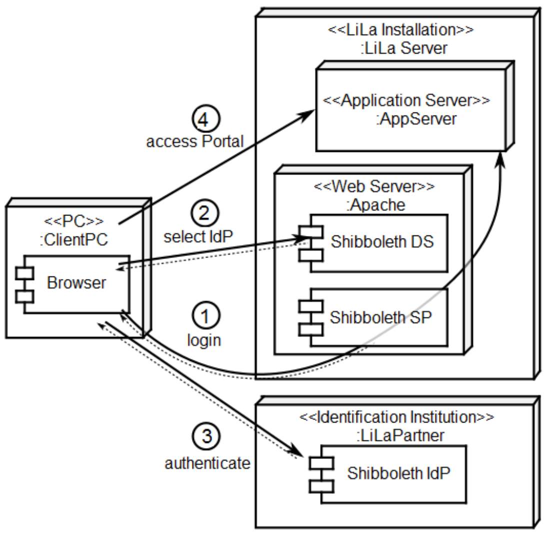


Figure 9. Authentication via Shibboleth

b) User authentication: Shibboleth

LiLa uses the Shibboleth® System to implement its authentication requirements. Shibboleth is a federative authentication infrastructure for web sign-on that allows sites to make informed authorization decisions for individual access of protected online resources in a privacy-preserving manner. [22]

There are two primary parts to the Shibboleth system: the Identity Provider (IdP), that answers to user identification requests; and the Service Provider (SP), that runs on the provider side and protects the restricted service. The SP can additionally run a discovery service (DS) that allows users to select an IdP from a list of trusted IdPs. (See Figure 9. .)

The official LiLa website includes already an IdP (for institutions that prefer not to install their own); and is itself a SP that can require user authentication. It has also a DS service installed to delegate the IdP administration to other institutions.

c) LiLa Portal

The LiLa Portal is a Web application that runs on an application server; uses a database to persist experiment information and user interactions; provides an interface that allows searching, classifying and commenting experiments; and makes the experiments available in Internet.

The LiLa Portal also implements the SCORM RTS, allowing the communication between experiments and LiLa.

d) Booking System

The LiLa Booking System is implemented as a client-server solution: The server is a Web application that runs on an application server. It implements a RESTful API (Application Programming Interface) that can be invoked from any external HTTP client (for example, a Web browser running JavaScript).

The server persists all reservations-related data in a database. The persisted data include information about the rigs, time-slots reserved and a codified user unique identifier (to preserve user privacy).

The client is provided in two flavours: first, the LiLa Portal already checks against the LiLa Booking System server that the user has a valid ticket for an experiment. If not, the experiment content will not be accessible. Second, LiLa also embeds into every uploaded experiment a JavaScript code responsible for checking this possession of a valid ticket. (See previous section IV.C, "LiLa Booking System and JavaScript.")

Note that if an LMS that hosts an experiment wants to enforce the booking system access control, the rig exercised by the experiment should itself communicate with the LiLa Booking System server, so an intrepid programmer cannot overpass the access control.

It is theoretically possible to use a different booking system for LiLa experiments. Nevertheless, this case is by now very unlikely to happen because the LiLa Booking System RESTful API has still not been open-sourced and will probably change in the near future through successive incompatible versions.

e) LMS

Some functionality like content organization, content authoring, students tracking, etc. already implemented by existing LMSs is out of the scope of the LiLa project, but complements very well its purpose. LiLa was originally conceived as a repository of experiments. The use of an LMS in combination with LiLa allows for a coherent and comprehensive delivery of learning content.

LiLa assumes that the usual case is that learning institutions already operate some kind of LMS. To leverage this existing infrastructure, LiLa offers the functionality of downloading experiments as SCORM-compatible content packages that can be uploaded into any SCORM-capable LMS. The learning institutions can this way profit from a booking system for their experiments with almost no extra-costs for them.

For students, this model provides the advantage of presenting a consistent view on the learning material, i.e. students never have to leave the LMS of the university to perform their homework.

V. TYPICAL SCENARIOS

In this section, we explain the general process of making a rig accessed by an LLO available for booking, and the process of booking and using it. After that, we describe the basic scenario for each of the different users involved in the LiLa Booking System to provide an overview of the conceptual model our work is based on.

A. Overview: activity diagram

The activity diagram in I.Adepicts the process of making a rig available and using it. As we can see, it separates the corresponding steps into three different lanes according to the Portal user role. This process is divided into the following steps:

- The content provider registers a RIG in LiLa using the specific functionality of the Portal. The reason why the rig registration is needed is that Rigs themselves are booked, rather than the experiments.
- The content provider uploads a RR-LLO (a LLO that requires reservation, unlike LLOs based on virtual experiments that are accessible without any booking procedure).
- The content provider creates at least one reservation for teachers, using the content provider functionality of the Booking System within the LiLa Portal. This reservation is associated with a unique “booking code”.
- The content provider provides this “booking code” to at least one teacher. Now, the rig and all experiments depending on it are available for the teachers.
- A teacher enters this “booking code” into the LiLa Booking System, using the teacher’s functionality of the LiLa Portal. This step provides the teacher with the possibility to divide this reservation into several reservations for students that they can individually book.

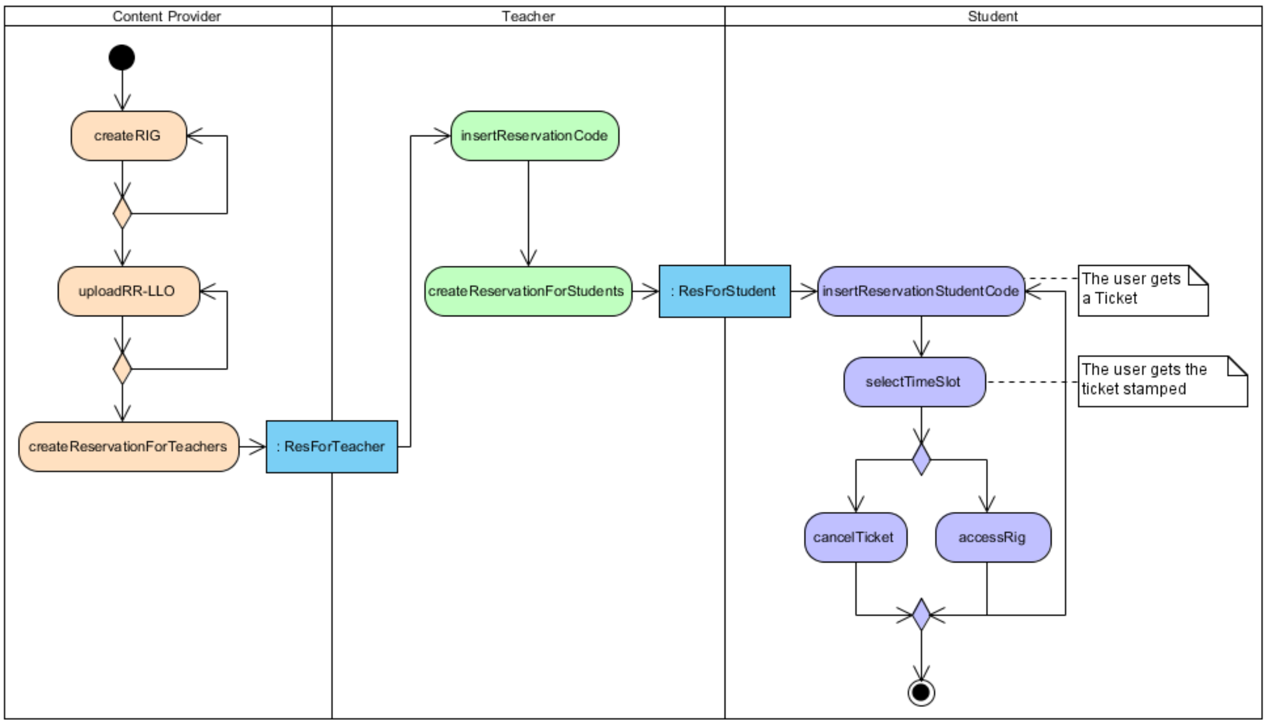


Figure 10. The UML activity diagram summarizes the process of making a rig available for booking, and booking and using it.

- The teacher creates at least one reservation for students, using a unique "student booking code" and provides this code to the students by any means they see fit.
- The student enters the "student booking code" in order to access the experiment. The system creates a ticket for the student.
- The student selects an available time-slot for running the experiment and the system stamps the ticket.
- Once the student has a stamped ticket, she can redeem it and access the rig during its assigned time-slot, or cancel the ticket and select a new time-slot. After redeeming a stamped ticket, the ticket becomes available for a new reservation of the same rig again.

Below, we describe how a LiLa authenticated user can use the LiLa Booking System, as a content provider, as a teacher and as a student.

B. *How a content provider makes a rig available for teachers*

In this step, additional information on the rig is required:

- The rig name and a short description of the hardware.
- The booking system by which this rig can be booked by users. The content provider can select "The LiLa Booking System" if she wants to use the booking system provided "out-of-the-box" with LiLa.
- The number of instances of the Rig; this number describes how many identical copies of the same setup the content provider installed. Even though this number would typically be one, some universities may choose to duplicate rigs to increase accessibility (for example, the University of Technology in Sydney – UTS – follows this strategy).
- The number of users per instance that represents how many users can run and view the identical rig at the same time. Again, this number is typically one, but for experiments running on a numerical cluster, the same hardware could accommodate more than one simultaneous user.

Once the content provider has created the rig successfully, the next step is uploading the RR-LLO. This step is divided into two smaller steps. In the first one, the content provider needs to provide the LLO zip file to upload and the name of the rig that the experiment runs on. The name of the rig is required if the content provider wants to use the LiLa Booking System. Otherwise, the content provider must select "No Rig Required" as LiLa is not concerned with booking in this case; then, of course, the LiLa protocol for booking is not relevant. The second step requires the content provider to provide some LLO metadata, such as title, description, creator, language, contact, rights holder, etc.

If an uploaded experiment identifies a rig it depends on, the LiLa Portal will offer to the content provider the functionality to create reservations for teachers. That is, in this step of the scenario, content providers will define the availability of a selected rig and create reservations for teachers; and along with such a reservation, a "reservation code".

For managing one reservation for teachers, the content provider is required to enter some sensible information about the reservation:

- The rig the reservation code shall be valid for. The content provider can select one of her owned rigs from a list.
- Booking code (or reservation code). The booking code has to be unique; Selecting a proper code is up to the content provider, the only constraint is its uniqueness. The very same code is then communicated to teachers for making experiments based on the booked rig accessible to students.
- Time-slots: the content provider must specify the time-slots within which the teacher can create reservation for their students. It is possible to specify as many time-slots as the content provider seems necessary, e.g. "Monday, Wednesday and Friday from 10 am. to 2 pm".

Finally, the content provider has to communicate the reservation code to the teachers or institutions interested. The rig and all experiments depending on it are now available for use.

The sequence diagram that explains the previous process is shown below.

C. *How a teacher allocates resources for her students*

One of the teachers' tasks is to locate and identify experiments that fit their pedagogical. If a teacher finds an interesting experiment that is subjected to scheduled access control, the teacher must contact the content provider of this rig used by the experiment, and negotiate with the provider on the availability as described above.

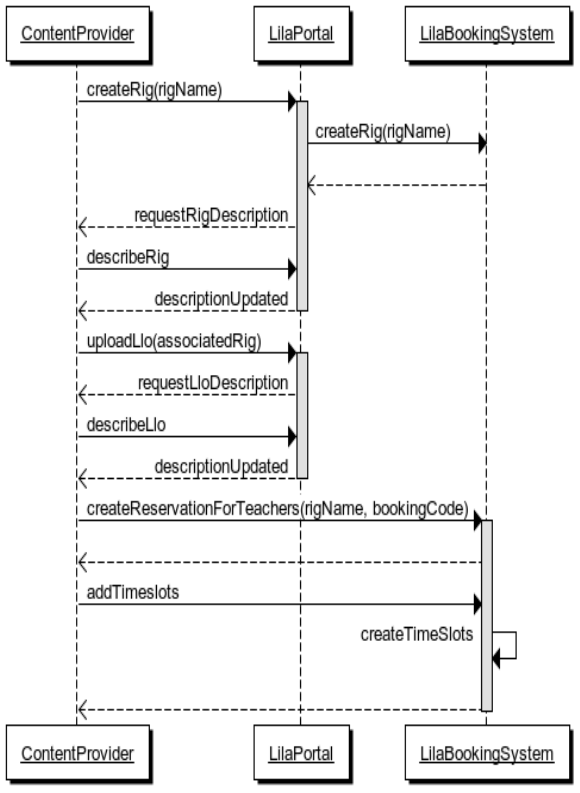


Figure 11. UML sequence diagram that specify how the content provider makes a rig available for teachers

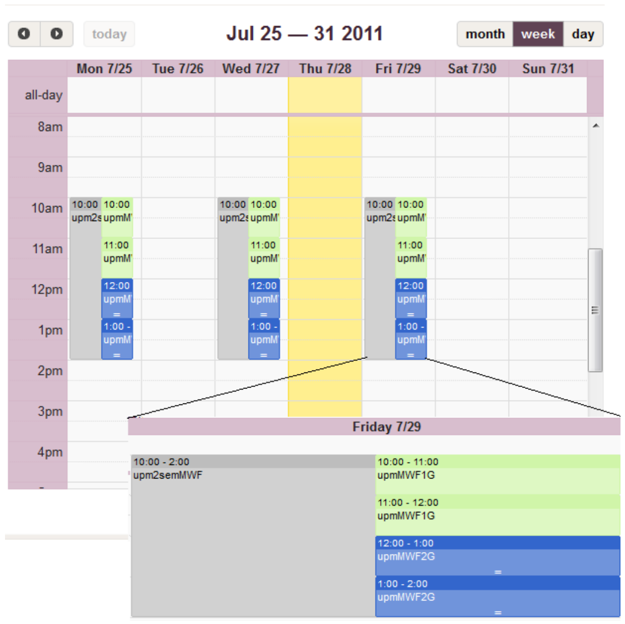


Figure 12. How to split the time-slots of a reservation for teacher into several reservations for students

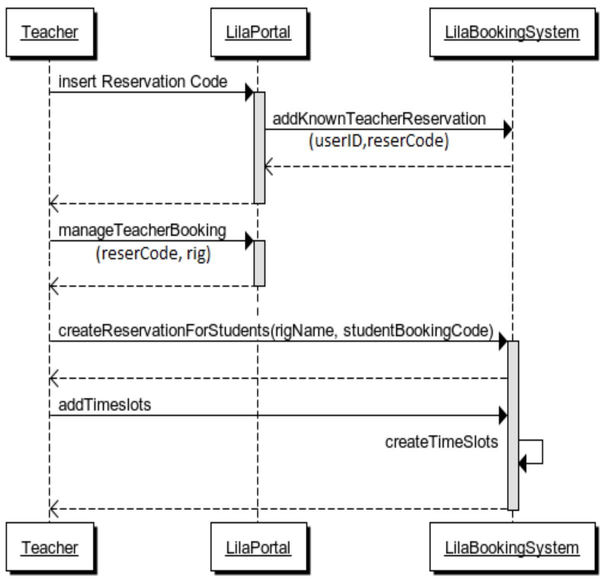


Figure 13. UML sequence diagram that specify how a teacher allocates resources for her students

As result of this negotiation, the teacher will get a “reservation code” from the content provider of the rig.

This booking code is valid for an infinite number of student bookings within a given time period, and allows the access to the rig reservation during this time; it is, however, not valid for student booking or accessing the experimentor rig directly. Instead, the LiLa booking system allows the owner of such a code to divide the booked time-slot up into several slots that students can book individually. The student slots, however, can no longer be broken up.

In the simplest possible use case, the entire time-slot reserved by the content provider is handed over to a single, large group of students, though set-ups that are more complicated are possible. For example, a teacher may give two lectures, both requiring the same experiment. In this case, she would probably want to break up the time-slot granted by the content provider into two slots, one for the first and another for the second student group, as is shown in the Figure 12. This ensures that the two groups cannot conflict with each other.

In Figure 12. , we can identify three big time-slots in grey color (Monday, Wednesday and Friday from 10 am to 2 pm) corresponding to the reservation for teachers whose reservation code is “upm2semMWF”, and 2 smaller groups of time-slots for students. The green colored is for the first group of students and the student booking code is “upmMWF1G”; the students who are in this group can access the rig on Monday, Wednesday and Friday from 10 am to 12 pm. The other group of students can access the rig the same days from 12pm to 2 pm using the student reservation code “upmMWF2G”. The time-slots for the second group are represented by means of blue color. In this use case, the teacher should create two different reservations for users or students.

To make the rig accessible to her students, the teacher then must log in into the LiLa Portal, and access the LiLa Booking System functionality; there, she must enter this reservation code to add the negotiated availability of the rig to her calendar of rigs availability.

As in the previous use case, the sequence diagram that explains the process of creating reservation for users and how a teacher can use the LiLa booking system is shown in Figure 13.

D. *How a student accesses a schedule-controlled rig*

Within the LiLa framework, access to the LLOs is provided to groups of students – typically members of a lecture. Along with the reservation code, it is in the responsibility of the teacher to hand out the information on the URL under which the experiments can be accessed (could be the LiLa Portal or an LMS).

Once the student has a student booking code and the URL to access the experiment, she needs to book a free time-slot of the set of time-slots that the teacher created for that rig and reservation booking code. For making a reservation of an LLO using the LiLa booking system (the student books the rig that the LLO uses), the student needs to do the following steps (see Figure 14.):

- Go to the URL in her Web browser, and sign into the system maintaining the experiments; this could either be the LiLa portal, but would typically be the Learning Management System of the university. The LiLa portal or the LMS will detect that the experiment requires the LiLa Booking System and this option requires an authenticated student. This authentication is thus handled by LiLa or by the LMS.
- The LMS or the LiLa portal will provide a student identity. This identity is communicated to the JavaScript code embedded into the LLO upon uploading it to the Lila portal.
- Enter the booking code into the provided interface, and click on Submit. Now, a calendar will appear that shows the student the available time-slots to run the experiment.

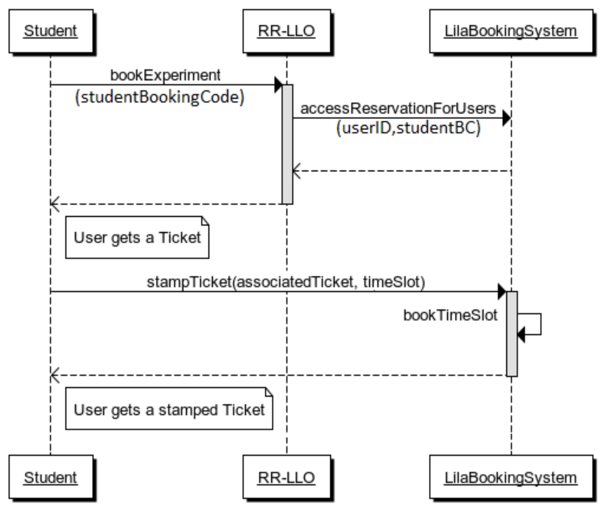


Figure 14. UML sequence diagram that specify how a student books a schedule-controlled rig

- Select a free time-slot by clicking on it. The LiLa Booking System will "stamp" the ticket for the student, granting her the access to the rig during the chosen time-slot. The "stamping" prohibits that the same ticket can be used again to reserve the same experiment twice. That is, the student cannot book the same experiment again before her reserved time-slot has passed.

With one "reservation code", a student gets one ticket; and with one ticket, she can only get one stamped ticket, valid for running the suitable LLO. A student can only request a new stamped ticket based on the same booking code by cancelling her old one; additionally, a stamped ticked becomes valid for a new reservation again – i.e. the "stamp is removed" – once the time-slot reserved by the ticked has been expired. As we can see, besides the student can book a rig or a LLO, she can also access or run the experiment using the current stamped ticket, or cancel it.

In order to run the experiment, the student tries to access the RR-LLO. The JavaScript code embedded into the LLO then checks whether the user has a stamped ticket valid during the access time. On that process, the userID, rigID, and stampedTickets for the user are required.

The student can cancel a stamped ticket by clicking on the time-slot associated with the stamped ticket, in the calendar. (See Figure 15.)

VI. CONCLUSIONS AND FUTURE WORK

This paper provides a detailed description of a solution to control access to virtual laboratories and remote experiments using learning objects, in the context of the LiLa project and the requirement analysis carried out by the Global Online Lab Consortium, GoLC. The solution is based on the identification of three different user roles and the idea of tickets. This solution gives educational institutions the flexibility of accessing the booking system functionalities provided by the LiLa portal through their own Learning Management Systems, facilitating the interchange not only of virtual experiments, but also of remote experiments accessing complex and costly equipment. The LiLa portal has already been released and feedback from the users is already driving the current work on the system in two main directions: improving the metadata-based search functionality to help teachers find the experiments that they can include in their courses, and improving access control to help content providers verifying the rights of the users accessing remote experiments through the LiLa booking system.

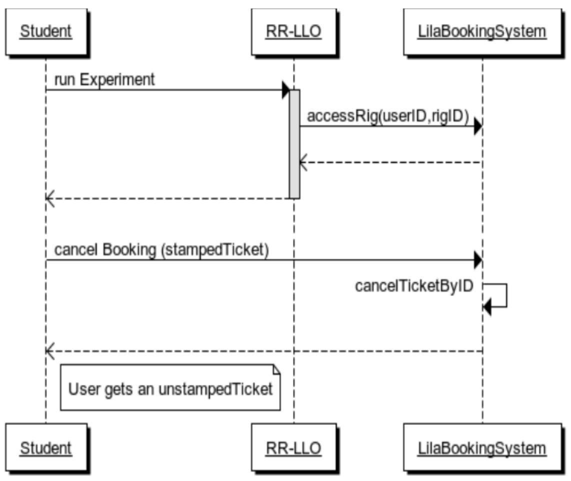


Figure 15. UML sequence diagram that show how to run an experiment and how to cancel a reservation

REFERENCES

[1] eContentplus project "Library of Labs". (2009) EU grant ECP-2008-EDU-428037.

[2] A. Gallardo et al., "A rig booking system for on-line laboratories," in *Global Engineering Education Conference (EDUCON), 2011 IEEE*, 2011, pp. 643 -648.

[3] A. Cardoso J. Ferreira, "A Moodle Extension to Book Online Labs," in *Remote Engineering and Virtual Experimentation Symposium (REV 2005)*, Brasov, Romania, 2005.

[4] D. Hercog, K. Jezernik S. Uran, "Remote Control Laboratory with Moodle Booking System," in *Int. J. Online Engineering*, vol. vol.1, n.2, 2005.

[5] Moodle.org community. (2010, November) Moodle.org: open-source community-based tools for learning. [Online]. http://moodle.org/

[6] M. Kvasnica and M. Fikar Ľ Čirka, "WebLab Module for the Moodle Learning Management System," in *Proceedings of the 9th International Conference Virtual University 2008*, E-academia Slovaca.

[7] MRBS Team. (2010, November) MRBS: Introduction. [Online]. http://mrbs.sourceforge.net/

[8] MIT iCampus Outreach Project Team. (2010, November) MIT iCampus: iLabs. [Online]. http://icampus.mit.edu/iLabs/

[9] A. Maiti, "NETLab: An Online Laboratory Management System," in *Education Engineering*

(EDUCON), 2010 IEEE, 24 June 2010.

[10] O. Wäldrich, W. Ziegler P. Wieder, "Advanced techniques for scheduling, reservation, and access management for remote laboratories," in *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*, 2006.

[11] (2010, November) VIOLA. [Online]. http://www.fz-juelich.de/jsc/grid/VIOLA

[12] H. Nakano et al., "Web-based time schedule system for multiple LMSs on the SSO/Portal environment," in *IEEE EDUCON Education Engineering 2010*, Madrid, April, 2010.

[13] Internet2 Middleware Initiative. (2011, September) Shibboleth®. [Online]. http://shibboleth.internet2.edu/

[14] P. Chang, "Computerized Reservation Systems," in *IEEE International Conference on Systems, Man and Cybernetics*, 1992.

[15] ADL. (2011, July) Home SCORM®. [Online]. http://www.adlnet.gov/Technologies/scorm/default.aspx

[16] ADL. (2009, August) Sharable Content Object Reference Model ® 2004 4th Edition Run-Time Environment Version 1.0.

[17] IMS Global Learning Consortium. (2011, September) IMS Content Packaging v1.2 Information Model. [Online]. http://www.imsglobal.org/content/packaging/cpv1p2pd2/imscp_infov1p2pd2.html#Xad1736213

[18] IETF. (1996, June) IETF RFC 1951 DEFLATE Compressed Data Format Specification version 1.3. [Online]. http://www.ietf.org

[19] P. Debicki, A. Gallardo, V. Mateos, T. Richter, V. Villagra L. Bellido. (2009, November) D2.1-Documentation of the software interfaces of the virtual portal. Document.

[20] (2005, January) IEEE 1484.11.2-2003 Standard for Learning Technology – ECMAScript Application Programming Interface for Content to Runtime Services Communication. Available at: http://www.ieee.org.

[21] Roy Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California, Irvine, Doctoral dissertation 2000.

[22] Internet2 Middleware Initiative. (2011, September) About Shibboleth®. [Online]. http://shibboleth.internet2.edu/about.html

[23] ADL. (2009) Sharable Content Object Reference Model ® 2004 4th Edition Content Aggregation Model (CAM) Version 1.1.

AUTHORS

Verónica Mateos received the M.S. degree on Telecommunications Engineering from the Technical University of Madrid (UPM), in 2008. She is a Ph.D. student at the Department of Telematics Systems Engineering at the Technical University of Madrid (UPM) since 2008. Her research interests are in the area of security networks, and ontologies and Semantic Web. She also has participated on several research national and international projects. (E-mail: vmateos@dit.upm.es)

Alberto Gallardo received the M.Sc. degree on Computer Science from the Technical University of Madrid (UPM), in 2001. He has worked in the air traffic-control software industry for eight years and is since 2009 Ph.D. student at the Department of New Media in Research and Teaching (NFL) of the University of Stuttgart Computing Center. His research interest are in the area of software quality, software architecture and software engineering. (E-mail: gallardo@rus.uni-stuttgart.de)

Luis Bellido is Associate Professor of telematic systems engineering at the Technical University of Madrid. He received his PhD degree in Telecommunication Engineering from the same University. His research interests include quality of experience evaluation, semantic web, multilingual web and advanced service design. He has acquired an extensive research, development and technical management experience in these fields through his involvement in a number of European Commission funded projects in collaboration with research and industrial organizations. (E-mail: lbellido@dit.upm.es)

Peter Debicki is student of Computer Science at the University of Stuttgart. (E-mail: ruspete@po2.uni-stuttgart.de)

Dr. Thomas Richter: Ph.D. in mathematical physics in 2000 from the Technical University of Berlin (TU-Berlin). Thomas Richter studied physics and mathematics in Berlin, and then worked as project manager at AlgoVision Technology, GmbH. In 2002, he returned to the TU for working on novel eLearning solutions. Today, he is a software architect at the RUS Computing Center of the University of Stuttgart, there in the department for "New Media for Research and Education" lead by Dr. David Boehringer. (E-mail: richter@rus.uni-stuttgart.de)

Dr. Víctor A. Villagrá: Ph.D. in computer science in 1994 from the Technical University of Madrid (UPM), Spain. He is an associate professor in telematics engineering at the UPM since 1992. He has been involved in several international research projects related with Network Management, Advanced Services Design and Network Security, as well as different national projects. He is author or co-author of more than 60 scientific papers and is author of a textbook about security in telecommunication networks. (E-mail: villagra@dit.upm.es)