Delta Communication: The Power of the Rich Internet Applications

Nalaka R. Dissanayake and G. K. A. Dias

***Abstract*—Web applications have evolved into much complex Rich Internet Applications, providing rich features and enhanced user experience. Proper definitions, which deliver abstract realization of the fundamental concepts of the Rich Internet Application are still missing. Definitions provide proper understanding of the subject and help in increasing the realization of the characteristics of the same. This paper focuses on conceptually identifying the characteristics of the rich communication model of the Rich Internet Applications – commonly known as the "Asynchronous Communication" – suggesting a better term "Delta-Communication", followed by a definition for it. Based on the definition, we propose a term and a definition for the abstract model of the simplest rich communication technique, which is exploited in AJAX. Additionally, based on the proposed definition for the rich communication model, the paper proposes a new term to replace the term AJAX to overcome the limitations expressed by the term AJAX. These terms and definitions specify and express the fundamental characteristics of the abstract concepts of the rich communication model of the Rich Internet Applications.**

***Index Terms*—AJAX, asynchronous, communication, delta, rich internet applications.**

I. Introduction

Rich Internet Applications (RIAs) are very complex Web based systems [1], [2]. RIAs are encompassed with various types of components, integrated in diverse ways, making the RIAs more complex systems [3]. Numerous Technologies and Techniques (TTs), frameworks, libraries, and tools had been introduced over the last decade for the development of these components of RIAs [4]. However, these TTs, frameworks, libraries, and tools do not improve the fundamental concepts in the core of the RIAs [2], and the complexity remains due to the lack of realization of the abstract architectural formalism of the RIAs. Moreover, these TTs may introduce new layers of complexity and learning curves into development. Therefore, in the terms of the complexity, the advancement of TTs and tools does not provide much assistance.

Even though lots of tools had been introduced and many researches had been conducted in the domain of RIAs, still few researches have focused on the abstract fundamental concepts of the RIAs; and standard definitions for these concepts have still not been articulated [5]. Definitions of the concepts provide precise common understanding of the subject towards proper utilization of the concept. If we can realize the fundamental abstract (by the means of TTs independent) architectural elements of RIAs and their characteristics well enough and strongly define them, the complexity can be reduced; since the complexity encloses the difficulties in understanding the software systems [6].

Our ongoing research intends to reduce the complexity of the RIAs by identifying the fundamental abstract architectural elements of RIAs, specifying their characteristics, and introducing terms and definitions for them; in the direction of introducing an abstract architectural style for RIAs. This paper focuses on the rich communication model of the RIAs, which can be seen as the power of the RIAs (This is further discussed in section III). This rich communication model has two communication modes according to the direction of the data transmitted: 1) data-pull and 2) data-push. In data-pull mode, the client requests and fetches data (or pulls data) from the server, based on the traditional request-response model. In data-push mode, the server-components push the data to the client-component(s) when needed, without receiving a request from the client. In this paper we focus only on the data-pull mode.

This paper discusses the fundamental characteristics of the rich communication model of the RIAs, as identified through the literature survey and empirical evidence gained from a series of experiments; then proposes a definition, indicating the limitations of the general term "Asynchronous Communication", which is currently used to denote this rich communication model. Based on this definition, the paper also delivers a term and a definition for the simplest abstract technique, which can be implemented using this rich communication model; which is exploited in the popular technique named Asynchronous Javascript And Xml (AJAX) [7]. Additionally, the paper proposes a better conceptual term to replace the term AJAX, considering the outdated impression expressed by the words in the abbreviation AJAX.

The proper understanding delivered via the definition of the abstract fundamental concept of the rich communication model of the RIAs, will increase the realization of the architectural connector element [8] of rich communication within the system. We hope to utilize this realization, to simplify the complexity of the RIAs, by separating the rich communication connector from the other components, towards deriving an architectural style for RIAs. We expect that the knowledge delivered in this paper can be utilized to improve the available tools, and also to introduce new momentous tools within the context of the rich communication connector of RIAs.

A. Methodology

Manuscript received January 24, 2017; revised April 10, 2017.

Nalaka R. Dissanayake is with the Informatics Institute of Technology, 57, Ramakrishna road, Colombo 6, Sri Lanka (e-mail: nalakadmnr@gmail.com).

G. K. A. Dias is with the University of Colombo School of Computing, Reid Avenue, Colombo 7, Sri Lanka (e-mail: gkad@ucsc.cmb.ac.lk).

doi: 10.18178/ijfcc.2017.6.2.484

We conducted a literature survey in the domain of RIAs covering the techniques like AJAX [7], technologies like Websocket [9], and some other related concepts like design patterns and architectural styles, associated with the rich communication model. The literature survey went to the depth of the specifications of the related techniques and technologies. Additionally, the literature survey enclosed the related frameworks, libraries, and tools, in order to understand how the rich communication handling is implemented in them.

A cross-sectional survey was conducted to examine the experience of the developers on developing the rich features in RIAs, the complexity level of developing them, and how the complexity is related to the difficulty of developing of rich communication utilizing features [10].

Even though the work covered in the paper is conceptual, in order to draw the arguments, we required to gain empirical evidence of the knowledge we gathered through the literature survey. A series of experiments were conducted to gain empirical evidence within the domain. These experiments were prototype based and continued in incremental iterations. The knowledge gained in early iterations were utilized to realize the characteristics of the fundamental concepts of the RIAs, in order to specify and define them, and to come up with better techniques to develop them. The knowledge gained in early iterations was applied to the later iterations, and also refined, as the conceptual realization improves. HTML5 and CSS3 were used to develop the pages of the prototypes; JavaScript (JS) and jQuery [11] were used to develop the client-side components; PHP was used to develop the server-side components; and MySQL was used to implement the databases. The prototypes were hosted in a local Apache server, using XAMPP tool.

B. Structure of the Paper

Section II delivers the background of the RIAs, discussing the available approaches to develop RIAs, indicating the approach focused in this paper. Section II also delivers the background of AJAX, since AJAX contains the basics of the focused rich communication model of the RIAs.

Section III discusses the characteristics of the rich communication model of the RIAs. Then the section introduces the new term "Delta-Communication" (DC) and its definition, focusing on the real power of this rich communication, while indicating the flaws of its current generic term "Asynchronous Communication".

Based on the introduced definition for the rich communication model of RIAs, section IV proposes a new term and a definition to describe the simplest rich communication implementation technique. Followed by that, the section also proposes a new term to replace the term AJAX, indicating the flaws of its current meaning.

Section V concludes the paper, specifying future work of our ongoing research.

II. Background

The first segment under this section delivers the background of the RIAs, stating the available approaches used to develop the RIAs, and specifying the approach focused in this paper. This segment also discusses the notion behind the richness of the RIAs. The second segment discusses about the XHR object and its usage in implementation of AJAX technique, since AJAX introduced the JS based RIA development approach and implements the simplest form of the rich communication model, which is focused in this paper.

A. Rich Internet Applications

The term "Rich Internet Application" had been first used by Jeremy Allaire at Macromedia, in 2002; introducing their new technology named "Macromedia Flash MX", which is a client-side application development platform with dedicated TTs [12]. As per Jeremy, the RIAs are supposed to have "media-rich power of the traditional desktop with the deployment and content-rich nature of Web applications" [12].

After Flash, different RIA development TTs had been introduced, and they can be specified under three approaches: 1) proprietary plugin based approach, 2) Open source JS based approach, and 3) the lease known browser-based approach [13], where Flash falls into the plugin based approach. The JS based approach became popular with the introduction of AJAX, while other two approached have faded away. The scope of this paper sets within the JS based approach. However, while experimenting we have noted that despite the approach and the TTs used in the development, they all share similar abstract fundamental concepts.

RIAs are capable of delivering rich features minimizing the user experience gap between the traditional Web applications and the desktop applications. The richness of the features in RIAs can be analyzed into three main aspects: 1) the rich GUIs, enabled by the advancement of client-side development TTs, which allow building desktop applications like GUIs; 2) the rich communication model – commonly known as the asynchronous communication – which permits to communicate only the smaller and needful set of data, faster and also asynchronously; and 3) the enhanced user experience, enabled by the synergy of the aforementioned two features [1], [4], [14], [15].

The rich communication model of the RIAs plays a vital role in the development of the rich features. Without it, the rich GUIs will communicated with the server via the traditional request-response model, based on the work-wait pattern, which is incorporated with page refreshes, thus slower and low in user experience. To implement the rich communication model of the RIA, different TTs are available like AJAX [7], Polling [16], Server-Sent-Events (SSE) [17], and WebSocket (WS) [9].

B. XHR Object and AJAX

Microsoft was working on a technology named XMLHTTP in their Exchange 2000 project [18], and it was first introduced to the world as an ActiveX control in Internet Explorer 5.0 in March 1999 [19], [20]; and later it was called the XMLHttpRequest (XHR) object, which has an Application Programmer Interface (API) in JS.

In 2005, Jesse James Garrett from Adaptive Path coined the name AJAX, introducing the first JS based rich communication technique for Web applications, which

utilizes the XHR object [21]. This technique became popular and took the traditional Web applications to a whole new era called Web2, which is the era of the RIAs. Later W3C acquired the control of the XHR object and released the first specification on 2006 [22]. Since then the term AJAX has become an informal name for RIAs, where even some developers refer the RIAs as AJAX applications.

AJAX can be seen as the beginning of the JS based RIA development approach, and it became a major breakthrough in the Web development area [23]. After its introduction, developers were learning how to use AJAX to create desktop-like GUIs in Web applications such as Google Maps; and later they subsequently used AJAX even to create entire enterprise RIAs [14]. Using the JS's ability to manipulate the Document Object Model (DOM) in HTML documents, AJAX achieves and enhances the interoperability capability of Web applications [23]. AJAX is a data-pull technique, employing the traditional request-response model.

It should be noted that the AJAX itself is not a technology, it is a technique; and the technology behind AJAX is the XHR object with its JS API. Combining HTML and CSS with JS, AJAX has become a powerful tool in RIA development, providing the fundamental implementation of the rich communication model.

III. Defining the Rich Communication Model of the RIAs

The first segment of this section discusses the features/ characteristics of the rich communication model of the RIAs, in order to gain adequate realization of the abstract concept behind it. The second segment generalizes and rationalizes the characteristics and introduces a term and a definition for the abstract concept behind the rich communication model of the RIAs.

A. Understand the Rich Communication Model

In pre-AJAX classical Web applications, when the user preforms an action and if the processing needs to communicate with the server, then the client-component will send a request to the server. When the response is received, the browser renders the results by refreshing the page, loading a complete web page and related resources, which are supposed to be the content of the response. Once the request is sent, till the response is received – and loaded completely to the browser – the user has to wait, being idle. This setting is called the work-wait pattern [24], which makes the mode of this traditional communication processing synchronous, thus slow and lack in responsiveness. As Mozilla Developer Network says "synchronous requests block the execution of code, which creates 'freezing' on the screen and an unresponsive user experience" [25], lowering the user experience. In summary, the communication model of the traditional web applications incorporates page refreshes and work-wait pattern, and it is always synchronous, where all these facts can be seen as limitations for providing good user experience.

There are three main aspects behind the rich communication model of the RIAs, which is generally known as the asynchronous communication, which helps to overcome the limitations of the communication model of the traditional web applications.

1) Background processing and partial page rendering

The rich communication of RIAs is processed in the background, behind the GUI of the Web page, and out of the sight of the user [24]. With advancement of client-side development technologies like JS, the results are displayed on the GUI by partial rendering the visual content. Background processing together with partial page rendering eliminates page refreshes. Exploiting the power of this feature, development of an entire RIA system on a single web page has become possible.

Looking into this setting deep, in traditional web applications, the requests are sent to the server by the browser, and the response is sent back by the server to the browser. When the response is received by the browser, the browser engine processes it and the rendering engine of the browser produces the visual representation of the response [26]. Since the response is a complete Web page (it could also be some resource like an image), the rendering engine replaces the current GUI with the new GUI of the response, by loading and displaying the new page. In contrast to this, in rich communication model of the RIAs, the response is passed to the JS interpreter of the browser, then processed by a dedicated client-component, developed using JS or JS based technology like jQuery [26]. The processing of the response is being done in the background, while the GUI of the latest web page is already displayed on the browser. By the end of the processing, the client-component will update the current GUI with the results of the processed response, by partial rendering the necessary sections of the page, instead of replacing the last web page [27].

2) Faster communication

In traditional web applications, when the client-components need to communicate with the server-components, even for a small change, once the request is sent, a complete web page and all the related resources are needed to be loaded to the browser as the response. In the rich communication model of the RIAs, only the required set of data by the particular feature – which is been executed by the user at that time – can be communicated. It is prominent that in this rich communication model, since only the necessary set of data is communicated – instead of complete Web page and related resources – the size of the data communicated is relatively smaller, affecting the communication to be accomplished faster. It also helps to reduce the bandwidth use of the network.

The increased speed of the communication helps to eliminate the work-wait pattern. The fast communication, along with background processing capabilities enable developing responsive rich features on web pages.

3) Synchronous vs. asynchronous mode

The distinct feature of the rich communication model of the RIAs is generally considered as the asynchronous mode of it, and this model is generally called asynchronous communication, because of this ability. However, the asynchronous mode is a controversial aspect

In asynchronous mode, the GUI of the web page does not

get blocked once the request is sent, till the response is arrived, processed, and the results are displayed [28]; instead, the user can continue using the application/GUI. In this asynchronous mode, once the response is received, it is processed in the background, by dedicated application components in the client-side [28]. The asynchronous communication model facilitates the user to experience the results of the communication, even without knowing the execution or the processing of the request.

Basically, there is nothing synchronous or asynchronous in this communication; it's only the way the features, which utilize this communication is developed. Through the experiments we noted that the user experience of the rich features, which utilize the rich communication model, can be offered in either synchronous (blocking) or asynchronous (non-blocking) mode; and it is determined upon the requirements. For example, in a case of performing a transaction, the user has to wait till the transaction is completed, before continuing to use some other features of the application; therefore it should be implemented in synchronous mode. In a case of sending a mail, the user can click on the send mail button and continue reading other mails, while the mail is being processed and sent in background; thus it can be implemented in asynchronous mode.

B. Defining Delta-Communication

Considering the aforementioned three aspects of the rich communication model of the RIAs, we argue that the most important aspect is its increased speed due to the smaller size of data communicated.

As discussed above, even though the rich communication model of RIAs is generally called asynchronous communication, not always the features are implemented providing asynchronous user experience. Therefore, even the asynchronous processing is unique for the rich communication model, it is not prominent in increasing the user experience of the user of RIAs. Since the execution is faster due to the fast communication, even for synchronous features the users may not experience the work-wait pattern similar to traditional web applications. If the communication is not fast and the asynchronous features are processed slower, then the GUI might not get updated adequate, thus the users might get confused with the information on the GUI. Therefore the increased speed is an essential factor, even though the features are implemented in asynchronous mode. Furthermore, if the communication is slower, the background processing capabilities alone will not be able to increase the user experience, since still there will be the work-wait pattern engaged. The synergy of the increased speed of the communication and the partial page rendering, increases the user experience, closer to the desktop applications.

Considering these facts, we can see that the increased speed is the prominent factor of this communication's richness towards increasing user experience, eliminating the work-wait pattern. As mentioned before, the smaller size of the data is the reason for the increment of the speed. Therefore, we propose to use the term "Delta-Communication" (DC) – as already had been used by Mesbah *et al*. [4] – to denote the rich communication model of RIAs, which emphasizes the smaller size of the data communicated.

Considering the facts discussed above, we define the DC as follows.

> Delta-Communication is the rich communication model used by the rich features of the RIAs, for client-component(s) to communicate with the server-component(s), to exchange only the needful dataset – for a particular feature executed at the time – which is smaller, compared to the size of the request/response of traditional communication. Since the size of the dataset communicated is smaller, the communication completes faster, eliminating the work-wait pattern. The processing of the response is done by the client-components in the background, therefore the page refreshes are eliminated and replaced by partial page rendering to update the content of the GUI with the results of the response. The user experience can be determined by the implementation of the feature, in either blocking (synchronous) or non-blocking (asynchronous) modes.

Rich GUIs of the RIAs can contain multiple features on the same page. Once a page is loaded to the browser, all the communications with the server, for all the features implemented on that page, can be done using DC, eliminating page refreshes; until the page is refreshed or redirected to another page, explicitly by the user or by an internal component.

IV. Defining Simple Pull Delta-Communication Model

The AJAX technique uses the data-pull mode, and supports both synchronous and asynchronous modes by API. Through experiments we have noted that the DC technique implemented by AJAX is the simplest form of the DC. Furthermore, the underlying concept used to implement the AJAX technique can be developed in other environments – like desktop applications – using other TTs like WS or C#, even without JS and/or XHR object. We have successfully developed and tested the technique using C#.Net, to develop a desktop application component, which communicates with server components using data-pull DC, asynchronously. Moreover, the same DC technique is utilized in other advanced DC technologies like WS, for the data-pull mode.

To denote the abstract concept of this simplest implementation of the DC model, we propose the term "Simple Pull Delta-Communication" (SPDC). Since the SPCD concept is abstract, it is TTs independent and can be developed for both browser-based and non-browser-based clients. We defined the SPDC technique as follows.

> Simple Pull Delta-Communication is the basic abstract Delta-Communication technique, based on the data-pull mode. It describes the simplest form of data-pull Delta-Communication, based on the request-response model; and this technique is technology independent.

Fig. 1 shows the SPDC model. It can be seen as a generalized version of the AJAX architecture. The AJAX engine has been replaced by the DC engine, and the XML-HTTP-Request/response is replaced by the DC request/response.

A. JS Based SPDC

Aligning to the definition of the SPDC, we can look at the

AJAX technique as a JS implementation of the SPDC, which is limited to the browser based applications. We find that the term AJAX expresses some flaws, and it is outdated with regards to the latest API version of the XHR object. Due to the evolution of the XHR object, the technical scope of the AJAX had been expanded in terms of both XML and Asynchronous aspects.

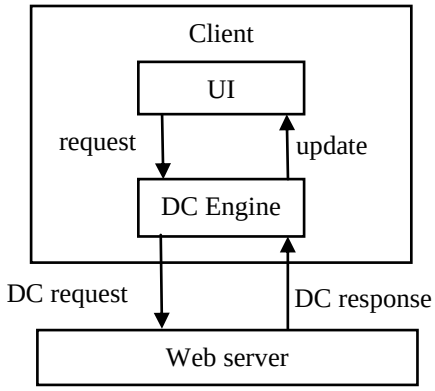

Fig. 1. Simple pull delta-communication model.

In W3C's first draft of the XHR object, the result from the server for a client request was accepted – as the name of the XHR object implies – in XML [22]. However, according to the latest draft, the XHR object supports various data types, such as plain-text, XML, JSON, Blob [7].

Furthermore, the XHR object has the ability to develop the rich communication not only in asynchronous mode, but also in synchronous mode, since the XHR API contains a setting to configure the synchronous/asynchronous mode. However, even using the asynchronous setting of the XHR object, still AJAX features can be developed and presented to the users to perform DC in synchronous manner, and the mode of the DC is determined by the way the feature is designed. In later rich communication development technologies like WS, a similar setting is not included, and the developers are given the flexibility to determine, in which mode the DC should be developed.

Based on these facts, considering the outdated and limited impression of the term "AJAX", we propose the term "JavaScript-based Simple Pull Delta-Communication" (JS-SPCD), in place of the term "AJAX". The term JS-SPCS indicates that it utilizes the SPDC technique, and developed using JS.

V. Conclusion

This paper has discussed the abstract characteristics of the rich communication model of the RIAs, which can be considered as the power of the RIAs. Considering the ability to communicate faster due to the smaller data-sets communicated, we have proposed the term Delta-Communication to denote the abstract concept of the rich communication model of the RIAs, followed by a definition for it.

Extending the definition of the DC, we have defined the simplest form of the DC implementation as SPDC. SPDC can be seen as the abstract version of the technique used in AJAX. And based on the definition of the SPDC, we have proposed a new term JS-SPDC in place of the term AJAX, by considering the advancements of the latest version of the XHR specification in the aspects of Asynchronous and XML.

Utilizing the realization provided by these definitions of the TTs independent abstract concepts of the RIAs, we expect to separate the DC components within the RIAs into a dedicated connector architectural element. This separation will increase the simplicity of the architectural realization of the RIAs, and also the visibility of the DC. We hope to direct this knowledge towards designing an abstract architectural style for RIAs, which can reduce the complexity of the RIAs, by increasing the realization of the abstract characteristics of the RIAs.

References

[1] M. Busch and N. Koch, "Rich internet applications-state-of-the-art," Ludwig-Maximilians-Universitat, Munchen, 2009.
[2] J. Kuuskeri and T. Mikkonen, "Partitioning web applications between the server and the client," in *Proc. SAC'09*, Honolulu, Hawaii, U.S.A., 2009.
[3] J. Offutt, "Quality attributes of web software applications," *IEEE Software special issue on Software Engineering for Internet Software*, vol. 19, no. 2, 2002.
[4] A. Mesbah and A. V. Deursen, "An architectural style for AJAX," in *Proc. Working IEEE/IFIP Conference on Software Architecture*, Mumbai, 2007.
[5] S. Casteleyn, I. Garrigo's, and J.-N. Mazo ́n, "Ten years of rich internet applications: A systematic mapping study, and beyond," *ACM Transactions on the Web*, vol. 8, no. 3, pp. 18:1-18:46, 2014.
[6] H. Zuse, *Software Complexity Measures and Models*, New York: de Gruyter & Co., 1992.
[7] W3C. (2014). XMLHttpRequest Level 1. [Online]. Available: http://www.w3.org/TR/2014/WD-XMLHttpRequest-20140130/
[8] R. T. Fielding, "Architectural styles and the design of network-based software architectures," University of California, Irvine, 2000.
[9] I. Fette, "The WebSocket protocol," *Internet Engineering Task Force*, 2011.
[10] N. R. Dissanayake and G. K. A. Dias, "What does the AJAX rich internet applications need to support the rapid application development," presented at 3rd International Conference on Human Computing, Education and Information Management System, Sydney, Australia, 2014.
[11] (2015). The jQuery Foundation. jQuery. [Online]. Available: https://jquery.com/
[12] J. Allaire, "Macromedia flash MX — A next-generation rich client," Macromedia, San Francisco, 2002.
[13] J. Farrell and G. S. Nezlek, "Rich internet applications the next stage of application development," in *Proc. the ITI 2007 29th Int. Conf. on Information Technology Interfaces*, Cavtat, Croatia, 2007.
[14] G. Lawton, "New ways to build rich internet applications," *Computer*, vol. 41, no. 8, pp. 10-12, August 2008.
[15] N. Koch, M. Pigerl, G. Zhang, and T. Morozova, "Patterns for the model-based development of RIAs," Springer, ICWE, Heidelberg, 2009.
[16] M. Carbou, "Reverse ajax, part 1: Introduction to comet," *IBM*, 2011.
[17] I. Hickson. (2015). Server-sent events. [Online]. Available: http://www.w3.org/TR/eventsource
[18] A. Hopmann. (2015). The story of XMLHTTP. [Online]. Available: http://www.alexhopmann.com/xmlhttp.htm
[19] S. Dutta. (24 January 2006). Native XMLHTTPRequest object. Microsoft. [Online]. Available: http://blogs.msdn.com/b/ie/archive/2006/01/23/516393.aspx
[20] K. Smith, "Simplifying ajax-style web development," *Computer*, pp. 98-101, May 2006.
[21] J. J. Garrett. (2005). Ajax: A new approach to web applications. *Adaptive Path*. [Online]. Available: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications
[22] W3C. (2006). The XMLHttpRequest object. [Online]. Available: http://www.w3org/TR/2006/WD-XMLHttpRequest-20060405
[23] S. Salva and P. Laurencot, "Automatic Ajax application testing," presented at Fourth International Conference on Internet and Web Application and Services, Venice, 2009.
[24] D. Crane, B. Bibeault, and J. Sonneveld, *Ajax in Practice*, Greenwich: Manning Publications, 2007.

[25] (2015). Synchronous and asynchronous requests. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Synchronous_and_Asynchronous_Requests
[26] A. Grosskurth and M. W. Godfrey, "Architecture and evolution of the modern web browser," Elsevier Science, 2006.
[27] J. Li and C. Peng, "jQuery-based ajax general interactive architecture," in software engineering and service science (ICSESS)," presented at 2012 IEEE 3rd International Conference, Beijing, 2012.
[28] T. S. Manu, "Advances & applications in ajax," *CUSAT*, 2011.

Nalaka R. Dissanayake was born in Anuradhapura, which is a sacred city in Sri Lanka, in 1982. He received the B.Sc. degree in information technology from Sri Lanka Institute of Information Technology, in 2007.

From 2007 to 2011, he was working as a student instructor, instructor, lecturer and a software designer in various institutes. Since 2012, he has been an assistant lecturer with the Informatics Institute of Technology, Colombo, Sri Lanka. He is the author of 24 peer reviewed conference papers and a journal paper. His research interests include software architectures, web engineering, and rich internet applications.

Mr. Dissanayake is currently reading for M.Phil in University of Colombo School of Computing, University of Colombo, Sri Lanka.

G. K. A. Dias received the bachelor of science degree in 1982 from the University of Colombo, post graduate diploma in computer studies in 1986 from University of Essex UK and MPhil by research in 1995 from the University of Cardiff. He is a member of the Association for Computing Machinery (ACM) and Member of the Computer Society of Sri Lanka. He is also a member of the Modelling and simulation research group of the University of Colombo School of Computing (UCSC).

He is an author of one book and a co-author of 4 books, and more than 30 publications. His research interest's incudes Computer aided software engineering, modelling and simulation and Computer aided education. He is currently a Grade 1 Senior Lecturer and served as the Head of the Communication and Media Technologies Department of the UCSC for 5 years (2010-2015). He has also served as the MPhil Coordinator of UCSC for 5 years (2010-2015).

Mr. Dias co-authored a publication titled "Developing a tourist arrivals forecasting system for Sri Lanka, in the Ruhuna International Science and Technology Conference, Matara, Sri Lanka: Jan, 2015, which won the best poster presentation award. Mr. Dias was in-charge of the K-8 Flight Simulator project jointly done with CRD Ministry of Defense, which won the Bronze award at the NBQSA 2015 (National Best Quality Software Award) under Education & Training category.