Rich Web-based Applications: An Umbrella Term with a Definition and Taxonomies for Development Techniques and Technologies

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

***Abstract*—There is a wide range of web-based applications, which use a rich communication model such as Rich Internet Applications, mobile apps, cloud-based systems, Internet of Things based systems, etc.; however, a proper term to address them all and a definition, which covers and explains the common characteristics of them are missing. A definition of a concept is important as the definition provides a precise common understanding of the focused artifacts, which helps in increasing the realization of these artifacts towards proper utilization of them. We propose the umbrella term Rich Web-based Application to address the aforementioned types of applications, and also propose a definition for it, aligning to the common architectural characteristics of these applications. In addition, we deliver a set of taxonomies to classify the techniques and technologies related to the development of Rich Web-based Applications, in the direction of improving the understanding of the proper utilization of them. In future, we expect to introduce an architectural style for these Rich Web-based Applications, based on the proposed definition.**

***Index Terms*—Definition, delta communication, development techniques and technologies, rich internet application, taxonomy, web-based application.**

I. Introduction

The concept of the service of the web was introduced in the early 90s [1], and it became popular, allowing the features of other types of network-based services such as email and file sharing, to be delivered via the service of the web. Nowadays, the client-server based distributed systems – commonly known as "web applications" – are mainly built exploiting the service of the web and this domain is evolving and expanding rapidly, introducing new breeds of systems with a variety of features, which use dedicated Techniques and Technologies (TTs) to develop them.

It is not easy to understand both the common and specific characteristics of these systems, and still, there are no proper definitions available to explain them too. The definitions and taxonomies of the concepts are important as they provide a precise common understanding of the focused subject, which helps in increasing the realization of the concept towards proper utilization of it. We have studied the systems, which utilize the service of the web, in depth, and have introduced the concept of web-based application and a definition for it, also discussing their development TTs [2]. The concept of web-based applications is detailed in Section II.A.

In our ongoing research, we narrowed down our study, focusing on the web-based applications, which utilize a rich communication model, explained in Section II.B. The systems in this focused domain are commonly known as Rich Internet Applications (RIAs); the characteristics of them are discussed in Section II.C.

Continuing the study, we noted that there are varieties of systems such as mobile apps, cloud-based systems, and Internet of Things (IoT) based systems, which share some common characteristics with the RIAs. Our ongoing research is focusing on introducing an abstract architectural style for these systems; thus, we required a common basis to address all these systems. While doing an in-depth literature survey we understood that still the common architectural characteristics of these systems have not been understood, and also a proper definition is not available.

In this paper – in Section III – we propose an umbrella term "Rich Web-based Application" (RiWA), to address a wide range of web-based applications, which use the rich communication model, commonly used by the RIAs. Then we propose a definition for the RiWA by extending the definition of the web-based application, aligning to the identified common architectural characters of the RiWAs. Based on the proposed definition, we present taxonomies for the development TTs of the RiWAs in Section IV, towards improving the understanding of the proper utilization of them.

We expect to continue our research in the direction of introducing an architectural style for the RiWAs, based on the proposed definition in this paper, which can help in realizing a wide range of systems covered by the proposed umbrella term "Rich Web-based Application".

A. Methodology

An intensive literature survey was conducted, focusing on the 1) web, internet, and related concepts like protocols, 2) web applications, RIAs, and related concepts like the rich communication model, 3) other types of web-based applications, which utilize the rich communication model, like mobile apps, etc., and 4) the development TTs of these systems. The search was executed online using Google Scholar search engine.

Towards experiencing the findings of the literature survey and gaining empirical evidence, a series of experiments were conducted. These experiments were focusing on

Manuscript received February 15, 2018; March 27, 2018.

Nalaka R. Dissanayake is with the Sri Lanka Institute of Information Technology, New Kandy Road, Malabe, Sri Lanka (e-mail: nalakadmnr@gmail.com).

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

identifying and understanding the common architectural characteristics of the web-based applications, which utilize the rich communication model and their development TTs. These experiments were prototype based and continued in incremental development. The facts identified during the literature survey were examined in the early iterations, then new facts were identified, and they were tested in later iterations. Solutions for the identified issues in early iterations were continuously tested and refined in later iterations. The new concepts such as web-based applications, Delta Communication (DC), Rich Web-based Applications (RiWA) were constructed inductively, to assist the discussions of the ongoing research, in the direction of introducing an architectural style for the rich applications, which are based on the concept of the web.

Various TTs – as discussed in Section IV – were used to develop the prototypes for testing different types of applications as well as their development TTs. Early prototypes were mainly browser-based applications, developed using HTML, JavaScript (JS), and PHP. The derived concepts, such as Simple Pull Delta Communication (SPDC) [3], were tested later using different server-side TTs like JAVA and .NET, and also tested for different other types of applications like mobile apps.

II. Background

This section discusses the basics of three main concepts, which lay the foundation for the rest of the paper.

A. Web-based Applications [2]

The term “web application” usually denotes a range of applications, which run in a web browser. In present, the use of the service of the web has expanded its limits, affecting the generic web applications to grow beyond the web browser. However, still the architecture of these systems are based on the client-server style, using the request-respond model over HTTP; thus, they can be grouped into a single domain, which is called Web-based Applications [2].

Web-based Application is defined as “*a system, with application component(s) in client-side [client-component(s)], which communicate(s) with application component(s) in a web server [server-component(s)], for processing data. They utilize the service of the web, based on the client-server architecture, request-response model, standard HTTP, and other related techniques and technologies*” [2]. Combination of the components of a standalone desktop application [client-component(s)] and components, which run in a web server [server-component(s)] is called as a standalone web-based application, which can be extended into a multi-tier web-based application, by adding external layers such as database layer, etc. [2].

Web-based applications can be mainly classified into two groups, based on the type of the client-components, as browser-based web-based applications and non-browser-based web-based applications [2]. The limitations of the web-based application can be seen as 1) poor Graphical User Interfaces (GUIs), 2) slow responses due to the work-wait pattern, 3) low user experience caused by the aforementioned limitations, 4) lack of management, maintenance, and modification, and 5) communication limitations [2]. Overcoming these limitations of web-based applications, a new breed of applications named Rich Internet Applications (RIAs) had been introduced, which use a rich communication model. The RIAs introduced a new era for the web, named Web2; and they have become increasingly important and popular. The rich communication model and other characteristics of the RIAs are discussed in the next two sections.

B. Delta Communication [3]

The rich communication model of the RIAs is called Delta Communication (DC), with three main characteristics: 1) capability of processing in background, which helps in partial page rendering, 2) faster communication than the communication used in web-based application, which leads to improving the responsiveness by eliminating the work-wait pattern, and 3) support for development of the features in both synchronous and asynchronous processing [3].

DC can be seen as the power of the RIAs and it is defined as “*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*” [3].

The basic model of the DC is called Simple Pull Delta Communication (SPDC) model, which is defined as “*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*” [3]. The JavaScript (JS) implementation of the SPDC – which is called the “JavaScript-based Simple Pull Delta-Communication” (JS-SPDC) – is the simplest implementation of the SPDC, which is commonly known as Asynchronous Javascript And XML (AJAX) [3].

C. 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 [4]. 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*” [4].

The client-side application in RIAs is called a rich-client, which is a thick-client. Jeremy suggests that the rich client development technologies should provide “*an efficient, high-performance runtime for executing code, content, and communications; Integrate content, communications, and application interfaces into a common environment; Provide* *powerful and extensible object models for interactivity; Enable rapid application development through components and re-use; Enable the use of Web and data services provided by application servers; Embrace connected and disconnected clients; and Enable easy deployment on multiple platforms and devices*", which Macromedia Flash MX attempted to address and enable [4].

The development approach of Flash is plugin-based and supports both browser-based and non-browser-based modes of client-components. Instead of discussing the Flash in depth, this paper tends to look into the other RIA development approaches, as discussed below.

1) Approaches for developing rich internet applications

There are three approaches for RIA engineering 1) proprietary plugin based approach; 2) Open source JS based approach, and 3) the least known browser-based approach [5]. Section IV details these approaches, specifying the development TTs used in them. The JS based approach has become popular while the other two approaches have faded away. Therefore, the features of the RIAs are mainly discussed in the context of JS based RIAs.

2) Features of RIAs

Lawton [6] explains RIAs as applications, which run online and have many of the features and functionalities of desktop applications. He continues the explanation showing that the RIAs have overcome problems with traditional Web applications – such as slow performance and limited interactivity – with their responsive UIs and interactive capabilities, which make the Internet-based programs easier to be used and more functional.

As per Koch *et al*. [7], RIAs are Web applications, augmented with desktop features, which use mechanisms of advanced communications for data handling, to execute operations on the client-side, minimizing the server requests. They mention that the rich look-and-feel, better responsiveness, performance, and accessibility of RIAs enthuse both the users and the software engineers; and improve user interaction facilities like drag-and-drop, multimedia presentations, while avoiding unnecessary page reloading.

Busch and Koch [8] also deliver a similar impression saying that the RIAs are Web applications, which provide look-and-feel similar to desktop applications, thus different from the earlier generation of Web applications. RIAs provide a variety of interactive GUI elements, the possibility of both on-line and off-line use of the application, and the transparent usage of the computing power of the client, server, and the network. Busch and Koch keep explaining that the RIAs are capable of processing data in both server and client, and the data exchange takes place in an asynchronous way so that the client stays responsive while continuously recalculating or updating parts of the UI, till the communication is being processed.

Furthermore, Busch and Koch [8] extend the explanation saying that the RIAs are complex Web applications based on thick-client architecture. On one side, the client processes and manages data, reducing the communication; and on the other side, most of the needed communication is done via DC; as the result of both these facts, the network traffic is reduced. They say that these facts also allow additional GUI features, increasing the usability and the interaction possibilities of the users.

3) Analysis of the feature of the RIAs

Taking the explanations of the experts into consideration, and based on the empirical evidence gained through the experiments, we would analyze all the features of the RIAs discussed above, into 3 main facts: 1) rich GUIs, enabled by the advancement of client-side development, which can bring desktop applications like GUIs and features; 2) DC, which allows communicating only the needful, smaller set of data, faster and asynchronously; and 3) the enhanced user experience, enabled by the cumulative effect of aforementioned two facts, which are delivered via advanced and rich features [6]-[9]. Based on these characteristics, the following section proposes and defines the concept of Rich Web-based Applications.

III. Defining the Rich Web-based Applications

Even though there are numerous researches that have been done in the past ten years in the domain of RIAs, still a standard definition for RIAs has not been articulated [10]. After surveying these researches in the domain, Casteleyn *et al*. [10] have captured the essence of RIAs and have introduced a definition: "*RIAs are Web applications that aim to provide the features and functionality of traditional desktop applications, thereby offering a richer, more satisfying user experience compared to traditional Web 1.0 applications. Therefore, RIAs must: (i) strive for better responsiveness, (ii) improve interaction capabilities, and (iii) provide a richer user interface.*"

While studying a variety of web-based applications and RIAs, we noted that the scope of the traditional RIA development is mainly limited to the Web browser in client-side (open source JS based approach is explicitly used for the browser); yet the server-side is still somewhat similar to traditional Web applications, other than requiring some additional dedicated components to handle DC. The above definition for RIAs by Casteleyn *et al*. [10] is also in favor of that.

We further noted that the scope of the RIAs can also be expanded beyond the browser – in client-side – similar to the concept of Web-based applications [2], as discussed in the Section II.A. For an example, there can be mobile Apps, which use DC, and they are not running in a web browser. This observation turned our focus into a new domain, which comprises of a variety of types of applications, which exhibits the characteristics discussed in the Section II.C.3. We surveyed for a proper term to address all these systems in the focused domain and identified that these types of applications are separately addressed and discussed, regardless of their common characteristics.

It was difficult to continue the discussions in our ongoing research without having a common basis to address the different types of applications in the focused domain. Considering the common features of these applications, which utilize DC, we propose the umbrella term "Rich Web-based Applications" (RiWAs) – to address all the types of applications, in and out of the browser – which exhibits the main characteristics of the RIAs as discussed in section

II.C.3. After a careful study of the architectural characteristics of these systems, we propose the following definition for the term "Rich Web-based Applications", which is based on the definition of the web-based applications [2].

Rich Web-based Application is a system, with application component(s) in client-side [client-component(s)], which communicate(s) with application component(s) in a Web server [server-component(s)], for processing data.

Rich Web-based Applications are based on the Client-Server architecture. The client-components of Rich Web-based Applications contain rich Graphical User Interfaces and advanced processing capabilities. For communication, other than standard HTTP, Delta-Communication techniques and technologies are used for faster communication, in both data-pull and data-push modes, which can be implemented in either synchronous or asynchronous mode. Rich Graphical User Interfaces of Rich Web-based Applications together with faster Delta-Communication provide an enhanced and rich user experience.

RiWAs can be explained in another perspective as RiWAs are the systems, which combine the power of the rich GUIs and DC TTs of RIAs with the Web-based applications. The scope of the RiWAs is wider than traditional Web-based applications in DC aspect; and wider than standard RIAs in types of client-components and their integration aspects. Furthermore, aligning to the types of Web-based applications based on the size [2], RiWAs can also be grouped as standalone RiWAs and multi-tier RiWAs.

The need for the introduction of this new domain RiWA and referring to it in this research is to address wider technological development possibilities, which are not likely covered generally by the term RIA under the domain of Web-based applications. The RiWAs can be seen as a hybrid concept of Web-based applications and RIAs; and RiWAs can gain benefits from the non-browser-based client-components, more than Web-based systems, as discussed in the following section.

IV. Techniques and Technologies for the Development of the RiWAs

Towards the TTs independency of an architectural style for RiWAs – which is the main focus of our ongoing research – it is essential to have an adequate understanding of the development TTs of the RiWAs and their proper utilization. This knowledge will also help in demonstrating how the intended style can be adopted in RiWAs development. An in-depth discussion of the features and characteristics of different types of RiWAs and their development TTs is intentionally avoided in this paper. Instead, this section analyses the RiWAs development TTs, classifies them, and presents taxonomies in the direction of structured understanding of their usage. We expect that these taxonomies will help to realize the functionalities of the components of the intended architectural style.

As Meliá *et al*. [11] say that the real challenge in SE is selecting the right and suitable TTs for the project from existing alternatives, and creating the optimal solution to satisfy the user requirements. Adequate understanding of the TTs used in the development of RiWAs may provide sufficient assistance in the effort of designing the proposed architectural style for RiWAs to be TTs independent.

To gain a structural knowledge of the TTs, classifications and taxonomies for them are presented in this section. This knowledge will also assist in the decision making of selecting proper TTs for the development of RiWAs, and hassle-less adoption of them, via the conceptual realization provided by the taxonomies.

Toffetti *et al*. [12] have presented an analysis of TTs and design methodologies for RIAs. Their TTs analysis is based on the available development approaches and technologies/platforms. They also discuss the level of abstraction of the technologies and also the supportive tools.

We classify the TTs of RiWAs into 4 categories, according to the architectural elements and aligning to the given definition of the RiWA: 1) the client-components, 2) the server-components, 3) connector elements, and 4) data elements. When developing RiWAs, decision making of selecting suitable and compatible TTs in all these elements is required.

We intend to keep the taxonomies abstract, leaving the leaf nodes to represent a specific type of Technique or Technology, instead of explicitly naming the available matching TTs. However, in the discussions, we have stated some examples towards giving a better understanding of the category. In-depth discussions of the specifications of these TTs are intentionally kept out of the scope of this paper. Instead, the intention is to classify them and introduce a taxonomy for a better understanding of their utilization.

A. *TTs for the Client-Components of RiWAs*

The client-components of RiWAs are similar to the client-components of Web-based applications [2]. Additionally, they should incorporate DC handing components, thus the ability of DC development is needed for the client-components development TTs. Fig. 1 illustrates the taxonomy for the client-components development TTs of RiWAs. It should be noted that when the client-side development of RiWAs is considered, it includes not only the application component development but also the Views/GUI development.

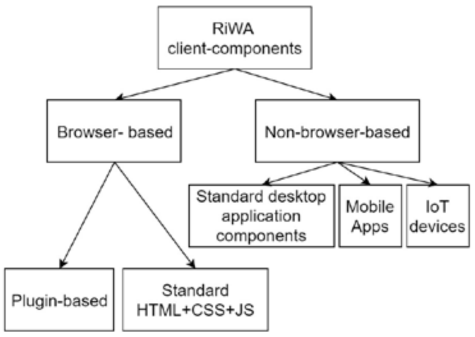

Fig. 1. Taxonomy for client-component(s) development TTs of RiWAs.

1) TTs for the Browser-based Client-Components of RiWAs

There are several approaches for the browser-based RiWAs, inherited from RIAs. The first approach, which is the proprietary plugin based approach, uses the technologies such as Adobe (former Macromedia) Flash/Flex [13], JAVA Applets, or MS Silverlight [5]. These technologies are enriched with utensils for developing desktop applications like rich GUIs. For the client-components developed using these TTs, there are additional requirements for installing and maintaining browser plug-ins or runtimes. It was the main reason for users to dislike the applications developed using this approach. The demand for this approach was reduced, after the introduction of AJAX and the evolvement of related JS-based TTs.

The second approach uses the JS-based TTs for the client-components [5]. This approach uses standard browser-based development languages: HTML and CSS for GUIs, and JS for behavior/processing development. With the introduction of HTML version 5, along with CSS version 3, the capabilities of JS-based client-components were increased. At present, JS has become powerful as the de facto application development language of HTML, and the JS-based TTs have become the default for the browser-based RiWAs. With compatible supplementary frameworks/libraries like Bootstrap [14] for GUI development, jQuery [15] and AngularJS [16] for client-component development, comprehensive enhanced features can be developed.

The third approach is the least publicized, which is known as the browser-based approach [5]. XUL language from Mozilla foundation can be given as an example of this approach. This approach and related TTs are not considered in this paper since it is not a widely used approach.

2) TTs for the non-browser-based client-components of RiWAs

The standard desktop application development TTs like JAVA or .Net and related libraries/frameworks can be utilized for non-browser-based client-components development. For DC development, these frameworks may contain their own tools or some third-party frameworks/libraries can be incorporated.

The TTs discussed under the proprietary plugin based approach (in Section IV.1.a) also have the capability of developing non-browser-based client-components, which are executed on dedicated runtimes; for example, Flash/Flex [13] use the Flash player or Adobe AIR runtimes. They are usually augmented with rich tools for developing communication connectors to utilize the service of Web.

Nowadays, the mobile apps also communicate with server-components thus, they can be seen as the client-components of RiWAs. Popular mobile development frameworks like Android and IOS encompass communication development tools into their packages.

Additionally, even devices in IoT based systems can be exploited as client components of RiWAs. Therefore, related TTs like Arduino Programming Language [17] for Arduino devices, Python for Raspberry Pi devices [18], and related frameworks/libraries/protocols like MQTT [19] [20] can also be listed under the non-browser-based client-component development TTs.

B. TTs for the Server-Components of RiWAs

The server-components of RiWAs are similar to the Web-based applications [2]. Additionally, in server-side, they need dedicated component(s) for handling DC. Fig. 2 illustrates the taxonomy for the server-component(s) development TTs of RiWAs.

The “Web application” node, represents the standard server-side development TTs such as PHP, JAVA, ASP.Net, Python, etc. and related frameworks/Libraries like CodeIgniter for PHP [21], Struts for JAVA [22], etc.

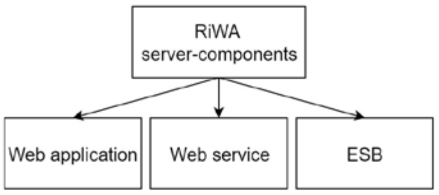

Fig. 2. Taxonomy for Server-component(s) development TTs of RiWAs.

The Web services use dedicated TTs such as SOAP [23], REST [24], and frameworks/libraries like JAX-WS [25] and JAX-RS [26] for JAVA, Slim for PHP [27], etc. The behavior of the web services is different from the Web applications in the context of service exposure to other components via Application Program Interfaces (API).

The Enterprise Service Bus (ESB) in Service Oriented Architecture (SOA) [28] or similar concepts can be used to extend the standalone RiWAs into multi-tire RiWAs using dedicated TTs. The concept of cloud computing [29] can also be related to these TTs, where the cloud-based systems provide a platform to deliver a service oriented functionalities. The concepts of Web services, SOA, and cloud computing incorporate wide and self-contained domains, which the deep discussions are intentionally avoided in this paper.

In the given taxonomy for the server-component(s) of RiWAs, all the types of TTs are supposed to be contained with DC implementation tools.

C. TTs for the Connector Elements of RiWAs

Connector development TTs can be seen as the core of RiWAs development TTs. In addition to the connector of Web-based applications, the connectors in RiWAs incorporate DC, where both the client and server should contain components for handling DC. For the communication in RiWAs, union of the communication development TTs of traditional Web-based systems and the DC development TTs is accepted.

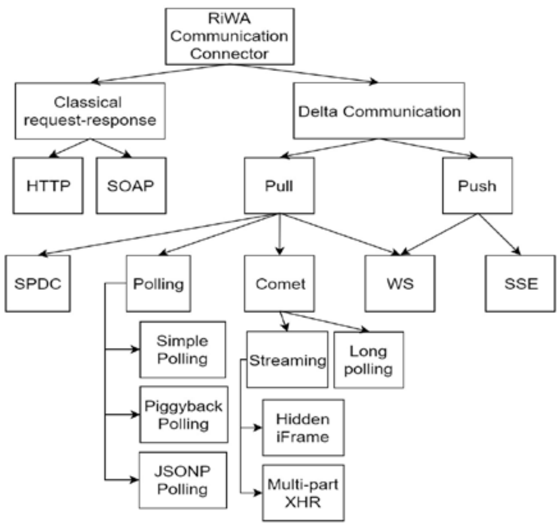

Fig. 3. Taxonomy for communication development TTs of RiWAs.

Fig. 3 illustrates the taxonomy for the TTs for the communication in connectors of RiWAs. In this taxonomy, we included the DC TTs identified via a deep literature survey.

If the client-component is non-browser-based, regular HTTP communication may not be required. Instead, all the communication could be accomplished using DC. For browser-based client-components, there is an approach of developing all the features in a single Web page, called single page paradigm. In such applications, other than the initial request for the Web page – which contains all the client-components – for rest of the communication, only DC can be used. For other standard RiWAs, a combination of regular HTTP requests – via HTML hyperlinks, JS redirections, or forms submission – and DC can be utilized, according to the requirements and the delivery of the features aligning to the system design.

After the introduction of AJAX/XHR [30] (SPDC [3]), the concept of the DC had been applied in some other TTs for both data pull and push modes, and each TT is associated with its own set of pros and cons [31]. These DC TTs can be mainly classified under data-pull and data-push modes.

The main limitation of the SPDC technique is that it supports only data-pull mode, therefore it does not suit for real-time data communication as in publisher-subscriber model or any other data-push concepts [31]. Several techniques had been introduced to simulate data-push using SPDC; some of them – like long-polling and streaming [32] – use the same XHR object as SPDC, thus also called reverse-AJAX. Later, a true data-push protocol named Server Sent Events (SSE) [33] was introduced, which is unidirectional, from the server to the client; however it did not become much popular. An advanced bi-directional DC protocol named WebSocket (WS) was introduced in 2011 [34], which supports both data pull and push modes, and it gained the attraction of the Web engineers.

A communication connector has two ends, usually one end in the client-side and the other end in the server. Therefore, once DC TT(s) is/are selected for a RiWA, compatibility of both the client and server TTs towards developing the selected DC TT(s) should be assured. For XHR [35] based TTs like SPDC/AJAX, Polling, and Long polling, a framework like jQuery [15] provides enough utilities for the browser-based client-side components, where standard server-side TTs like JAVA or PHP natively provides sufficient support. For much advanced WS, dedicated frameworks/libraries like socket.io [36] for browser-based JS clients, Ratchet [37] for PHP, spring [38] for JAVA, and Tornado [39] for python can be used.

D. TTs for the Data Element

Fig. 4 shows the taxonomy for the TTs used to prepare the data for communication – mainly for DC.

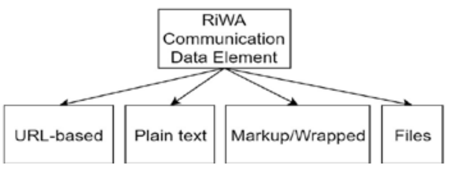

Fig. 4. Taxonomy for communication data TTs of RiWAs.

The URL-based technique represents the techniques like query string [40], [41] and REST API [42], [24]. Plain text can be considered as a word, sentence, paragraph, or even an essay, written in natural language. Markup/Wrapped node denotes the data wrapped by any markup language like XML [43] or even HTML or wrapped by other TTs like JSON [44]. The File node covers a range of file types and formats, including text, image, audio, video files.

When selecting a TT to prepare data for DC, not only the suitability of the TT for the target data type but also the final size of the prepared data also matters. The size of the data directly affects the communication speed, which is an important fact towards improving the user experience of RiWAs. Furthermore, the size of the data may affect some other aspects like network bandwidth. Therefore, careful understanding of the requirements is recommended in the direction of selecting TT(s) for the data element of the RiWAs.

V. Conclusion and Future Work

This paper has introduced the umbrella term “Rich Web-based Application (RiWA)”, which covers a variety of types of Web-based applications, such as RIA, mobile apps, cloud-based systems, Internet of Things based systems, etc., which share similar fundamental characteristics: 1) rich GUIs, 2) DC, and increased user experience. The paper also provides a definition for the RiWAs considering the similarities of the architectural characteristics of these applications; this definition extends the definition of the Web-based Applications. Additionally, aligning to the given definition for RiWAs, taxonomies to classify the development TTs of the RiWAs are introduced, towards improving the realization of utilization of these TTs.

Our ongoing research focuses on introducing an architectural style for the RiWAs, bearing the given definition in mind. We expect to make this style independent from the development TTs, thus, abstract, exploiting the realization provided by the taxonomies introduced by this paper.

References

[1] CERN, Ten years public domain for the original web software, April 30, 1993.

[2] N. R. Dissanayake and G. Dias, "Web-based Applications: Extending the General Perspective of the Service of Web," in *Proc. 10th International Research Conference of KDU (KDU-IRC 2017) on Changing Dynamics in the Global Environment: Challenges and Opportunities*, Rathmalana, Sri Lanka, 2017.

[3] N. R. Dissanayake and G. Dias, "Delta communication: The power of the rich internet applications," *International Journal of Future Computer and Communication*, vol. 6, no. 2, pp. 31-36, 2017.

[4] J. Allaire, "Macromedia flash MX — A next-generation rich client," *Macromedia*, San Francisco, 2002.

[5] 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.

[6] G. Lawton, "New ways to build rich internet applications," *Computer*, vol. 41, no. 8, pp. 10-12, August 2008.

[7] N. Koch, M. Pigerl, G. Zhang, and T. Morozova, "Patterns for the model-based development of RIAs," *Springer ICWE*, Heidelberg, 2009.

[8] M. Busch and N. Koch, "Rich internet applications - state-of-the-art," Ludwig-Maximilians-Universitat, Munchen, 2009.

[9] A. Mesbah and A. v. Deursen, "An architectural style for AJAX," in *The Working IEEE/IFIP Conference Software Architecture*, Mumbai, 2007.

[10] 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. 1-46, 2014.

[11] S. Meliá, J. Gómez, S. Pérez, and O. Díaz, "Architectural and technological variability in rich internet applications," *IEEE Internet Computing,* pp. 24-32, May/June 2010.
[12] G. Toffetti, S. Comai, J. C. Preciado, and M. Linaje, "State-of-the art and trends in the systematic development of rich internet applications," *Journal of Web Engineering,* vol. 10, no. 1, pp. 70-86, 2011.
[13] Adobe. (2018). *Adobe Flex*. [Online]. Available: https://www.adobe.com/products/flex.html
[14] Bootstrap. (2018). *Bootstrap*. [Online]. Available: https://getbootstrap.com/
[15] jQuery. (2018). *The jQuery Foundation*. [Online]. Available: https://jquery.com/
[16] AngulaJS. (2018). [Online]. Available: https://angularjs.org/. [Accessed 20 01 2018].
[17] Arduino. (2018). *Language Reference*. [Online]. Available: https://www.arduino.cc/reference/en/
[18] R. P. Foundation, "Python," *Raspberry PI Foundation*, 2018.
[19] MQTT. (2018). [Online]. Available: http://mqtt.org/
[20] Eclipse. (2018). *Go Client.* [Online]. Available: http://www.eclipse.org/paho/clients/golang/
[21] I. EllisLab. (2018). *EllisLab CodeIgniter*. [Online]. Available: https://ellislab.com/codeigniter
[22] Apache, "Apache struts," *The Apache Software Foundation*, 2018.
[23] W3C, *SOAP Version 1.2 Part 1: Messaging Framework*, Second Edition, 2007.
[24] R. T. Fielding, "Architectural styles and the design of network-based software architectures," University of California, Irvine, 2000.
[25] Oracle. (2013). Building Web Services with JAX-WS. [Online]. Available: https://docs.oracle.com/javaee/6/tutorial/doc/bnayl.html
[26] Oracle. (2013). Building RESTful Web Services with JAX-RS. [Online]. Available: https://docs.oracle.com/javaee/6/tutorial/doc/giepu.html
[27] Slim. (2018). [Online]. Available: https://www.slimframework.com/
[28] M. T. Schmidt, B. Hutchison, P. Lambros and R. Phippen, "The Enterprise Service Bus: Making service-oriented architecture real," *Ibm Systems Journal,* vol. 44, no. 5, pp. 781-797, 2005.
[29] A. Taivalsaari and T. Mikkonen, "Objects in the cloud may be closer than they appear towards a taxonomy of web-based software," in *Proc. 13th IEEE International Symposium on Web Systems Evolution (WSE),* Williamsburg, 2011.
[30] J. J. Garrett, "Ajax: A new approach to web applications," *Adaptive Path*, February 18, 2005.
[31] N. R. Dissanayake and G. Dias, "A comparison of delta-communication technologies and techniques," in *Proc. 10th International Research Conference of KDU (KDU-IRC 2017) on Changing Dynamics in the Global Environment: Challenges and Opportunities*, Rathmalana, Sri Lanka, 2017.
[32] M. Carbou, "Reverse ajax, Part 1: Introduction to comet," IBM, 2011.
[33] I. Hickson. (February 3, 2015). *Server-Sent Events*. [Online]. Available: http://www.w3.org/TR/eventsource/
[34] I. Fette, "The websocket protocol," *Internet Engineering Task Force*, 2011.
[35] W3C. (January 30, 2014). XMLHttpRequest Level 1. [Online]. Available: http://www.w3.org/TR/2014/WD-XMLHttpRequest-20140130/
[36] socket.io. (2018). [Online]. Available: https://socket.io/
[37] Ratchet. (2018). Ratchet WebSockets for PHP. [Online]. Available: http://socketo.me/
[38] Spring, *Pivotal Software*, 2018
[39] Tornado. (2018). [Online]. Available: http://www.tornadoweb.org/en/stable/
[40] Wikipedia. (2017). Query string. [Online]. Available: https://en.wikipedia.org/wiki/Query_string
[41] M. Belshe, Bitgo, R. Peon, I. Google, E. M. Thomson, and Mozilla, "Hypertext transfer protocol version 2 (HTTP/2)," *Internet Engineering Task Force (IETF),* 2015.
[42] RESTfulAPI.net, *REST Resource Naming Guide*, 2018.
[43] W3C, "Extensible Markup Language (XML) 1.0, Fifth Edition, Nov 26, 2008.
[44] E. T. Bray, "The JavaScript Object Notation (JSON) data interchange format," *Internet Engineering Task Force (IETF),* 2014.

Nalaka R. Dissanayake was born in Anuradhapura, 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 and the M.Phil. degree from University of Colombo School of Computing, in 2017.

From 2007 to 2017, he was working as a student instructor, instructor, assistant lecturer, and a software designer in various institutes. He is currently working as a senior lecturer at Sri Lanka Institute of Information Technology, Malabe, Sri Lanka. He is the author of over 30 peer-reviewed conference papers and 3 journal papers. His research interests include software architectures, design patterns, web engineering, and Rich Internet Applications. He has contributed to the domain of web engineering by introducing architectural styles, design pattern, and terms and definitions for some concepts.

Mr. Dissanayake won the first place in the international competition: InnoServe 2016 in Taipei, Taiwan, which he worked as the mentor. Mr. Dissanayake co-authored the publication titled “Annotation based Offload Automation Approach for Cyber Foraging Frameworks”, in the 9th International Research Conference of KDU, Rathmalana, Sri Lanka, September 8-9, 2016, which won the best poster, and also co-authored the publication titled “Towards ICT based Solution for Stuttering”, in the 10th International Research Conference of KDU, Rathmalana, Sri Lanka, August 3-4, 2017, which won the best poster.

G. K. A. Dias received the bachelor of science degree in 1982 from the University of Colombo, Postgraduate 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 includes Computer-aided software engineering, modeling 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.

.