2013 FRONTIERS IN EDUCATION CONFERENCE

OCTOBER 23 - 26, 2013 | OKLAHOMA CITY, OKLAHOMA, USA

Frontiers In Education FIE

ENERGIZING OUR FUTURE

PROCEEDINGS

Technical Support:
Conference Catalysts, LLC
Phone: +1 785 341 3583
cdyer@conferencecatalysts.com

IEEE Catalog Number: CFP13FIE-USB ISBN: 978-1-4673-152604

SPONSORED BY:

IEEE

IEEE computer society

ASEE

2013 Frontiers in Education Conference Proceedings

To obtain copies of the USB version of the proceedings, please contact:

Mail IEEE Customer Service Department
445 Hoes Lane
PO Box 1331
Piscataway, NJ 08855-1331 USA

Phone Toll-free 800-678-IEEE (4333) or 732-981-0060

USB Version of Proceedings IEEE Catalog Number and ISBN

IEEE Catalog Number: CFP13FIE-ART
USB version, IEEE Catalog Number: CFP13FIE-USB
ISBN: 978-1-4673-5261-1

Copyright and Reprint Permission:

Online laboratories as a cloud service developed by students

R. Pastor, R. Hernández, S. Ros, D. Sánchez, A. Caminero, A. Robles, L. Tobarra
Communications and Control Systems Department
Spanish University for Distance Education – UNED
Madrid, Spain
{rpastor,roberto,sros,dsanchez,accaminero,arobles,llanos}@scc.uned.es

M. Castro, G. Díaz, E. Sancristobal, M. Tawfik
Electrical and Electronic Engineering and Automatic Control Department
Spanish University for Distance Education – UNED
Madrid, Spain
{mcastro,gdiaz,elio, mtawfik}@ ieec.uned.es

***Abstract*— On-line laboratories (virtual or remote) are widely used in experimental engineering subjects as part of the learning process. In order to develop these laboratories, a development framework called RELATED (Remote Laboratories exTendED) is used by the Communications and Control System Department of the Spanish University for Distance Education of Spain (UNED). This framework defines a structured and methodological development procedure, allowing the students the generation of their own laboratories. Once the laboratory is developed (based in its components), students have to configure their own computing resources in order to make their labs available. However, several problems must be faced by students in the "deployment" of their labs: network configuration, hardware availability, and so on. So, in order to solve these problems, an automatic system based on cloud providers is defined to allow students having their own cloud network/resources for their developed labs. This system simplifies the lab deployment and avoids common errors/mistakes in the development of laboratories with RELATED.**

Keywords—remote/virtual laboratories, development, cloud service model, laboratory deployment

I. INTRODUCTION

On-line laboratories (virtual or remote) [1], [2], [3] are widely used in experimental engineering subjects as part of the learning process [4]. In order to develop these laboratories, a development framework called RELATED (Remote Laboratories exTendED) [5] is used by the Communications and Control System Department of the National Distance Education University of Spain (UNED). This framework defines a structured and methodological development procedure, allowing the students the generation of their own laboratories. Also, students can integrate different elements included in other laboratories provided by the department due to the modular approach in the development. These student's laboratories are considered as part of the evaluation process in several subjects offered by the department in the EHEA (European Higher Education Area).

In the development and laboratory deployment steps, some resources are required in order to use the framework's tools (configuration validation and deployment process application). These resources are based on the availability of a java virtual machine and a correct network configuration (needed for laboratory access). The first software requirement is easy to meet (java versions could be another problem, but it is resolved updating the java distribution) but not the network configuration. Most of students usually use their own hardware (pc, laptops, etc.) with a fixed network configuration (ADSL modems routing ports not available, private ips, etc). So, in order to get a solution for students to resolve the before requirements, a cloud service model [6], [7] can be used to provide software/hardware/network resources. This means that a private/public cloud be used in order to provide these resources, and students could have a cloud panel to have control over their "cloud" deployments.

The "cloud" deployment of remote/virtual laboratories developed using RELATED uses the same elements as the "standalone" deployment, i.e., using the local deployment tool provided by RELATED. These elements are: 1) a configuration file with the lab structure which is XML based; 2) code implementations for "run-able" entities defined in the laboratory (modules and views, in RELATED terminology). With this new deployment option, student gets a full laboratory with an automatic network configuration ready to be used. Thus, lecturers can test the performance of student's labs, in order to evaluate them without running them in their own limited computing resources. It will also prevent the local execution of labs in lecturer's computers as usually it's done in the evaluation process (this simplifies the evaluation process and saves time for lecturers/students).

The paper shows the differences between deployment process using the cloud and standalone approaches, using the same development methodology defined in RELATED. In section II, a brief introduction of development process for remote/virtual labs is presented. The fundamentals of RELATED framework will be presented. In section III the "standalone" deployment process for a virtual/remote lab is reviewed. In section IV, details about cloud configuration and service model will be shown, and how it is managed. Finally, statistical information about lab cloud service used by students will be presented, focusing in two subjects, which are part of Computer Science Degree of the UNED: Distributed systems and Distributed Applications.

978-1-4673-5261-1/13/$31.00 ©2013 IEEE

II. Methodological Development of Virtual/Remote Labs

In order to use the RELATED framework facilities, an RLAB (Remote LABoratory) system must be defined. This can be done using a formal specification which is named LEDML (Laboratory Experimentation Description Markup Language). LEDML is based on XML language, and it is used to define structured components for the laboratory. The laboratory description, LEDML based, is really a lab configuration, and it is used in the same way as other configuration files in other software tools (for example, the server.xml configuration file in Tomcat). Every component in a RLAB XML file is represented by its corresponding XML fragment. These fragments are text based, so they can be reused in a simple way including them in the configuration files of RELATED laboratories.

The main component is an experiment, so lecturers/students can run these experiments using RELATED facilities as, painting data from the laboratory, change any relevant variable or have a look to the laboratory structure. In order to integrate the laboratory as an RLAB system, researchers/lecturers/students only have to develop local access to equipment using Java technology. The element responsible for carrying out this function is called a module.

A module is a run-able entity, executed into the RELATED server facilities. The module is responsible to get/set data from/to the equipment. They can be seen like a "black box" used to describe a component as a set of input/output variables. Considering modules are "black boxes", and using the RELATED modular architecture, it is possible to flow data between modules. Developers must provide read/write functions to get/set data for these variables. Java programming language is used to develop these read/write functions and XML tags are used to set-up a laboratory and to specify the module configuration.

There are also other main components in a RLAB LABORATORY, the views. Views are responsible of human interaction with the lab and provide the lab GUI. As it can be seen in figure 1, modules and views are the main components of an experiment. One or several experiments compose a laboratory (RLAB). One of the most interesting features in RELATED is that RELATED uses the "module paradigm" and, this way, modules and views can be reused in several experiments, even in several laboratories distributed on different Internet locations (for example, laboratories developed by students that can have modules running in the own student's equipment).

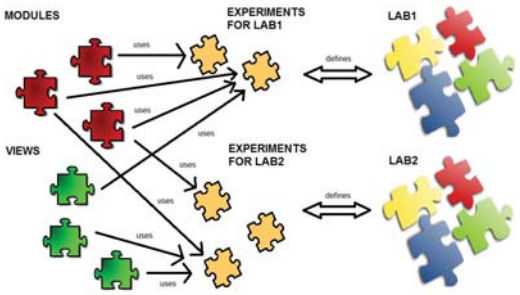

Fig. 1. RELATED modular structure.

As mentioned above, views provide the visual information to the final user (students, teachers, etc.). So, graphical interfaces (GUI, Graphical User Interface) can be included in these views in order to get a better user experience. These views use data from modules to update the experiments state, and it is possible to change and send values from the GUI to the modules.

Experiments define the behavior expected from the lab, so one or more experiments can be defined on the lab. Basically, it is possible to define an experiment by stating the set of modules and views that will be used in the experiment. Any convenient combination of modules and views can be used as an experiment.

III. Standalone Deployment for the Student's Lab

Once the lab definition and components are available, the next step is to deploy/run the lab. In this study case for Distributed Systems and Distributed Applications subjects, the student has to develop a virtual lab [8] representing a simple signal generator with three signal types (Sine, square and triangle) which will be connected (the module component) to a real equipment (magnetic levitator). First, student develops the module representing the behavior of signal generator (module named "SG_MODULE" in figure 2) and a view to get a visual representation of the module. Both of them are developed following the RELATED guidelines. In order to test the generated lab, an experiment is defined in the lab specification file (experiment "Generate signals" in figure 2). Once the specification file is ready (and also the components), students have to deploy their labs using the RELATED tools.

```
<?xml version="1.0" encoding="UTF-8"?>
<system description="Signal generator for SINE, SQUARE and TRIANGLE modes WITH GLG Generated View" type="0"
name="Rafael Pastor Vargas: GUI BASED SIGNAL GENERATOR VIRTUAL Laboratory">
  - <module name="SG MODULE">
      Signal generator module for SINE, SQUARE and TRIANGLE modes
      <param type="double" name="SampleTime" value="0.01"> Sample time for sigmal generator thread</param>
      <var type="double" name="amplitude" units="N/A" min="0" max="10" initial="1">Signal amplitude</var>
      <var type="double" name="frequency" units="Hz" min="0" max="100" initial="1">Signa frequency (Hz)</var>
      <var type="string" name="wave" units="N/A" min="0" max="1" initial="SINE">Signal type</var>
      <var type="double" name="output" units="N/A" min="-10" max="10" initial="0">Generator Signal output</var>
      <var type="double" name="time" units="Seconds" min="0" max="10" initial="0">Generator Signal generated time</var>
      <implementation type="JAVA" classname="es.uned.scc.grados.appdist.related.modules.SignalGeneratorModule"
          helperUrls="../examples/signal_generator/code/SignalModel.jar"
          jarfile="../examples/signal_generator/code/RLABSignalGeneratorModule.jar">Module for generation
          signal</implementation>
  </module>
  - <view name="SIGNAL_GENERATOR_GLGVIEW" classname="es.uned.scc.rlab.views.signalgenerator.SignalGView"
  helperUrls="../examples/signal_generator_gui/code/GlgCE.jar,../examples/signal_generator_gui/code/GlgGraphLayout.jar"
  jarfile="../examples/signal_generator_gui/code/SignalGeneratorGLGView.jar">
      Show virtual view
      <use name="amplitude" as="amplitude" module="SG MODULE"/>
      <use name="frequency" as="frequency" module="SG MODULE"/>
      <use name="wave" as="type" module="SG MODULE"/>
  </view>
  - <experiment name="Generate signals" concurrentUsers="1" slotTime="5" logging="no" sampleTime="10">
      Test
      <duration type="time" time="300"/>
    - <run module="SG MODULE">
        <interactives show="true,true,true" names="amplitude,frequency,wave"/>
        <paint names="output" colors="black"/>
        <paint names="time" colors="blue"/>
      </run>
      <open view="SIGNAL_GENERATOR_GLGVIEW"/>
  </experiment>
  <manager name="rpastor"/>
  <student name="demo"/>
</system>
```

Fig. 2. Signal generator's lab definition using LEDML.

To do the lab's deployment, student has to configure a properties file which defines several hardware/software/network parameters (See figure 3):

- Local path of lab's definition.
- Local path of logs generated by the RLAB server which is responsible of the laboratory's execution.
- Network configuration for the RLAB server (the lab) which will be run on the local host of student.

The network configuration parameters are especially important due to the laboratory will listen client connections on

the defined ports. So, if these ports are not available to external clients, there will no opportunity to run experiments from the experimentation client. Usually, students use their own personal computers to develop the laboratories, and these personal computers have private ip's. This is a big problem to lecturers and students in order to check the online laboratory, due to lack of visibility of private ip's on the internet network.

```
# Properties file for initiating values for server

# XMl file with LEDM Specification
xmlfile=../examples/signal_generator_gui/LEDML_Specification/signal_generator_gui.xml
#Show GUI (values --> yes, no )
show=yes
#Debug (values --> yes, no). If yes, output will be redirected to log/rlab.log
debug=yes
#log_file
log_file=./log/signal_generator_gui/rlab_signalgeneratorGUI.log
# Publish (values --> yes, no). If yes, rlab system is publish as public from
# rlab website, once the LEDML file is parsed
publish=no
# Rmi port, port for start rmi registry
rmiport=1099
#RMI Service port. RMI service port for an RLAB system --> It is different of rmiport
(assigned to naming service)
rlabserviceport=1095
#UDP port. Port for data communication
udpport=10003
# REST Server port assigned. Jetty needs a port to listen REST request
restport = 9996
```

Fig. 3. Properties file needed for run a RELATED laboratory.

Once the properties file is configured, then the student has to run the publish application tool provided by RELATED. This application checks the xml definition and creates the laboratory using the parameters defined before. Now, students can carry out the experiments defined in the lab specification from the related web server, showing the experimentation client like in the figure 4.

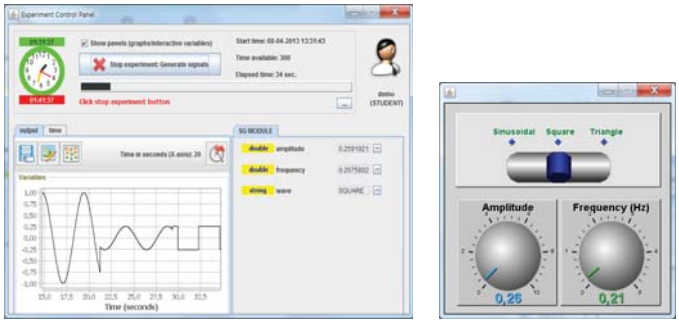

Fig. 4. Experiment client running the experiment defined in the student's laboratory.

Students claim that they waste so much time in a correct configuration for the parameters for their computing nodes (their personal computers). This hide the main objective of the evaluation task: the development of their own laboratories. So, the first problem to face is the simplifying of lab's configuration process. Also, another big problem consists in to get a public availability of the lab, avoiding the use of private ip's. This availability must consists not only in terms of public access across Internet, but using computing resources (computing nodes) to get a "full time" availability of lab. This feature allows to lecturers and students the testing of their lab, and in the lecture's case the evaluation of the tasks associated to the developed lab by students.

IV. USING CLOUD PROVIDERS TO DEPLOY RELATED BASED LABS

In order to solve the problems detected in the above section, it's mandatory to get computing resources using public/private available resources. For this objective, the well-defined cloud service model [9], [10] (see figure 5) can be used. The idea is to use the public/private resources defined in several cloud providers to get "computing nodes" which will run the student's labs [11]. Nowadays, RELATED was using the cloud model for the management services [12] but not for the lab's execution. To move the lab's execution to the cloud, the PaaS (Platform as a Service) layer of cloud computing model is used to develop a new "deployment" application for RELATED laboratories named "RELATED Cloud Labs". The PaaS layer focus on applications and the "RELATED Cloud Labs" application provides the lab's deployment services for users, as it is required in this case.

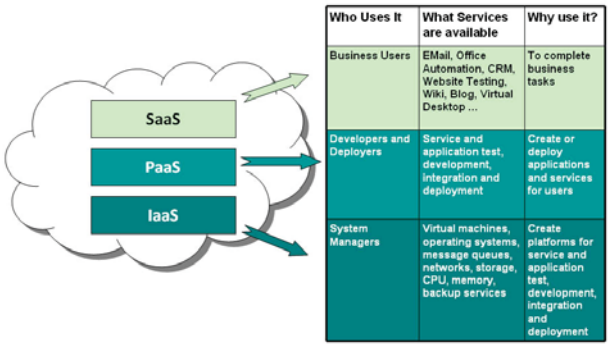

Fig. 5. Cloud service model. Referenced from http://info.apps.gov/content/what-are-services.

This application represents a computing node, and it will take care of the student's laboratories, running them inside the "computing node". The computing node will be represented as a Java Web Application running inside a Java EE 6 container. Thus, any compliant PaaS provider can be used to create "computing nodes". There are several options for public PaaS providers which offer this kind of environment, hosted in the cloud. Actually, three nodes are available for students in order to be used:

- http://cloudlabs.rpastorvargas-uned.cloudbees.net. Running in the public PaaS provider name "Cloud Bees" [13].
- http://cloudlabs-rpastor.rhcloud.com/. Running in the public PaaS provider name "Openshift" [14].
- http://lab-app.scc.uned.es:8080/CloudLabsDeployer/. Running in a private node owned by RELATED project.

The process of adding "computing nodes" is quite simple: installation of the "RELATED Cloud Labs" application in a cloud provider with PaaS/J2EE support. Thus, to get more computer nodes is easy and other PaaS providers like CloudFoundry [13] or AWS Elastic Beanstalk [14] will be added as computing nodes providers in the future.

Students can select the "computing node" to deploy their labs from the RELATED web server. Even, they can deploy and run the same laboratory in several nodes (up to three now) to test the performance of the computing node.

V. Details about the Deployment Process in the Cloud

Once the student selects the computing node, he/she has access to the web application (RELATED Cloud Labs). This application allows to student the uploading of their xml definitions for labs, and also the components which compose the labs (basically the implementing jar files of views/modules). In figure 6 is represented the main window for the web application (with the student "demo" logged in the system) running in the computing node provided by Cloud Bees. From this window (a web page in a browser), the student can view their cloud labs deployments and upload new laboratories using the corresponding options.

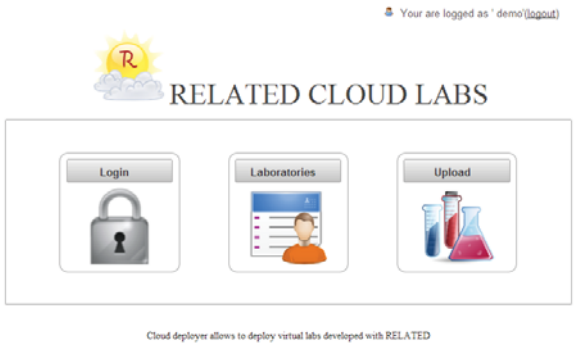

Fig. 6. Main form for the computing node hosted by CloudBees.

In order to upload a new laboratory, students simply click the "Upload icon" (see figure 6) and a new form will be presented to upload the lab's definition. This form is presented in figure 7. The first step is the uploading of the xml file and its validation. The validation is used to detect components and definitions errors in the laboratory specification.

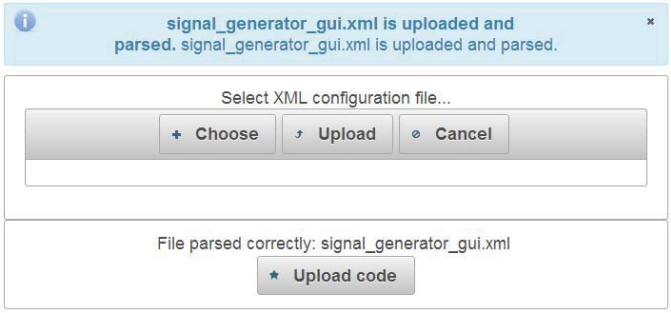

Fig. 7. Uploading and verifying of laboratory specification.

If the laboratory's definition is ok, the "Upload code" button is enabled in order to allow to students to upload components defined in the lab, as it seen in figure 8. In this case, the student's lab is composed by one module representing the signal generator and a view to interact with the lab. As mentioned before, these components are developed using the RELATED methodology. Every component must have at least one main jar (implementing the behavior of component) and optionally, one or more "helper" jar files. These helper files allow including third part libraries in an easy way.

In order to upload lab components, the "Choose" button must be clicked to upload all the implementing jar files. Students can upload all the files in one step, using the multiples items capability of the upload component (see figure 9). Once all the files are uploaded, the lab is ready to be added to the student's cloud labs using the corresponding button (enabled when all files are uploaded and checked) in figure 10.

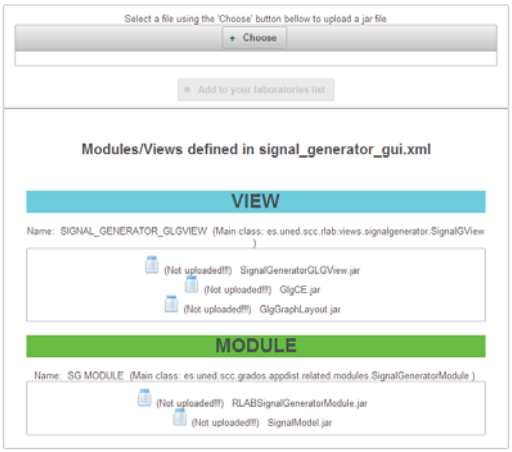

Fig. 8. Uploading laboratories components: modules and views.

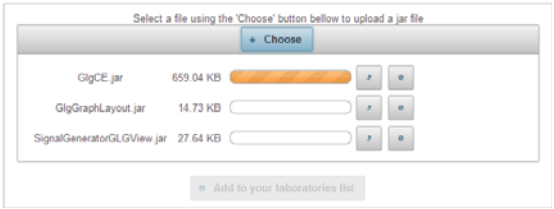

Fig. 9. Uploading multiple laboratories components at the same time.

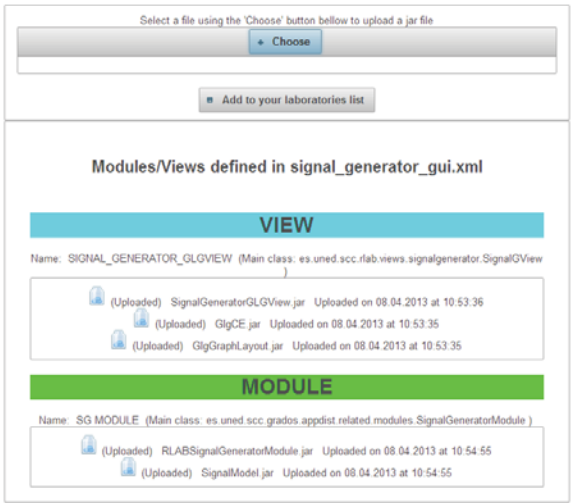

Fig. 10. Laboratory components uploaded and ready to be deploy/run in the cloud.

Finally, once the laboratory is ready, it will be available in the laboratory list (see figure 6). Now, students can get details about their clouds labs, like it is shown in figure 11. In this case, the student's lab being deployed is shown in figure 11 (marked with a red box). From every cloud lab defined, the name and description are presented and also, the deployment options and management options.

Fig. 11. Laboratories deployed on a computing node.

The last step to get the public availability of lab consists in changing the state of cloud lab. This is done using the icon. Clicking this icon, it starts the running of the laboratory in the computing node, making available from the laboratory list in the related web server. At the start of lab running, the computing node assign automatically the parameters for the configuration of a RELATED laboratory (see figure 3), based on the availability of resources in the computing node (network ports and paths to logs for laboratory). This feature allows students and lecturers forget all about the configuration file and focus on the testing of the laboratory.

In figure 12, the running laboratory list in related web server [17] is presented. In this figure, two laboratories with the same name are showed, one running using the "standalone" deployment process and the same running on the cloud.

Fig. 12. Laboratories list available from the related web server.

Students can manage their cloud labs from the details panel of lab. They can delete (undeploy) the laboratory using the icon or edit the components in order to upload new versions of the components (using the icon).

VI. STUDENT'S CLOUD LABORATORIES AND SURVEYS

Once the practical experience with virtual/remote laboratory is over, students must perform a survey used to get development/services satisfaction information. This survey is mandatory and it has 20 questions, classified in 5 main areas (questions/ratings are shown in Table 1 only for the main interest area of this paper, focused on deployment options). The survey was performed by 45 students from 151 enrollment students in the two subjects considered. Due to the university's regulations, students can deliver the practical homework in two dates (February 2013 and September 2013), so only the first date surveys are included in this paper.

TABLE I. QUESTIONS AND RATING

Question (scored 1-5)	Code	Average rating
Deployment options (A4)		
Consider the two deployment options for your labs; do you consider that cloud options have a lot of advantages over the standalone deployment?	Q17	*4.9*
Consider the cloud deployment option; do you think is easy?	Q18	*4.8*
Consider the standalone deployment option; do you think is easy?	Q19	*2.7*
In the standalone deployment process, have you had a lot of problems with the network parameters?	Q20	*3.2*

All students have to do both deployments: "standalone" deployment in order to debug the code associated to the laboratory's component. Once the laboratory is fully tested and debugged, they have to deploy their labs to the cloud in order to be evaluated with no need of sending to lecturers their solutions.

In table I, it can be seen as all the students prefer the cloud deployment, due to the easy procedure defined for the uploading of laboratories and their execution without the use of their own resources. Also, they complain about configuration process in the "standalone" deployment process, but in general they are satisfied with both deployment options.

VII. CONCLUSIONS

Components of a laboratory in RELATED are defined by XML fragments and its corresponding implementing files, so the "deployment process" must deal with these components in an easy way. In the "standalone" deployment process, a previous configuration step is needed in order to run the laboratory, complicating the testing of laboratories for students and lecturers. Also, the testing of a RELATED laboratory implies the use of own hosted hardware/software/network services.

To solve these problems, and to get an automatic configured laboratory and computing resources related, it is introduced the concept of "computing node". This computing node represents an application running in the cloud, using the PaaS layer of the cloud service model. Using this concept, it is possible to use public PaaS providers like Cloud Bees to have

computing nodes running RELATED laboratories in a simple way.

The use of options like the automatic configuration and cloud deployment, allows students to run their own laboratories using external computing resources. Also, lecturers avoid testing their student's laboratories in their own personal computers, getting a better experience in the evaluation of the tasks associated to the laboratories' development.

ACKNOWLEDGMENT

The Authors would like to acknowledge the support of the following European Union projects: RIPLECS (517836-LLP-1-2011-1-ES-ERASMUS-ESMO), PAC (517742-LLP-1-2011- 1-BG-ERASMUS-ECUE), EMTM (2011-1-PL1-LEO05- 19883), MUREE (530332-TEMPUS-1-2012-1-JO-TEMPUSJPCR), and Go-Lab (FP7-ICT-2011-8/317601). Furthermore, we thank Spanish Ministry of Science and Innovation for the Project TIN2008-06083-C03/TSI and the Region of Madrid for the support of E-Madrid Network of Excellence (S2009/TIC-1650).

REFERENCES

[1] L. Gomes and S. Bogosyan, "Current Trends in Remote Laboratories," Industrial Electronics, IEEE Transactions on, vol.56, pp. 4744-4756, 2009.

[2] E. G. Guimaraes, E. Cardozo, D. H. Moraes, and P. R. Coelho, "Design and Implementation Issues for Modern Remote Laboratories," Learning Technologies, IEEE Transactions on, vol. 4, pp. 149-161, 2011.

[3] E. Sancristobal, M. Castro, S. Martin, M. Tawkif, A. Pesquera, R. Gil, G. Diaz, and J. Peire, "Remote labs as learning services in the educational arena," in Global Engineering Education Conference (EDUCON), 2011 IEEE, 2011, pp. 1189-1194.

[4] A. Nourdine, R. Pastor, G. Vivas, "Limitations of remote laboratories in control engineering education". International Journal of Online Engineering, on vol. 6, pp. 31-33, 2010.

[5] R. Pastor, R. Hernández, S. Ros y M. Castro, "Methodological specificacion of implementation and development of experimental environments", Latin-american Learning Technologies Journal, RITA, on vol. 1, pp. 27-35, 2006.

[6] Walz J., Grier D.A.: "Time to Push the Cloud", IT Professional, on vol. 12, Issue:5, pp. 14-16, 2010

[7] Dikaiakos M.D., Katsaros D., Mehra P., Pallis G, Vakali A.: "Cloud Computing: Distributed Internet Computing for IT and Scientific Research", IEEE Internet Computing, on vol. 13, Issue: 5, pp. 10-13, 2009

[8] Pastor, R; Sanchez, D; Nourdine Aliane, Roberto Hernández, Antonio Robles-Gómez, Agustín Caminero, Salvador Ros, Gabriel Diaz, Manuel Castro; "Practical experiences on building structured remote and virtual laboratories from the student's point of view" Frontiers in Education Conference, 2012.

[9] Diez, O., Silva A.; "Govcloud: Using Cloud Computing in Public Organizations", IEEE Technology and Society Magazine, on volume 32, issue: 1, pp. 66- 72, 2013.

[10] http://info.apps.gov/content/what-are-services, date of last access: April 8, 2013.

[11] Correia, R.C., Fonseca, J.M.; Donellan, A.; " Euronet lab a cloud based laboratory environment". Global Engineering Education Conference (EDUCON), 2012.

[12] D. Sánchez, A. C. Caminero, R. Hernández, R. Pastor, S. Ros, A. Robles-Gómez, L. Tobarra; "On the use of cloud technologies to provide remote laboratories as a service". WCLOUD 2012, 1st International Workshop on Cloud Education Environments, 2012.

[13] http://www.cloudbees.com/, date of last access: July 5, 2013.

[14] http://www.openshift.com/, date of last access: July 5, 2013.

[15] http://www.cloudfoundry.com/, date of last access: July 5, 2013.

[16] http://aws.amazon.com/es/elasticbeanstalk/, date of last access: July 5, 2013.

[17] http://lab.scc.uned.es/related/, date of last access: July 5, 2013.

AUTHOR INFORMATION

Rafael Pastor, Associate Professor at the Control and Communication Systems Department of Spanish University for Distance Education, UNED, IEEE Senior Member, rpastor@scc.uned.es

Roberto Hernández, Associate Professor at the Control and Communication Systems Department of Spanish University for Distance Education, UNED, IEEE Senior Member, roberto@scc.uned.es

Salvador Ros, Associate Professor at the Control and Communication Systems Department of Spanish University for Distance Education, UNED, IEEE Senior Member, sros@scc.uned.es

Daniel Sánchez, Researcher at the Control and Communication Systems Department of Spanish University for Distance Education, UNED, dsanchez@scc.uned.es

Agustín C. Caminero, Assistant Professor at the Control and Communication Systems Department of Spanish University for Distance Education, UNED, IEEE Member, accaminero@scc.uned.es

Antonio Robles-Gómez, Assistant Professor at the Control and Communication Systems Department of Spanish University for Distance Education, UNED, IEEE Member, arobles@scc.uned.es

Llanos Tobarra, Assistant Professor at the Control and Communication Systems Department of Spanish University for Distance Education, UNED, IEEE Member, llanos@scc.uned.es

Manuel Castro, Full Professor at the Electrical, Electronic and Control Department of Spanish University for Distance Education, UNED, IEEE Fellow Member, mcastro@ieec.uned.es

Gabriel Diaz, Associate lecturer at the Electrical, Electronic and Control Department of Spanish University for Distance Education. IEEE Member, gdiaz@ieec.uned.es.

Elio San Cristóbal, Assistant Professor at the Electrical, Electronic and Control Department of Spanish University for Distance Education, UNED, IEEE Gold Member, elio@ieec.uned.es

Mohamed Tawfik, Researcher at the Electrical, Electronic and Control Department of Spanish University for Distance Education, UNED, IEEE Member, mtawfik@ieec.uned.es