A CONCEPT OF BALANCED-CLIENT FOR RICH INTERNET APPLICATIONS

N. R. Dissanayake[1], G. U. Liyanage[2] and G. K. A. Dias[3]

[1] University of Colombo School of Computing, 35, Reid Avenue, Colombo 7, Sri Lanka.
Email: nalakadmnr@gmail.com
[2] Informatics Institute of Technology, 57, Ramakrishna Road, Colombo 6, Sri Lanka.
Email: Upeksha1996@aol.com
[3] University of Colombo School of Computing, 35, Reid Avenue, Colombo 7, Sri Lanka.
Email: gkad@ucsc.cmb.ac.lk

ABSTRACT

The Rich Internet Applications engineering is marked as complex and difficult, since they lack of architectural formalism. If we can identify the separation of the modules clearly and generalize the architectural structure in Rich Internet Applications, it might increase the realization of the system, hence reduce the complexities and speed up the development too. We propose an architectural structure based on Model-View-Controller pattern, which describes the separation of the logic; and the distribution of the modules and related components; between the client and the server, which we expect to specify as a standard and balanced layout, we name it the "Balanced-client", referred to the concepts of Thin-client and Thick-client. We expect to incorporate this concept to design a general architectural model for Rich Internet Applications, to provide a good realization, which can addresses the complexities in Rich Internet Applications engineering.

***Key words*:** Rich Internet Applications, Thin-client, Thick-client, Model-View-Controller

1. INTRODUCTION

With the increased popularity of the script-based approach – against the plugin-based approach – and the asynchronous communication techniques like Asynchronous Javascript And Xml (AJAX) [1] [2], the use of JavaScript (JS) and related technologies have been increased [3] in Rich Internet Applications (RIAs) development.

Traditionally in classical web applications, the business logic and other algorithms were developed using server-side languages and the executions of these components or modules were done in the server-side, where only the results were sent to the client. This architectural concept is called the thin-client, since the client-side development was limited [4] [5].

In some web applications development, the business logic along with other algorithms are taken more or completely to the client-side and developed with JS-based or plugin-based technologies like Adobe Flash/MS Silverlight [6]. This approach is called thick-client, where most or all the components are loaded to the client-side and the execution is done in client's browser [5].

With the asynchronous communication mode in RIAs, the concepts of thin-client and thick-client lose their boundaries, as in the same app both client and server side development contain high amount of algorithms and processing. The RIAs have a scattered logic [7] and lack of standards; and the complexity of RIAs is high where the design and the development is difficult [8].

2. METHODOLOGY

We have conducted a literature survey to gain the domain knowledge in the fields of RIAs, Web Architectures and Architectural patterns. We identified the relationships of these fields within the domain and noted the difficulty factors and complexities engaged.

A cross-sectional survey was conducted to identify the correlation between the difficulty level and the number of rich features per page. Targeted population was the individuals engaged in RIA engineering. For data gathering a structured questionnaire with closed end questions was used. Gathered data were analyzed and the results were derived using statistical methods.

Parallel to the surveys, we conducted a series of experiments to gather the empirical evidence in the domain. These series of experiments used a prototype based incremental development, where

the knowledge of the early iterations were applied and tested in the later iterations. For the rich features development AJAX is used; for the client-side development JS and jQuery was used; and for the server-side development PHP was used. Database development was done using MySQL server, and for hosting the RIAs Apache web server was used. All the development and testing were done in MS Windows platform.

3. DISCUSSION

In the literature survey we noted [9] that the complexities and difficulties in RIA engineering are caused by the lack of architectural formalism in RIAs, which reduces the realization [10].

When analyzing the data, gathered in cross-sectional survey, we noted that the difficulty level of implementing rich features increases with the number of rich features per page [11].

As we continue with our experiments, we noted that the complex nature of the RIAs – caused by the lack of architectural formalism – is mainly due to its scattered logic. And we realized that the identification of architectural properties and clear boundaries between the modules is essential in RIA engineering [12].

Since the Model-View-Controller (MVC) pattern provides a good architectural structure for web applications [13], we decided to use the MVC pattern as the basic structure of defining a new layout for the separation of logic in RIAs.

By analyzing the available MVC based libraries/frameworks such as Zend [14], CodeIgniter [15], AngulaJS [16], etc… we noted that the MVC modularization is limited to either server-side (Zend, CodeIgniter) or client-side (AngulaJS). The client-server nature of the web application has restricted the implementation of MVC pattern, in the same way it's being used in desktop applications. We identified this as the reason, which makes the web applications to have a scattered logic and lack of separation of MVC modules [7].

Our intention was to define a standard layout for separating the modules between client and server based on the MVC pattern, and we name the approach as "Balanced-client". Based on this Balanced-client layout, the engineers can have a better realization of the application structure and make decisions on moving the logic more to the client-side and make the system a thick-client or move logic more to the server-side and make the system a thin-client as needed, according to the requirements.

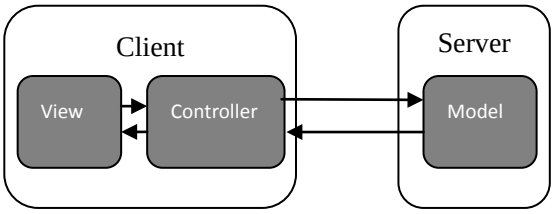

Figure 1: Use of MVC in Balanced-client

As illustrated in figure 1, we arrange the MVC modules in a way, the Model is executed in server-side where the Views and the Controllers are loaded to the client-side and executed in the browser.

We decided to take the Controller completely to the client-side, based on following criteria. The Views are always loaded to the client-side and the user interacts with the View(s), hence the user events are triggered in the client-side. Our aim was to implement all the event handlers in the Controller and place it in the client-side. Since the execution of the Controller is done in the client's browser, the Controller should be developed using JS or/and JS based libraries. When the event handler(s) need(s) to communicate with the server, it might be done via the AJAX engine, as asynchronous communication; unless it is a redirection.

Other than the event handling, components for some basic form validation algorithms also can be implemented in the client-side. We consider such scenarios as implementing the client-side components of the Model – as shown in figure 2. These client-side Model components may perform the mandatory/empty field validations and other non-sensitive validations. Additionally, these client-side Models may contain algorithms for simple data processing, such as calculating the age using the birthday provided, etc.... For our proposed Balanced-client concept, we recommend to perform all the non-sensitive validations using client-side Model components. The inputs can be validated further in the server-side also, as needed by the security constraints.

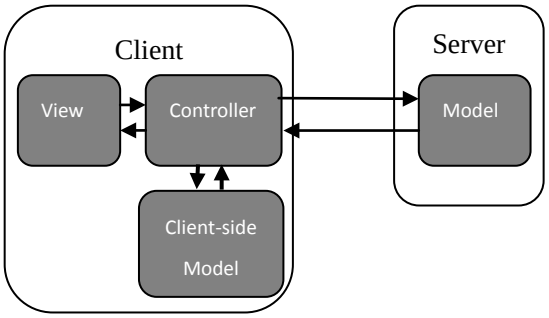

Figure 2: With client-side model

When some heavy DOM manipulations are needed as responds to the client events, algorithms for these DOM manipulations can be implemented as client-side Models, using JS/JS-based libraries. Such implementation might make the application a thick-client, with considerable amount of algorithms in client-side Models.

The Views contain just the presentation and can be developed using HTML, CSS and related web User Interface (UI) technologies. To have a good separation and lower coupling between the Views and the Controllers, we suggest that the specifications of the event handlers should be done in the Controller – as shown in figure 3 – using JS, instead using event handling attributes in HTML elements.

Using HTML attributes

```
<button onclick="processData();">
```

Using JavaScript

```
document.getElementById("btnProcess"
).onclick = function() { ... }
```

Figure 3: Events specification

The Model contains all the business logic and can be developed using a server-side language like PHP, complying with Object Oriented Programming (OOP) concepts. It should contain a component to handle asynchronous requests, sent by the controller. To minimize the complexities in asynchronous request handling, we have proposed a mechanism, we call it the RIA-Bus [17]. Model may contain addition components to utilize external layers such as databases, web services, etc…, which enables the expansion of simple client-server model up to n-tier architecture.

4. CONCLUSION AND FUTURE WORK

We propose a concept named “Balanced-client”, which provides an MVC pattern based layout, to be used as a standard model for RIA engineering. It describes the separation of the modules by the means of MVC pattern, based on the simple client-server model. With the realization provided by the Balanced-client model, we expect to have a better support in decision making of – how the logic should be arranged in the terms of thin-client and thick-client, without letting the logic to be scattered in a lack of manageable manner.

In our ongoing research, we plan to design and propose a general architectural model, which contains this Balanced-client concept. With this proposed architectural model, we expect to provide sufficient realization for RIA engineering, which can minimize the complexities and difficulties, hence speed up the development.

5. REFERENCES

[1] J. Farrell and G. S. Nezlek, *"Rich Internet Applications The Next Stage of Application Development,"* in Proceedings of the ITI 2007 29th Int. Conf. on Information Technology Interfaces, Cavtat, Croatia, 2007.

[2] L. D. Paulson, *"Building rich web applications with Ajax,"* Computer, vol. 38, no. 10, pp. 14 - 17, Oct 2005.

[3] E. J. Correia, *"What's Next for HTML5?,"* Intel Software Adrenaline, 2013.

[4] J. Nieh, S. J. Yang and N. Novik, *"A Comparison of Thin-Client Computing Architectures,"* Network Computing Laboratory, Columbia University, 2000.

[5] Margevicius, *"How to choose between thick- and thin-client architectures,"* Gartner, 2004.

[6] H. Dworak, *"A Concept of a web application blending thin and fat client architectures,"* in Fourth International conference on dependability of computer systems, warsaw, 2009.

[7] D. W. Cheung, T. Y. Lee and P. K. Yee, *"Webformer A Rapid Application Development Toolkit for Writing Ajax Web Form Applications,"* in Distributed Computing and Internet Technology. 4th

International Conference., Bangalore, India, 2007. December 17-20..

[8] J. Li and C. Peng, *"jQuery-based Ajax General Interactive Architecture,"* in Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference, Beijing, 2012.

[9] N. R. Dissanayake, G. K. A. Dias and M. Jayawardena, *"An Analysis of Rapid Application Development of AJAX based Rich Internet Applications,"* in International Conference on Advances in ICT for Emergin Regions (ICTer), Colombo, Sri Lanka, 2013.

[10] A. Mesbah and A. v. Deursen, *"An Architectural Style for AJAX,"* in Software Architecture, 2007. WICSA '07. The Working IEEE/IFIP Conference, Mumbai, 2007.

[11] N. R. Dissanayake and G. K. A. Dias, *"The Significance of Importance of an Architectural Pattern for AJAX Based Rich Internet Applications,"* in KDU International Reseach Conference 2014, Colombo, 2014.

[12] N. R. Dissanayake and G. K. A. Dias, *"Essential Features a General AJAX Rich Internet Application Architecture Should Have in Order to Support Rapid Application Development,"* International Journal of Future Computer and Communication, vol. 3, no. 5, pp. 350-353, 2014.

[13] D. M. Selfa, M. Carrillo and M. d. R. Boone, *"A Database and Web Application Based on MVC Architecture,"* in Electronics, Communications and Computers, 2006. CONIELECOMP 2006. 16th International Conference, 2006.

[14] *"Zend Framework,"* Zend Technologies Ltd, 2015. [Online]. Available: http://framework.zend.com. [Accessed 30 03 2015].

[15] *"EllisLab CodeIgniter,"* EllisLab, Inc, 2015. [Online]. Available: https://ellislab.com/codeigniter. [Accessed 30 03 2015].

[16] *"AngularJS,"* Google, 2015. [Online]. Available: https://angularjs.org. [Accessed 30 03 2015].

[17] N. R. Dissanayake, G. K. A. Dias and C. Ranasinghe, *"RIA-Bus: A conceptual technique to facilitate the AJAX-based rich internet application development,"* in Proceedings of the Research Symposium of Uwa Wellassa University, Badulla, Sri Lanka, 2015.