Interoperating Remote Laboratory Management Systems (RLMSs) for more Efficient Sharing of Laboratory Resources

Authors

David Lowe (david.lowe@uts.edu.au): Faculty of Engineering and IT, University of Technology, Sydney (UTS).

Herbert Yeung (Herbert.Yeung-1@uts.edu.au): Faculty of Engineering and IT, University of Technology, Sydney (UTS).

Mohamed Tawfik (mtawfik@ieec.uned.es) (corresponding author): Electrical and Computer Engineering Department (DIEEC), Spanish University for Distance Education (UNED).

Elio Sancristobal (elio@ieec.uned.es): Electrical and Computer Engineering Department (DIEEC), Spanish University for Distance Education (UNED).

Manuel Castro (mcastro@ieec.uned.es): Electrical and Computer Engineering Department (DIEEC), Spanish University for Distance Education (UNED).

Pablo Orduña (pablo.orduna@deusto.es): the Internet Unit of the Deusto Institute of Technology - DeustoTech, University of Deusto.

Thomas Richter (richter@rus.uni-stuttgart.de): the Multimedia Department for Research and Education of the Computing Center of the University of Stuttgart (ITSUS).

Abstract

Remote laboratories have emerged as a viable addition to the tools supporting experiential learning. The advent of Remote Laboratory Management Systems (RLMSs) has allowed several integrated heterogeneous remote laboratories, developed with various technologies, to be administrated and accessed through a common portal. A key benefit arising from the ability to access the laboratories remotely through RLMSs is the opportunity to share laboratory resources across multiple institutions. This sharing is, however, restricted by the inability of a contemporary RLMS installed at an institution to communicate with other RLMS installed at other institution. In this paper we propose a novel Application Programming Interface (API) design pattern for inter-communication between RLMSs accommodating different levels of functional support and thereby allowing more efficient sharing of laboratory resources regardless of their hosting RLMS. The pattern is based on common profile roles supported by the predominating RLMSs, along with their underlying service calls. Afterwards, we present initial results and demonstrate the feasibility and effectiveness of this pattern by developing an API for two common RLMSs, Sahara and the iLab shared Architecture (ISA). As a result, users logging into a Sahara server managed to access and manipulate a radio-activity experiment hosted on an ISA server.

Keywords

Distance learning, interfaces, services architectures, standards and interoperability.

1. Introduction

Laboratories are generally recognized as important tools for education and scientific research. The challenges in resourcing conventional laboratories [1] and the rapid evolution of computer technology and communication networks have supported the emergence of remote laboratories as a viable educational tool [2, 3]. As a result, remote laboratories have strongly been adopted in many engineering disciplines, especially those oriented to control and industrial real world, such as industrial informatics [4-7] and industrial electronics [8, 9]. Remote laboratories have become increasingly sophisticated, with a growing body of research considering aspects such as flexibility of access [10], ability to share resources and labs [11, 12], security of users, data, and devices [13], and accessibility for disadvantaged users [14] among many others [15]. To a significant extent, many of these issues have been successfully overcome, with continuous, reliable and high quality services being maintained for much of the past decade.

Cross-institutional laboratory sharing is one of the most often raised justifications for the use of remote labs [11, 16]. This can lead to improved utilization levels, shared costs, and access to a much broader range of laboratory apparatus. Thus, The focus of remote laboratory development is now moving towards more sustainable models that promotes both institutional and individual engagement [16]. Rather than individual academics custom building equipment for their specialized subjects, remote laboratory development is increasingly being carried out by multi-institution consortia [16]. Despite its claimed benefits, the related initiatives have been historically very limited and struggled to achieve sustainability. Early examples of attempts to support shared access RLMSs include the World Wide Student Laboratory [17] and PEARL [18] projects. Similarly, the LearNet [19], ProLearn [14] and PEMCWebLab [20] projects were carried out during the early to mid-2000's with little success. More recently the LiLa [21], Sahara [11, 16, 22, 23], ISA [24-26], Weblab-Deusto [27, 28], NANSLO (Western Interstate Commission for Higher Education 2012) and UniSchooLabS [29] projects have continued to attempt to support laboratory sharing.

A key element of these developments is the creation of Remote Laboratory Management Systems (RLMSs) that provide a common online framework for accessing and administrating a wide pool of heterogeneous developed remote lab systems that might be distributed across different institutions and geographical locations [13]. They provide services such as booking, assessment, tracking, and synchronous and asynchronous communication tools, as shown in Figure 1.

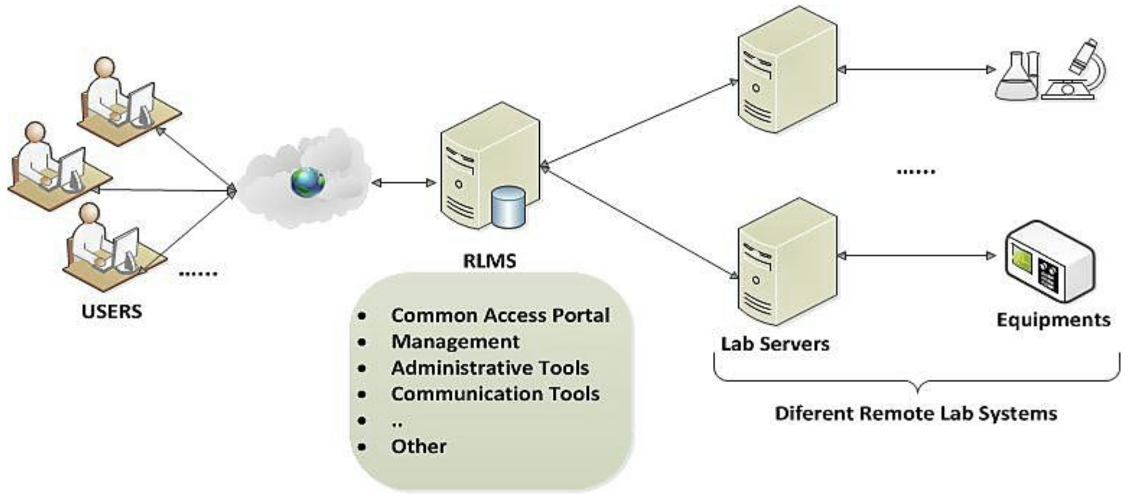


Figure 1. Architecture of RLMS.

RLMSs should be agnostic with regard to the remote laboratory design in order to support the widest range possible of remote laboratories. Examples of RLMSs that are specifically designed to directly manage a collection of remote laboratories include iLab Shared Architecture (ISA) [24-26], and Sahara [11, 16, 22, 23]. Other systems have some related functionality but have a different focus. For example, LiLa [12] provides some laboratory services (such as location and booking services) but not others (such as monitoring the laboratory status), though it does extend into wider support for the experimental activities. Each of the different RLMSs has typically arisen from a different set of pedagogic and technical design parameters, and philosophical approaches. Each has different strengths and weaknesses, and potentially addresses a different set of needs. We argue that this diversity is an asset that should be encouraged rather than avoided. However the diversity does represent a challenge with regard to supporting cross-institutional sharing of laboratories between institutions using different RLMSs.

Currently, for a remote laboratory that is provided by one institution to be accessed by users from a different institution, typically those users must directly access either the laboratory itself or the providers RLMS (if one exists). This can represent a significant hurdle to effective sharing given that it potentially places a substantial burden on the provider of the laboratory to coordinate access for the users. It can also mean that students who access multiple laboratories (possibly shared from multiple providers) will need to manage numerous access accounts and system interfaces.

Common issues facing RLMS could be summarized as follows:

- Inability to achieve Inter-institutional sharing and interoperability with heterogeneous RLMSs in order to share learning objects (ex., courses, remote labs, etc.) among institution.
- Lack of a standard design pattern, which increases the developing time for attaching the same remote laboratory to different RLMSs at different institutions.
- Inability to access labs from different systems within a single interface as well as duplicated maintenance efforts for accounts on multiple systems.

This can be partially addressed by an architecture that uses federation and single sign-on technologies so that a student from institution A can access the RLMS from institution B through a mechanism defined by the federation and contracts between institution A and B. The downside of this is that the students must still access different remote laboratories through different systems. A common portal might address this to some extent, but the system would then potentially lack a consistent look-and-feel. An alternative, which we focus on in this paper, is to allow the home institution of the students to maintain their own RLMS, and for this system to be able to interoperate with other RLMSs so that students can directly access provider's experiments. In other words, consider the scenario shown in Figure 2, where the RLMS ISA is installed at the Massachusetts Institute of Technology (MIT) and providing access for the MIT users (MIT 1, MIT 2...) to experiments installed at MIT (MIT Exp A, MIT Exp B...). On the other hand, the RLMS Sahara is installed at the University of Technology, Sydney (UTS), providing access for the UTS users (UTS 1, UTS 2...) to experiments installed at UTS (UTS Exp A, UTS Exp B …). Adopting an Application Programming Interface (API) to communicate between both systems, it would be possible for a UTS user to gain access through Sahara to experiments installed at MIT and integrated in ISA if there is an agreement between both institutions and as long as both architectures support this API in order to interconnect with each other. Since API is meant to be language independent, any RLMS provider could implement them even if the RLMS architecture was not initially considering its provision in its designed. For instance, it could be provided as a plug-in. It is also necessary to mention that the API approach provides a solution for a higher level communication among RLMSs, and once the communication is stablished and access to a particular remote lab is granted, communication will be dictated by the lab itself.

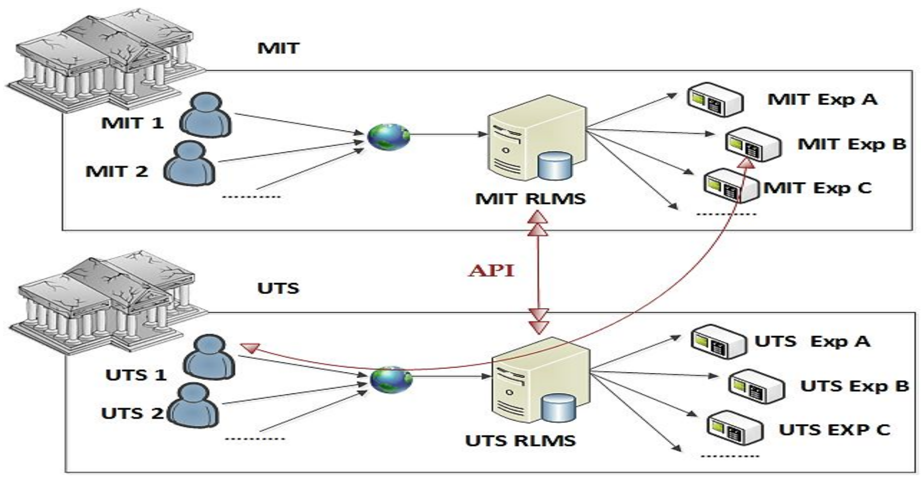


Figure 2. Topology of connection between two RLMSs using standard API.

In this paper we address this limitation and describe a novel API design pattern for inter-communication between RLMSs defining the underlying design principles, the supported profiles and the set of calls underlying each profile, as shown in Section 2. In the same section, we explain the capabilities of the proposed design using a sequence of illustrative scenarios. Afterwards, in Section 3, and basing on the proposed design pattern, we develop and implement an API to allow a radio-activity experiment hosted on the ISA server at the University of Queensland to be accessed by users logging into a Sahara server at the UTS. Finally, the conclusion and the future works are addressed in Section 4.

2. Design Pattern

The proposed pattern was developed by the faculty of Engineering and IT at UTS with the aid of several partners from the Global Online Laboratory Consortium (GOLC) [30], which encompasses pioneers in remote laboratory development from all over the world. In this section the pattern is defined and discussed. However, it is anticipated that as the builders of RLMSs gain experience with this pattern further refinements will be required. The pattern was realized from a wide perspective in order to allow developing APIs that are adaptable with any existing RLMS. Thus, it was devised on the basis of the common basic functionality provided by most of the contemporary RLMSs. As well, it adopts common terminology used by most of the RLMSs developers. The purpose of the design pattern is to achieve a common interface between RLMSs rather than dictating how the functionality or web service calls are implemented internally to each system (this is left to the developers or maintainers of those systems).

2.1. Terminology

In this section, a detailed definition for each term adopted in the design pattern is presented:

- **Rig Instance (Rig):** A single instance of a physical system made from various devices (including associated software and hardware) that can be used by one or more students in carrying out a distinct and discrete learning activity.
- **Rig Type:** The class to which a rig belongs, and within which any rig can be used interchangeably. A given institution (or laboratory provider) will often have more than one rig type, and for each rig type they may have multiple rigs.
- **Rig Set (Pool):** The collection of homogeneous rigs, all of the same rig type, which may be distributed across multiple physical locations and managed by different organizations, but which can be treated as a single logical collection for access purposes.
- ***Rig Capability:*** A list of identifying tags which may be used to correlate rigs into collections to queue or book for. This allows, for example, multiple rig types to be collectively queued to get the first free rig in any of the rig types.

- **User:** An individual uniquely identifiable within a remote laboratory system. The user must be able to be authenticated (by the consumer RLMS), but may be made anonymous to the provider RLMS.
- **Consumer User/Organization/RLMS:** A user/organization/RLMS that accesses remote laboratories that have been made available by a provider. It is up to the consumer RLMS to appropriately authenticate the user and determine which consumer groups they belong to.
- **Consumer Group:** A cohort of users from a consumer organization which corresponds to a particular usage agreement between a consumer organization and a provider organization, and which may have a specific set of access permissions and constraints (such as allowable periods during which the apparatus can be accessed, maximum session times, or allowable total usage for the cohort).
- **Provider Organization/RLMS:** An organization/ RLMS that makes available remote laboratories for use by consumers.

In general terms, a provider (whether it is an organization or a RLMS) is responsible for making available rigs associated with a specific service policy. A consumer group is granted access to provider's rigs under an agreement with the provider, which is based on the provider's service policy. Consumers belonging to this group access the provider's rigs after getting authenticated through their group's RLMS (though the consumers' RLMS may rely on other systems to support such authentication). A provider will control access based on the identified consumer groups, not individual users, and only requires identification of the user in order to track usage.

2.2. Principles

- The design pattern is based on a set of underlying principles that establish an accepted baseline, and all ongoing development should be consistent with these principles.
- The API shall provide for mandatory baseline profiles that must be supported in all RLMSs that adhere to it and additionally, optional profiles that define sets of functional capabilities beyond the baseline capabilities, e.g. support for lesson plans could be added in an optionally-supported profile. The API shall include a mechanism for querying which optional profiles a given RLMS supports.
- The API shall accommodate interoperability between RLMSs that are compliant with any version of it.
- Versions of this API shall, wherever possible, be upwardly compatible, i.e. a later version should not invalidate functionality specified in an earlier version. Where this is unavoidable the deprecated functionality should be clearly identified.
- The API shall accommodate varying choices by laboratory providers regarding the level of service they wish to provide with regard to their RLMS (in terms of service availability, information security, quality, retention, etc.).
- The API shall not address user-level authentication and authorization, but rather user consumer-level authentication and group-based authorization based on industry standard authentication mechanisms. The API shall support at least:
    a) Consumer-level authentication (i.e. identifying the institution that is requesting access – e.g. UTS – from a given provider – e.g. MIT).
    b) Multiple agreements with the same consumer, e.g. multiple different UTS course conveners may have been given different access levels as part of the UTS access agreement with MIT.
    c) Optional anonymous user-level identification by the consumer to the provider. The consumer may indicate that a given request for rig access is coming from a specific, but anonymous, user, e.g. UTS makes a request to MIT, indicating that the request is for user 37, in group B. Where unique users are not distinctly identified this should be made clear in the request.

2.3. Profiles

Along with the underlying principles, the design pattern encompasses a set of profiles that defines the communication patterns between systems. The base-line profiles that are supported in this version of the standard and that should be common in most of the developed RLMSs are the following:

- **System Query profile:** Supports basic system enquiries required to determine agreed supported services between consumers and providers (the baseline principles)

- **Rig Query profile:** Supports basic system enquiries required to determine the rig sets that are made available by a lab provider.
- **Experiment Query profile:** Supports basic system enquiries required to determine the executed experiments that have been carried out by a lab provider.
- **Basic Rig Access profile:** This profile provides support for basic access to provided rig sets or specific rigs. Two primary modes of access are (optionally) supported by the provider: queuing, where the user can request access and they then enter a queue with access being granted when they reach the front of the queue and a rig becomes available; and booking, where the user can make a booking for a particular time slot.
- **Basic Batch Experiment profile:** Basic support for execution of batch based laboratories using either provider–supplied or consumer-supplied experiment interface.
- **Basic Interactive Experiment profile:** Basic support for execution of interactive based laboratories using either provider –supplied or consumer-supplied experiment interface.

Each base-line profile supports a collection of calls which, when used together can enable a range of different scenarios. Table 1 illustrates and defines the calls of each profile.

Table 1. List of base-line profiles and their supported calls.

System Query profile	QueryProfiles()	Obtains a list of profiles (and profile versions) supported by the provider.
	QueryServiceLevel()	Obtains provider-specific information on the level of his provided service.
Rig Query profile	GetRigs()	If no rig sets specified, it returns a list of unique top level rig set laboratory identifiers that are accessible to this consumer. If a rig set is specified, it returns the children rig sets.
	GetRigStatus()	Provides the current status of the rig set, and whether it is currently operational.
	GetRigInfo()	Obtains any relevant property information pertaining to the rig set.
Experiment Query profile	GetExperimentIDs()	Retrieves a list of experiment IDs that are active and associated with rigs in this rig set, and related to a particular consumer, consumer group or user.
	GetExperimentStatus()	Indicates the current status of an experiment, whether the experiment is running (and if so then for how long), or if it is in an idle state.
	GetExperimentResults()	Retrieves the results from a (partially or fully) completed experiment execution.
Basic Rig Access profile	RigAccessType()	Determines whether a particular rig set is bookable or queuable (or both).
	RigAvailability()	For bookable rig sets, it allows querying the time periods within a given timeframe that are currently available for booking.
	RigBooking()	Requests a booking to be made for the given experiment definition. An email address can be associated with the booking for confirmation.
	RigBookingsQuery()	Retrieves a list of all existing bookings that have been made for the specified consumer, user group, or user.
	RigBookingCancel()	Cancels an existing booking.
	RigBookingStatus()	Queries the status of an existing booking and in particular if the booking is currently available for redemption.
	RigQueueQuery()	Adds the given experiment definition into the queue.
	RigQueueCancel()	Removes a given experiment definition from a queue.
	RigQueueStatus()	Requests information on the current status of an experiment definition with a queue.

Basic Batch Experiment profile	BatchExpCreate()	Creates a blank "empty" experiment definition on the provider, to be executed on the permitted specified rig set.
	BatchExpDelete ()	Deletes a previously created experiment definition.
	BatchExpGetSpecs()	Obtains a list of the information required for successful execution of an experiment.
	BatchExpSetDefn()	Provides the information required to define a valid experiment and saves this into an existing experiment.
	BatchExpGetDefn()	Retrieves the experiment definition from a previously defined experiment.
	BatchExpExecute()	Requests the execution of a previously defined batch experiment. The experiment (if accepted for execution) will then be queued for execution.
	BatchExpLaunchProvIF()	Provides a URL to use in launching a user interface associated with the specified experiment.
Basic Interactive Experiment profile	InteractiveExpCreate()	Creates a blank "empty" experiment definition on the provider, to be executed on the permitted rig set.
	InteractiveExpDelete()	Deletes a previously created experiment definition.
	InteractiveExpLaunchConsIF()	Requests the commencement of an interactive experiment using a consumer provided interface.
	InteractiveExpLaunchProvIF()	Provides a URL to use in launching a user interface associated with the specified experiment.

2.4. Scenarios

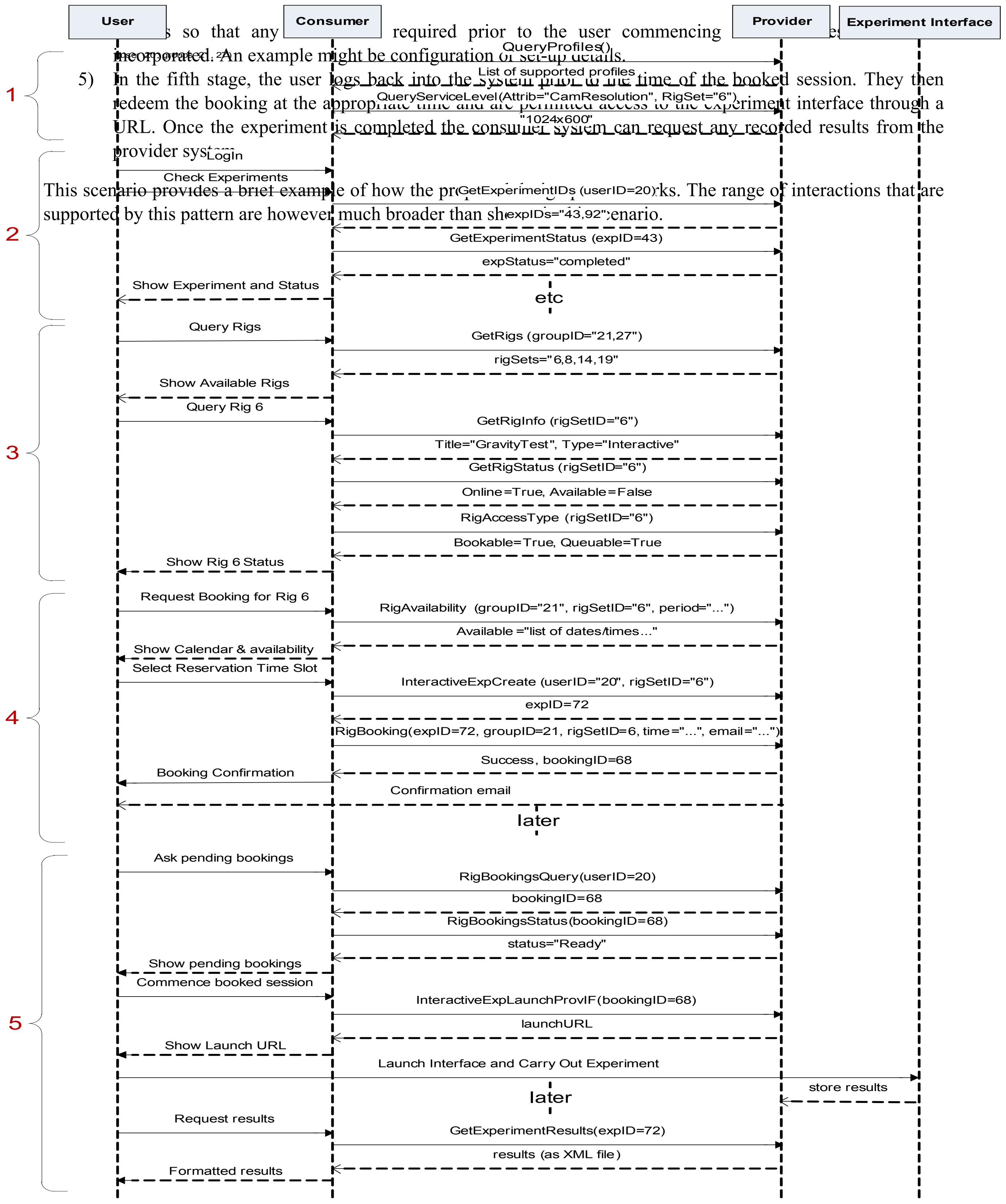
To illustrate the capabilities that are enabled by these profiles, and allow an evaluation of any potential limitations, we present an illustrative example shown in Figure 3. The example shows a sequence of interactions that collectively represent a specific scenario involving booking and then accessing an interactive experiment. In this example, it is assumed the consumer has contracted with the provider for access to numerous rig sets, but user groups 21 and 27 (to which user 20 belongs) only have access to numbers 6, 8, 14 and 19. In this case, the communication between consumer and provider can be classified into five main stages:

1) In the first stage, the consumer organization retrieves information on the profiles that are supported by the provider and the level of service associated with particular sets rigs, so that this can be used either to provide information to their users, or to adapt the subsequent requests. For example, the camera resolution may be retrieved so that the user interface can be adapted, or the policy on information retention might be checked so that the users can be warned when results might be removed by the provider.
2) In the second stage, the consumer user number 20 (who belongs to consumer groups 21 and 27) logs into the consumer system. Note that the consumer system is responsible for authenticating their own users and managing the groups that they belong to. Further, whilst the provider gives the consumer system access to certain agreed sets of rigs, it is the consumer systems responsibility to determine which users should be able to access which rigs. The user then enquires about the status of any experiments that they previously had created and executed. The consumer system retrieves the list of experiments for this user (43 and 92), and then requests information on each experiment, before presenting this back to the user. i.e., those numbers (43 and 92) are assigned randomly as an example.
3) In the third stage, the user wishes to carry out a new experiment. They begin by asking to see a list of the rigs to which they have access. They then select a specific rig (ID=6) and check its availability and access type. The response indicates that the rig is bookable and quotable, and whilst it is currently online it is unavailable at that moment.
4) In the fourth stage, the user wishes to make a reservation for Rig 6. The consumer system defaults to showing a booking calendar for the next 48 hours, so queries the provider system for availability over this period. The calendar is shown to the user by the consumer system, and the user requests a booking at a particular time. The consumer system attempts to make the booking which is then confirmed (bookingID=68). The Provider system also emails the user a confirmation of the booking. Note that the system had to first create an "Experiment" on the provider system that was associated with the booking.

[illegible]s so that any [illegible] required prior to the user commencing [illegible] incorporated. An example might be configuration or set-up details.

5) In the fifth stage, the user logs back into the system prior to the time of the booked session. They then redeem the booking at the appropriate time and are permitted access to the experiment interface through a URL. Once the experiment is completed the consumer system can request any recorded results from the provider syst[illegible]

This scenario provides a brief example of how the pr[illegible]ks. The range of interactions that are supported by this pattern are however much broader than sh[illegible]enario.



User
Consumer
Provider
Experiment Interface
1
QueryProfiles()
List of supported profiles
QueryServiceLevel(Attrib="CamResolution", RigSet="6")
"1024x600"
2
Login
Check Experiments
GetExperimentIDs (userID=20)
expIDs="43,92"
GetExperimentStatus (expID=43)
expStatus="completed"
Show Experiment and Status
etc
3
Query Rigs
GetRigs (groupID="21,27")
rigSets="6,8,14,19"
Show Available Rigs
Query Rig 6
GetRigInfo (rigSetID="6")
Title="GravityTest", Type="Interactive"
GetRigStatus (rigSetID="6")
Online=True, Available=False
RigAccessType (rigSetID="6")
Bookable=True, Queuable=True
Show Rig 6 Status
4
Request Booking for Rig 6
RigAvailability (groupID="21", rigSetID="6", period="...")
Available ="list of dates/times..."
Show Calendar & availability
Select Reservation Time Slot
InteractiveExpCreate (userID="20", rigSetID="6")
expID=72
RigBooking(expID=72, groupID=21, rigSetID=6, time="...", email="...")
Success, bookingID=68
Booking Confirmation
Confirmation email
later
5
Ask pending bookings
RigBookingsQuery(userID=20)
bookingID=68
RigBookingsStatus(bookingID=68)
status="Ready"
Show pending bookings
Commence booked session
InteractiveExpLaunchProvIF(bookingID=68)
launchURL
Show Launch URL
Launch Interface and Carry Out Experiment
store results
later
Request results
GetExperimentResults(expID=72)
results (as XML file)
Formatted results

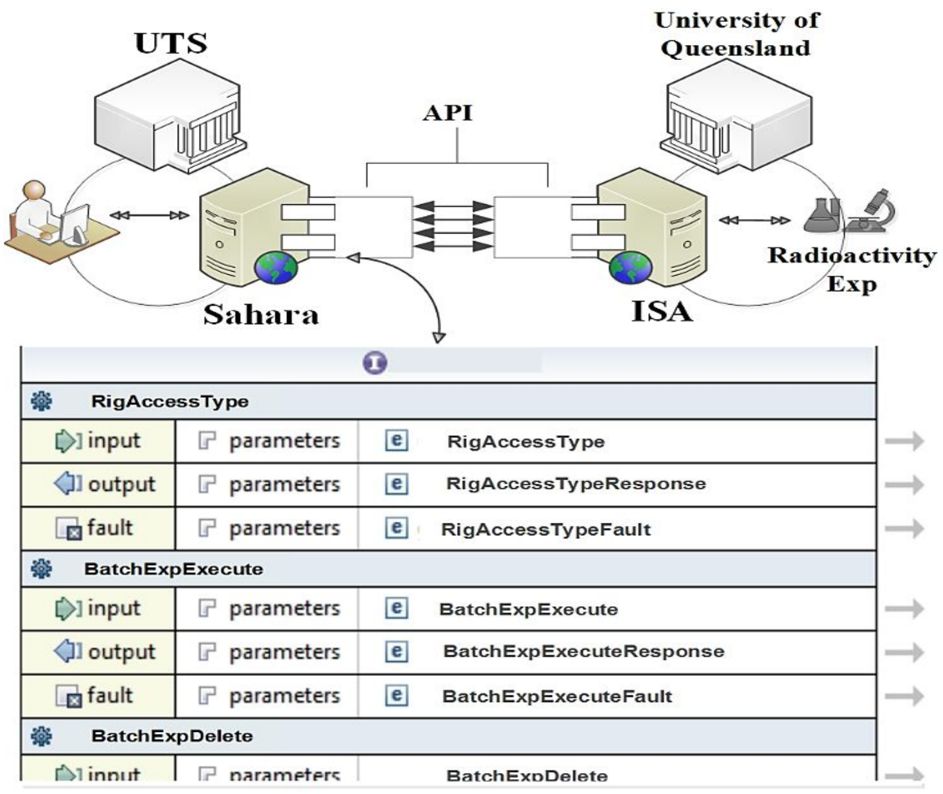
3. Application

Sahara and ISA are considered to be leading extant RLMSs, owing to their successfully implementation and deployment across many universities [23, 24, 31]. Sahara and ISA offer similar functionalities but their architectures are significantly different [16, 32]. ISA offers a relative comparative advantage in batched labs and its distributed architecture, whereas Sahara offers advantages for interactive labs, queuing, and seamless integration of new experiments. There are some elements (such as integration with LMSs) which are not well-supported by either system, but which are provided by other systems (such as LiLa). A comparison between both architectures is provided in Table 2.

Table 2. A comparison between ISA and Sahara.

	ISA (iLab)	Sahara (Labshare)
Developer	MIT	UTS
Web Technologies	Microsoft (ASP.NET, MS SQL, and IIS)	PHP, Java, MySQL, PostgreSQL, and Apache
Compatibility	Only runs on MS Windows Server	Cross-platform
Authentication	Database authentication	Database authentication and LDAP
Client-server communication	TCP/IP	TCP/IP and RDP
Features	➢ user accounts and administrative tools ➢ scheduling ➢ interactive experiments ➢ user tracking ➢ batched experiments ➢ strong support to distributed and federated user account management owing to the functionality of service broker, LSS, USS, and ESS	➢ user accounts and administrative tools ➢ scheduling ➢ interactive experiments ➢ queuing ➢ arbitration of access to multiple identical experiment workbenches
Adding experiments	Complex; Web services API should be design for communication with service broker, LSS, USS, and ESS	Complex; for each experiment GUI should be developed, but simple in case of RDP
No. of universities	3 (Africa), 3 (Australia), 2 (Asia), 5 (Europe), 2 (North America) [33]	5 (Australia) [23]
No. of laboratories added	Estimated to be >20	11 in operation and 12 under development [23]

The goal is to allow both RLMSs to interconnect with each other and with any other RLMS as described in Figure 2. As a final stage we test the proposed pattern by developing an API for connecting two RLMS, Sahara and ISA. In this case, a radio-activity experiment hosted in an iLab server at the University of Queensland has been accessed by a user logging into a Sahara server located at UTS as shown in Figure 4.



RigAccessType		
input	parameters	RigAccessType
output	parameters	RigAccessTypeResponse
fault	parameters	RigAccessTypeFault
BatchExpExecute		
input	parameters	BatchExpExecute
output	parameters	BatchExpExecuteResponse
fault	parameters	BatchExpExecuteFault
BatchExpDelete		
input	parameters	BatchExpDelete

Figure 4. Inter-communication between Sahara and ISA using an API.

In order to create a communications API that supports interoperability between both RLMSs, we begin by considering common functionalities that are understood across both platforms can be categorized into the following areas: lab access, lab and experiment operations (encompassing status and execution), retrieval of results, queuing, and the ability for the laboratory manager or administrative staff to conduct maintenance activities if need be. An iLabs user will use the API interface as though it is an iLabs LabServer. Conversely, a Sahara user will use the API interface as though it is part of the scheduling server interface. The web interface across the API bridge is defined in a Web Service Definition Language (WSDL) file. The full list of Web services is derived from Table 1. An example of the WSDL used can be shown in Figure 5 for the RigAccessType web method. Simple Object Access Protocol (SOAP) was adopted as the messaging protocol based on its common use in many existing RLMSs. There could also be performance and latency issues inherent with the use of web services and SOAP, but these should be able to be addressed by the adoption of optimization techniques such as reducing the size of the SOAP message or the number of SOAP messages that are sent between the systems. However, the proposed API accepts any web service-enabling technology even if such implementation wasn't provided by the RLMS itself by developing an API plug-in a an extention.

```
<xsd:element name="RigAccessType">
  <xsd:annotation>
  </xsd:annotation>
  <xsd:complexType>
     <xsd:sequence>
        <xsd:element name="experimentID" type="xsd:int" maxOccurs="1" minOccurs="1"/>
     </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="RigAccessTypeResponse">
  <xsd:complexType>
     <xsd:sequence>
        <xsd:element name="RigTypeValue" type="tns:RigAccessType"/>
     </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

The radio-activity experiment is registered in the Sahara scheduling server through the API layer and accessed from

Figure 5. Results retrieved from the radio-activity experiment hosted at an ISA server.

the Sahara Web interface as an experiment hosted within Sahara—while being hosted at an ISA server—to fulfill the parameter input values. The experiment is a batch experiment where user only submits the parameter values—such as radioactivity setup, source name, distance, duration and repeat—and retrieves results later when available. The results returned from this experiment submission (obtained by selecting "Experiment Results Available" button) are shown in Figure 6.

```
<experimentResults>
  <timestamp>Mon 12 Jul 2010 11:50:37 PM</timestamp>
  <title>Radioactivity</title>
  <version>3.1</version>
  <experimentId>124</experimentId>
  <unitId>0</unitId>
  <setupId>RadioactivityVsTime</setupId>
  <setupName>Radioactivity over Time</setupName>
  <sourceName>Strontium-90</sourceName>
  <absorberName>None</absorberName>
  <distance>20,40,60</distance>
  <duration>5</duration>
  <repeat>4</repeat>
  <dataType>Real</dataType>
  <dataVector>307,341,320,331</dataVector>
  <dataVector>123,119,129,126</dataVector>
  <dataVector>45,55,53,62</dataVector>
</experimentResults>
```

Figure 6. Results retrieved from the radio-activity experiment hosted at an ISA server.

3.1. User Perception

A recent survey was conducted among the most 10 active RLMS student users at UNED, by number of logins during the last month, who are using a different RLMS based on Moodle and completely anonymous to both SAHARA and iLab in order to avoid biasing. All the students are enrolled in a distance education master degree and they are from different countries (3 from Europe and 7 from Latin America). The students were asked through an online google questionnaire. Only two of them discarded the idea for the following reasons:

- If a system server goes down or fail for any reason, all its associated remote laboratories would not operate and consequently all associated universities that are bridging to this server would be affected.
- Excessive number of communication layers required to tunnel remote laboratories through two systems and consequently performance may be degraded.
- Campus access restrictions on their installed systems, either technical or political.
- Each RLMS was designed from a different perspective and outlook and each provides functions or features that are not supported by the other. Thus, this integration might result in accessing experiments but with limited features.
- Since external provided experiments are designed by different teachers, these experiments might cause some confusion.

The rest of the users however were strongly in favor agreeing on the fact that this integration would allow them to access a wide range of experiments online beyond their institution capabilities and without extra login procedure. They also stated other advantages as follows:

- This integration will allow me to gain an international experience particularly when the external laboratories are provided along with instructions and/or evaluations from external teachers. Interestingly, if the experiment is collaborative and allows users from different universities to work together.

- This integration will be a breeding ground for inter-institutional collaboration and will promote the provision of distributed programs. For instance, an engineering master degree program could be established among different universities each participating with a subject (along with remote lab access if required).

4. Conclusion and Future Works

In this paper a novel API design pattern was proposed in order to address the current lack of communication and interoperability between different RLMSs, which is considered one of the major challenges in the future development and deployment of remote laboratories. Initial results derived from implementing this pattern in the API development was successfully achieved, where a user logging into Sahara server could access a batch radio-activity experiment hosted at an ISA server. These results addressed many features and capabilities identified in the proposed design pattern such as Experiment Query profile, Basic Rig Access profile, and Basic Batch Experiment profile. However, the pattern includes more capabilities since it was devised from a wider scope that could cope with most of the contemporary RLMSs and with the potential future RLMSs as well. Thus, future works will be carried out in order to integrate various types of experiments and to leverage all the capabilities of the pattern for further refining and enhancements. On the other hand, works will also be carried out on specifying the choice of the adopted communication protocol (e.g. considering which of REST, SOAP, JSON, or others might be appropriate), taking into account ease of migration/integration with existing systems, as well, on defining location of WSDL and XSD files, character en-coding, data types and formats, management of security, list of error codes, list of service level attributes, and list of rig set properties.

5. References

[1] A. Hofstein and V. N. Lunetta, "The laboratory in science education: Foundations for the twenty-first century," *Science Education,* vol. 88, pp. 28-54, 2004.

[2] J. E. Corter, J. V. Nickerson, S. K. Esche, C. Chassapis, S. Im, and J. Ma, "Constructing reality: A study of remote, hands-on, and simulated laboratories," *ACM Transactions on Computer-Human Interaction (TOCHI),* vol. 14, p. 7, 2007.

[3] M. Tawfik, E. Sancristobal, S. Martin, R. Gil, A. Pesquera, E. Tovar, *et al.*, "Common Multidisciplinary Prototypes of Remote Laboratories in the Educational Curricula of Electrical & Computer Engineering," presented at the 119th American Society for Engineering Education (ASEE) Annual Conference & Exposition, San Antonio, Texas, 2012.

[4] M. Garcia Valls and P. Basanta Val, "Usage of DDS Data-Centric Middleware for Remote Monitoring and Control Laboratories," *Industrial Informatics, IEEE Transactions on,* vol. 9, pp. 567-574, 2013.

[5] P. Gaj, J. Jasperneite, and M. Felser, "Computer Communication Within Industrial Distributed Environment&#x2014;a Survey," *Industrial Informatics, IEEE Transactions on,* vol. 9, pp. 182-189, 2013.

[6] I. Santana, M. Ferre, E. Izaguirre, R. Aracil, and L. Hernandez, "Remote Laboratories for Education and Research Purposes in Automatic Control Systems," *Industrial Informatics, IEEE Transactions on,* vol. 9, pp. 547-556, 2013.

[7] H. Hassan, J. Martinez-Rubio, A. Perles, J. Capella, C. Dominguez, and J. Albaladejo, "Smartphone-Based Industrial Informatics Projects and Laboratories," *Industrial Informatics, IEEE Transactions on,* vol. 9, pp. 557-566, 2013.

[8] A. Yazidi, H. Henao, G. A. Capolino, F. Betin, and F. Filippetti, "A Web-Based Remote Laboratory for Monitoring and Diagnosis of AC Electrical Machines," *Industrial Electronics, IEEE Transactions on,* vol. 58, pp. 4950-4959, 2011.

[9] A. Melonyan, A. Gampe, M. Pontual, G. Huang, and D. Akopian, "Facilitating Remote Laboratory Deployments Using a Relay Gateway Server Architecture," *Industrial Electronics, IEEE Transactions on,* vol. PP, pp. 1-1, 2013.

[10] C. Bright, E. Lindsay, D. Lowe, S. Murray, and D. Liu, "Factors that impact learning outcomes in Remote Laboratories," presented at the Ed-Media 2008: World Conference on Educational Multimedia, Hypermedia and Telecommunications, Vienna, Austria, 2008.

[11] D. Lowe, S. Conlon, S. Murray, L. Weber, M. de la Villefromoy, E. Lindsay*, et al.*, "Labshare: Towards Cross- Institutional Laboratory Sharing," in *Internet Accessible Remote Laboratories: Scalable E-Learning Tools for Engineering and Science Disciplines*, ed: IGI Global, 2012, pp. 453-467.

[12] T. Richter, D. Boehringer, and S. Jeschke, "LiLa: A European Project on Networked Experiments," presented at the REV 2009: 6th International Conference on Remote Engineering and Virtual Instrumentation, Bridgeport, USA, 2009.

[13] C. Gravier, J. Fayolle, B. Bayard, M. Ates, and J. Lardon, "State of the Art About Remote Laboratories Paradigms - Foundations of Ongoing Mutations," *International Journal of Online Engineering (iJOE),* vol. 4, 2008.

[14] L. Gomes and S. Bogosyan, "Current Trends in Remote Laboratories," *IEEE Transactions on Industrial Electronics,* vol. 56, p. 13, December 2009.

[15] M. Tawfik, E. Sancristobal, S. Martin, R. Gil, G. Diaz, J. Peire*, et al.*, "On the Design of Remote Laboratories," presented at the IEEE Global Engineering Education Conference (EDUCON), Marrakesh, Morocco, 2012.

[16] D. Lowe, S. Murray, E. Lindsay, and L. Dikai, "Evolving Remote Laboratory Architectures to Leverage Emerging Internet Technologies," *Learning Technologies, IEEE Transactions on,* vol. 2, p. 6, 2009.

[17] A. Arodzero, "World Wide Student Laboratory. Description of Project," in *Physics Education*, ed, 1998.

[18] E. Scanlon, C. Colwell, M. Cooper, and T. Di Paolo, "Remote experiments, re-versioning and re-thinking science learning," *Computers & Education,* vol. 43, pp. 153-163, 2003.

[19] C. Rohrig and A. Bischoff, "Web-based environment for collaborative remote experimentation," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, 2003, pp. 2514-2518 Vol.3.

[20] P. Bauer, V. Fedak, and O. Rompelman, "PEMCWebLab - Distance and virtual laboratories in electrical engineering: Development and trends," in *Power Electronics and Motion Control Conference, 2008. EPE-PEMC 2008. 13th*, 2008, pp. 2354-2359.

[21] A. Gallardo, T. Richter, P. Debicki, L. Bellido, V. Mateos, and V. Villagra, "A rig booking system for on-line laboratories," in *IEEE Global Eng. Educ. Con. (EDUCON)* 2011, pp. 643-648.

[22] D. Lowe, S. Murray, L. Weber, and M. d. l. Villefromoy, "LabShare: Towards a National Approach to Laboratory Sharing," presented at the 20th Australasian Association for Engineering Education Conference, University of Adelaide, 2009.

[23] (2013, June). *Labshare - Home*. Available: http://www.labshare.edu.au/

[24] (2012). *MIT iCampus: iLabs*. Available: http://icampus.mit.edu/ilabs/

[25] V. J. Harward, J. A. del Alamo, S. R. Lerman, P. H. Bailey, J. Carpenter, K. DeLong*, et al.*, "The iLab Shared Architecture: A Web Services Infrastructure to Build Communities of Internet Accessible Laboratories," *Proceedings of the IEEE,* vol. 96, p. 20, June 2008.

[26] J. L. Hardison, K. DeLong, P. H. Bailey, and V. J. Harward, "Deploying interactive remote labs using the iLab Shared Architecture," in *IEEE/ASEE Frontiers in Educ.Con. (FIE)*, 2008, pp. S2A-1-S2A-6.

[27] J. Garcia-Zubia, P. Orduna, D. Lopez-de-Ipina, and G. R. Alves, "Addressing Software Impact in the Design of Remote Laboratories," *IEEE Trans. Ind. Electron.,* vol. 56, pp. 4757-4767, 2009.

[28] P. Orduña, J. Irurzun, L. Rodriguez-Gil, J. García-Zubia, and D. López-de-Ipiña, "Reusing requirements among remote experiments for their development and integration under WebLab-Deusto," presented at the Remote Eng. & Virtual Instrum. (REV) 2011, Brasov, Romania, 2011.

[29] (2012, Abril 4). *UniSchooLabS*. Available: http://unischoolabs.eun.org

[30] (2012, January 1). *GOLC* Available: http://online-lab.org/

[31] (2012, May 1). *iLabCentral - The place to share remote online laboratories*. Available: http://www.ilabcentral.org/

[32] H. Yeung, D. Lowe, and S. Murray, "Interoperability of Remote Laboratories Systems," *International Journal of Online Engineering (iJOE),* vol. 6, p. 10, 2010.

[33] (2013, May). *MIT iCampus: iLabs*. Available: http://ilab.mit.edu/wiki