Towards a microRLMS approach for shared development of remote laboratories

Pablo Orduña*, Luis Rodriguez-Gil*, Ignacio Angulo*, Olga Dziabenko*,
Unai Hernandez*, Diego López-de-Ipiña*, Javier Garcia-Zubia†
*DeustoTech - Deusto Institute of Technology
University of Deusto, Bilbao, Spain
Email: {pablo.orduna,luis.rodriguezgil,ignacio.angulo,olga.dziabenko,dipina}@deusto.es
†Faculty of Engineering
University of Deusto, Bilbao, Spain.
Email: zubia@deusto.es

***Abstract*—Educational remote laboratories are a software and hardware tool that allows students to remotely access real equipment located in universities as if they were in a hands-on-lab session. They have been used for almost two decades. And most remote labs use at least a subset of the following features: authentication (verifying who is the user), authorization (granting permissions to laboatories), scheduling (usually a queue or a calendar), user tracking (registering students activities), federation or administrative tools. Systems that provided these features in a unified approach arose, called Remote Laboratory Management Systems (RLMSs). RLMS provide toolkits for making the development of remote labs easier: a remote lab developer uses one of these toolkits and all the features are automatically inherited. Furthermore, new versions of the same RLMS will provide new features. However, sometimes these RLMS do not allow remote lab developers to consume only certain features, implementing the rest themselves. This is a problem when integrating external laboratories, and increments the learning curve. The focus of this contribution is to describe a lighter approach based on multiple coupled small optional services called microRLMS.**

I. Introduction

An Educational Remote Laboratory is a software and hardware solution that enables students to access real equipment located in their institution, as if they were in a hands-on-lab session. There are many kinds of remote laboratories, including fields such as Physics, Electronics, Robotics or Chemistry. Once these laboratories are available through the Internet, it becomes possible to share them with other universities.

With time, several remote laboratory developers had to develop different remote labs of different nature. Instead of starting from scratch when developing these new remote laboratories, they started building software systems that could be reused among these different labs. This way, the development was splitted in two blocks: the laboratories code (e.g., the connection with the real equipment and the logic of the laboratory), and the management code (e.g., authentication, authorization, scheduling, user tracking mechanisms or administrative tools). These systems have been called Remote Laboratory Management Systems (RLMS). This way, a fair degree of shared development of remote laboratories is achieved: if one (or more than one) institution develops a RLMS, other institutions developing their remote laboratories on top of this RLMS will not need to develop those features again and will instead use that RLMS.

However, depending on the design of the RLMS, a set of restrictions is also imposed. For those new labs requiring all these features, this integration might be straightforward. But some RLMS do not easily support that a lab provides its own authentication schema (or simply does require that no system is used), or its own scheduling system. Instead of being provided as a set of independent, optional and easy to bind services, they are usually provided as a monolithic solution.

The focus of this contribution is to explore the drawbacks of this approach, listing examples of laboratories not fitting in some RLMS, and how a microRLMS approach could avoid this, by providing few services which are easy to bind (e.g., connect authorization to authentication), but none required. It also lists the inherited problems of a microRLMS approach.

The paper is structured as follows: Section II introduces the concepts of remote laboratory and remote laboratory management system. Then, Section III briefly describes the development process in some of these systems. Section IV explains the focus of this contribution. Finally, Section V proposes the outline of a potential solution, and Section VI describes the conclusions.

II. Background

This section introduces the concepts of Remote laboratories and Remote Laboratory Management Systems.

A. Remote Laboratories

A remote laboratory is a software and hardware tool that allows students to remotely access real equipment located in the university. Users access this equipment as if they were in a traditional hands-on-lab session, but through the Internet. To show a clear example, Figure 1 shows a mobile low cost robot laboratory described in [1]. Students learn to program a Microchip PIC microcontroller, and they write the code at home, compile it with the proper tools, and then submit the binary file to a real robot through the Internet. Then, students can see how the robot performs with their program through the Internet (e.g., if it follows the black line according to the submitted program, etc.) in a real environment.

978-1-4799-2024-2/14/$31.00 ©2014 IEEE Polytechnic of Porto (ISEP) in Porto, Portugal from 26-28 February 2014
2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV)

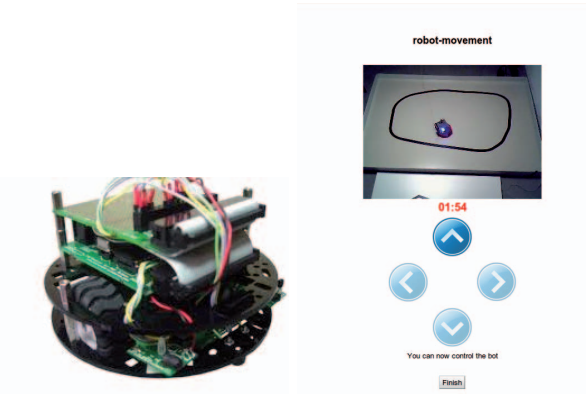

Fig. 1. Robot laboratory [1]. At the left, the mobile robot itself. At the right, the user interface once the program has been submitted.

In this line, there are many examples and classifications in the literature [2], [3]. Indeed, remote laboratories were born nearly two decades ago [4], [5], [6], and since then they have been adopted in multiple fields: chemistry [7], [8], physics [9], [10], electronics [11], [12], robotics [13], [14] and even nuclear reactor [15].

Remote Laboratories have been considered as part of the *Five Major Shifts in 100 years of Engineering Education* in the Special Centennial Issue of the Proceedings of the IEEE [16], in respect to the influence of Information Communications and Computational Technologies.

B. Remote Laboratory Management Systems

Every remote laboratory manages at least a subset of the following features: authentication, authorization, scheduling users to ensure exclusive accesses -typically through a queue or calendar-based booking-, user tracking and administration tools. These features are common to most remote laboratories, and are actually independent of the particular remote laboratory settings. For example, an authentication and queuing system is valid both for an electronics laboratory and for a chemistry laboratory.

For this reason, Remote Laboratory Management Systems (RLMSs) arose. These systems (e.g., MIT iLabs[1], WebLab-Deusto[2] or Labshare Sahara[3]) provide development toolkits for developing new remote laboratories, as well as management tools and common services (authentication, authorization, scheduling mechanisms). The main idea is that by adding a new feature to one of them (e.g., supporting LDAP, or LMSs), all the laboratories which are developed on top of them will support this feature automatically.

One of the features that RLMSs started supporting was federating their remote laboratories. For example, if two universities (*University A* and *University B*) install a particular RLMS, they support federation protocols so *University A* shares a laboratory with students of *University B* without knowing these students. The key here is that the provider university does not need to register particular students, but rather groups or simply universities. It is the consumer system who defines that a set of local users can access a particular laboratory of the provider system.

Therefore, the relationship between two federated entities is the following:

- The consumer system manages the authentication and authorization of its students.
- The provider system manages the scheduling and the access to the laboratories, storing what the users did.
- The consumer system will later ask for results to the provider system.
- In every moment, the provider system does not need to know anything related to the particular students.

The interest on federation of remote laboratories is growing. The Labshare project survey [17], made on all 34 Australian universities offering undergraduate engineering programs, reflects that the interviewed executives were more interested in getting involved for the pedagogic merits of the remote laboratories, and were more inclined on initially being laboratory consumers than providers.

C. Differences with other development tools

In the literature, sometimes RLMS are listed along with different systems which may help in the development of remote laboratories, while they are not purely remote laboratory management systems and therefore are outside the scope of this contribution.

First, equipment such as ELVIS (National Instruments) are sometimes pointed out as toolkits for the development of remote labs. While these systems are clearly useful when developing remote laboratories, since they support remote labs developers in the hardware side, they do not provide any of the features previously listed, neither they intend to do so.

Second, parallel and related efforts have been placed on systems that index remote laboratories located at different institutions such as lab2go [18] or even grant access to laboratories as LiLa [19], [20], [21] (which is closer to a meta-architecture than to an architecture on top of which one can develop remote labs).

III. Analysis of the Remote Laboratories Management Systems development tools

This section aims to illustrate what is the current status of development tools for remote laboratories using Remote Laboratory Management Systems. As described in each case, while they are all extensible through APIs, they all assume more or less that laboratories are created in their particular context, and while they have been forced to open their interfaces to support more laboratories than originally did, it might still be difficult to integrate existing laboratories or to remove features provided by the RLMS.

[1] http://ilab.mit.edu/
[2] http://www.weblab.deusto.es
[3] http://labshare-sahara.sf.net

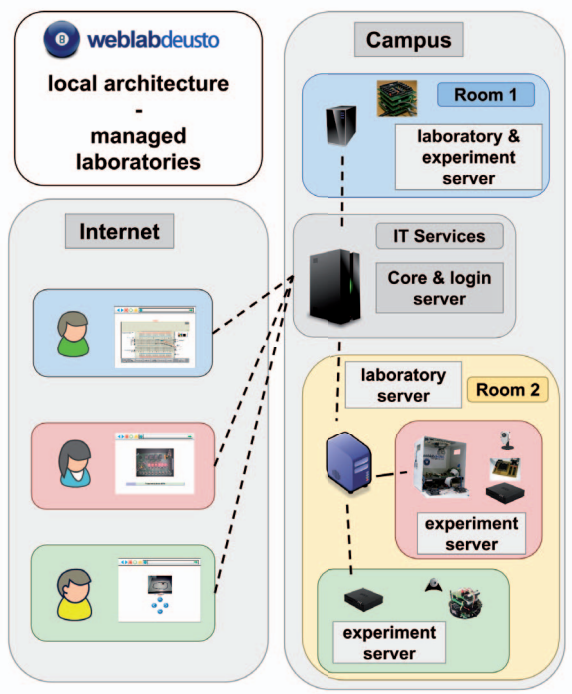

Fig. 2. WebLab-Deusto managed architecture

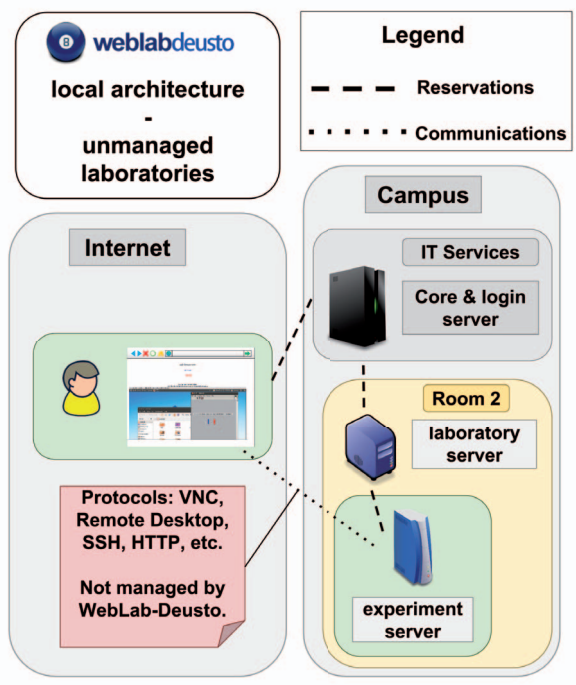

Fig. 3. WebLab-Deusto unmanaged architecture

A. *WebLab-Deusto*

WebLab-Deusto is a RLMS developed in the University of Deusto, used in the University of Deusto as well as in others for developing remote laboratories [22], as well as shared to other institutions (universities and schools). Its architecture has been widely described in the literature [23]. It is divided in two different subarchitectures: one for managed laboratories and the other for unmanaged laboratories, described below.

Managed laboratories

The managed architecture, illustrated in Figure 2, targets those laboratories where all the communications can be managed by WebLab-Deusto. Basically, as shown in the figure, the client submits a set of commands through the WebLab-Deusto Client API, which are finally sent to the final laboratory using the WebLab-Deusto Server API.

This way, the remote lab developer is expected to implement a client that submits messages through commands, as well as a server which receives those commands, processes them and returns a response to the client. For example, a typical client would submit a command such as *turn switch 1 on*, and the server would interact with the equipment to do this task, returning some information in the form of command.

Both APIs are implemented in a number of programming languages. The client API is available for developing in JavaScript, Java Applets or Adobe Flash. The server API is available for developing in Python, Java, C, C++, .NET, LabVIEW and Node.js. This way, it is possible to develop a client in Adobe Flash and communicate it with a server developed in Java using the proper and provided libraries. The administrator can later customize the internal communications to support SSL (encrypting the messages), or define how many copies of the same laboratories are deployed (and the commands will go to the proper laboratory copy). Finally, given that all the commands are submitted through WebLab-Deusto, they are automatically stored for performing analytics.

Unmanaged laboratories

So as to support other types of laboratories, especially previously existing laboratories, a new architecture was developed, called *unmanaged*, and represented in Figure 3.

In this version, while the user still connects to the main servers for authentication, authorization and scheduling, it is later forwarded to other application to use other protocol not based on commands. This includes existing web applications (using HTTP), LabVIEW Remote Panels or Virtual Machines (using VNC and Remote Desktop). WebLab-Deusto does not manage these protocols, so the responsability for securing or storing them relies on the laboratory developer.

Drawbacks

While the target of the managed approach is to make it easy to create new laboratories by managing all the communications and providing multiple libraries for different programming languages, in the end the learning curve required for adapting a new laboratory to it makes it more difficult than enabling the developer to use conventional technologies as in the unmanaged version.

For authentication and authorization, WebLab-Deusto provides several protocols, but it actually requires one of them to be used. It does not support that laboratories are used anonymously. A work around for this is creating a new user with some restrictions (not being able to change the password) for demo users, but this still requires that a user is logged in through WebLab-Deusto. It is not easy to make that a existing laboratory with its own users and authentication mechanisms

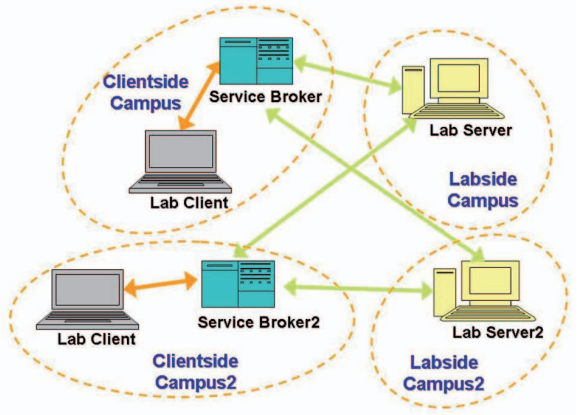

Fig. 4. ISA batch architecture [26]

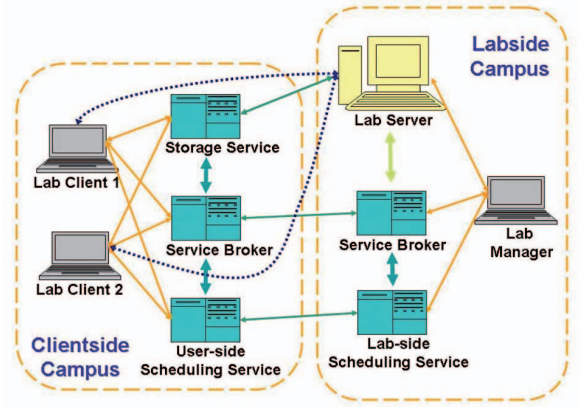

Fig. 5. ISA interactive architecture [26]

uses them. The only way to support this is by using the federation protocols as done in the integration with LMS [24].

Regarding scheduling, it only supports priority queues. This way, for the managed laboratories it does not support any way to use other third party scheduling mechanisms. For the unmanaged laboratories, it is possible to define that they always finish once the protocol is established and rely on the laboratory to manage the scheduling system. This is used for instance in the integration with the FCEIA-UNR laboratories [25].

Finally, while coarse grained user tracking is available in any option (storing who used what laboratory and at what time), fine grained user tracking (storing all the interactions of the user with the laboratory) is only available for the managed laboratories at the time of this writing.

B. iLab Shared Architecture

The iLab Shared Architecture (ISA) [26] has been widely available and use in the five continents for years. It started supporting batch remote laboratories based on queues and it started supporting interactive laboratories with other mechanisms. Both architectures are briefly described below.

Batch labs

The batch architecture is presented in Figure 4. On it, students do not interact directly with the remote equipment, but submit a setup and obtain the results once the remote equipment processes the laboratories.

The design of this architecture was focused on performance. A queue is used, where all the setups submitted by different students in different universities are enqueued and processed. Given that requests to a Lab Server come from different Service Brokers, the scheduling is implemented in the Lab Server.

Interactive labs

While the batch architecture supported a high throughput, in some contexts it was desired that the student could connect to the final equipment directly. For this reason, a new architecture was developed, called interactive architecture and illustrated in Figure 5.

The ISA interactive labs do not support queues, since the time for execution could be long. Instead, it provides calendar-based booking, although it also supports the option of not using any scheduling mechanism and rely on the Lab Server.

Given that the ISA was designed for sharing laboratories with other institutions, this new architecture included a number of new elements. When a user was authenticated, authorized and scheduled through the different layers of the ISA, it could access the final Lab Server. This Lab Server, through a set of bidirectional SOAP requests and processing certain SOAP headers, could check that the user was valid and may store messages in the Storage Service.

Drawbacks

In the batch architecture, given that Lab Servers implement the queuing system, developing a new batch remote laboratory in other technology than .NET under Microsoft Windows. For instance, so as to develop a remote laboratory in Java, the full stack must be implemented from scratch, as done in [27].

In the interactive architecture, the Lab Servers need to process complex SOAP structures in bidirectional messages that would require a non-trivial development if it was attempted to be reimplemented in other development framework. Indeed, in the literature nearly all the interactive ISA labs were developed in .NET or LabVIEW with provided libraries.

For authentication and authorization, the ISA supports a schema that can only be consumed through the web application (and not through web services), making it difficult, as in the previous RLMS, the deployment of an external remote laboratory that delegates on the ISA for access to these mechanisms.

Regarding scheduling, the ISA is highly customizable: in the batch architecture the laboratory implements its own scheduling system that matches the queue-based API, and in the interactive architecture it supports the *no scheduling* mode, where it can rely on the final remote laboratory. Using this feature, it was possible to integrate any WebLab-Deusto remote laboratory inside the ISA [28].

Finally, regarding user tracking, the system provides the ability to store messages directly from the Lab Server as an external service, once the SOAP protocol authentication is processed.

C. Labshare Sahara

The Labshare Sahara architecture [29] is developed by the University of Technology, Sydney, and it has been used by multiple universities under the Labshare project umbrella, including RMIT. The architecture is divided in three types of laboratories, described below and illustrated in 6.

In all the cases, the system provides a complex system that supports both calendar-based booking and queueing at the same time. Depending on the laboratory, the administrator will select one, the other or both. This depends on the particular laboratory and not on the laboratory types explained below.

Peripheral control

In the peripheral control laboratories, the connections between the student and the equipment are direct. As in the case of the unmanaged laboratories of WebLab-Deusto, it is suitable for loading Virtual Machines or using LabVIEW Remote Panels.

Direct or primitive control

In the direct or primitive control laboratories, the communications are managed through the internal layers of Sahara. It is therefore conceptually similar to the managed architecture in WebLab-Deusto, and it is suitable for new laboratories without any control application already developed, and where low latency is not essential.

Batch control

This last type of laboratories are used when there is no direct interaction between the students and the final equipment, but they submit a setup which is executed in the final system.

IV. Discussion

As shown in the previous section, each of the detailed systems provide a set of features, which are in some combinations optional but in most of them they are not. Authentication and authorization are not optional in any of the systems, while there might be work arounds in particular scenarios (such as the demo accounts of WebLab-Deusto). Provided scheduling mechanisms are optional in some of the settings (e.g., unmanaged architecture of WebLab-Deusto, or interactive ISA). However, these two systems (WebLab-Deusto and ISA) do not provide the full scheduling options provided by Labshare Sahara (no laboratory supports calendar-based booking in WebLab-Deusto neither queueing in the ISA interactive laboratories, unless implemented by the Lab Server in both cases).

The important point is that these RLMS have been originally designed for particular labs and have widen their target to become general purpose systems. However, they all require their own APIs, which are often complex to use and require the lab developer to adapt many existing mechanisms to the equivalent in the RLMS. For these reasons, if existing remote laboratory developers try to integrate their remote lab in the RLMS, the process will result in a failure unless a big amount of re-engineering work is done. For this reason, the target of this contribution is to explore a different approach, where RLMS are separated in different pluggable and optional components.

V. Proposed approach

If each of the RLMS described in Section III were splitted in different optional, pluggable components, the learning curve for adopting them in a first step would be considerably lower. Let us analyze each of these features:

- *Authentication:* It should be possible in all these systems to enable the anonymous access by a special user without username and password. This way, external search engines could find these labs even if internally the application is running with an existing user. Laboratories that do not want to use any authentication would not need to manage any protocol, just grant permissions on this special user to their laboratory.
- *Scheduling:* It should be possible to make the scheduling system optional as it is done in ISA interactive, and provide it as an optional service. The ideal optional scheduler would be the one provided by Sahara, where both queues or calendar-based booking are available for all the types of laboratories.
- *Communications and deployment:* It should be possible in all these systems to support laboratories developed in other toolkits. Not only in other programming languages as done in WebLab-Deusto managed laboratories, but also in different web frameworks as in the unmanaged laboratories and Labshare Peripheral control. Requiring a particular technology (such as .NET, Java or Python), makes adoption more complex. Documentation and live examples should exist for a couple of frameworks to make this clear.
- *User tracking:* The storage of what the user should be an optional layer that the laboratory optionally calls, as done in ISA interactive. However, this also arises security issues if the laboratory is not well designed. For instance, it could report that the user did something that did not do. A trade-off must be done in this sense by the remote lab developer.

The microRLMS approach is indeed the summary of these changes on existing RLMS. In the case of WebLab-Deusto, the architecture has been adapted to fully support pure web unmanaged laboratories, as well as no scheduling services built on top of them. However, this enters in conflict with user tracking, since there is no service at the time of this writing for retrieving usage from these labs automatically. Avoiding authentication has not been developed, except for using a demo account, which is a particular permission in the system. This approach would benefit that existing remote laboratories could adopt RLMS in a much simpler way and step by step adopt those features that are interesting, while not adopting those which are not interesting. For instance, a remote laboratory that already counts with a scheduling system designed for that remote laboratory might not benefit from the RLMS system, while still it can benefit from the tools provided by the RLMS.

VI. Conclusions

Nowadays RLMS are solutions that make it easy to develop new remote laboratories, but which still have a gap when

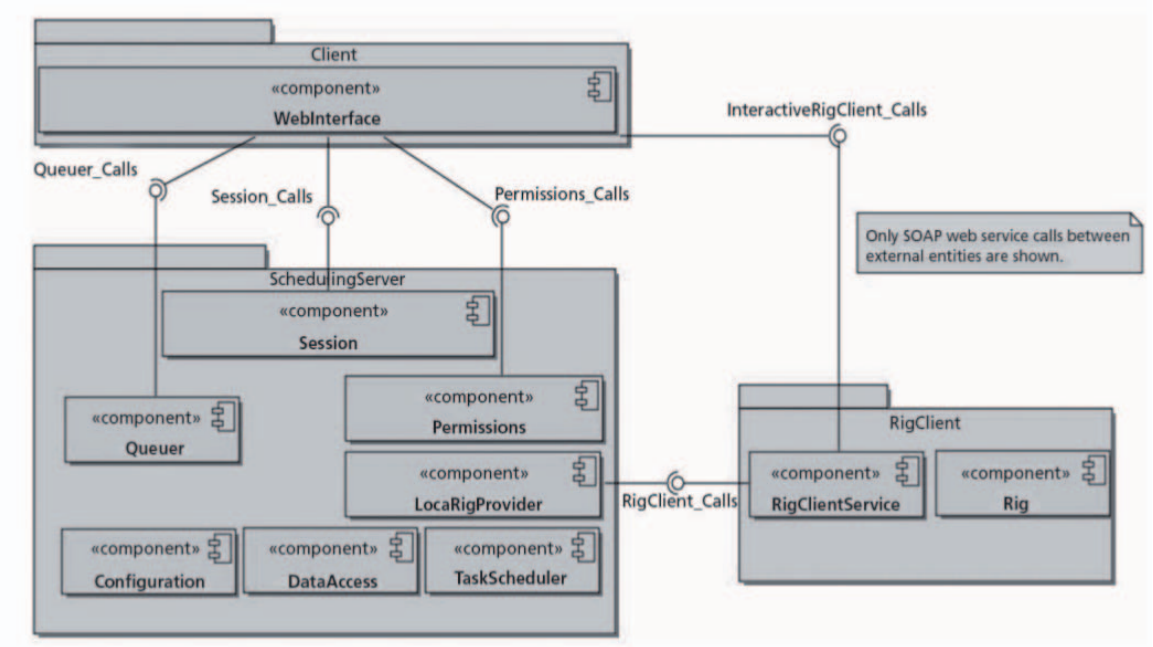

Fig. 6. Sahara architecture [29]

integrating existing remote laboratories. Few examples of interoperability are available in the literature where laboratories can be integrated in an straightforward way [30], [28], [25].

If RLMS were splitted in different optional components that were consumed by lab developers, it would become easier to adopt the RLMS with an existing laboratory and then consider to use the rest of the provided features for other future new laboratories.

REFERENCES

[1] O. Dziabenko, J. García-Zubia, and I. Angulo, "Time to play with a microcontroller managed mobile bot," in *Global Engineering Education Conference (EDUCON), 2012 IEEE*. IEEE, 2012, pp. 1–5.

[2] L. Gomes and S. Bogosyan, "Current trends in remote laboratories," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 12, pp. 4744–4756, 2009.

[3] C. Gravier, J. Fayolle, B. Bayard, M. Ates, and J. Lardon, "State of the art about remote laboratories paradigms-foundations of ongoing mutations," *iJOE*, vol. 4, no. 1, 2008.

[4] B. Carisa, A. Burain, S. Molly H, and C. Lawrence, "Running control engineering experiments over the internet," 1995.

[5] B. Aktan, C. Bohus, L. Crowl, and M. Shor, "Distance learning applied to control engineering laboratories," *Education, IEEE Transactions on*, vol. 39, no. 3, pp. 320–326, 1996.

[6] J. Henry, "Running laboratory experiments via the world wide web," in *ASEE Annual Conference*, 1996.

[7] A. Coble, A. Smallbone, A. Bhave, R. Watson, A. Braumann, and M. Kraft, "Delivering authentic experiences for engineering students and professionals through e-labs," in *Education Engineering (EDUCON), 2010 IEEE*. IEEE, 2010, pp. 1085–1090.

[8] R. Cedazo, F. Sanchez, J. Sebastian, A. Martínez, A. Pinazo, B. Barros, and T. Read, "Ciclope chemical: a remote laboratory to control a spectrograph," *Advances in Control Education–ACE*, vol. 6, 2006.

[9] J. Del Alamo, L. Brooks, C. McLean, J. Hardison, G. Mishuris, V. Chang, and L. Hui, "The mit microelectronics weblab: A web-enabled remote laboratory for microelectronic device characterization," in *World Congress on Networked Learning in a Global Environment, Berlin (Germany)*, 2002.

[10] D. Gillet, H. Latchman, C. Salzmann, and O. Crisalle, "Hands-on laboratory experiments in flexible and distance learning," *Journal of Engineering Education*, vol. 90, no. 2, pp. 187–191, 2001.

[11] I. Gustavsson, J. Zackrisson, L. Håkansson, I. Claesson, and T. Lagö, "The visir project–an open source software initiative for distributed online laboratories," in *Proceedings of the REV 2007 Conference, Porto, Portugal*, 2007.

[12] Z. Nedic, J. Machotka, and A. Nafalski, "Remote laboratory netlab for effective interaction with real equipment over the internet," in *Human System Interactions, 2008 Conference on*. IEEE, 2008, pp. 846–851.

[13] R. Safaric, M. Truntič, D. Hercog, and G. Pačnik, "Control and robotics remote laboratory for engineering education," *International Journal of Online Engineering (iJOE)*, vol. 1, no. 1, 2005.

[14] F. Torres, F. Candelas, S. Puente, J. Pomares, P. Gil, and F. Ortiz, "Experiences with virtual environment and remote laboratory for teaching and learning robotics at the university of alicante," *International Journal of Engineering Education*, vol. 22, no. 4, pp. 766–776, 2006.

[15] J. Hardison, K. DeLong, P. Bailey, and V. Harward, "Deploying interactive remote labs using the ilab shared architecture," in *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*. IEEE, 2008, pp. S2A–1.

[16] J. Froyd, P. Wankat, and K. Smith, "Five major shifts in 100 years of engineering education," *Proceedings of the IEEE*, vol. 100, no. 13, pp. 1344–1360, 2012.

[17] T. Kotulski and S. Murray, "The national engineering laboratory survey," *Labshare Project. December*, 2010.

[18] D. Zutin, M. Auer, C. Maier, and M. Niederstatter, "Lab2go – a repository to locate educational online laboratories," in *Education Engineering (EDUCON), 2010 IEEE*. IEEE, 2010, pp. 1741–1746.

[19] T. Richter, Y. Tetour, and D. Boehringer, "Library of labs-a european project on the dissemination of remote experiments and virtual laboratories," in *Multimedia (ISM), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 543–548.

[20] T. Richter, D. Boehringer, and S. Jeschke, "Lila: A european project on networked experiments," *Automation, Communication and Cybernetics in Science and Engineering 2009/2010*, pp. 307–317, 2011.

[21] V. Mateos, A. Gallardo, T. Richter, L. Bellido, P. Debicki, and D. Villagrá, "Lila booking system: Architecture and conceptual model of a rig booking system for on-line laboratories," *International Journal of Online Engineering (iJOE)*, vol. 7, no. 4, pp. pp–26, 2011.

[22] K. Martin, P. Orduña, J. Garcia-Zubia, M. Fikar, and C. Lubos, "Sharing control laboratories by remote laboratory management system weblab-deusto," 2013.

[23] P. Orduña, J. Irurzun, L. Rodriguez-Gil, J. Garcia-Zubia, F. Gazzola, and D. López-de Ipiña, "Adding new features to new and existing remote experiments through their integration in weblab-deusto," *International Journal of Online Engineering (iJOE)*, vol. 7, no. S2, pp. pp–33, 2011.

[24] P. Orduña, S. Botero, N. Hock, E. Sancristobal, M. Emaldi, A. Pesquera-Martin, K. DeLong, P. Bailey, D. López-de Ipiña, M. Castro, and J. Garcia-Zubia, "Generic integration of remote laboratories in learning and content management systems through federation protocols," 2013.

[25] P. Orduña, F. Lerro, P. Bailey, S. Marchisio, K. DeLong, E. Perreta, O. Dziabenko, I. Angulo, D. López-de Ipiña, and J. Garcia-Zubia, "Exploring complex remote laboratory ecosystems through interoperable federation chains," in *Education Engineering (EDUCON), 2013 IEEE*. IEEE, 2013.

[26] V. J. Harward *et al.*, "The ilab shared architecture: A web services infrastructure to build communities of internet accessible laboratories," *Proceedings of the IEEE*, vol. 96, no. 6, pp. 931–950, 2008.

[27] L. J. Payne and M. F. Schulz, "Java implementation of the batched ilab shared architecture." *iJOE*, vol. 9, no. S3, pp. 4–8, 2013.

[28] P. Orduña, P. H. Bailey, K. DeLong, D. Lopez-de Ipiña, and J. Garcia-Zubia, "Towards federated interoperable bridges for sharing educational remote laboratories," *Computers in Human Behavior*, vol. 30, no. 0, pp. 389 – 395, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0747563213001416

[29] D. Lowe, T. Machet, and T. Kostulski, "Uts remote labs, labshare, and the sahara architecture," *Using Remote Labs in Education: Two Little Ducks in Remote Experimentation*, vol. 8, p. 403, 2012.

[30] H. Yeung, D. Lowe, and S. Murray, "Interoperability of remote laboratories systems," *International Journal of Online Engineering (iJOE)*, vol. 6, no. 5, pp. pp–71, 2010.