Implementation of Reusable Solutions for Remote Laboratory Development

http://dx.doi.org/10.3991/ijoe.v12i07.5825

Anzhelika Parkhomenko, Olga Gladkova, Aleksandr Sokolyanskii,
Vladislav Shepelenko, Yaroslav Zalyubovskiy
Zaporizhzhya National Technical University, Zaporizhzhya, Ukraine

***Abstract*—Development of remote laboratory for embedded systems complex hardware/software design is an actual task, because the challenges existing in this area, require qualitatively new techniques, technologies and tools of design. By using the possibilities of remote labs and reusable hardware/software components, developer can more optimally organize the project and realize it in a shorter time. Development and usage of remote labs for designers can give new opportunities and ways for accumulation and study of existing design experience and ready solutions. On the other side, today, in the area of remote laboratories development are no common standards and approaches. Different developers offer different solutions for laboratory functionality, interfaces, a set of experiments, etc. At the same time, there are a number of ready-made solutions that can be used repeatedly for more optimal development and rapid integration with existing projects. Therefore, investigation and implementation of re-use methodology and its practical realization is an urgent task. The paper presents the structural components and API of remote laboratory RELDES, proposed for reuse in other projects for creation of mobile applications, new clients or services. Open questions of RESTful API documenting are also discussed.**

***Index Terms*—embedded systems design; remote laboratory; software/hardware components reuse; API; REST - request.**

I. INTRODUCTION

Nowadays reusable solutions are becoming more and more necessary in various fields. Different reasons of reuse are traced, but main goal - optimization. Reuse brings a lot of benefits, from which time and cost savings are the most important. A huge number of works demonstrate, that the reuse is demanded in the engineering. Reusable solutions are of interest to engineering community because they give possibility to replicate and to implement the separate parts of system without spending time for their development [1-5].

Embedded systems (ESs) design is one of the fields of reusable solutions creation and application. Requirements to the ESs functional complexity and design time are enhanced. Under these conditions, the development of systems based on ready solutions becomes more effective. Thus, the accumulation of technical solutions and their subsequent reuse are perspective directions of ESs design efficiency improving.

As known, the ESs includes the hardware and software components. Increasing reuse opportunities is a well-known task for software as well as for hardware designers. But current software and hardware engineering practices have embraced different approaches to this problem [1].

The actual question is: at what stage of the life cycle of the hardware/software systems should decide that certain components (or modules entirely) worthy of reuse? Normally this decision makes on the fly somewhere between analysis and design, sometimes it is taken at the design stage, sometimes - during realization or generalization.

For a successful selection of software components for reuse are important the following aspects [6]:

- Understand the architecture of the original code.
- Define potential reuse.
- Evaluation of expenses of time for reuse as compared to reworking of components.
- Decide for each component: what and how to reuse.

Revolution of platform-based approach in the design was the beginning of the new concepts of ESs quick development and prototyping. Ready hardware/software platforms give possibility of systems components reuse for the design process efficiency improvement. The *Basic* Concept for Remote-based Embedded Systems Design was proposed in the authors' previous works [7]. Development and application of remote laboratories for ESs rapid prototyping based on reusable hardware and software components is an actual task today for professionals [8-10].

There are many successful projects, moving faster by reusing existing software (WebLab-Deusto, iLab, Sahara, etc.) and hardware (VISIR, etc.) solutions. [11-13]. Knowledge about reuse can be useful for developers of such remote laboratories (RLs) for ESs design.

That is why the aim of this work is investigation of reuse experience in the field of remote lab development and identification of remote labs reusable hardware/software components.

II. INVESTIGATION OF RL INFRASTRUCTURE AS A SERVICE

As shown last decade the number of remote laboratories increased. Each developer finds his own way to develop remote laboratory from scratch. This way is always difficult and includes the different kind of problems during the development and testing RL.

Hence, in the recent years, the question about the new approach to the development of the remote laboratory, which consists of independent modules, becomes relevant. The development of such an approach will contribute the lab's individual components reuse and simplify the creation of RLs.

One such approach is Laboratory as a Service (LaaS) paradigm which is described in [14]. The main idea of this

paradigm is developing modular remote laboratory. All functionalities of which are implemented using REST web service (but not limited) and will be delivery in the "service description file" as a set of abstraction service. As authors underline the first interpretation of this approach is called Anything as a Service (XaaS), but it was no clear and had some undecided question. Moreover in the [15] was said that LaaS is a not new idea and came from Software to Service (SaaS) paradigm. Also in this work authors proposed their own approach to delivering laboratory server infrastructure as a service (LIaaS). In contrast to the LaaS paradigm in the presented approach user don't have to deployed and implemented loosely coupled services themselves. LIaaS have two way how to access to the experiment will be delivered (at the lab owner's discretion): using a Remote Laboratory Management System (RLMS), or without RLMS directly with laboratory using Experiment Dispatcher.

On a practice we implement of an *experiment engine (EE)* using the Experiment Dispatcher for ELVIS lab which was developed on iLab Sharing Architecture. The Experiment Dispatcher is a software framework that allows for the centralization of some typical functionality of online laboratory servers. It shifts the core features of an Online laboratory server to a central location and allows for a seamless reuse of the lab server infrastructure by heterogeneous online laboratories. In other words, it abstracts the development of the software necessary to deliver remote experimentation.

PHP language was choosing like the programing language for developing *EE*. For communication *experiment engine* with server the Dispatcher has REST Web Services API. API includes several HTTP methods such as PUT/POST, GET and other. All requests are presented in the JSON format. Requests should contain the Authorization HTTP header (username and password) and the HTTP header "X-apikey". The value is a key that will identify this particular experiment engine with a lab server. The values of both username and password (for the Authorization header) and X-apikey were received after registration on iLab Service Broker.

Example of the HTTP request *//get experiment status*

```
$response = \Httpful\Request::get($url."status")
                    -
>authenticateWith($user,$password)
                    ->addHeader('X-
apikey',$apiKey)
                    ->expectsJson()
                    ->send();
$result = json_decode($response);
echo "Experiment status:";
if($result->success) {echo "true";}
else echo "false";
```

The result generated by the batched lab is a data in JSON format, so we should decode it.

The one examples of the use of laboratory's infrastructure and functionalities has been shown above. This approach allows the reuse remote lab infrastructure to crate your own experiment, without spending the time to develop whole RL.

Research has shown that in general exist several remote laboratories which promote the reuse paradigm in the field of remote labs development: iLab Shared Architecture, WebLab-Deusto, Labshare Sahara. These laboratories have been investigated by various authors in the works [16, 15, 17]. All of these remote laboratories have their own RLMS that help to organize a centralized management of the laboratory's functionalities (registration, authorization, queue, management of experiment and other). The functionality of which is available through the Web-service API.

Based on this experience of using a Web-service API for creation additional client for remote lab, it was decided to merge all the functionality of the remote lab RELDES into the RELDES management system and to create API which was used REST technology for connection client with service. This will allow easier to create a new server's client.

III. IMPLEMENTATION OF REMOTE LABORATORY FOR DESIGN OF EMBEDDED SYSTEMS

Remote Laboratory for Design of Embedded Systems (RELDES) is user friendly application for testing and rapid prototyping of the ESs based on reusable hardware and software components [18]. Today 4 experiments based on Arduino boards and control equipment - actuators, ultrasonic distance sensor, traffic light and LCD display are implemented [19].

All specifications, schemes of experiments and program code (templates for the experiments usage) are available at the RL web-page. Like other developers of Relle lab [20], we also have chosen Arduino platform, as one of the most popular and simple reusable solution. Nevertheless, the proposed set of experiments can be expanded for other hardware/software platforms, and tasks connected with realization of various classes of ESs.

RELDES Management system includes next modules: Registration; Administration; Execution of the experiment; API (Fig. 1).

For Registration in the system a common procedure of filling the standard fields is implemented: Username, Password, Email, Name. If a login is successful, the user receives an email with link to complete the registration process. After confirmation, the user will be able to use API RELDES with his individual API-KEY.

Administration of the system includes monitoring of experiments current state, statistics of experiments execution, as well as feedback to users. Admin module is implemented as panel which includes area for information visualization and 3 points of menu: Notification of users; Keeping statistics; Management of experiments (Fig. 2).

On the *Notification of users* is a list of all users registered in the system with the ability of mail to individual user or to a group of users using simple check box.

On the *Keeping statistics* the system displays a table with all successfully compiled source code. The chart of each experiment activity is also presented.

On the *Management of experiments* the system displays a table with the current state of each experiment and if experiment is occupied - it shows the login of the user who is working on the experiment now. Also for each of experiments will be shown a Queue table and users will receive access to experiments in accordance with this queue.

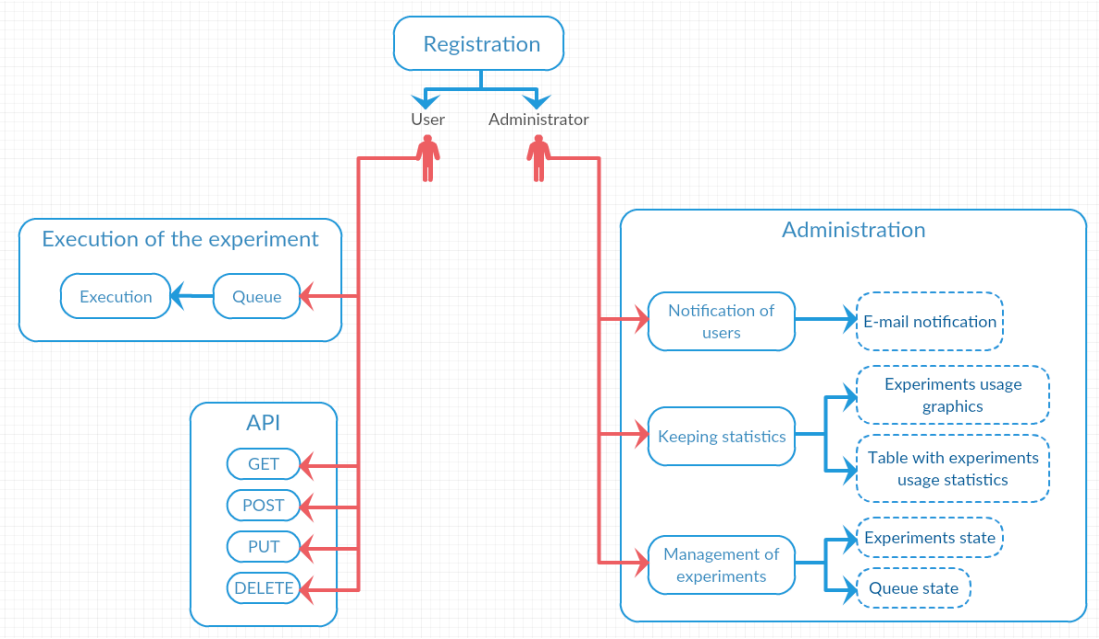

Figure 1. RELDES Management system scheme

Queue panel (Fig. 3) demonstrates the current state of each experiment (free/busy), allows the view of all users who are in the queue list. Start and finish time for each user and experiment is also displayed as the basic information.

When the experiment successfully occupied by the user, he has the next opportunities for *Execution of the experiment:*

- Create program code from scratch in the RL special window with syntax highlighting.
- Upload own file with program code.
- Use ready software solutions (several scenarios are proposed for each experiment).

After that, the source code can be sent to the server for compilation and loading to Arduino board of the current experiment.

RELDES API provides access to the lab server infrastructure and allows a seamless reuse of the lab functionality.

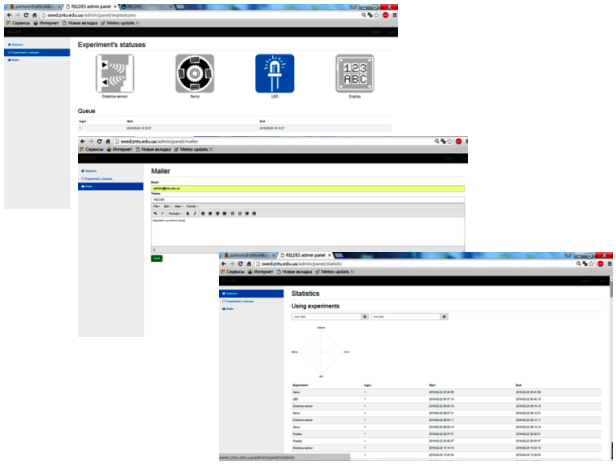

Figure 2. RELDES Admin panels

The API includes functions for the next HTTP requests (*id* - experiments number from 1 to 4 and *username* – user login):

- GET/…/expStatus/id - checks on the status of experiment and returns result: busy(1) or free (0).
- POST/…/queueUp/id/username – checks on the status of experiment and puts the user with the given userName in the experiment queue. Returns the result about adding to the queue.
- GET/…/getQueue/ id/username - gets the waiting time specified user #UserName for a given experiment #id
- GET/…/getExpQueue/id - gets a list of all queues on the experiment #id
- GET/…/getAllQueue - gets a list of all queues on experiments
- POST/…/compile/id/ - checks the validity of the program code for Arduino board.
- POST/.../upload/id/username/ - uploads the hex file after compiling if current user has taken the experiment.

For example, function *compile* receives and handles REST- requests [21-22] that can be sent by the users. It does the following: sends the code from request to controller; compiles it on the server using Ino; generates result of compiling as response.

Any registered developers can use the API for their own purposes. It may be use for creation of mobile applications, new clients or services.

IV. IDENTIFICATION OF REUSE POSSIBILITIES OF RELDES COMPONENTS

A priori, we proceeded from the fact that the possibilities of reuse should be laid from the beginning of development and that is why we started with a clear and simple architecture, which is well solves the task (Fig. 4).

As a template for application separation to the levels, we have chosen a MVC (Model-View-Controller) pattern. The MVC pattern describes a simple method for constructing the

structure of an application, the aim of which is to separate the business logic from the user interface [23-24].

As a result, the application is easier developed, scaled, tested and accompanied. The correct separation of applications helps to maintain a strict separation of functionality that provides the flexibility and convenience and ease of maintenance. This makes the different functionality of the application reusable.

MVC pattern helped functionally divided our system to further the reusable components creation. *Model, View* and *Controller* interfaces have been developed.

Interface *Controller* demonstrates the implementation of system interaction with user, so all users main methods are located in classes that inherited from this interface. We have created three classes and inherited from *Controller* (*front, experiment* and *admin*) to divide the system to structural components (Fig.6). All functions that connected with all methods from main system page are located in *front* class. *Experiment* class includes all methods that provide performance of the experiment and experiment page. *Admin* class contains all administrative functions for keeping statistics, email notifications etc.

Model is developed to store data that retrieved to the *Controller* and displayed in the *View*. So three models for controllers: *front_model, experiment_model, admin_model* have been developed. Objects of *front_model, experiment_model* and *admin_model* respectively for access to their functions in controllers have been created. In particular, the *Model* deals with the registration and authorization of users (for example, functions: addUser, getUser, getUserID). Also, the *Model* provides information about the experiments: occupied or free experiment (isBusy); to occupy experiment (occupy); to free experiment (free).

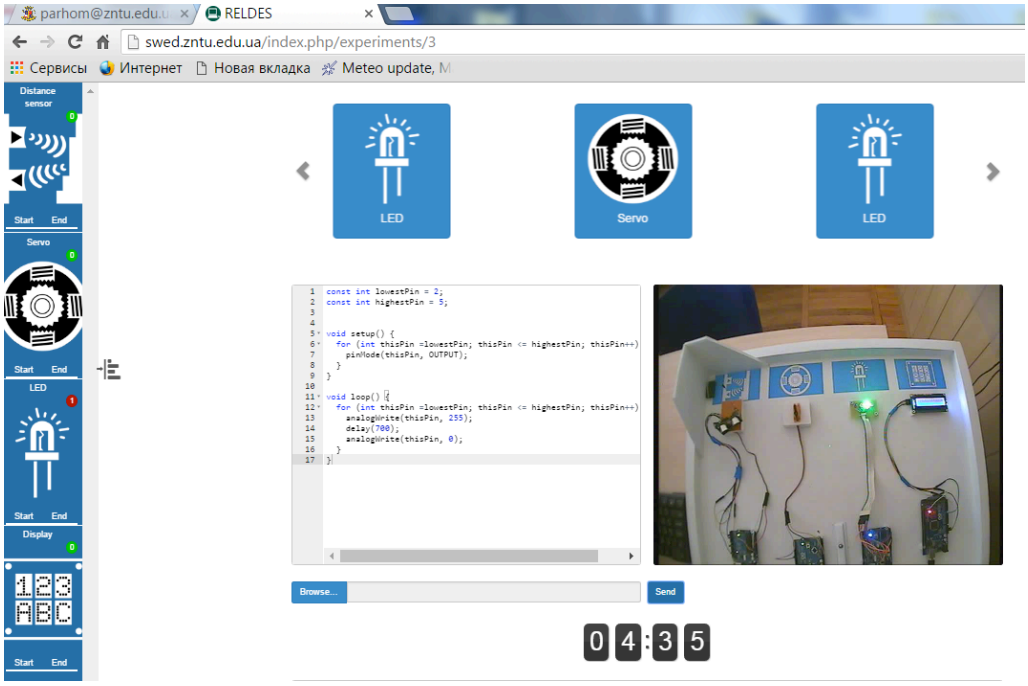

Figure 3. RELDES interface with Queue panel

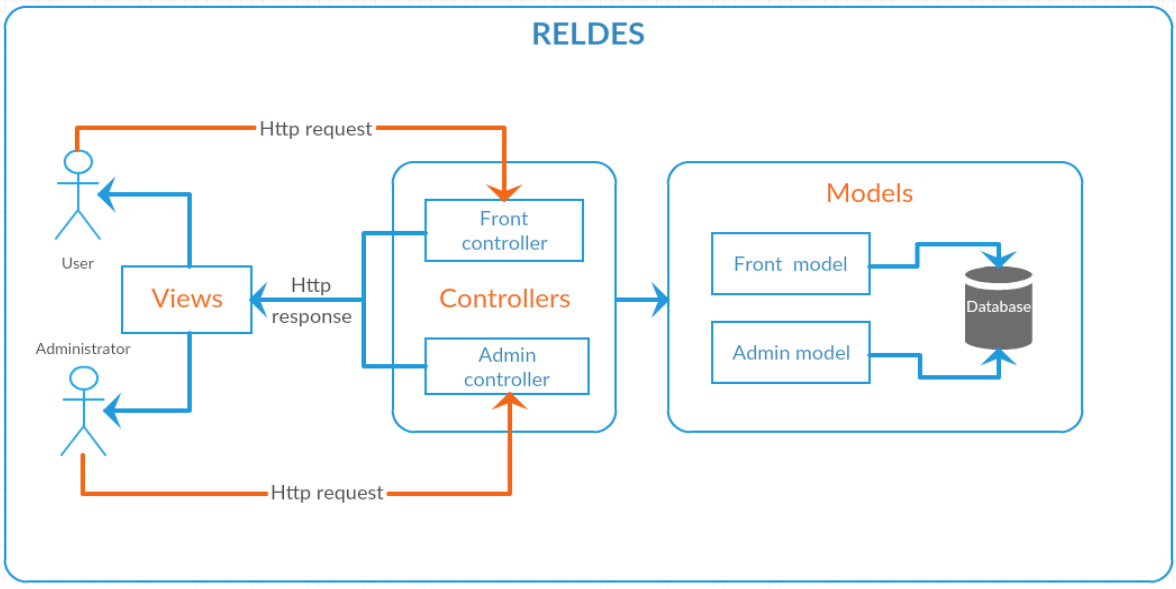

Figure 4. Overview of the RELDES architecture

View is responsible for displaying information (visualization). Often it serves as a representation of the form (window) with graphic elements. Consider the purpose of each View: index – main page; header – header of main page; footer – footer of main page; Arduino – review of Arduino board; IDE – review of Arduino IDE; experiments – shared part of all experiment; LED – specific part for experiment with LED; servo – specific part for experiment with servo; distance – specific part for experiment with distance sensor; display – specific part for experiment with display.

The *Controller* provides the connection between the user and the system: It controls users input and uses *Models* and *Views* for necessary response implementation. Also the *Controller* implements site navigation (for example, function: index, Arduino, IDE, experiments, etc.).

Broadcast video is implemented like individual module. Specifically it is implemented as MPEG-broadcast through a web sockets, and drawing in the tag <canvas>. Video from the camera is encoded by ffmpeg. Launched on nodejs JavaScript code sends images via a web sockets. Upon receipt of the user frames are decoded in the client using jsmpeg (MPEG1 Video decoder). After decoding the frame is drawn in the tag <canvas> via WebGL.

So if our system will be expanded and requires more components we can easy create more classes and inhere from our interfaces.

We assume that our RELDES project contains a few modules for reuse: Individual users' module queuing to experiments; Individual module implemented broadcast video; Admin panel; API. Queue module has been already successfully used and checked in another our project.

V. FEATURES AND APPROACHES TO API DOCUMENTATION DEVELOPMENT

After the development of API, it is very important to make a detailed and well-structured documentation of it, because exactly this step will determinate how effectively this API will be used. Today API RELDES documentation is presented in plain text descriptions on the labs webpage and consists of General description, API routes and methods, Input parameters, Output data, possible errors (Fig. 5).

The issue of documenting is not so simple, as it looks. Most companies, even large ones, still prefer to use a simple text description. But the disadvantages of this approach lies in the fact that such documentation is hard to maintain up to date and its functionality is quite limited.

To simplify the process of API documenting exist different specialized tools and services (API Blueprint, Swagger, RESTUnited, MireDot, Kittn API, apiDOC, apiGen and others). As a rule, they generate documents based on the description in a standardized formats (JSON, Markdown, YAML, RAML). For example, Swagger supported YAML format. YAML is much more convenient than the JSON, but it does not allow easy describe the repeating elements, which are often present in the API descriptions. API BluePrint use format Markdawn wich was intended primarily for text formatting and not for use as a basis to generate. So, it is very difficult to adjust it under API documenting. API Designer – is interactive editor, based on RAML format for writing documents online, and a platform for testing. Its undoubted advantages are simplicity and consistency, but the problem

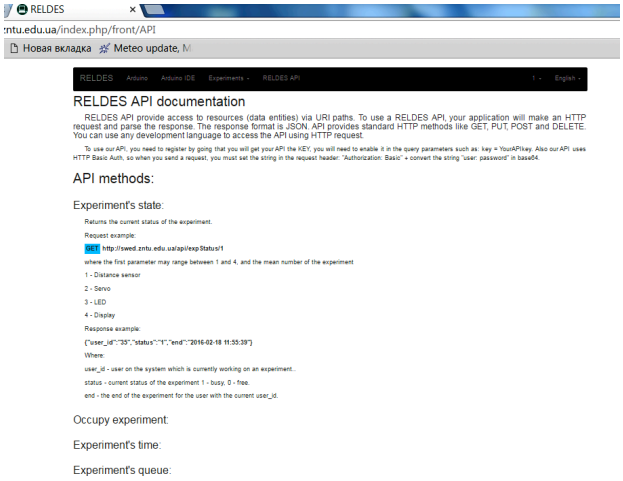

Figure 5. Web-page with API RELDES description

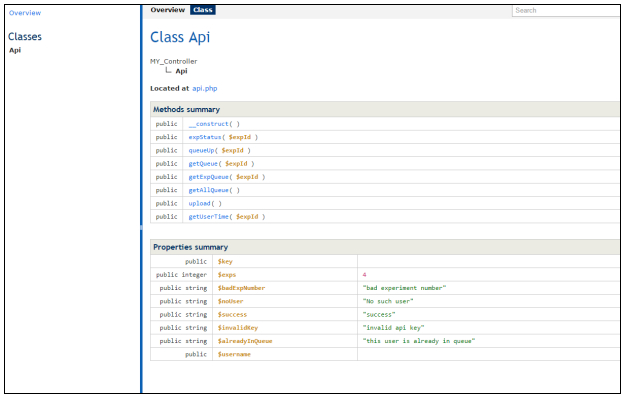

Figure 6. Web-page with RELDES API description generated by apiGen.

is the lack of tools developed by the community. Some generators support limited number of programming languages, some are not for free and all have their own advantages and disadvantages.

So, after the all possibilities were considered, one of the most suitable was found apiGen. ApiGen is the tool for creating professional API documentation from PHP source code. ApiGen has support for PHP 5.3 namespaces, packages, linking between documentation, cross referencing to PHP standard classes and general documentation, creation of highlighted source code and experimental support for PHP 5.4 traits [25]. Use of this tool speeds up the project due to the fact that just in the process of developing you write the necessary comments for the code and then simply pass this file to apiGen, and it generates a ready-made page with search, links, etc., i.e. there is no need for the additional development of the page with description of API documentation. We think this tool really allow us to create smart and readable documentation for our PHP project (Fig. 6).

However, in the future, when the RELDES API will expand, its functionality will be appropriate to use another standards and generators for developing and documenting.

VI. CONCLUSIONS

As in other high-technology fields, in remote lab development there is great benefit in the reuse of different intellectual property. Creation of reusable lab components is a step towards the reduction of labor costs in the remote lab

development. However, the problem is that the costs of design, implementation and maintenance of publicly available code are much higher than the cost of custom simple solutions. It begins with the fact that reusable code requires more completed and detailed documentation, better quality and test coverage, as well as examples of use and extra time for all these tasks.

Creation of reusable components isn't the key to improving "reuse" code. The existence of high-quality code available for reuse does not guarantee its use. Because, developers should have reuse culture for normal reuse process.

Standardization and unification of API descriptions, as well as API documentation issues are important for both - developers and users. In fact, no special solutions are unlikely to be effective in addressing these problems in the future if the API will be extended or modified.

REFERENCES

[1] D. Bornstein, "Model Reuse through Hardware Design Patterns Recycling Unused Medicines to Save Money and Lives", March, 2015. http://opinionator.blogs.nytimes.com/2015/03/20/recycling-unused-medicines-to-save-money-and-lives/?_r=0

[2] F. Xiang, G. Ling, and Y. Jin, "User-driven GIS software reuse solution based on SOA and Web2.0 concept", IEEE Computer Science and Information Technology, 8-11 August 2009, pp. 5-9 (ICCSIT 2009. 2nd IEEE International Conference on)

[3] T. Barbour, "Developing Reusable Components", Proceedings Embedded Systems Conference, San Francisco, CA, 2001.

[4] J. Fredriksson, and L. Rikard, "Reusable Component Analysis for Component-Based Embedded Real-Time Systems", Proceedings of the ITI 2007 29th Int. Conf on Information Technology Interfaces, Cavtat, Croatia, pp. 615-620, June 25-28, 2007. http://dx.doi.org/10.1109/iti.2007.4283842

[5] L. Baum, and M. Becker, "Generic Components to Foster Reuse, Proceedings. 37th International Conference Technology of Object-Oriented Languages and Systems, IEEE, Sydney, NSW, pp.266-277, November 20-23, 2000.

[6] J. L. Scouler, and M.R. Bakal, "Successful code reuse with code-centric development and modeling", January 24, 2012. http://www.ibm.com/developerworks/rational/library/reuse-code-centric-development-modeling/index.html

[7] A. Parkhomenko, O. Gladkova, S. Kurson, A. Sokolyanskii, and E. Ivanov, "Internet-Based Technologies for Design of Embedded Systems", Journal of Control Science and Engineering (Serial Number 5), vol.3, no.2, pp.55-63, Mar.-Apr.2015

[8] A.Parkhomenko, A. Sokolyanskii,V. Shepelenko, Y. Zalyubovskiy, and O. Gladkova "Reusable Solutions for Embedded Systems Design", International Conference on RemoteEngineering and Virtual Instrumentation, REV2016, Madrid, Spain, February 25-27, 2016, pp. 313-317

[9] K. Henke, G. Tabunshchyk, H.-D. Wuttke, T. Vietzke, and St. Ostendorff "Using Interactive Hybrid Online Labs for Rapid Prototyping of Digital Systems", International Journal of Online Engineering (iJOE), Vol 10 (2014), pp.57-62. http://dx.doi.org/10.3991/ijoe.v10i5.3994

[10] G. Tabunshchyk, D. Van Merode, P. Arras, and K. Henke "Remote Experiments For Reliability Studies Of Embedded Systems", International Conference on RemoteEngineering and Virtual Instrumentation, REV2016, Madrid, Spain, February 25-27, 2016, pp.68-71

[11] WebLab Deusto – Scalable, web-based and experiment-agnostic remote laboratory management system. https://github.com/weblabdeusto/weblabdeusto

[12] The gateway4labs project/Go-Lab Smart Gateway. https://github.com/gateway4labs/

[13] SAHARA Labs. http://sourceforge.net/projects/labshare-sahara/

[14] M. Tawfik, C. Salzmann, D. Gillet, D. Lowe, H. Saliah-Hassane, E. Sancristobal, and M. Castro, "Laboratory as a Service (LaaS): a Novel Paradigm for Developing and Implementing Modular Remote Laboratories," International Journal of Online Engineering (iJOE), vol. 10, no. 4, pp. 13-21, Jun. 2014. http://dx.doi.org/10.3991/ijoe.v10i4.3654

[15] D. G. Zutin, M. Auer, P. Ordu˜na, and Ch. Kreiter "Online Lab Infrastructure as a Service: A new Paradigm to Simplify the Development and Deployment of Online Labs" International Conference on RemoteEngineering and Virtual Instrumentation, REV2016, Madrid, Spain, February 25-27, 2016, pp.202-208

[16] P. Ordu˜na, L. Rodriguez-Gil, I. Angulo, and O. Dziabenko "Towards a microRLMS approach for shared development of remote laboratories" International Conference on RemoteEngineering and Virtual Instrumentation, REV14, Porto, Portugal, Feb. 26-28, 2014, pp.375-381

[17] M. Niederstaetter, Th. Klinger, and D.G. Zutin "An Image Processing Online Laboratory within the iLab Shared Architecture" International Journal of Online Engineering (iJOE), vol. 6, no. 2, pp. 37-40, May. 2010.

[18] RELDES. http://swed.zntu.edu.ua, http://youtu.be/u2anq--UYFg

[19] A. Parkhomenko, O. Gladkova, E. Ivanov, A. Sokolyanskii, and S. Kurson, "Development and Application of Remote Laboratory for Embedded Systems Design", International Journal of Online Engineering (iJOE), vol.11, no. 3, pp.27-31, 2015. http://dx.doi.org/10.3991/ijoe.v11i3.4519

[20] Remote Labs Learning Environment. Deployment Environment for Arduino. http://relle.ufsc.br/

[21] REST-approach to building web applications client-server architecture. Mind Team, January 2015. http://mindteam.com.ua/ua/feed/rest-approach

[22] A. Naiden, Types of HTTP requests and REST philosophy. Habrahabr, January 24, 2009. http://habrahabr.ru/post/50147/

[23] Model-View-Controller. https://msdn.microsoft.com/en-us/library/ff649643.aspx

[24] K. Waterson, "Model-View-Controller MVC". http://www.phpro.org/tutorials/Model-View-Controller-MVC.html

[25] Smart and Readable Documentation for your PHP project http://www.apigen.org/

AUTHORS

Anzhelika Parkhomenko, Olga Gladkova, Aleksandr Sokolyanskii, Vladislav Shepelenko, and **Yaroslav Zalyubovskiy** are with the Software Tools Department, Zaporizhzhya National Technical University, 64, Zhukovskogo str., Zaporizhzhya, Ukraine (E-mail: parhom@zntu.edu.ua).

This work is supported by the TEMPUS project DESIRE "Development of Embedded System Courses with implementation of innovative Virtual approaches for integration of Research, Education and Production in UA, GE, AM" (544091- TEMPUS-1-2013-1-BE-TEMPUS-JPCR). Submitted 06 May 2016. Published as resubmitted by the authors 15 June 2016.