Cloud-based remote laboratory supported by RTAI

Zoltán Janík and Katarína Žáková

***Abstract*—**The paper presents an online remote laboratory environment that utilizes the cloud computing model. The core of the remote laboratory is based on the real-time server using the features of the open-source RTAI project (Real-Time Application Interface). We have enhanced the possibilities of RTAI-based systems and developed a new unified interface for remote access to such systems using the philosophy of the Platform as a Service and the Software as a service cloud computing models. The described solution enables users to design custom schemes and compile those remotely using provided web services. Thus, the new functionality creates an environment that allows not only the remote execution of pre-defined real-time tasks like in standard SaaS clouds, but it offers the ability to create custom tasks remotely as well, following the principles of the PaaS model. Simple integration into existing web applications is ensured by using WSDL (Web Services Description Language) and SOAP (Simple Object Access Protocol) technologies.

***Keywords*—**computer aided engineering, control education, on-line control, real-time tasks.

I. Introduction

MANY experts agree that the cloud computing is going to be a standard technology used for the provision of IT-based services both in business and educational sector[11], [12], [16], [18]. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned or released with minimal management effort or service provider interaction [15]. Clouds have many advantages, such as elimination of client-side installation expenses, measurable pay-per-use approach, system independency on the client's side, reduction of malware and virus infection risk and seamless continuous updates of provided applications. On the other hand, they may suffer from performance decrease affected by network connection speed and quality, security risks during data transfer and complete dependence on network and the cloud services provider. [18]

We present a new solution of deployment of remote laboratories for teaching of automation and control. Remote and virtual experiments in education process are constantly gaining higher importance, therefore many universities put great effort in development of remote and virtual laboratories. Institute of Automotive Mechatronics at Faculty of Electrical Engineering and Information Technology, Slovak University of Technology is not an exception. The most common solution is on-line access to the systems that are accessible for students during standard daily courses. In this way, each system acquires the added value of system availability besides the time that is reserved for teaching. By adding more features to web-based laboratories, on-line experiments gain higher level of interactivity and extended competitiveness to standard hands-on experiments of daily courses.

This work was supported by the grant "Program for support of young researchers" of Slovak University of Technology. It has also been supported by the Slovak Grant Agency, Grant KEGA No. 032STU-4/2013 and APVV-0343-12.

Z. Janík and K. Žáková are with the Institute of Automotive Mechatronics of Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Bratislava, Slovakia (e-mail: zoltan.janik @ stuba.sk, katarina.zakova @ stuba.sk).

II. Cloud Computing Essentials

The concept of cloud computing offers several characteristic features [15]:

- *on-demand self-service* – a consumer can use the computing capabilities immediately as needed, without requiring human interaction with each service provider;
- *broad network access* – capabilities can be used over the network and they can be accessed through standard mechanisms such as thin or thick clients;
- *measured service* – cloud systems can monitor, control, optimize and report the resource usage providing transparency for both the provider and the consumer of the service.

In cloud computing, we can consider several service models. Among rare models such as Storage as a Service, Security as a Service, Application as a Service or Desktops as a Service, the most common cloud computing models are [15]:

- *Software as a Service (SaaS)* - the capability provided to the consumer is to use the provider's applications running on a cloud infrastructure,
- *Platform as a Service (Paas)* – the provided capability is to deploy onto the cloud infrastructure consumer-created or acquired applications.
- *Infrastructure as a Service (IaaS)* – the provided capability is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software,

The three basic cloud models are visualized in Fig. 1 that shows a level of the service provider's and consumer's reach.

Regarding the deployment scheme, we can consider following cloud types: private cloud, community cloud, public cloud and hybrid cloud. These cloud types differ in consumer's access to the cloud's infrastructure, whether it can be accessed by single organization comprising multiple consumers, the community of consumers sharing the same concerns, the general public, or the combination of these types.

Given the cloud service models described above, the proposed cloud laboratory solution cannot be clearly classified since it shows signs of SaaS and PaaS service model types. On the one hand, the consumer can use the laboratory capabilities in a form of a Software as a Service model (using standard predefined laboratory experiments), but on the other hand, we have enhanced the current state-of-the art of the remote environment and we provide our remote laboratory platform in order that consumers can affect the experiments by using their own algorithms as defined in the Platform as a Service model. Nevertheless, the deployment scheme of our solution can be clearly considered as the hybrid cloud as it bears the signs of both private and public clouds (consumers may get free access to private services, though some services may be available only for certified users).

III. Application of Cloud technology to a Remote Laboratory Environment

We have focused on development of unified solution that extends standard on-line laboratories to be able to provide real-time control and remote access by using PaaS (Platform as a Service) model. The developed solution offers a package of modules that can be integrated to systems that realize remote and virtual experiments and provide on-line access to laboratory infrastructure. The most important module is used for remote access to experiments using hard real-time control with enhanced interactivity that exceeds the available alternative solutions.

On-line laboratories that contain tools for real-time based experiments (e.g. Simulink Coder, also known as Real-Time Workshop in Matlab, Scicos-HIL and other alternatives) depend on host operating system. Thus, in addition to the real-time support in applications, the real-time support in operating system is also necessary. General purpose operating systems such as Windows, Mac OS or standard Linux distributions do not match this requirement [9]. For this reason, it is necessary to use operating system with special modified kernel. One of the possible choices among open-source solutions is Linux-RTAI (Real-Time Application Interface) kernel that supports task execution in hard real-time mode. The RTAI kernel is a part of the Real-Time Suite that will be discussed in next

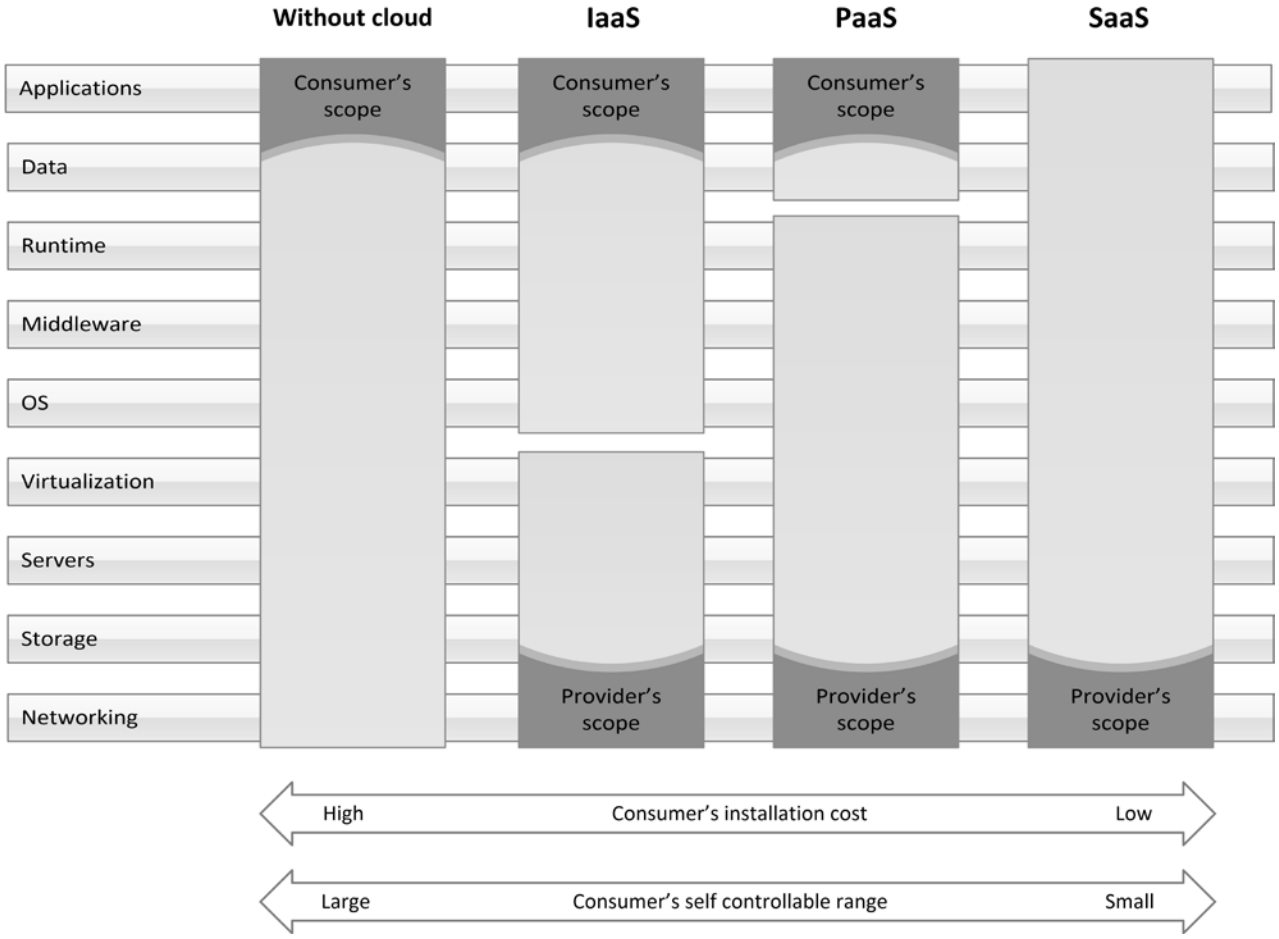

Fig. 1 Comparison of basic cloud models

section.

Our solution covers a gap in current state-of-the-art of the Real-Time Suite. Whereas there is a way to access the real-time task remotely using RTAI-XML server extension, the remote real-time task has to be provided in advance by the system administrator. This task was fully automated and it was delegated to the web application that runs within the real-time server powered by RTAI platform. Referring to available sources, such solution has not been deployed on any similar project of remote laboratory.

The architecture of the remote laboratory environment can be seen in Fig. 2. The diagram illustrates the laboratory environment from two different perspectives – the application server and the real-time server. Each one of them is divided into several separate layers that are dedicated to provide basic partial functions.

Starting from the bottom to top, the presented remote laboratory environment utilizes an operating system layer that is used for general functionality of both server environments. In the case of the real-time server, the operating system layer is further divided into real-time operating system and general-purpose operating system. The application server does not have to use real-time operating system as the control algorithm is always executed on the real-time server.

The middle layer of application server is reserved to provide functionality for designing the control algorithms using block schemes and providing them to the real-time server. The associated layer of real-time server is dedicated for processing provided schemes in order to use them by the top layer, which is the remote access layer.

The remote access layer is used to create a communication channel between the application and the real-time server.

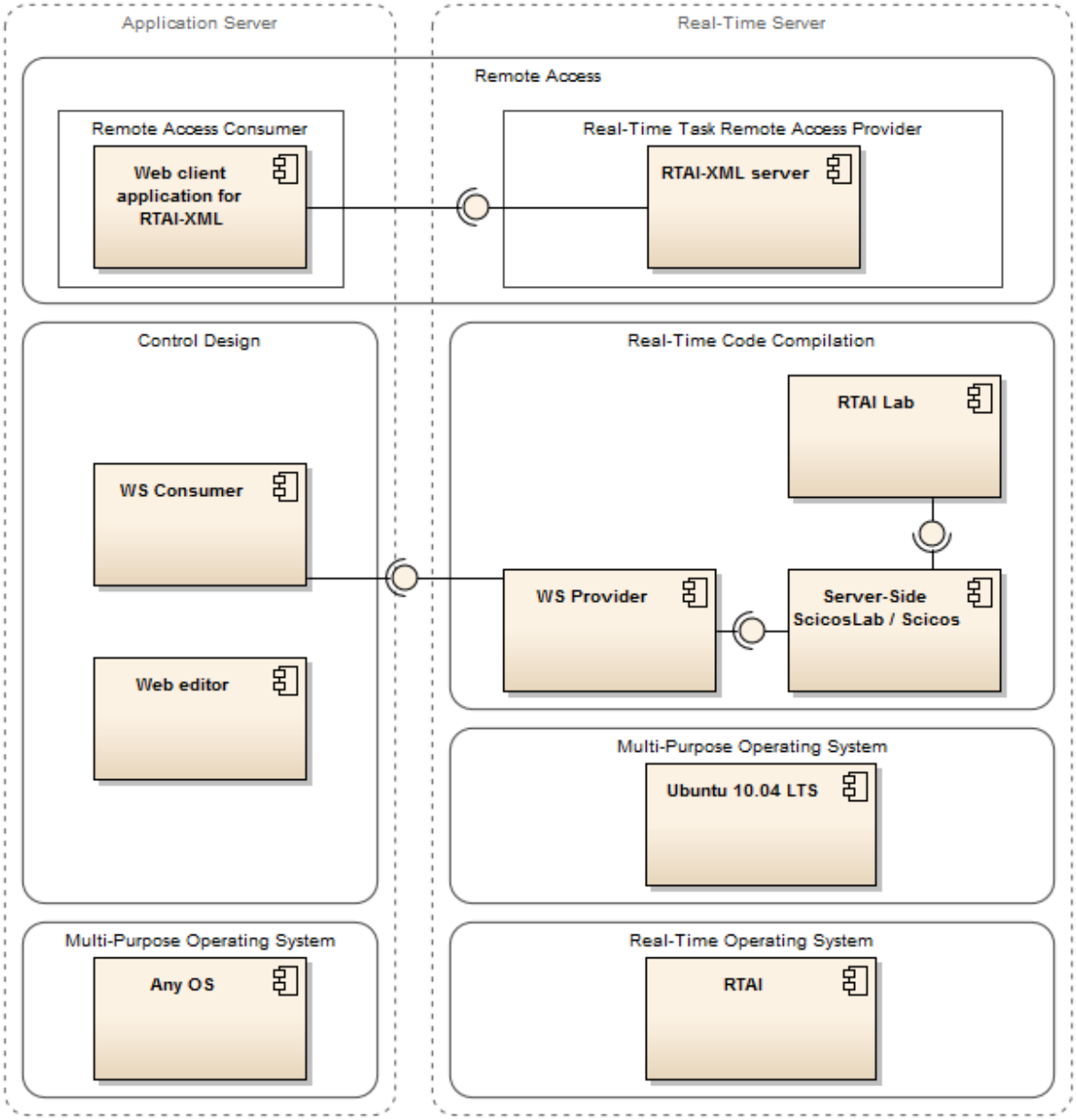

Fig. 2 Architecture diagram of the on-line laboratory environment

The application sever uses this channel to connect to the remote real-time task, execute it, interact with it and spectate the results.

IV. Real-Time Environment

Standard installation of Real-Time Suite contains a patch for Ubuntu's kernel that adds hard real-time capabilities to the system. The RT Suite also offers:

- Comedi drivers for real-time communication with hardware (e.g. data acquisition boards),
- ScicosLab and Scicos for design of block schemes (similar to Matlab and Simulink),
- RTAI-Lab and Comedi-lib toolboxes for ScicosLab that allow use of specialized real-time blocks in Scicos,
- RTAI-XML server that makes compiled real-time task available via Internet using XML-RPC (Extensible Markup Language – Remote Procedure Call) protocol.

We have developed a web-based client application that is able to communicate with such real-time tasks [10].

Despite the ability of RTAI-XML server to provide Internet access to real-time tasks, the only way to create custom tasks was to design block schemes in Scicos, compile them locally on the Real-Time server and set-up the RTAI-XML configuration to allow remote connection to the newly compiled real-time task.

It was necessary to extend the standard functionality of Real-Time server powered by Real-Time Suite to be able to compile block schemes remotely using RTAI-Lab toolbox for ScicosLab. We have developed a web service interface to communicate with client applications using SOAP (Simple Object Access Protocol).

The standard installation of Real-Time server had to be extended by Apache Web Server and PHP interpreter to be able to provide web services interface. Also, the MySQL database server had to be installed to provide storage for user and scheme settings and logs.

V. Web Services Provider

Our solution of the Real-Time server contains a component called WS Provider. This component is essentially a web application that communicates with clients using SOAP protocol. The WSDL (Web Services Description Language) file that accurately describes all services is also available. Thanks to the WSDL, the user can use a generic (general purpose) SOAP client, thus there is no need for difficult development of client applications. However, we recommend using the developed module that is dedicated to provide communication interface with WS Provider (see section VI).

The WS Provider component enables clients to submit an XML file containing the custom block scheme. Compatible XML files can be exported directly from desktop version of Scicos (using any operating system) or from provided web-based scheme editor module [7]. The submitted scheme is processed on the Real-Time server afterwards. Each scheme is checked for presence of potentially malicious blocks or code that could harm the server or hardware equipment. After a successful verification, the submitted scheme is inserted into the framework scheme. The selection of the desired framework scheme is done by the user before the custom scheme submission.

Framework schemes in our laboratory environment always contain all necessary blocks for generation of signals, measurement and hardware communication, but the core part of the scheme (primarily used for controller design) is blank. Thanks to this feature, users are relieved from designing the whole blocks schemes, thus they can focus to the main part of the block scheme (e.g. the controller). The example of such framework scheme can be seen in Fig. 3. Given scheme

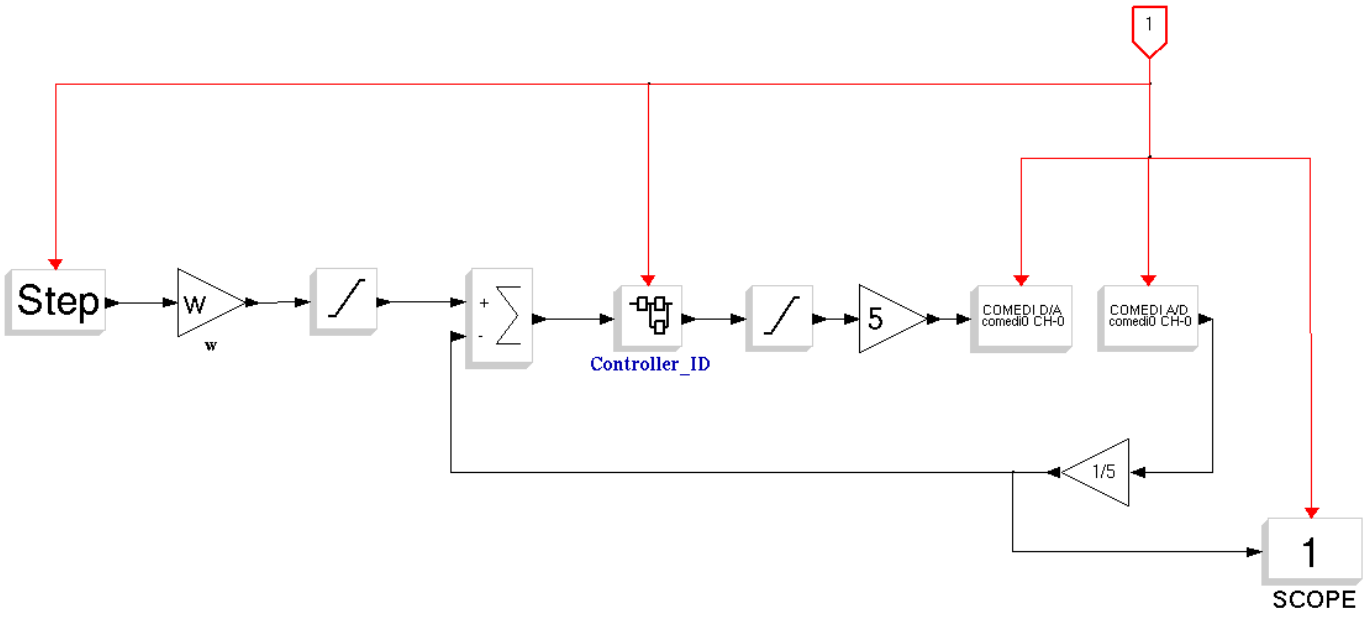

Fig. 3 Framework scheme for magnetic levitation control

contains blocks for generating desired value (*w*), closed loop for controller, blocks for limitation of maximum value of action signal to connected hardware, communication with hardware (*Comedi D/A* and *Comedi A/D* blocks), measurement (*scope*), clock signal (input block on the upper right-hand side) and empty superblock marked as *Controller_ID* that is prepared for insertion of custom controller scheme.

The resultant block scheme created by combining the framework scheme and the custom structure of the controller is saved into application's working directory. The WS Provider component executes ScicosLab using a special startup script afterwards. The script loads necessary macros that are required for proper functionality of Scicos, RTAI-Lib toolbox and non-interactive compilation of block scheme into executable binary file. The automated scheme compilation is discussed in [8]. We have configured the RTAI-XML server to allow remote access to the newest real-time task that was compiled using the described method.

Web services provided by WS Provider component are divided into two separate parts: standard user functions and administration. The standard functionality contains following functions:

- *frameWorkSchemeList()* – retuns a list of all framework schemes that are currently available within the WS Provider component on current Real-Time server;
- *frameworkSchemeDescription(schemeName)* – returns a formatted detailed description of the framework scheme whose name was provided in the argument (framework scheme descriptions are managed by administrators using web service described later in this section);
- *compile(frameworkSchemeName, customScheme)* – checks the custom scheme and identifies potentially malicious blocks or code, inserts custom scheme into the framework scheme, executes the scheme compilation and returns ScicosLab's console output;
- *compileFull(customScheme)* – checks the custom scheme and identifies potentially malicious blocks or code, executes the scheme compilation and returns ScicosLab's console output. This method is available only for experienced users who want to design the whole scheme, not only the reserved part (i.e. the controller);
- *compileTest(frameworkSchemeName,customScheme)* – performs security verification, combines the framework and the custom scheme and returns a status message without executing the compilation step. This function is suitable for developers while integrating the laboratory portal with WS Provider.

For security reasons, we have added a verification procedure and a logging functionality to each provided service. Each user has to be authenticated to be able to execute any of the provided web service successfully. User's credentials have to be provided using HTTP Basic authentication in each SOAP requests. Usage of SSL is strongly recommended regarding the fact that user name and password are sent in open form. Each operation is automatically logged into the database that stores user names, date and time of service call and operation details.

The user and the privilege settings as well as the framework scheme descriptions can be managed using administrative web service interface. The administration offers several functions:

- *aclGrant(user, object, operation)* – grants the *user* a privilege to perform *operation* on a selected *object*;
- *aclRevoke(user, object, operation)* – revokes a privilege to perform *operation* on selected *object* from specified *user*;
- *userInsert(user, password)* – inserts new *user* and his *password* into the database;
- *userUpdate(user, password)* – updates *user*'s *password*;
- *userActivate(user)* – activates *user*'s account (newly created users do not have to be manually activated – they are created as active by default);
- *userDeactivate(user)* – deactivates *user*'s account (users may also deactivate their accounts accidentally after several failed attempts to call any method using wrong user name and password combination);
- *frameworkSchemeDescriptionInsert(schemeName, description)* – inserts new version of *description* of the selected *schemeName*;
- *logRead(dateFrom, dateTo, type, user)* – returns application's logs filtered by specified arguments.

Every method mentioned above returns a status message about a success or a failure of the invoked operation. User authentication and logging is implemented in the same way as in the standard user web services.

For proper functionality of WS Provider component, it is necessary to use a server with real-time environment described in section IV. Minimal software requirements are: RTAI-patched Ubuntu Linux, RTAI-XML server and ScicosLab with RTAI-Lib toolbox and compilation macros that are provided together with the WS Provider. The web server and PHP interpreter must be configured to be able to work with HTTP basic authentication and SOAP.

VI. Web Services Consumer

As a counterpart to the real-time server's WS Provider, we have developed an associated part hosted on the application server – called WS Consumer. The WS Consumer component is used for communication of client applications with WS Provider. It can be easily integrated into existing on-line laboratory portals that are intended to be used for custom scheme processing and compilation on the Real-Time server. It may also serve as a sample for developers of custom client applications for the Real-Time server's web services.

It is to say that the WS Consumer component is not a separate web application as the WS Provider. However, it is a PHP class that is prepared in form of a PHP wrapper. Each function described in previous function is encapsulated inside PHP object methods. These methods are accessible via PHP calls in any web application powered by PHP that includes the WS Consumer class. In this way, the component can be easily integrated into any existing project. The only demand on web

server and PHP interpreter is to be able to use PHP's SOAP extension.

Thanks to the open modular architecture of the presented solution, the alternative application-server-side client may be used. In fact, the developers of third-party laboratory portal solutions can use their own custom SOAP clients for integration with WS Provider that was described in the previous chapter.

VII. Conclusion

The hard real-time control experiments were almost exclusively a domain of desktop systems. However, more real-time experiments appear in public on-line laboratories. Unfortunately, they lack certain level of interaction with user since the user can execute usually only a pre-defined experiment with ability to change only a limited number of properties. Thanks to the presented solution, we have managed to enhance possibilities of on-line laboratory system and we have demonstrated a new possible trend in evolution of PaaS-like on-line laboratory systems. Together with the rest of features of the developed on-line laboratory modules portal, this solution could be used as a very flexible support for on-line courses in the area of automation and control.

The described solution is already deployed in our department on the Humusoft CE152 magnetic levitation plant [6]. We are planning to continuously extend the scope of the on-line laboratory, provide more plants and possibly to make public access to the on-line laboratory platform to all interested users.

We are continuously observing a growing trend in introduction of cloud-based services that replace the previous "on premise" philosophy. The same trend can be also seen in deployment of on-line laboratories that complement the standard "hands-on" laboratory courses. In this paper, we have presented an example of merging the cloud computing PaaS model to an on-line remote laboratory solution that can be very beneficial in teaching of technical studies.

References

[1] R. Bucher and S. Balemi, "Scilab/Scicos and Linux RTAI – A Unified Approach," *IEEE Conference on Control Applications*, Toronto, Canada, 2005.

[2] H. M. El-Bakry and N. Mastorakis, "Performance Evaluation of XML Web Services for Real-Time Applications", *International Journal of Communications*, Issue 2, Vol. 3, 2009.

[3] A. Gambier, "Real-time Control Systems: A Tutorial," *5th Asian Control Conference*, Vol. 2, pp. 1024-1031, 2004.

[4] A. Guiggiani. (2011). RealTime Suite [Online]. Available: http://www.rtaixml.net/realtime-suite/

[5] M. Huba and M. Šimunek, "Modular Approach to Teaching PID Control," *IEEE Transactions on Industrial Electronics*, ISSN 0278-0046, Vol. 54, No. 6, pp. 3112-3120, 2007.

[6] Humusoft (1991-2014), CE 152 Magnetic Levitation Model website [Online]. Available: http://www.humusoft.cz/produkty/models/ce152/

[7] Z. Janík and K. Žáková, "Online Design of Matlab/Simulink Block Schemes," *International Journal of Emerging Technologies in Learning (iJET)*, ISSN 1863-0383, Vol. 6, Special Issue 1, pp. 11-13, 2011.

[8] Z. Janík and K. Žáková, "A Contribution to Real-Time Experiments in Remote Laboratories," *International Journal of Online Engineering (iJOE)*, ISSN 1861-2121, Vol. 9, Issue 1, pp. 7-11, 2013.

[9] Z. Janík and K. Žáková, "One Example of RTAI-Based Remote Experiment," *IN-TECH 2013: Proceedings of International Conference on Innovative Technologies*, Budapest, Hungary, ISBN 978-953-6326-88-4. – pp. 273-276, 2013.

[10] Z. Janík and K. Žáková, "Performance Tests of MyISAM and InnoDB Database Engines for Online-Based Real-Time Experiments," *Cybernetics & Informatics '14*, Oščadnica, Slovakia, February 5 – 8, 2014.

[11] M. Jansen, "Will Cloud Computing Change Standards in IT Service Management?" *International Journal of Computers and Communications*, Issue 1, Volume 6, pp. 9-16, 2012.

[12] L.-H. Kuo, J.-C. Yu, H.-H. Yang, W.-C. Hu and H.-J. Yang, "Creating a High-Scope On-Line Course for Cloud Computing," *International Journal of Education and Information Technologies*, Issue 4, Volume 7, pp. 146-159, 2013.

[13] Z. Magyar and K. Žáková, "SciLab Based Remote Control of Experiments," *9th IFAC Symposium on Advances in Control Education ACE'12*, Nizhny Novgorod, Russia, 2012.

[14] Z. Mahmood, "The Promise and Limitations of Service Oriented Architecture," *International Journal of Computers*, Issue 3, Vol. 1, 2007.

[15] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *National Institute of Standards and Technology Special Publication* 800-145, September 2011.

[16] S. Pastore, "Distributed computing platforms like clouds and web standards: what could be the solution in an open environment?" *Proceedings of the 10th WSEAS international conference on Applied computer and applied computational science*, Venice, pp. 195-200, 2011.

[17] M. T. Restivo, J. Mendes, A. M. Lopes, C. M. Silva and F. Chouzal, "A Remote Lab in Engineering Measurement," *IEEE Transactions on Industrial Electronics*, vol. 56, no.12, pp. 4436-4843, 2009.

[18] A. Seth, H. Agarwal and A. R. Singla, "Integrating SOA and Cloud Computing for SME Business Objective," *WSEAS Transactions on Computers*, Issue 3, Volume 11, March 2012.

[19] F. Schauer, M. Ožvoldová and F. Lustig, "Real Remote Physics Experiments across Internet – Inherent Part of Integrated E-Learning," *International Journal of Online Engineering (iJOE)*, 4, No 2, 2008.

[20] I. Zolotová, M. Bakoš and L. Landryová, "Possibilities of communication in information and control systems," *Annals of the University of Craiova, Series: Automation, Computers, Electronic and Mechatronic*, Vol.4(31), No.2, pp.163-168, ISSN 1841-062, 2007.

[21] J. G. Zubía and G. R. Alves, *Using Remote Labs in Education: Two Little Ducks in Remote Experimentation*, University of Deusto, Bilbao, ISBN: 978-84-9830-335-3, 2011.