AN INTEGRATED THREE-FLOW APPROACH FOR FRONT-END SERVICE COMPOSITION

LIM MEI TING

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2020

AN INTEGRATED THREE-FLOW APPROACH FOR FRONT-END SERVICE COMPOSITION

LIM MEI TING

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SOFTWARE ENGINEERING (SOFTWARE TECHNOLOGY)

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2020

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Lim Mei Ting

Matric No: 17005835/1

Name of Degree: Master of Software Engineering (Software Technology)

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

An Integrated Three-Flow Approach for Front-End Service Composition

Field of Study: Computer Programming

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright of this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date: 28 December 2020

Subscribed and solemnly declared before,

Witness's Signature Date: 28 December 2020

Name:

Designation:

AN INTEGRATED THREE-FLOW APPROACH FOR FRONT-END SERVICE COMPOSITION

ABSTRACT

End-User Service Composition (EUSC) aims to enable end-user programmers who are not professional developers, develop applications by composing or aggregating existing web services. Despite the effort, studies have shown that end-user programmers are not able to deal with the technical complexities involved in EUSC. One way to deal with this issue is Front-End Service Composition (FESC), which allows end-user programmers to compose web services at the presentation layer of an application by configuring User Interface (UI) widgets that represent the back-end web services. However, apart from there not being enough studies on FESC, end-user programmers also experience a number of conceptual and usability issues in service composition. Following that, this research proposes an integrated three-flow approach namely application flow, control flow and data flow, to help deal with the current limitations of FESC. The approach generates the Graphical User Interface (GUI) of web services automatically, thus allowing the UI of the application to be developed at the same time the required web services are assembled. The approach allows end-user programmers to explicitly configure the three different types of flows involved in service composition. A proof-of-concept prototype, QuickWSC, that incorporates the three-flow approach was developed. It adopts a side-by-side multiple-view design to support visual configuration of the three flows in an uncluttered yet synchronized manner that adheres to established design guidelines. A user evaluation study which comprised the think-aloud protocol, observation and survey was conducted for data collection purpose where end-user programmers were recruited to evaluate QuickWSC. During the user evaluation study, the end-user programmer was given a brief introduction about the research. Thereafter, a predefined scenario was given to the end-user programmer for a web service composition task. The composition process was recorded on video for data analysis purposes. Framework analysis approach and descriptive statistics were used for qualitative and quantitative data analysis respectively. The results achieved was decently positive. Triangulation was performed during discussion over the results by using the qualitative and quantitative analysed data, and providing a more comprehensive finding of the prototype usability and its features. The evaluation results show that QuickWSC has a high level of usability and it is easy to compose web services by explicitly specifying the three flows, the three-flow configurations integrated into the two views helps in composing application from web services, and that no technical knowledge is required to use QuickWSC. This research has successfully implemented the prototype based on the proposed approach to address a number of conceptual and usability issues in service composition faced by the end-user.

Keywords: end-user programmer, front-end service composition

SATU PENDEKATAN TIGA ALIRAN BERSEPADU UNTUK KOMPOSISI PERKHIDMATAN MELALUI BAHAGIAN HADAPAN

ABSTRAK

Komposisi Perkhidmatan Pengguna Akhir (EUSC) bertujuan untuk membolehkan pengaturcara pengguna akhir yang bukan pembangun profesional, membangunkan pelbagai aplikasi dengan mengkomposisi atau menggabungkan perkhidmatan laman web yang sedia ada. Walaupun wujudnya usaha sebegitu, kajian menunjukkan bahawa pengaturcara pengguna akhir tidak dapat menangani kerumitan teknikal yang terlibat di dalam EUSC. Salah satu cara untuk menangani masalah ini ialah komposisi perkhidmatan bahagian hadapan (FESC), yang membolehkan pengaturcara pengguna akhir mengkomposisi perkhidmatan web pada lapisan persembahan sesuatu aplikasi dengan mengkonfigurasi '*widget'* antara muka pengguna yang mewakili bahagian belakang perkhidmatan web '*back-end*'. Walau bagaimanapun, kajian mengenai FESC ini tidak seberapa, dan ada juga pengaturcara pengguna akhir yang mengalami masalah konsep dan kebolehgunaan komposisi perkhidmatan. Berikutan itu, penyelidikan ini mencadangkan satu pendekatan tiga aliran bersepadu (aliran aplikasi, aliran kawalan dan aliran data) untuk menangani batasan semasa FESC. Pendekatan ini menghasilkan antara muka grafik perkhidmatan web secara automatik, selanjutnya membolehkan antara muka aplikasi dibangunkan pada masa yang sama perkhidmatan web yang diperlukan digabungkan. Pendekatan ini membolehkan pengaturcara pengguna akhir mengkonfigurasi ketiga-tiga jenis aliran berlainan yang terlibat di dalam komposisi perkhidmatan secara eksplisit. Satu '*proof-of-concept'* prototaip, QuickWSC, yang menerapkan pendekatan tiga aliran telah dibangunkan. Ia menggunakan satu reka bentuk pandangan-pelbagai sebelah-menyebelah untuk menyokong konfigurasi visual ketiga-tiga aliran tersebut dengan satu cara yang tidak bersepah tapi selaras yang mematuhi garis panduan reka bentuk yang tersedia ada. Satu kajian penilaian pengguna yang terdiri daripada protokol *think-aloud*, pemerhatian dan tinjauan telah dijalankan di mana pengaturcara-pengaturcara pengguna akhir telah direkrut untuk menilai QuickWSC. Semasa kajian penilaian pengguna, pengaturcara pengguna akhir diberi pengenalan ringkas mengenai penyelidikan tersebut. Selepas itu, senario yang telah ditetapkan diberikan kepada pengaturcara pengguna akhir untuk mengkomposisi perkhidmatan web. Proses mengkomposisi perkhidmatan web dirakam sebagai video untuk tujuan analisis data. Pendekatan analisis kerangka dan statistik deskriptif digunakan untuk menganalisis data kualitatif dan kuantitatif. Analisis data menunjukkan keputusan yang positif. Triangulasi dilakukan semasa perbincangan hasil dengan menggunakan data yang dianalisis secara kualitatif dan kuantitatif untuk memberikan penemuan yang lebih komprehensif mengenai kebolehgunaan dan ciri-ciri prototaip. Hasil penilaian menunjukkan bahawa QuickWSC mempunyai satu tahap kebolehgunaan yang tinggi dan adalah mudah untuk mengkomposisi perkhidmatan web dengan menspesifikasikan tiga aliran tersebut secara eksplisit, konfigurasi tiga aliran yang disatukan di dalam dua pandangan membantu di dalam mengkomposisi aplikasi daripada perkhidmatan web, dan pengetahuan teknikal tidak diperlukan untuk mengguna QuickWSC. Penyelidikan ini berjaya membangunkan prototaip berdasarkan pendekatan yang dicadangkan untuk mengatasi sejumlah masalah konseptual dan kebolehgunaan komposisi perkhidmatan yang dihadapi oleh pengguna akhir.

Kata Kunci: pengaturcara pengguna akhir, komposisi perkhidmatan bahagian hadapan

ACKNOWLEDGEMENT

I would like to take this opportunity to thank my supervisor, Dr. Su Moon Ting, who has provided generous support and guidance throughout my research project. She has been very patient with my pace and has encouraged me endlessly to persevere in anticipation of the completion of this project. I would also like to dedicate my deepest gratitude to my family, especially my parents who has given me their tremendous support throughout the duration of my Master of Software Engineering study. My gratitude also goes out to Dr. Khor Sook Mei from the Faculty of Science, University of Malaya, who has helped in the recruitment of participants for the user evaluation study of my research. Last but not least, I am also thankful to all the participants who contributed in completing the user evaluation study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

EUSC	:	End-User Service Composition
FESC	:	Front-End Service Composition
UI	:	User Interface

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

1.1 Backgrounds

Service Oriented Architecture (SOA) has emerged as an important distributed computing paradigm as it makes use of services available in the network as the fundamental elements to support rapid and low-cost development of distributed applications in heterogeneous environments (Latih, Patel, & Zin, 2014). The services in SOA are loosely-coupled with independent platforms because they are allowed to be published and hosted by different providers. According to Patel and Shah (2016), services in SOA are requested by standard protocols and consumed by applications or other services over a network without any understanding of the internal implementation.

Web services have standard-based interface that describes the web services. It also has a general accessible way by different communication protocols and the standard network protocol used to incorporate the diverse systems over a network (Sheng et al., 2014). The usage of web services has increased due to the utilization of SOA (AlSedrani & Touir, 2016). Web service technologies help websites and companies to offer simple accessibility to their web resources. It enables third parties to combine and reuse the services (Sheng et al., 2014) offered in the websites and by the companies.

SOA and web services technologies promote service composition with the goal of composing the existing reusable web services into a complex process or more capable application (AlSedrani & Touir, 2016; Sheng et al., 2014). Web services composition has become a field of research (Sheng et al., 2014). Applications can be composed from a set of suitable web services without been written manually (B. Srivastava & Koehler, 2003).

Professional programmers are able to access and use web services APIs to combine multiple services from many websites to support certain tasks (Wong & Hong, 2007). However, non-professional programmers (also known as end-user programmers) are unable to do so due to their lack of programming skills and knowledge in applying the web service APIs (Wong & Hong, 2007).

Nevertheless, studies have shown a significant increase in end-user programmers who develop their own software application as compared to professional software developers or professional programmers (Burnett & Myers, 2014). "End-user programmers" are individuals or entities who write programs to assist themselves to accomplish their primary tasks. End-user programmers teach themselves to program as they are not experienced in programming languages (Latih et al., 2014). Web 2.0 technologies has boosted the number of end-user programmers (Latih et al., 2014).

End-User Service Composition (EUSC) refers to end-user programmers composing their own applications by aggregating existing web services (Hang & Zhao, 2015). In EUSC, the end-user programmers who compose the services are also the person who are going to use the composite or composed service application (Andy Ridge & O'Neill, 2014). The users not only interact with the product software, but they are also involved in the product development process (Zhao, Loucopoulos, Kavakli, & Letsholo, 2019). These end-user programmers typically have a low level of technical knowledge (Andy Ridge & O'Neill, 2014). Even though many approaches have been introduced to support EUSC, end-user programmers still require some techniques to help them to perform EUSC (Hang & Zhao, 2015).

The common web services used in web service compositions are SOAP and RESTful (Lemos, Daniel, & Benatallah, 2015; Sheng et al., 2014). SOAP and RESTful are encapsulated as web service components (Lemos et al., 2015) or software components (B. Srivastava & Koehler, 2003). Web service composition enables end-user programmers to combine services when an existing single service is insufficient to serve the needs of the end-user programmers (Tabatabaei, Kadir, & Ibrahim, 2011).

Front-end Service Composition (FESC) was also introduced to assist end-user programmers who are lack in programming skills in composing web services. FESC enhances the intuitiveness of the service composition process for these end-user programmers (Laga, Bertin, & Crespi, 2010). FESC's composition approach is characterized by composition of the web services in the User Interfaces (UIs) (Laga et al., 2010).

1.2 Problem Statement

Even though end-user programmers welcome the opportunity to assemble or compose web services, they experience a number of conceptual and usability issues in service composition (Cappiello, Matera, & Picozzi, 2015; Namoun, Nestler, & De Angeli, 2010). Common conceptual issues are; not knowing what service composition is, (Cappiello et al., 2015; Namoun et al., 2010) or having misconceptions about service compositions such as not realizing that web services can be connected (Namoun, Owrak, & Mehandjiev, 2019); confused between design time and runtime whereby data is inserted into the input fields of web services and expected to produce results during design time (Cappiello et al., 2015; Namoun et al., 2010; Namoun et al., 2019; Radeck, Blichmann, & Meißner, 2013), and expect to see the runtime effect during the composition so as to understand and inspect the behaviour of application (Namoun et al., 2019); having difficulty in specifying the execution order of the web services (Namoun et al., 2010) and logic of application (Namoun et al., 2010; Zhai et al., 2016); not understanding some of the technical terms or terminology used during the composition process such as 'operator', being a function for the end-user and 'parameter', being the input or output field. (Namoun et al., 2010; Namoun et al., 2019); worrying about the security of sensitive information being provided to the web services that requires this information (Namoun et al., 2010). Common usability issues are difficulties in positioning web services based on their preferences to create an organized visual layout and uncertainty in whether they have done the right things (Namoun et al., 2010). Another issue that poses difficulty to end-user programmers in EUSC is lack of a unified model to coordinate the web services (Zhai et al., 2016).

1.3 Research Objectives (ROs)

To address the conceptual and usability issues of service compositions mentioned above (excluding the fear of the security of sensitive information), this research aims to develop a new FESC approach that leverages a tight integration between the development an application's UI and the composition of the web services required by the application, to enable end-user programmers to compose service-based applications out of existing web services without much difficulty.

To achieve the aim of the research, the following objectives were identified:

RO1: To review the existing web service composition approaches/techniques and features of FESC tools.

This involves reviewing the existing work on web service composition and FESC to analyse the approaches/techniques applied, as well as the strengths and weaknesses of the current work.

RO2: To develop a new FESC approach that integrates the development of an application's UI with the composition of the web services required by the application.

The proposed approach would allow end-user programmers to configure three types of flows (application flow, control flow and data flow) in composing the required web services and creating the UI of the application at the same time. It has the potential to help end-user programmers to visualize the logic of the composed service application.

RO3: To develop a proof-of-concept FESC prototype based on the proposed approach.

A proof-of-concept prototype that incorporates the approach would be developed as a working model.

RO4: To evaluate the usability of the prototype that incorporates the approach through a user evaluation study.

The user evaluation study would include think-aloud protocol, observation and survey. Participants will be asked to compose web services by using the prototype and they will be video-recorded if they have no objection to that. Think-aloud protocol and observation will be used to gather verbal responses from the participants and to observe their behaviour and performance of the required task while they are using the prototype. A questionnaire will be given to the participants after they have completed the task.

1.4 Research Questions (RQs)

The following are the RQs of this research:

RQ1: What are the current web service composition approaches/techniques and features of FESC tools?

RQ2: How to develop an approach that integrates UI application development and service composition in FESC?

RQ3: How to implement the prototype?

RQ4: How to measure the usability of the prototype?

1.5 Research Scope

This research focuses on composition of SOAP and RESTful web services. Since it is related to EUSC, the targeted users of the prototype are end-user programmers and not professional software developers. End-user programmers are individuals or entities who write programs to help themselves accomplish their primary tasks. For this research, the end-user programmers refers to the user who do not have well-trained programming knowledge and do not have web service composition knowledge, but is able to use a computer competently.

1.6 Significance of Research

The proposed approach addresses all the conceptual and usability issues of service composition (excluding the fear in security of sensitive information) mentioned in Section 1.2 Problem Statement. Refer to Section 4.2 for further details.

The novelty of the proposed approach is in enabling end-user programmers without technical knowledge to easily compose applications from existing web services by configuring three different types of flow (control flow, data flow and application flow) that represents the composition logic for the FESC process. The approach also allows them to create and visualize the graphical User Interface of the web services together with the composition logic during design time and runtime. This helps them to understand the logic of the composed application.

1.7 Report Organization

The remaining part of this report is organized as follows. Chapter 2 reviews the existing web service composition approaches and its related work on Front-end Service

Composition (FESC). Chapter 3 describes the research methodology employed in this research and the proposed approach. The system architecture and the User Interface of the prototype, and underlying design theory are included in Chapter 4. Chapter 5 discusses the results of the user evaluation to the prototype. Chapter 6 presents the conclusion and future works.

CHAPTER 2: LITERATURE REVIEW

This chapter provides reviews of existing web service composition approaches and their limitations. It also contains a section specifically on Front-End Service Composition (FESC). This section explains UI generation - different types of flows, multiple views, existing works on web service composition at the presentation layer, the related tools, and their limitations and features.

2.1 Web Service Composition Approaches

Web service composition can be classified into static or dynamic service composition based on when the aggregation of services takes place (Hang & Zhao, 2015; Lemos et al., 2015; Sheng et al., 2014). The limitations of these approaches are in italic.

2.1.1 Static Web Service Composition

In static web service composition, the aggregation of services takes place at design time (Sheng et al., 2014). The users manually selects the primitive services, and designs the composition logics, data and control flows (Hang & Zhao, 2015), and the primitive services are bound to the process at design time (Lemos et al., 2015). Static compositionis suitable in situations where business partners and service functionality requirements remain fairly constant (Sheng et al., 2014). The main approaches adopted by static service composition for end-users are workflows, spreadsheet-based, wizardand form-based, (Hang & Zhao, 2015) and block-based. *However, static approach is time consuming and error-prone.*

Workflow approach allows end-users to define the sequence of connecting web services by using graphical workflow diagrams (Hang & Zhao, 2015). It has been applied in a number of works such as Baya (Roy Chowdhury, Rodríguez, Daniel, & Casati, 2012), Flow Editor (Pi et al., 2012), Hypermash (Hang & Zhao, 2013), Co-Taverna (Zhang, 2010) and VIEW (C. Lin et al., 2008). The abstract process model involved includes a predefined list of tasks and their data dependency, with each task containing a query to search for relevant primitive service (Tabatabaei et al., 2011).

A recent systematic review of EUSC activities and tools revealed that workflow diagram editors are the most popular tools for end-users (Hang & Zhao, 2015). Visual workflow composition UI makes the composition process easy and friendly (Pi et al., 2012). Many implementations from workflow approach employs the drag-and-drop feature (Hang & Zhao, 2013; C. Lin et al., 2008; Pi et al., 2012; Roy Chowdhury et al., 2012). For example, Baya (Roy Chowdhury et al., 2012) provides an interactive modelling environment for end-users to compose and connect services on a canvas through composition actions such as select, drag, drop, delete and connect.

One of the common ways in creating workflows through visual editors is by using the pipeline method (Hang & Zhao, 2013; Roy Chowdhury et al., 2012) and blocks (Zhang, 2010). The pipeline method connects web services and includes their inputs and outputs, while the latter are pre-developed blocks that support certain tasks or functions (Latih et al., 2014).

The respective domain of the workflow affects the architectural design of the workflow management systems (C. Lin et al., 2008). Business workflows tends to be control oriented in carrying out business logic to achieve a business goal. On the other hand, scientific workflows are data oriented "aimed at enabling, facilitating, and speeding up the derivation of scientific results from raw datasets" (C. Lin et al., 2008).

However, workflow approach requires extensive domain knowledge in the service composition process (AlSedrani & Touir, 2016).

Spreadsheet-based service composition allows users to compose services in a spreadsheet environment (Hang & Zhao, 2015), where end users use spreadsheet formulas to achieve coordination among services (Obrenović & Gašević, 2008). Examples of where this approach has been applied includes AMICO:CALC (Obrenović & Gašević, 2008), Marmite (Wong & Hong, 2007), Mashroom (Wang, Yang, & Han, 2009) and Vegemite (J. Lin, Wong, Nichols, Cypher, & Lau, 2009). Some of the spreadsheet-based service compositions introduce functions that enables the use of services in aggregating data. Vegemite (J. Lin et al., 2009) and Mashroom (Wang et al., 2009) are extensions to Mozilla Firefox browser. They can import and aggregate data from different web sites. The data imported is usually arranged in an interactive table where users can filter and manipulate the results. In Marmite (Wong & Hong, 2007), users selected operators and chained them together in a data flow where data flowing through operators are shown in a table. The operators are either codes that access web services, or functions that operate locally on data (Wong & Hong, 2007). For Gneiss (Chang & Myers, 2017), it streams the hierarchical data (such as JSON and XML) from REST web service into a spreadsheet editor and allows the user to perform simple data manipulation (sort and filter) by drag-and-drop. However, it still requires spreadsheet formulas for more complex data manipulations. The user can then select the desired data from the spreadsheet editor to create a web application.

Some commercial spreadsheet environments provide mechanisms to extend their functionalities and include functions that may access web services (Obrenović & Gašević, 2008) such as StrikeIron SOA Express for Excel (as cited in (Obrenović & Gašević, 2008)), where web services are wrapped within the spreadsheet. End-users can connect the web services parameters to spreadsheet fields and call the functions to return the results (Obrenović & Gašević, 2008).

By looking at the existing works in spreadsheet approach, they require certain knowledge in spreadsheet formulas and table structures in order to successfully compose the web services.

Wizard and form-based service composition approach solicits key information (such as location of primitive service, order of invoking service, and so on) of the service composition from the users (Hang & Zhao, 2015) by using forms and wizards. Easy SOA (Yamaizumi, Sakairi, Wakao, Shinomi, & Adams, 2006) is an example of those who have used this. Easy SOA provides an environment for end-users to develop web services and web applications by placing cards on a sheet constructed in a web browser. Each of these cards acts as a single-function application and contains data or expressions that evaluates the data at run time. End-users fill in the information on these cards such as WSDL URL and other variables in order to extract the data structures and generate the interface. The interface between cards which represents - service methods using web browsers are connected through simple operations. *However, information configuration for every web service is troublesome for the end-users.*

Programming-by-demonstration approach provides service composition platforms that can record the composition logics and configurations by end-users and then reapply them in other composite services that bears similarities to the existing one (Hang & Zhao, 2015). The concept is to generate a script based on the demonstrated procedures which can be used for different variations and parameters (Barricelli, Cassano, Fogli, & Piccinno, 2019). For example, in web scripting or web macros, repetitive common tasks in a web browser is recorded and replayed. One specific example is CoScripter (Bogart, Burnett, Cypher, & Scaffidi, 2008). CoScripter uses programming-by-demonstration language. It records and replays the actions of users for common tasks in the FireFox browser. The actions are transformed into a script and then saved. Users can reuse the script as it is or with some modification. CoScripter can save the data in a minimalist data structure which does not support any looping or conditioning features. This database is used in future executions instead of re-entering the data. It helps to resolve the need of memorizing the detailed information and long navigation sequence. *However, end-users have to record a new action or rearrange the script for a new scenario.*

Block-based programming approach allows the end-user to drag and drop graphical blocks to create their own program (Bak, Chang, & Choi, 2020). It is built by assembling jigsaw puzzle pieces which presents visual cues (Coronado, Mastrogiovanni, Indurkhya, & Venture, 2020). This approach is usually engaged to the rule-base such as *if-then-else* conditional statements, ECA (Event-Condition-Action) rule or trigger-action rules and feature configurations. Smart Block (Bak et al., 2020) and EUD-MARS (Akiki, Akiki, Bandara, & Yu, 2020) are examples of those that has used this block-based programming to program 'Internet of things' (IoT) applications by using the online services. *However, conflicting rules* (Ardito et al., 2019) *and difficulty in understanding the implications of multiple rules* (Coronado et al., 2020) *are the drawbacks of this approach for non-programmers.*

2.1.2 Dynamic Web Service Composition

Dynamic web service composition automatically creates composite services based on users request and context (Hang & Zhao, 2015). To do that, the execution system is required to support automatic service discovery, selection and binding (Sheng et al., 2014). In dynamic composition, the determination and replacement of constituent services takes place during runtime (Sheng et al., 2014) or deployment time (Lemos et al., 2015). It is particularly useful when runtime change of requirements are frequent and when services cannot be predicted at design time (Sheng et al., 2014).

Some of the approaches employed in dynamic service composition are: use of high-level graphical language to define composite service, visualization, wizard-based and natural language processing (Hang & Zhao, 2015). Other approaches make use of semantic technologies and AI planning techniques (Sheng et al., 2014). End-users usually specify the business goal in a description language or selected notation (Tabatabaei et al., 2011). FUSION (Sheng et al., 2014), SWORD (Sheng et al., 2014) and OWLS-Xplan (Sheng et al., 2014) are some examples of dynamic service composition.

Fusion (as cited in (Sheng et al., 2014)) provides a graphical interface for users to specify the abstract requirements of a composition goal. The system takes the inputs of a user specification and generates an optimized execution plan. The plan will be executed and verified to ensure that the results meets the user requirement criteria and that the appropriate recovery process will be initiated if verification failure occurs before the response is delivered to the users.

OWLS-Xplan (as cited in (Sheng et al., 2014)) is using a way similar to Fusion in defining the abstract requirements of a composition goal. Xplan which is an artificial intelligence planner is used to generate the service composition plan from PDDL (Problem and Domain Description Language) description of OWL-S services and a planning query. Xplan consists of pre-processing and planning modules. The pre-processing module is used to create the required data structures, generating the initial connectivity graph and goal agenda while the planning module supports the heuristically relaxed graph-plan generation and enforced hill-climbing search.

SWORD (as cited in (Sheng et al., 2014)) uses Entity-Relationship (ER) model to specify web services, instead of using emerging service standards such as WSDL. Each of the service is defined in terms by its inputs and outputs in an ER model consisting of the entities and the relationship among the entities. The initial and final states of the composite service need to be specified in order to create a composite service. A rule-based engine is used to automatically determine whether the required composite service can be realized by using existing services.

A research (Driss, Aljehani, Boulila, Ghandorh, & Al-Sarem, 2020) proposes FCA (Formal Concept Analysis) and RCA (Relational Concept Analysis) - Driven Approach for web service composition. The user is required to model the composition scenario by using Business Process Model and Notation (BPMN) with its notational elements. The composition scenario provides semantic description for the user's requirement by using Unified Foundational Ontology (UFO). The appropriate web services are discovered by filter matching based on similarities of keywords and requirement descriptions. FCA is used to select the optimal web service that offers the best compromise of QoS (Quality of Service), QoE (Quality of Experience), and QoBiz (Quality of Business) properties while RCA is used to minimize the required adaptation efforts for composability and offers maximum QoS, QoE, and QoBiz before executing the composition.

Some researches use natural language-base to perform service composition. The approach processes the natural language provided by the end-user as users request and the system finds the services to achieve its goal. This approach usually (a) apply restrictions on sentence constructions to match the service descriptions, (b) requires lexical database to compute the concept similarity of constructed semantic graphs that represents the service description and (c) match the grammatical relations of natural language requests to the semantic web service descriptions (as cited in (Romero, Dangi, & Akoju, 2019)). A research NLSC (Romero et al., 2019) allows the end-users to express their needs by using unrestricted natural language for compositions. It had adopted the semantic service matching by embedding sentences instead of description languages in terms of service name, functionality, parameters and conditions. *However, the web service in the approach requires the natural language descriptions annotation provided by the service developer which many existing web services do not have.*

Nevertheless, these dynamic web service composition approaches rely on knowledge of a certain kind of modelling (such as abstract requirement specification, and states of composite service, BPMN), that most end-users do not have.

2.1.3 Semi-automated Web Service Composition

Although dynamic service composition automates some of the tasks involved in assembling services, it limits the freedom of users in the process of service composition. A study on establishing requirements for EUSC tools shows that an appropriate amount of end-user's involvement in service composition is required (Andy Ridge & O'Neill, 2014). Some non-programmers think that manual service composition could provide the freedom to develop personalized application (Namoun et al., 2019). High degree of automation in service composition is generally not desired (Vulcu, Bhiri, Hauswirth, & Zhou, 2008).

Some research has tried to leverage both manual and automatic composition to assist users in the composition process (Sheng et al., 2014). An example of semi-automated service composition can be seen in a template-based service composition (Mehandjiev, Lecue, Wajid, & Namoun, 2010) that allows users to compose services by selecting a template according to their needs. This approach usually provides several templates that are classified in a domain taxonomy. A user is allowed to choose and modify the tasks workflow in the templates based on his or her preference to meet his or her particular requirements. The underlying system will suggest suitable services for the tasks once the template modification is done. The system will also adjust the list of tasks dynamically if necessary. *Despite that, limited templates are available and the templates provided may not fulfil all the system requirements of end-users.*

DoCoSoc (Marin & Lalanda, 2007) is another example of semi-automated service composition where it uses domain SOA model to automate the service composition. It produces a fast and easy service-based application development. However, users have to provide the domain SOA meta-model. The code is automatically obtained from the abstract application model using artefacts available in the service repositories. *This modelling process required expertise in Model Driven Architecture (MDA) and is not suitable for end-users.*

A research (Kasmi, Jamoussi, & Ghézala, 2018) adopted an intentional modelling of web service composition based on MAP formalism (process model represented by the directed graph). This is a collaborative and interactive web service composition approach. The approach allows the user to construct the software requirement specification in a MAP model which is partially drawn by the domain experts. The intentional services will be identified and associated to the section in map to generate the "COLMAP" model. The intentional service is a service model that comprises the intention to accomplish the task, pre-condition, post-condition, input parameter and output parameter which is corresponding to the users requirements. Group Recommender System (GRS) recommends a set of web service to user. The user selects the web services and execute the composite services. It is a collaborative and interactive web service composition approach. *This approach relies on the MAP model provided by the experts and users might need to modify the MAP model based on different requirements which is not suitable for the end-users.*

Table 2.1: Summary of Web Service Composition Categories and Approaches

Composition category	Approach	Example	Limitation of approach
Static Composition	Workflow	Baya Flow editor Hypermash Co-tarvena VIEW	• Requires extensive domain knowledge
	Spreadsheet	AMICO:CALC Marmite Mashroom Vegemite	• Requires understanding of spreadsheet formulas and table structures
	Wizard-and form-based	Easy SOA	• Requires information configurations
	Programming-by-demonstration	Co-script	• Requires recording a new action or rearranging the scripts for different scenarios
	Block-based	Smart Block EUD-MARS	• Conflicting rules and difficulties in understanding the implications of multiple rules
Dynamic Composition	Abstract model	Fusion OWLS-Xplan	• Requires abstract requirement specifications
		FCA and RCA Driven Approach	• Requires business process models
	Entity-Relationship model	SWORD	• Requires specifying the initial and final states of the composite service
	Natural language	NLSC	• Relies on the natural language description provided by the service provider
Semi-automated Composition	Template-based	Template-based service composition prototype	• Limited templates are available and they might not fulfil the end-users' requirement.
	Abstract model	DoCoSoc	• Requires MDA knowledge
	MAP formalism	Collaborative and interactive WSC	• Relies on the MAP model and requires modifying the MAP model

2.2 Front-end Service Composition (FESC)

FESC applies an approach of service composition at the presentation layer, in which applications are developed by composing web services using their UIs rather than application logic or data (Nestler, Feldmann, Hübsch, Preußner, & Jugel, 2010). The idea originated from graphical UI integrations which refers to integrating components by combining their front-end presentations, rather than their application logic or data (Daniel et al., 2007). The UI component models for FESC usually employs reusable UI components (Daniel et al., 2007; Radeck et al., 2013) or generates suitable UIs based on description files (Laga, Bertin, Glitho, & Crespi, 2012; Nestler et al., 2010; Zhai et al., 2016) and these UI components represents the services. Most of the FESC make use of drag-and-drop and wiring in composition design paradigms which do not require manual coding (Pietschmann, Nestler, & Daniel, 2010). FESC allows the end-users to play the role as service composer and application designer at the same time (Nestler et al., 2010), and this is an advantage for the end-user programmers. This composition at the presentation layer helps to reduce the cognitive challenges and efforts faced by the non-programmers during the service composition process (Namoun et al., 2019).

Another area that contributes to FESC is mashup. A number of works (iGoogle (http://www.google.com/ig), Yahoo! Pipes (http://pipes.yahoo.com/), Yahoo! Dapper (http://open.dapper.net), Netvibes (http://www.netvibes.com), JackBe Presto Cloud (http://prestocloud.jackbe.com/), Microsoft Popfly (http://www.popfly.com), OpenKapow and Kapow Katalyst (www.kapowsoftware.com), AMICO (http://amico.sourceforge.net/), Marmite (http://www.cs.cmu.edu/~jasonh/projects/marmite/) or EzWeb (http://ezweb.morfeo-project.org/) "mashup" interoperable and highly configurable visual interface elements into a user-centered interface that can be used to invoke the backend services (Lizcano, Alonso, Soriano, & Lopez, 2011). The word "mashup" is initially used in audio domains, to refer to the remixing of two or more audios into a new entity (Liu, Hui, Sun, & Liang, 2007). The word later became a common term in the web application area with many researches working on mashup solutions (Yu, Benatallah, Casati, & Daniel, 2008).

Mashup in web application is a process of integrating data or content from different sources from Internet (Liu et al., 2007). These mashup solutions mainly create data mashups but not service mashups (Lizcano et al., 2011). Later, researchers begun using mashup as one of the web service composition method (Liu et al., 2007; Sheng et al., 2014). This has motivated the practice of mashing up front-ends of resources to simplify end-users' exploitation and invocation of web services tailored to their context and knowledge (as cited in (Lizcano et al., 2011)).

2.2.1 UI, Types of Flow and Multiple-View

Since the target users of FESC are end-user programmers, the composition process should not involve any code writing or technical knowledge (Andrew Ridge, 2014). This could be achieved with proper support for UI generation, configuration of different types of flow and multiple-view.

The UI of services should be presented during the composition process to enable an end-user to see the outcome of the composition process (Andrew Ridge, 2014). Therefore, the generation of UI is an important feature of FESC.

Control flow and data flow are the two essential types of composition constructs in web service composition (Lemos et al., 2015). In web service composition, control flow refers to the execution order of atomic services (Paik, Lemos, Barukh, Benatallah, & Natarajan, 2017) and the dependency among activities (Agarwal et al., 2005) while data flow refers to the data flowing from one activity to another (Yang, 2003) and the dependency among data manipulations (Agarwal et al., 2005). The control flow and data flow between web service components should be represented in the composition because users need to be able to identify the execution order of components and data being passed between the components (Andrew Ridge, 2014). Besides that, the user should be able to edit the order of service components in composition because the user might position the components in a wrong order in the initial stages or change their mind on the components execution order during the composition process (Andrew Ridge, 2014).

Applications composed out of web services typically include multiple UI pages/screens and the transition between these pages. The term “application flow” is used in this research to refer to the transitions between the UI pages of a composed service application. Other studies have used different terms to refer to the same thing: “program flow” in WIDE (Okamoto, Dascalu, & Egbert, 2006) , “page flow”/“process flow” in ServFace (Nestler, Dannecker, & Pursche, 2009) page transition framework in image-oriented web programming (Shimomura, 2004). It is important to make application flow visible as it helps to sort out page-transfer relationships (Shimomura, 2004) and it provides an overview of the whole structure of an application (Okamoto et al., 2006).

It is useful to overlay the various representations of information under one viewport (Roberts, 1998). However, problems arise when too much information are shown in one view such as increasing irrelevant details which will confuse the final results and issues in interpreting or perceiving the information by the end-users (Roberts, 1998). Therefore, it is useful to split the information into multiple views (Roberts, 1998).

Multiple-view system is a system which uses two or more different views to support the study of a single conceptual entity (Baldonado, Woodruff, & Kuchinsky, 2000). It can help to create better understanding of the underlying information by interpreting the information from different perspectives (Roberts, 1998). A design guideline for multiple-view has been proposed to help designers decide when multiple-view is desirable (Baldonado et al., 2000). The guideline shows a model of multiple-view system based on three dimensions-selection, presentation and interaction (Baldonado et al., 2000). Eight rules were proposed to guide the designer on when and how to use multiple-view (Baldonado et al., 2000).

2.2.2 Existing Works on FESC

This section describes existing studies on FESC. Apart from the limitations (shown in italic) of the studies, this section explains how these studies provide the following features that are important in FESC: UI generation, type of flow supported and multiple view. *ServFace Builder*: Figure 2.1 shows the screenshot of ServFace Builder. To integrate it into an application, ServFace Builder allows an end-user to drag a service operation from its Service Component Browser as shown in Figure 2.1 (4) to its composition canvas (Namoun et al., 2010; Nestler et al., 2009; Nestler et al., 2010). It then automatically generates the corresponding service UI as shown in Figure 2.1 (1) and (2). ServFace Builder allows end-users to design the data flow between connected services by connecting the UI element from where the data is obtained to the UI element serving as the destination of the data. The data flow is represented by a linking arrow between the two UI elements as shown in Figure 2.1 (3). Service operations can be dragged to the same page or different pages to create a multi-page application. The end-user can connect two pages to create a page transition as shown in Figure 2.2, which signifies the page flow or application flow. The order of execution of the service operations, or control flow, is implicit, following the application flow. *As shown in Figure 2.2, the end-user needs to switch to a different view to define the page flow. The page flow configuration is targeted on experienced users to define more complex process. ServFace Builder requires service developers to provide web services annotations to improve the visual appearance of the resulting applications (Nestler, Dannecker, et al., 2010).*

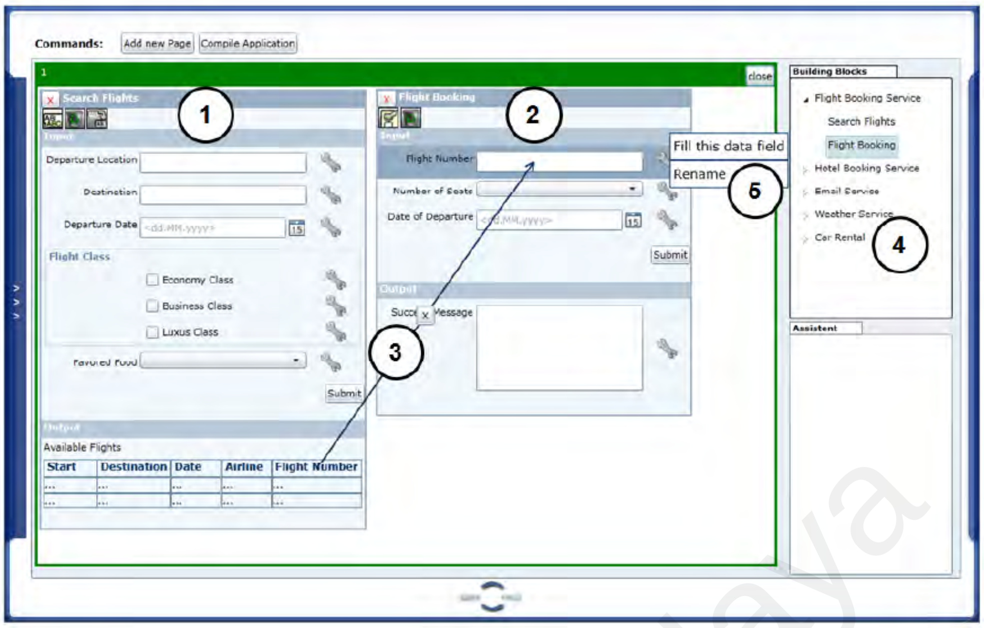


Figure 2.1: Screenshot of ServFace Builder

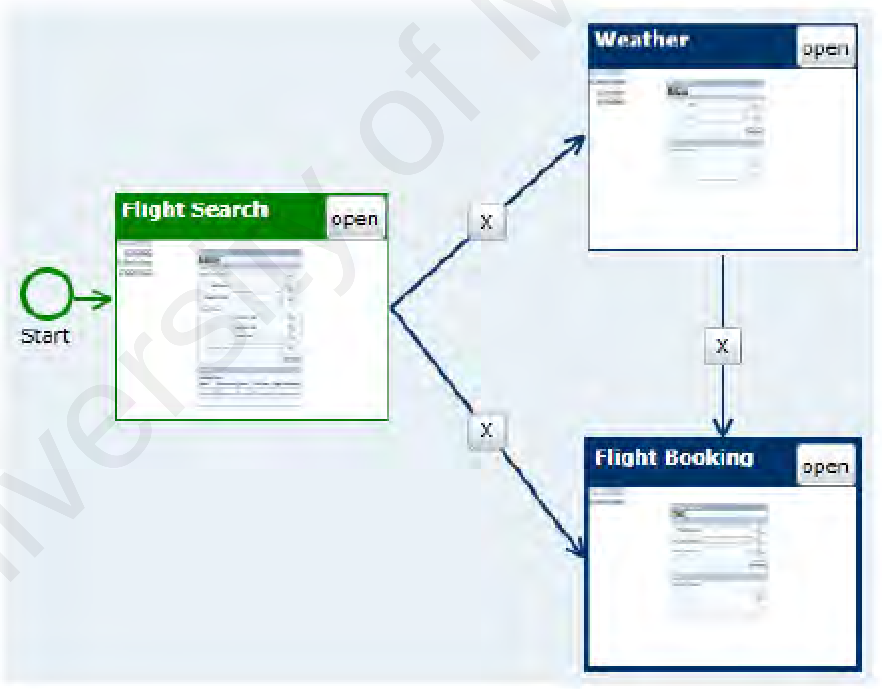


Figure 2.2: Page Flow View of ServFace Builder

MashArt: MashArt (Daniel, Casati, Benatallah, & Shan, 2009) proposes to create composite web applications by integrating data, application, and UI components. It allows the modelling of the three types of components by using a unified model. It combines event-driven philosophy of UI and control-flow-based philosophy of service orchestrations. Basically, the service components are linked to the UI components via connectors while the events are attached to the UI components. It supports various types of components such as RSS and Atom feeds for data component, SOAP and RESTful web services for service components, and JavaScript UI components. Core functionality service component models which are reusable are provided to users. The composition canvas in the MashArt editor provides a visual model view of composition logic as shown in Figure 2.3. *However, the components are not properly organized in the canvas because users are free to drag-and-drop and arrange them. In addition, MashArt targets advanced web users as it requires the users to have the understanding of communication protocol* (Pietschmann et al., 2010).

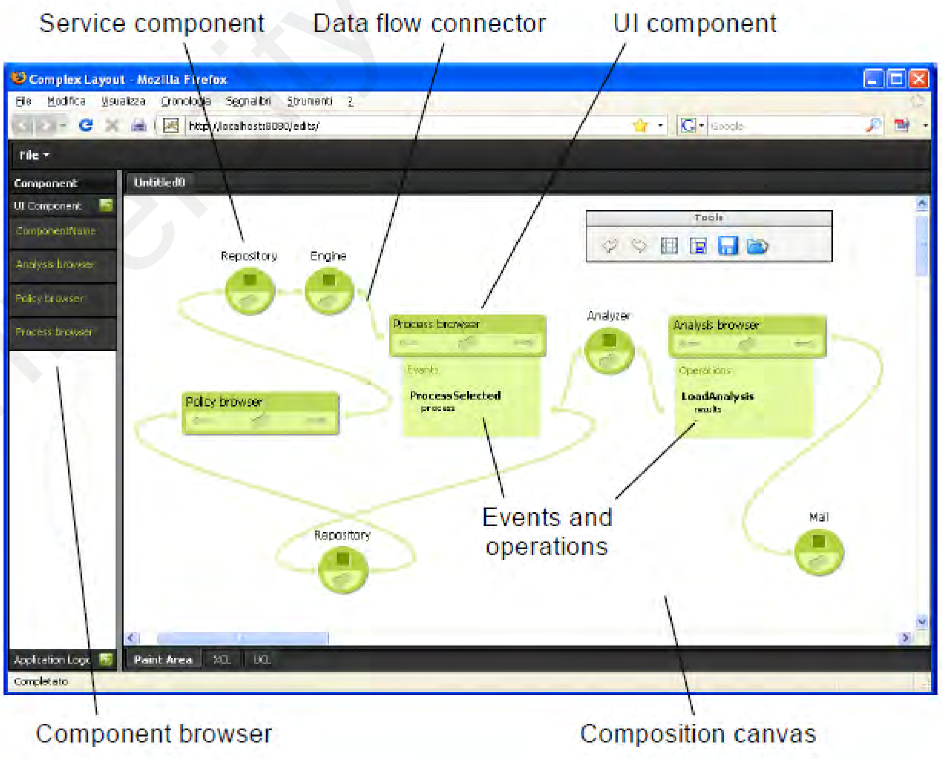
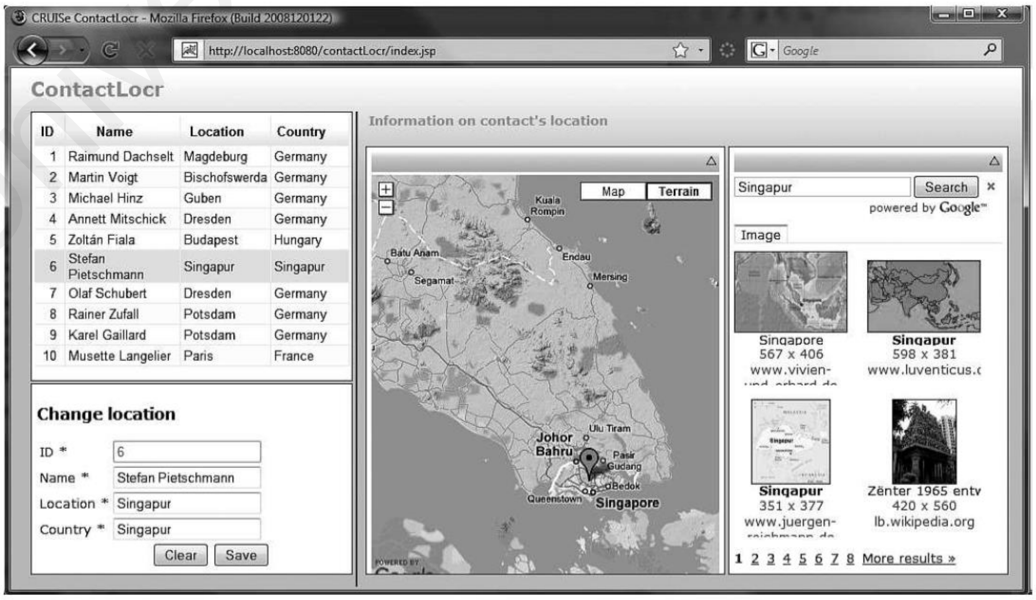


Figure 2.3: Screenshot of MashArt

CRUISe: Figure 2.4 shows the screenshot of CRUISe. CRUISe (Pietschmann et al., 2010; Pietschmann, Voigt, & Meissner, 2009; Pietschmann, Voigt, Rümpel, & Meißner, 2009) which employs service-oriented paradigm for web-based UI developments. In CRUISe, UI components are provided as reusable services, namely, User Interface Services (UIS). They can be selected, configured and exchanged dynamically based on the model context. These reusable UI components integrates the UI logics at the presentation layer. The data or application logics are provided by back-end services. With a homogeneous access layer, the backend services can be bound to UI services. The UI components concept eases the development, maintenance and upgrading of the UI. Integration of services is carried out on the client's side to achieve a lightweight service orchestration on the presentation level. It also supports dynamic adaptationssuch as UIS reconfiguration and exchange. *However, CRUISe is not suitable for end-user programmers as it requires them to have the knowledge of component-based software and event-based communication in defining the composition description. Besides that, CRUISe lacks of application flows to organise the services and does not show the control flow and data flow.*


CRUISe ContactLocr - Mozilla Firefox (Build 2008120122)
http://localhost:8080/contactLocr/index.jsp
Google
ContactLocr
Information on contact's location
Map
Terrain
Singapur
Search
powered by Google
Image
Singapore 567 x 406 www.vivien-
Singapur 598 x 381 www.luventicus.
Singapur 351 x 377 www.juergen-
Zënter 1965 entv 420 x 560 lb.wikipedia.org
1 2 3 4 5 6 7 8 More results »

ID	Name	Location	Country
1	Raimund Dachselt	Magdeburg	Germany
2	Martin Voigt	Bischofswerda	Germany
3	Michael Hinz	Guben	Germany
4	Annett Mitschick	Dresden	Germany
5	Zoltán Fiala	Budapest	Hungary
6	Stefan Pietschmann	Singapur	Singapur
7	Olaf Schubert	Dresden	Germany
8	Rainer Zufall	Potsdam	Germany
9	Karel Gaillard	Potsdam	Germany
10	Musette Langelier	Paris	France

Change location

ID *	6
Name *	Stefan Pietschmann
Location *	Singapur
Country *	Singapur

Clear Save

Figure 2.4: Screenshot of CRUISe

Service Creation Environment (SCE): Figure 2.5 shows the screenshot of SCE. One study proposed a Service Creation Environment (SCE) comprising a widget-based abstraction layer and a two-step service composition mechanism (Laga et al., 2012). A widget is a reusable GUI that is linked to one or more functionalities of a service. Every widget has a description file that contains abstract description and implementation description. Abstract description is used to explain the functionality of the widget while implementation description refers to the index URL that provides access to the functionality. The GUI can be generated semantically based on the widget description. The first steps of the mechanism includes GUI generation based on selected widgets and creation of a composite service through automatic semantic matching, and linking of all connectable widgets. The connectable links and GUI elements involved are shown to users for further modification. In the second step, the user can manually personalize the services composed. *The emphasis on semantic service composition restricts it to minor customization such as removing the unused generated links. Besides that, there is no proper organization of application flow and no visualization of the control flow of composed services.*

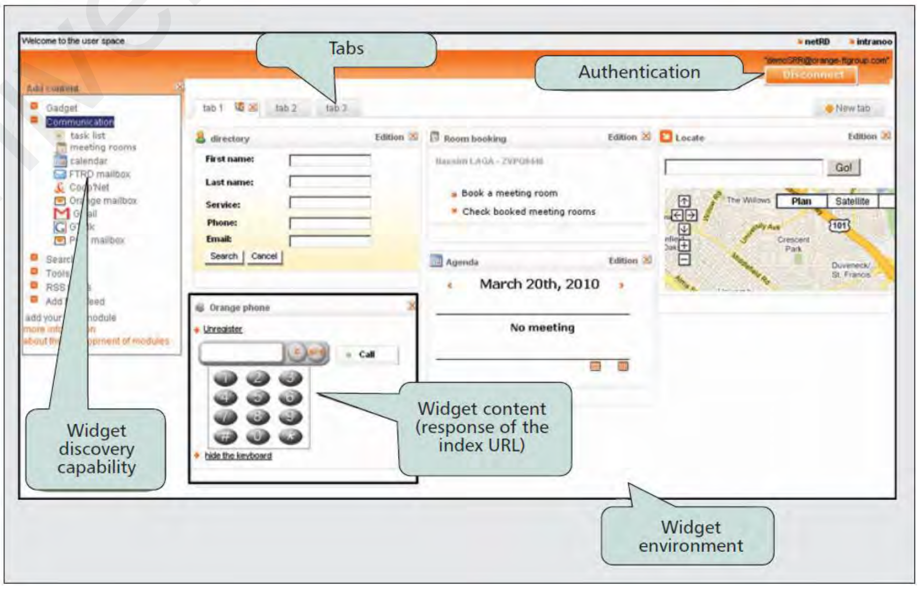

Tabs
Authentication
Widget discovery capability
Widget content (response of the index URL)
Widget environment

Figure 2.5: Screenshot of SCE

Lightweight Service Creation Environment (LSCE): Figure 2.6 shows the screenshot of LSCE. Another study which has developed a Lightweight Service Creation Environment (LSCE) based on a data-driven service creation approach to compose services for mashup applications (Zhai et al., 2016). In this study, service is the basic data unit, and a Service Data Model (SDM) was created to support service description, data transforms, visualization, and extension of services. The study developed an IFrame implementation for the SDM. The LSCE provides a drag-and-drop workspace for developing applications by drawing dataflow graphs, also known as Service Process Graphs (SPGs) which are shown in the right column of the LSC workspace in Figure 2.6. SPGs would be parsed into JSON-based script before being sent for execution purpose. *Being data-driven, LSCE does not provide a clear visualization of application flow and control flow.*

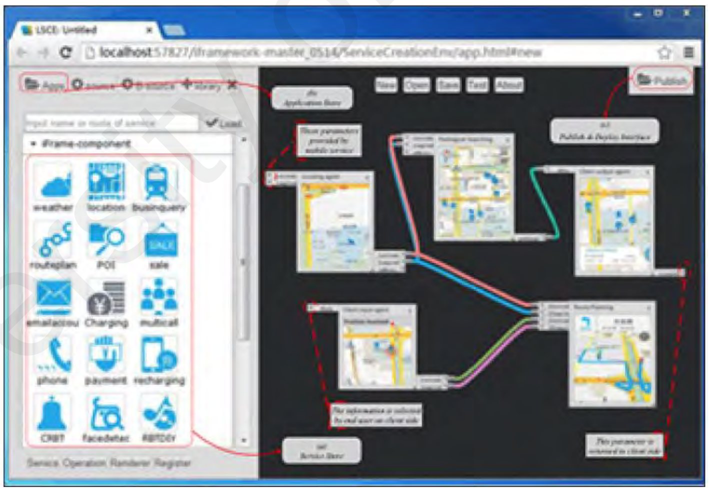


Figure 2.6: Screenshot of LSCE

CapView: Figure 2.7 shows the screenshot of CapView. CapView (Radeck et al., 2013) proposes an approach of composing the web services (encapsulated as service components) on the functional level instead of structural units. It provides the functional abstraction to ease users into visualizing the functionalities of components and composing the components. Every component has a representation which describes the capabilities and properties of the component in natural language that is derived from the semantic annotation. Short sentences are displayed to explain the functionalities of components as well as the meaning of their properties to the users, as shown in Figure 2.7. The capability of the component is described as a tuple with *activity* (actions to be performed, such as search and display), *entity* (domain objects, such as Hotel and Flight) and, *requiresInteraction* (activities in whichthe user is involved). Users will be guided on the connectible capabilities and properties, enablingconfirmation of the coupled functionalities (data flow) before the running UI components are presented in another view, namely, LiveView (runtime mode). Service components are recommended to users (based on semantic matching) in a separate recommendation menu, as shown in the right column of CapView in Figure 2.7. Service components that can be coupled are shown in blue, and orange if otherwise. *Users tend to get mixed up between CapView and LiveView, as they assumed that the components would display the execution results directly in CapView. Besides that, users interpret the input and output ports of services differently from when they interpret them through a human-centred perspective. For example, users expect "Select an event" function to only have an input. However, through the system-oriented perspective, the function also has an output (such as list or details). Users need to switch to CapView in order to configure composition logic, and switch to LiveView to witness the outcome of composed services. Aditionally, CapView does not provide control flow and application flow in organizing the order of the services selected for the composition.*

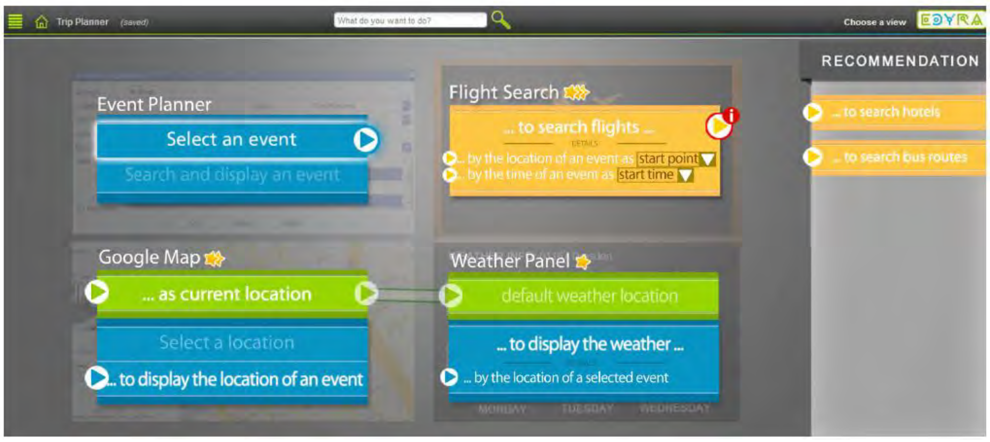


Figure 2.7: Screenshot of CapView

Table 2.2: Summary of Existing Works on FESC

Related Work/Tool	Approach	Features	Limitation on the features
ServFace Builder (Namoun et al., 2010; Nestler et al., 2009; Nestler et al., 2010)	Wizard-and form-based	*UI generation:* Based on service description and attached annotations, UI generated represent web services	
		Type of flow: Application/page flow, control flow and data flow	Lack of control flow visualization.
		Multiple view: Not applicable	Need to switch view to visualise application flow and data flow.
MashArt (Daniel et al., 2009; Pietschmann et al., 2010)	Workflow	*UI generation:* Reusable UI components provided by component developers	No flexibility on UI generation as it relies on component developers.
		Type of flow: Control flow and data flow	Lack of control flow and application flow visualization to organise the services.
		Multiple view: Not applicable	
		Other: 3 types of components (data, application, and UI components)	Require understanding of 3 types of components for composition process Require knowledge on communication protocol between components
CRUISe (Pietschmann et al., 2010; Pietschmann, Voigt, & Meissner, 2009; Pietschmann, Voigt, Rümpel, et al., 2009)	Wizard-and form-based	*UI generation:* Auto search for suitable UI components captured as reusable services in database	Return the UI component based on description (might return unsuitable UI component)
		Type of flow: Control flow and data flow	Lack of application flow visualization to organise the services
		Multiple View: Visualization of information in multiple views	
		Other: Context-aware composition	Requires knowledge on component-based software and event-based communication in providing composition description

Table 2.2 continued

Related Work/Tool	Approach	Features	Limitation on the features
SCE (Widget-based two-step service composition mechanism) (Laga et al., 2012)	Wizard-and form-based	*UI generation:* Based on description file of the data model of GUI widgets (reusable GUI attached to a service)	
		Type of flow: Data flow	Lack of control flow and application flow visualization to organise the services
		Multiple view: Not applicable	
		Other: Provide suggestions for linking of connectable widgets	
LSCE (Data-driven service creation approach) (Zhai et al., 2016)	Wizard-and form-based	*UI generation:* Generate Iframe (represent services) based on annotation template provided by service provider	No flexibility on UI generation as it relies on service provider
		Type of flow: Control flow and data flow	Lack of control flow and application flow visualization to organise the services
		Multiple view: Not applicable	
CapView (Radeck et al., 2013)	Workflow	*UI generation:* UI components provided by service providers	No flexibility on UI generation as it relies on component developers UI is only viewable in LiveView (runtime mode)
		Type of flow: Data flow	Lack of control flow and application flow visualization to organise the services
		Multiple view: Not applicable	Need to switch between CapView and LiveView
		Other: Semantic description of component functionality; Provide recommendation menu and hints on coupled-able components	

Table 2.2 summarizes the existing works on FESC presented in this section. In summary, there are two main problems with the existing works. Firstly, lack of control flow and/or application flow visualization. Secondly, some still requires users to have certain technical knowledge such as communication protocol (Daniel et al., 2009), component-based software (Pietschmann, Voigt, Rümpel, et al., 2009) and event-based communication (Pietschmann, Voigt, Rümpel, et al., 2009) that end-user programmers in general do not possess.

2.2.3 Comparison to Related Works

A number of works discussed in Section 2.1 show efforts in helping end-users with web service composition. However, those approaches consists of some limitations, as shown in Table 2.1. which forms obstacles for end-users. The end-user is required to know the features (workflow modelling, table structure and information configuration properties) and have the knowledge (spreadsheet function and formula, and understanding of script) for static web service composition approaches. For dynamic and semi-automated web service composition, the end-user is required to have the modelling knowledge, rule-based and specify the requirements. Some approaches rely on the templates and service description provided by the service provider, which restricts the composition coverage. Therefore, this research applies an approach of service composition at the presentation layer where the service is represented by the GUI. This FESC combines the presentation front-ends rather than the composition techniques. Even though there are existing works on FESC, but there are also limitations as shown in Table 2.2, which is mainly lack of control flow and/or application flow visualization, and the fact that some still requires users to have certain technical knowledge. Hence, three types of flows (control flow, data flow and application flow as described in Section 2.2.1), are used to represent a simple composition logic at the same time provide a visually organized layout for the composed services.

Table 2.3 summarises the comparisons of this research with the related works on FESC. In terms of composition model, CRUISe requires composition knowledge to define the composition logic and description in the context module to find a suitable UI for the web service. The components are configured with description files and connected one from the other, through the definition of the events and operations in the users requirement context; MashArt requires the understanding of communication protocol between components and event attachment to the UI components. Users need to attach the events and operations to UI components and connect to service components to compose the mashup service - ServFace Builder employs the hybrid of control flow and data flow in its composition model; SCE, LSCE and CapView use the data flow as the composition model. This research employed the hybrid of application, control and data flows as the composition model.

Table 2.3: Comparison with Related Works

Related works	**Composition model**	**Type of flow**			**Technical knowledge required**
		Application flow	**Control flow**	**Data flow**	
ServFace Builder (Namoun et al., 2010; Nestler et al., 2009; Nestler et al., 2010)	Hybrid of control and data flows	Explicit	Implicit	Explicit	None
MashArt (Daniel et al., 2009; Pietschmann et al., 2010)	Event-based	-	Implicit	Explicit	Communication protocol
CRUISe (Pietschmann et al., 2010; Pietschmann, Voigt, & Meissner, 2009; Pietschmann, Voigt, Rümpel, et al., 2009)	Abstract model	-	Implicit	Implicit	Component-based software and event-based communication
MashArt (Daniel et al., 2009; Pietschmann et al., 2010)	Data flow	-	-	Explicit	None
LSCE (Zhai et al., 2016)	Data flow	-	Implicit	Explicit	None
CapView (Radeck et al., 2013)	Data flow	-	-	Explicit	None
This research	Hybrid of application flow, control and data flows	Explicit	Explicit	Explicit	None

All the works in Table 2.3 includes data flow. This could be due to the semantics of data flow being easy to understand (Pietschmann et al., 2010). All works except SCE and CapView, includes control flow. Only ServFace Builder in its proposed works include application flow. This research views a provided flow as explicit if it is visually visible to the end-user programmers, and as implicit if the flow is implied and not directly visible. MashArt allows end-user programmers to connect the data flow between components explicitly in its editor. In MashArt, control flow is implicitly configured as events attached to the components. CRUISe allows control flow and data flow to be configured in the user's requirement context and this is implicit. SCE allows end-user programmers to compose services by configuring data flow through creation of explicit links between GUI widgets. In LSCE, data flow configuration is done through creation of explicit links between services, the control flow is implied by the rules in the services. CapView allows the data flow to be connected between the service components. In terms of technical knowledge required, studies have reported that CRUISe (Pietschmann et al., 2010; Pietschmann, Voigt, & Meissner, 2009; Pietschmann, Voigt, Rümpel, et al., 2009) and MashArt (Daniel et al., 2009; Pietschmann et al., 2010) are not for end-users programmers (Pietschmann et al., 2010).

ServFace Builder is the closest to this research. It also provides the three-flow features but the control flow is implicitly implied by the application or page flow. Control flow could be implied by the application flow in simple applications, but in more complex applications, web services might need to be executed concurrently, requiring the control flow to be separated from the application flow. ServFace Builder allows the definition of two of the five basic control flow patterns (sequence, merge, split, condition and loop) (Zhai et al., 2016). They are sequential control flow (sequence) and alternative control flow (condition). This research supports the definition of sequence, merge and split. In ServFace Builder, the standard view enforces a sequential application/page flow. To alter it, the end-user programmers need to switch to the “page flow” view. The three-flow approach in QuickWSC shows the visualization of application, control and data flows explicitly by using a multiple-view design which results in end-user programmers not having to switch to a different view.

2.3 Summary

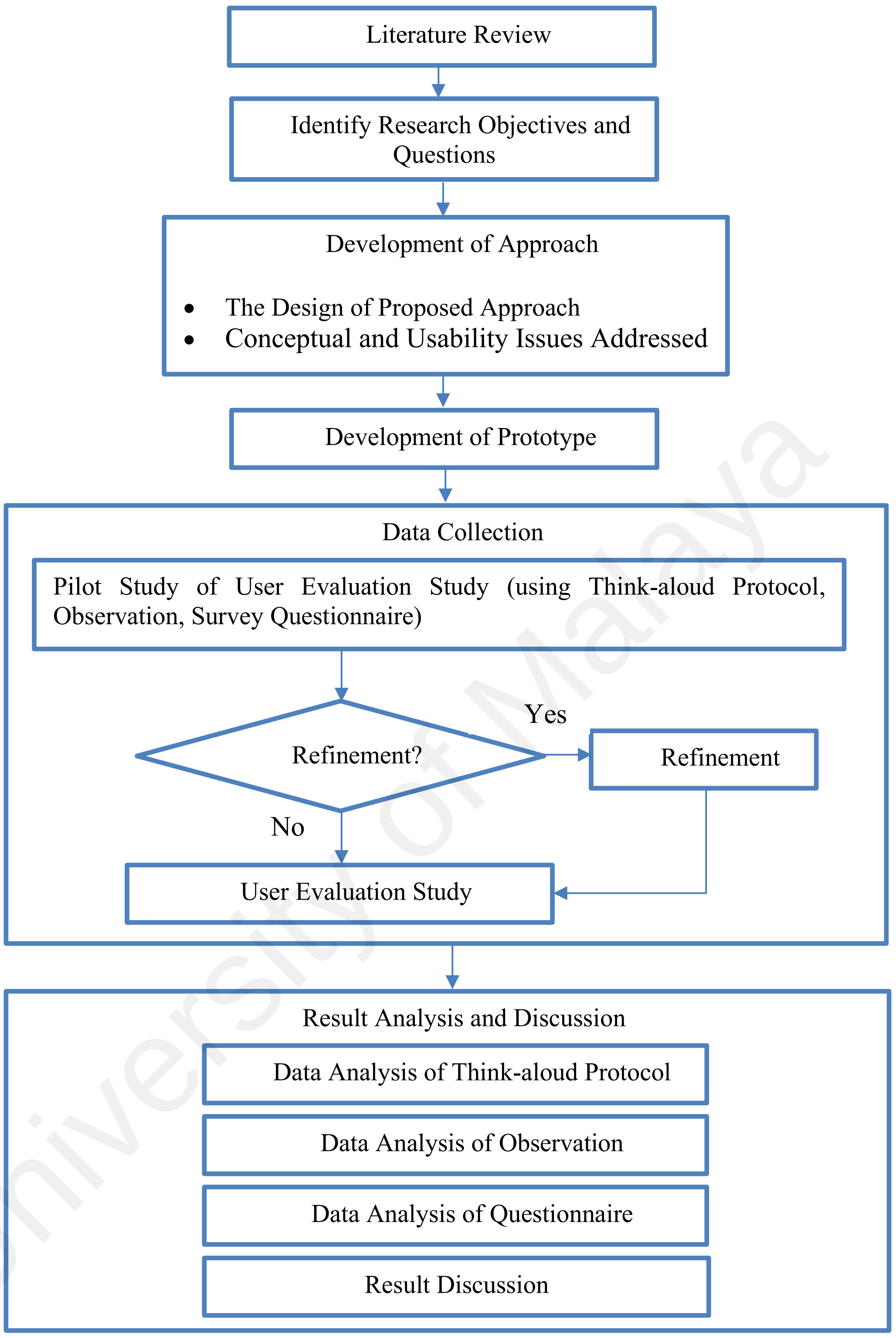
This chapter starts by reviewing the existing approaches/techniques applied in EUSC which can be classified into static web service composition, dynamic web service composition and semi-automated web service composition. The concept of the approaches/techniques was discussed in detail for each section. Each of the approaches/techniques possesses strengths and weaknesses which were also discussed. Following that, existing works about the FESC was reviewed. FESC applies an approach of service composition at the presentation layer, in which applications are developed by composing web services using their UIs rather than application logic or data. This chapter also discusses and summarises the aspects of existing works on FESC such as UI generation, types of flows supported, multiple view concept and limitation of features. Lastly, comparison with the related works is carried out to demonstrate the differences between the proposed works and the related works.

CHAPTER 3: RESEARCH METHODOLOGY

This chapter presents the research methodology adopted in this research. It explains the key steps involved in this research. It also outlines relevant design guidelines/techniques and methods where applicable, with further details given in the respective chapters.

3.1 Research Methodology

Figure 3.1 shows the flow of the research activities. It comprises 6 key steps as explained in the following sections.



Literature Review
Identify Research Objectives and Questions
Development of Approach
• The Design of Proposed Approach
• Conceptual and Usability Issues Addressed
Development of Prototype
Data Collection
Pilot Study of User Evaluation Study (using Think-aloud Protocol, Observation, Survey Questionnaire)
Refinement?
Yes
Refinement
No
User Evaluation Study
Result Analysis and Discussion
Data Analysis of Think-aloud Protocol
Data Analysis of Observation
Data Analysis of Questionnaire
Result Discussion

Figure 3.1: Flow of Research Activities

3.2 Literature Review

Reviewing research literatures helps the researcher to determine the research problems from the related works. A literature review is conducted to study the different approaches applied in EUSC. The approaches/techniques and features of FESC are analysed and presented in Chapter 2. The purpose is to inspect the problems and limitations of current approaches and techniques in order to define the research problems. The problem statement in Section 1.2 stated the main problems that this research aimed to resolve.

3.3 Identify Research Objectives and Questions

Based on the defined problem statement in Section 1.2, the researcher has identified the research objectives in Section 1.3 which are concrete statements of what the researcher is trying to achieve and done in this research in order to design and develop an approach that integrates application UI development and service composition in FESC. It helps to bring the focus of the research to its essential.Thereafter, the researcher identified the research questions in Section 1.4. Research questions are important in the research because they outline the uncertaint and concern points that needs to be investigated in order to rationalize the objectives of the research.

3.4 Development of Approach

In this step, a new approach was proposed and developed to address the conceptual and usability issues faced by end-user programmers in web services composition. The approach enables end-user programmers to compose applications from existing web services by configuring three different types of flow that represent the composition logic. The development of approach aims to provide an understanding of the proposed conceptual idea that describes how the work would be done and how the proposed approach addresses the conceptual and usability issues faced by end-user programmers in web services composition.

3.4.1 The Design of the Proposed Approach

This research proposed and developed an integrated three-flow approach (application flow, control flow and data flow) that enables the UI of the respective application and the composition of the web services required by the application to be constructed concurrently. Figure 3.2 illustrates the approach, including its processes and features involved that makes up the essential parts of the proposed approach. It explains the roles and importance of the processes and features. It also explains through the scope of the works, how they interrelate between each other within to produce a desired outcome. The outcome of the proposed approach aims to overcome the common conceptual and usability issues faced by end-user programmers in web services composition.

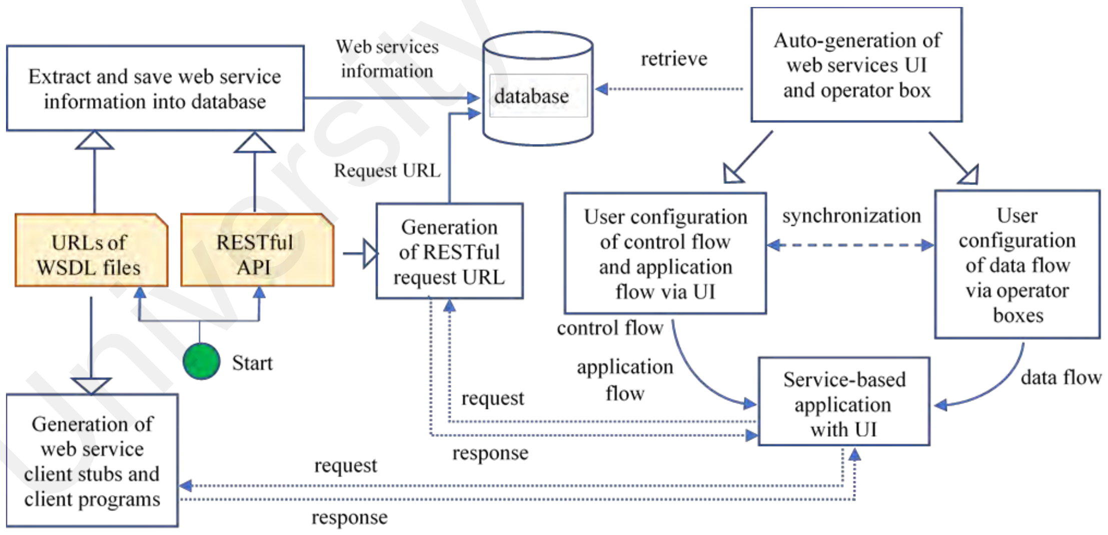

Extract and save web service information into database
Web services information
database
retrieve
Auto-generation of web services UI and operator box
Request URL
URLs of WSDL files
RESTful API
Generation of RESTful request URL
User configuration of control flow and application flow via UI
synchronization
User configuration of data flow via operator boxes
Start
control flow
application flow
request
Service-based application with UI
data flow
response
Generation of web service client stubs and client programs
request
response

Figure 3.2: Integrated Three-flow Approach

The proposed approach extracts web services information from the URLs of Web Service Description Language (WSDL) files and RESTful web service API. The extracted web services information from WSDL file and RESTful API is then saved into a local database. The WSDL file is used to generate the respective web services client stubs and web service client programs for the purpose of web services execution. These two processes are executed by Java servlets. A web service client stub acts as a remote procedure call that provides the entry point between a web service client program and the web service server (Microsystems, 2002). A web service client program is a remote client that contacts the web service and invokes the web service's methods (Microsystems, 2002) . RESTful API is used to generate the request URL for the purpose of web services execution. The request URL will be saved into database as well.

The approach retrieves web service information from the database and auto-generates the UI elements and operator boxes from the web services of end-users selected for the composition process. It allows the end-user programmers to explicitly configure the three flows - the data flow connecting web services via the operator boxes as the data mapping between these web services, the control flow or the execution order of the web services included in the composition, and the application flow that determines the order of transition of UI pages in a multi-page application. Three types of control flow patterns are provided - sequence (sequential control flow), merge (convergence of two or more services into a single subsequent service) and split (divergence of a service into two or more parallel services each of which executes concurrently). The composed application will be saved as HTML.

Control flow and data flow are the two essential types of composition constructs in web services composition (Lemos et al., 2015). The approach also includes application flow to specify the order of the applications UI pages for the composed application.

Control flow could be implied by the application flow in simple applications, but in more complex applications, web services might need to be executed concurrently, requiring control flow to be separated from the application flow.

The three flows are used as the composition logic for the web service composition process and are integrated into the UI of the composed web services. The composition logic is embedded in the web services' UI for execution purpose. Input data can be provided through the UI of the composed web services and the execution of the web services can be initiated. For SOAP web service, the web service client program will be invoked, the web service client stub will be sending a request to the web service server and subsequently returning the response from the server to the client program. For RESTful web service, the requested URL will be invoked and a response by HTTP client will be received and shown in the UI.

The approach uses web services' names that are meaningful to the end-user programmers instead of the services' technical names to help them in choosing the suitable web services for the composition. They would also be assisted in this aspect through the early visualization of the composed application UI during design time with the actual effects reflected instantly on the scene. The UI elements serves as concrete mediums for end-user programmers to design the composition logic of the application by using the three flows made visible in a multiple-view design (Section 5.3). The multiple-view design requires the display of information regarding the different views to be synchronized because despite their different aspects, they are presenting the same web services. Besides that, the data flow configured in one view must be synchronized to the data mapping between the web services appearing in the application UI in order to transfer the data between the web services correctly during runtime. Data transformation checking is done for data flow configuration to make sure the data format and data type of service output can be converted to the data format and data type of another service input.

In summary, the approach leverage on instant integration, visualization and synchronization of the application UI development and web service composition, to simplify the process of web service composition. Through concrete visualization of the web service composition and the application UI, as well as their synchronization, the complexity of composing and modifying assembled applications can be reduced to cater to end-user programmers who have no technical knowledge.

3.4.2 Conceptual and Usability Issues Addressed

The approach addresses some of the most common conceptual and usability issues faced by end-user programmers in web services composition (Cappiello et al., 2015; Namoun et al., 2010). In particular, the approach

(i) enables end-user programmers to gain basic understanding of web services composition.

(ii) eliminates the need to differentiate between design time and run/execution time as it allows end-user programmers to input data into the UI fields of the web services and to execute the services necessary for output during the design time.

(iii) reduces the difficulties that may be faced by end-user programmers in specifying the execution order of the web services and logic of application by providing an intuitive graphical click-drag-release mechanism when selecting the required web services, and specifying the control and application flow. It also allows end-user programmers to perform a simple two-click movement when specifying the data flow between two web services. This is done by first clicking on the output field of the first web service, and then clicking on an input field of the subsequent web service.

(iv) presents minimal technical terms to the end-user programmers where no other technical term is used apart from "web services" and "composed services".

(v) produces an organized visual layout of the UI to the selected web services so as to reduce end-user programmers' difficulties in positioning layouts of the web services.

(vi) provides instant visual updates to the end-user programmers' actions as a means to reduce their uncertainty in whether or not they have done it right.

(vii) an integrated model that allows the composition of SOAP and RESTful web services; data transformation between web services.

As mentioned in Section 1.3, the approach does not address the end-user programmers concerns for security issues or sensitive information submitted as input data for certain web services.

3.5 Development of Prototype

A proof-of-concept prototype, QuickWSC was developed based on the proposed approach. QuickWSC was built as a layered system with presentation, logic and data layers. Multiple-view design was adopted in QuickWSC's UI to incorporate the integrated three-flow approach. This research followed the design guidelines/rules for multiple-view selection, presentation and interaction (Baldonado et al., 2000), and applied the recommended techniques according to the design guidelines.

Table 3.1 shows the types of technology used in the development of the prototype along with its supporting reasons. The selected technology types are proven reliable and well documented.

Table 3.1: Technology Types Used in The Development of Prototype

Aspect	Technology Type	Reason
Database	Microsoft SQL Server (Microsoft)	Open source, Easy to manage
Back-end programming	Java Object-Oriented Programming (W3Schools, 2020)	Easy for troubleshooting, Flexibility through polymorphism
Front-end programming	HTML (W3Schools, 2020), CSS (W3Schools, 2020), JavaScript (W3Schools, 2020)	Platform independent

Table 3.1 continued

Aspect	Technology Type	Reason
Server	Apache Tomcat (T. A. S. Foundation, 2020)	Open source, Reliable, Update without restarting server
Development tool	Eclipse IDE (E. Foundation, 2020)	Easy to use, Easy for deployment
System platform	Web platform	Accessible through any web browser

3.6 Data Collection

Data collection is a process that uses standard and validated instruments for gathering data and evaluating the outcome. An integrated method which combines quality and quantity was employed for this process.

A user evaluation study was conducted in an effort to collect the data of prototype assessment done by participants. It employed qualitative data collection using think-aloud protocol and observation, followed by both quantitative and qualitative data collection using a questionnaire survey. Refer to Section 3.6.3 for further details.

3.6.1 Pilot Study of the User Evaluation Study

Prior to the actual user evaluation study, a pilot study was conducted to gather feedback as a means to refine the prototype and the user evaluation study instrument. The pilot study was conducted by the researcher with two participants at a computer lab where the prototype was deployed through a laptop. Feedbacks collected from these participants were used to improve the user evaluation study before conducting it with the rest.

The criteria in recruiting the participants were:

1. Not essentially equipped with well-trained programming knowledge, but to at least be able to use a computer proficiently.
2. Web service composition knowledge not being a necessity.

3.6.2 Data Collection Method

3.6.2.1 Think-aloud Protocol

Think-aloud protocol is a method that collects verbal translation from the subjects in whatever goes through their minds (Jääskeläinen, 2010). The subjects only need to verbalize what is in their mind when performing the required task, but they do not need to explain what they are doing. Participants in the pilot study were asked to use the prototype to compose web services based on a scenario given by the researcher. They were asked to articulate whatever they were seeing, feeling and doing while using the prototype to compose the web services. The process was video recorded with the consent of the participants for further analysis. A data collecting instrument was used to log the verbal transcriptions by participants for the think-aloud protocol.

3.6.2.2 Observation

Observation is a method where data is collected through observation of the subject when performing a task (Diah, Ismail, Ahmad, & Dahari, 2010). An overt observation signifies participants being aware that they are being observed (Anne, 2013). In this observation exercise, the participant was engaged in a think aloud protocol while using the prototype to compose web services. A one-to-one direct observation was carried out where the researcher observed the behaviour of the participant and any incidents that took place.

A structured checklist was prepared for the data collecting instrument to record pre-identified activities while the participants were composing these web services. Any incidents not listed in the checklist were still recorded in a field note.

3.6.2.3 Survey

A questionnaire is used as part of the user evaluation study. The questionnaire comprised of 2 parts (Part A and Part B). Part A questioned the participants' educational background, programming experience, and level of computer skills. Questions 1 to 12 of Part B were semi-structured questions screening the features of QuickWSC with a Likert scale of 1 to 5 (1-Strongly Disagree, 2-Disagree, 3-Neither Agree nor Disagree, 4-Agree, 5-Strongly Agree). The participants were asked to state their reasons or opinions if the response was 3 and below. Question 13 and Question 14 were open-ended questions encouraging opinions on the proposed approach and tools. The intention of the questionnaire survey was to evaluate the features of the prototype and the composition approach applied.

3.6.3 Refinement of User Evaluation Study

Necessary refinements were made after acquiring feedbacks from participants of the pilot study before conducting the actual user evaluation study.

3.6.4 User Evaluation Study

A user evaluation study was conducted with 20 end-user programmers that served as participants to evaluate the prototype and the underlying approach.

3.7 Result Analysis and Discussion

The results were summarized and discussed (Section 6.4). Both qualitative and quantitative analysis were performed on the data collected from the user evaluation study due to the use of a mixed method research.

3.7.1 Qualitative Data Analysis

Framework analysis approach (Pickup, Holloway, & Samsi, 2015; A. Srivastava & Thomson, 2009) was used for qualitative data analysis. The collected data was sifted, charted, and sorted according to the different key issues and themes. Framework analysis comprises the following five steps.

3.7.1.1 Familiarization

The videos recorded during the data collection process were watched repeatedly to familiarized and grasp the overview of the collected data. Familiarization helps the researcher realize the themes and issues that emerged within the data set which in this case, is the verbal responses from the participants during the think-aloud protocol as well as the incidents during the observation.

3.7.1.2 Identifying a Thematic Framework

This step was aimed to identify the potential categories offering the best fit for the data. The themes and issues were identified based on the data set observed by the observer and then devised into a thematic framework. This step helps classify and filter the data which requires logical thinking. This is because the observer needs to make a judgement on the identified themes and issues.

3.7.1.3 Indexing

Indexing is a step where allocation of the relevant data into the appropriate themes/subthemes is done. The incidents witnessed by the researcher during observation, and comments of the participants during the think-aloud protocol were categorized according to the different themes to form an initial thematic table.

3.7.1.4 Charting

Charting is a step to revise and finalize the themes, subthemes and related data. The data clearly states the source of response. A thematic table was finalized.

3.7.1.5 Mapping and Interpretation

This step provides a clear view of the event or phenomena surrounding the research field. The analysis summary is presented with significant explanation of the charted themes (mapping the data to the cause and interpreting the data sets).

3.7.2 Quantitative Data Analysis

Descriptive statistics were used to analyse the data and quantitatively summarize the data collection. According to Fisher and Marshall (2009), "Descriptive statistics are simply the numerical procedures or graphical techniques used to organise and describe the characteristics or factors of a given sample." (pg. 93).

3.7.3 Data Analysis of Think-aloud Protocol

The collected data was manually analysed by using a framework analysis approach. A thematic table was formed to summarise the analysis result.

3.7.4 Data Analysis of Observation

Qualitative and quantitative data analysis were used to analyse the data obtained from the observation exercise. Data collected from the structural checklist was analysed by using a frequency distribution method. The incidents that happened which were not listed in the structural checklist, were manually analysed by using framework analysis. Both of the collected data were combined and collectively analysed.

3.7.5 Data Analysis of Questionnaire

The data collected on Participants backgrounds in Part A of the questionnaire was used to obtain the sample population, whereas the data obtained from Part B of the questionnaire was transferred into Microsoft Excel for data analysis. The descriptive statistics (frequency distribution, mean, median, mode, standard deviation) was used to analyse and present the collected data for Question 1 to Question 12 from the questionnaire. For the open-ended Question 13 and Question 14, the responses were grouped into a few reasons by thematic analysis before calculation of frequency distribution.

3.7.6 Results Discussion

The purpose of having a results discussion was to discuss the findings of the user evaluation study based on the obtained results. The reasoning of the results in terms of usability and features of the prototype which incorporates the proposed approach was enveloped within the discussion. ISO 9241-11 standard was used to measure the usability of prototype in terms of its effectiveness, efficiency and satisfaction. Triangulation was also involved in the explanation of results by combining the qualitative and quantitative data to provide the confirmation of findings and comprehensive results. Triangulation was used in more than one particular approach during the research for the purpose of obtaining richer and fuller data in helping to confirm the research results (Wilson, 2014).

3.8 Summary

This chapter describes the research methodology for this research. The methodology of this research included six activities, starting with the literature review wherethe researcher reviewed the existing approach/techniques for EUSC and FESC in order to identify the research problems. Based on the produced statement of problems discovered, the researcher identified the objectives and research questions. Subsequently, anapproach was developed to introduce the designs of the proposed idea. Following that, a proof-of-concept prototype as a working model was developed based on the proposed approach and a combined qualitative and quantitative user evaluation study was designed to evaluate the prototype and its underlying approach. This research used three data collection methods (think-aloud protocol, observation and survey) to collect data for result analysis. Framework analysis approach was used to analyse the collected qualitative data, while descriptive statistics were used to analyse the collected quantitative data. Finally, the analysed data was used during the results discussion.

CHAPTER 4: DESIGN AND IMPLEMENTATION OF PROTOTYPE

A proof-of-concept prototype (QuickWSC) was developed based on the proposed approach. It allows end-user programmers to aggregate and execute web services to achieve results. This chapter explains the requirements of QuickWSC, the architecture design, UI design and implementation of QuickWSC. It also contains a section that details the design of the multiple-view and navigation flow support adopted.

4.1 Requirements of QuickWSC

The user requirements were identified based on the analysis of existing FESC tools and their limitations as discussed in Chapter 2, as well as the common conceptual and usability issues faced by the end-user programmers (Cappiello et al., 2015; Namoun et al., 2010). Figure 4.1 shows the use case diagram of QuickWSC. It illustrates the actors involved and use cases that represents the functional requirements of QuickWSC followed by the use case description.

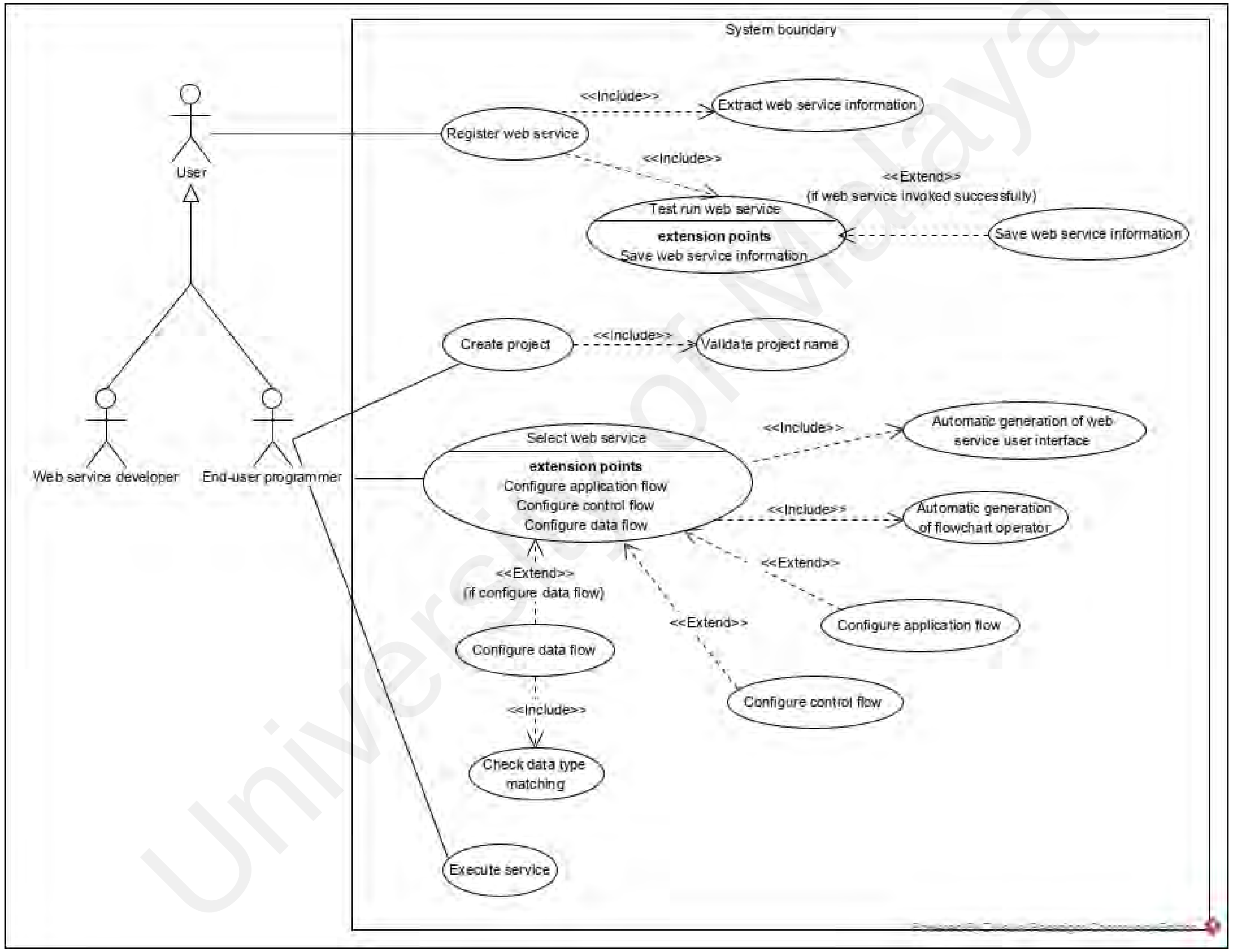
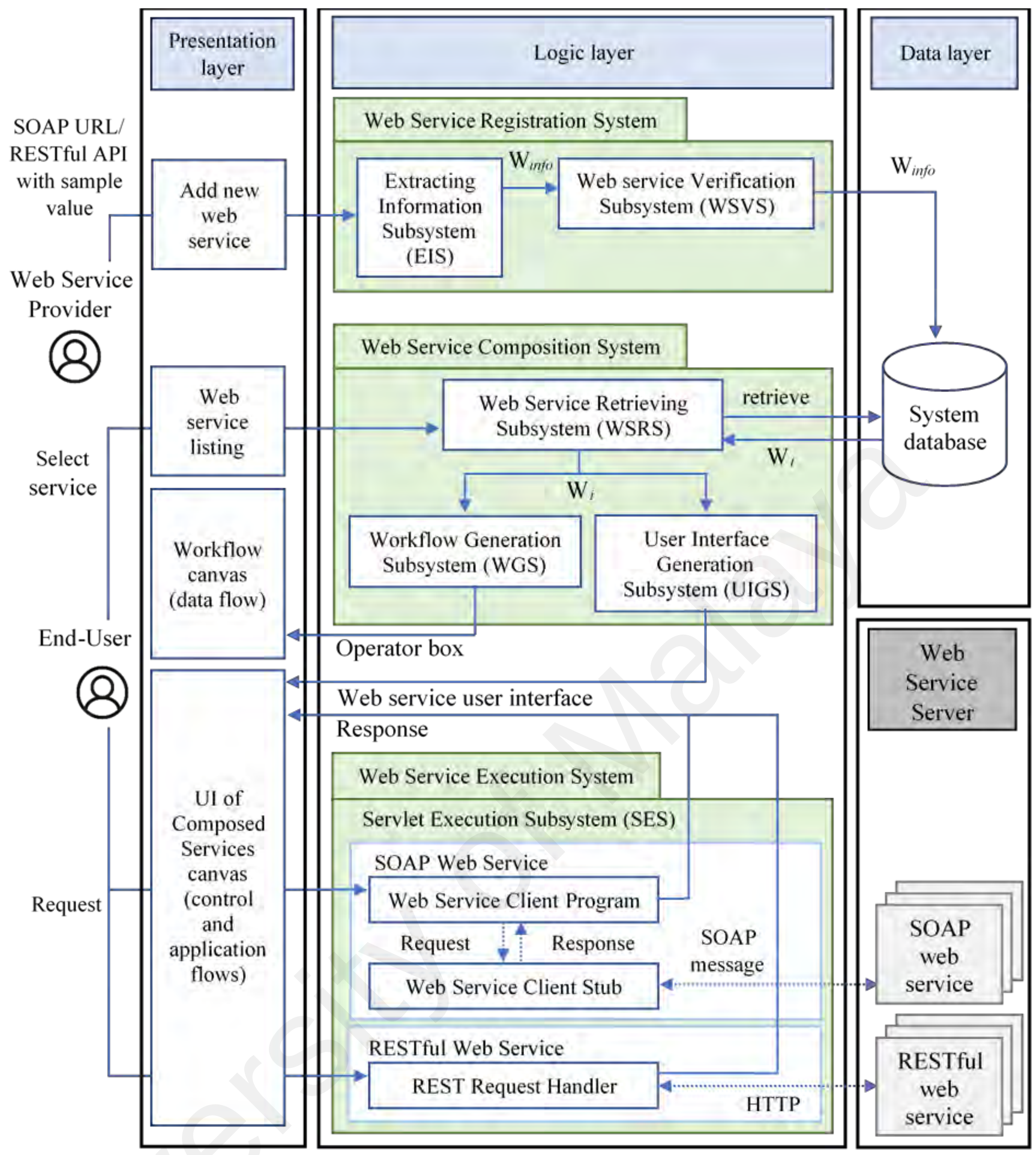


Figure 4.1: Use Case Diagram of QuickWSC

Use case 1: **Actor(s):** **Summary Description:**	Register web service User Allows all users to register the existing web services
Use case 2: **Actor(s):** **Summary Description:**	Extract web service information QuickWSC system QuickWSC system extracts the web service information.
Use case 3: **Actor(s):** **Summary Description:**	Test run web service QuickWSC system QuickWSC system tests the web service.
Use case 4: **Actor(s):** **Summary Description:**	Save web service information QuickWSC system QuickWSC system saves the web service information into its system database.
Use case 5: **Actor(s):** **Summary Description:**	Create project End-user programmer End-user programmer creates a project to compose web services.
Use case 6: **Actor(s):** **Summary Description:**	Validate project name QuickWSC system QuickWSC system checks the project name against the system database.
Use case 7: **Actor(s):** **Summary Description:**	Select web service End-user programmer End-user programmer selects a web services for composition.
Use case 8: **Actor(s):** **Summary Description:**	Automatic generation of web service user interface QuickWSC system QuickWSC system generates the UI of the selected web service.
Use case 9: **Actor(s):** **Summary Description:**	Automatic generation of flowchart operator QuickWSC system QuickWSC system generates the flowchart operator of the selected web service.
Use case 10: **Actor(s):** **Summary Description:**	Configure application flow End-user programmer Allows end-user programmer to configure the application flow.
Use case 11: **Actor(s):** **Summary Description:**	Configure control flow End-user programmer Allows end-user programmer to configure the control flow.
Use case12: **Actor(s):** **Summary Description:**	Configure data flow End-user programmer Allows end-user programmer to configure the data flow.
Use case 13: **Actor(s):** **Summary Description:**	Check data type matching QuickWSC system QuickWSC system checks the data type between the web services.
Use case 14: **Actor(s):** **Summary Description:**	Execute service End-user programmer Allows end-user programmer to execute services.

4.2 Architecture Design of QuickWSC

Figure 4.2 shows the architecture design of QuickWSC. The system consists of three main systems (*Web Service Registration System, Web Service Composition System* and *Executing System*). Web Service Registration System is used by the web service provider to add new web services into the system database. It consists of two subsystems, namely *Extracting Information Subsystem* and *Web service Verification Subsystem*. Web Service Composition System is responsible for the composition process and it comprises three subsystems (*Web Service Retrieving Subsystem*, *User Interface Generation Subsystem, Workflow Generation Subsystem*). Executing system runs the invocation process of the composed web services and it consists of *Servlet Execution Subsystem*. The functionality of the subsystems is described next.



Presentation layer
Logic layer
Data layer
SOAP URL/ RESTful API with sample value
Web Service Provider
Add new web service
Web Service Registration System
Extracting Information Subsystem (EIS)
W info
Web service Verification Subsystem (WSVS)
W info
Web Service Composition System
Select service
Web service listing
Web Service Retrieving Subsystem (WSRS)
retrieve
System database
W i
W i
Workflow canvas (data flow)
Workflow Generation Subsystem (WGS)
User Interface Generation Subsystem (UIGS)
End-User
Operator box
Web Service Server
Web service user interface
Response
Web Service Execution System
Servlet Execution Subsystem (SES)
SOAP Web Service
UI of Composed Services canvas (control and application flows)
Request
Web Service Client Program
Request
Response
SOAP message
Web Service Client Stub
SOAP web service
RESTful Web Service
REST Request Handler
HTTP
RESTful web service

Figure 4.2: System Architecture of QuickWSC

4.2.1 Extracting Information Subsystem (EIS)

EIS extracts the web services information provided by web service providers from SOAP URL and RESTful API. The extracted web service information is a set of object $W_{info} = \{wsop, wsdesc, wsinmsg, wsoutmsg, inelm, outelm\}$, where

wsop is the web service name;

wsdesc is the web service description;

wsinmsg is the web service input message;

wsoutmsg is the web service output message;

inelm is the web service input parameters and data type;

outelm is the web service output parameters and data type.

4.2.2 Web Service Verification Subsystem (WSVS)

WSVS will test run the web services to verify them. The test run uses the sample values given by the respective web service provider. The W_{info} will be saved into the system database if the web service can be invoked successfully.

4.2.3 Web Service Retrieving Subsystem (WSRS)

WSRS retrieves the respective web service information (W_i) from the system's database upon the end-user programmer's selection from the list of web services provided. The WSRS will return a set of object *Wi = {wsid, wsop, wsdesc, wsinmsg, wsoutmsg, inelm, outelm}*, where *wsop, wsdesc, wsinmsg, wsoutmsg, inelm* and *outelm* were described in Section 5.2.1 and *wsid* is the web service id in the system database. The web service object will be sent to *User Interface Generation subsystem* and *Workflow Generation subsystem*.

4.2.4 User Interface Generation Subsystem (UIGS)

UIGS generates the UI for the respective web service based on the web service object from WSRS. A UI element will be generated for each input data required by the web service based on the type of input data, as shown in Table 4.1. For example, if the input data type is String or numeric, a TextBox will be generated. The results or output of executing the web service will be shown in a table form. All the input UI elements and the output table will be enclosed in a container box and shown in the "UI of Composed

Services" canvas. The generated graphical UI serves as a concrete medium for the respective web service and provides a visualization of its input and output requirements. The whole composed services' UI is generated in HTML format and shown in the "UI of Composed Services" canvas.

Table 4.1: **UI Element Type Generation Logic**

Data type	**Element type**
String	Text
Numeric (int, float, decimal, double)	Text
Boolean	Radio
Range	Dropdown
Time	Time box
Datetime	Datetime box

4.2.5 Workflow Generation Subsystem (WGS)

WGS generates operator boxes for the respective web service based on the web service object acquired from WSRS. Each operator box represents a primitive web service and shows the names of its input and output parameters, alongside their data type in text format. Operator boxes were developed using jquery.flowchart (JavaScript Jquery plugin). The operator boxes are shown in the Workflow canvas.

4.2.6 Servlet Execution Subsystem (SES)

When executing any web service from the service UI, SES serves as a core component to communicate with the web service server in order to invoke the respective web service. The XML HttpRequest is used to transfer the data from a web browser to an application server. SES will send a SOAP message or call a HTTP command to the web service server for service invocation. SES will receive the response from web service server and send it to back to the web browser to display its results.

(a) SOAP Web Service

When the end-user programmer executes the SOAP web service, SES will execute the respective web service client program. The program will send a request to web service client stub to invoke the corresponding method (Figure 4.2). The runtime system of web service client stub then sends a SOAP message to web service server. When the web service receives the SOAP message, the runtime system of web service will execute the web service and send the response back to the web service client stub. Web service client stub extracts the SOAP message and sends the response to web service client program in the requested format. Client stub acts as a proxy between the client program and the web service. The system will create the web service client stub for every web service. Every method/operation in a SOAP web service will be created as an individual web service client program in the system.

(b) RESTful Web Service

When the end-user programmer executes a RESTful web service, REST request handler will receive the query parameter from web browser to form the invocation URL before sending it through HTTP request to the web service server. The runtime system of web service will return the response to SES.

4.3 User Interface Design of QuickWSC

Figure 4.3 shows the UI of QuickWSC which consists of three frames. The right frame is the web services listing. It shows the available web services in QuickWSC. The centre frame is the web service composition workplace ("User Interface of Composed Service" canvas) for end-user programmers to compose web services. End-user programmers are allowed to arrange the application/page flow and control flow within this canvas. The order of the numbers (1, 2, 3, 4) shows the application flow (namely, the sequence of UI pages of the composed services). End-user programmers can also arrange the web services into different vertical levels (for example, as labelled by 'A', 'B', 'C' in Figure 4.3) to configure the control flow of the composed services. The order of the alphabets (A, B, C) shows the control flow (namely, the sequence of execution of the constituent services). The left frame is the workflow workplace ("Workflow" canvas). It shows the operator boxes that represents the web services selected for the composition. An operator box shows the corresponding web service's input and output parameters alongside their data types. In this canvas, the End-user programmers can configure the data flow between the web services by connecting the output parameter/field of a web service to an input parameter of the second web service. This makes the output data from the first web service to flow as an input to the second web service, resembling a data flowchart. An example of the data flow of the service composition is shown by the blue connecting lines in Figure 4.3. The data flow is presented by connections among the flowchart operators while the control flow and application flow are presented by the order and arrangement of web service UI. In short, the UI of QuickWSC presents the three different flows (application flow, control flow and data flow) in an integrated view.

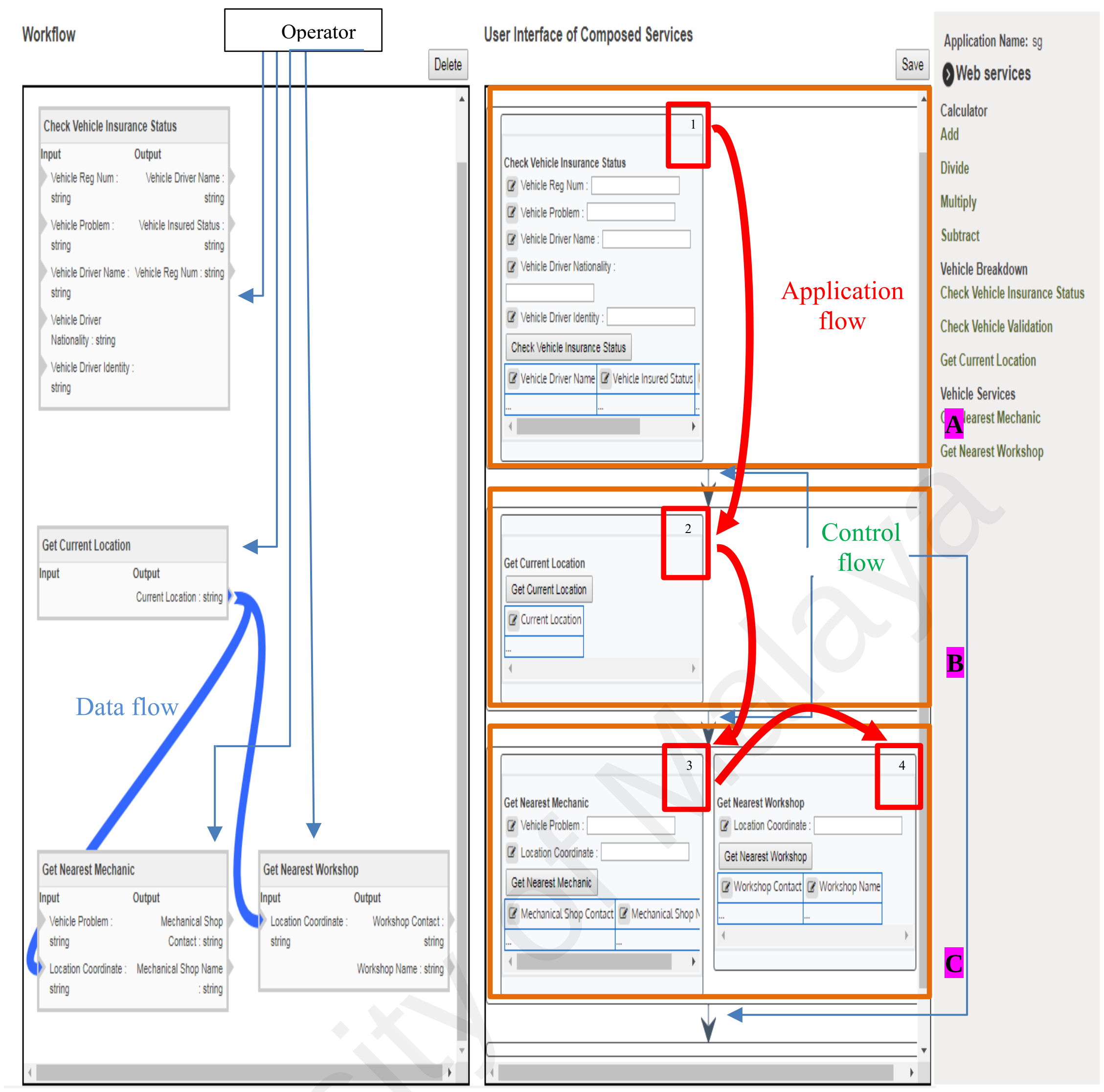


Figure 4.3: User Interface Design of QuickWSC

4.4 Multiple View and Navigation Flow Support Design

Apart from a multiple-view design, cascading view or single view could be used in the design of a EUSC/FESC system. Nevertheless, QuickWSC adopted a multiple-view design in its UI in order to incorporate the proposed integrated three-flow approach and provide justification in this section. CRUISe system also adopted a multiple-view design (Pietschmann, Voigt, Rümpel, et al., 2009).

Multiple-view design uses two or more different views to support the study of a single conceptual entity(Baldonado et al., 2000). The integrated three-flow approach calls for a diversity of views that are complimentary, thus fulfilling two guideline rules that advocates a multiple-view design (Baldonado et al., 2000). Multiple-view is applicable when there is a diversity of attributes, models, user profiles, levels of abstraction or genres (rule of diversity) (Baldonado et al., 2000). The approach allows end-user programmers to explicitly configure the three different types of flows required in web service compositions. Instead of cramping all the three flows into one single view, QuickWSC splits the configuration of data flow from the configuration of the control flow and application flow by putting the first into a separate canvas (Workflow canvas) and the latter two into another canvas known as "UI of Composed Services" canvas. This is done to reduce unnecessary information overload to the end-user programmers. Since control flow which chains the execution order of web services is more closely related to the application flow that depicts the order of pages transition, they are put in the same canvas. On the other hand, data flow that depicts which web services' output serves as which web services' input, is of a different genre, and is therefore captured in another canvas. The benefit of having the data flow in another canvas is particularly obvious when the same output of a web service serves also as the input to more than one web services.

According to the rule of complementarity, multiple views is applicable when different views bring out correlations and/or disparities (Baldonado et al., 2000). Having multiple views can help to show otherwise hidden relations. QuickWSC auto-generates a corresponding operator box for each web service selected for the composition in the Workflow canvas. An operator box shows the basic information of the respective web service such as its name, input and output parameters together with their data types. When using operator boxes to configure the data flow, end-user programmers will be able to identify compatible output data type to serve as input for another web service. When generating the operator box, QuickWSC also generates the UI elements of the respective web service in the "UI of Composed Services" canvas. The data flow connections which end-user programmers configured into the Workflow canvas shows the expected transfer of data between the web services when the end-user programmers execute the services, and this is not visible in the "UI of Composed Services" canvas, especially in the case of an output of a web service serving as input for multiple web services.

Having justified the use of a multiple-view design, the following explains the decisions made on QuickWSC's view presentation and interaction. QuickWSC follows four design guidelines/rules for these aspects (Baldonado et al., 2000). First, QuickWSC chooses to present the multiple views side-by-side (rule of space/time resource optimization) instead of sequentially, to save end-user programmers time in looking at the two views (canvases). . Understanding the relationships between views can be difficult for the user and perceptual cues can be used to make the relationships more obvious to the user (rule of self-evidence). Two perceptual cue techniques are applied in the design of QuickWSC to help user understand the relationships between the Workflow and the "UI of Composed Services" canvases: brushing and navigational slaving. Brushing technique depicts users highlighting or selecting items in one view and the system highlighting the corresponding items in another view. The brushing technique is applied in both canvases. When a user moves the mouse pointer over the UI of a web service in "UI of Composed Services" canvas, QuickWSC will highlight it and the respective operator box in the Workflow canvas with a green boundary, and vice versa (Figure 4.4). This helps the user to identify the corresponding entities between the two canvases.

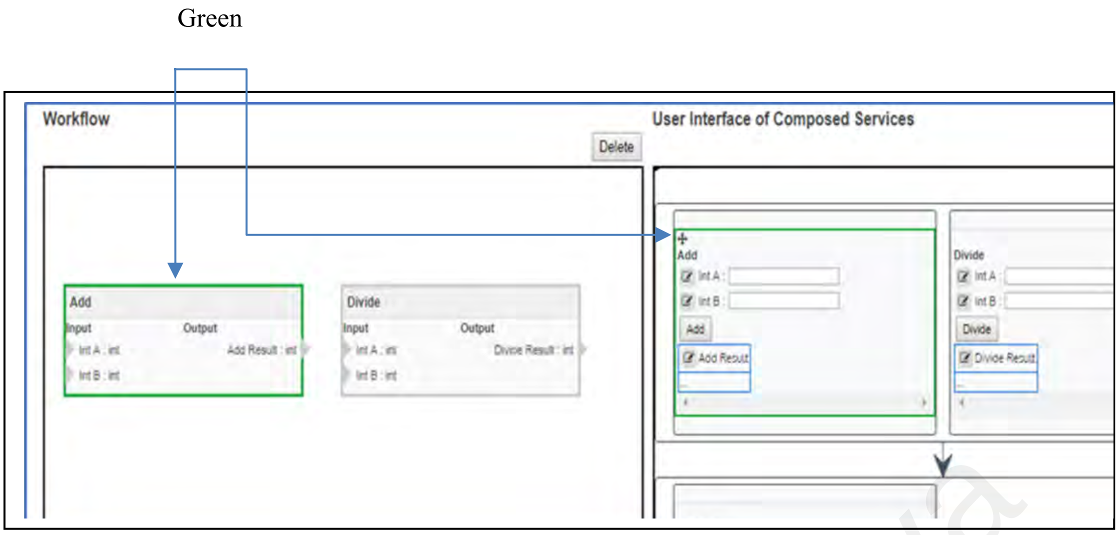


Figure 4.4: Highlighting Corresponding Items in the Two Views

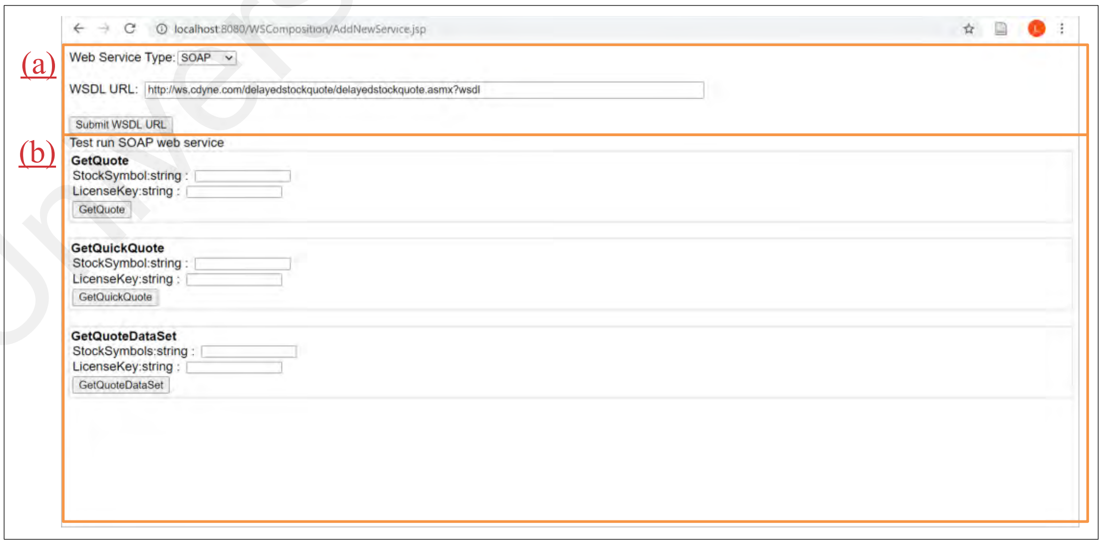
The navigational slaving technique refers to propagating movements in one view automatically to other views. The application of this technique in fact achieves the design required by the rule of consistency and will be explained in the following. Rule of consistency requires making the interfaces for multiple views consistent, and making the states of multiple views consistent. For example, if the objects or regions are shown/highlighted in one view, the corresponding objects or regions in the related view should also be shown. Consistent views facilitates learning and consistent states helps in object comparisons. QuickWSC applies the consistency rule to the Workflow and "UI of Composed Services" canvases. The operator boxes in the Workflow canvas and the respective UI elements of the corresponding web services in the second canvas are positioned at the same horizontal coordinate points (y-axis) in the two canvases. The navigational slaving technique is applied to the horizontal and vertical scrollbars of both canvases, resulting in the display of the corresponding regions in both the two canvases when the end-user programmer scrolls either one of the canvases using its scrollbars. Changes in "UI of Composed Services" canvas are reflected in Workflow canvas.

Rule of attention management is about using the perceptual techniques to focus the user's attention on the right view at the right time. It is a challenge to ensure the user's attention is at the right place at the right time when there are multiple views so that they are not distracted away from the view. QuickWSC uses colour highlighting technique for attention management, where it highlights the items of focus with a green boundary in both views (Figure 4.4).

4.5 Implementation of QuickWSC

4.5.1 Web Service Registration System

The user interface for web service registration was developed as a HTML page by using HTML, JavaScript and CSS. Figure 4.5 shows the web service registration user interface of SOAP web services. After submitting the WSDL URL as in Figure 4.5 (a), all the operations of the web services will be displayed as in Figure 4.5 (b) for test run by providing the sample values. The web service information will be saved into the database when all the operations are invoked successfully.



Web Service Type: SOAP

WSDL URL: http://ws.cdyne.com/delayedstockquote/delayedstockquote.asmx?wsdl

Submit WSDL URL

Test run SOAP web service

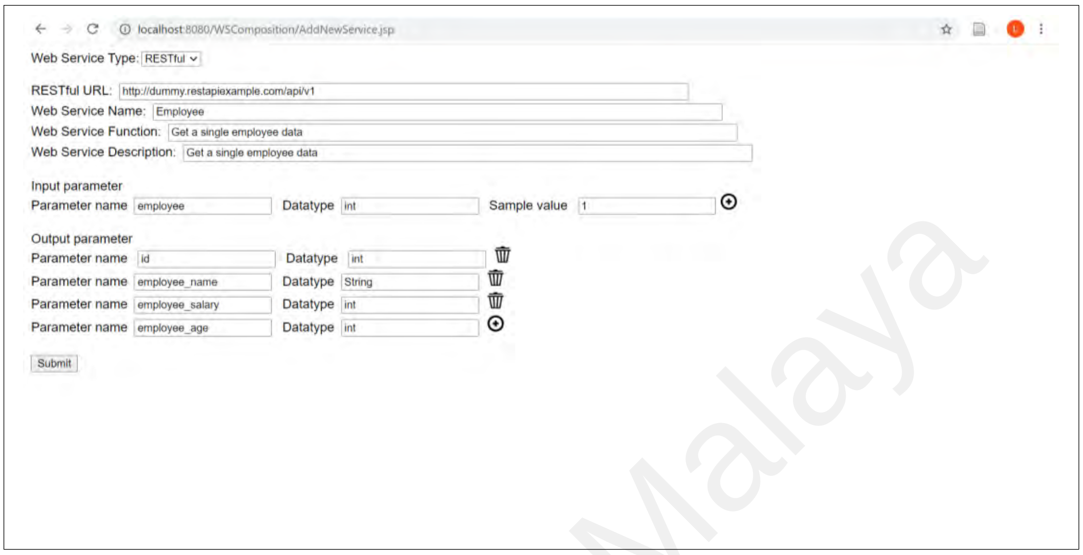
GetQuote
StockSymbol:string :
LicenseKey:string :
GetQuote

GetQuickQuote
StockSymbol:string :
LicenseKey:string :
GetQuickQuote

GetQuoteDataSet
StockSymbols:string :
LicenseKey:string :
GetQuoteDataSet

Figure 4.5: Web Service Registration User Interface of SOAP Web Services

Figure 4.6 shows the web service registration user interface for RESTful web services. The RESTful information and sample values must be provided. The web service information will be saved into database when it is invoked successfully.


localhost:8080/WSComposition/AddNewService.jsp

Web Service Type: RESTful

RESTful URL: http://dummy.restapiexample.com/api/v1
Web Service Name: Employee
Web Service Function: Get a single employee data
Web Service Description: Get a single employee data

Input parameter
Parameter name employee Datatype int Sample value 1

Output parameter
Parameter name id Datatype int
Parameter name employee_name Datatype String
Parameter name employee_salary Datatype int
Parameter name employee_age Datatype int

Figure 4.6: Web Service Registration User Interface of RESTful Web Services

Web service registration system was developed using Java object-oriented programming. Figure 4.7 shows the class diagram for web service registration. It consists of SOAP and RESTful web service registration and verification.

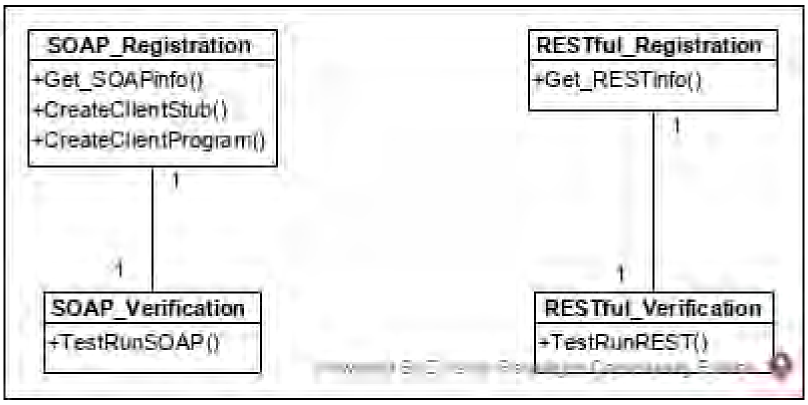
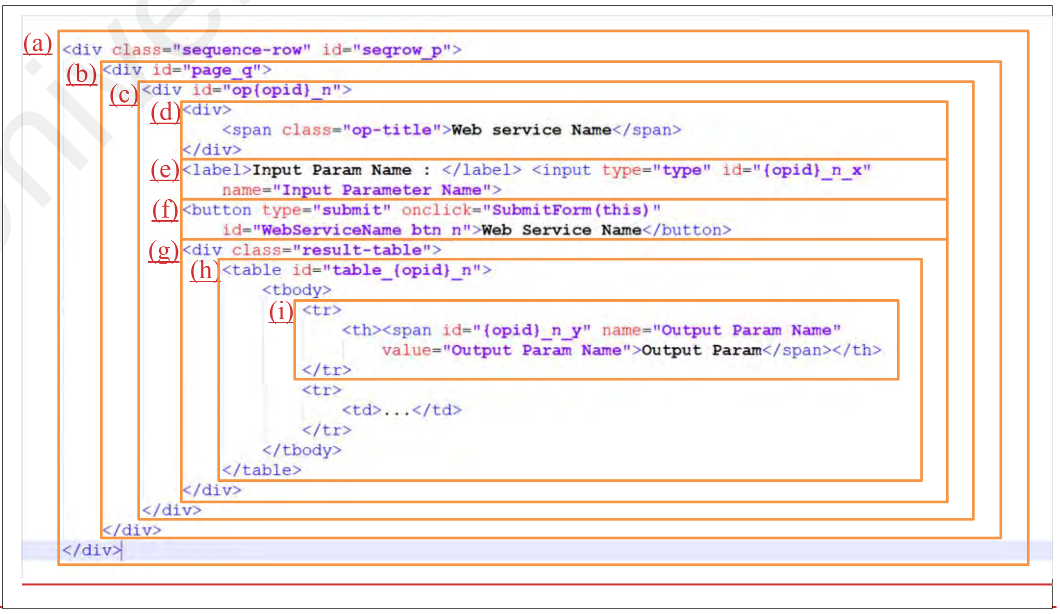


Figure 4.7: Class Diagram for Web Service Registration

4.5.2 Web Service User Interface Generation

The web service User Interface is created by using HTML. Figure 4.8 shows the HTML structure of web service User Interface. Figure 4.8 (a) is a ‘row’ container that can arrange multiple pages to run concurrently. The container has an id pattern of ‘seqrow_p’ where ‘seqrow’ is the syntax of row container and *p* is the increment number for the row div. Figure 4.8 (b) is a ‘page’ container which can comprise of multiple web services with the id pattern of ‘page_q’, where ‘page’ is the syntax of page and *q* is the increment number for page. Figure 4.8 (c) is a container structure for single web service. Each of the web service User Interface consists of the following: Figure 4.8 (d) the web service name, Figure 4.8 (e) input parameter, Figure 4.8(f) submission button and Figure 4.8(g) results table. The syntax of *op* represents the web service User Interface, *{opid}* is the web service id in the database and *n* is the increment number of duplicate web service. The input parameter structure Figure 4.8 (e) is looped or repeated by the number of input parameter, *x*. The output of the web service is arranged in table form Figure 4.8 (h) and the output parameter structure Figure 4.8 (i) is looped by the number of output parameter, *y*.



```
<div class="sequence-row" id="seqrow_p">
<div id="page_q">
<div id="op{opid}_n">
<div>
    <span class="op-title">Web service Name</span>
</div>
<label>Input Param Name : </label> <input type="type" id="{opid}_n_x"
    name="Input Parameter Name">
<button type="submit" onclick="SubmitForm(this)"
    id="WebServiceName btn n">Web Service Name</button>
<div class="result-table">
<table id="table_{opid}_n">
    <tbody>
        <tr>
            <th><span id="{opid}_n_y" name="Output Param Name"
                value="Output Param Name">Output Param</span></th>
        </tr>
        <tr>
            <td>...</td>
        </tr>
    </tbody>
</table>
</div>
</div>
</div>
</div>
```

Figure 4.8: HTML Structure of Web Service User Interface

4.5.3 Workflow Generation

jquery.flowchart was used to generate the operator in "Workflow" canvas which represents a primitive web service selected by the end-user programmers. The operator is used to configure the data flow. Figure 4.9 shows the flowchart structure including the operator, title, input connector, output connector and data flow link.

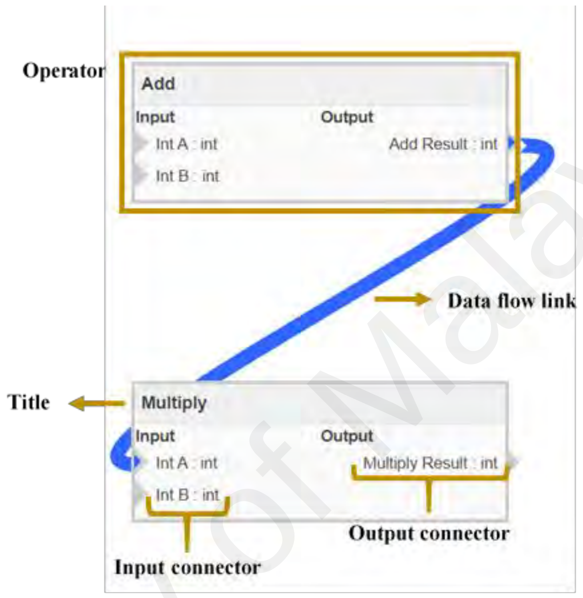
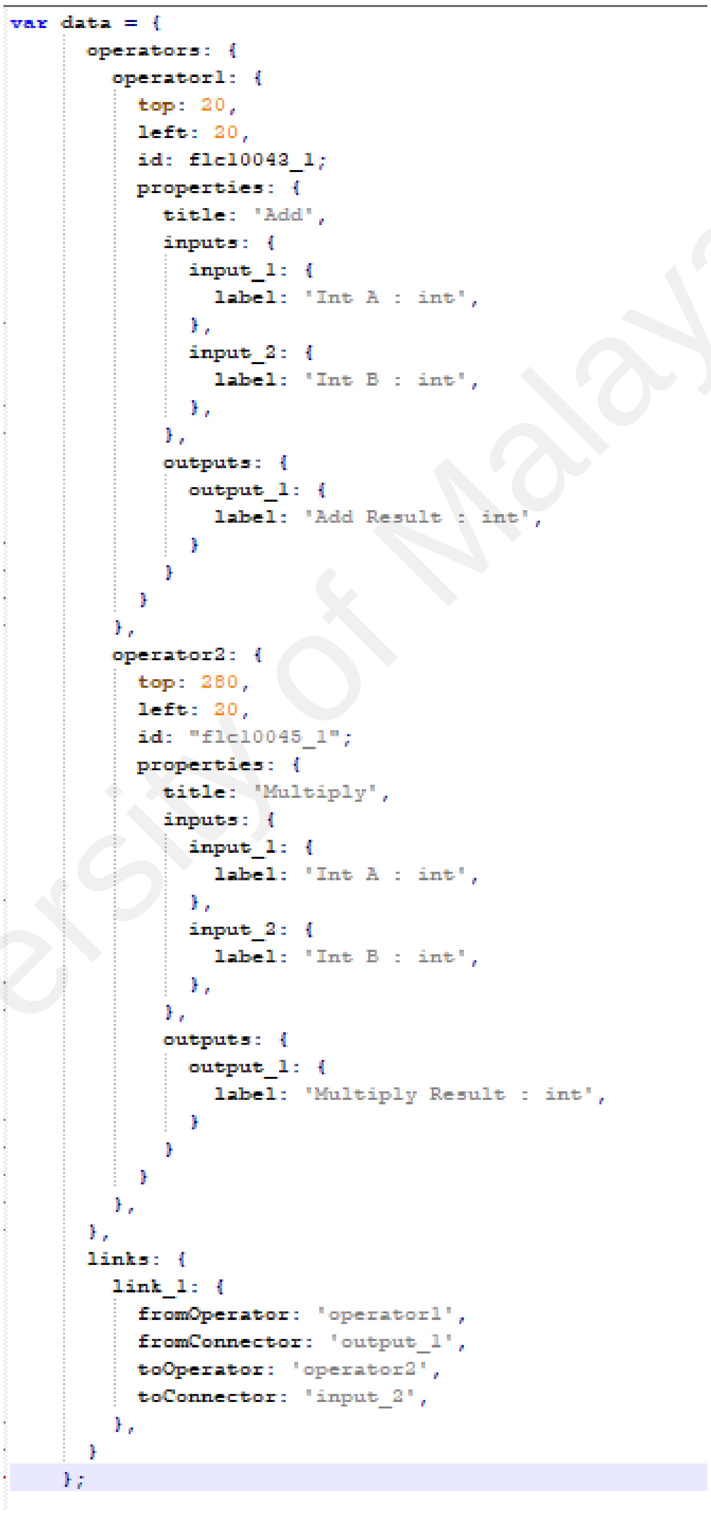


Figure 4.9: Flowchart Structure

The flowchart information was saved as a variable 'data' by using json object. The 'data' consists of two main objects which are operators and links. Figure 4.10 shows the sample of variable 'data' and its information with corresponding values. The 'top' and 'left' are the positions of operators from the left edge in the unit of pixel. The 'id' is the unique identity that represents the operators in the flowchart. The id is created by a pattern 'flc{opid}_n', where flc represents the corresponding flowchart operator, *{opid}* is a five digit web service id retrieved from database and n is the increment number for the redundant web service. The 'properties' comprises the 'title' of operator (name of web service) , and 'inputs' and 'outputs' of operator (parameter name and data type). The connected "links" are saved with four informations. The 'fromOperator' refers to the link connecting from which operator, ‘fromConnector’ refers to the link connecting from which output connector, ‘toOperator’ refers to the link connecting to which operator and ‘toConnector’ refers to the link connecting to which input connector.



```
var data = {
      operators: {
        operator1: {
          top: 20,
          left: 20,
          id: flc10043_1;
          properties: {
            title: 'Add',
            inputs: {
              input_1: {
                label: 'Int A : int',
              },
              input_2: {
                label: 'Int B : int',
              },
            },
            outputs: {
              output_1: {
                label: 'Add Result : int',
              }
            }
          }
        },
        operator2: {
          top: 280,
          left: 20,
          id: "flc10045_1";
          properties: {
            title: 'Multiply',
            inputs: {
              input_1: {
                label: 'Int A : int',
              },
              input_2: {
                label: 'Int B : int',
              },
            },
            outputs: {
              output_1: {
                label: 'Multiply Result : int',
              }
            }
          }
        },
      },
      links: {
        link_1: {
          fromOperator: 'operator1',
          fromConnector: 'output_1',
          toOperator: 'operator2',
          toConnector: 'input_2',
        },
      }
    };
```

Figure 4.10: Sample Data in Flowchart

4.6 Summary

This chapter presents the requirements of the prototype (QuickWSC), the architecture QuickWSC that describes the functionalities and responsibilites of the three main systems (Web Service Registration, Web Service Composition and Web Service Execution) and their subsystems. Following that, the User Interface design of the QuickWSC was introduced to facilitate the interaction between the end-user programmers and prototype system. Besides that, multiple view design was used to design the application user interface. The use of multiple view design was supported by the guidelines with justification. Finally, the details of implementation was explained.

CHAPTER 5: USER EVALUATION STUDY

This chapter presents the user evaluation study conducted to evaluate the prototype and the approach. It also describes the results collected from the pilot study which was conducted prior to the user evaluation study. Feedback from pilot study was used to refine the prototype and design of the user evaluation study.

5.1 Pilot Study Design and Results

In the pilot study, participants were asked to use QuickWSC to build an application by aggregating relevant web services based on a predefined scenario.

5.1.1 Predefined Scenario

The predefined scenario (that requires the use of four web services) is as below.

"A man, while traveling between two cities, suffers from an electrical breakdown in his car. He turns on the QuickWSC client installed on his PDA/laptop (alternatively, the driver could make a call to an operator at the control center who will use the application on the caller's behalf). He then enters the registration number of his vehicle, problem of vehicle and driver information to **Vehicle Insurance service** to check the insurance status. Upon getting the confirmation from the **Insurance service**, he sends the location as well as the problem from which the car has been suffering via **Mechanic service** to find out the nearest mechanic to the car's location who is capable to fix the stated problem i.e. electrical breakdown. He can search for the nearest workshop's address via the **Workshop service** by providing the location, in case the mechanic could not handle the breakdown and might need to tow-away the car to a workshop."

5.1.2 Pilot Study Procedure

The following is the pilot study procedure. It was estimated that a participant would take about 30 to 45 minutes to complete the pilot study.

1) A brief introduction to the research was shown to the participant (as shown in Appendix A), comprising mainly of the research purpose, terms and conditions of participation, and researchers' contact details.
2) Participants were given a manual user guide that describes the main functionalities of QuickWSC.
3) Participants were given a set of tasks to be performed, namely, use QuickWSC to build or compose an application by aggregating relevant web services based on the given scenario, to execute the composed services, to check on the composition results, and to state the start and end time of building the application. While composing the application, participants were required to engage in *think-aloud protocol* which required them to say whatever came into their mind as they were building the application. This might include what they were seeing, thinking, doing, and feeling.
4) Direct observation was also carried out during the composition process. The sessions were video-recorded (with participants' permission) for further analysis. Participants were asked to write the start and end time of building the composed application in Section 1 of the user evaluation study instrument (as shown in Appendix B).
5) After completing the task of composing the application based on the scenario, the participants were also asked to complete a short questionnaire. Section 2 Part A of the questionnaire asks about participants' educational background, programming experience, and level of computer skills. Section 2 Part B of questionnaire asks

participants' opinions on the features of QuickWSC and usefulness of three flows configurations for service composition (as shown in Appendix B).

5.1.3 Pilot Study Result

An acquaintance from Faculty of Science, University of Malaya helped to disseminate the information about the recruitment of participants for the study and to obtain the contact details of students who has volunteered to take part in the study. The researcher contacted the students and arranged for individual evaluation session.

Two students from the Faculty of Science, University of Malaya, participated in the pilot study. The results are summarized below: Both of the participants were able to compose the services based on the given scenario. One took 13 minutes to compose the services. The other took 10 minutes. From the observation, it was noticed that the participants knew what they were supposed to do but they were not sure how to do it. For example, the participants knew that they had to drag the service from the web services listing frame and drop it in the "UI of Composed Services" canvas, but they verbally confirmed with the researcher regarding the dropping position. Besides that, they knew that they had to specify the data flow, but they did not know how to create the required connections in the Workflow canvas. One of the participants had suggested to synchronize the data flow between the two canvases.

5.2 Refinement to Pilot Study Design

Based on the findings from the pilot study, the following refinements were made. The rest of the study designs remained the same for user evaluation study.

1) Refine QuickWSC

   The refinement is in the form of synchronizing the data flow between the two canvases ("UI of composed service" canvas and "Workflow" canvas).

2) Change the printed user guide to a screencast user guide video

   The screencast user guide video shows how to use the tool, focusing mainly on conveying the idea of the three flows and how to configure them.

5.3 User Evaluation Study

5.3.1 User Evaluation Study Procedure

The user evaluation study procedure is corresponding to the pilot study procedure (Section 6.1.2), except for step 2 where the participant was given a video user guide that describes the main functionalities of QuickWSC.

5.4 Results of User Evaluation Study

This section presents the background of participants, results of observation and the opinions of the participants on the features of QuickWSC obtained from the questionnaire.

5.4.1 Participants Background

Twenty students from Faculty of Science, University of Malaya were recruited to take part in the user evaluation study. There were two master students and eighteen undergraduate students. The average participation time was 20 minutes per participant. The participation took place at the Human Computer Interaction Lab, Faculty of Computer Science and Information Technology, University of Malaya.

The participants' levels of programming and computer skills are as below: Majority of the participants (fifteen) have not learned or had any prior knowledge in programming.

Twelve of these participants possessed basic computer skills such as Internet, email, hardware, software concepts, word processing, formatting, presentations, graphics, multimedia, and spreadsheets. The remaining three had intermediate computer skills including Internet, email, hardware, software concepts, terminology, word processing, formatting, tables, presentations, graphics, multimedia, spreadsheets, and databases. One participant acquired some self-learnt Python, C++, R and Arduino, but the participant told the researcher that he could not write a complete program with the languages. Another four participants also indicated that they have learnt programming on their own and could write simple programs. However, when further probed further by the researcher, they mentioned that they attended a one-day programming class during their matriculation study where codes were given to them by the tutor, and they were able to execute the codes but did not learn about programming theory. Three of these four participants have basic computer skills and one have intermediate computer skills.

5.4.2 Think-aloud Protocol Results

Appendix D shows the indexing table of data collected during the think-aloud protocol. Table 5.1 shows the thematic table obtained from analysis done on data collected from think-aloud protocol. Three participants were concerned about familiarity with the prototype system. They felt that the users need to have knowledge on and familiarity with the system in order to use it. One participant doubted when placing the first web service in the UI of composed service canvas. One participant was confused with the use of operators, and asked the researcher if ‘the operator represent a status of service?’. Three participants showed a lack of understanding on data flow. Two of them asked the researcher on how to configure data flow while one of them thought input parameter of one service can be connected to the input parameter of another service. There is only one participant who wondered whether she was supposed to key in the input before executing the service. Seven out of 20 participants mentioned the problems they faced when using the prototype, while the rest were silent and used the prototype without any difficulty.

Table 5.1: Thematic Table for Data Collected from Think-aloud Protocol

Themes	Subthemes	Responses
System understanding	System unfamiliarity	P1, P3, P7
Service selection	Doubtful on service placing	P7
Data flow configuration	Confuse with the usage of operator	P8
	Lack of data flow understanding	P10, P11, P20
Service execution	Doubtful on how to key in input data	P1

Note: “P” refers to a participant

5.4.3 Observation Results

Table 5.2 shows the checklist table used during the observation session to record whether the participants performed the pre-identified activities, and Table 5.3 shows the themes obtained from analysis done on data collected from the observation. Appendix E shows the indexing table of data collected by the researcher during observation.

Table 5.2: Checklist Table of Pre-identified Activities for Observation

	Participants																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Service Selection																				
Select vehicle insurance service	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Select current location service	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Select nearest mechanic service	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Select nearest workshop service	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Select non related service	1	1	1	1	-	-	1	-	-	-	-	-	1	1	-	1	1	1	-	-
Delete non related service	-	-	1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
Application flow configuration																				
Configure application flow	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Control flow configuration																				
Configure control flow	1	1	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	1	1	1
Data flow configuration																				
Configure data flow 1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
Comfigure data flow 2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Service compositon																				
Successfully composed the services	1	1	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	1	1	1
Service Execution																				
Input the data	1	1	1	1	-	1	1	-	1	1	-	1	1	-	1	1	1	1	1	1
Execute the composed services and get the results	1	1	1	1	-	1	1	-	1	1	-	1	1	-	1	1	1	1	1	1

"1" : performed successfully; "0" : performed but failed ; "-" : did not perform

Table 5.3: Thematic Table for Data Collected from Observation

Themes	Subthemes	Responses
Service selection	Doubtful on the functionality of web service	P3, P7
Application flow configuration	Doubtful on placing a new web service	P3, P7
Data flow configuration	Misunderstanding on how to connect	P1, P4, P16, P17
	Ask for configuration confirmation	P12
	Lack of data flow understanding	P10, P11
Service execution	Hesitation on method invocation	P3, P16, P17

Observation results for choosing web services from the web services listing frame: After reading the given scenario, majority of the participants (eighteen) were able to choose the four required web services from the web services listing frame without any difficulty. Two participants asked the researcher about the functionality of one of the web services in the web service list before they selected the web service. Despite that, these two participants were able to choose the correct web services from the web services listing frame (as shown in Table 5.2). Half of the participants (ten) selected one extra web service which was not required by the scenario. Three of them realised that they have chosen a non-related web service and deleted it. Nevertheless, the non-related web service did not affect the service composition results. In summary, the twenty participants had all chosen the four required web service successfully.

Observation results for application flow configuration: All of the participants were able to configure the application flow. Two of the participants tried to drop a new service into a UI of composed service canvas, attempting to add the service but had failed to do so. However, they eventually found the way to add a new service into the UI of composed service canvas on the second attempt.

Observation results for control flow configuration: Majority of the participants (sixteen) were able to configure the control flow. Four of the participants failed to configure the control flow.

Observation results for data flow configuration: Eighteen of the participants were able to configure the data flow. Two participants were not able to configure data flow. They asked the researcher (who was also the observer) on how to configure data flow during the composition session. Four participants made a few attempts to configure the data flow due to misunderstanding on how to connect. They tried connecting the data flow by dragging instead of clicking.

Observation results for service composition: Majority of the participants (sixteen) successfully composed the services based on the given scenario, by configuring the three flows. Four participants failed in composing the web services. They did not manage to configure the control flow, but were able to configure the application and data flow.

Observation results for service execution: From the sixteen participants who successfully composed the services, all of them were able to key in the input for all the services, execute the composed application, and get the results.

5.4.4 Questionnaire Results

Figure 5.1 shows the distribution of the participants' opinion on the ease of QuickWSC feature usage which was obtained from questions 1-6 of the questionnaire. Figure 5.2 shows distribution of the participants' opinion on the visualization and execution features of QuickWSC which was obtained from questions 7-12 of the questionnaire. Table 5.4 shows the mean, median, mode and standard deviation (SD) for questions 1-12 of the questionnaire.

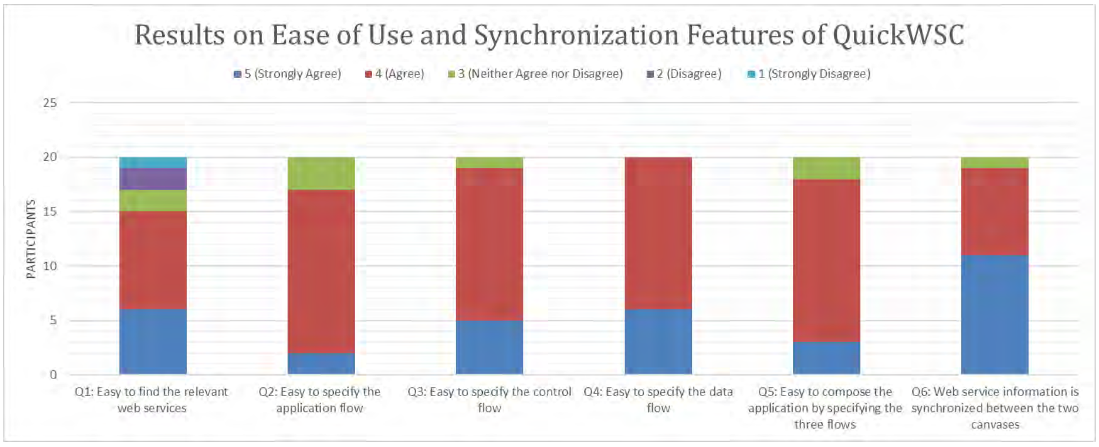


Figure 5.1: Results on Ease of Use and Synchronization Features of QuickWSC

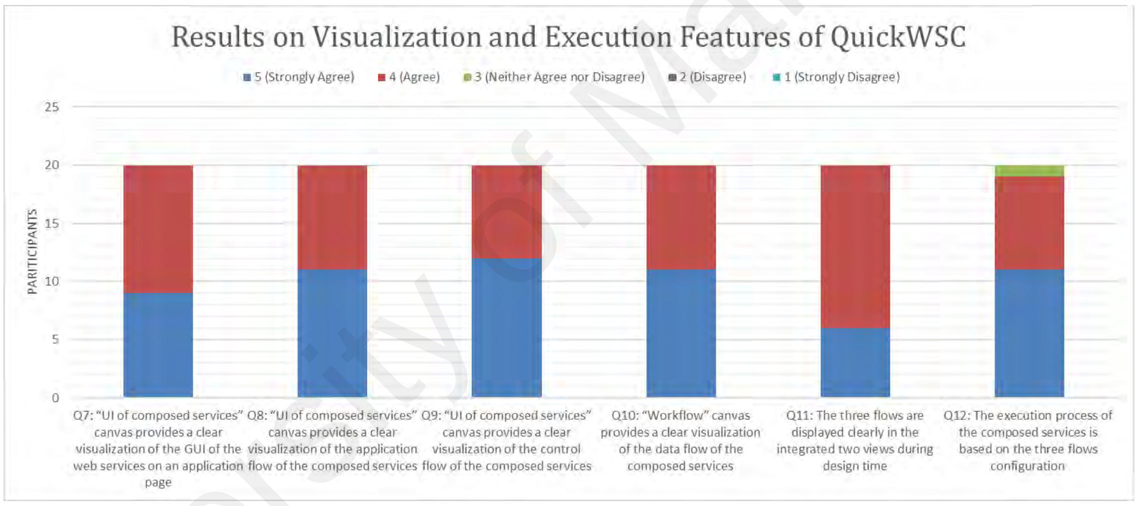


Figure 5.2: Results on Visualization and Execution Features of QuickWSC

Table 5.4: Descriptive Statistics for Question 1 to Question 12 of the Questionnaire

	Mean	Median	Mode	SD
Question 1	3.85	4	4	1.14
Question 2	3.95	4	4	0.51
Question 3	4.2	4	4	0.52
Question 4	4.3	4	4	0.47
Question 5	4.05	4	4	0.51
Question 6	4.5	5	5	0.61
Question 7	4.45	4	4	0.51
Question 8	4.55	5	5	0.51
Question 9	4.6	5	5	0.50
Question 10	4.55	5	5	0.51
Question 11	4.3	4	4	0.47
Question 12	4.5	5	5	0.61

Question 1 states "It is easy to find the relevant web services for the scenario. If your response is 3 and below, please state the obstacles of finding the relevant web services." Fifteen (75%) participants agreed or strongly agreed that it was easy to find the relevant web services in QuickWSC. Two (10%) participants were undecided on the ease of finding the relevant web services while three (15%) participants disagreed or strongly disagreed that it was easy to find the relevant services. The mean is 3.85 (SD=1.14) which falls between 'Neither Agree nor Disagree' and 'Agree', and has a wider spread around the mean (2.71-4.99) shows that some participants faced some problems when finding the relevant services. Median 4 shows that more than half of the participants rated 'Agree' or 'Strongly Agree' on the ease to find the relevant web services in QuickWSC. Similarly, mode 4 also shows that majority of participants rated 'Agree' on the ease to find the relevant web services in QuickWSC.

Question 2 states "It is easy to specify the application flow in the "UI of composed services" canvas. If your response is 3 and below, please state why it is not easy to specify the application flow in the respective canvas." Seventeen (85%) participants agreed or strongly agreed that it was easy to specify the application flow. Three (15%) participants were undecided on this aspect. The mean is 3.95 (low SD=0.51) which falls between ‘Neither Agree nor Disagree’ and ‘Agree’, it shows that a few participants faced some problems when configuring the application flow. Median 4 shows that more than half of the participants rated ‘Agree’ or ‘Strongly Agree’ on the ease to specify the application flow in QuickWSC. Similarly, mode 4 also shows that majority of participants rated ‘Agree’ on the ease to specify the application flow in QuickWSC.

Question 3 states “It is easy to specify the control flow in the “UI of composed services” canvas. If your response is 3 and below, please state why it is not easy to specify the control flow in the respective canvas.”. All except one of the participants agreed or strongly agreed that it was easy to specify the control flow. One participant was undecided on this aspect. The mean is 4.2 (low SD=0.52) which falls between ‘Agree’ and ‘Strongly Agree’, it shows that it was easy to specify the control flow. Median 4 shows that more than half of the participants rated ‘Agree’ or ‘Strongly Agree’ on the ease to configure the control flow in QuickWSC. Similarly, mode 4 also shows that majority of participants rated ‘Agree’ on the ease to configure the control flow in QuickWSC.

Question 4 states “It is easy to specify the data flow in the “workflow” canvas. If your response is 3 and below, please state why it is not easy to specify the data flow in the respective canvas.” All participants agreed or strongly agreed that it was easy to specify the data flow. The mean is 4.3 (low SD=0.47) which falls between ‘Agree’ and ‘Strongly Agree’, it shows that it was easy to specify the data flow. Median 4 shows that more than half of the participants rated ‘Agree’ or ‘Strongly Agree’ on the ease to configure the data flow in QuickWSC. Similarly, mode 4 also shows that majority of participants rated ‘Agree’ on the ease to configure the data flow in QuickWSC.

Question 5 states “It is easy to compose the application by specifying the three flows (application flow, control flow and data flow) in the integrated two views (workflow canvas and UI of composed services canvas). If your response is 3 and below, please state why it is not easy to compose the application by specifying the three flows in the integrated two views.”. Eighteen (90%) participants agreed or strongly agreed that it was easy to compose the application by specifying the three flows (application flow, control flow and data flow) in the integrated two views of Workflow canvas and “UI of Composed Services” canvas. Two (10%) participants were undecided on this aspect. The mean is 4.05 (low SD=0.51) which falls between ‘Agree’ and ‘Strongly Agree’, it shows that it was easy to compose the application by specifying the three flows (application flow, control flow and data flow) in the integrated two views (workflow canvas and UI of composed services canvas). Median 4 shows that more than half of the participants rated ‘Agree’ or ‘Strongly Agree’ on the ease to compose the application by specifying the three flows in the integrated two views. Similarly, mode 4 also shows that majority of participants rated ‘Agree’ on the ease to compose the application by specifying the three flows in the integrated two views.

Question 6 states “The web service information is synchronized between the “workflow” canvas and “UI of composed services” canvas. If your response is 3 and below, please state what web service information is not synchronized between the two canvases.”. Nineteen (95%) participants agreed and strongly agreed that web service information was synchronized between the Workflow canvas and “UI of Composed Services” canvas. Only one (5%) participant was undecided on this aspect. The mean is 4.5 (low SD=0.61) which falls between ‘Agree’ and ‘Strongly Agree’, it shows that web service information is synchronized between the “workflow” canvas and “UI of composed services” canvas. Median 5 shows that more than half of the participants rated ‘Strongly Agree’ on the synchronization of web service information between the two views. Similarly, mode 5 also shows that majority of participants rated ‘Strongly Agree’ on the synchronization of web service information between the two views.

Question 7 states ""UI of composed services" canvas provides a clear visualization of GUI of the web services on an application page. If your response is 3 and below, please state why the "UI of composed services" canvas does not provide a clear visualization of the GUI on an application page.". All the participants agreed or strongly agreed that the "UI of Composed Services" canvas provides a clear visualization of the GUI of the web services on an application page. All participants agreed or strongly agreed that "UI of composed services" canvas provides a clear visualization of the GUI of the web services on an application page. The mean is 4.45 (low SD=0.51) which falls between 'Agree' and 'Strongly Agree', it shows that "UI of composed services" canvas provides a clear visualization of the GUI of the web services on an application page. Median 4 shows that more than half of the participants rated 'Agree' or 'Strongly Agree' on "UI of composed services" canvas provides a clear visualization of the GUI of the web services. Similarly, mode 4 also shows that majority of participants rated 'Agree' on "UI of composed services" canvas provides a clear visualization of the GUI of the web services.

Question 8 states "UI of composed services" canvas provides a clear visualization of the application flow of the composed services. If your response is 3 and below, please state why the "UI of composed services" canvas does not provide a clear visualization of the application flow of the composed services." All participants agreed or strongly agreed that "UI of composed services" canvas provides a clear visualization of the application flow of the composed services. The mean is 4.55 (low SD=0.51) which falls between 'Agree' and 'Strongly Agree', it shows that "UI of composed services" canvas provides a clear visualization of the application flow of the composed services. Median 5 shows that more than half of the participants rated 'Strongly Agree' on "UI of composed services" canvas provides a clear visualization of the application flow of the composed services. Similarly, mode 5 also shows that majority of participants rated 'Strongly

Agree' on "UI of composed services" canvas provides a clear visualization of the application flow of the composed services.

Question 9 states "UI of composed services" canvas provides a clear visualization of the control flow of the composed services. If your response is 3 and below, please state why the "UI of composed services" canvas does not provide a clear visualization of the control flow of the composed services.". All the participants agreed or strongly agreed that the "UI of Composed Services" canvas provides a clear visualization of the application and control flows of the composed services. The mean is 4.6 (low SD=0.50) which falls between 'Agree' and 'Strongly Agree', it shows that "UI of composed services" canvas provides a clear visualization of the control flow of the composed services. Median 5 shows that more than half of the participants rated 'Strongly Agree' on "UI of composed services" canvas provides a clear visualization of the control flow of the composed services. Similarly, mode 5 also shows that majority of participants rated 'Strongly Agree' on "UI of composed services" canvas providing a clear visualization of the control flow of the composed services.

Question 10 states "Workflow" canvas provides a clear visualization of the data flow of the composed services. If your response is 3 and below, please state why the "Workflow" canvas does not provide a clear visualization of the data flow of the composed services." All the participants agreed or strongly agreed that the "Workflow" canvas provides a clear visualization of the data flow of the composed services. The mean is 4.55 (low SD=0.51) which falls between 'Agree' and 'Strongly Agree', it shows that "Workflow" canvas provides a clear visualization of the data flow of the composed services. Median 5 shows that more than half of the participants rated 'Strongly Agree' on "Workflow" canvas providing a clear visualization of the data flow of the composed services. Similarly, mode 5 also shows that majority of participants rated 'Strongly

Agree' on "Workflow" canvas providing a clear visualization of the data flow of the composed services.

Question 11 states "The three flows (application flow, control flow and data flow) are displayed clearly in the integrated two views (workflow canvas and UI of composed services canvas) during design time. If your response is 3 and below, please state which part(s) is/are not displayed clearly in the integrated two views during design time." All the participants agreed or strongly agreed that Workflow canvas provides a clear visualization of the data flow of the composed services, and the three flows were displayed clearly in the integrated two views during design time. The mean is 4.3 (low SD=0.47) which falls between 'Agree' and 'Strongly Agree', it shows that the three flows (application flow, control flow and data flow) are displayed clearly in the integrated two views (workflow canvas and UI of composed services canvas) during design time. Median 4 shows that more than half of the participants rated 'Agree' or 'Strongly Agree' on the clear display of the three flows in the integrated two views. Similarly, mode 4 also shows that majority of participants rated 'Agree' on the clear display of the three flows in the integrated two views.

Question 12 states "The execution process of the composed services is based on the three flows configuration (application flow, control flow and data flow). If your response is 3 and below, please state why you say that the execution process of the composed services is not based on the three flows configuration." Nineteen participants (95%) agreed or strongly agreed that the execution process of the composed services was based on the three flows configuration and they managed to get the execution results. One (5%) participant was undecided on this aspect. The mean is 4.5 (low SD=0.61) which falls between 'Agree' and 'Strongly Agree', it shows that the execution process of the composed services is based on the three flows configuration (application flow, control flow and data flow). Median 5 shows that more than half of the participants rated 'Strongly Agree' on the execution process of the composed services based on the three flows configuration. Similarly, mode 5 also shows that majority of participants rated 'Strongly Agree' on the execution process of the composed services is based on the three flows configuration.

Question 13 states "Does having the three flows configurations integrated in the two views help you in composing the application? Please state your reason.". All of the participants indicated a "Yes". Figure 6.3 shows the distribution of the reasons given for a 'Yes' answer for Question 13.

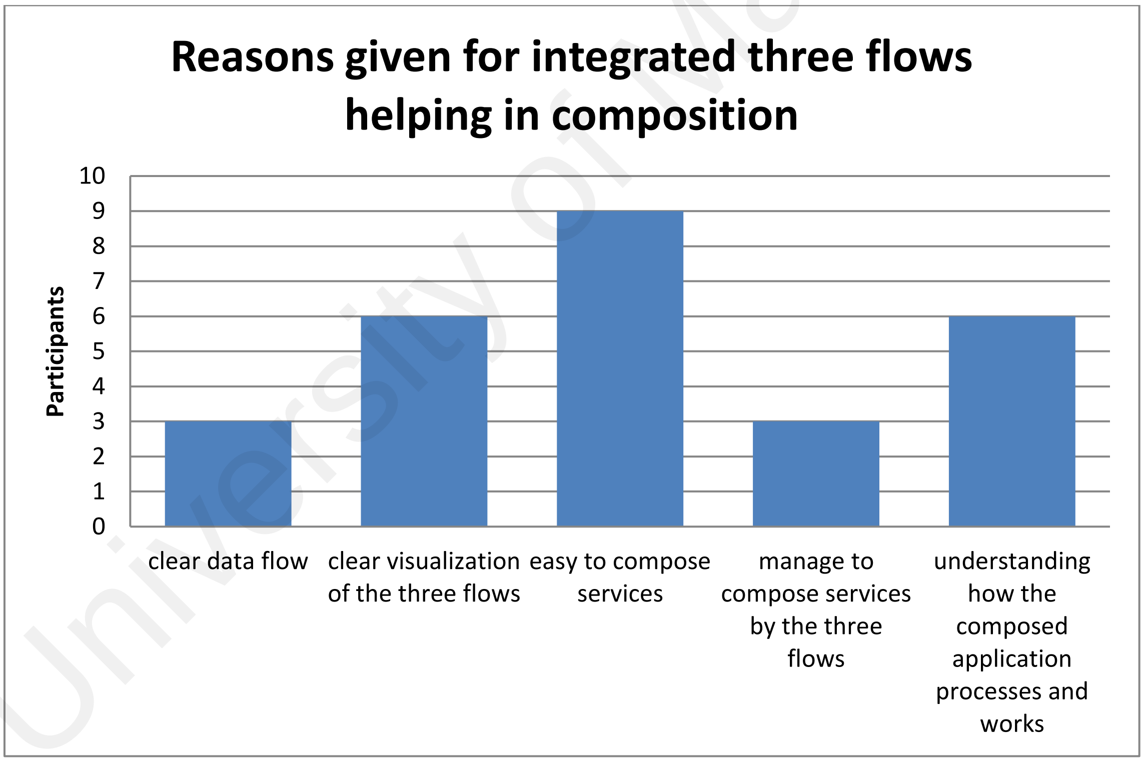


Figure 5.3: Reasons Given for Question 13

Question 14 states "Is it useful to have an end-user service composition tool that helps you to create the User Interface of the composed application when you are composing the services? Please state your reason." All participants stated "Yes". Figure 6.4 shows the distribution of the reasons given for a 'Yes' answer for Question 14.

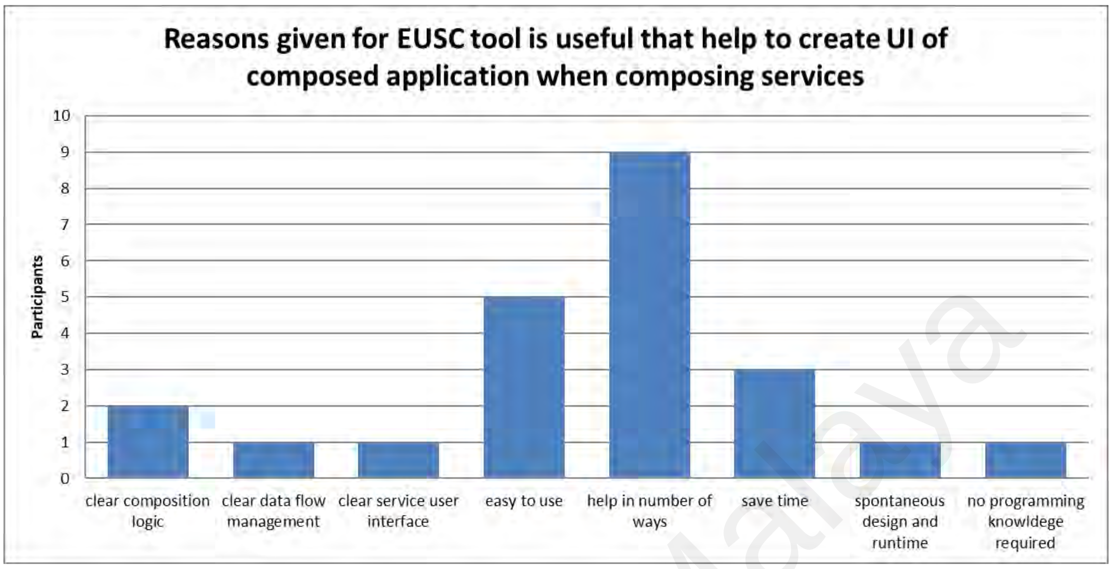


Figure 5.4: Reasons Given for Question 14

5.5 Results Discussion

This section discusses the results of the user evaluation study. ISO 9241-11 standard (Bevan, 1999; Mifsud, 2020; Speicher, 2015) was used to measure the usability of the prototype, QuickWSC. It was used to measure the usability of a product by specified users in terms of three factors (effectiveness, efficiency and satisfaction).

Effectiveness: In this research, the effectiveness of QuickWSC was measured using the following equation (Alturki, Gay, & Alturki, 2017) by using the observation results (Section 6.4.3):

$$Effectiveness = \frac{Number\ of\ tasks\ completed\ successfully}{Total\ number\ of\ tasks\ undertaken}\ x\ 100\%$$

The effectiveness of QuickWSC was calculated based on the successful completion of the 6 types of tasks the participants performed, namely, 4 service selection tasks, 1 application flow configuration task, 1 control flow configuration task, 2 data flow configuration tasks, 1 service composition task and 1 service execution task. Figure 6.5 shows the effectiveness of QuickWSC for twenty participants.

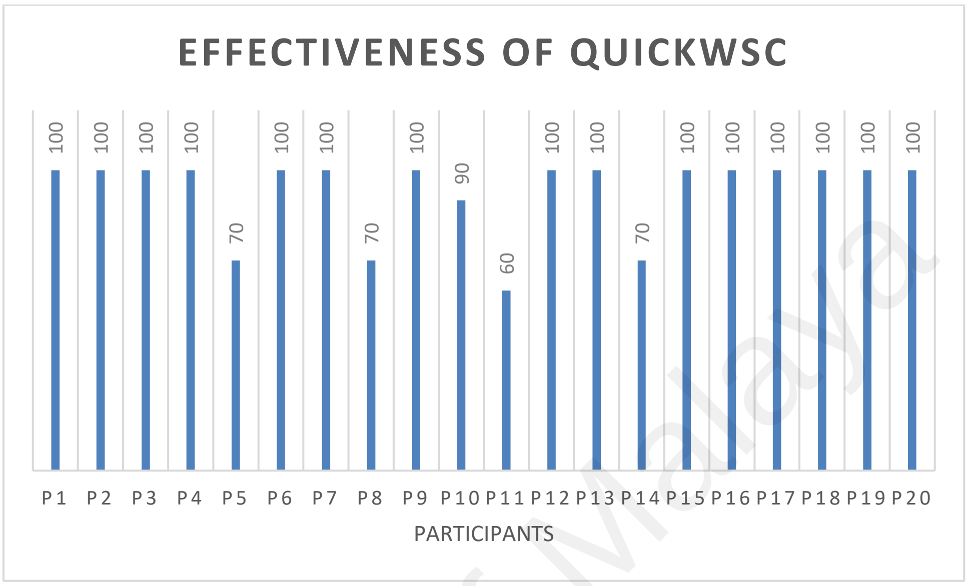


Figure 5.5: Effectiveness of QuickWSC

Fifteen participants (P1, P2, P3, P4, P6, P7, P9, P12, P13, P15, P16, P17, P18, P19, P20) scored 100% for effectiveness. This indicates that 75% of the participants completed the 6 types of tasks successfully although some of them experienced some minor issues as described below: Two of them (P3, P7) had doubts on the functionality of certain selected web services and the location to place the web services when they were selecting the web services. The participants would be able to complete the service selection task more smoothly if it was easy to find the relevant services from QuickWSC. The median and mode of question 1 in Table 5.4 are both 4 which shows that majority of the participants agreed on the ease of finding the relevant services. However, the mean score of question 1 (3.85) in Table 5.4 shows some participants have had certain concerns when finding the relevant services based on the scenario. Two participants were undecided on the ease of finding the relevant web services giving the reasons that they were new to web services and not sure whether or not they were choosing the correct web services. Three participants disagreed or strongly disagreed that it was easy to find the relevant services. The reasons they gave were that they did not know what web services was and one of them have never explored web services.

From Table 5.2, all of the participants were able to configure the application flow. The participants should be able to complete the application flow configuration task if it is easy to specify the application flow when using QuickWSC. The median and mode of question 2 in Table 5.4 are both 4 which shows that majority of the participants agreed on the ease of specifying the application flow. The mean score of question 2 (3.95) in Table 5.4 near to 4 shows that the participants agreed on the ease of specifying the application flow. However, three participants gave their opinions about the application flow configuration. One of them stated that they were unfamiliar with the tools when using it for the first time. Another was confused with arranging the application flow in the canvas. The third was confused between the application flow and control flow.

The participants should be able to complete the data flow configuration task if it is easy to specify the data flow when using QuickWSC. Although the positive result of mean (4.3), median (4) and mode (4) for question 4 in Table 5.4 shows majority of the participants agreed that it was easy to specify the data flow, but the researcher observed that participants experienced some minor issues as follows which did not affect the results. One participant (P8) was confused with the use of the data flowchart, one participant (P12) double checked with the researcher on the data flow configuration and four participants (P1, P4, P16, P17) made a few attempts to configure the data flow due to misunderstanding on how to connect. Besides that, one participant (P20) lacked the understanding on data flow but manage to complete all the tasks.

One participant scored 90% for effectiveness. This participant (P10) failed on data flow configuration task because of the lack of understanding on data flow. Participant should be able to complete the control flow configuration task if it is easy to specify the control flow, and to complete the service composition task if it is easy to specify the three flows in the integrated two views. The mean (4.2), median (4) and mode (4) for question 3 in Table 5.4 shows majority of the participants agreed that it is easy to configure the control flows. The mean (4.05), median (4) and mode (4) for question 5 in Table 5.4 shows majority of the participants agreed that it is easy to configure the three flows in the integrated two views. Participants should also be able to complete the service execution task if the composed services executed based on the three flows configuration. The mean (4.5), median (5) and mode (5) for question 12 in Table 5.4 shows majority of the participants strongly agreed that the execution process of composed services is based on the three flows configuration. However, one participant (P1) hesitated to key in the input value and three participants (P3, P16, P17) took some time to think on how to invoke for the first web service during the service execution task. There were three participants (P5, P8, P14) who scored 70%, and one participant (P11) who scored 60% for effectiveness. All of these four participants failed on control flow configuration task, service composition task and service execution task. One participant who was undecided on the ease of specifying the control flow was confused between the application flow and control flow. Two participants who were undecided on the ease to compose the application by specifying the three flows in the integrated two views stated that this was new to them and they were not familiar on their first use of this tool, but would be easy to use once familiar with it. One participant who was undecided that the composed services executed based on the three flow configurations stated that he was not sure whether he was composing in the right way due to confusion between application flow and control flow. The control flow configuration task affected the service composition task and service execution task because the service composition task and service execution task would fail if control flow configuration task failed. P11 also failed one data flow configuration task due to a lack of understanding on data flow. The average of effectiveness found for the 20 participants were taken as the overall effectiveness of QuickWSC as below:

$$Overall\ effectiveness\ of\ QuickWSC$$

$$= \frac{\begin{array}{c}100+100+100+100+70+100+100+70+100+90+\\60+100+100+70+100+100+100+100+100+100\end{array}}{20}$$

$$= 93\%$$

The overall effectiveness of QuickWSC at 93% shows a high level of effectiveness which indicates that the participants were able to carry out the tasks involved in service composition by using QuickWSC which adopted an integrated three-flow approach and that QuickWSC is effective.

Based on the data analysis of question 13 in Section 5.4.4, the three-flow configuration integrated in the two views helped the participants in composing the application because it was easy to compose the services. Besides that, the system showed a clear visualization of the three flows and it assisted them in understanding how the composed application processes and works. Some participants stated that they managed to compose services by applying the three flows and the system showed a clear data flow.

Efficiency: Efficiency refers to the time taken by participants to complete a task *(Alturki et al., 2017)*. In this research, efficiency is used to measure the time taken to complete the web service composition by the twenty participants. The efficiency of QuickWSC was measured using Overall Relative Efficiency, using the equation *(Alturki et al., 2017)* below.

$$Overall\ Relative\ Efficiency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{n} n_{ij}t_{ij}}{\sum_{j=1}^{R}\sum_{i=1}^{n} t_{ij}}\ x\ 100\%$$

where:

R = The number of users

n_{ij} = The result of task i by user j; if the user successfully completes the task, then

$n_{ij} = 1$, else $n_{ij} = 0$

t_{ij} = The time spent by user j to complete task i in minute. If the task is not successfully completed, then time is measured until the moment the user quits the task.

$Overall\ Relative\ Efficiency$ of QuickWSC

$$= \frac{\begin{gathered}(1x15)+(1x6)+(1x9)+(1x5)+(0x5)+(1x3)+(1x9)+(0x7)+(1x7)+(1x8)+\\(0x5)+(1x10)+(1x7)+(0x7)+(1x7)+(1x5)+(1x8)+(1x5)+(1x6)+(1x8)\end{gathered}}{15+6+9+5+5+3+9+7+7+8+5+10+7+7+7+5+8+5+6+8}\ x\ 100\ \%$$

= 83.1%

The 83.1% overall relative efficiency of QuickWSC shows that QuickWSC is efficient and it supports the participants to achieve the goal which is composing web services.

Satisfaction: According to ISO9241-11 (Diah et al., 2010; Speicher, 2015), satisfaction is defined as ‘freedom from discomfort and positive attitudes toward the use of the system’. The satisfaction level can be looked into the areas of ease of use, organization of information, clear labeling, visual appearance, contents, and error corrections of the system (Alturki et al., 2017; Jeng, 2005). This research focused on the ease of use, organization of information and clear visualization aspect of satisfaction. The average score of the 5-point Likert scale Questions 1 to 11 in Part B of the questionnaire were used to measure the satisfaction level of QuickWSC. Questions 1 to 5 ask about the ease of use of QuickWSC. Question 6 asks about the synchronization of web service information between two views where the two views presents the different configuration information. Question 7 asks about the clear visualization of web service GUI while question 8 to 11 asks about the clear visualization of three flows. Figure 5.6 shows the average satisfaction score for Questions 1 to 11 of the twenty participants. The average satisfaction score of the twenty participants is taken as the overall satisfaction score of QuickWSC with the maximum score of 5.

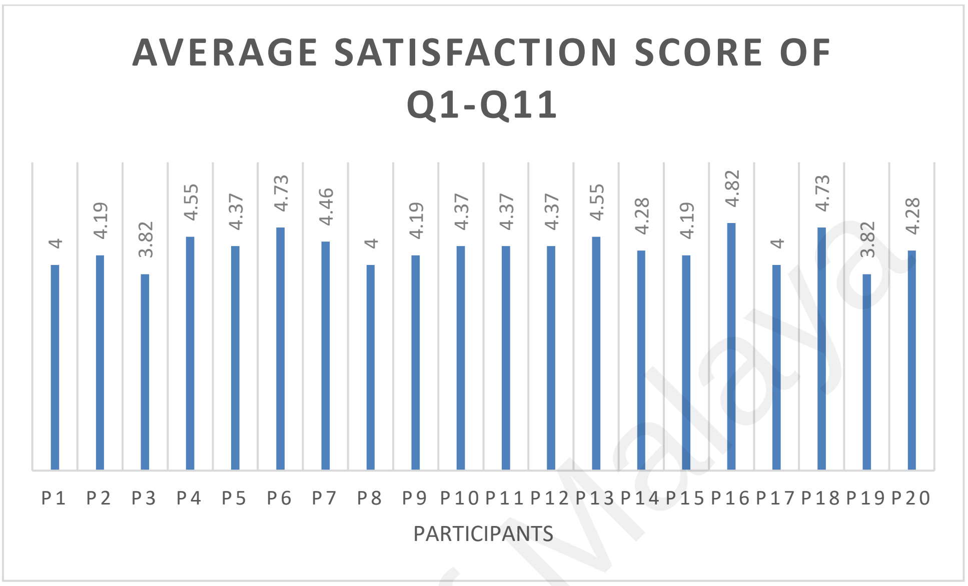


Figure 5.6: Average Satisfaction Score

$Overall\ satisfaction\ score\ of\ QuickWSC$:

$$= \frac{\begin{array}{c}4 + 4.19 + 3.82 + 4.55 + 4.37 + 4.73 + 4.46 + 4 + 4.19 + 4.37 + \\ 4.37 + 4.27 + 4.55 + 4.28 + 4.19 + 4.82 + 4 + 4.73 + 3.82 + 4.28\end{array}}{20}$$

$= 4.30$

The average satisfaction score of 18 participants (P1, P2, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, P18, P20) for the 11 questions are 4, which falls in the range between 'Agreed' and Strongly Agreed'. The average satisfaction scores of the remaining 2 participants (P3, P19) are between 3 and 4. P3 reasoned that they were unfamiliar to the system. P19 reasoned that is was difficult to find the relevant web services. However, the overall satisfaction score of QuickWSC is 4.30 out of 5, which falls between 'Agreed' and 'Strongly Agreed', shows a high satisfaction. This result shows that the participants were satisfied with QuickWSC. Based on the data analysis of question 14 in Section 5.4.4, all of the participants also agreed that it is useful to have an end-user service composition tool that helps them to create the User Interface of the composed application when composing the services. The reasons they gave were that the tool is easy to operate and saves time in composing the web services; that they could use the tool to solve some simple problems especially during emergencies. There were a minority of opinions that gave the reasons of clear composition logic, clear data flow management, clear service user interface, spontaneous design and runtime, and no programming knowledge required. However, one participant stated that it is slightly difficult for first-time users as they are not familiar with the tool.

Based on the high percentage of effectiveness, efficiency and satisfaction score, QuickWSC showed a high level of usability.

Discussion of Other Results from Questionnaire: The positive results on the ease of finding the relevant web services and on the ease of specifying each of the three flows (Part B, Questions 1 – 4) shows that QuickWSC is easy to use. It reduces the difficulty for end-users in specifying the execution order of the web services, as well as logic of application and data flow configuration by explicitly configuring the three flows proposed by the proposed approach.

Results are also positive in terms of synchronization of the web service information between the two views/canvases (Part B, Question 6) and execution of composed services that follows the three flows configuration (Part B, Question 12). The synchronization of web service information between the two canvases reflects an instant update of changes and composition results.

Besides that, results also show that QuickWSC provides a clear visualization of the GUI of the web services and the three flows (Part B, Questions 7 – 11) and the clear visualization of three flows assists the participants in understanding how the composed application processes and works (Part B, Question 13). This indicates that the auto generation of web service UI and three-flow configurations in the proposed approach provides a concrete and organized visual layout of the composed application which reduces the difficulty in positioning the layout of the web services. In addition, it enables end-user programmers to gain basic understanding of web services composition process through the visualization of the three types of flows.

The results also show that it is easy to compose web services by explicitly specifying the three flows in an integrated two-view using QuickWSC (Part B, Question 5) and the three flows configurations integrated with the two views helps in composing application from web services (Part B, Question 13). This shows that the proposed three-flow approach successfully simplifies the process of web service composition and it practically leverages a tight integration between the UI development and the composition of the web services that helps end-user programmers to compose service-based applications without much difficulty.

The results also show that EUSC tool which provides UI development during the composition process is useful for end-user programmers because it could help in a number of ways (as stated by the participants: solving complicated tasks or problems, great during emergencies, in choosing own preferences of tools, acquiring the help needed, only one software to solve problems, tailored application according to needs easily) (Part B, Question 14) and it is easy to use and save time. EUSC tool is suitable for the end-user programmers who have no technical knowledge. The results of the user evaluation study show a promising potential of QuickWSC and its underlying approach.

5.6 Summary

This chapter describes the pilot study design and results followed by the user evaluation study to collect the data from 20 participants for the evaluation purpose. The user evaluation study included the qualitative and quantitative data collection and analysis. The data collection methods used were think-aloud protocol, observation and survey. Framework analysis approach was used to analyse the collected qualitative data. Thematic tables were produced based on the framework analysis approach to analyse the problems encountered by the participants from data collected over the think-aloud protocol and observation. Framework analysis approach was also used to categorize the open-ended questionnaire in the survey for the opinions given by the participants towards the proposed approach and EUSC tool. Descriptive statistics was used to analyse the quantitative 5-point Likert scale questionnaire. After that, the analysed data were used to discuss the usability of QuickWSC in terms of effectiveness, efficiency and satisfaction, and the features of QuickWSC that incorporated the proposed approach. The results show a promising potential of QuickWSC and its underlying approach.

CHAPTER 6: CONCLUSION

This research proposed a three-flow approach that supports a close integration between web service composition and the development of the User Interface of the composed services. This chapter discusses the achievements of the research objectives, threats to validity, contribution of research and future work.

6.1 Achievement of Research Objectives

To achieve RO1, existing web service composition approaches/techniques and features of FESC tools were reviewed (Chapter 2).

RO2 was achieved through the development of the proposed integrated three-flow approach (application flow, control flow and data flow) for end-user service composition support, concurrent UI development, and web service composition to deal with the current limitations of FESC (RO2). The approach allows end-user programmers to explicitly configure the three different types of flows (application flow, control flow and data flow) involved in service composition.

A proof-of-concept prototype, QuickWSC that incorporates the three-flow approach was developed (RO3). It adopts a side-by-side multiple-view design to support visual configuration of the three flows in an uncluttered yet synchronized manner that adhered to established design guidelines.

A user evaluation study on QuickWSC (RO4) was conducted. The results show that QuickWSC has a high usability and the underlying approach is promising. It is easy to compose web services by explicitly specifying the three flows, configurations integrated in the two views using the three flows helps in composing application from web services, and that no technical knowledge is required to use QuickWSC.

6.2 Threats to Validity

This section discusses the threats to internal and external validity of the research and explains the appropriate design strategies to mitigate the threats.

According to Brewer and Crano (2000), internal validity is defined as “the true value that can be assigned to the conclusion that a cause-effect relationship between an independent variable and dependent variable has been establish within the context of the particular research setting”. In a simple way, internal validity is the degree of confidence that the causal relationship tested is not influenced by other factors or variables. The participants had been given a brief introduction about the purpose of the study. This could have made the participants aware of what was studied and influenced their performance. However, this possible “awareness” threat was minimized as the participants did not know the actual scenario of web services composition that they were supposed to perform until the composition task was given. Besides that, the participants were given the freedom to compose the web services. This could have caused them to perform some unnecessary tasks. However, all the participants underwent the same user evaluation study design and setting, and therefore the threat to “instrumentation bias” was minimized.

External validity refers to the generalization of causal finding, that is, whether it can be concluded that the same cause-effect relationship would be obtained across different subjects, settings and methods (Brewer & Crano, 2000). One of the threats to external validity of this study is that a small number of participants were recruited for the user evaluation study due to the difficulty of recruiting participants and the time constraint of this project. This caused the results to be non-generalizable. Nevertheless, all the participants recruited have a similar background and fulfilled the end-user criteria of the research. In addition, ISO standard was used to measure the usability of prototype and this helped to minimize the threat to “reliability of measure”.

6.3 Contribution of Research

This research makes the following contributions:

a) A new three-flow approach that leverages instant integration, synchronization and visualization of the application’s UI development and web service composition. The approach simplifies the process of web service composition by allowing end-user programmers to configure three different types of flows that are important in service-assembled applications through graphical UIs without the need of any technical knowledge. The proposed approach addresses a number of conceptual and usability issues of service composition found in the literature. The user evaluation study results show that the approach as implemented in the prototype tool (QuickWSC) is easy to use and has promising potential in the field of FESC.

b) QuickWSC, a proof-of-concept prototype serves as a working model of the approach. The prototype can be extended to include more web services for use in the composition of different types of applications.

6.4 Future Works

The current approach only covered three (sequence, merge and split) out of the five control flow patterns, the other two control flow patterns (condition and loop) can be future works. From the observation results, some end-user programmers were doubtful when they were selecting the web services from the listing. The prototype can be improved in future to show the functionalities of the respective web services more explicitly to the end-user programmers. Besides that, the security for sensitive information can be taken into consideration in future because it is one of the issues end-user programmers are concerned about. Another future work is to recruit more participants to evaluate QuickWSC in order to get more findings.

6.5 Conclusion

End-User Service Composition (EUSC) allows end-user programmers to compose their own applications by aggregating existing web services. However, end-user programmers typically have a low level of technical knowledge. They still require some techniques to help them to perform EUSC even though many approaches have been introduced to support it. Front-end Service Composition (FESC) has been introduced to assist end-user programmers who are lacking in programming skills for composing web services. FESC enhances the intuitiveness of the service composition process for the end-user programmers where the composition approach is characterized by composition of the web services in the User Interface. However, there are not many studies on FESC and end-user programmers has also experienced a number of conceptual and usability issues of service composition.

This research developed a new FESC approach that leverages a tight integration between the development of the UI of an application and the composition of the web services required by the application, to enable end-user programmers to compose service-based applications by existing web services without much difficulty. The three-flow approach which was integrated in a multiple view design was successfully implemented as a FESC tool (QuickWSC). The three flows play important roles because they represent the composition logic of the composition process. The UI of the web services is automatically generated and this reduces the effort required for front-end development. A side-by-side multiple-view design was adopted during the development to support visual configuration of the three flows in a synchronized manner by using established design guidelines. QuickWSC that incorporates the three-flow approach was developed as a working model of the proposed approach.

A mixed method research comprising qualitative and quantitative method was used to design the user evaluation study to evaluate QuickWSC that incorporates the three-flow approach. The user evaluation study that includes think-aloud protocol, observation and survey was conducted where 20 end-user programmers were recruited to evaluate QuickWSC. The results show that QuickWSC has a high level of usability. It simplifies the composition of service applications for end-user programmers by explicitly specifying the three flows in the two views and that no technical knowledge is required to use QuickWSC.

The outcome of the result shows that the underlying approach is promising and it has achieved the aim of the research.

REFERENCES

Agarwal, V., Chafle, G., Dasgupta, K., Karnik, N., Kumar, A., Mittal, S., & Srivastava, B. (2005). Synthy: A system for end to end composition of web services. *Journal of Web Semantics, 3*(4), 311-339.

Akiki, P. A., Akiki, P. A., Bandara, A. K., & Yu, Y. (2020). EUD-MARS: End-user development of model-driven adaptive robotics software systems. *Science of Computer Programming, 200*, 102534.

AlSedrani, A., & Touir, A. (2016). Web service composition processes: A comparative study. *7*(1), 1-21.

Alturki, R., Gay, V., & Alturki, R. (2017). *Usability testing of fitness mobile application: methodology and quantitative results*.

Anne, H. (2013). Direct Observation. In V. Fred R. (Ed.), *Encyclopedia of Autism Spectrum Disorders* (pp. 980). New York, NY: Springer.

Ardito, C., Costabile, M. F., Desolda, G., Manca, M., Matera, M., Paternò, F., & Santoro, C. (2019). *Improving Tools that Allow End Users to Configure Smart Environments.* Paper presented at the International Symposium on End User Development.

Bak, N., Chang, B.-M., & Choi, K. (2020). Smart Block: A visual block language and its programming environment for IoT. *Journal of Computer Languages, 60*, 100999.

Baldonado, M. Q. W., Woodruff, A., & Kuchinsky, A. (2000). Guidelines for using multiple views in information visualization. In *Proceedings of the working conference on Advanced visual interfaces* (pp. 110-119): Association for Computing Machinery.

Barricelli, B. R., Cassano, F., Fogli, D., & Piccinno, A. (2019). End-user development, end-user programming and end-user software engineering: A systematic mapping study. *149*, 101-137.

Bevan, N. (1999). Quality in use: Meeting user needs for quality. *Journal of systems and software, 49*(1), 89-96.

Bogart, C., Burnett, M., Cypher, A., & Scaffidi, C. (2008). End-user programming in the wild: A field study of CoScripter scripts. In *2008 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 39-46): IEEE.

Brewer, M. B., & Crano, W. D. (2000). Research design and issues of validity. 3-16.

Burnett, M. M., & Myers, B. A. (2014). Future of end-user software engineering: beyond the silos. In *Proceedings of the on Future of Software Engineering - FOSE 2014* (pp. 201-211).

Cappiello, C., Matera, M., & Picozzi, M. (2015). A UI-Centric Approach for the End-User Development of Multidevice Mashups. *ACM Transactions on the Web, 9*(3), 1-40. doi:10.1145/2735632

Chang, K. S.-P., & Myers, B. A. (2017). Gneiss: spreadsheet programming using structured web service data. *Journal of Visual Languages & Computing, 39*, 41-50.

Coronado, E., Mastrogiovanni, F., Indurkhya, B., & Venture, G. (2020). Visual Programming Environments for End-User Development of Intelligent and Social Robots, a Systematic Review. *Journal of Computer Languages*, 100970.

Daniel, F., Casati, F., Benatallah, B., & Shan, M.-C. (2009). Hosted universal composition: Models, languages and infrastructure in mashart. In *International Conference on Conceptual Modeling* (pp. 428-443). Springer, Berlin: Springer.

Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., & Saint-Paul, R. (2007). Understanding ui integration: A survey of problems, technologies, and opportunities. *IEEE Internet Computing, 11*(3), 59-66.

Diah, N. M., Ismail, M., Ahmad, S., & Dahari, M. K. M. (2010). Usability testing for educational computer game using observation method. In *2010 international conference on information retrieval & knowledge management (CAMP)* (pp. 157-161): IEEE.

Driss, M., Aljehani, A., Boulila, W., Ghandorh, H., & Al-Sarem, M. (2020). Servicing Your Requirements: An FCA and RCA-Driven Approach for Semantic Web Services Composition. *8*, 59326-59339.

Fisher, M. J., & Marshall, A. P. (2009). Understanding descriptive statistics. *Australian Critical Care, 22*(2), 93-97.

Foundation, E. (2020). Eclipse IDE for Java Developers. Retrieved from https://www.eclipse.org/downloads/packages/release/oxygen/3a/eclipse-ide-java-developers

Foundation, T. A. S. (2020). Apache Tomcat. Retrieved from http://tomcat.apache.org/

Hang, F., & Zhao, L. (2013). HyperMash: a heterogeneous service composition approach for better support of the end users. In *2013 IEEE 20th International Conference on Web Services* (pp. 435-442): IEEE.

Hang, F., & Zhao, L. (2015). Supporting End-User Service Composition: A Systematic Review of Current Activities and Tools. In *2015 IEEE International Conference on Web Services* (pp. 479-486).

Jääskeläinen, R. (2010). Think-aloud protocol. *1*, 371-374.

Jeng, J. (2005). Usability assessment of academic digital libraries: effectiveness, efficiency, satisfaction, and learnability. *Libri55*(2-3), 96-121.

Kasmi, M., Jamoussi, Y., & Ghézala, H. H. B. (2018). *Towards a Collaborative, Interactive Web Services Composition Approach Based on an Intentional Group Recommender System.* Paper presented at the International Conference on Design Science Research in Information Systems and Technology.

Laga, N., Bertin, E., & Crespi, N. (2010). Composition at the frontend: The user centric approach. In *2010 14th International Conference on Intelligence in Next Generation Networks* (pp. 1-6): IEEE.

Laga, N., Bertin, E., Glitho, R., & Crespi, N. (2012). Widgets and composition mechanism for service creation by ordinary users. *IEEE Communications Magazine, 50*(3), 52-60.

Latih, R., Patel, A., & Zin, A. M. (2014). A Systematic Literature Review of end-user programming for the web mashup. *60*(1), 119-132.

Lemos, A. L., Daniel, F., & Benatallah, B. (2015). Web service composition: A survey of techniques and tools. *ACM Comput. Surv., 48*(3), 1-41. Retrieved from https://doi.org/10.1145/2831270

Lin, C., Lu, S., Lai, Z., Chebotko, A., Fei, X., Hua, J., & Fotouhi, F. (2008). Service-oriented architecture for VIEW: a visual scientific workflow management system. In *2008 IEEE International Conference on Services Computing* (Vol. 1, pp. 335-342): IEEE.

Lin, J., Wong, J., Nichols, J., Cypher, A., & Lau, T. A. (2009). End-user programming of mashups with vegemite. In *Proceedings of the 14th international conference on Intelligent user interfaces* (pp. 97-106): Association for Computing Machinery.

Liu, X., Hui, Y., Sun, W., & Liang, H. (2007). *Towards service composition based on mashup.* Paper presented at the 2007 IEEE Congress on Services (Services 2007).

Lizcano, D., Alonso, F., Soriano, J., & Lopez, G. (2011). A new end-user composition model to empower knowledge workers to develop rich internet applications. *Journal of Web Engineering, 10*(3), 197-233.

Marin, C., & Lalanda, P. (2007). Docosoc-domain configurable service-oriented computing. In *IEEE International Conference on Services Computing (SCC 2007)* (pp. 52-59): IEEE.

Mehandjiev, N., Lecue, F., Wajid, U., & Namoun, A. (2010). Assisted service composition for end users. In *2010 Eighth IEEE European Conference on Web Services* (pp. 131-138): IEEE.

Microsoft. Microsoft SQL Server 2008 R2 Express. Retrieved from https://www.microsoft.com/en-my/download/details.aspx?id=26729

Microsystems, S. (2002). A Simple Example: HelloWorld. Retrieved from http://www.inf.fu-berlin.de/lehre/SS03/19560-P/Docs/JWSDP/tutorial/doc/JAXRPC3.html

Mifsud, J. (2020). Usability Metrics – A Guide To Quantify The Usability Of Any System. Retrieved from https://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/

Namoun, A., Nestler, T., & De Angeli, A. (2010). Conceptual and usability issues in the composable web of software services. In *International Conference on Web Engineering* (pp. 396-407). Berlin, Heidelberg: Springer.

Namoun, A., Owrak, A., & Mehandjiev, N. (2019). Non-Programmers Composing Software Services: A Confirmatory Study of the Mental Models and Design Challenges. *9*(24), 5558.

Nestler, T., Dannecker, L., & Pursche, A. (2009). User-centric composition of service front-ends at the presentation layer. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops* (pp. 520-529). Berlin, Heidelberg: Springer.

Nestler, T., Feldmann, M., Hübsch, G., Preußner, A., & Jugel, U. (2010). The ServFace builder-A WYSIWYG approach for building service-based applications. In *International Conference on Web Engineering* (pp. 498-501). Berlin, Heidelberg.: Springer.

Obrenović, Ž., & Gašević, D. (2008). End-User Service Computing: Spreadsheets as a Service Composition Tool. *IEEE Transactions on Services Computing, 1*(4), 229-242. doi:10.1109/tsc.2008.16

Okamoto, S., Dascalu, S., & Egbert, D. (2006). Web interface development environment (WIDE): software tool for automatic generation of web application interfaces. In *2006 World Automation Congress* (pp. 1-7): IEEE.

Paik, H.-y., Lemos, A. L., Barukh, M. C., Benatallah, B., & Natarajan, A. (2017). *Web Service Implementation and Composition Techniques* (Vol. 256): Springer.

Patel, S., & Shah, T. (2016). A survey on issues and challenges of web service development, composition, discovery. *5*(1).

Pi, B., Zou, G., Zhong, C., Zhang, J., Yu, H., & Matsuo, A. (2012). Flow editor: Semantic web service composition tool. In *2012 IEEE Ninth International Conference on Services Computing* (pp. 666-667): IEEE.

Pickup, J. C., Holloway, M. F., & Samsi, K. (2015). Real-time continuous glucose monitoring in type 1 diabetes: a qualitative framework analysis of patient narratives. *38*(4), 544-550.

Pietschmann, S., Nestler, T., & Daniel, F. (2010). Application composition at the presentation layer: alternatives and open issues. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services* (pp. 461-468).

Pietschmann, S., Voigt, M., & Meissner, K. (2009). Dynamic composition of service-oriented web user interfaces. In *2009 Fourth International Conference on Internet and Web Applications and Services* (pp. 217-222): IEEE.

Pietschmann, S., Voigt, M., Rümpel, A., & Meißner, K. (2009). Cruise: Composition of rich user interface services. In *International Conference on Web Engineering* (pp. 473-476). Berlin, Heidelberg: Springer.

Radeck, C., Blichmann, G., & Meißner, K. (2013). CapView–functionality-aware visual mashup development for non-programmers. In *International Conference on Web Engineering* (pp. 140-155). Berlin, Heidelberg: Springer.

Ridge, A. (2014). *On the Design of End-user Service Composition Applications.* University of Bath,

Ridge, A., & O'Neill, E. (2014). Establishing requirements for End-user Service Composition tools. *Requirements Engineering, 20*(4), 435-463. doi:10.1007/s00766-014-0207-x

Roberts, J. C. (1998). On encouraging multiple views for visualization. In *Proceedings. 1998 IEEE Conference on Information Visualization. An International Conference on Computer Visualization and Graphics (Cat. No. 98TB100246)* (pp. 8-14). London, UK: IEEE.

Romero, O. J., Dangi, A., & Akoju, S. A. (2019). *NlSC: Unrestricted natural language-based service composition through sentence embeddings.* Paper presented at the 2019 IEEE International Conference on Services Computing (SCC).

Roy Chowdhury, S., Rodríguez, C., Daniel, F., & Casati, F. (2012). Baya: assisted mashup development as a service. In *Proceedings of the 21st International Conference on World Wide Web* (pp. 409-412).

Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., & Xu, X. (2014). Web services composition: A decade's overview. *280*, 218-238.

Shimomura, T. (2004). A page-transition framework for image-oriented Web programming. *29*(2), 10-10.

Speicher, M. (2015). What is usability? a characterization based on ISO 9241-11 and ISO/IEC 25010.

Srivastava, A., & Thomson, S. B. (2009). Framework analysis: a qualitative methodology for applied policy research.

Srivastava, B., & Koehler, J. (2003). *Web service composition-current solutions and open problems.* Paper presented at the ICAPS 2003 workshop on Planning for Web Services.

Tabatabaei, S. G. H., Kadir, W., & Ibrahim, S. (2011). A review of web service composition approaches. In *Proceedings of the 1st International Conference on Computer Science and Information Technology (CCSIT).*

Vulcu, G., Bhiri, S., Hauswirth, M., & Zhou, Z. (2008). A user-centric service composition approach. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 160-169). Berlin, Heidelberg: Springer.

W3Schools. (2020). Java OOP. Retrieved from https://www.w3schools.com/java/java_oop.asp

Wang, G., Yang, S., & Han, Y. (2009). Mashroom: end-user mashup programming using nested tables. In *Proceedings of the 18th international conference on World wide web* (pp. 861-870).

Wilson, V. (2014). Research methods: triangulation. *Evidence based library and information practice, 9*(1), 74-75.

Wong, J., & Hong, J. I. (2007). Making mashups with marmite: towards end-user programming for the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 1435-1444).

Yamaizumi, T., Sakairi, T., Wakao, M., Shinomi, H., & Adams, S. (2006). Easy soa: Rapid prototyping environment withweb services for end users. In *2006 IEEE International Conference on Web Services (ICWS'06)* (pp. 931-932): IEEE.

Yang, J. (2003). Web service componentization. *46*(10), 35-40.

Yu, J., Benatallah, B., Casati, F., & Daniel, F. (2008). Understanding mashup development. *12*(5), 44-52.

Zhai, Z., Cheng, B., Tian, Y., Chen, J., Zhao, L., & Niu, M. (2016). A Data-Driven Service Creation Approach for End-Users. *IEEE Access, 4*, 9923-9940. doi:10.1109/access.2017.2647838

Zhang, J. (2010). Co-Taverna: a tool supporting collaborative scientific workflows. In *2010 IEEE International Conference on Services Computing* (pp. 41-48): IEEE.

Zhao, L., Loucopoulos, P., Kavakli, E., & Letsholo, K. (2019). User studies on end-user service composition: a literature review and a design framework. *13*(3), 1-46.

LIST OF PUBLICATIONS AND PAPERS PRESENTED

Conference Presentation

Lim, M. T., & Su, M. T. (2018). A Review on End-User Service Composition Approaches. In *Malaysian Software Engineering Conference (MySec 2018), Universiti Malaysia Sarawak (UNIMAS)*.

Journal Publication (ISI indexed)

Lim, M. T., & Su, M. T. (2019). AN INTEGRATED THREE-FLOW APPROACH FOR FRONT-END SERVICE COMPOSITION. *Malaysian Journal of Computer Science*, 1-24.