Features for a Style for Push-communication Integrated Rich Web-based Applications

Nalaka R. Dissanayake[1,*], Dharshana Kasthurirathna[2] and Shantha Jayalal[3]

[1]*Department of IT, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka*
[2]*Department of SE, Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka*
[3]*Department of Industrial Management, Faculty of Science, University of Kelaniya, Kelaniya, Sri Lanka*
E-mail: nalakadmnr@gmail.com
[*]*Corresponding Author*

Received 16 April 2022; Accepted 11 April 2023;
Publication 04 July 2023

Abstract

The development aspects of rich web-based applications have evolved; however, abstract concepts, like styles and patterns, are still lacking. If an abstract style for rich web-based applications is available, it can support the whole engineering process in many ways, like assisting in designing aspects and the system's evolution. We have produced an abstract architectural style named RiWAArch style for standard rich web-based applications, and we are working on extending the same to realize integrating push-communication. Push-communication has become a contemporary requirement in developing features like real-time notifications in rich web-based applications. However, the features to be expected from a style to realize the integration of the push-communication are not yet recognized. This concept paper proposes a set of features to be expected from a style for push-communication-integrated

Journal of Web Engineering, Vol. 22_3, 515–542.
doi: 10.13052/jwe1540-9589.2236

rich web-based applications. Our ongoing research will later utilize these features to form requirements and design a comprehensive style by extending the RiWAArch style to realize the abstract features of integrating true push-communication into rich web-based applications.

Keywords: Architectural style, features, push-communication, rich web-based applications.

1 Introduction

The core concepts of rich web-based applications (RiWAs) [1] were introduced in the early 2000s. Since then, RiWAs have evolved over two decades in many ways, especially in the user experience and development aspects. Modern RiWAs benefit from push-communication (PC) to implement functionalities such as notifications, which can further improve the user experience. Even though the development aspects of RiWAs are much evolved, abstract concepts for RiWAs like architectural styles – which can offer many advantages like increased realization and knowledge sharing – are still lacking [2]. Software architectural styles can be considered an abstract description of tried and tested good practices of the generic forms of architecture [3]. Section 2 details these background concepts.

The subsequent sections under the introduction discuss the following aspects within the context of this paper: the problem focused on by this paper, the aim, objectives, and hypotheses, and finally, the methodology. Then the structure of the paper is given.

1.1 Research Problem and Motivation

The problem focused on by this paper is the non-recognition of features to expect from an abstract style which can realize the integration of PC into RiWAs. This problem is further elaborated on below.

As mentioned earlier, despite the evolution of RiWAs' development aspects, they still lack abstract concepts like styles. During this research, it was noted that the development aspects of RiWAs have been much matured to the present day to identify styles and patterns in them. However, abstract styles or patterns focusing on RiWAs' architectural and logical implementations are still lacking [2].

Our ongoing study looks into the architectural formalism of the RiWAs to understand their common characteristics towards introducing abstract

styles and patterns. We have already introduced an architectural style named RiWAArch style, which can realize the abstract features of the standard request-response-based RiWAs [4] (refer to Section 5.3 for the RiWAArch style review). As we continue our study, focusing more on PC-related functionalities like notifications, it was understood that there are no abstract styles for RiWAs, which can realize the integration of true push-communication (TPC – refer to Section 3.3.1) into RiWAs. Our current research work focuses on extending the RiWAArch style to realize the integration of TPC into RiWAs. Nevertheless, to extend the RiWAArch style, it is essential to understand the common characteristics of TPC-enabled RiWAs and identify the features to be expected from a style, which can realize TPC-enabled RiWAs. During the literature survey, it was comprehended that such features are not yet recognized and discussed.

The motivation for recognizing the features to expect from an abstract style to realize PC integration into RiWAs is as follows. Suppose features to be expected from an architectural style to realize the TPC integration with RiWAs can be identified. In that case, they can be utilized to extend the RiWAArch style to realize TPC within RiWAs. The simplicity and visibility provided by the resulting style may immensely assist the architectural properties, like reusability and maintainability [5]. As a result, the rapid development of RiWAs can also be administered. We have highlighted the importance of having an abstract architectural style to realize the PC integration into the RiWAs in other forums [6, 7].

1.2 Research Aim, Hypothesis, and Objectives

This paper aims to identify a set of features to expect from an architectural style and for this style to realize the integration of TPC into the RiWAs. This paper presents and discusses the results of the initial steps of the current phase of our ongoing research, which proposes a set of features to be expected from a style for TPC-enabled RiWAs. These features will be later utilized to extend the RiWAArch style to realize the TCP integration into RiWAs by our ongoing research.

The literature considers that RiWAs are unique, therefore, do not have general characteristics and common features [8]. Non-identification of the general characteristics and features can be why RiWAs lack a coherent and precisely described set of architectural formalism, mainly to realize the TPC integration. The following hypothesis is set to support defining objectives toward achieving the paper's aim, bearing the uniqueness of the RiWAs in mind.

H_0: Common characteristics and essential abstract features of true push-communication integrated RiWAs can be identified – regardless of the development TTs – which can then be used to identify the features expected from an architectural style for push-communication integrated RiWAs.

The following objectives are set to prove this hypothesis in the direction of achieving the paper's aim.

- **OB1**. *Understand PC features in the context of RiWAs*. This objective lays the foundation for OB2. Section 3 discusses the PC features in the direction of achieving this objective.
- **OB2**. *Identify common characteristics and essential abstract features of TPC-integrated RiWAs*. This objective aligns with step 1 of the method specified in Section 1.3.1. The discussions are given in Section 4.1, built on the knowledge of PC features identified in Section 3 while achieving the OB1.
- **OB3**. *Identify the features expected from an abstract style for TPC-integrated RiWAs*. This objective aligns with step 2 of the method specified in Section 1.3.1. The discussions are given in Section 4.2, based on the knowledge gained by achieving OB2. By achieving this objective, the aim of this paper is also achieved.

1.3 Methodology

Our ongoing research is conceptual work, which differs from general empirical software engineering research. The produced knowledge is mainly based on the literature and the experience of the individuals who contributed to our ongoing research, combined with extensive brainstorming by the researchers. However, some formal methods have been utilized to govern the research in different stages. The methods utilized within the scope of this paper are discussed below.

1.3.1 Identification of the features to expect from a style for TPC-enabled RiWAs

Unlike software systems, identifying the requirements for an architectural style is not straightforward. The architectural style is an abstract concept that realizes the general characteristics of similar systems; therefore, identifying these general characteristics of the focused systems can assist in recognizing the features to be expected from a style. Based on this notion, a process was formulated containing three steps to define the requirements to design the

RiWAArch style [4] during the early phases of our ongoing research. This process is shown in Figure 1 below.

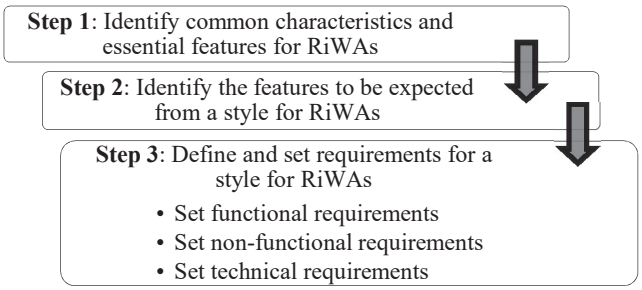

Figure 1 Requirements setting process to design a new style [4].

The exact process is utilized in the current research phase to set the requirements for a style for the TPC-enabled RiWAs. This paper focuses on step 2 of this process, which concentrates on identifying the features expected from a style for TPC-integrated RiWAs. The steps of this process and their employment in this paper are explained below.

Step 1 of this process tries to identify the common characteristics of the targeted systems and their essential features. This step is mainly accomplished based on the knowledge gained through intensive literature surveys and studying the existing technologies and techniques used to develop RiWAs. Also, the experience of the contributed researchers gained by developing different aspects of RiWAs is utilized. The foundation knowledge required for this stage is developed in Sections 2 and 3 while achieving OB1 of this paper. Based on that foundational knowledge, the common characteristics and essential features of TPC-integrated RiWAs are discussed in Section 4.1 towards achieving OB2 of the paper.

In step 2, the features expected from a style are recognized based on the knowledge of characteristics and features for the RiWAs identified in step 1. These features should realize the identified common characteristics and essential features. Section 4.2 proposes the features to expect from a style to realize the TCP integration in RiWAs while achieving OB3 of this paper.

In step 3, functional, non-functional, and technical requirements are derived and set. The ultimate style should deliver the identified features to be expected by satisfying these requirements. By contrast, if these defined requirements are satisfied by a style, it is expected to deliver the identified features. These aspects are not discussed in this paper.

1.3.2 Review of available styles

The review of available styles for RiWAs – in Section 5 – uses a set of criteria to compare and contrast the available similar work towards emphasizing the value of the ongoing research. This set of criteria aligns with the features proposed in Section 4 and is given at the beginning of Section 5.

1.4 Structure of the Paper

Section 2 discusses the core background concepts of the research to form the basis for the rest of the paper, based on the findings of this ongoing research, some of which are already published. Section 3 extends the background to achieve objective 1 of the paper, discussing the concepts of the PC towards understanding the features of the TPC. Section 4 includes the main discussions of this paper – achieving objectives 2 and 3 – which proposes a set of features to be expected from an architectural style for TPC-enabled RiWAs towards understanding the integration of TPC into RiWAs. Section 5 reviews similar work to emphasize the importance of the work of our ongoing research. Finally, Section 6 concludes the paper, also stating the future work and the motivation for it.

2 Background

This section provides the terms, definitions, and related details within the context needed to continue the discussions in the paper. Some of these terms and definitions are the outcomes of the early stages of this ongoing research; in such cases, the early publications of the ongoing research, which contain in detail discussions of these outcomes, are given.

2.1 Software Architecture and Architectural Styles

Software architecture can be considered the foundation of any software system. The support gained from a carefully designed sound architecture is significant throughout all the phases of a software engineering (SE) project [9]. The architecture provides an overall picture of the system and assists in realizing it by depicting the formalism of the architectural elements and their relationships. The increased realization helps the software systems to evolve easily and rapidly by managing the changes [10–12]. Fielding [5] explains that the *configuration* of the following three elements defines the software architecture.

(1) *Components*, which process data.
(2) *Connectors*, which communicate data.
(3) *Data*, which represents the data structures and values.

The *constraints* in the relationships between these elements help to induce the desired set of *architectural properties*.

The architectural style is an abstract concept which can be utilized to design software architectures. The style is defined by Fielding [5] as "a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style." Fielding [5] states that "an architectural style encapsulates important decisions about the architectural elements and emphasizes important constraints on the elements and their relationships." He explains that it is easier to communicate the characteristics of typical constraints by using an architectural style as a method of abstraction rather than as an indicator of personalized design [5]. The improved simplicity and visibility provided by the proper style(s) selected to design a system's architecture can help improve the architectural properties like reusability and maintainability of the software system [5].

There are some valuable styles to support implementing web-based systems. The basic style of web-based systems is the client-server style, also called the two-tier style. As required, the two-tier style can be extended to three-tier or even *n*-tier. Model-view-controller (MVC), service oriented architecture (SOA), and micro-services are some widely used styles in web systems development. Also, there are some styles focusing on the specificity of RiWAs, like jAGA [8], SPIAR [13], and RiWAArch style [4], which are reviewed in Section 5.

2.2 Rich Web-based Applications and Delta-communication

Rich Web-based Application (RiWA) [1] is a type of web-based application [14] that provides a rich user experience – similar to desktop applications – via the combination of rich graphical user interfaces (GUIs) and a faster communication model [15–17], called delta-communication (DC) [18]. Systems like Google apps and Facebook are examples of RiWAs.

The client-side app of a RiWA can be either a browser-based application, a mobile app, or even an embedded system in an internet-of-things (IoT) system [1]. Regardless of the nature of the client-side or server-side elements, the RiWAs can be abstractly seen as a type of web-based system developed based on the client–server architecture, with elements on the client-side

(*client-elements)*, which communicate with the elements in a web server (*server-elements*), for processing data.

Many new technologies and techniques (TTs) have been introduced throughout the last two decades to support the development of different aspects of RiWAs. Some examples are given below.

- Server-side – compatible frameworks for web servers like JAVA, Spring, PHP, and CodeIgnitor.
- Client-side – JavaScript (JS), jQuery, Angular, and Android.
- DC – AJAX [19], Comet [20], and WebSocket [21].

DC TTs are used for faster communication in both pull and push modes. This research concentrates on the push-DC and related aspects utilized to implement PC-enabled features in RiWAs, which are discussed in the following section.

3 Understand the Features of Push-communication

Puch-communication (PC) is not a new concept; it has been a well-used technique since the beginning of human communication, starting from techniques like smoke signalling and pigeon posts and continuing to develop through Morse code and radio broadcasting [22]. The literature shows that PC styles have been used in implementing data networks from the beginning of digital data networks [23–25]. Since the introduction of the web, PC has been implemented by exploiting different technologies to develop solutions like webcasting [26, 27]. We have discussed in depth the fundamental concepts of PC and the evolution of PC towards RiWAs in a different forum [22]. This section briefly discusses the fundamental concepts of PC in the context of RiWAs towards achieving the OB1 of this paper.

3.1 Basics of Push-communication

Generally, HTTP-based communication in the web environment is built on pull communication, where the client requests from the server and pulls data using the request–response model. Unlike pull communication, PC is where the server pushes the data/updates to the client(s) without explicit requests. Since the beginning of the web, different TTs have been exploited to implement PC in web-based systems. RiWAs would benefit from PC to implement features like notification and real-time updates, which have become necessary in modern web-based systems [22, 28].

In general, PC means the server pushing data/content to a client app where the user has not requested the same data/content from the server or any other user/client. Even though the PC and related concepts are discussed in the literature, none of them strongly defines the PC, particularly in the context of RiWAs. Terms and definitions from the literature are given below.

- Techopedia explains *push technology* as "push technology is an internet communication system in which the transaction request is generated by the central web server or publisher" [29]. It further describes that push technology can "push information to a user's desktop instead of waiting for user to make a request" [29].
- The *request for comments* of *Generic Event Delivery Using HTTP Push* [30], which offers the *HTTP WEBPUSH* technology, provides some PC-related terminologies as follows. A push message is "a message sent from an application server to a user agent via a push service", where a push service is "a service that delivers push messages to user agents". A user agent is "a device and software that is the recipient of push messages".
- The working draft of the *push API* [31] defines similar terms: "A push message is data sent to a web application from an application server", where "application server refers to server-side components of a web application".
- Rute and Paulo [32] state that "in a push-based model interested parties get the information if they have previously subscribed it, and information gets distributed when it is available", in their work related to *information-centric networking*, which focuses on IoT-based systems.

These terms and definitions mainly explain PC in general and some other related concepts within the context of their work. They do not attempt to look at PC's implementation or identify its features explicitly. In order to understand the features of the PC towards introducing an architectural style for RiWAs, we need to look into in-depth aspects of the PC. To assist that, having a solid definition of PC is essential. Since there is no firm and explicit definition for PC, this paper proposes to define PC as follows in the context of RiWAs.

Push-communication in rich web-based applications is a communication model where the server-components push data to the client-components. The user of the RiWA is not explicitly triggering any events on the client to send a request to the server-components, requesting the same data pushed by the server-components.

In this setting, the user gets the impression of receiving real-time pushed content from the server, regardless of the actual implementation of the PC. The features related to the implementation of the PC are discussed in the following sections.

3.2 Integrate Push-communication into RiWAs

The concept of PC is not very complex; however, integrating the same into a much more advanced RiWA would introduce a higher level of complexity, which may negatively affect the system in many aspects like maintainability, sustainability, and modifiability. Usually, in a comprehensive RiWA, PC-enabled features might be integrated with other features, which are implemented to use pull-DC and standard HTTP-based request–response communication. When the number of rich features that utilize the PC increases, there is a possibility of exponentially increasing the system's complexity, lowering the maintainability and evolvability.

DC [18] in RiWAs works in pull and push communication modes. There are various TTs to implement DC in RiWAs, where some techniques like AJAX [19] are used for pull-communication, and TTs like Comet and WebSocket [21] are used to implement PC in RiWAs. Some other protocol-level solutions have been recently introduced for implementing PC, namely, server push in HTTP/2 [33] and Webpush protocol [34].

3.3 Features of Push-communication in Rich Web-based Applications

While studying the PC-enabled functionalities in RiWAs, we have identified the features of PC, which are analyzed and briefly discussed below. This discussion achieves OB1 of this paper. The knowledge of the features of PC is required to discuss the features to be expected from a style for PC-enabled RiWAs.

3.3.1 Push-simulation vs. true push-communication

Even though PC generally means the server pushing data to the client(s) – aligning with the proposed definition in Section 3.1 – there are two approaches to implementing PC in RiWAs, which are discussed below.

In the context of DC, early attempts to implement PC in RiWAs only exploit pull-DC TTs to simulate PC based on the request–response model. For example, in polling and long-polling [20], XHR [35] requests (the technology used in AJAX [19]) are automatically sent by the client-component(s) to pull

updates from the server-components and then show them on the GUI. The user gets the impression that the server has pushed the updates since the user has not explicitly sent a request by triggering any event on the GUI. This implementation simulates the PC in RiWAs. This research calls the concept of simulating the PC – built on the request–response model – *push-simulation*. The early attempts to implement PC in RiWAs were based on push-simulation TTs like polling and long-polling [20].

Later, advanced technologies like SSE [24] and WebSocket [25] were introduced to implement PC in RiWAs in the form of DC. Using these advanced TTs makes it possible to implement server-elements to push data to the clients without receiving requests for the same data from the same clients. These TTs also support implementing client-components to capture the data pushed by the server and show them on GUIs. There can be cases where a different client or server may initiate the PC (refer to Section 3.3.3); however, the receiving client is still not explicitly requesting data from the server. This research uses the term *true push-communication* (TPC) to denote the concept of a server pushing data to a client without receiving a request from the same client. The WebSocket is a TPC-implementing technology.

Since the request–response model can realize push-simulation implementation, a style that realizes a request–response-based DC can also realize push-simulation. The RiWAArch style [4] firmly realizes the request–response-based DC; hence, the support for implementing push-simulation can be well expected. This ongoing research focuses on extending the RiWAArch style to realize the integration of TPC based on the features proposed by this paper.

3.3.2 Push-communication styles/modes

There are multiple architectural styles available to explain the formalisms of PC, which are unicasting, multicasting, broadcasting, and publisher–subscriber styles [22, 36, 37]. However, realizing these styles through a single comprehensive style for PC-integrated RiWAs is yet to be discussed and initiated by this paper. In this paper, these styles are referred to as the *push-modes*. A style for PC-integrated RiWAs is expected to realize these push-modes providing robust support to implement PC in RiWAs using these modes. The server must contain the necessary elements to understand the push-mode and push the content to the target client(s). The support for push-modes can improve a style's adoptability and evolvability since such a style can assist in implementing PC functionalities using the required push-mode(s).

3.3.3 Server-event-driven

In the literature, even though the term event is used in the context of PC, it is not firmly defined. The term event is primarily used in PC-related literature to denote a set of data – associated with the events triggered in the server – to be pushed to the client by the server, like in the technology named *Server-Sent Events* [38]. Nevertheless, in this research, the term event explicitly indicates the events triggered in the server, similar to *event-driven programming*.

The PC-related literature [36, 37] uses the concepts *aperiodic* and *periodic* to explain the event-driven aspects of the PC. Aperiodic PC is explained as an event-driven PC, where the PC is performed as the events – such as user actions or data updates – are triggered. Periodic PC is described as the PC performed according to a pre-arranged schedule established for the information transfer operations to follow. Technically, a schedule also triggers events to initialize some process; therefore, both aperiodic and periodic concepts can be seen as event-based mechanisms.

The position of this paper on the server-event-driven nature of the PC is as follows. Execution of logic in a server – to process either a user request or respond to a scheduler – which makes the server initiate PC, can be considered triggering events in the server to perform PC. In this paper, the events triggered in a server to perform PC are called *push-events*. Some use cases that trigger push-events in a server are given below as examples.

- When a user sends a chat message to the server, it triggers a push-event in the server to push the message to the target user. The sending user's action triggers a push-event in the server, asking the server to push the data to the receiving user on behalf of the sending user.
- A user inserts a new item into a database, and the server pushes the new item's data to the users who are viewing the items. In this case, the insert item logic triggers a push-event for the viewing users to receive real-time updates.
- When an author publishes an article, a notification is sent to the readers who have subscribed to the author. Here, the notification is pushed to the readers due to the author's publishing process, which triggers a push-event to push the notification. This case is an example of a PC feature, based on the publisher–subscriber mode.
- A schedule, which triggers a push-event to count the number of users logged in within the last hour and push the count to a dashboard of the admin users of the system. This is an example of a periodic PC.
- When a user signs into a system, notifications of the earlier events – which had been triggered when the user was not online – can be pushed

to the client. These notifications are pushed in non-real-time as the user's sign-in process triggers a push-event. This is an example of non-real-time PC.

- An embedded system with a sensor to identify the temperature changes sends the data to the server to push the data to the users in real-time. The embedded system's request to send data to the server triggers a push-event, which means the logic of the request handler in the server has to trigger the push-event. This case is an example of implementing PC in IoT-based systems.

Looking at these examples, we can understand that a push-event should be triggered in the server to initiate the TPC from server to client. These push-events can be triggered by the execution of different types of logic in the server, and the server is the true sender of the PC. Based on this notion, this research uses the term *push-event*, defined as below.

> A push-event is any event that triggers the server to push content to the client(s). It can be understood that the push-events can be triggered as a part of some other processes. When a push-event is triggered, the server takes the necessary actions to push the appropriate content to the relevant client(s).

It should be noted that the TPC is always server-event-driven, and the server implements the event handlers to perform the TPC.

3.3.4 Real-time vs. non-real-time push-communication

The PC can perform in either real-time or non-real-time. If the receiver of the PC is available when the push-event is triggered, the server can push the content to the receiver in real-time. Nevertheless, the server cannot push the content to the receiver in real-time if the receiving user is unavailable online when the push-event is triggered. In such a case, there should be a mechanism for the server to save the content and push it later when the user is available again. For example, when a user sends a chat message to another user if the receiving user is not online, the message should be saved and delivered when the receiving user connects to the system later.

The PC, performed when the push-event is triggered, is considered a *real-time PC*. The real-time PC allows the users to receive content when a push-event is triggered. In that case, the received content is directly related to the push-event triggered in the server, which initiated the TPC. The PC, which happens later, after the original push-event is triggered, can be considered a

non-real-time PC. The server must keep the push-content saved to be pushed later when the target user is available again. Since the PC is server-event-driven, there is supposed to be a push-event to initiate the non-real-time PC when the client connects to the server again. The client's first request to connect to the server can be exploited to trigger a push-event for the server to look for any saved push-content for the client and push them in non-real-time. In this case, the pushed content is not directly related to the latest push-event since the latest push-event is not the original push-event, which wanted to push the content.

4 Features to Expect from a Push-communication Integrated Style for Rich Web-based Applications

This section first discusses the common characteristics and essential features identified in PC-enabled RiWAs based on the features of PC discussed in Section 3.3. Based on that knowledge, this section proposes a set of features to be expected from a style for PC-enabled RiWAs, following the method stated in Section 1.3.

Our ongoing research has already produced a style named RiWAArch style [4], which can realize the standard pull communication-based RiWAs. Since we have already discussed the features of the RiWAArch style in a different paper [4], they are not discussed again in this paper. The features proposed in this section only focus on extending the RiWAArch style to realize integrating PC to standard pull communication-based RiWAs.

4.1 Step 1: Identify Common Characteristics and Essential Features for Push-communication Enabled RiWAs

This section fulfils OB2 of this paper, aligning with step 1 of the requirements-setting process given in Section 1.3.1.

We have previously identified and discussed the common characteristics and essential features of standard general pull-communication-based RiWAs. Common characteristics are client-side event handling and split business logic between client and server, and the essential features are effective modularization and proper DC handling and management [4]. Since we have already discussed them in different forums [4, 39], this paper does not detail them. This section only discusses the characteristics and features related to the PC-based functionalities in RiWAs, identified by studying PC-enabled RiWAs, in the direction of extending the RiWAArch style.

4.1.1 Common characteristic 1 – push-simulation vs. true push-communication

The push-communication is implemented in RiWAs using push-simulation or TPC TTs (refer to Section 3.3.1). It was noted that RiWAs mostly use push-simulation TTs, which are easier to develop PC-enabled features since push-simulation is based on the traditional request–response model.

The TPC implementation TTs require additional elements and settings, which make the development of RiWAs much more complex. For example, when WebSocket [21] is used, a WebSocket server is needed to be implemented, and for the server components to communicate with the WebSocket server – to push content to the clients – some other TTs like RowSocket should be used [40]. However, WebSocket improves the scalability and performance of RiWAs [41], and as a result, the user experience can also be increased. This research focuses on the RiWAs, which use advanced TPC TTs like WebSocket to implement PC-enabled features toward higher scalability and performance.

4.1.2 Common characteristic 2 – push-events in server

When using the PC, the push-events are triggered in the server to initiate the TPC from the server to the client(s) (refer to Section 3.3.3). This behavior is typical in TPC-enabled RiWAs, regardless of the type of the event or the push-initiator. The TPC-integrated RiWAs should include proper logic in the server to process these push-events. Section 4.2.3 discusses this aspect further.

4.1.3 Common characteristic 3 – real-time vs. non-real-time

According to the recipient's availability, the server components of RiWAs may require pushing content to the client(s) either in real-time or non-real-time. This setting has been further discussed in Section 3.3.4, providing an example. The RiWAs are generally supposed to handle both real-time and non-real-time PC. Non-real-time PC requires using a database to save the content to push later; hence, additional support from the database design and development activities is necessary. Elements that handle real-time/non-real-time TPC may need implementing logic to save and read push content into and from the database as required.

4.1.4 Essential feature 1 – PC mode support

Different PC-enabled functionalities in a RiWA may be implemented using different PC modes (refer to Section 3.3.2 for PC modes). For example, a chat application may allow users to chat with a single person based on unicast, chat

with all the members in a group based on broadcast, or even chat with some selected members in the group based on multicast. There should be necessary elements to support implementing functionalities using different PC modes in RiWAs. Moreover, a RiWA should generally include elements to support all the PC modes.

4.1.5 Essential feature 2 – push-DC connectors

The PC in RiWAs is supposed to be implemented in the form of DC. Since push-DC is a type of communication, the PC-enabled RiWAs need dedicated connectors in the client and server to communicate in the required push-mode(s) using true push-DC. The server can then push content to the client(s) via a server-connector, and the client(s) can capture the push-content through a client-connector. To handle true push-DC, it is essential to have dedicated connectors in both the server and the client. True push-DC development TTs like WebSocket and supporting tools like libraries can be utilized to develop these dedicated connectors.

4.1.6 Essential feature 3 – decision-making components for PC

This feature is related to support implementing real-time/non-real-time PC and the push-modes upon triggering the push-events in the server. The server required dedicated components to implement PC-related logic in PC-enabled RiWAs. These components would decide aspects such as when the push-events should be triggered, what push-mode to use, whether to execute in real-time/non-real-time, and what data and structures to be pushed. These components in the server can utilize the push connectors to push the content to the client(s) based on PC-related logic. This feature is further discussed in Section 4.2.3.

4.2 Step 2: Identify the Features to be Expected from a Style for Push-communication Enabled RiWAs

This section proposes a set of features to be expected from a style for PC-enabled RiWAs, in the direction of fulfilling OB3 of this paper while achieving the aim of the paper. This discussion aligns with step 2 of the requirements-setting process given in Section 1.3.1.

The crux of the common characteristics and essential features discussed in the previous section is extracted and analyzed to formulate these features. The rationale behind the notion of this discussion is that if a style exhibits these features, it will support implementing the aforementioned common characteristics and essential features in PC-enabled RiWAs.

4.2.1 Feature 1 – true push-communication and push-communication modes

A style for PC-enabled RiWAs should realize TPC from the server to the client (refer to Section 3.3.1 for TPC) and all the PC modes (refer to Section 3.3.2 for push-modes) based on TPC. The style will then support implementing the TPC-enabled features using the required mode(s). This feature covers the common characteristic 1 and essential feature 1 for PC-enabled RiWAs.

Since the ongoing research focuses on extending the RiWAArch style to realize TPC, it is considered that the push-simulation is already realized by the RiWAArch style, as the style firmly realizes pull-DC.

4.2.2 Feature 2 – push-DC handling in server and client

There should be a comprehensive dedicated set of connectors to process push-DC on the server and client sides of RiWAs. The inclusion of a complete set of push-DC connectors to a style would enable the style to deliver essential feature 2 of the PC-enabled RiWAs (refer to Section 4.1.5). Dedicated push-DC connectors will improve the simplicity and visibility of a style, which can increase the system's evolvability. Hence, rapidly evolving RiWAs would benefit from a set of push-DC connectors which can perform TPC.

4.2.3 Feature 3 – PC related decision making

A component is needed in the server to implement logic to take PC-related decisions. These decisions are associated with PC-related aspects like push-mode, mentioned under essential feature 3 in Section 4.1.6.

For example, when an author publishes an article, consider that a notification must be sent to the readers who have subscribed to the author. The notification may contain the title of the article and the author's name. If some subscribed readers are offline, the notification should be delivered when they are online again. Based on these requirements, it can be understood that a component should contain the logic to decide who should be notified (based on the push-mode), the content of the notification, and other related matters like real-time/non-real-time notification. It should be noted that the push-event is triggered in the server as a part of the author's publishing process, and the author is not explicitly notifying the readers. Hence, the component(s) dedicated to making PC-related decisions are supposed to trigger the push-events in the server to perform TPC.

Based on PC-related case studies, similar to the above example, the following aspects are identified to be taken care of by a dedicated

decision-making component in the server.

- Trigger push-events in the server based on the requirements of other processors.
- Decide the content and the structures to be pushed.
- Decide to whom the content should be pushed based on the push-mode.
- Handle real-time vs. non-real-time push.
- Saving pushed content and related meta-data for persistence as required.

Suppose a style for PC-enabled RiWAs supports implementing a dedicated PC-related decision-making element(s), in that case, it will support developing common characteristics 2 and 3 and implementing essential feature 1 based on essential feature 3.

4.2.4 Feature 4 – support the standard general functionalities of RiWAs

A style for PC-enabled RiWAs should realize not only the common characteristics and essential features identified in the context of TPC-enabled RiWAs but also the general characteristics and features of the standard RiWAs with pull-DC [4]. It is required for a style for RiWAs to realize both pull and push formalisms entirely since the PC-based features also benefit from pull-DC and non-DC. For example, the chat feature page would be loaded to the browser upon a standard HTTP request in the chat application scenario. When a user sends a chat message to the server, it can be implemented using pull-DC. Then the chat message is pushed to the target recipient(s) using true push-DC.

The features of the RiWAArch style [4] realize the general characteristics and features of the standard RiWAs based on pull-communication. This ongoing research tries to extend the RiWAArch style to deliver the TCP-related features discussed in this section. The resulting style is expected to realize the RiWAs completely, including non-DC, pull-DC, and true push-DC.

5 Review of Available Styles for Push-communication Enabled RiWAs

This section reviews the available architectural solutions that try to realize the RiWAs. The review is based on the following criteria, which align with the features proposed in Section 4.2. The architectural properties expected to be satisfied by each criterion are indicated.

1. Specificity for RiWAs, therefore, realizes the common characteristics of RiWAs. This research appreciates the abstraction of a solution by

not depending on TTs. The abstraction improves the adoptability and portability regardless of the development TTs.
2. Support for TPC. This criterion ensures scalability and performance.
3. Ability to realize PC modes. If the solution can realize many PC modes, the adoptability of the solution is high.
4. Centralized push-DC handling in both client and server, using dedicated connectors. This criterion safeguards simplicity and visibility in the direction of increased evolvability.
5. Support push-DC integration with standard HTTP and pull-DC, improving performance, simplicity, and visibility.

5.1 jQuery-based Ajax General Interactive Architecture (jAGA) [8]

The research by Li and Peng introduces an architecture for AJAX-based RiWAs, named jAGA. This solution focuses on the browser-based RiWAs, and thus support for other types of RiWAs – like mobile apps – cannot be guaranteed. As the name implies, this solution is based on jQuery and AJAX; hence, not abstract. Since jAGA only targets browser-based RiWAs – limiting the scope of the platform – and based on specific TTs, the portability and adoptability are low.

Since this solution is based on AJAX, it supports only pull-based DC. Therefore, support for TPC cannot be expected. However, push-simulation techniques like XHR [35] based on polling or Comet can still be realized by jAGA. Even though Li and Peng state that "the utility of jAGA can effectively reduce the work of code, cut down the difficulties of AJAX realization, make the coding process simple and hierarchical and decrease the chance of mistakes. What's more, due to its reasonable design, jAGA owns nice flexibility and scalable performance", since TPC is not realized, higher performance and scalability cannot be expected from jAGA when implementing PC-related features.

As PC is out of the scope of this research by Li and Peng, it does not contain elements for push-DC handling and does not discuss how the PC modes are realized.

5.2 A Component- and Push-based Architectural Style for AJAX Applications [42]

In this research, Mesbah and Deursen introduce an architectural style named Single Page Internet Application aRchitectural style (SPIAR style)

for single-paged browser-based RiWAs, which focuses more on the front-end implementation; thus, SPIAR has a narrow scope. SPIAR style tries to "minimize user-perceived latency and network usage, and improve data coherence and ultimately user experience" [42]. The SPIAR style is based on the characteristics of some available frameworks: Echo2, GWT (a web framework offered by Google), Backbase (a commercial package delivered by Backbase), cometd (a server-side push framework), and Dojo (a client-side framework to work with cometd). The narrow scope and the TTs dependencies of the SPIAR style make it not an abstract solution, limiting the adoptability and portability.

As the paper's name denotes, the context is based on AJAX applications. It means that the SPIAR targets the systems which use XHR and request-response model-based polling and Comet to implement PC. It confirms that the SPIAR does not support TPC, and the scalability is lower than the TCP.

The SPIAR style does not explicitly realize the PC modes. By design, the clients who need server updates have to subscribe to the server, and the server updates are broadcasted to the subscribed clients. The clients subscribe to the *push server* via an element named *push client*, and the *push server* pushes the updates to the *push client* element, where the *push client* and *push server* elements create a connector pair for push-DC. In the case of pull-DC, the requests from the element named *AJAX Engine*, which is on the client-side, are handled by an element named *Decoder* on the server-side, and the responses are sent back to the *AJAX engine* by an element named *Encoder*. *Ajax Engine and the Encoder and Decoder create* a set of connectors for pull-DC. These connectors for pull and push DC enormously improve the visibility and simplicity of the SPIAR style. However, the SPIAR style does not depict how non-DC communication is integrated. Since the SPIAR style targets single-page applications, standard HTTP communication is eliminated after the first time the application loads its elements; therefore, it does not require non-DC for page navigation once the application is fully loaded to the browser. In this setting, SPIAR does not need to realize non-DC integration as a style, and within its context, the SPIAR realizes all the required types of communication well with intense simplicity and visibility.

5.3 RiWAArch Style [4]

RiWAArch style is an abstract style that realizes the common characteristics and essential features of the general pull-communication-based RiWAs. Since

it does not depend on specific TTs, the adoptability and portability are higher than the other solutions.

Since the RiWAArch style realized the pull-DC, it only realizes the push-simulation and does not realize TCP. Therefore, higher scalability and performance similar to when using WebSocket cannot be expected. However, it may realize the push-modes to implement with push-simulation TTs, which should be further experimented.

The RiWAArch style has a comprehensive connector pair to implement pull-DC, which improves simplicity and visibility. However, when implementing push-simulation, the same connectors should be exploited, and it may lower the simplicity and visibility compared to the RiWAs without PC.

When using the RiWAArch style, it can realize the integration of push-simulation with non-DC and pull-DC. Still, as explained above, the simplicity, visibility, and performance will not be at their highest compared to a hypothetical style which can realize TCP.

6 Conclusion, Future Work, and Motivation

Using a requirements setting process to design a new software architectural style (refer to Figure 1 in Section 1.3.1), this paper identifies the features of PC in RiWAs (refer to Section 3.3) and then identifies common characteristics and essential features for PC-enabled RiWAs (refer to Section 4.1). Based on those identified common characteristics and essential features, the paper suggests features to be expected from a style for PC-enabled RiWAs while emphasizing TCP (refer to Section 4.2).

RiWAArch style is a comprehensive style which realizes the standard RiWAs (reviewed in Section 5.3) [4]. Our ongoing research focuses on extending the RiWAArch style to realize the integration of TPC into RiWAs, by delivering the features proposed in this paper. After that, we expect to discuss adopting the new style into development by demonstrating the development of a comprehensive set of PC-related functionalities through a use case. This case study will be designed to discuss every aspect of the introduced style, which extends the RiWAArch style. At the end of our ongoing research, the introduced style will be evaluated with the help of domain experts to validate the practicality and adoptability of the style.

Introducing a comprehensive abstract architectural style is advantageous for PC-enabled RiWAs engineering. Architectural styles offer a framework for designing system architectures [43], and styles can be considered an abstract description of tried and tested good practices of the generic forms

of architecture [3]. In other words, styles provide a way to capture the knowledge from successful software solutions in the past, and we can expect the styles will also be successful in similar systems [43]. Since a style also can provide an overall abstract picture of a system – similar to software architecture – it can assist in realizing the system, which helps in reducing complexity [44], where the complexity in systems can be reduced by increasing the understanding and improving the realization of the elements and their configuration [45]. Moreover, the architecture acts as a bridge from requirements to development; hence, the realization provided by an architectural style will be able to assist the development activities directly and also enhance some aspects like the reuse of code, the evolution of the system, and management of the project [46]. Considering all these facts, we can conclude that having an architectural style, which can realize a specific set of constraints in targeted systems is advantageous in software engineering, which is also true for RiWAs engineering.

RiWAs are already marked as the de facto standard in the web-based environment due to the higher user experience they provide [1]. Regular RiWAs may use only pull-DC to communicate with the server; however, with trending features such as push notifications and real-time updates [22, 28], RiWAs can benefit from push-DC. Hence, PC can be seen as a contemporary requirement for RiWAs. In this setting, the design and development activities of PC-enabled RiWAs may require dedicated software engineering methods. Considering the advantages mentioned above of architectural styles, we can conclude that the PC-enabled RiWAs would benefit from a robust and dedicated style, which can realize the integration of TPC into RiWAs, hence, assist the rapid evolution of RiWAs via ample simplicity and visibility provided by the style. We think that the features proposed in this paper (refer to Section 4.2) to be expected from a style for PC-integrated RiWAs would immensely assist in setting requirements to extend the RiWAArch style in our ongoing research. The resulting style will be able to realize implementing TCP using advanced TTs like WebSocket.

References

[1] N. R. Dissanayake and K. Dias, "Rich web-based applications: An umbrella term with a definition and taxonomies for development techniques and technologies," *International Journal of Future Computer and Communication*, vol. 7, no. 1, pp. 14–20, 2018.

[2] N. R. Dissanayake and G. K. A. Dias, "Abstract concepts: A contemporary requirement for rich internet applications engineering," in *9th International Research Conference of KDU (KDU-IRC 9)*, *Colombo, Sri Lanka, 2016.*

[3] I. Sommerville, *Software Engineering*. India: Dorling Kindersley, 2011.

[4] N. R. Dissanayake and K. Dias, "RiWAArch Style: An architectural style for rich web-based applications," in *Proceedings of the 2020 Future Technologies Conference (FTC), Canada*, 2020.

[5] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*. Irvine: University of California, 2000.

[6] N. R. Dissanayake, D. Kasthurirathna, S. Jayalal, "Towards a style for push-communication enabled rich web-based applications [presentation]," in *Colombo SIGCHI Research Showcase, Colombo, Sri Lanka*, 2021.

[7] N. R. Dissanayake, D. Kasthurirathna, S. Jayalal, "Towards an abstract style for true-push-communication enabled rich web-based applications [extended abstract]," in *21st International Conference on Advances in ICT for Emerging Regions (ICTer)*, *Colombo, Sri Lanka*, 2021.

[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] Architech Solutions, *The Importance of Software Architecture*, Toronto, Ontario: Architech Solutions, 2014.

[10] P. Oreizy, N. Medvidovic, R. N. Taylor, *Architecture-Based Runtime Software Evolution*. Irvine: University of California 1998.

[11] J. Gustafsson, J. Paakki, L. Nenonen, I. Verkamo, "Architecture-centric software evolution by software metrics and design patterns," in *Proceedings of the Sixth European Conference on Software Maintenance and Reengineering (CSMRí02)*, 2002.

[12] F. Cuadrado, B. García, J. C. Dueñas, H. A. Parada, "A case study on software evolution towards service-oriented architecture," in *22nd International Conference on Advanced Information Networking and Applications – Workshops*, 2008.

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

[14] N. R. Dissanayake and G. Dias, "Web-based applications: Extending the general perspective of the service of web," in *10th International Research Conference of KDU (KDU-IRC 2017) on Changing Dynamics*

in the Global Environment: Challenges and Opportunities, Rathmalana, Sri Lanka, 2017.
[15] M. Busch and N. Koch, *Rich Internet Applications – State-of-the-Art*. Munchen: Ludwig-Maximilians-Universitat, 2009.
[16] G. Lawton, "New ways to build rich internet applications," *Computer*, vol. 41, no. 8, pp. 10–12, Aug 2008.
[17] N. Koch, M. Pigerl, G. Zhang, T. Morozova, "Patterns for the model-based development of RIAs," in *Springer ICWE, Heidelberg*, 2009.
[18] 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.
[19] J. J. Garrett, "Ajax: A New Approach to Web Applications," 18 February 2005. [Online]. Available: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications.
[20] M. Carbou, *Reverse Ajax, Part 1: Introduction to Comet*. IBM, 2011.
[21] I. Fette, Google Inc, A. Melnikov, Isode Ltd., *The WebSocket Protocol*. Internet Engineering Task Force, 2011.
[22] N. R. Dissanayake, D. Kashthurirathna, S. Jayalal, "Evolution of push-communication towards the rich web-based applications," in *Proceedings of FTC 2020, Canada*, 2020.
[23] R. M. Metcalfe and D. R. Boggs, E*thernet: Distributed Packet Switching for Local Computer Networks.* California: Xerox Palo Alto Research Center, 1975.
[24] J.-M. Chang and N. F. Maxemchuk, "Reliable broadcast protocols," *ACM Transactions on Computer Systems*, vol. 2, no. 3, pp. 251–273, 1984.
[25] S. Technologies, *A Survey of the History of Internet Multicast*. Stardust.com, Inc., 1999.
[26] S. Ramakrishnan and V. Dayal, "The PointCast network," in *Proceedings of the 1998 ACM SIGMOD international Conference on Management of Data*, 1998.
[27] cnet, "PointCast unveils free news service," cnet, 13 02 1996. [Online]. Available: https://www.cnet.com/news/pointcast-unveils-free-news-service/. [Accessed 26 11 2018].
[28] M. Pielot and L. Rello, "Productive, anxious, lonely – 24 hours without push notifications," in *MobileHCI '17, Vienna, Austria*, 2017.
[29] Techopedia, *Push Technology*. Techopedia Inc., 2012.

[30] M. Thomson, Mozilla, E. Damaggio, E. B. Raymor, Microsoft, "WEB-PUSH – Generic Event Delivery Using HTTP Push draft-ietf-webpush-protocol-12," Internet-Draft, 2016.

[31] W3C, "Push API – W3C Working Draft 04 February 2020," W3C, 2020.

[32] R. C. Sofia and P. M. Mendes, "An overview on push-based communication models for information-centric networking," *Future Internet*, vol. 11, no. 3, Mar 2019.

[33] M. Belshe, Bitgo, R. Peon, I. Google, E. M. Thomson, Mozilla, *Hypertext Transfer Protocol Version 2 (HTTP/2)*. Internet Engineering Task Force (IETF), 2015.

[34] M. Thomson, Mozilla, E. Damaggio, E. B. Raymor, and Microsoft, "Generic Event Delivery Using HTTP Push draft-ietf-webpush-protocol-12," 2016.

[35] Web Hypertext Application Technology Working Group (WHATWG), "XMLHttpRequest Living Standard," 19 October 2015. [Online]. Available: https://xhr.spec.whatwg.org/. [Accessed 03 November 2015].

[36] M. Franklin and S. Zdonik, ""Data in your face": Push technology in perspective," in *SIGMOD '98 Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA*, 1998.

[37] J. T.-S. Quah and G. L. Lim, "Push selling – multicast messages to wireless devices based on publish/subscribe model," *Electronic Commerce Research and Applications*, vol. 1, no. 3–4, pp. 235–246, 2002.

[38] I. Hickson, "Server-Sent Events," 3 Feruary 2015. [Online]. Available: http://www.w3.org/TR/eventsource/. [Accessed 15 May 2015].

[39] 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.

[40] "Ratchet," Ratchet, 2020. [Online]. Available: http://socketo.me/. [Accessed 15 May 2020].

[41] N. R. Dissanayake and G. Dias, "A comparison of delta-communication technologies and techniques," in *10th International Research Conference of KDU (KDU-IRC 2017) on Changing Dynamics in the Global Environment: Challenges and Opportunities*, *Rathmalana, Sri Lanka*, 2017.

[42] A. Mesbah and A. v. Deursen, "A component- and push-based architectural style for AJAX applications," *The Journal of Systems and Software*, vol. 81, pp. 2194–2209, 2008.

[43] D. M. Selfa, M. Carrillo, 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.
[44] D. Hough, "Rapid Delivery: An eveolutionary approach for application development," *IBM System Journal*, vol. 32, no. 3, pp. 397–419, 1993.
[45] H. Zuse, *Software Complexity Measures and Models*. New York: de Gruyter & Co., 1992.
[46] M. H. Valipour, B. Amirzafari, K. N. Maleki, N. Daneshpour, "A brief survey of software architecture concepts and service oriented architecture," in *IEEE, Beijing*, 2009.

Biographies

Nalaka R. Dissanayake received a B.Sc. degree in information technology from the Sri Lanka Institute of Information Technology in 2007 and an M.Phil. degree from the University of Colombo School of Computing in 2017. He is currently reading for a Ph.D. at the Sri Lanka Institute of Information Technology, Sri Lanka.

From 2007 to 2023, he worked as a student instructor, instructor, assistant lecturer, software designer, and senior lecturer in various institutes. He has authored over 30 peer-reviewed conference papers and 3 journal papers. His research interests include software architecture, design patterns, web engineering, and rich internet applications. He has contributed to the domain of web engineering by introducing architectural styles, design patterns, and terms and definitions for some concepts related to rich web-based applications. He has also served as a reviewer of some conferences in Sri Lanka.

Dharshana Kasthurirathna graduated from the Department of Computer Science & Engineering, University of Moratuwa, in 2004. He has over 7 years of experience in the ICT industry as a software engineer and a research engineer. He obtained his master's degree in computing from the University of Colombo, School of Computing, in 2011 and his Ph.D. in Complex Systems from the Faculty of Engineering & IT, University of Sydney, in 2016. His research interests include complex systems, network science, computational game theory, machine learning and distributed computing.

Shantha Jayalal received his Ph.D. in Computer Science from Keele University in the United Kingdom in 2006. He holds a Postgraduate Diploma in Computer Science from the University of Colombo and a bachelor's degree in industrial management from the University of Kelaniya, Sri Lanka. Currently, he is a Senior Lecturer in the Department of Industrial Management, Faculty of Science of the University of Kelaniya, Sri Lanka. His research interests are in web engineering, semantic web, data science and machine learning.