A FUZZY-ORIENTED FRAMEWORK FOR SERVICE RANKING AND SELECTION IN CLOUD e-MARKETPLACES

EZENWOKE, AZUBUIKE ANSALEM

CU033020045

JULY, 2017

A FUZZY-ORIENTED FRAMEWORK FOR SERVICE RANKING AND SELECTION IN CLOUD E-MARKETPLACES

BY

EZENWOKE, AZUBUIKE ANSALEM
B.Sc. and M.Sc. Computer Science
(Covenant University)
MATRICULATION NUMBER: CU033020045

BEING A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES, COLLEGE OF SCIENCE AND TECHNOLOGY, COVENANT UNIVERSITY, OTA, NIGERIA. IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF DOCTOR OF PHILOSOPHY (Ph.D) DEGREE IN COMPUTER SCIENCE.

JULY, 2017

ACCEPTANCE

This is to attest that this thesis is accepted in partial fulfilment of the requirements for the award of the degree of **Doctor of Philosophy in Computer Science** in the Department of Computer and Information Sciences, College of Science and Technology, Covenant University, Ota.

PHILIP JOHN AINWOKHAI ___________________________________

(Secretary, School of Postgraduate Studies) Signature and Date

PROFESSOR SAMUEL WARA ___________________________________

(Dean, School of Postgraduate Studies) Signature and Date

DECLARATION

I, EZENWOKE Azubuike Ansalem (CU33020045), declare that this research work was carried out by me under the supervision of Professor Matthew Adigun of the Department of Computer Science, University of Zululand, South Africa, and Dr. Olawande Daramola of the Department of Computer and Information Sciences, Covenant University, Nigeria. I attest that this thesis has not been presented either wholly or partly for the award of any degree elsewhere. All the sources of the materials consulted in the course of this academic research are duly acknowledged.

EZENWOKE, AZUBUIKE ANSALEM ________________________________

Signature and Date

CERTIFICATION

We certify that the thesis titled "**A Fuzzy-Oriented Framework for Service Ranking and Selection in Cloud E-Marketplaces**" is an original research work carried out by **EZENWOKE, AZUBUIKE ANSALEM (CU033020045)**, in the Department of Computer and Information Sciences, College of Science and Technology, Covenant University, Ogun State, Nigeria, under the supervision of Professor Matthew Adigun and Dr. Olawande Daramola. We have examined and found the work acceptable for the award of a degree of Doctor of Philosophy in Computer Science.

PROFESSOR MATTHEW ADIGUN ______________________
(Supervisor) Signature and Date

DR. OLAWANDE DARAMOLA ______________________
(Co-Supervisor) Signature and Date

DR. OLUFUNKE OLADIPUPO ______________________
(Head of Department) Signature and Date

PROFESSOR CHARLES UWADIA ______________________
(External Examiner) Signature and Date

PROFESSOR SAMUEL WARA ______________________
(Dean, School of Postgraduate Studies) Signature and Date

DEDICATION

To God, the Source of All Life and Force behind All Living.

To Chidi and Angelina Ezenwoke, my loving parents, who brought me up in the ways of the Lord and sacrificed everything to give me the best education. Mummy, although Daddy is no more with us, all your dreams and aspiration will still come to pass.

To all young and aspiring researchers desiring to make a mark in this world, I say to you, that all things are possible to him that believes.

ACKNOWLEDGEMENTS

... I, Wisdom, dwell with prudence and search out knowledge of witty inventions
...But there is a spirit in man and the inspiration of the Almighty gives him understanding.

I thank God, My Source, My Help, My Shield and Buckler, by whom all things were created, in whom all things consist, and for whom all things exist. This Ph.D. journey began with inspiration from HIM, as HE began to guide and shape my understanding of how concepts relate and how to make sense of all things. The results in this thesis are beyond those listed in the concluding sections; rather, it was a process that changed me as an individual. I believe the Ph.D. adventure opened me up to greater possibilities than I could have ever been exposed to, through the various trains of explorations, and flights of new discoveries. By undertaking this Journey, God has revealed my passion, pointing me to the contributions I look forward to making in my lifetime. Through God, I am becoming a thought leader in the variety of domains I have come to appreciate in Computer science and beyond. For these changes in me, I am deeply grateful to God.

Most times, God sends help through people. The people who made this research and thesis a reality are forever esteemed.

First, I most respectfully acknowledge Dr. David Oyedepo (Daddy), whose life and values provided the platform for me to fulfil my destiny. I am always inspired to action by his lifestyle of continuous contribution and the firm belief that nothing is impossible. The Covenant University platform and all that it represents remains the stage for the blossoming and unveiling of many destinies and the forte of many career exploits. Thank you, daddy, for your strong but gentle nudges to get this Ph.D. over with and challenging me to give more than is expected.

Prof. Matthew Adigun came into my life at a time when I needed light and has since then shaped and reshaped my whole concept of research during the three months visit he facilitated, to the centre of mobile e-services, University of Zululand, South Africa. I thank God, that that privilege afforded me the proximity required to learn from you the differences between the practice of software engineering and research; which remains my most favourite lesson learnt under your tutelage. I salute your passion for training and mentoring young researchers, and I watched you turn 'boys' and 'girls' into 'men' and ‘women’, and the privilege of being one of them. In addition, your research journey reminds me the place of consistency of focus in the pursuit of my future endeavour; your strong guidance and unapologetic corrections together with those from Dr. Olawande Daramola, my Co-supervisor, are inestimable and have made this thesis a reality.

Dr. Olawande Daramola was present, interpreting both the high concepts and expectations of Prof. Adigun in a language I could understand. At many cross-roads, your promptings, suggestions and encouragements resolved many deadlocks in the course of this research. Your dexterity in both written and spoken words is a standard I look up to attaining, as I remember with fondness the first time you confirmed to me that I could write too, after reading the very first draft of my proposal. Honestly, I felt like I had won a million dollars! That affirmation gave me all the impetus I needed to forge ahead and I always look forward to affirming younger research students in the same manner and hope it sparks the same fire of confidence in them.

I acknowledge the immense contributions of HELPERS whom God placed in my path of destiny. These individuals sowed great seeds that are becoming a forest of values today. Professor Aize Obayan, Pastor Yemi Nathaniel and Pastor Daniel Rotimi. You all provided your shoulders for me to stand, taught me valuable life principles, led your lives as examples to follow, and looked forward to the completion of this thesis with great expectation. I am glad to let you know that your wait is over. I’ll always cherish your motherly and fatherly inputs, I celebrate you all.

I acknowledge members of the Covenant University management, spearheaded by Professor AAA Atayero, for forging an inspiring **ReCITe** agenda towards the actualization of the 1-of-10-in-10 mandate. Also, I acknowledge the immediate past VC, Professor CK Ayo for his constant encouragements, as well as all the members of the software engineering and intelligent systems cluster, and the members of the Department of Computer and Information Sciences, led by our amiable and action-driven Head of Department, Dr. Olufunke Oladipupo. These individuals made a very valuable contribution to my ability to articulate and communicate effectively through their very numerous technical suggestions to improve the work.

I sincerely appreciate Professor Ezekiel Adebiyi, for believing in me. You saw something in me right from my undergraduate days and made me my first offer in the world of research. You connected me to destiny and through your life stories, you gave me an example of what I can become and contribute. I am always inspired by your story and your passion for quality. It was your firm conviction for obtaining the best that linked me with Professor Matthew Adigun, and I am also deeply grateful to you, Professor Adebiyi, for the sacrifices you had to make to establish that connection with Professor Adigun.

I acknowledge the critical contributions of the imminent individuals who served as members of the panel of assessors and examiners, and their vital inputs indeed improved the quality of this thesis. They include Professor C. Uwadia, Professor S. Chiemeke, Dr. J. Oyelade, Dr. E. Adetiba, Dr. U. Uwuigbe, Dr. P. Adamu, and Dr. A. Boluwaji.

I thank my friends who were a constant reminder of the **cause**: Dr Olumuyiwa Oludayo, Arch. Gbenga Alalade (Ph.D.), and Dr Stephen Oluwatobi. I'm glad that you guys went ahead, and paved the path of possibilities for me. Now I know that we can and shall change the world together. I appreciate Ndukwe Uwakwe (Mr. Kalu) for his kind support throughout the period of the thesis and excitement in seeing to its completion.

Finally, I thank my ever loving and patient Wife, whose deep and insightful perspectives on issues is unparalleled. Thankfully, it is not good for man to be alone, and with you, I have learnt that all a man needs is just one right company. I have thoroughly enjoyed every waking, living, and sharing moments with you all these years. This Ph.D. journey started just about when we got married in 2011 and your support towards its completion since its inception has remained unwavering; just as your love and commitment to our marriage. From sharing my thoughts with you, demonstrating my presentations, asking your help on statistical data analysis, to caring for our beautiful young Eden and then yet to be born Beth-el, all by yourself, while I am unavailable doing research, are all treasured supports I'll never forget. To my Eden (of God), those times you missed daddy, insisted that daddy stayed home with you or promised not to disturb daddy while he 'works on his laptop' is treasured memories I appreciate. I love you more than words can describe. To my Beth-el, your arrival at the concluding stages of this thesis is a strong reminder of the power of faith in God's word. The enemy tried to distract me with those negative medical reports, but your presence with us today remains a song of victory to God and joy to our hearts. As you both grow up to appreciate the sacrifices you have made, I excitedly look forward to the intellectual contributions you both will make towards advancing God's kingdom and humanity.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AHP	Analytic Hierarchy Process
AWS	Amazon Web Service
CRM	Customer Relationship Management
CRMaaS	Customer Relationship Management as a Service
CSP	Constraint Satisfaction Problem
DM	Decision Maker
eEUD	Exponential Euclidean Distance
EUD	Euclidean Distance
FM	Feature Model
GUI	Graphical User Interface
HCI	Human Computer Interaction
IaaS	Infrastructure as a Service
IV	Information Visualisation
KRC	Kendall Tau Rank Coefficient
MAP	Mean Average Precision
MAUT	Multi-Attribute Utility Theory
MCDA	Multi-Criteria Decision Analysis
MCDM	Multi-Criteria Decision Making
MCKP	Multiple Choice Knapsack Problem
MCSP	Multi-Criteria Selection Problems
MF	Membership Function
MOEA	Multi-Objective Evolutionary Algorithms
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
PaaS	Platform as a Service
PC	Product Configuration
PSSUQ	Post-Study-Satisfaction-User Questionnaire
QoS	Quality of Service
SaaS	Software as a Service
SAW	Simple Additive Weighting
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SPLE	Software Product Line Engineering
SRC	Spearman Rank Coefficient
TFN	Triangular Fuzzy Number
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
UI	User Interface
VSSE	Very Small Software Enterprise
XaaS	Anything or Everything as a Service

ABSTRACT

Despite the successes of commercial cloud service e-marketplaces, opportunities still exist to improve user experience as these e-marketplaces do not yet enable dynamic composition of atomic services to satisfy complex user requirements. More so, the platforms employ keyword-based search mechanisms that only allow the selection of atomic services. The elicitation mechanisms do not consider user's QoS requirements, nor support the elicitation of these requirements in ways akin to subjective human expressions. In addition, search results are presented as unordered lists of icons, with minimal comparison apparatus to simplify decision making. Existing cloud selection approaches do not currently provide the sophistication required to optimise user experience in the cloud e-marketplace, hence this study proposed a framework to address the observed limitations. First, a state-of-the-art survey was conducted and six design criteria were identified for a selection framework suitable for cloud e-marketplaces. These criteria guided the formulation of an integrated framework, Fuzzy-Oriented Cloud Service Selection (FOCUSS) framework. The proposed framework comprises four modules: Cloud ecosystem and service directory, Graphical User Interface (GUI) & Visualisation, QoS Requirement Processing, and Service Evaluation & QoS Ranking modules. In the first module, atomic services are combined to realise the set of composite services offered in the e-marketplace; subjective QoS requirements are then inputted via the GUI module, and processed in the QoS requirements processing module. In service evaluation and ranking module, the requirements are optimised and used to rank services and the ranking results are shown to the users via bubble graph visualisation. The utility of the proposed framework was demonstrated via a Java-based web application prototype using a case study of a Customer Relationship Management-as-a-Service e-marketplace. Simulation experiments and user studies were performed to evaluate the performance of the proposed framework in terms of its scalability, ranking accuracy, and quality of user experience. A linear regression analysis showed that the proposed framework is linearly scalable when measured by the time it took to rank top-20 services as the number of alternatives increased. Kruskal-Wallis and Mann-Whitney tests revealed that ranking accuracy of proposed framework is not compromised by using subjective descriptors to approximate user's QoS requirements, and the ranking accuracy is higher compared to existing approaches. Based on Wilcox signed tests, the results of the user studies showed that users can complete tasks faster and easier compared to traditional tabular representations. These results confirmed that the proposed framework is viable for cloud service selection in cloud e-marketplaces. This study contributes to knowledge by providing an integrated framework for cloud service selection that organises atomic services within the cloud ecosystem and guides formal service composition on the fly beyond what atomic services can deliver; handle both subjective users QoS preferences and aspiration, and enable easy comparison of query results along multiple QoS dimensions. In addition, it provides a framework will improve user experience, which in turn boosts the commercial viability of cloud e-marketplaces.

Keywords: Cloud Service Selection, Cloud Ecosystem, Cloud e-Marketplace, Feature modelling, Fuzzy set theory, Information visualisation

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND

Advancement in information and communications technology has significantly influenced the computing landscape, as more than ever, it is now fashionable to contract out technology demands to the cloud. Cloud computing is a model of internet-based service provisioning where dynamically scalable and virtualized resources (infrastructure, platform and software) are delivered and accessed as services (Rimal *et al.*, 2011; Lewis, 2011; Qaisar, 2012). It enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell and Grance, 2011). Basic cloud computing service models are Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) (Lewis, 2011; Qaisar, 2012), with more complex model morphing into the concept of Anything- or Everything-as-a-Service (XaaS).

Because everything and anything can be offered as a service, the maturity of cloud computing is fast-tracked by commoditizing services in an e-marketplace facilitated by cloud ecosystem (Buyya *et al.*, 2008; Menychtas *et al.*, 2014). A service ecosystem consists of a platform, a set of internal and external providers and a community of service brokers providing value-added services to a community of service users, who consume relevant services (Bosch and Bosch-Sijtsema, 2010; Menychtas *et al.*, 2014). In the future, there will be a large number of cloud services available from multiple providers and brokers (Zeng *et al.*, 2009; Jung *et al.*, 2013), more so that multiple actors (for example, service providers, users, brokers, infrastructure providers etc.) will congregate in an ecosystem to provide, broker, and consume cloud services in a more sophisticated market environment that transcend existing traditional e-marketplaces (Khadka *et al.*, 2011; Vigne *et al.*, 2013; Menychtas *et al.*, 2014). The overabundance of services, that are sometimes functionally equivalent, will leave users with the dilemma of which service to choose; a phenomenon that can be referred to as service choice overload (Chernev *et al.*, 2015; Haynes, 2009; Toffler, 1970). Examples of existing cloud service e-marketplace include SaaSMax, Oracle e-marketplace, Google play store, AppExchange etc. Although, these e-marketplaces consist of basic features that underscore the operations of an e-marketplace, like the product (or service) search, product catalogue, billing etc., more sophisticated features that maximise the dynamism of service orientation and optimise user experience are still lacking (Akolkar *et al.*, 2012).

The cloud service e-marketplace of the future provides, among others, an environment where service providers can combine their offerings in unprecedented ways to create composite services that fulfil complex business processes on the fly; and users can then discover, consume and pay for these services (Akolkar *et al.*, 2012; Barros and Dumas, 2006). It is expected that these e-marketplaces incorporate service combination, service discovery and presentation mechanisms; service combination will be based on service interrelationships established on specific rules and constraints; service discovery and selection mechanisms that enables the elicitation of subjective user requirements, and ranks available services according to those requirements; presentation mechanisms for showcasing suitable service options in a highly interactive and easy-to-understand manner in response to user requirements.

1.2 MOTIVATION

The motivation of this study is threefold: i) Handling of complex user requirements; ii) Enabling flexibility to accommodate the subjective Quality of Service (QoS) requirements, and iii) Improved presentation format for the search results to aid decision making. Next, each point is discussed in details.

1.2.1 Handling of Complex User Requirements

An important enabler for the realisation of a true cloud service e-marketplace is the possibility of formal and/or incidental service composition to satisfy complex user requirements (Akolkar *et al.*, 2012). Formal composition refers to the combination of one or more services into composite services beforehand, while the incidental composition is described as ‘on the spot’ service composition based on specific user request (Akolkar *et al.*, 2012). Existing e-marketplaces possess basic features like product search and directory but lacks the sophistication that can enable dynamic service composition in order to support the realisation of complex business processes (Akolkar *et al.*, 2012). A cloud e-marketplace can benefit from an ecosystem, such that atomic services can be aggregated into composite offerings to be listed in the e-marketplace directory (Akolkar *et al.*, 2012; Barros and Dumas, 2006).

Existing proposals for a cloud service e-marketplace (Gatzioura *et al.*, 2012; Menychtas *et al.*, 2014; Akolkar *et al.*, 2012) did not specify particular methodology of realising service composition but rather presented architectural blueprints of possibilities. Likewise, most cloud service selection methods only enable a user to make selections from a list of predefined atomic services, which cannot address more complex situations where a user's requirements extend beyond the limit of individual atomic services (Zeng *et al.*, 2009; Garg *et al.*, 2011; Rehman *et al.*, 2011). But some authors, such as Wittern *et al.* (2012), Quinton *et al.* (2014), and Quinton *et al.* (2013) have attempted to address these kinds of complex scenario, by enabling prospective users to select desirable features that are available in specific atomic services in order to realise their complex set of requirements. This usually includes specifying both the QoS requirements and selecting features of the services. Still, the drawback of these attempts is that it is cognitively demanding because the user is expected to have deep knowledge of the domain in order to make useful selections. This gap is bridged in this study by the aggregation of atomic services in a way that satisfies complex user requirements.

1.2.2 Enabling flexibility to accommodate subjective QoS requirements

The abundance of cloud service options leaves users in a dilemma of selecting the right service or services amidst a variety of similar services that conforms to certain quality criteria (Zeng *et al.*, 2009; Jung *et al.*, 2013; Garg *et al.*, 2011; Martens *et al.*, 2011; Alrifai *et al.*, 2010). This dilemma, also referred to as choice overload (Toffler, 1970), underscores the paradox of choice (Schwartz, 2004), by describing how difficult it is making a choice in the face of multiple options, particularly when such decisions are made by considering several criteria. Apart from the functional capabilities they provide, cloud services possess non-functional or quality of service (QoS) attributes (e.g. reliability, response time, cost, availability etc.), which becomes the criteria by which selection of services are made (Chen *et al.*, 2013; Barros and Dumas, 2006; Garg *et al.*, 2011).

Although a number of cloud service selection techniques have been proposed in the literature, many of these techniques require that user's QoS requirements are specified in exact or precise terms. Most times, users do not provide QoS requirements in exact crisp terms, but rely on subjective descriptions that approximate these requirements; thus shrouding QoS requirements elicitation with vagueness and imprecision (Barros and Dumas, 2006; Wittern *et al.*, 2012; Rehman *et al.*, 2011; Akolkar *et al.*, 2012). Qu and Buyya (2014) observed that user's QoS requirements can indeed be specified in terms of preferences (user's priority for each QoS dimension) and aspiration (user's values of QoS dimension); which are two important considerations for determining which cloud services to select. However, some existing approaches that have considered subjectivity in user requirements elicit either QoS preferences or QoS aspirations alone from the user but rarely both (e.g. (Esposito *et al*., 2016; Yu and Zhang, 2014). Still, some others, like Esposito *et al*. (2016), Mirmotalebi *et al.* (2012), Rehman *et al.* (2011), and Qu and Buyya (2014), require users to assign priority weights to QoS attributes, with the downside of being less accurate compared to a pairwise comparison of the relative importance of QoS attributes (Millet, 1997).

Noting that the ranking of service alternatives depends on the user's QoS requirements, the accuracy of the ranking should not be compromised by subjective approximate descriptions. Nonetheless, giving users the flexibility of expressing QoS requirements by allowing for subjective descriptions is a plus to the user experience, as the cognitive load of having to craft crisp or precise values is reduced (Akolkar *et al.*, 2012). Hence, this study explores the elicitation of user's requirements in a way that reduces choice overload and improves the user experience of cloud service e-marketplace.

1.2.3 Improved presentation format for the search results to aid decision making

Another dimension of user experience is how information is presented. The search results from existing e-marketplaces are usually presented as an unordered list of icons representing the services that best fit user's keyword-based queries. The drawback of such presentation mechanisms is that users are not able to immediately discriminate among the cloud services presented. Users are required to explore each service one after the order to gain more insights about the QoS attributes to guide their decisions. The additional complexity on the part of the users impacts negatively on user experience.

A number of cloud service selection approaches (Esposito *et al.*, 2016; Yu and Zhang, 2014; Qu and Buyya, 2014; Wittern *et al.*, 2012) have been proposed; however, some of these frameworks present service rankings in textual format, either in a list or tables, which does not fully describe the implicit trade-off factors inherent in the search results. Such presentations are ineffective in supporting the decision making in online e-marketplace environment and can increase cognitive load of users (Beets and Wesson, 2011; Lurie and Mason, 2007; Adnan *et al.*, 2008; Pajic, 2014). Others have used Information Visualizations (IV) like the radar or kiviat charts; which are limited in presenting a large number of cloud services. In addition, such IV approaches exhibit low object coherence and object correlation (Teoh and Ma, 2005), referring to how compactly and distinctly the visual encodings represents the services and their relationships to facilitate easy decision making. Realising the vision of a true cloud service e-marketplace in the face of the growing trend for personalised products and services requires that user satisfaction and user experience be given top priority (Riemer and Totz, 2003; Schubert and Ginsburg, 2000; Liang *et al.*, 2006). Hence the need to simplify cloud service selection, while optimising user experience and satisfaction in the decision-making process (Almulla *et al.*, 2012).

1.3 STATEMENT OF THE PROBLEM

Despite the successes of commercial cloud e-marketplaces (e.g. AppExchange and SaaSMax), these platforms do not yet enable dynamic composition and employ keyword-based search mechanisms that do not consider the user's QoS requirements, nor support the elicitation of these requirements in ways akin to subjective human expressions (Sun *et al*, 2014; Qu *et al.*, 2014). In addition, search results on these platforms are presented as unordered lists of icons, with little or no comparison apparatus that simplifies decision making (Gui *et al.*, 2014).

Existing cloud selection approaches do not currently provide the sophistication to optimise user experience in the cloud service e-marketplace (Akolkar *et al.*, 2012); which ultimately hamper the user experience in the cloud service e-marketplace. Hence the need for an integrated framework that caters for observed limitations in the existing cloud service selection approaches.

Concisely the questions investigated in this study include:

i. *How do we formally combine atomic service offerings from different service providers in order to satisfy complex user requirements?*

ii. *How do we elicit user's QoS requirements, in a way that accommodates human subjective expressions and judgment?*

iii. *How can we present query results in a manner that simplifies decision making?*

1.4 Research Aim and Objectives

The aim of this research is to develop a framework for cloud service selection that improves the quality of user experience in cloud service e-marketplace.

In order to realise the aim of this study, the objectives of this study include the following:

1. The formulation of an integrated service selection framework that will improve the quality of user experience in a cloud service e-marketplace.
2. A design of models and algorithms that will enable the components of the cloud service selection framework.
3. The implementation of a prototype of the cloud service selection framework and a demonstration of its plausibility.
4. An evaluation of the proposed framework in terms of its performance and usability attributes.

1.5 Research Methodology

The research approach employed in carrying out this study is a design cycle that comprises of five sub-processes as developed by Takeda *et al.* (1990). The five sub-processes include the awareness of research problem; the suggestion of a solution; the development and implementation of the solution; the solution validation and evaluation; and the conclusion. The application of each sub-process in this research is summarised as follows:

a) **Awareness**: Based on the state-of-the-art study of the problem of cloud service selection in the context of cloud service e-marketplace and the various attempts at proffering solutions by existing approaches, the gap in the literature that necessitates this study was identified, presenting an opportunity for contribution. The framed problem is accomplished by research.

b) **Suggestion**: From the study of the literature, key requirements and candidate techniques were identified to develop a selection framework that is suitable for a cloud service e-marketplace and solves the problem identified above.

c) **Development**: Design and implementation of a solution based on key requirements and techniques identified in the previous sub-process.

d) **Evaluation**: Validation and evaluation of the solution developed using established validation methods in software engineering to answer the research questions and test the hypotheses.

e) **Conclusion**: Present the validity of the developed solution and the possibility for generalisation of results.

Within the five-phase research design described above, the research methods employed to carry out this study are a literature review, model formulation, prototyping, and experimentation. Literature survey allows the classification of the existing body of knowledge on the subject matter, while modelling is used to describe real world concept. Prototyping uses an experimental prototype implementation to demonstrate proof-of-concept of the proposed model and experiments are employed to test a hypothesis and arrive at a conclusion. While the summary of research design, objectives, and methods employed in this study is presented in Table 1.1, the research methods employed in this study are details as follows:

1.5.1 Literature Survey

To achieve the objectives of this research and answer the research questions posed in this thesis, a state-of-the-art survey of the literature pertaining to cloud service selection was conducted and six issues were identified and captured as key requirements for a cloud service selection framework suitable for e-marketplace context. Based on the requirements identified, an integrated framework for cloud service selection was formulated.

1.5.2 Model Formulation

The proposed framework christened **F**uzzy-**O**riented **C**lo**U**d **S**ervice **S**election (FOCUSS) framework is presented as an improvement to existing cloud service selection approaches. The framework employed an integration of relevant models and algorithms such as i) feature modelling and constraint-based reasoning - to organize atomic services within the cloud ecosystem and to guide formal service composition on the fly; ii) Fuzzy-based prioritization and analysis methods – to handle subjective user QoS preferences and aspiration; and iii) information visualization – to enable easy comparison of query results along multiple QoS dimensions.

1.5.3 Prototyping

To demonstrate the plausibility and the utility of the proposed framework, a prototype web application within an illustrative case study was undertaken. A collection of tools was identified, categorised into front-end components, and back-end components, with Java as the primary programming language used to implement components of the proposed framework in NetBeans 8.1. The front-end technologies employed consist of JavaServer Pages (JSP) and BootStrap 3.3.6. The bubble graph was implemented using customizable JavaScript classes provided by Google charts for visualising data on web pages. For the back-end components, Java servlets and classes were used to implement the business logic of the proposed framework. The application was deployed in the GlassFish web server.

1.5.4 Experiments

Lastly, an evaluation of the proposed framework was performed using controlled experiments. In software engineering, controlled experiments are one of the three often used validation methods (Wohlin *et al.*, 2012). Simulation experiments and user studies were performed to compare the performance of FOCUSS in terms of scalability, ranking accuracy, and quality of user experience in comparison to existing techniques. The scalability was measured in terms of execution time it took to rank top-k services as the number of alternatives increased. Four ranking accuracy metrics were used as metrics to measure the accuracy of the rank results. User studies were carried out to ascertain the quality of user experience and satisfaction dimensions of the visualisation component of the proposed framework. Thereafter, the results were analysed for statistical significance using a variety of statistical techniques.

Table 1.1: Summary of Research Approach, Objectives, Methods and Activities

Research Phase	Research Objectives	Research Methods	Activities
Awareness and Suggestions	Objective 1	Literature Survey	• Derive taxonomy of cloud service selection techniques. • Identify Key Requirements of an e-marketplace-worthy cloud service selection framework. • Identify candidate techniques for evolving a suitable cloud selection framework
Solution Development	Objective 2	Model Formulation	• Evolve an integrated cloud service selection Framework. • Developed models and algorithms to implement key processes of the framework.
Solution Implementation	Objective 3	Prototyping	• Implement prototype of the framework • Demonstrate the plausibility of the framework using an illustrative case study
Evaluation and Conclusion	Objective 4	Experiments	• Performance Evaluation results for: o Scalability Simulation Experiment o Accuracy Simulation Experiments o User Experience User Studies • Analysis of Results and Generalisation

1.6 RESEARCH CONTEXT

This work was carried out in the context of the GUISET project (Shezi *et al.*, 2006). GUISET is envisioned as both an enabling infrastructure and a suite of on-demand Services. The primary motivation for the GUISET project is economic advantages of enterprise clusters over stand-alone organisation such as resource sharing, cost reduction, and the ability to compete with larger firms (Braun, 2005). As a cloud computing model, GUISET is aimed at offering affordable e-enabling and “appliance-like” technology services through the Internet to lower the total cost of ownership. The GUISET infrastructure would provide small businesses with business-relevant services on a pay-as-you-go basis. These services are aimed at e-enabling the activities of under-resourced local Very Small Software Enterprises (VSSE) and provide the platform for these VSSE to participate in the global market of e-services. VSSE can leverage the capabilities of the GUISET platform to offer business-relevant services, which is then searched for and used by other small businesses that are part the GUISET cloud service ecosystem. In this research, a GUISET service use-case scenario was adopted as the basis of developing and demonstrating the utility and evaluation the cloud service selection approach proposed in this work.

1.7 SIGNIFICANCE OF THE STUDY

The significance of this study is in several folds as follows:

a) This study identifies some gaps in existing cloud service selection literature and proposes a set of key requirements for designing a service selection technique for the cloud service e-marketplace. Cloud e-marketplace platform creators will find these design requirements useful as fulfilling these requirements will both serve the e-marketplace users satisfactorily and facilitate the achievement of the business objectives of the marketplace platform itself.

b) The pursuit of an ecosystem-driven e-marketplace initiative provides a viable platform for local under-resourced small-scale service providers to readily participate in a global ecosystem of e-services. Since the framework proposed in this study encourages variability in the ecosystem, multiple functionally equivalent atomic services can collaborate in service provisioning; thus promoting the profitability of these service providers by multiplying their revenue and economic impact (Venesaar and Loomets, 2006; Hamwele, 2005).

c) The automated analysis reveals a number of useful information about the ecosystem. Therefore, the e-marketplace provider is abreast of the number of composite services that can be offered based on the number of participating atomic services. The provider can also determine those atomic services that will not fully benefit from the value-chain of the ecosystem (partly or fully due to their presence in a few or none of the likely compositions), and advise accordingly.

d) The framework proposed in this study makes it easier to accommodate new atomic services in a manner that is seamless and natural to an e-commerce platform, with little or no disruption to e-marketplace operations. With each case of entrants or exits based on the stated entrance and exit policies on the e-marketplace, such that if the feature model is altered; a seamless automated update of the e-marketplace service directory can still be achieved. This presupposes that service registration and disengagement from the ecosystem is performed offline, not at request time.

e) Existing e-marketplaces (e.g. SaaSMax and AppExchange) readily benefit from the findings made in this study by incorporating visualisation mechanisms to aid effective browsing and comparison of alternatives towards improved and satisfactory decision-making by users of the e-marketplace.

1.8 DELIMITATIONS OF THE SCOPE OF THE STUDY

The functions of a cloud service e-marketplace can be summarized as follows (Menychtas *et al.*, 2014; Bakos, 1998; Vigne *et al.*, 2012; Akolkar *et al.*, 2012): (a) Facilitate a structured platform for service provision and consumption within an infrastructural regulatory framework and policies that enable the efficient operations of the e-marketplace. (b) Match providers' offerings with users' requirements. (c) Negotiate service pricing and conditions associated with service delivery. (d) Facilitating the delivery of services and payments. However, the focus of this study is limited to identifying those cloud services that meet the user's QoS requirements.

1.9 ORGANISATION OF THE THESIS

The rest of this thesis is outlined as follows (see Figure 1.1):

i. Chapter two contains discussions of relevant cloud computing concepts and technologies in relations to cloud ecosystems and cloud service e-marketplaces, and a state-of-the-art survey of the cloud service selection techniques to reveal a set of requirements, which formed the basis for the design of the proposed framework.

ii. In Chapter three, the general overview of the proposed framework, which is modelled as a decision-making framework for cloud service selection in e-marketplace context, was presented. This chapter also presents insights into the strategies of the proposed framework and its underlining assumptions, process model, conceptual architectural framework, and a description of the sub-components.

iii. Chapter four reports the utility of the proposed framework as demonstrated by a prototype Java-based web application with an illustrative case study. This chapter also outlines the limitations of the proposed framework.

iv. Chapter five contains the reports of three experiments conducted to evaluate the proposed framework on account of scalability, ranking accuracy, and user experience; and presents the basis for generalisation of results.

v. Chapter six concludes this thesis with a summary of the findings and opportunity for further research.

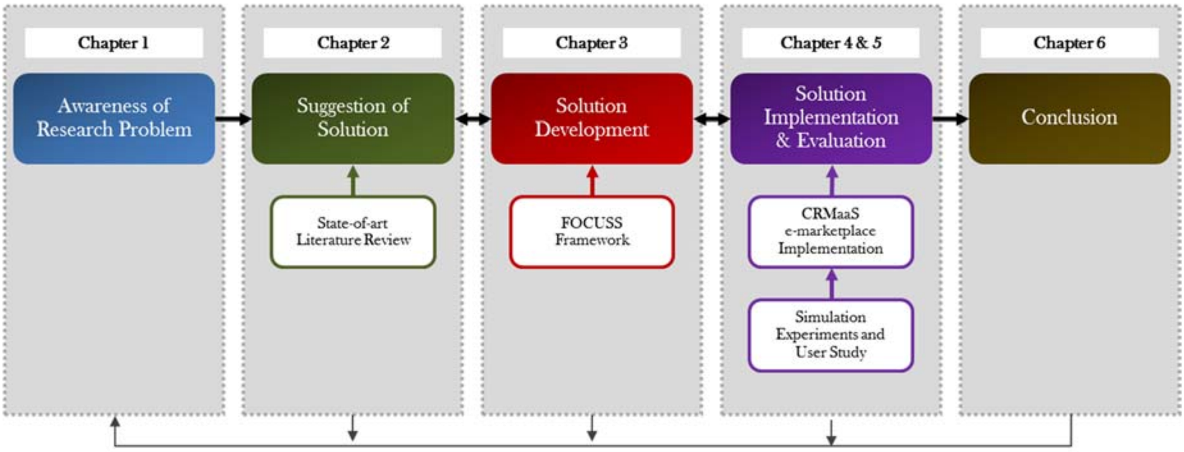

Chapter 1
Awareness of Research Problem
Chapter 2
Suggestion of Solution
State-of-art Literature Review
Chapter 3
Solution Development
FOCUSS Framework
Chapter 4 & 5
Solution Implementation & Evaluation
CRMaaS e-marketplace Implementation
Simulation Experiments and User Study
Chapter 6
Conclusion

Figure 1.1: Outline of the Thesis

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

This chapter is structured into three parts that are closely linked. The first part discusses a general background of cloud computing concepts in relation to cloud ecosystems and cloud service e-marketplaces, and the effect of choice overload on the selection of cloud services. The overview of the background in the first part led to the second part, which discusses related work in cloud service selection and a comparative analysis of existing approaches. The emergent perspectives from the review of related work, which formed the basis of the framework proposed in this thesis, are presented in the last part. A conceptual view of the content structure of this chapter is presented in Figure 2.1.

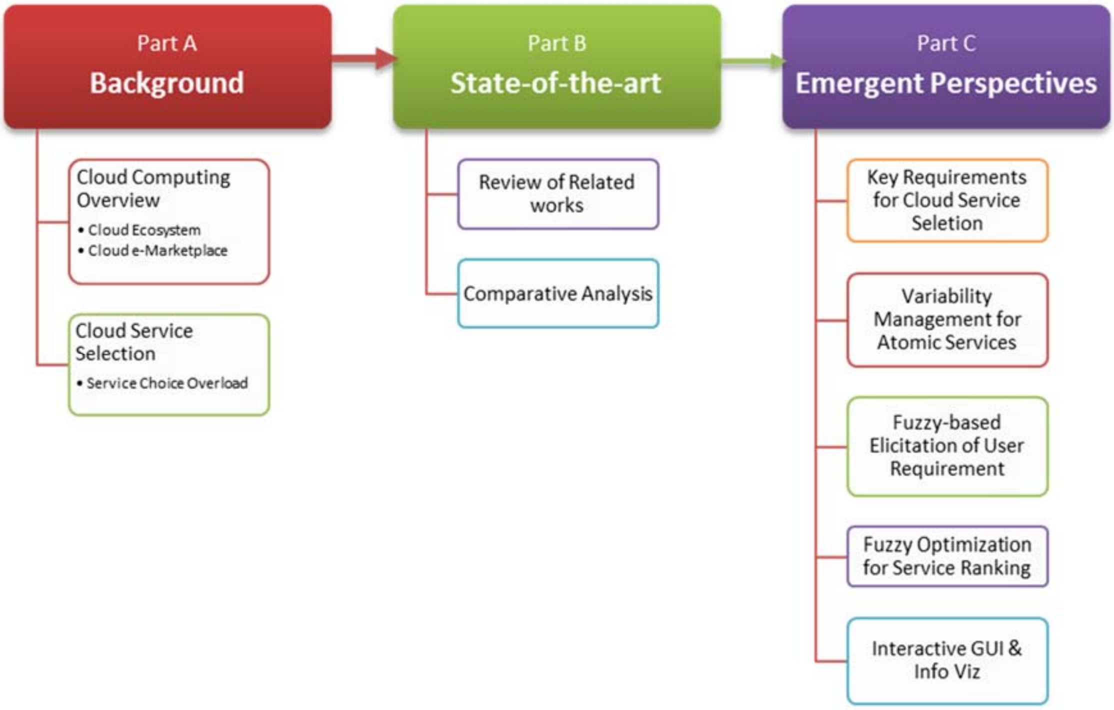

Part A
Background
Cloud Computing Overview
• Cloud Ecosystem
• Cloud e-Marketplace
Cloud Service Selection
• Service Choice Overload
Part B
State-of-the-art
Review of Related works
Comparative Analysis
Part C
Emergent Perspectives
Key Requirements for Cloud Service Seletion
Variability Management for Atomic Services
Fuzzy-based Elicitation of User Requirement
Fuzzy Optimization for Service Ranking
Interactive GUI & Info Viz

Figure 2.1: Conceptual View of the Content Structure of Chapter Two

2.2 CLOUD SERVICES AND CLOUD SERVICE SELECTION

This section contains an overview of cloud computing, cloud services, and the selection of cloud services in the cloud service e-marketplace context.

2.2.1 Overview of Cloud Computing

Cloud computing has recently emerged as a new paradigm for hosting and delivering services over the Internet. Cloud computing is a growing phenomenon in the IT landscape with over $677 billion spent on cloud services worldwide between 2013 and 2016, and about $310 billion spent on cloud advertising in 2014 (Gartner, 2016). Cloud computing has been referred to as the fifth utility along with electricity, gas, water and telecommunication services (Al-Shammari and Al-Yasiri, 2014). Cloud computing, as a paradigm, has the potential to technologically enable new business models, which may not have existed before (Qaisar, 2012). More than ever, new business models are finding relevance in the emergence of cloud computing, which promises an infinite and reliable source of computing facilities on a pay-as-you-go basis over the Internet (Qaisar, 2012). These facilities are hosted and managed by a third party, usually called the cloud providers, and this model of service provisioning introduces flexibility to organisations that rely on such cloud providers' infrastructure (Quinton *et al.*, 2012). This is a radical departure from the traditional IT provisioning models of on-premise computing, in which the computing facilities are owned and managed in-house by an organisation. In the traditional models, organisations make concrete upfront plans on expansions and extensions of these facilities to avoid sudden inability of IT infrastructure to cope with business demands. Furthermore, traditional computing models are characterised by over provisioning and under-provisioning due to the organisations' inability to accurately predict the demands on IT resources per time (Avram, 2014).

Some key attributes of cloud computing that makes it more desirable than traditional models include: *Elasticity*-which refers to the ability to expand and reduce the computing resource as required. *Scalability*-scalability means the ability to handle a sudden increase in demands for processing capabilities, storage and bandwidth as required. *Multi-tenancy*-Multi-tenancy is the ability to share a given cloud resource among many consumers (tenants) seamlessly to make cloud computing economically viable for commercially hosted public cloud providers. *Pay-as-you-go Utility Model*- Cloud computing offers a metered usage scenario in which payment is made only as resources are consumed, rather than a fixed cost associated with acquiring on-premise IT infrastructure.

2.2.2 Cloud Services and Web Services

Cloud services are somewhat similar to web services and many existing cloud services (SaaS, PaaS, and IaaS) are enabled by web services (Sun *et al.*, 2014); (For example, Amazon cloud services are called Amazon Web Service -AWS). Due to the connection between cloud services and web services, a significant body of work has been done in the context of web service selection, not all of which is applicable in the cloud service selection context. According to Sun *et al.* (2014), some key dimensions of cloud computing differ from web services such as:

i. ***Different target user groups***- cloud services are categorised into SaaS, PaaS and IaaS, targeted at different user group;

ii. ***Payment policies***- the cloud supports pay-as-you-use model compared to the fixed pricing model of web services, which adds another layer of complexity to the selection;

iii. ***Evaluation criteria*** – even though quality criteria such as response time, reliability etc. also applies to web services, cloud service is exclusively characterised by other criteria such as eco-friendliness, virtual machine capacity, geographical location etc. (Gatzioura *et al.*, 2012; Jung *et al.*, 2013);

iv. ***Heterogeneity in cloud providers***- a standard means for representing cloud service properties is still in its infant stages, compared to a more established web service description (Sundareswaran *et al.*, 2012).

The differences between cloud and web service paradigms necessitate new approaches for service selection suitable for the cloud environment (Dastjerdi and Buyya, 2011; Sundareswaran *et al.*, 2012; Sun *et al.*, 2014).

2.2.3 Cloud Service Ecosystem

In spite of the promises of cloud computing, some challenges with the current monolithic model require an extension to the current stack. The monolithic model still imposes vendor lock-in such that services cannot be dynamically combined with other services from external third party sources to offer more value-adding functionalities to the users. Papazoglou *et al.* (2011) proposed blueprinting the cloud, a model that allows the syndication, configuration, and deployment of virtual service-based applications in the cloud. Such proposal is followed by the emergence of the cloud ecosystem (Papazoglou and van den Heuvel, 2011). However, the current state of cloud ecosystem does not support the ultimate vision of offering XaaS (Gatzioura *et al.*, 2012), as it is expected that the cloud ecosystem would evolve to offer XaaS in the future.

The advancements in Service Orientation and Virtualization provide the opportunity to fast-track the evolution of cloud ecosystems (Li and Jeng, 2010); as current success in the cloud ecosystem domain, is hinged on the full adoption of a Service-Oriented Architecture (SOA). SOA is an architectural model for application development that supports the use of services as application building blocks (Papazoglou *et al.*, 2007); and services are autonomous, technology-independent software functionalities with prescribed interfaces that can be described, published, discovered, and invoked over a network (OASIS, 2007).

In the context of cloud computing, a cloud ecosystem describes the complex system of interdependent atomic services that work together to enable cloud services. The future of cloud computing would be fast-tracked by successful partnerships and collaborations with multiple service providers to tie services together, and enabling an environment where anything/everything as services are delivered to meet business needs (Baek *et al.*, 2014).

A cloud ecosystem model is a natural way to manage the evolution of cloud computing as an unconstrained model of possibilities and plethora of services available in and through the cloud. Organisations are realising that the competencies and services required to deliver business services cannot be domiciled in one organisation alone, as there is a need to collaborate with other third party providers to make up for required services. Therefore, in a cloud service ecosystem, several heterogeneous cloud service providers across the cloud computing stack come together in ways that are unprecedented to deliver anything/everything as a service (XaaS) that extends the value chain and meets business goals.

A typical example of a cloud ecosystem is Saleforce.com (Salesforce.com, 2000-2015). Salesforce.com is reputed to pioneer the cloud business model based on partnerships. Salesforce.com is a PaaS ecosystem that allows thousands of independent software vendors, developers and consultants contribute to the ecosystem.

2.2.4 Cloud Service e-Marketplace

The popularity of cloud computing will culminate in more service providers joining the cloud ecosystem, interconnecting heterogeneous computing capabilities to co-create value-adding services through composition strategies, to satisfy complex user requirements (Akolkar *et al.*, 2012; Gatzioura *et al.*, 2012). As this trend continues, the need for a platform arises to enable co-creation, showcasing and trading of value-adding service offerings all in one e-marketplace environment (Akolkar *et al.*, 2012). To this end, the future of cloud computing comprises the evolution of cloud ecosystem and the rise of cloud services e-marketplaces for trading cloud-based services; enabling service composition, service discovery, service selection, service deployment, service monitoring, and payment resolutions in a single one-stop shop infrastructure (Menychtas *et al.*, 2014; Akolkar *et al.*, 2012; Gatzioura *et al.*, 2012).

The cloud e-marketplace extends the concept of an electronic e-marketplace, which is a platform where the demand and supply for certain products or services are fulfilled using information and communication technologies (Bakos, 1998; Menychtas *et al.*, 2014; Akingbesote *et al.*, 2014). On this platform, service providers store their offerings, and deploy cloud services capable of integrating with other services to form composite services; while users can discover, consume and pay for service offerings (Papazoglou and van den Heuvel, 2011; Menychtas *et al.*, 2014; Javed *et al.*, 2016; Gatzioura *et al.*, 2012; Akolkar *et al.*, 2012; Vigne *et al.*, 2013; Schulz-Hofen, 2007). The vision of an e-marketplace for cloud services is similar to Amazon.com model, where multiple providers, showcase variety of offerings and an e-marketplace mechanism regulates the interactions and transactions between providers, consumers, and other participants (Akolkar *et al.*, 2012).

The e-marketplace provides a unified view of all available offerings and becomes a single point of access to offerings available in the ecosystem, and hides the complexity of the underlying interconnections among the partners of the ecosystem (Gatzioura *et al.*, 2012). While offering a single portal for interaction for all ecosystem parties, the e-marketplace also integrates a mechanism for managing pricing, revenue sharing, service advertisement and promotion, and billing (Gatzioura *et al.*, 2012; Menychtas *et al.*, 2014).

Examples of commercial cloud e-marketplaces include Windows Azure Marketplace, Amazon Web Service, Google Apps Marketplace, App Store, AppExchange, Android Market, SuiteApp.com, and Zoho (Menychtas *et al.*, 2011). Based on a survey conducted by Menychtas *et al.* (2011), AppExchange was adjudged the most advanced as covering e-marketplace requirements for trading services. AppExchange is the business app store of the Salesforce.com ecosystem, and it expands Salesforce.com's cloud-based Customer Relationship Management (CRM) software into a larger business software portfolio and exposes this portfolio as a collection of services. AppExchange showcases thousands of enterprise and small business applications, and over 1.8 million users shop for services (Apps) from AppExchange (AppExchange, 2015).

In spite of the success of existing commercial e-marketplace, the actualization of the true vision of a XaaS e-marketplace is in its early stages. On-going research provides blueprints and framework to enable the services e-marketplace of the future (Akolkar *et al.*, 2012; Gatzioura *et al.*, 2012; Menychtas *et al.*, 2011; Menychtas *et al.*, 2014). A case to mention is the 4CasST platform (Menychtas *et al.*, 2014; Menychtas *et al.*, 2011). 4CasST is a cloud e-marketplace model that enables an integrated platform for the development and trading XaaS. On the 4CaaST platform, software developers are provided with applications, services and components that simplify the process of building applications and service providers can sell services by a platform infrastructure that support the whole process of an actual e-marketplace transaction.

2.2.5 Key enablers for Cloud Service e-marketplaces

Motivated by a number of existing services e-marketplaces, Akolkar *et al.* (2012) identified six enablers for the realisation of the vision of an electronic emporium of cloud-based services, referred to as the e-marketplace of the future. They include service composition, consumability, social network-driven ecosystem, e-marketplace economy and support, producibility, and intelligence. The detailed descriptions of these enablers are presented as follows:

a) Service Composition

The ability to compose services into more complex business solutions as part of the ecosystem will increase the number of valuable offerings in the e-marketplace. Service compositions can be formal or incidental. Formal composition refers to the combination of one or more services from same or heterogeneous providers; the composite services are offered as a commodity in the e-marketplace, and several instances of the service are created on-demand. The incidental composition is a one-time composition based on a specific user request.

b) *Consumability*

Consumability addresses how easily consumers are able to access services that match their requirements, noting that consumers naturally express such requirements in vague terms that do not necessarily relate directly to actual service descriptions. Therefore, the e-marketplace must possess a deep understanding of consumer requirements in a way that can translate into actual solutions. Desirable are alternatives in the form-based and menu-based interfaces for eliciting requirements; likewise, presenting the results as a table containing a list of alternatives makes it complex for the consumer to fully understand the relationships among these alternatives (Song *et al.*, 2007). Flexibility in expressing requirements is a must for cloud service e-marketplace of the future; and as such incorporates natural language processing, and mechanisms to turn vague and imprecise requirements into actual search queries. Furthermore, the e-marketplace should be able to engage users in a conversation to further elicit details of requirements, allowing for the exploration of candidate solutions, and to perform trade-off analysis, after which an URL can be provided for the consumer to use the service.

c) *Social Network-driven Ecosystem*

This promotes the exchange of information among providers and consumers. The pattern of information exchange is categorised into consumer-to-provider, provider-to-provider and consumer-to-consumer networking. In consumer-to-provider networking, providers can discern popular consumer demands and get consumer's feedback to upgrade or improve their service offerings; while consumers can access the available variety of offerings. Provider-to-provider networking allows the exchange of information among providers to identify opportunities for collaborations to offer more value-added offerings through service composition and can learn from other provider's product reviews to enhance its' own offerings. In networking among themselves, consumers willingly volunteer experiences on services consumed, providing a sufficient basis for other consumers' to make a decision as to engage a particular service. Leveraging on the consumer purchase information, the e-marketplace makes intelligent service recommendations to other users.

d) E-marketplace Economics and Operations Support

This is the e-marketplace's core framework for business and operations support to enable the actual commoditization and commercialization of service offerings. This includes bringing together and managing the underlying computing infrastructure and services to support e-marketplace operations; while providing a mechanism for multi-tenancy, self-service configuration, APIs, pricing management, profile management, billing, payment, monitoring, revenue sharing (particularly in cases of composite services), etc. supporting service provisioning.

e) Producibility

The e-marketplace should provide means for creating and enlisting services on the services catalogue. For example, *Salesforce* provides *Lightning Design System*, *Lightning App Builder* and *Lightning Components*, for developing enterprise apps on the Salesforce platform. These apps are available on AppExchange e-marketplace. Such move would attract more developers to participate in the ecosystem, promoting innovation and value co-creation (Baek *et al.*, 2014). Additionally, providers should be able to publish as much information as possible about the service capabilities, QoS features, and pricing etc. Such information should be machine-readable, which is useful in matching user requirements.

f) Intelligence

The intelligence of a service e-marketplace refers to the e-marketplace ability to know a lot about the semantic properties and capabilities of service offerings either single or composite and what application domain it belongs to (e.g. Insurance, IT, financial, accounting, etc.) and the variations of those services. In addition, it should return precise results to request queries. It should be able to incorporate advancement in Natural Language Processing (NLP), information retrieval and machine learning.

2.2.6 Service Choice Overload

The concept of a cloud service e-marketplace naturally culminates in a plethora of services, with varied quality factors that appeal differently to different users. Service selection in the face of so many options (along multiple decision criteria), without proper articulation of requirements, can be overwhelming, increasing the cognitive demand of the user, and affects user satisfaction of both the process and outcome of decision making (Javed *et al.,* 2016; Iyengar and Lepper, 2000; Schwartz, 2004; Haynes, 2009; Scheibehenne *et al.,* 2010).

The difficulties experienced when selecting from an assortment is referred to as choice overload (or overchoice). According to Alvin Toffler (Toffler, 1970), who first introduced the term, *"overchoice takes place when the advantages of diversity and individualization are cancelled by the complexity of the buyer's decision-making process"*; in other words; the more the number of options, the lesser the motivation to choose or the lesser the satisfaction with the final choice (Chernev *et al.,* 2015; Haynes, 2009). In the context of this thesis, the term *Service Choice Overload* was coined to describe this phenomenon; the consequence of which is that users may end up selecting a suboptimal option or not make any decision at all (Jung *et al.,* 2013; Townsend and Kahn, 2014). Table 2.1 shows the four major factors, classified into extrinsic and intrinsic factors have been identified to impact choice overload in classical choice assortment literature (Chernev *et al.,* 2015).

Table 2.1: Extrinsic and Intrinsic Factors Affecting Choice Overload

Factors		Description	Items
Extrinsic Factors	Decision task difficulty	This includes the structural properties of the decision problem	The number of alternatives available
			Number of attributes describing each alternative
			Time constraints
			Decision Accountability
			Information Presentation Format
	Choice set complexity	This involves the particular value of a choice alternatives or options	The similarity among the alternatives
			The overall attractiveness of the alternatives
Intrinsic Factors	Preference uncertainty	This refers to the extent to which the decision maker has articulated preferences	Knowledge of product and product properties
			The availability of a well-defined ideal point
	Decision goal	This refers to the consumer's goal which involves choosing among the options in a given assortment	Decision intent (buying vs. browsing)
			Decision focus (choosing a set of alternative vs. choosing a particular one)

Source: Chernev *et al.* (2015)

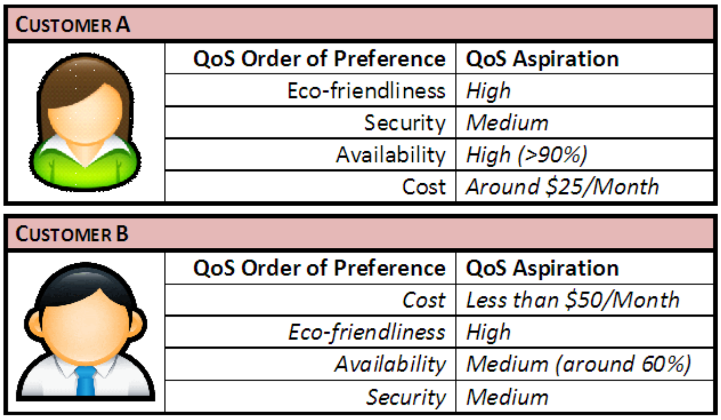
From Table 2.1, extrinsic factors refer to the decision aspect that borders on the structural characteristics of the problem, defined as decision task difficulty and choice set complexity, whereas intrinsic factors pertain to the decision maker in particular and consist of preference uncertainty and decision goal (Chernev *et al.*, 2015). Service choice overload can be minimised by using low cognitive demand decision support mechanisms for eliciting user requirements, in a way that captures the vagueness and uncertainty that characterise human decision making.

2.2.7 Modelling User QoS Requirements

Apart from the capabilities they provide, cloud services possess non-functional or quality attributes classified into technical concerns; for example, reliability, response time, cost, availability; and business concerns- security, usability, eco-friendliness, geographical location and political dimensions etc. (Barros and Dumas, 2006; Gatzioura *et al.*, 2012; Garg *et al.*, 2011; Rehman *et al.*, 2011; Soltani *et al.*, 2012). The measure of these attributes in service usage scenarios, as perceived by the user, is described as Quality-of-Service (QoS). QoS factors represent the non-functional performance of cloud services and are among the key determinants in the selection of cloud services (Chen *et al.*, 2013; Choi and Jeong, 2014), in which the system returns services that meet the required threshold defined by users (Qu and Buyya, 2014). QoS performance information is obtained using an objective and/or subjective assessment. Objective QoS assessment is obtained from QoS monitoring and benchmark testing, whereas subjective assessments are based on user feedback and rating of the service quality after use. Sometimes, service providers can self-publish QoS information as contained in the service-level-agreement (SLA). When service requestors express their expectation from services, they identify functional and non-functional QoS characteristics of the required service; they also have to identify which of the QoS criteria are more important compared to the others. One of the primary ways to model the importance of criteria of user's preferences is to ask the user to weigh each criterion. However, the major drawback of this approach is the complexity of finding proper weight coefficients in the real world applications (Millet, 1997). Furthermore, user's QoS preferences in terms of tendencies have to be considered. For example, it has to be defined whether a parameter value is more desirable for a particular user when it is smaller or greater. In this study, the user's QoS requirements are described in terms of both QoS preferences and QoS aspirations and are discussed in more details below.

a) User’s QoS Preference

QoS preferences are determined by the relative importance given to each service attribute. Since cloud service cannot be evaluated based on one attribute alone, the degree of relevance of each attribute is not the same to the user. The user’s order of preference for each of the attributes contributes to the overall quality of the final option and determines the user’s satisfaction about the option. For example, given the QoS attributes: Cost, Security, Availability and Eco-friendliness; order of preference for the attributes for Users A and B’s is shown in Figure 2.2. The search results should only present service offerings that have duly considered these inputs during preference elicitation (Knijnenburg and Willemsen, 2009).



CUSTOMER A

QoS Order of Preference	QoS Aspiration
Eco-friendliness	*High*
Security	*Medium*
Availability	*High (>90%)*
Cost	*Around $25/Month*

CUSTOMER B

QoS Order of Preference	QoS Aspiration
Cost	*Less than $50/Month*
Eco-friendliness	*High*
Availability	*Medium (around 60%)*
Security	*Medium*

Figure 2.2: QoS Preference and Aspiration for Two Users
User A rates Eco-friendliness as highest priority irrespective of the cost. User B is more budget conscious and is willing to compromise Security for Lower Cost.

b) User’s QoS Aspiration

QoS aspirations define the users’ desired ideal points for each of the service attributes. It comprises the goals and constraints for each QoS criteria. QoS attributes have specific values that define the actual non-functional performance of the cloud service. Users are able to define their own ideal values, and/or constraints on those values, which serve as inputs to generating optimal service alternatives (see Figure 2.2).

It is obvious that QoS preferences and aspiration differ from one user to the other, as shown in Figure 2.2, thereby increasing the complexity of meeting user requirements

(Sahri *et al.*, 2014; Javed *et al.*, 2016). Each user desires to maximise (or minimise) to a certain extent the values of each attribute and requires the most optimal service that meets these requirement thresholds. User's preference and aspiration define utility functions which form the basis for the ranking of service alternatives and ultimately determines which alternative is selected by the user. Moreover, the heterogeneity of service providers and disparity in QoS data of cloud services requires a model that can serve as a basis for comparison and evaluation of services based on user's QoS requirements (Patiniotakis *et al.*, 2014). Hence, a more holistic QoS model of cloud services is required.

2.2.8 Cloud Services QoS Model

A cloud service quality model encompasses the critical aspects and Key Performance Indicators (KPIs) for decision-making to adopt a particular cloud service. The cloud service quality model comprises the important comparable criteria (or metrics) that define each service, the inter-criteria relationships among those criteria. It is used for matching QoS requirements to available services in the service directory (Tajvidi *et al.*, 2014; Gui *et al.*, 2014).

One of the most comprehensive cloud service QoS models is the Service Measurement Index (SMI) (CSMIC, 2014). The Cloud Services Measurement Initiative Consortium (CSMIC) was launched by Carnegie Mellon University to develop the Service Measurement Index (SMI). The SMI is a framework of critical characteristics, associated attributes, and metrics that can be used to compare and evaluate cloud-based services from different service providers (Garg *et al.*, 2013; Garg *et al.*, 2011). SMI was designed as the standard method to measure any type of cloud service (i.e. XaaS) based on the user requirements. The SMI is a hierarchical framework, with seven top level categories, and each category is further broken into four or more attributes that underscore the categories

The seven main categories of the SMI framework include (see Figure 2.3): Accountability, Agility, Assurance, Financial, Performance, Security and Privacy, as well as Usability. The attributes of the various categories are described below:

i. **Accountability**: Accountability refers to a set of attributes used to measure the properties related to the service provider organisation, and may be independent of the services being provided. Securing trust of the user is important to any provider, as users will find it more convenient to use service from a provider that complies with required standards. Attributes like Ownership, Governance, Provider Support, Compliance, and Auditability measure the dependability of the service provider.

ii. **Agility**: Agility indicates how seamlessly, and effectively the service/service provider is able to adapt to changes in user's demand or cloud environment with minimal disruptions or expenditure. Attributes like Adaptability, Elasticity, Extensibility, Scalability, Portability, and Flexibility underscores the agility of a cloud service.

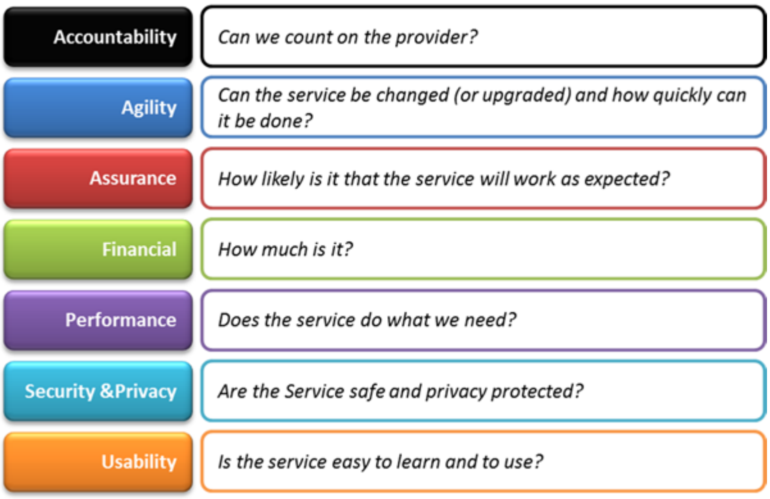
iii. **Assurance**: This category describes key attributes that measure the likelihood that a service will be available as stated. Assurance is made up of the following attributes: Availability, Reliability, Fault Tolerance/ Resiliency, Maintainability, Recoverability, Service Stability, and Serviceability.

iv. **Financial**: Financial indicates the cost of service and how cost effective it is to adopt a particular service/service provider. It is measured by Billing process, Cost, Financial Agility, and Financial Structure.

v. **Performance**: Performance covers the features and functions of the provided services and users need assurance as to how the service meets expected business requirements as claimed. It is measured by Accuracy, Functionality, Suitability, Interoperability and Response time.

vi. **Security and Privacy**: This category includes measures to access the effectiveness of a service provider's control of access to services, data and the physical facilities from which services are provided. This is an important criterion, especially for security-critical applications in finance or health. More specifically, metrics include Security Management, Retention/Disposition, Access control and Privilege Management, Physical and Environmental Security, Data Privacy and Data Loss, Data Integrity, Data Geographic/Political, Proactive Threat and Vulnerability Management.

vii. **Usability**: Usability describes how easy to use a service and it is measured in terms of Accessibility, Client personnel requirement, Installability, Learnability, Operability, Transparency and Understandability.



Accountability	*Can we count on the provider?*
Agility	*Can the service be changed (or upgraded) and how quickly can it be done?*
Assurance	*How likely is it that the service will work as expected?*
Financial	*How much is it?*
Performance	*Does the service do what we need?*
Security &Privacy	*Are the Service safe and privacy protected?*
Usability	*Is the service easy to learn and to use?*

Figure 2.3: SMI 7 Top categories of attributes
Source: CSMIC (2014)

2.2.9 Cloud Service Selection as a Decision-Making Problem

Some cloud services available in the service directory may have similar functionalities with varied QoS dimensions, and the user's choice of these dimensions defines the basis on which the user evaluates available service. The need for this type of evaluation increases the difficulty of making an optimal selection from the list (Zeng *et al.*, 2009; Jung *et al.*, 2013; Garg *et al.*, 2011). For many real world problems, decision making requires that many alternatives be evaluated along some criteria, in order to arrive at the best choice, which is a nontrivial process (Abraham *et al.*, 2005; Bollen *et al.*, 2010). Therefore selecting a service(s) from a cloud e-marketplace can be regarded as a Multi-Criteria Decision Analysis (MCDA) problem, because the properties that define an MCDA problem are similar to the cloud service selection problem (Garg *et al.*, 2011; Gui *et al.*, 2014; Rehman *et al.*, 2011).

MCDA is a popular branch of the decision making and consists of decision alternatives-representing a finite number of available alternatives. These alternatives usually have multiple attributes, and the attributes are the decision criteria (also referred to as goals, interestingness dimensions or objectives) by which the alternatives are evaluated by a Decision Maker (DM). The criteria often conflict (e.g. cost and availability are attributes of a cloud service, a service with low cost may not be high on availability); and the units of measurement are often disproportionate (e.g. cost can be measured in Dollars, while availability is measured in percentage). Furthermore, the criteria may not be of equal priority to the DM, therefore weights are apportioned to determine the degree of importance of each criterion. To this end, an MCDA problem can be defined using a matrix format as described in (Triantaphyllou, 2013):

Definition 2.1: Let $A = \{A_i, for\ i = 1, 2, 3, \dots, m\}$ be a set of decision alternatives and $C = \{C_j, for\ j = 1, 2, 3, \dots, n\}$ be a set of criteria according to which the desirability of an alternative is evaluated. An MCDM problem is to determine the optimal alternative A^+ with the highest degree of desirability with respect to all relevant criteria C_j (See Figure 2.4).

$$\begin{array}{cccccc}
 & \multicolumn{5}{c}{\text{Criteria}} \\
 & C_1 & C_2 & C_3 & \cdots & C_n \\
\text{Alternatives} & (w_1 & w_2 & w_3 & \cdots & w_n) \\
\hline
A_1 & a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\
A_2 & a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
A_m & a_{m1} & a_{m2} & a_{m3} & \cdots & a_{m4} \\
\hline
\end{array}$$

Figure 2.4: A Typical MCDM Decision Matrix
Source: Triantaphyllou (2013)

Multi-Criteria Decision Analysis (MCDA) is a well-established area in the field of operations research and has proven its effectiveness in addressing different complex real-world decision-making problems. The requirements of MCDA are similar across all decision-making methods and includes the following elements- a finite or infinite set of actions, at least two evaluation criteria, and a decision maker (DM) (see Figure 2.4). The goals of MCDA include choosing, ranking, or sorting alternatives (Whaiduzzaman *et al.*, 2014). Typically, it is necessary to use DM's preferences and goals to differentiate the solutions. An *Ideal Solution* is an alternative that has the highest values for all criteria; conversely, an *Anti-ideal Solution* is the alternative that has the lowest values for all criteria. Both ideal and anti-ideal solutions rarely exist in the decision matrix. A more feasible solution is referred to as a Non-dominated Solution. A non-dominated solution is an alternative that is not dominated by any other alternative. For example, an alternative X is said to dominate alternative Y if X is at least as good as Y against all criteria and is better than Y in at least one criterion (Rehman *et al.*, 2012). A non-dominated solution has the property that without sacrificing at least one criterion, it is not possible to move away from it to any other solution.

Service selection is effectively enabled by matching the representations of the user's QoS requirements of the properties of the service offerings (Wittern *et al.*, 2012). With well-articulated preferences and aspiration, the decision maker would be selecting an optimal alternative from the list of all non-dominated alternatives (Aruldoss *et al.*, 2013; Rehman *et al.*, 2012). Therefore, understanding user's QoS requirements, which also include how to both elicit correct priority weights for each criterion (QoS preferences) and actual QoS values (QoS aspirations), is the key to solving an MCDA problem. Decision-making techniques that consider both dimensions are effective for searching and navigating the product/service space in e-marketplace environments (Pu *et al.*, 2011). Many of such techniques abound in the literature, and an exploration of some of these techniques is the focus of the next section.

2.2.10 Approaches to Cloud Service Selection

In this study, the approaches to cloud service selection have been classified into five categories, which include-MCDM-based, Optimization-based, Recommendation-based, Proximity-based approaches, and others. This classification is based on the commonalities among cloud service selection techniques. Figure 2.5 graphically depicts a classification of approaches that have been used for cloud service selection so far in the literature. A detailed overview of each category is presented as follows:

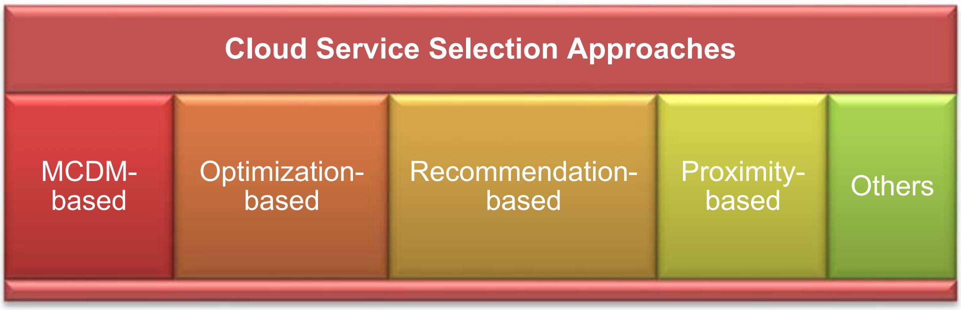


Figure 2.5: Approaches for Cloud service selection

I. MCDM-based Cloud Service Selection Approaches

MCDM-based approaches are also referred to as (Multi-Attribute) Decision-Making (MADM) (Dastjerdi and Buyya, 2011; Whaiduzzaman *et al.*, 2014; Triantaphyllou, 2013), or Multi-Criteria Selection Problems (MCSP) (Rehman *et al.*, 2012). MCDM-based approaches are best suited for scenarios with multiple finite alternatives, known *a* *priori* (Triantaphyllou, 2013; Dastjerdi and Buyya, 2011). The aim is to select one that best satisfies the DM's goals and constraints (Dastjerdi and Buyya, 2011; Sun *et al.*, 2014).

Specific techniques in the MCDM-based approaches for cloud service selection include Multi-Attribute Utility Theory (MAUT), the Analytic Hierarchy Process (AHP), Simple Additive Weighting (SAW), Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), Outranking Methods (e.g. Elimination and choice expressing reality - ELECTRE), Compromise Programming, Min-Max, and Max-Min methods (Rehman *et al.*, 2012). An overview of the popular methods in the MCDM-based approaches is discussed as follows:

a) Multi-Attribute Utility Theory

Multi-Attribute Utility Theory (MAUT) is a value-based model that uses a utility function to aggregate the decision makers' preferences on the decision criteria. The goal of MAUT is to find a certain function reflecting usefulness (or utility) of a particular alternative (Ehrgott *et al.*, 2009). According to MAUT, the overall evaluation $v(x)$ of an object x is defined as a weighted addition of its evaluation with respect to its relevant utility objectives (Schäfer, 2001). The overall utility function is defined as $U(x) = \sum_{i=1}^{n} w_i u_i(x),$ where n is the number of evaluation criteria relevant to the decision problem; w_i represents the weight of the decision makers' preference on the i^{th} criteria; and $u_i(x)$ is the marginal utility for the i^{th} criteria.

b) Analytic Hierarchy Process

Analytic Hierarchy Process (AHP) was developed by Saaty (1988), and it is based on priority theory, founded on mathematics and psychology. AHP is applicable to complex problems that involve the consideration of multi-criteria/alternatives simultaneously by reducing multidimensional problem into one dimension (Saaty and Sodenkamp, 2010). Apart from its application in cloud service selection, AHP and has been applied extensively in problems such as choice, ranking, prioritisation, resource allocation, benchmarking, quality management, and conflict resolution (Forman and Gass, 2001). AHP uses the straightforward mathematical structure of consistent matrices and eigenvectors to determine priority weights of each criterion relative to other criteria (Forman and Gass, 2001; Garg *et al.*, 2013). In contrast to MAUT method, the AHP method uses pairwise comparisons of decision criteria based on the Saaty scale as shown in Table 2.2, rather than utility and weighting functions. Details of the AHP method are available in (Forman and Gass, 2001).

Table 2.2: Saaty's Relative Rating Scale

INTENSITY OF IMPORTANCE	DEFINITION
1	Equal importance
3	Somewhat more important
5	Definitely more important
7	Much more important
9	Extremely more important

Source: Forman and Gass (2001)

c) Simple Additive Weighting

The SAW method is the simplest and one of the most commonly known and very widely applied approaches for solving MCDM problems (Afshari *et al.*, 2010; Chou *et al.*, 2008). It combines the values of criteria and priority weights associated with them into a relevant estimation value used to evaluate each alternative (Abdelhamid, 2012). SAW is also known as a weighted linear combination or scoring methods (Abdelhamid, 2012; Afshari *et al.*, 2010), and is based on a weighted average using the arithmetic mean. An evaluation score for each alternative is obtained by the summation of all the products of the value of each criterion and the weight of relative importance of that criterion (Abdelhamid, 2012). The weights can be assigned directly by the decision maker or obtained by determining the relative importance of each criterion to each other by pairwise comparison prioritisation methods (e.g. Eigenvector method of AHP). The weight assigned to a criterion affects the final score for all alternatives, and also the eventual ranking of alternatives. The linear transformation of the raw data is proportional to the order of magnitude of the standardised evaluations (Abdelhamid, 2012). The strength of the SAW method is its ease of implementation and use (Abdelhamid, 2012). The details of the steps of SAW method are available in (Abdelhamid, 2012; Afshari *et al.*, 2010).

d) Technique for Order Preference by Similarity to Ideal Solution

The Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) method was originally developed by Hwang *et al.* in 1981 (Hwang and Yoon, 1981; Yoon, 1987; Hwang *et al.*, 1993). TOPSIS method ranks as best the alternative that is both closest to the ideal solution (Positive Ideal Solution [PIS]) and far from the anti-ideal solution (Negative Ideal Solution [NIS]) (Abdelhamid, 2012).

The PIS maximises the 'performance' criteria and minimises the 'cost' criteria. In TOPSIS, the decision matrix is first normalised into a dimensionless scale using vector normalisation in order to identify the ideal and anti-ideal solutions. This is done to achieve monotonically increasing or decreasing criteria values that have commensurable units. Next, the distance of each alternative to both the ideal and anti-ideal solution is determined using a similarity or distance metrics. Each alternative is ranked according to the value obtained from the similarity or distance metrics, which is a measure of the relative distances or similarity to both the ideal and anti-ideal solutions. The 'best' alternative simultaneously is one with the shortest distance from the PIS and the farthest from the NIS. The computation involved is not complex compared to outranking methods. The detailed steps of the TOPSIS methods are available in (Jahanshahloo *et al.*, 2006).

e) Outranking Methods

In outranking methods, one alternative is evaluated to be higher than another, or otherwise, denoted by outranking relations derived by pairwise comparison (Bouyssou, 1996; Garg *et al.*, 2013). The underlying principle of outranking method is evaluating the extent to which an alternative dominates another, without necessarily seeking to derive one best alternative (Garg *et al.*, 2013). Outranking Methods compares the performance of alternatives for each criterion and identifies the extent of a preference of one alternative over another, and is applied if the unit of measurement of criteria is incomparable and when it is complex to aggregate criteria metrics (Garg *et al.*, 2013). Besides the technicality of implementation, another drawback of the outranking method is that it does not always arrive at a decision because Outranking Methods allows for the expression of incomparability (Garg *et al.*, 2013; Bouyssou, 1996). Two methods fall under the outranking approaches: ELECTRE (Benayoun *et al.*, 1966; Roy, 1991) and PROMOTHEE (Brans *et al.,* 1986).

II. Optimization-based Cloud Service Selection Techniques

Generally, the application of optimisation approaches in decision making usually favours scenarios with a large set of alternatives. These alternatives are often times not known a priori (Triantaphyllou, 2013; Dastjerdi and Buyya, 2011). The aim is to select an alternative that best satisfies the decision maker's preference, goals and constraints, by minimising or maximising one or several criteria (Dastjerdi and Buyya, 2011; Sun *et al.*, 2014). The constraints imposed by a decision maker demands that the preferable alternative minimises or maximises one or several criteria while observing the imposed constraints (Dastjerdi and Buyya, 2011).

The cloud service selection problem has been formulated as a Constraint Satisfaction Problem (CSP), Multiple Choice Knapsack Problem (MCKP) and its variants, tree-search problem etc. The solutions to these optimisation problems are either optimal solutions or near-optimal solutions and employed the use of heuristics, greedy algorithm, evolutionary algorithm etc. (Dastjerdi and Buyya, 2011; Sun *et al.*, 2014).

III. Recommendation-based Cloud Service Selection Techniques

Cloud service selection has also been formulated as a recommendation problem. The field of recommendation is concerned with assisting consumers to deal with information and choice overload by providing more personalised items recommendations (e.g. products or services) from a large assortment of items. Recommendation techniques have been applied in recommender systems, which are a type of decision-support systems that leverage historical data on consumer, consumer preferences, and items, to provide recommendations (Han *et al.*, 2009).

Recommender systems have been successfully deployed in e-commerce, movies and book retail and rental sites, with success (e.g. Amazon.com, Netflex.com) and have been adapted to the domain of cloud service selection also. Cloud service selection approaches based on recommendation proposed service alternatives to a potential user based on the similarity between existing/previous users of that service and the potential user. There are basically two types of filtering techniques in the recommendation, collaborative filtering or content-based filtering approaches, together with a hybrid of the two techniques.

Collaborative filtering approaches recommend to the current user, items that other users with similar tastes (ratings) liked in the past. The similarity in the rating of two users is calculated based on the similarity in the rating history of the users. The drawbacks of the collaborative filtering approach are cold-start and data sparseness (Han *et al.*, 2009). In content-based filtering approach, the system learns to recommend items that are similar to the ones that the user liked in the past. The similarity of items is calculated based on the properties associated with the compared items. Collaborative and Content-based filtering approaches are often combined with hybrid approaches for more effective recommendation result.

IV. Proximity-based Approaches

Proximity-based cloud service selection approaches employed similarity or distance metrics to rank cloud services. The similarity metric is a measure of proximity between two or more objects or variables (Ayeldeen *et al.*, 2015). A number of cloud service selection methods are based on such proximity-driven scheme that explores the similarity between the QoS attributes of the user's requirements and the features description of specific cloud services in order to rank them (Mirmotalebi *et al.*, 2012). The most popularly used distance metric for cloud service selection in the literature is the Euclidean distance metrics and its variants.

V. Other Cloud Service Selection Approaches

Apart from the cloud service selection approaches discussed in preceding sub-sections, some methods for cloud service selection can be classified according to specific methodologies used to rank cloud services. A number of these methods employ semantic models, that includes the use of ontologies or specific data model to represent cloud service QoS information. These methods also use logic-based techniques, like constraint programming, to reason on the models in order to evaluate or rank cloud services with respect to users' requirements (Sun *et al.*, 2014).

2.3 STATE-OF-THE-ART IN CLOUD SERVICE SELECTION

This section contains the review of the state-of-the-art in cloud service selection, as well as a comparative review of existing works in the literature.

2.3.1 Review of Cloud Service Selection Techniques

Cloud service selection techniques provide means to capture decision alternatives, elicit and interpret user QoS requirements, evaluate and ranks alternatives, according to user requirements, and present results to users in a manner that is easy to understand. While these techniques can be distinguished by their support for handling fuzziness or subjectivity in QoS information, in this section, various techniques have been classified into the following five categories. These categories include the following:

i. MCDM-based approaches

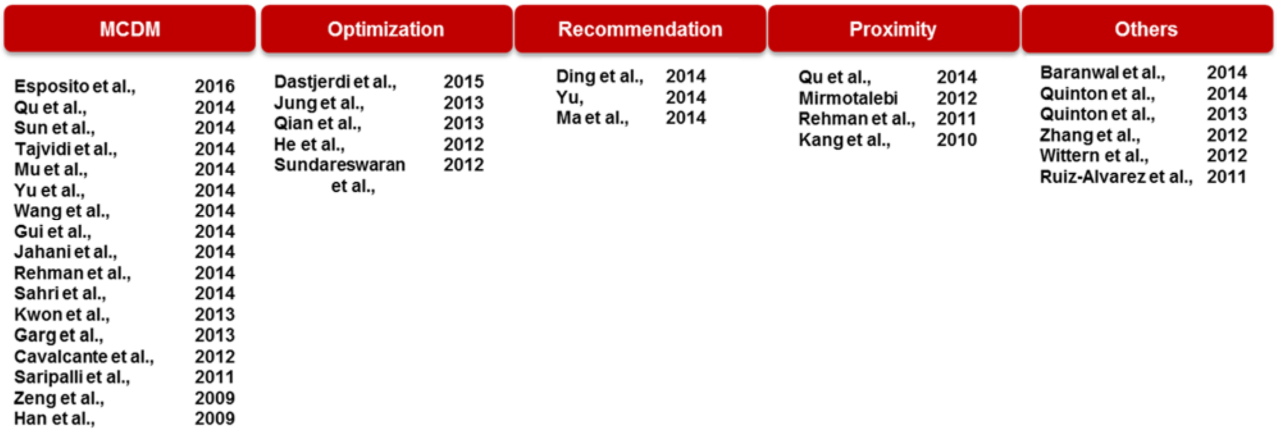
ii. Optimization-based approaches

iii. Recommendation-based approaches

iv. Proximity-based techniques approaches

v. Other approaches

Figure 2.6 summarises the techniques in the literature grouped under each cloud service selection approach.



MCDM		Optimization		Recommendation		Proximity		Others	
Esposito et al.,	2016	Dastjerdi et al.,	2015	Ding et al.,	2014	Qu et al.,	2014	Baranwal et al.,	2014
Qu et al.,	2014	Jung et al.,	2013	Yu,	2014	Mirmotalebi	2012	Quinton et al.,	2014
Sun et al.,	2014	Qian et al.,	2013	Ma et al.,	2014	Rehman et al.,	2011	Quinton et al.,	2013
Tajvidi et al.,	2014	He et al.,	2012			Kang et al.,	2010	Zhang et al.,	2012
Mu et al.,	2014	Sundareswaran et al.,	2012					Wittern et al.,	2012
Yu et al.,	2014							Ruiz-Alvarez et al.,	2011
Wang et al.,	2014								
Gui et al.,	2014								
Jahani et al.,	2014								
Rehman et al.,	2014								
Sahri et al.,	2014								
Kwon et al.,	2013								
Garg et al.,	2013								
Cavalcante et al.,	2012								
Saripalli et al.,	2011								
Zeng et al.,	2009								
Han et al.,	2009								

Figure 2.6: Taxonomy of cloud service selection techniques

I. Review of MCDM-based Cloud Service Selection Techniques

A systematic framework to filter, evaluate and select cloud services was proposed in (Gui *et al.*, 2014). Specifically, the framework comprises a hierarchical information model for bringing together disparate cloud information from a variety of providers; a cloud service classification model; a schema for generating rules for creating specific solutions; a dynamic preference-driven evaluation model that recommends service solutions based on application's provider preferences; and visually communicate a comparison of solutions through an interactive user interface. The service evaluation is performed using MAUT-based and TOPSIS-based techniques. Another proposed scalable service selection algorithm that considers user preferences for optimal performance at minimum cost is presented by Zeng *et al.* (2009). The service selection algorithm proposed computes the service cost and gains, as the user only needs to specify two goals (maximum gains or performance and minimum cost). The service proxy will then review the service attributes and select the optimal service that aligns with the goals specified by the user. The proxy selects all related services from the cloud service repository, evaluates the services' state and availability and based on a SAW technique, aggregated score of each service against a threshold. The proxy then computes the performance and cost utility functions and ranks the optimal services that satisfy the goal of the clients using an MAUT. CloudIntegrator is another MAUT-based approach that performs service composition by searching for services that fulfil the activities designated in a workflow and generates candidate execution plans as an orchestration of a set of actual services (Cavalcante *et al.*, 2012). The proposed algorithm employs the cost and the metadata of services' QoS parameters to optimise the selection process by first filtering out what the authors called coincident services. They described coincident services as services that are always part of any execution plan, contributing to any execution plans, in terms of cost and quality values. The authors argue that this filtering would reduce the time it takes to select services since the evaluation process considers fewer services. The identification and removal of coincident services precede the actual service selection process, while the process itself involves computing the global cost and quality values based on each QoS dimension and then combine these values in MAUT-based technique to rank and select the alternatives with maximal utility value.

Some approaches based on aggregated weighted sum include Cloud service recommender system (CSRS) and Multiple Attribute Decision Methodology for Adoption of Clouds (MADMAC). CSRS is a cloud service selection framework proposed for the cloud market (Han *et al.*, 2009). The CSRS is based on a Service-Rank (S-Rank) algorithm that ranks services with respect to user requirements. S-Rank value is the weighted aggregate of quality of virtualization hypervisors, QoS values, and user feedback ($S - Rank_{final} = \alpha * e^{VM_{factor}} + \beta * e^{QoS} + Uf$), and services can be selected based on the result of S-rank after applying cost filters. MADMAC is a cloud adoption framework that utilizes a careful description of attributes, alternatives and priority weights on attributes in order to build a decision matrix, for generating relative rankings in identifying the optimal alternative (Saripalli and Pingali, 2011). MADMAC uses a modified Wideband Delphi method for determining relative weights for each QoS attribute and rankings are achieved using the SAW that incorporate these weights. Wideband Delphi is a highly moderated iterative convergent expert opinion survey, used to collect input from subject matter experts to determine unanimity on the relative importance of the weights.

Rather than considering QoS evaluation results in real-time or average historical QoS information of cloud services when recommending a service alternative as the best, the approach presented by Rehman *et al.* (2014) utilises the QoS history of cloud services from different time periods. A parallelized MCDM-based method is used to rank all cloud services in each time period with respect to users' preferences before combining the results used in ranking the alternatives. Rehman *et al.* (2014) argued that utilising an average historical QoS hides the frequent variations in QoS performance, and real-time QoS monitoring does not consider the performance history; hence may yield a sub-optimal alternative in both cases. The approach integrates users' preference information in a TOPSIS and ELECTRE-based approach to rank services at different non-overlapping time slots. The evaluations at each time slot are independent of each other and are executed in parallel after which the results are aggregated to determine the overall best alternative. The entropy method used in information theory was employed to estimate the relative weights of the importance of the criteria (Wang *et al.*, 2007).

Since the interdependence between each QoS attribute affects the service evaluation process, and its impact on overall ranking depends on their eventual priority weight in the overall selection process, Garg *et al.* (2013) proposed SMICloud, an approach based on SMI QoS model and uses historical QoS measurements, combined with self-published QoS information from service providers to derive the actual QoS values. The SMICloud is an AHP-based implementation that assigns weights to QoS attributes by considering the interdependence between them, thereby providing a quantitative basis to rank cloud services. In the same vein, DBaaS-Expert is an AHP-based framework proposed to assist in choosing the right DBaaS provider among several Database-as-a-Service (DBaaS) offerings (Sahri *et al.*, 2014). The DBaaS-Expert framework consists of an ontology modelling and a ranking module. The ontology model is employed to capture the concepts of data management systems such as workload type, data model etc. The ranking module based on AHP to rank DBaaS offerings according to quality, capacity and cost of service dimensions. After the user submits a query, the list of DBaaS offerings that does not match user requirements is filtered out, while the pruned list is then ranked based on priorities assigned to each criterion by the user. The Weight Service Rank

(W_SR) approach for cloud service ranking, proposed by Jahani *et al.* (2014), is similar to the Min-Max algorithm elaborated by Rehman *et al.* (2012), and it compares the different services based on user defined preference on QoS, so as to select the most optimal service. When compared to AHP, the performance of the W_SR approach showed a significant computational advantage.

A number of MCDM-based approaches do consider uncertainty information in the service evaluation process. Specifically, a cloud service selection model was proposed by Mu *et al.* (2014), which combines both the uncertainty inherent in user's subjective preference information and objective weights. In this approach, subjective weight preferences are explicitly expressed by users using linguistic terms and these inputs are processed using intuitionistic fuzzy set theory. The objective weight preference is useful when the user has no knowledge of the preference and based on the user's incomplete history of preference information on that service; the rough set is used to derive objective weights. The aggregation of the subjective and objective weights is integrated with TOPSIS to obtain a ranking of the alternatives. Wang *et al.* (2014) introduced an approach to accurately evaluate the QoS of cloud services for a service-oriented cloud computing context. The approach employs fuzzy synthetic decision to estimate cloud services in accordance with users' preferences and computes the uncertainty of cloud services based on monitored QoS data. After which final evaluation of cloud service is obtained using fuzzy logic control. A personalised trust evaluation system to support IaaS selection is proposed by Qu and Buyya (2014). The approach measures the trust of cloud services as the degree of satisfaction of specific user requirements based on past QoS performances. Membership functions and fuzzy hedges were used to elicit users' subjective QoS requirements and generated trust levels for each cloud service through a hierarchical fuzzy inference system.

In order to address uncertainty in the input into MCDM-based evaluation process and the evaluation itself, such as uncertainty in service requests, QoS descriptions, user preferences, Sun *et al.* (2014) proposed a hybrid fuzzy MCDM-based framework for cloud service selection that uses fuzzy-ontology for function matching and service filtering. Based on the pruned alternatives, a Fuzzy-AHP technique was adopted to derive informed criteria weights based on vague expression, and, together with fuzzy TOPSIS approach, the fuzzy weights were used for service ranking based on fuzzy descriptions on service performance. In the same line of work, Kwon and Seo (2013) present an IaaS selection model based on Fuzzy-AHP, to enable the user to select a suitable service provider that aligns with the goals of the company. Furthermore, Tajvidi *et al.* (2014) proposed a four-phase fuzzy-based multi-criterion decision-making framework that works with cloud service data gathered from third party runtime QoS monitoring tools, together with user feedback about the past performance of services. This approach handles the imprecision in user's QoS preferences by capturing the linguistic weight of criteria using fuzzy logic, which then converts the triangular fuzzy numbers into precise numbers. These numbers were later used in the ranking algorithm, located in the service selection process module. This module has two components, metrics calculation and ranking, and the ranked resulted was presented to the user via the user interface layer. Complementing the hierarchical SMI cloud QoS model, this approach employed a fuzzy AHP-based technique to rank cloud services. The ranking is based on the fuzzy perception of users' preferences on QoS dimensions, expressed as weights derived using the Buckley's method (Buckley, 1985).

An approach was presented by Esposito *et al.* (2016) to handle uncertainty in users' QoS preferences in the face of untrustworthy indications concerning the QoS levels and prices of services posed by selfish providers. The approach, based on multi-objective optimisation, maximises the satisfaction and minimises the cost based on user requirements. The proposed approach uses fuzzy set theory to handle uncertainty in users' subjective preferences to derive priority weights and employs a TOPSIS-based method to rank the alternatives. The approach further integrates the *Dempster-Shafer theory of evidence* to perform a distributed selection of services; and a *Mechanism Design*, based on game theory to reveal actual QoS performance evaluation of service offerings; which the authors' believe promotes truth-telling among service providers. The distribution of the selection process is motivated by the limitation of the centralization of the overall process, which often results in performance bottleneck that reduces the efficiency of the overall infrastructure.

Apart from the single cloud user, there are scenarios where a cloud service is to be selected based on the preferences of members of a group, and the service selected must optimise all preferences of members of that group. To solve this problem, a QoS-aware SaaS Services Selection with Interval Numbers for Group User (QSSSIN_GU) is proposed by Yu and Zhang (2014). The approach integrates vague QoS preferences of members of a group in the evaluation process using Interval Numbers (IN). The authors argue that the vagueness in QoS preferences of group users can be expressed in a range of values, using IN. Since the QoS preference of the member of a group varies, the use of IN can conveniently capture the variety of QoS preferences and obtain a collective satisfactory ranking. To normalise the varying dimensions of QoS properties, QSSSIN_GU applies a linear scale transform normalisation function to ensure that the range of normalised interval numbers belongs to [0, 1]. QSSSIN_GU applies TOPSIS to rank and identify the most optimal alternatives.

Following the review of the MCDM-based techniques, it is observed that none of them provided a means to organise or aggregate atomic services to meet composite user request. Also, a number of these techniques require users to express their requirements using crisp or exact entities. Some other MCDM-based techniques elicit either subjective QoS preferences or QoS aspirations but do not elicit both subjective QoS preferences and aspirations. Hitherto, the most techniques did not include a user interface to elicit those requirements nor provide a means to visualise the ranking results to simplify decision making.

II. Review of Optimization-based Cloud Service Selection Techniques

Noting that the number of service alternatives is very large in a cloud service marketplace, Sundareswaran *et al.* (2012) proposed a brokerage model that uses a unique indexing technique for handling the large information from a large number of services and efficient service selection algorithms that rank potential service providers. The cloud broker analyses and index providers, according to similarities in their properties using the B+-tree as the base structure. A *k*-means algorithm is used to group all the service providers according to the Hamming distance between the encodings of the service information, after which the concept of iDistance is used to generate the indexing key to index service points as data points on the B+-tree. The indexing enables efficient arrangement of services in a way that the speed of retrieval is enhanced. A simplified GUI is provided to facilitate requirement elicitation and based on those requirements the broker will search the index, using a greedy algorithm, to generate a ranked list of candidate services (single or composite) that best match user requirements.

CloudAdvisor enables interactive exploration of various cloud configurations and recommends optimal configurations in line with the users' workload and preferences (Jung *et al.*, 2013). The preference dimensions include budget, performance expectation, and energy saving for a given workload. It also allows the comparison of a present configuration to other cloud offerings. The approach includes an easy to use interface for specifying preferences and making a comparison such that the user need not specify preferences in crisp terms. The estimated near optimal configuration is determined using a constraint optimisation method that considers user's preferences, availability of resources, and dependency of proper hardware and software. The constraint optimisation problem is solved using A* search algorithm, while the comparison of current configuration to other near-optimal configurations offered by other providers are formulated as a knapsack problem, solved by a benchmarking based approximation technique based on a greedy algorithm.

A greedy algorithm was also employed in the MSSOptimiser (Multi-tenant SaaS Optimiser) approach (He *et al.*, 2012). The multi-tenant nature of cloud services, in which a single computing resource is shared by a large pool of users, necessitates that a multi-tenant SaaS serves same functional SaaS to multiple end-users with varying QoS requirements. The decision process to customise and deploy SaaS for multiple tenants is complex; more so, because SaaS developers usually composed services with varied QoS to fulfil end-users' requirements in a way that optimises the cost of resources with the best system performance. Since existing QoS-aware service selection approaches are targeted at a single tenant, MSSOptimiser (Multi-tenant SaaS Optimiser) (He *et al.*, 2012) is proposed to overcome this limitation. MSSOptimiser capture and model users' QoS requirements and constraints; and both assist in selecting services to be composed into SaaS that approximates the QoS requirements, while generating a near optimal deployment environment that minimises the cost of resource usage and maximises overall SaaS performance irrespective of usage cost. The selection problem is formulated as a constraint optimisation problem, which employed a greedy algorithm to efficiently find a near optimal solution.

CloudPick simplifies cross-cloud deployment via QoS modelling and deployment optimisation (Dastjerdi *et al.*, 2015). Ontology-enriched cloud service description can be discovered with improved accuracy, particularly considering QoS descriptions from a variety of domains. CloudPick uses two deployment optimisation algorithms based on genetic and Forward-Checking-Based Backtracking (FCBB) algorithms to deploy networks of virtual appliances based on minimum cost, high reliability and low latency. Through CloudPick, the deployment optimisation is expected to yield the near optimal configuration (combination of cloud virtual machines) that optimises the cost of data communication, latency and reliability between multiple clouds based on user preferences. The VM configuration is achieved through the aggregation of multiple cloud services. Qian *et al.* (2013) argued that proximity plays an important role in choosing IaaS, and designed an approach called Cloud Service Selection (CSS), which considers the location of IaaS cloud infrastructures, the application clients, and how the intercommunication among application components affect IaaS selection. The approach manages the scalability issue arising from a large number of data centres and applications by introducing a heuristic-based stepwise application placement optimisation algorithm that is able to discover near optimal solution in a short time, with the objective of minimising cost and maximising high QoS performance of the applications. The trade-off between cost and proximity is determined by assigning importance weights.

The review of the optimization-based cloud service selection techniques revealed that these techniques scarcely provided means to compose atomic services or consider subjective user requirements; instead, the techniques rely on definite or exact QoS preference provided by the user. In addition, only the techniques proposed by Sundareswaran *et al.* (2012), Dastjerdi *et al.* (2015), and Jung *et al.* (2013) included a user interface to elicit user requirements; meanwhile just the technique presented by Jung *et al.* (2013) technique integrates a visualization mechanism to explore alternatives.

III. Review of Recommendation-based Cloud Service Selection Techniques

Since the cold start problem inherent in collaborative filtering and differences in client-side context (location, device, or integrated development environment [IDE]), the accuracy of QoS evaluations and feedback cannot be uniform as its best to express such variation in a range rather than real, binary or integer numbers. To this end, Ma and Hu (2014) proposed RecTIN, a cloud service recommendation approach to cater for this variation by using ternary interval numbers (TIN). TIN enabled the description of QoS evaluations from existing users in order to determine the QoS trustworthiness of a cloud service for potential cloud service users. *K-means* clustering algorithm was employed on the basis of multi-attributes trust aggregation, which uses Fuzzy-AHP to rank TIN while selecting trustworthy services.

The trustworthiness of a cloud service will be in question if the feedback from service usage is at variance with the expectations on such service expectations. Therefore, trust is recognised as a key point of consideration in cloud service selection. Sometimes, the information that determines trust degree of service is determined through objective and/or subjective feedback assessments. Objective assessment is obtained from QoS monitoring and benchmark testing, whereas subjective feedback is obtained from user rating of the service quality. Adopting either assessment approach has inherent drawbacks. Specifically, it is difficult to evaluate the qualitative aspect of the services using objective assessment; whereas subjective assessments are based on the subjective feelings of the cloud user, and may contain biases and also depends on the context of the user. Considering the fact that many trustworthiness evaluation problems require both objective and subjective assessments some cloud service recommendation approaches have combined both assessment methods.

CSTrust, proposed by Ding *et al.* (2014), is a framework for determining the trustworthiness of cloud services by combining QoS prediction obtained from objective assessment, and subjective user satisfaction estimation. CSTrust uses collaborative filtering and a utility function, referred to as Constant Relative Risk Aversion (CRRA), to improve the accuracy of QoS value prediction, by predicting the missing QoS value of quantitative attributes from the previous usage scenario of other similar services. Furthermore, Yu (2014) advocated that sole dependence on the performance evaluation reports from the service providers or experts is not in alignment with the distributed nature and openness of the cloud. CloudRec is proposed as a cloud selection framework that utilises a user-focused strategy for personalised QoS evaluation of cloud services (Yu, 2014). CloudRec is able to use an iterative algorithm on community-based QoS assessment model to discover a set of similar user and service communities from scarce and large-scale QoS data, as users connect to approximate the QoS values of unknown cloud services. CloudRec employs the Regularised Posterior Probabilistic Nonnegative Matrix Factorization (RPPNMF). Since RPPNMF is able to handle data scarcity characteristic of a cloud environment, it is used to capture the inherent cloud-related features, and group cloud services and its users into communities based on this feature.

Arising from the review recommendation-based techniques for cloud service selection, the following observations were made: apart from Ma and Hu (2014), all the recommendation-based techniques expect crisp QoS inputs from the users. Moreover, none of the techniques aggregates atomic service to form composite offerings, in addition to built-in means by which user requirements can be elicited and the mechanism to visualise the services recommended.

IV. Review of Proximity-based Cloud Service Selection Techniques

Mirmotalebi *et al.* (2012) argue that modelling users' online behaviour would profit search engines as well as e-commerce sites and those benefits could be extended to the software service selection context. According to Mirmotalebi *et al.* (2012), ranking services would be more satisfactory when users' preferences are understood, and the authors proposed an approach to generate a personalised ranking of cloud services based on both explicitly stated and implicitly determined user preferences on non-functional properties. While the explicitly stated requirements are clearly expressed by the decision maker, the implicit preferences are determined based on information from a stored user profile of the decision maker of past usage. The approach by Mirmotalebi *et al.* (2012) assumes the existence of an exact matching algorithm and the personalised ranking is computed as the similarity between user's non-functional preferences and the values of the non-functional properties of services. Services with higher matching scores with the user's profile are ranked higher in the result list.

The need for a search engine for cloud services motivated the work of (Kang and Sim, 2010), in which *Cloudle* was proposed. Cloudle is a multi-criteria search engine for cloud services with a matching algorithm for cost, technical and functional requirements. The search engine's accuracy is powered by a cloud ontology model, which is designed to determine similarity among cloud services, based on the following similarity dimensions- *concept, object property* and *datatype property* similarities. The functional aspects of Cloudle include *Query processing module*, where the user query is received and processed via a web page and sent to the *Similarity Reasoning Module* to perform similarity reasoning. The query is also sent to the price and timeslot utilise *Matching Module* to determine which services match the price and time slot. Finally, in the *Rating Functional Module*, each service from the providers is evaluated based on a utility score.

The service with the highest utility score is ranked as the best match and the search result is presented as a textual ranked list of cloud services.

Based on the formal description of the cloud service selection problem, Rehman *et al.* (2011), proposed two weighted sum-based cloud service selection methods (*Weighted Difference* and *Exponential Weighted Difference*) that compute the similarity between two vectors representing user requirement criteria and each service's properties. Based on the similarity index, the service whose properties best match user requirements is selected as the best. Three comparison cases were identified which include 1) Exact match between properties vector and user requirement vector. 2) Properties vector has (generally) lower values than user requirement vector. 3) Properties vector has (generally) higher values than user requirement vector. The Weighted Difference (WD) approach is a sum of the weighted difference between the criteria of the user and service properties ($Sim(UserReq, Ser) = \sum_{i=1}^{n} w_i * (UserReqVect_i - ServiceDesVect_i)$); while the Exponential Weighted Difference (EWD) overcomes the drawback of WD in that the criteria in which the service's properties is below the user requirement is balanced by those exceeding user requirements. EWD ($Sim(UserReq, Ser) = \sum_{i=1}^{n} e^{-w_i * (UserReqVect_i - ServiceDesVect_i)}$) utilizes an exponential function to limit the effect of mutual elimination between criteria that is below or exceeds the user requirement.

Qu *et al.* (2014) proposed a context-sensitive service selection model that compares and aggregates subjective assessment extracted from the feedback of previous cloud service users and objective assessment obtained from quantitative performance testing. Biased subjective assessment is eliminated by objective assessment; while both subjective and objective assessments and their context information (relating to time-based and location-based contexts) are combined in evaluating the global performance of cloud services with respect to personalised requirements of a potential user. The comparison is performed by using a modified bipartite *SimRank* algorithm to compute the context similarity of the objective and subjective assessments, so as to dynamically adjust the benchmark level, in order to enhance the exactitude aggregation process to reflect the total quality of cloud services. Based on the rating matrix obtained, potential user's preference is acquired via linguistic weights and converted to fuzzy numbers to determine the importance weights assigned to both objective and subjective attributes. Services ranking is then achieved using fuzzy-SAW computation.

Resulting from the review of proximity-based cloud service selection techniques, it is observed that just two of the techniques consider either subjective QoS aspiration (Qu *et al.*, 2014) or subjective QoS preferences (Mirmotalebi *et al.*, 2012) in the evaluation of service alternatives with respect to user requirements, as well as integrating a user interface to elicit user requirements. So far, the techniques in this category did not include any visualisation mechanism nor focused on the composition of atomic services to meet complex user requirements.

V. Review of Other Cloud Service Selection Techniques

An extensible approach for cloud storage service selection was proposed by Ruiz-Alvarez and Humphrey (2011). The approach is used to select the service that best matches each dataset of a given target user application by relying on XML schema containing service capabilities and attributes of each cloud storage system. The XML schema is algorithmically processed using a matchmaking framework based on the work of Raman *et al.* (1998) to match services and users' requirements, such that data storage recommended satisfies users' requirements of availability and durability, meets performance expectations of latency and throughput, and with corresponding cost estimates. Based on the SMI QoS model, Baranwal and Vidyarthi (2014) applied ranked voting method for ranking and selecting cloud services combined with Data Envelopment Analysis (DEA) technique. In ranked voting methods, voters rank the alternatives in order of preference. More specially, the approach considers each QoS criteria as voters, and the cloud providers are alternatives to be voted for. Since DEA suggests more than one optimal alternative, additional rank voting techniques are required to discriminate optimal alternatives. However, the ranking order is usually affected by the information about other non-optimal alternatives. The approach presented here is formulated as a linear programming model (Obata and Ishii, 2003). The model augments DEA with a rank voting technique, while eliminating inefficient candidates, and identifying efficient candidates derived from the DEA in order to consequently determine the best alternative.

CloudRecommender, proposed by Zhang *et al.* (2012), is a declarative approach for selecting Cloud-based infrastructure services. In CloudRecommender, cloud service configurations are captured in an ontology-based data model and manipulated using regular expressions and SQL. The domain knowledge representing a variety of infrastructure service configurations is identified and formalised by a declarative logic-centred language and implemented as a recommender module atop a relational data model. CloudRecommender work based on transactional SQL queries semantics used to query, insert, and delete infrastructure services' configurations. Users interact with CloudRecommender via an intuitive widget-based interface both to set criteria, and to browse recommendation results.

Furthermore, the cloud ecosystem involves the interplay of a wide variety of cloud capabilities at a different scale of functionalities that must be correctly combined or configured by a variety of stakeholders for the application to work efficiently (Quinton *et al.*, 2014). The plethora of cloud providers and the variability among cloud services usually increases the complexity and the error propensity of configuration choices made in an ad hoc manner (Quinton *et al.*, 2014). Software Product Line (SPL) Engineering is a software engineering approach that supports the systematic reuse of software assets in a pre-planned way to achieve quick, cost effective and quality software products. It enables the effective capture of the commonalities and variabilities of software artefacts under one variability model and reuses those artefacts to derive the software products automatically, therefore reducing the cost of development while the reliability of software products is increased. The concept of adopting SPL-based approaches in the cloud service context has been explored in (Benlachgar and Belouadha, 2013; Wittern *et al.*, 2012; Garcıa-Galán, 2013).

An SPL-based approach for cloud service selection that employs feature models, extended with cardinalities and attributes, to describe the variability in cloud environments has been proposed by Quinton *et al.* (2014). The approach utilises a domain model to support the consistent configuration of the complete stack of cloud services that comply with user's functional and quality requirements and automates the deployment of such configurations by generating executable deployment scripts. Feature models provide the template for how artefacts are to be combined to yield a complete software product that satisfies a set of defined constraints. A tool support was developed based on Constraint Satisfaction, as part of an earlier SALOON framework to demonstrate the plausibility of this approach (Quinton *et al.*, 2013). Meanwhile, the limitation imposed by using a given cloud service and the benefit inherent in using several cloud platforms to deploy multi-cloud applications necessitate approaches that can handle the intrinsic variabilities among heterogeneous cloud service providers.

SALOON is a model-driven Ontology-based approach founded on feature models, to handle the variability in cloud services while managing the derivation of specific cloud configurations (Quinton *et al.*, 2013). Ontology was employed to model the semantics underlying the description of a variety of cloud systems. SALOON is proposed as a solution that can assist in deploying the multi-cloud application, particularly when one provider is incapable meeting all application requirements rather than doing so in an ad hoc manner. The SALOON framework is extensible by adding new feature model that conforms with the originating SALOON-based feature model meta-model. Cloud services are modelled as features, and selected features are transformed into propositional logic and constraints, and satisfiability (SAT) solvers (e.g. Sat4j) are used to confirm the validity of the configuration.

In the same line, Wittern *et al.* (2012) argue that the increase in cloud services provides the need for a means to capture the variety of capabilities, and asserts that many cloud service section approaches assume the underlying representation of the cloud service capabilities which should serve as input to the selection process. Therefore, Wittern *et al.* (2012) presented an approach to harness cloud service capabilities using variability model. The variability models serve as representation mechanisms and are called *Cloud Feature Models* (CFMs). CFMs are used to elicit requirements and to perform filtering operations within a process the authors referred to as a cloud service selection process (CSSP). The CSSP prunes the list of likely candidates based on decision makers' requirements, and these candidates (called *Alternative models*) are configurations that can be deployed. The *Alternative models* are subjected to a preference-based ranking process, subject to decision maker preferences on QOS values. The QoS values expressed by the decision maker are considered as the minimum threshold by the CSSP and the CSSP allows for evolutionary Cloud service selection, in which requirements can be updated in an iterative manner.

The approach is encapsulated in a prototypical tool based on the Eclipse Modelling Framework (EMF) that uses a Choco-based reasoning engine to perform automated analysis on the CFM; and requirement matching module, to determine alternative models that satisfy the decision makers' requirements.

The review of techniques for cloud service selections in this category showed that a number of these techniques made provision for the mechanism to aggregate atomic services, as well as a user interface to elicit the users' QoS preferences and aspirations. However, these techniques do not support the elicitation of subjective user requirements, and most of the techniques lack the means to present ranking results in a manner that reduces the complexity of exploring service alternatives.

2.3.2 Comparative Analysis of Cloud Service Selection Techniques

In order to foster the objectives of this study, a comparative analysis of cloud service selection techniques was conducted to identify gaps in the literature using a comparative framework that embraced some of the key issues in cloud service selection. As the first step, 35 related works in the literature were carefully selected based on their relevance to the objectives of the comparative survey. These identified works were analysed along six dimensions based on the issues observed in the review, and the analyses were captured in a tabular format. The comparison framework comprises six analysis dimensions, which are:

i. **Organisation and Composition of Atomic Services-** describes how a specific cloud service selection technique organises and combines atomic services to satisfy more complex user requirements.

ii. **The techniques employed to evaluate and rank service alternatives**- which includes the specific method employed to rank services.

iii. **Elicitation of users' QoS requirements-** explores how the selection technique elicits subjective user's QoS requirements as it relates to QoS preferences and aspiration.

iv. **Interactive GUI support-** analyse the presence of a user interface mechanism to elicit QoS information from users.

v. **Presentation of ranking results-** describes the visualisation mechanism employed to display ranking result in a manner to aid easy decision making.

vi. **Evaluation metrics employed-** explores the metric for evaluating the performance of the cloud service selection techniques.

The findings of the comparative review are as follows:

I. Organise and Compose Atomic Services

Most techniques in the literature, except for (Wittern *et al.*, 2012), (Quinton *et al.*, 2013) and (Quinton *et al.*, 2014), assume an underlying decision matrix, comprising of service alternatives together with their QoS properties (see Table 2.3). To effectively galvanise the potentials of cooperating atomic services, feature models from the domain of Software Product-line engineering were employed in Wittern *et al* (2012), Quinton *et al.* (2013) and Quinton *et al.* (2014).

Table 2.3: Summary of method for organising atomic services

#	Method	Source
1	Feature Models	– SALOON (Quinton *et al.*, 2013), – CSSP (Wittern *et al.*, 2012) – Quinton *et al.* (2014)
2	None	– Qu and Buyya (2014) – ALPHA (Sun *et al.*, 2014) – Kwon *et al.* (2013) – Tajvidi *et al.* (2014) – Mu *et al.* (2014) – QSSSIN_GU (Yu and Zhang, 2014) – Esposito *et al.* (2016) – Wang *et al.* (2014) – SMICloud (Garg *et al.*, 2013) – Gui *et al.* (2014) – Zeng *et al.* (2009) – CSRS (Han *et al.*, 2009) – MADMAC (Saripalli and Pingali, 2011) – CloudIntegrator (Cavalcante *et al.*, 2012) – W_SR (Jahani *et al.*, 2014) – Rehman *et al.* (2014) – DBaaS-Expert (Sahri *et al.*, 2014) – MSSOptimiser (He *et al.*, 2012) – Sundareswaran *et al.* (2012) – CloudAdvisor (Jung *et al.*, 2013) – CloudPick (Dastjerdi *et al.*, 2015) – CSS (Qian *et al.*, 2013) – Qu *et al.* (2014) – Kang and Sim (2010) – Mirmotalebi *et al.* (2012) – Rehman *et al.* (2011) – CSTrust (Ding *et al.*, 2014), – CloudRec (Yu, 2014) – RecTIN (Ma and Hu, 2014) – CloudRecommender (Zhang *et al.*, 2012) – Ruiz-Alvarez *et al.* (2011) – Baranwal *et al.* (2014)

II. Techniques Employed to Evaluate and Rank Service Alternatives

Specific methods employed by existing techniques to evaluate, rank and select services were classified into five categories, which include approaches based on MCDM, optimisation, recommendation, proximity metrics, and others. Within each category, techniques that provide a mechanism to handle fuzziness in relation to the user's QoS requirements were also explored.

Table 2.4 shows that existing techniques employ a variety of techniques for service evaluation, ranking and decision making to assist users to select the most optimal cloud services. Specifically, MCDM-based techniques employ AHP, TOPSIS, SAW, MAUT, and ELECTRE. To manage subjectivity in QoS information, other MCDM-based techniques employed uncertainty theories like fuzzy set theory, rough sets, interval number arithmetic, fuzzy inference and the fuzzy synthetic decision to evaluate service alternatives.

In optimization-based techniques the cloud service evaluation and selection problem were formulated as Constraint Satisfaction and/or Optimization Problem (CSP/CSOP), multi-objective optimization problem, the Multiple-Choice Knapsack Problem (MCKP) and its variants, tree-search problem etc.; while solutions are either optimal solutions or near-optimal solutions by the use of heuristics, greedy algorithm, and genetic algorithms.

Recommendation-based approaches rely on historical QoS information on services and evaluations from previous users to provide recommendations (Han *et al.*, 2009), while similarity computation based on similarity/distance metrics is applied in proximity-based techniques to determine the closeness of the user's QoS requirement to the QoS description of cloud services. Some other techniques employ semantic models based on ontology and custom matching algorithms to determine optimal services.

Table 2.4: Summary of Service evaluation and ranking methods

Category	Source	Summary of QoS-based Service Ranking and Evaluation Techniques
Fuzzy-MCDM-based	Qu and Buyya (2014)	Hierarchical Fuzzy Inference
	ALPHA (Sun *et al.*, 2014)	Fuzzy-based Ontology Similarity Matching, Fuzzy-AHP, Fuzzy-TOPSIS
	Kwon and Seo (2013)	Fuzzy-AHP
	Tajvidi *et al.* (Tajvidi *et al.*, 2014)	AHP and Fuzzy-AHP,
	Mu *et al.* (Mu *et al.*, 2014)	Intuitionistic Fuzzy Set, Rough Set, and TOPSIS
	QSSSIN_GU (Yu and Zhang, 2014)	Arithmetic on Interval Numbers and TOPSIS
	Esposito *et al*. (2016)	Fuzzy Inference, TOPSIS, Dempster-Shafer theory of Evidence, Mechanism Design (Game Theory)
	Wang *et al.* (2014)	Fuzzy Synthetic Decision
MCDM-based	SMICloud (Garg *et al.*, 2013)	AHP
	Gui *et al.* (Gui *et al.*, 2014)	MAUT, TOPSIS
	Zeng *et al.* (2009)	SAW, MAUT
	CSRS (Han *et al.*, 2009)	SAW
	MADMAC (Saripalli and Pingali, 2011)	SAW
	CloudIntegrator (Cavalcante *et al.*, 2012)	MAUT
	W_SR (Jahani *et al.*, 2014)	Min-Max (Rehman *et al.,* 2012)
	Rehman *et al.* (2014)	TOPSIS, ELECTRE
	DBaaS-Expert (Sahri *et al.*, 2014)	Ontology, AHP
Optimization -based	MSSOptimiser (He *et al.*, 2012)	Constraint Optimisation (Greedy Algorithm)
	Sundareswaran *et al.* (2012)	B+-Tree indexing, Greedy Algorithm
	CloudAdvisor (Jung *et al.*, 2013)	Constraint optimisation Satisfaction with Greedy Algorithm, benchmarking-based approximation technique
	CloudPick (Dastjerdi *et al.*, 2015)	Description Logic Matching Algorithm based on Genetic Algorithm
	CSS (Qian *et al.*, 2013)	Multi-objective Optimization- heuristic algorithm
Proximity - based	Qu *et al.* (2014)	Similarity Computation, Fuzzy-SAW-based approach
	Kang and Sim (2010)	Ontology similarity reasoning, Matching Algorithm
	Mirmotalebi *et al.* (2012)	Similarity Computation
	Rehman *et al.* (2011)	Similarity Computation based on Weighted Difference and Exponential Weighted Difference methods
Recommendation-based	CSTrust (Ding *et al.*, 2014)	Collaborative Filtering and Utility Computation
	CloudRec (Yu, 2014)	Regularised posterior probabilistic nonnegative matrix factorization
	RecTIN (Ma and Hu, 2014)	Ternary Interval Number, Fuzzy-AHP
Others	CloudRecommender (Zhang *et al.*, 2012)	Declarative SQL, Ontology Mapping
	Ruiz-Alvarez *et al.* (2011)	Matching Algorithm
	Baranwal *et al.* (2014)	Rank Voting Method, Data Envelope Analysis
	Quinton *et al.* (Quinton *et al.*, 2014)	Feature Modelling, Constraint Satisfaction
	CSSP (Wittern *et al.*, 2012)	Matching Algorithm, Constraint Satisfaction Programming
	SALOON (Quinton *et al.*, 2013)	Feature modelling, Ontology Similarity Reasoning, Prepositional Logic based on SAT

III. Elicitation of Users’ QoS Requirements

Decision-making has been defined as a process in which alternative(s) are identified and selected choosing an alternative(s) in accordance with the goals of, preferences of and constraints imposed by a decision maker. The assumption is usually that there are many alternatives available and the aim is to select the one that best approximates decision makers’ requirements. Most techniques unrealistically assumed that the user would provide perfectly crisp, precise and exact preference and aspiration information in the evaluation process, which is not congruent with the way humans think and communicate (Esposito *et al.*, 2016; Sun *et al.*, 2014; Qu and Buyya, 2014). The analysis explored how existing techniques elicit users’ preferences and aspirations in these three dimensions: Handling subjectivity in user’s QoS requirements, evaluating interrelationship of QoS criteria when eliciting preferences, and if the requirements elicitation covers both users’ QoS preferences and QoS aspirations.

a) Managing Subjectivity of Users’ QoS Requirements

The complexity of QoS factors blurs the preference perception of users (Dastjerdi and Buyya, 2011), thereby affecting how users express the degree of relative importance of each criterion and expected ideal points. Some techniques focused on measuring precise quantitative data and expect users to express requirements in the same manner, which sometimes requires expert knowledge (Qu and Buyya, 2014). Although user requirements are elicited in the form of weights and/or aspiration values, the difficulty inherent in expressing such requirements in exact or crisp values necessitates a QoS-aware techniques that can capture the vagueness in both user’s QoS preferences and aspiration (Barros and Dumas, 2006; Sun *et al.*, 2014; Qu and Buyya, 2014; Esposito *et al.*, 2016). In the literature, a few techniques have considered fuzziness in the elicitation process for QoS preferences by using fuzzy set and rough set theory; while the predominant technique for handling fuzziness in determining preference weights is fuzzy-AHP, as shown in Table 2.6. Table 2.5 shows that the subjectivity inherent in the users’ QoS aspiration requirements is elicited using: fuzzy set theory e.g. (Qu and Buyya, 2014), (Esposito *et al.*, 2016) and (Mirmotalebi *et al.*, 2012); interval numbers (e.g. (Ma and Hu, 2014) and (Yu and Zhang, 2014)). However, the approach presented in (Wang *et al.*, 2014) engaged fuzzy synthetic decision method of eliciting QoS requirements, and all other techniques elicit expected QoS values by users expressing crisp values.

Table 2.5: Eliciting QoS aspiration in Cloud Service Selection Techniques

QoS Aspiration Information	Method	Sources
Fuzzy	Interval Number	– RecTIN (Ma and Hu, 2014) – QSSSIN_GU (Yu and Zhang, 2014)
	Fuzzy Set Theory	– Qu and Buyya (2014) – Esposito *et al.* (2016) – Mirmotalebi *et al.* (2012)
	Fuzzy Synthetic Decision	– Wang *et al.* (2014)
Non-Fuzzy	Direct Crisp Elicitation	– CloudRecommender (Zhang *et al.*, 2012) – Gui *et al.* (2014) – Sundareswaran *et al.* (2012) – Ruiz-Alvarez and Humphrey (2011) – Quinton *et al.* (2014) – Zeng *et al.* (2009) – CSSP (Wittern *et al.*, 2012) – CloudAdvisor (Jung *et al.*, 2013) – Kang and Sim (2010) – CloudPick (Dastjerdi *et al.*, 2015) – CSS (Qian *et al.*, 2013) – CSRS (Han *et al.*, 2009) – MSSOptimiser (He *et al.*, 2012) – SALOON (Quinton *et al.*, 2013) – Rehman *et al.* (2011) – W_SR (Jahani *et al.*, 2014) – Rehman *et al.* (2014) – DBaaS-Expert (Sahri *et al.*, 2014)

b) *Considering Relationship among QoS Criteria*

When evaluating multiple criteria in decision-making scenarios, the priority of importance of each criterion in relation to other criterion is important in determining the overall best alternative(s). In most cases, user preferences are captured as weights denoting the priority of each criterion. Quantifying the relative importance of each criterion to another criterion is a precise means to capture user preferences, and promotes objectivity in the evaluation of services (Garg *et al.,* 2013; Sun *et al.,* 2014). It is desirable that techniques should objectively determine the priorities by catering for the interrelationships among criteria and one way to achieve this is by employing pairwise comparison.

The approaches for eliciting weights that denote relative importance were summarised and classified into pairwise comparison and non-pairwise comparison approaches while analysing how fuzziness is handled in the elicitation process (see Table 2.6).

Table 2.6: Eliciting QoS preferences in Cloud Service Selection Techniques

Domain	Preference Information	Method	Sources
Pairwise Comparison	Fuzzy	Fuzzy-AHP	– RecTIN (Ma and Hu, 2014) – Qu and Buyya (2014) – ALPHA (Sun *et al.*, 2014) – Kwon and Seo (2013) – Tajvidi *et al.* (2014)
	Non-fuzzy	AHP	– DBaaS-Expert (Sahri *et al.*, 2014) – SMICloud (Garg *et al.*, 2013) – Wang *et al.* (2014)
Non-pairwise Comparison	Fuzzy	Arbitrarily fuzzy weights assigned by users (Fuzzy set and rough set theories)	– Mu *et al.* (2014)
		Arbitrarily fuzzy weights assigned by users using fuzzy set theory	– Qu *et al.* (2014) – Esposito *et al.* (2016)
	Non-Fuzzy	Arbitrarily static weights assigned by users	– Gui *et al.* (2014) – Sundareswaran *et al.* (2012) – Baranwal and Vidyarthi (2014) – Zeng *et al.* (2009) – Kang and Sim (2010) – CSS (Qian *et al.*, 2013) – CSRS (Han *et al.*, 2009) – Mirmotalebi *et al.* (2012) – CloudIntegrator (Cavalcante *et al.*, 2012) – MSSOptimiser (He *et al.*, 2012) – Rehman *et al.* (2011) – W_SR (Jahani *et al.*, 2014)
		From Expert (Wide-band Delphi method)	– MADMAC (Saripalli and Pingali, 2011)
		Significance Weighing Method (Zheng *et al.*, 2011)	– Kang and Sim (2010)
		Entropy Method (Wang *et al.*, 2007)	– Rehman *et al.* (2014)

As presented in Table 2.6, the techniques classified under pairwise comparison that used AHP include Sahri *et al.* (2014), Garg *et al.* (2013), Wang *et al.* (2014); while those that employed fuzzy-AHP include Ma and Hu (2014), Qu and Buyya (2014), Sun *et al.*(2014), Kwon and Seo (2013), and Tajvidi *et al.* (2014).

However, it is observed that more techniques are classified under non-pairwise comparison as priority weights are arbitrarily assigned by users as static weights to signify the importance of criteria, without consideration for the interrelationships among the criteria. Qu *et al.* (2014), Esposito *et al.* (2016) and Mu *et al.,* (2014) are classified under non-pairwise comparison, and they allow users' to express subjectivity in arbitrarily assigning priority weights using fuzzy set theories and rough sets. Apart from the user directly assigned weights arbitrarily, weights are sometimes obtained from expert surveys employing Wideband Delphi method, significance weighing method (Zheng *et al.,* 2011) and entropy method (Wang *et al.,* 2007).

c) Service Evaluation Based on both QoS Preferences and Aspirations

QoS factors are rarely of equal importance to users (Sahri *et al.*, 2014; Javed *et al.*, 2016), and the importance of each QoS criteria is specified by weights that reflect QoS preferences, with which a ranking of the cloud services can be realised. QoS aspirations define the user's desired ideal points for each criterion, and it comprises the goals and constraints for each QoS criteria as it pertains to each user; as such, users should be able to define their own ideal values, and/or constraints on those values, which serve as important inputs to the evaluation process of service alternatives.

Simultaneously considering both user preferences and aspiration in the service evaluation process requires a service evaluation and ranking approach that is able to incorporate subjective preference weights while resolving the subjective goals and constraints on QoS values expressed by the user. The analysis of QoS preference and aspiration information employed in techniques was classified into three categories: those that employ information of both QoS preference and aspiration, QoS preference alone, and QoS aspiration alone; the consideration of subjectivity in this QoS information was also surveyed. Although Table 2.7 shows that a lot of techniques incorporate both weights and aspiration values in the evaluation of service alternatives, most of these techniques do not cater for subjectivity in QoS requirements.

As shown in Table 2.7, the techniques that absolutely catered for the fuzziness in both QoS preference and aspiration include (Ma and Hu, 2014), (Qu and Buyya, 2014) and (Esposito *et al.*, 2016); however, (Mirmotalebi *et al.*, 2012) and (Wang *et al.*, 2014) elicited QoS aspiration as fuzzy inputs, while the priority weights are captured as crisp values (see footnote in Table 2.7). Other techniques require users to express either preference or aspiration information, which is sometimes based on the assumption that the alternatives have m*et al*l other user's criteria.

Table 2.7: QoS Preference and Aspiration in Cloud Service Selection Techniques

QoS Requirement Information	QoS Aspiration and QoS Preferences	Preferences Only	Aspiration Only
Fuzzy	– RecTIN (Ma and Hu, 2014) – Qu and Buyya (2014) – Esposito *et al.* (2016) – Mirmotalebi *et al.* (2012) * ⁻ Wang *et al.* (2014)*	– Kwon and Seo (2013) – ALPHA (Sun *et al.*, 2014) – Mu *et al.* (2014) – Tajvidi *et al.* (2014) – Qu *et al.* (2014)	⁻ QSSSIN_GU (Yu and Zhang, 2014)
Non-fuzzy	– Gui *et al.* (Gui *et al.*, 2014) – Sundareswaran *et al.* (2012) – Zeng *et al.* (2009) – Kang and Sim (2010) – DBaaS-Expert (Sahri *et al.*, 2014) – Rehman *et al.* (2014) – W_SR (Jahani *et al.*, 2014) – Rehman *et al.* (2011) – MSSOptimiser (He *et al.*, 2012) – CSRS (Han *et al.*, 2009) – CSS (Qian *et al.*, 2013) – Mirmotalebi *et al.* (2012) ** – Wang *et al.* (2014)**	– SMICloud (Garg *et al.*, 2013), – Baranwal and Vidyarthi (2014) – CloudIntegrator (Cavalcante *et al.*, 2012) – MADMAC (Saripalli and Pingali, 2011) – CSTrust (Ding *et al.*, 2014)	– CloudRecommender (Zhang *et al.*, 2012) – Ruiz-Alvarez and Humphrey (2011) – Quinton *et al.* (2014) – CSSP (Wittern *et al.*, 2012) – CloudAdvisor (Jung *et al.*, 2013) – SALOON (Quinton *et al.*, 2013) – CloudPick (Dastjerdi *et al.*, 2015)
* QoS aspiration are elicited as fuzzy inputs(fuzzy) ** QoS preference weights are elicited as crisp weights (non-Fuzzy)			

IV. Interactive GUI Support

Users' engagement with the marketplace to select cloud service should be facilitated by intuitive and interactive Graphical User Interfaces (GUI). The essence of such interfaces is not to overwhelm users with excessive input fields, so as to reduce the cognitive load on users when specifying requirements (Zhang *et al.*, 2012). The interface captures the requirements and converts it into queries used to search for optimal alternatives. Therefore, such interfaces should support the input of the subjective requirements by incorporating fuzziness in the input process in a manner that is easy to understand. Noting the complexity of eliciting exact, crisp numerical values, the interface should intuitively allow for and interpret vague user input requirements by incorporating linguistic expressions and on-screen interaction elements such as sliding and clicking (Sundar *et al.*, 2014).

However, observed in Table 2.8 is that most techniques (22 out of 35 techniques reviewed) do not incorporate intuitive user interfaces in their approaches. The GUI support identified with techniques can be mainly classified into two domains: web-based and window-based; with the exception of (Sundareswaran *et al.*, 2012) and (Kwon and

Seo, 2013). User interface support was reported in (Sundareswaran *et al.*, 2012), but it was difficult to ascertain the domain it belonged; also, the techniques proposed in (Kwon and Seo, 2013) employed a third party desktop application, called *Expert Choice 11.5* to capture user requirements.

Table 2.8: The use of GUI in Cloud Service Selection Techniques

GUI Domain	Sources	
Web-based	– Qu and Buyya (2014), – Gui *et al.* (2014), – CloudAdvisor (Jung *et al.*, 2013) – Kang and Sim (2010)	– CloudPick (Dastjerdi *et al.*, 2015) – CloudRecommender (Zhang *et al.*, 2012)
Windows-based	– Ruiz-Alvarez and Humphrey (2011) – Quinton *et al.* (2014) – CSSP (Wittern *et al.*, 2012)	– Mirmotalebi *et al.* (2012) – SALOON (Quinton *et al.*, 2013)
Third Party Software (Expert Choice 11.5)	– Kwon and Seo (2013)	
Unspecified	– Sundareswaran *et al.* (2012)	
No GUI Support Reported	– RecTIN (Ma and Hu, 2014) – ALPHA (Sun *et al.*, 2014) – SMICloud (Garg *et al.*, 2013) – Baranwal and Vidyarthi (2014) – Zeng *et al.* (2009) – CSTrust (Ding *et al.*, 2014) – Qu *et al.* (2014) – CSS (Qian *et al.*, 2013) – CSRS (Han *et al.*, 2009) – Tajvidi *et al.* (2014) – MADMAC (Saripalli and Pingali, 2011) – Esposito *et al.* (2016) – Mu *et al.* (2014)	– CloudIntegrator (Cavalcante *et al.*, 2012) – QSSSIN_GU (Yu and Zhang, 2014) – MSSOptimiser (He *et al.*, 2012) – Wang *et al.* (2014) – Rehman *et al.* (2011) – W_SR (Jahani *et al.*, 2014) – CloudRec (Yu, 2014) – Rehman *et al.* (2014) – DBaaS-Expert (Sahri *et al.*, 2014)

V. Presentation of Ranking Result

Analysis of techniques in the literature revealed a minimal emphasis on presentation of ranking results; with respect to means to explore evaluation and ranking results (see Table 2.9). Only 5 out of 35 studies incorporated visual exploration mechanisms, including: charts (line and radar chart), as in (Gui *et al.*, 2014); and kiviat charts, as in (Garg *et al.*, 2013); multi-cloud comparison tables, as in (CloudAdvisor (Jung *et al.*, 2013)); web-based widgets, as in CloudRecommender (Zhang *et al.*, 2012) and third party desktop application software, Expert Choice 11.5, as in the work of Kwon and Seo (2013); while most techniques did not incorporate any intuitive mechanism for visualizing service evaluations and rankings.

Table 2.9: Visualisation Mechanism Employed in Cloud Service Selection

Visualization Type	Sources
Charts (e.g. line, kiviat and radar)	Gui *et al.* (Gui *et al.*, 2014), SMICloud (Garg *et al.*, 2013)
Multi-cloud Comparison Table	CloudAdvisor (Jung *et al.*, 2013)
Web Widgets	CloudRecommender (Zhang *et al.*, 2012)
Third-party Software (Expert Choice 11.5)	Kwon and Seo (2013)
No Information Visualization support Reported	– ALPHA (Sun *et al.*, 2014) – Baranwal and Vidyarthi (2014) – CloudIntegrator (Cavalcante *et al.*, 2012) – CloudPick (Dastjerdi *et al.*, 2015) – CloudRec (Yu, 2014) – CSRS (Han *et al.*, 2009) – CSS (Qian *et al.*, 2013) – CSSP (Wittern *et al.*, 2012) – CSTrust (Ding *et al.*, 2014) – DBaaS-Expert (Sahri *et al.*, 2014) – Esposito *et al.* (2016) – Kang and Sim (2010) – MADMAC (Saripalli and Pingali, 2011) – Mirmotalebi *et al.* (2012) – MSSOptimiser (He *et al.*, 2012) – Mu *et al.* (2014) – QSSSIN_GU (Yu and Zhang, 2014) – Qu and Buyya (2014) – Qu *et al.* (Qu *et al.*, 2014) – Quinton *et al.* (2014) – RecTIN (Ma and Hu, 2014) – Rehman *et al.* (2011) – Rehman *et al.* (2014) – Ruiz-Alvarez and Humphrey (2011) – SALOON (Quinton *et al.*, 2013) – Sundareswaran *et al.* (2012) – Tajvidi *et al.* (2014) – W_SR (Jahani *et al.*, 2014) – Wang *et al.* (2014) – Zeng *et al.* (2009)

VI. Metrics for Evaluating Cloud Service Selection Techniques

Performance evaluation results are vital benchmarks to determine the utility, plausibility and applicability of existing techniques. It forms the basis to appraise the pros and cons of techniques in order to motivate new proposals or identify new research directions. A summary of performance evaluation methods of existing techniques was presented in Table 2.10 and five main performance metrics employed in the techniques under review was identified. They include accuracy, efficiency, scalability, use case/case study, and usability. Accuracy describes the ability of the proposed techniques to evaluate and rank service alternatives with respect to approximating users' requirements. Efficiency is a measure of the time cost and computational overhead of the proposed approach, while scalability describes the performance of the techniques with an increase in the number of service alternatives. To show the practicality of the techniques, use case or case studies were employed and usability describes empirical user studies to test the applicability of techniques. As illustrated in Table 2.10, accuracy metric topped the list of performance evaluation methods as it was employed in 18 out of 35 sources.

Table 2.10: Performance Evaluation Metrics Employed for Cloud Service Selection

#	Sources	Accuracy	Efficiency	Scalability	Use Case	Usability	Availability	Extendibility
1	ALPHA (Sun *et al.*, 2014)	●	●	○	○	○	○	○
2	Baranwal *et al.* (2014)	○	○	○	○	○	○	○
3	CloudAdvisor (Jung *et al.*, 2013)	●	○	○	○	○	○	○
4	CloudIntegrator (Cavalcante *et al.*, 2012)	○	○	○	○	○	○	○
5	CloudPick (Dastjerdi *et al.*, 2015)	○	●	●	○	○	○	○
6	CloudRec (Yu , 2014)	●	○	○	○	○	○	○
7	CloudRecommender (Zhang *et al.*, 2012)	○	○	○	●	○	○	○
8	CSRS (Han *et al.*, 2009)		●		●	○	○	○
9	CSS (Qian *et al.*, 2013)	●	●	●	○	○	○	○
10	CSSP (Wittern *et al.*, 2012)		○	○	○	○	○	○
11	CSTrust (Ding *et al.*, 2014),	●	○	○		○	○	○
12	DBaaS-Expert (Sahri *et al.*, 2014)		○	○	●	○	○	○
13	Esposito *et al.* (2016)	●	●	○	○	○	○	○
14	Gui *et al.* (2014)	●	○	○	○	○	○	○
15	Kang and Sim (2010)	●	○	○	○	○	○	○
16	Kwon and Seo (2013)	○	○	○	●	○	○	○
17	MADMAC (Saripalli and Pingali, 2011)	○	○	○	●	○	○	○
18	Mirmotalebi *et al.* (2012)	●	○	○	●	○	○	○
19	MSSOptimiser (He *et al.*, 2012)	●	●	○	○	○	○	○
20	Mu *et al.* (2014)	●		○	○	○	○	○
21	QSSSIN_GU (Yu and Zhang, 2014)		●	○	●	○	○	○
22	Qu and Buyya (2014)	●	○	●	●	○	○	○
23	Qu *et al.* (2014)	●	○	○	○	○	○	○
24	Quinton *et al.* (2014)		○	●	○	●	○	○
25	RecTIN (Ma and Hu, 2014)	●	○	○	○	○	○	○
26	Rehman *et al.* (2014)	●	○	○	○	○	○	○
27	Rehman *et al.* (2011)	○	○	○	○	○	○	○
28	Ruiz-Alvarez *et al.* (2011)	○	●	○	○	○	●	○
29	SALOON (Quinton *et al.*, 2013)	●	○	○	●	○	○	○
30	SMICloud (Garg *et al.*, 2013)	○	●	●	●	○	○	○
31	Sundareswaran *et al.* (2012)	○	●	○	○	○	○	○
32	Tajvidi *et al.* (2014)	○	○	○	●	○	○	○
33	W_SR (Jahani *et al.*, 2014)	○	○	●	○	○	○	○
34	Wang *et al.* (2014)	●	○		○	○	○	○
35	Zeng *et al.* (2009)	○	○	●	○	○	○	●
	Count	18	12	7	14	1	1	1

Extendibility: refers to the cost of extending the proposed algorithm to process new elements and attributes in the XML descriptions of the cloud provider;
Availability: Describing the ubiquitous nature of the algorithm to be deployable to any device by context-awareness.
●= Present ○= Absent

Accuracy metric is closely followed by the use of use cases to demonstrate how the techniques work. The time efficiency of techniques in relation to baseline approaches or other techniques occurred 12 times, with scalability evaluations occurring 7 times out of 35 sources. A user study to determine the utility and applicability of the technique was only reported in (Quinton *et al.*, 2014), where authors conducted the experiment with a group of real participants to evaluate the effectiveness compared to a manual process.

Extendibility and availability (see footnote on Table 2.10) are other metrics found during analysis. Based on this analysis, there seems to be more emphasis on performance metrics such as accuracy, efficiency and scalability compared to user satisfaction.

2.3.3 Gaps Identified in the Literature

From the foregoing, the comparative survey revealed that a number of key issues have attracted the attention of authors on the subject of cloud service selection and this has influenced the trends of research in this domain so far. However, there exist some gaps with respect to the suitability of the existing techniques for service selection in cloud e-marketplaces. The gaps have been identified based on the following – the organisation and composition of atomic services; elicitation of users' QoS preferences and QoS aspiration; interactive GUI support to elicit QoS information from users; mechanisms for the presentation of ranking results; and the evaluation processes employed. The gaps in the existing techniques were summarised in Table 2.11.

The analysis of the 35 techniques summarised in Table 2.11 shows that only 3 out of 35 techniques reviewed provided a means to organise and aggregate atomic services into composite offerings to meet complex user requirements. Meanwhile, 8 techniques possess the mechanism to elicit subjective QoS preferences and only 6 techniques elicit subjective QoS aspirations. Besides, RecTIN (Ma and Hu, 2014), as well as, the techniques proposed by Qu and Buyya (2014) and Esposito *et al.* (2016), are the only techniques that elicit both the QoS preferences and aspirations from the users. Five techniques employed the use of a user interface through which users can express their QoS requirements, while only 5 techniques reviewed used a form of visualisation to present ranking results. Although user experience is a vital consideration when designing a cloud service selection technique, only one technique reported a usability evaluation of its service selection technique. Meanwhile, the result of our analysis showed that no technique completely addressed the vital dimensions that are required to reduce service choice overload and improve user experience in cloud service e-marketplaces. Therefore, this study fills these gaps by formulating a framework for cloud service selection that will improve the quality of user experience in cloud service e-marketplace.

Table 2.11: Summary of Gaps Identified in the Literature

#	Source	Summary of Technique	Organise Atomic Services	Elicit Subjective QoS Preference	Elicit Subjective QoS Aspiration	Employ GUI	Incorporate Visualization	Usability Evaluation
1	Qu and Buyya (2014)	A cloud service evaluation system using hierarchical fuzzy inference system	○	●	●	●	○	○
2	Sun *et al.*(2014)	A fuzzy framework for cloud service selection	○	●	○	○	○	○
3	Kwon and Seo (2013)	A model to choose a cloud service using fuzzy AHP	○	●	○	●	●	○
4	Tajvidi *et al.* (2014)	A Fuzzy-based cloud service selection framework	○	●	○	○	○	○
5	Mu *et al.* (2014)	service selection based on uncertain user preference	○	●	○	○	○	○
6	Yu and Zhang (2014)	Group user SaaS services selection using interval numbers	○	○	●	○	○	○
7	Esposito *et al.* (2016)	Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory	○	●	●	○	○	○
8	Wang *et al.* (2014)	A fuzzy synthetic decision and fuzzy logic based cloud service selection framework	○	○	●	○	○	○
9	Garg *et al.* (2013)	An AHP-based framework for comparing and ranking cloud services	○	○	○	○	●	○
10	Gui *et al.* (2014)	A service brokering and recommendation mechanism for better-selecting cloud services	○	○	○	●	●	○
11	Zeng *et al.* (2009)	A SAW and MAUT-based approach for cloud service selection	○	○	○	○	○	○
12	Han *et al.* (2009)	A service recommendation system for cloud computing market	○	○	○	○	○	○
13	Saripalli and Pingali (2011)	A multiple attribute decision methodology for adoption of clouds	○	○	○	○	○	○
14	Cavalcante *et al.*(2012)	An approach to optimize service selection in cloud Multiplatform Scenarios	○	○	○	○	○	○ ○
15	Jahani *et al.* (2014)	A Min-Max QoS-based ranking approach for ranking cloud services	○	○	○	○	○	○
16	Rehman *et al.* (2014)	Parallel cloud service selection and ranking based on QoS history	○	○	○	○	○	○
17	Sahri *et al.* (2014)	A recommender system for the selection of the right cloud database	○	○	○	○	○	○
18	He *et al.* (2012)	A QoS-driven service selection for multi-tenant SaaS	○	○	○	○	○	○
19	Sundareswaran *et al.* (2012)	A brokerage-based approach for cloud service selection	○	○	○	●	○	○
20	Jung *et al.* (2013)	A recommendation platform for cloud configuration and pricing	○	○	○	●	●	○
21	Dastjerdi *et al.* (2015)	A cross-cloud framework for QoS-aware service deployment	○	○	○	●	○	○
22	Qian *et al.* (2013)	An approach for cloud service selection in IaaS platforms	○	○	○	○	○	○
23	Qu *et al.* (2014)	Context-aware cloud service selection based on assessment aggregation	○	●	○	○	○	○
24	Kang and Sim (2010)	A multi-criteria cloud service search engine	○	○	○	●	○	○
25	Mirmotalebi *et al.* (2012)	A preference-based approach for personalized service ranking	○	○	●	●	○	○
26	Rehman *et al.* (2011)	Distance-based approach for cloud service ranking	○	○	○	○	○	○
27	Ding *et al.* (2014)	An approach for evaluating trustworthiness of cloud services	○	○	○	○	○	○
28	Yu (2014)	A framework for personalized service recommendation in the cloud	○	○	○	○	○	○
29	Ma and Hu (2014)	Cloud service recommendation using ternary interval numbers	○	●	●	○	○	○
30	Zhang *et al.* (2012)	A recommender system for cloud infrastructure services selection	○	○	○	●	●	○
31	Ruiz-Alvarez and Humphrey (2011)	An approach to cloud storage service selection based on matchmaking	○	○	○	●	○	○
32	Baranwal *et al.* (2014)	A framework for cloud service selection using ranked voting method	○	○	○	○	○	○
33	Quinton *et al.* (2014)	A selection and configuration of Cloud environments using SPL	●	○	○	●	○	●
34	Wittern *et al.* (2012)	Cloud service selection based on variability modelling	●	○	○	●	○	○
35	Quinton *et al.* (2013)	An approach for cloud configurations using feature models and ontologies	●	○	○	●	○	○
		Count	**3**	**8**	**6**	**9**	**5**	**1**

● = Supported ○ = Not Supported

2.4 EMERGENT PERSPECTIVES IN CLOUD SERVICE SELECTION

Consequent on the findings from the comparative survey, a number of emerging perspectives on the key ingredients of a service selection framework that will improve the user experience in a cloud service e-marketplace are highlighted in this section. These emergent perspectives cover the key requirements for a cloud service selection framework that will suffice for a cloud service e-marketplace context, as well as relevant concepts that enable the realisation of the framework.

2.4.1 Key Requirements for Cloud Service Selection Framework

Addressing some of the open issues based on the comparative review is the first step to uncovering the requirements for an effective technique suitable for the e-marketplace context. The key requirements for a cloud service selection framework are listed and described as follows:

a) Requirement 1: Organise and Compose Cloud Ecosystem Atomic Services

A cloud marketplace is an ecosystem of heterogeneous services from multiple providers. The different ways in which these services are aggregated creates a plethora of potential offerings with varied QoS factors that can satisfy complex user needs of users (Barros and Dumas, 2006). There is a need to explicitly capture the cloud service attributes (functional and non-functional), and the cross-service relationships and constraints that guide the cloud service compositions (Akolkar *et al.*, 2012) in a logical and structural manner (Wittern *et al.*, 2012). Previous works have proposed the use of feature models to capture the variabilities of Cloud services and applied automated means generate valid cloud service offerings (Wittern *et al.*, 2012; Quinton *et al.*, 2014). However, users are still expected to painstakingly configure cloud services, with the assumption that all users are full domain experts. A cloud marketplace should among others, provide a real online shopping experience similar to existing e-commerce platforms (Akolkar *et al.*, 2012; Menychtas *et al.*, 2014), where available service offerings can be listed in the marketplace catalogue and seamlessly updated in a manner completely transparent to the users.

b) *Requirement 2: Elicit both Fuzzy QoS Preference and Aspirations from users*

An accurate elicitation of user requirements involves the interpretation of fuzzy expressions in evaluating services (Qu and Buyya, 2014; Esposito *et al*., 2016; Sun *et al.*, 2014). The ability to naturally express vague preferences or aspiration using linguistic terminologies is a better way to explore cloud services for selection purposes and would enable easier and quicker expression of requirements (Esposito *et al*., 2016; Qu and Buyya, 2014; Gatzioura *et al.*, 2012). For example, it is more convenient to use the following linguistic terminologies when expressing QoS aspiration "*the threshold of reliability metric should be in the vicinity of x*", or "*cost should be the in the range of x and y*" or "*High availability close to the value z*" etc., (where *x*, *y* and *z* are specific QoS values).

Furthermore, the advantage of pairwise comparisons is that it allows the derivation of priority weights of the criterion from comparison matrices, rather than arbitrarily assigning weights directly (Javanbarg *et al.*, 2012). Since human judgment is shrouded with impression and vagueness in most practical cases, users might be reluctant or unable to assign exact numerical values in comparison judgements (Mikhailov and Tsvetino, 2004). It has been proposed that a better approach to capturing the user's claim about the relative importance of criteria is to delineate comparison ratios as fuzzy numbers (Cakir and Canbolat, 2008; Tajvidi *et al.*, 2014; Mikhailov and Tsvetino, 2004). In addition, a cloud service selection framework should consider both users' QoS preferences and aspiration in the service evaluation process.

c) *Requirement 3: Evaluation and Rank a Large Assortment of Service Alternatives*

Cloud services are characterised by multiple QoS attributes, and there is need to evaluate the overall performance of all services by some utility functions, with respect to users' QoS requirements. The cloud e-marketplace context requires approaches that can deal efficiently with a large number of alternatives without accruing high computational overhead (Dastjerdi *et al.*, 2015).

d) Requirement 4: Integrate Fuzzy-based User Interfaces

The user interface underscores input and output features of the cloud service e-marketplace; input is how a user expresses QoS requirements, whereas the output presents the result of those requests to the user (Galitz, 2007). In eliciting users' requirements, user interface designs that intuitively capture these requests that are subjective in nature are desirable, because the user's perception of the interface affects attitude to what comes out through it (Sundar *et al.*, 2014), and ultimately affects user satisfaction (Kuniavsky, 2003; Sundar *et al.*, 2014). Furthermore, integrating fuzzy-enabled web-based widgets for eliciting vague preferences and aspirations under one integrated visual interface can also enhance user experience.

e) Requirement 5: Visualise Cloud Service Ranking Results

One of the laws of e-commerce states that if users cannot find it, they cannot buy it either; the primary medium of user's engagement of the cloud service e-marketplace is visual, enabling an information visualisation mechanism aid effective user interaction and simplifies decision making. Most cloud service selection approaches act like black boxes that generate a ranked list of cloud services without providing insight into the basis of the rankings (Chen *et al.*, 2013). Cloud service selection frameworks should incorporate visualisation mechanism that improves users' understanding of the rationale of rankings.

f) Requirement 6: Take into cognizance usability and user experience factors

Apart from the efficiency and accuracy evaluations which are predominant in the literature, more user studies should be carried out on techniques to ascertain its suitability for a cloud service e-marketplace context. The user interface obscures all the technical and computational processes underlying marketplace operations while showcasing a productive, enjoyable and satisfying means to explore and select services. Cloud service selection frameworks should include unobtrusive graphical user interfaces to both exploration and selection. An unduly complex design increases the difficulty in performing the both tasks, and negatively impacts on user experience (Galitz, 2007).

2.4.2 Considerations for the Design of a framework for Cloud Service Selection

Having identified the key requirements for a service selection technique that will suffice for selecting services in a cloud service e-marketplace, this subsection elaborates on considerations of relevant concepts and techniques that could realise a cloud service selection framework that meets these requirements. These concepts and techniques formed the basis for the framework proposed in this study. The concepts and techniques include the following: i) Organize cloud ecosystem atomic services and populate the service e-marketplace directory; ii) Elicit user fuzzy QoS preferences and aspirations; iii) Perform QoS-based evaluation and ranking of cloud service alternatives with respect to user QoS requirements; iv) Wrap the underlying functionalities of (i), (ii) and (iii) in a tidy graphical user interface. Figure 2.7 shows the elements of the considerations for the design of a cloud service selection framework, and details concerning each of the elements are presented in subsequent sections.

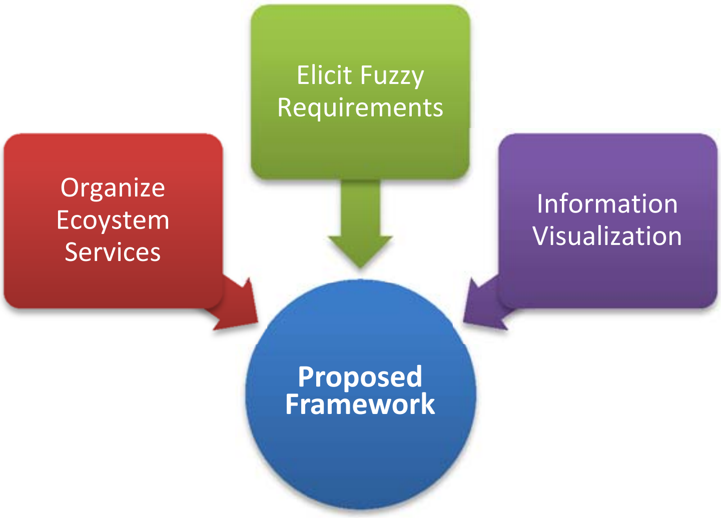

Elicit Fuzzy Requirements
Organize Ecoystem Services
Information Visualization
Proposed Framework

Figure 2.7: Considerations for designing a suitable framework

2.4.3 Variability Management for Atomic Services in Cloud Ecosystems

The cloud service e-marketplace provider is the one who manages the ecosystem and decides on the strategies for enhancing the value chain of the ecosystem. Enhancing the value inherent in the ecosystem entails deciding how services can be combined to deliver maximum value. Besides, to determine valid combinations of service in an ad hoc manner, would undermine the net value characteristic of ecosystems; more so, such ad hoc processing is error-prone and time-consuming (Deelstra *et al.*, 2005; White *et al.*, 2008; Rabiser *et al.*, 2009). Therefore, to adequately estimate the value of the ecosystem, first, there is a need for a logical hierarchical arrangement of all the participating services into a knowledge model based on a specific combinatorial blueprint and, secondly, a means to automatically derive useful information from the analysis of the logical hierarchy of these services. Automating the analysis of the ecosystem knowledge model produces a number of useful information about the ecosystem and aids the e-marketplace to make informed decisions about the ecosystem. For example, the provider may be interested in knowing how many valid combinations are possible in the ecosystem; this information implies the number of composite services indexed in the service directory and provisioned via the e-marketplace. Potentially, this number can be very high depending on the number of collaborating atomic services and knowing the number of possible composite services is enough basis for the e-marketplace provider to decide the range of services the e-marketplace would offer. Other useful information is identifying atomic services that will not fully benefit from the value chain in the ecosystem (partly or fully due to their presence in a few or none of the possible combinations). Consequently, a structured model and automated analysis would offer some strategic benefit to service providers, so that service providers can estimate the profitability of the e-marketplace platform to make strategic decisions for improving the competitiveness in the ecosystem.

The structure of the cloud ecosystem is analogous to the concepts of Software Product Line Engineering (SPLE) and product configuration (PC) (Hubaux *et al.*, 2012; Berger *et al.*, 2014) Therefore, the variability modelling techniques used in the SPLE and PC is applicable and can be adapted to effectively structure the hierarchical interrelationships among the ecosystem services. The PC domain is concerned with the ability to mass customise products targeted at specific requests and/or user segments, which is a crucial determinant of reducing lead time, and increase business process efficiency in mass-manufacturing (Haug *et al.*, 2011). Mass-customization techniques have been applied to concrete products, for example, bicycles (bikeconfig.com) and baby strollers (bugaboo.com), as well as insubstantial products like software and services (e.g. insurance, tourism, etc.). Configuration software is employed to adapt products or services to suit specific requirements by combining components, characterised by specified attributes, based on the constraints that underlay the valid combinations of those components (Hvam *et al.*, 2008). On the other hand, a software product line is a "set of software-intensive systems that share a common, managed a set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" (Bass and Kazman, 2003). The cornerstone of achieving product configuration and coming up with software instance from a software product line is:

i. The knowledge representation of the component/software features based on variabilities and commonalities;

ii. The computer-aided reasoning techniques employed to support both product configuration and software product line;

On the grounds that the domains of product configuration and software product lines share a lot of similarities with the concept of cloud ecosystems, the application of variability modelling and automated reasoning techniques to organise and populate the service directory with valid composite services were explored in this study.

I. Variability Management Techniques

Variability models are used to describe and centrally organise variabilities in the product line and product configuration and to support product derivation and configuration. Modelling variability is the core of any software product line engineering and product configuration endeavour and has received a lot of attention in the research community, with several techniques reported in the literature (Deelstra *et al.*, 2005; Czarnecki *et al.*, 2012; Hubaux *et al.*, 2012). These approaches are classified into two main categories: Feature-based Modelling and Decision-based Modelling. While feature model first abstracts the product line constituents as hierarchical features with cross-tree relationships, creating a basis for product derivation, decision models are the set of decisions that are adequate to distinguish among the products of an application engineering product line and to guide the adaptation of outputs of application engineering. For the purpose of this study, the feature-based modelling approach was adopted since both approaches are equally viable for managing variability (Czarnecki *et al.*, 2012).

a) Feature Model

In software product line engineering, a feature model is a graphical representation of common aspects and di□erences in a collection of products in a product line and is used to structure and constrain the product options. A feature is defined as the end-users' understanding of the capabilities of systems in the domain (Berger *et al.*, 2014). A feature model is a hierarchically arranged collection of features and consists of the interrelationships between a parent feature and its child features, and a set of cross–tree constraints that define the criteria for feature inclusion or exclusion. A feature model represents in a single model, all possible alternatives that the scope of the feature covers. Each solution is a valid instance of the feature model.

In this study, each participating atomic service has been defined and abstracted as a feature in a feature model, and the range of possible solutions that are obtainable from the ecosystem is defined by the entire model. Cross-tree constraints provide a 'legal' basis of how services and their QoS attributes can be legally combined. Benavides *et al.* (Benavides *et al.*, 2010), identified three main types of feature-based models: basic feature models (Kang *et al.*, 1990), cardinality-based feature models (Czarnecki *et al.*, 2005); and extended feature models (Benavides *et al.*, 2006).

The basic feature model describes three feature types-Mandatory, Optional, and Alternative, and two cross-tree constraints-Requires and Excludes. A *mandatory* feature is a feature that must be included in a product, while an optional feature is a feature that may or may not be included in a feature. Given a set of features from which only one feature is selected to be included in a product is called an alternative feature. However, the inadequacy of alternative relationship to model situations with multiple children features motivated cardinality-based feature model, in which numbers are introduced to denote the multiplicities of the set of features of the basic feature model.

Although basic feature model and cardinality-based feature model can be used to provide a basis for automated configuration of actual products, there is need to sometimes include in the feature model quality information about features (such as non-functional attributes). In extended feature models, feature model is annotated with quality information; the analysis could use these qualities as a basis to determine valid combinations. In classic software product line domain, extended feature models are desirable variability modelling techniques for modelling cloud ecosystem; they can capture cloud services, their QoS attributes and interrelationships constraints, which is important in order to generate valid combinations to populate the e-marketplace service directory; an example of a feature in the extended feature model is shown in Figure 2.8. Benavides *et al.* (2010) presented the concepts that describe the extended feature model as follows:

i. **Feature:** A functional characteristic of a product or an increment in product functionality. E.g. an SMS notification cloud service, or an email cloud service

ii. **Attribute:** Any measurable characteristic of a feature that can be measured. For example, the SMI factors defined by the CSMIC are measurable entities that form the attributes of a cloud service. For Example, reliability is a cloud service QoS attributes.

iii. **Attribute domain:** The attribute domain specifies the range of values that an attribute can assume. Domain covers either qualitative or quantitative (discrete and continuous) values corresponding to the heterogeneous QoS of cloud services.

iv. **Attribute value:** Attribute values define the actual value that belongs to a particular domain. The attributes values of a concrete product are usually an aggregation of all the values of corresponding features of the final product. For example, the cost of a product aggregates all the cost of the features included in a product.

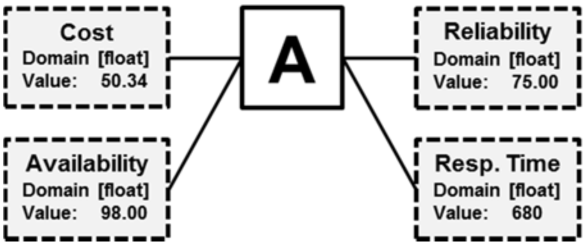


Figure 2.8: Extended Feature Model
Showing, mandatory, alternative, Optional and 'Or' features and relationships
Adapted from Benavides *et al.* (2010)

b) Automated Analysis of Feature Model

Deriving useful information from the ecosystem model requires an automated mechanism that is able to reason on and analyse the knowledge model upon which the service interrelationship is built (Benavides *et al.*, 2006; Benavides *et al.*, 2010; Karataş *et al.*, 2012; Elfaki *et al.*, 2012). Automated analysis of feature models uses computer-aided mechanisms to extract important information from feature models (Batory *et al.*, 2006; Benavides *et al.*, 2010). The automated approach entails mapping the feature models into a specific formal logic-based representation, which becomes inputs to solvers, and analysis operations are performed to obtain useful information. A Solver is a software package that accepts formal representations as inputs and determines some satisfiability criteria (Benavides *et al.*, 2010). Logic representations are classified into description logic, propositional logic, and constraint programming.

i. **Description Logic-** Description logic represents a family of formal languages used to conceptualise, reason about knowledge and are more expressive than propositional logic. Feature models are mapped into description logic formalism and logic reasoners such as RACER or Pellet are used for analysis and provide explanations for the result.

ii. **Propositional Logic-** Propositional logic (PL) is the branch of logic that studies propositions defined over a set of Boolean variables and the logical operators: $\neg$, $\wedge$, $\vee$, $\Rightarrow and$ $\Leftrightarrow$. In the PL approach, the feature models are translated into a propositional formula and solvers are used to perform analysis operations based on the propositional formulae. The propositional formulae is either encoded as a conjunctive normal form (CNF), and then solvers such as satisfiability solvers (SAT solvers) is employed to perform, or as Directed Acyclic Graph (DAG), used by Binary Decision Diagram Solvers (BDD solver) (Benavides *et al.*, 2010; Benavides *et al.*, 2006).

iii. **Constraint Programming**- Constraint programming uses constraints as a programming method to encode and solve Constraint Satisfaction Problems (CSP). Formally, CSP is fined as:

Definition 2.2 (CSP): A Constraint Satisfaction Problem (CSP) is defined as a finite set of variables, each of which is associated with a finite domain, and a set of constraints that restrict the values the variables can simultaneously take.

Feature models are mapped into a CSP model and CSP solvers use constraint programming to find an assignment for each variable that satisfies the constraints (Benavides *et al.*, 2010). The mapping from a feature model to a particular CSP solver is less straightforward than with propositional logic because the encoding structure is solver-dependent. However, the following steps apply (Benavides *et al.*, 2010):

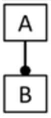
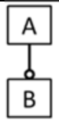
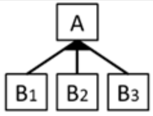
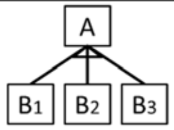
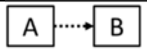
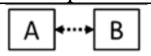
i. **Step 1:** Each feature of the feature model maps to a variable of the CSP with a domain of $[0..1]$ (i.e. true or false), depending on the kind of variable supported by the solver.

ii. **Step 2:** Each relationship in the model is mapped into a constraint depending on the type of relationship.

iii. **Step 3:** The resulting CSP is the one defined by the variables of step 1 and the corresponding domains and constraints that are the conjunction of all precedent constraints plus additional constraint assigning true to the variable that represents the root, depending on the variable's domain.

The rules mapping feature model to propositional logic and CSP are presented in Table 2.12.

Table 2.12: Feature Model Mapping to CSP and PL

Relationships in CEFM	CSP Mapping	PL Mapping
 Mandatory	$A = B$	$A \leftrightarrow B$
 Optional	$if(A = 0)$ $B = 0$	$B \rightarrow A$
 OR	$if(A > 0)$ $Sum\ (B_1, B_2 \ldots B_n) in\ (1 \ldots n)$ $else$ $B1 = 0, B2 = 0 \ldots B_n = 0$	$A \leftrightarrow (B_1 \vee B_2 \vee \ldots \vee B_n)$
 Alternative	$if(A > 0)$ $Sum\ (B_1, B_2 \ldots B_n) in\ (1 \ldots 1)$ $else$ $B1 = 0, B2 = 0 \ldots B_n = 0$	$(B_1 \leftrightarrow (\neg B_2 \wedge \ldots \wedge \neg B_n \wedge A)) \wedge$ $(B_2 \leftrightarrow (\neg B_1 \wedge \ldots \wedge \neg B_n \wedge A)) \wedge$ $(B_n \leftrightarrow (\neg B_1 \wedge \neg B_2 \ldots \wedge \neg B_{n-1} \wedge A))$
 Requires	$if(A > 0)$ $B > 0$	$A \rightarrow B$
 Excludes	$if(A > 0)$ $B = 0$	$\neg(A \wedge B)$

Source: Benavides *et al.* (2010)

c) Automated Analysis Operations on Feature Models

After the transformation of the knowledge model into a formal logic-based representation, mathematical operations based on the semantics of the underlying logic-representation can be performed to derive useful information about the feature model. A number of analysis operations exist (Benavides *et al.*, 2006; Benavides *et al.*, 2010), but the following analysis operations are relevant to the cloud ecosystem context are: *Determine the Satisfiability of a feature model, solutions count,* and *generate all the valid solutions.* Next, each of the operations is discussed in details.

i. **Determine the Satisfiability of a model-** This operation examines the feature model and determines returns a verdict that determines the satisfiability of the feature model, by telling if the feature model is void or not. A feature model is said to be satisfiable, when at least one valid combination, can be derived from it.

ii. **Count Number of Products-** This operation returns the number of valid combinations that can be derived from the feature model. The e-marketplace provider can estimate at every point the number of services that can be offered in the e-marketplace.

iii. **Generate all the valid products-** This operation generates all valid combinations in the feature model that satisfies all the constraints in their interrelationship. In the context of this study, the set of valid combinations forms the set of services from which the user selects a cloud service that approximates user requirements.

II. Feature Modelling for Cloud Service Ecosystem

Based on the foregoing discussions, Figure 2.9 depicts a way of organising ecosystem information into a model for obtaining useful information pertinent to operationalizing the cloud service e-marketplace:

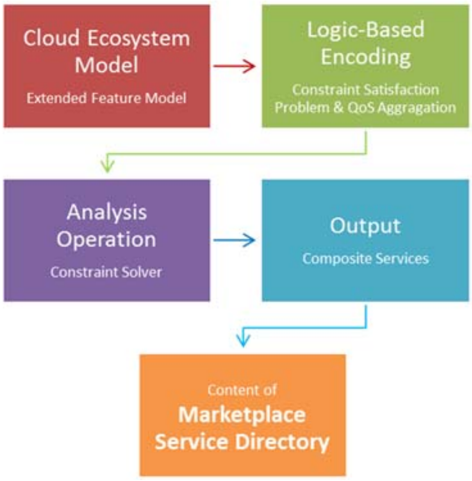


Figure 2.9: Process for Organising and Composing Ecosystem Atomic Services

One way to model the cloud ecosystem is the adopt feature models (Berger *et al.*, 2014); and term Cloud Ecosystem Feature Model (CEFM) can be adopted. The CEFM employs the extended feature model due to its flexibility for modelling of services, their QoS and the constraints that exist among them. This decision is further strengthened by the availability of existing tool support. The CEFM can then be encoded as a formal representation using constraint programming approach. The CSP-based logic encoding was engaged in this study for its suitability for automated reasoning on attributed feature models, such as CEFM. The CSP-based encoding could then be cast into the solver to perform automated analysis of the CEFM. The overall QoS attributes of the valid combinations are determined by the QoS factors of constituent services. The result of the analysis operations is used to update the e-marketplace service directory with candidate solutions that would be offered via the e-marketplace platform (Wittern *et al.*, 2012). However, this approach also automatically captures scenarios of entrants and exits of services. With each case of entrants or exists based on the stated entrance and exit policies of the e-marketplace, the CEFM is altered; and a seamless automated update of the e-marketplace service directory can still be achieved.

2.4.4 Fuzzy-Oriented Elicitation of User QoS Requirements

An accurate elicitation of user requirements involves the interpretation of fuzzy expressions and the use of this information in evaluating service alternatives. The difficulty imposed by expecting users to use exact or crisp values when expressing requirements necessitates the employment of uncertainty theories, such as fuzzy set theory, to effectively capture and interpret the vagueness that characterizes user QoS requirements for services (Qu and Buyya, 2014; Esposito *et al.*, 2016; Sun *et al.*, 2014). To this end, vague QoS preferences or aspirations can be expressed using linguistic terminologies, which is a preferable mode of communicating such requirements (Esposito *et al.*, 2016; Qu and Buyya, 2014; Gatzioura *et al.*, 2012). This section discusses how fuzzy set theory applies in the elicitation of user's QoS preferences and aspirations. More specifically, the preference weights derivation is achieved using the fuzzy pairwise comparison of the fuzzy extension of the AHP technique, Fuzzy AHP (or FAHP). Also, the fuzziness in user's QoS aspirations can be elicited and analysed as a system of fuzzy goals and constraints using fuzzy linguistic variables and linguistic hedges. The decision-making technique used to determine optimal service alternative is based on fuzzy multi-objective optimisation, in which the objectives of the user, which is mainly to maximise their private utility (of the most optimal alternative available) while satisfying their aspiration and constraints. A depiction of a proposed fuzzy decision-making model is shown in Figure 2.10.

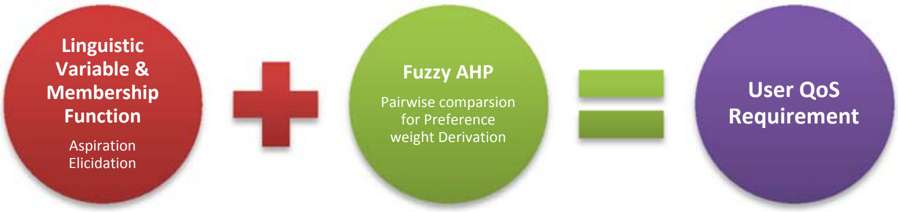

Linguistic Variable & Membership Function
Aspiration Elicidation
Fuzzy AHP
Pairwise comparsion for Preference weight Derivation
User QoS Requirement

Figure 2.10: User Requirements Elicitation Model

I. Overview of Fuzzy Set Theory

Many classes of objects encountered in the real world do not have precisely defined inclusion criteria, e.g. the class of expensive holiday resorts, the class of cheap cars, etc., and such class expressions underlie human judgements, particularly in decision making (O'Hagan, 1993). Fuzzy Theory, proposed by Zadeh (Zadeh, 1974), is one way to handle such vagueness. The use of fuzzy theory is a potent tool that allows us to represent objects or concepts in a vague or ambiguous way, similar to a human concept and thought process (Bai and Wang, 2006). However, a formal definition of a fuzzy set is given as follows:

Definition 2.3: *Let* $X = \{x\}$ *denote a collection of objects denoted generically by* x. *Then* $fuzzy\ set\ A\ in\ X$ *is a set of ordered pairs:*

$$A = \{(x, \mu_A(x))\}, \qquad x \in X \tag{2.1}$$

$\mu_A(x)$ is the grade membership of $x\ in\ A$, and $\mu_A: X \to M$ is a function from X to a space M, called the membership space; M represents the interval [0,1], with 0 and 1 representing the lowest and highest membership grades respectively.

a) *Basic Definitions of Fuzzy Sets*

Definition 2.4 (Intersection): Intersection (or *logical and*) is the membership function of the intersection of two fuzzy sets A and B defined as:

$$\mu_{A\cap B}(X) = \min(\mu_A(x), \mu_B(x)), \quad \forall x \in X \tag{2.2}$$

Definition 2.5 (Union): Union (or *exclusive or*) is the membership function of the union of two fuzzy sets A and B defined as:

$$\mu_{A\cup B}(X) = \max(\ \mu_A(x), \mu_B(x)),\ \ \forall x\ \in X \tag{2.3}$$

b) *Linguistic Variable*

To overcome the complexity involved in quantifying certain real world phenomena, Zadeh (1974) introduced the notion of linguistic variables to conveniently describe and quantify real-world concepts using linguistic terminologies. A linguistic variable is decomposed into a set of linguistic terms or values, and each term (or value) represents a fuzzy set and makes up a portion of the variable's domain (or Universe of Discourse). A linguistic term can be described using a fuzzy number, connecting the linguistic variable to a base numeric value, and are defined by an associated membership function. Formally, the linguistic variable is defined as follows:

Definition 2.9: A linguistic variable is characterised by a quintuple $(x,\ T(x),\ U,\ G,\ \widetilde{M})$ in which x is the name of the variable, $T(x)$ (or simply T) denotes the term set of x, that is, the set of names of linguistic values of x. Each of these values is a fuzzy variable, denoted generically by X and ranging over a universe of discourse U, which is associated with the base variable u; G is a syntactic rule (which usually has the form of a grammar) for generating the name, X, of values of x. M is a semantic rule for associating with each X its meaning. $\widetilde{M}(X)$ is a fuzzy subset of U. A particular X, that is, a name generated by G, is called a term.

c) *Fuzzy Numbers*

A much larger class of fuzzy sets represents approximate numbers of one type or another. Some of these fuzzy sets are explicitly "fuzzified" numbers, whereas others simply represent fuzzy numeric intervals over the domain of a particular variable. Fuzzy numbers can take many shapes: bell curves, triangles, and trapezoids. Within each of these shapes, the actual meaning of the fuzzy set depends on the width or spread of the set itself. The flexibility and robustness of fuzzy sets are made possible by fuzzy numbers. A bell-shaped, triangular-shaped, or trapezoid-shaped fuzzy set represents a central value and is, in essence, a fuzzy number.

i. **Bell Shaped Fuzzy Number-** Figure 2.11 illustrates a typical bell-shaped fuzzy number. This is a numeric quantity, Around 20. The fuzzy set About 20 shows two principal attributes of fuzzy numbers: a central value and a degree of spread around the value.

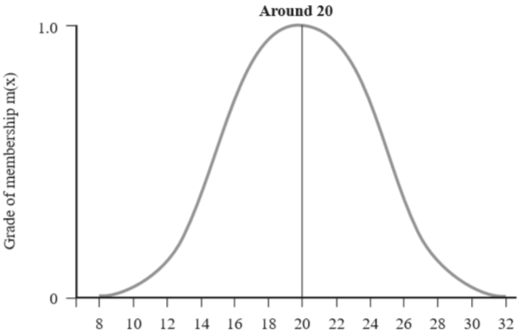


Figure 2.11: Bell-shaped fuzzy set: 'Around 20'
Source: Cox (2005)

ii. **Trapezoid fuzzy number-** The descriptions of a trapezoidal number are somewhat different from the bell and triangular numbers because the set does not hinge around a single central crisp value. However, a trapezoidal fuzzy number can be considered a special case of the triangular fuzzy set (with a plateau width of zero) (Cox, 2005). The trapezoidal fuzzy number is defined by:

$$\mu_{\tilde{M}}(x) = \begin{cases} \dfrac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \dfrac{d-x}{d-c}, & c \leq x \leq d \\ 0, & Otherwise \end{cases} \quad \textbf{(2.4)}$$

iii. **Triangular fuzzy number-** Triangular fuzzy number (TFN) is popular for its low computational cost; however, it is less flexible than a bell-shaped fuzzy number. The form of a triangular fuzzy number is $\tilde{a} = (l,\ m,\ u)$, where $l \leq m \leq u$, and l is the lower bound of $\tilde{a}$, m is the middle value of $\tilde{a}$, while u is the upper bound of $\tilde{a}$. A TFN can be described by:

$$\mu_{\tilde{M}}(x) = \begin{cases} \dfrac{x-l}{m-l}, & l \leq x \leq m \\ \dfrac{u-x}{u-m}, & m \leq x \leq u \\ 0, & otherwise \end{cases} \quad \textbf{(2.5)}$$

d) Membership Function

A Membership Function (MF) is considered as a curve that defines how a crisp input is mapped to a membership grade. Each fuzzy set, quantified by a linguistic variable, is defined by an associated membership functions. There are several types of membership functions, which includes (but not limited to): triangular, trapezoidal, Gaussian, bell-shaped, and sigmoidal MF. The type of MF to employ depends on the specific situation (Bai and Wang, 2006).

i. **Triangular membership function-** A triangular MF is described by three parameters a, b and c; where a and c, is located at the base of the triangle, and the parameter b locate the peak. Variable x is the crisp value, whose membership grade is to be determined by the membership function within the UoD. The triangular MF is defined as follows:

$$f(x; a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{2.6}$$

ii. **Trapezoidal membership function-** A trapezoid MF is described by four parameters a, b, c and d; where a and d, is located at the base of the trapezoid, and the parameters b and c is located at the 'shoulder'. The shoulder of a trapezoid can either be narrow or wide. Variable x is the crisp value, whose membership grade is to be determined by the membership function within the UoD. The trapezoid MF is defined as follows:

$$f(x; a, b, c, d) = \max\left(\min\left(\frac{x-a}{b-a}, \quad 1, \quad \frac{d-x}{d-c}\right), 0\right) \tag{2.7}$$

iii. **Gaussian membership function-** A Gaussian MF is described by two parameters c and σ; where c, is the centre of the distance from the origin, corresponding to the centre of the graph, σ is the width of the graph, while x is the crisp value, whose membership grade is to be determined by a membership function. The Gaussian MF is defined as follows:

$$f(x; c, \sigma) = e^{\frac{-(x-c)^2}{2\sigma^2}} \tag{2.8}$$

iv. **Bell-shaped membership function-** A bell-shaped MF has a symmetrical shape and it is described by three parameters a, b and c. The parameter c is the centre of the curve, b is usually positive, (a negative b value would produce an inverted bell), while a represents the width of the curve. The bell-shaped MF is smooth and non-zero at all possible points of x. The bell-shaped MF is defined as follows:

$$f(x; a, b, c) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}} \tag{2.9}$$

v. **Sigmoidal membership function-** Generally, a sigmoidal MF is open to the left or right and has two parameters a and c. The parameter c is the centre of the curve, while a determines the gradient of the curve at crossover point $x = c$; and determines the direction (left or right) of the opening of the curve (when a is positive, MF curve opens to the right and left otherwise). In linguistic terms, the MF can be used to represent concepts such as *'very large'* or *'very negative'*, depending on the sign of parameter a. The sigmoidal MF is described as follows:

$$f(x; a, c) = \frac{1}{1 + e^{-a(x-c)}} \tag{2.10}$$

II. Preference Weight Derivation Using Fuzzy Pairwise Comparison

Although the AHP method proposed by (Saaty, 1980) allows for some flexibility in judgment by providing intermediate values in the Saaty's discreet scale, it requires that users make comparison judgements based on the crisp or exact numerical values (Cakir and Canbolat, 2008). However, in many practical cases, the human judgment is shrouded with impression and vagueness and users' decision- makers might be reluctant or unable to assign exact numerical values to the comparison judgements (Mikhailov and Tsvetino, 2004). Comparison judgement using on crisp numerical values lacks the flexibility and robustness required to effectively capture the vague perception inherent in human judgement, and sometimes, lead to unsatisfactory decisions (Yang and Chen, 2004; Cakir and Canbolat, 2008; Torfi *et al.*, 2010; Javanbarg *et al.*, 2012; Mikhailov and Tsvetino, 2004). It has been proposed that a better approach to capturing the user's claim about the relative importance of criteria is to define comparison ratios as fuzzy numbers (Yang and Chen, 2004; Cakir and Canbolat, 2008; Torfi *et al.*, 2010; Javanbarg *et al.*, 2012;

Mikhailov and Tsvetino, 2004). The application of a fuzzy model to handle the user's vague perception of priorities of all QoS factors is presented in this section.

a) Main Steps in Fuzzy AHP

The main steps of fuzzy AHP are as follows: Establish the dimension for evaluation using fuzzy numbers and linguistic variables; Perform pairwise comparison judgments; Check consistency of judgments, and determine the fuzzy priority weights.

i- Step 1: Evaluation Dimension using Fuzzy Number and Linguistic Variables

The blurriness in human judgement can be best captured as an approximation of the crisp or exact comparison ratio; such that, when an exact comparison ratio a_{ij} is represented as a fuzzy number, $\tilde{a}_{ij}$, the assessment of users' judgement can correspond to '*about* a_{ij}' or 'close to a_{ij}' which is closer to how humans think. Fuzzy linguistic variables are used to define comparison judgement values and to represent the underlying fuzzy numbers; and Triangular Fuzzy Numbers (TFN), characterised by triangular membership function, are popularly used in this regards. As earlier discussed, linguistic variables are variables, whose values are words or sentences in a natural language and each fuzzy comparison judgment can be performed by using linguistic terms such as "absolutely important", "very strongly important", "essentially important", "weakly important", and "equally important" with respect to a fuzzy comparison scale (as shown in Table 2.13). So rather than users making comparison judgements mapped to exact values, Nine fuzzy linguistic terms, defined by TFN would naturally capture the imprecision and vagueness inherent in human judgment and preferences (Cakir and Canbolat, 2008).

ii- Step 2: Perform Pairwise Comparison Judgements

Based on the established dimensions, users can use linguistic terms to evaluate the importance of QoS criteria, thus performing the mutual pairwise comparison for each of the QoS factors. The user assigns a fuzzy weight that reflects the user's subjective preference using fuzzy linguistic terms. The total number of comparisons is $n(n-1)/2$, where n is the number of criteria, and the output of the pairwise comparisons is captured in a comparison matrix as shown in Figure 2.12.

Table 2.13: Fuzzy Version of Saaty's 9-point Comparison Scale

Linguistic Term	Description	Comparing Criterion i to Criterion j			Comparing Criterion j to Criterion i (Reciprocal)		
		Saaty Scale	Fuzzy Number	TFN	Saaty Scale	Fuzzy Number	TFN
Equally Important	Criterion i is fuzzily equally as important as criterion j	1	$\tilde{1}$	$(1, 1, 2)$	1	$\tilde{1}^{-1}$	$\left(\frac{1}{2}, \frac{1}{1}, \frac{1}{1}\right)$
Moderately Important	Criterion i is fuzzily moderately more important than criterion j	3	$\tilde{3}$	$(2, 3, 4)$	$\frac{1}{3}$	$\tilde{3}^{-1}$	$\left(\frac{1}{4}, \frac{1}{3}, \frac{1}{2}\right)$
More Important	Criterion i is fuzzily more important compared to criterion j	5	$\tilde{5}$	$(4, 5, 6)$	$\frac{1}{5}$	$\tilde{5}^{-1}$	$\left(\frac{1}{6}, \frac{1}{5}, \frac{1}{4}\right)$
Strongly Important	Criterion i is fuzzily more strongly important than criterion j	7	$\tilde{7}$	$(6, 7, 8)$	$\frac{1}{7}$	$\tilde{7}^{-1}$	$\left(\frac{1}{8}, \frac{1}{7}, \frac{1}{6}\right)$
Absolutely Important	Criterion i is fuzzily absolutely more important than criterion j	9	$\tilde{9}$	$(8, 9, 9)$	$\frac{1}{9}$	$\tilde{9}^{-1}$	$\left(\frac{1}{9}, \frac{1}{9}, \frac{1}{8}\right)$
Intermittent Values between two adjacent scales		2	$\tilde{2}$	$(1, 2, 3)$	$\frac{1}{2}$	$\tilde{2}^{-1}$	$\left(\frac{1}{3}, \frac{1}{2}, \frac{1}{1}\right)$
		4	$\tilde{4}$	$(3, 4, 5)$	$\frac{1}{4}$	$\tilde{4}^{-1}$	$(\frac{1}{5}, \frac{1}{4}, \frac{1}{3})$
		6	$\tilde{6}$	$(5, 6, 7)$	$\frac{1}{6}$	$\tilde{6}^{-1}$	$(\frac{1}{7}, \frac{1}{6}, \frac{1}{5})$
		8	$\tilde{8}$	$(7, 8, 9)$	$\frac{1}{8}$	$\tilde{8}^{-1}$	$(\frac{1}{9}, \frac{1}{8}, \frac{1}{7})$
Criterion i or criterion j is compared to itself (i.e. $i = j$, representing the diagonals)		1	$\acute{1}$	$(1, 1, 1)$	1	$\acute{1}^{-1}$	$(1, 1, 1)$

Source: Ayhan (2013)

$$\widetilde{\boldsymbol{A}} = \begin{bmatrix} \widetilde{\boldsymbol{a}}_{11} & \widetilde{\boldsymbol{a}}_{12} & \cdots & \widetilde{\boldsymbol{a}}_{1n} \\ \widetilde{\boldsymbol{a}}_{21} & \widetilde{\boldsymbol{a}}_{22} & \cdots & \widetilde{\boldsymbol{a}}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{\boldsymbol{a}}_{n1} & \widetilde{\boldsymbol{a}}_{n2} & \cdots & \widetilde{\boldsymbol{a}}_{nn} \end{bmatrix}$$

Figure 2.12: Fuzzy Comparison Matrix
Source: Ayhan (2013)

iii- Step 3: Check Consistency of Judgement

Checking for consistency in comparison judgement is an important step before deriving the priorities from the pairwise comparison matrix. Saaty's AHP requires that the regularity of the crisp comparison judgements be checked to ensure consistency. The Consistency Ration (CR) is employed to check consistency in comparison judgement, and is determined using the formula $CR = \frac{CI}{RI}$.

Where CI is the Consistency Index and defined as: $CI = \frac{\lambda_{max} - n}{n-1}$ where λ_{max} is the largest Eigen value of comparison matrix and RI is the random index, a 9-point scale consistency index generated through pairwise comparison. The value of CR is expected to be ≤ 0.1

for a matrix larger than 4 × 4 (Saaty, 1990). However, for fuzzified comparison matrix, Csutora *et al.* (Csutora and Buckley, 2001) provided a proof that, for a fuzzy, positive, and reciprocal matrix $\tilde{A} = [\tilde{a}_{ij}]$, where $\tilde{a}_{ij} = (\alpha_{ij}, \beta_{ij}, \gamma_{ij}, \delta_{ij})$, (a trapezoidal fuzzy number) select a value $\tilde{a}_{ij} \in [\beta_{ij}, \gamma_{ij}]$ to generate a corresponding crisp matrix $A = [a_{ij}]$. The consistency of the generated matrix A confirms the consistency of matrix $\tilde{A}$. According to (Sun *et al.*, 2014), $\beta_{ij} = \gamma_{ij}$, for TFN, and the crisp matrix A can be generated using values $a_{ijm} = (\beta_{ij} + \gamma_{ij}/2)$ of all the fuzzy numbers in fuzzy matrix $\tilde{A}$, while the consistency ratio is computed.

iv- Step 4: Determine Fuzzy Priority Weights to obtain Crisp Priority Weights

The fuzzy priority vector $\widetilde{w}^T$can be obtained by applying prioritization methods, after comparison matrix $\tilde{A}$ passes the consistency check. Prioritization is the process of deriving the priority values for column vector $w^T = [w_i], i = 1, \dots, n$ from the comparison judgment matrix $\tilde{A}$. There are two ways in which priorities can be derived (Zhu *et al.*, 2012):

i. By deriving *fuzzy weights* from the comparison matrix. For example, the Logarithmic Least Square (LLS) method (Van Laarhoven and Pedrycz, 1983), Lambda-Max Method and, the Geometric means method (Buckley's Method) (Buckley, 1985).

ii. Deriving a set of *crisp weights* directly from the comparison matrix. For example, the Synthetic Extent Analysis method (SEA) (Chang, 1996), and the fuzzy preference programming method (Mikhailov, 2003).

In the former category, fuzzy weights are converted to crisp weights by applying defuzzification methods, whereas the defuzzification step is not required in the latter.

b) Overview of Buckley's Prioritisation Method

Buckley (Buckley, 1985) initially investigated fuzzy weights and fuzzy utility for AHP technique, extending AHP by the geometric mean method to derive the fuzzy weights. The Buckley's method considered a fuzzy positive reciprocal matrix $A = [a_{ij}]$ extending the geometric mean technique to define the fuzzy geometric mean of each row $\tilde{r}_i$ and fuzzy weight $\widetilde{w}_i$, corresponding to each criterion as follows:

$$\tilde{r}_i = \left[\prod_{j=1}^{n} \tilde{a}_{ij}\right]^{\frac{1}{n}}, i = 1,2, \ldots n \tag{2.11}$$

And the fuzzy weight is obtained by

$$\widetilde{w}_i = \widetilde{\tilde{r}_i} \otimes (\tilde{r}_1 \oplus \tilde{r}_2 \oplus \tilde{r}_3 \oplus \ldots \oplus \tilde{r}_n), i = 1,2, \ldots, n \tag{2.12}$$

Where $\tilde{a}_{\mathrm{ij}}$ is fuzzy comparison value of dimension i to criterion j, thus, $\tilde{r}_i$ is a geometric mean of fuzzy comparison value of criterion i to other criteria; $\widetilde{w}_i$ is the fuzzy weight of the i^{th} criterion, can be indicated by a TFN, $w = (lw_i, mw_i, uw_i)$. The lw_i, mw_i, and uw_i stand for the lower, middle, and upper values of the fuzzy weight of the i^{th} dimension (Sun, 2010). Since the fuzzy weight $\widetilde{w}_i$is a fuzzy number, defuzzification is applied to obtain crisp values using centre of area method. The result is then normalized to obtain the weight vector. More details on the application of the Buckley's method are available in (Ayhan, 2013).

III. Aspiration Elicitation as Fuzzy Goals and Constraints

Fuzzy decision making is concerned with the decision-making process in which the goals and/or the constraints are fuzzy in nature. In other words, the goals and/or constraint constitute set of elements whose boundaries are not sharply defined, as they refer to an objective which can be characterised as a fuzzy set in an appropriate space. Examples of a fuzzy goal and constraints are "The Cost of the service should be *low*", or "Cost should be *close to* c; in the *vicinity* of c or *substantially less than* c", where c is a specified constant or cost value as indicated by the user. The linguistic words term "low", "*vicinity of*", "*close to*" and "*substantially less than"* represent the source of fuzziness and model human judgement. Bellman and Zadeh (Bellman and Zadeh, 1970) were the first to explore decision-making problems in a fuzzy environment, and they introduced the concepts of fuzzy decision based on fuzzy goals and fuzzy constraints. This is done with the assumption that the goals and constraints are fuzzy, but the system under control (in this case, the cloud service e-marketplace) is deterministic (Yager, 1977; Bellman and Zadeh, 1970).

User's QoS aspiration towards the selection of a cloud service can be modelled using fuzzy goals and constraints. Modelling user's aspiration as a combination of fuzzy goals and constraints, based on the proposal in (Bellman and Zadeh, 1970; O'Hagan, 1993) would allow cloud users to articulate QoS aspirations in a way that captures the vagueness in such judgement. By illustration, a simple example of a fuzzy goal would be that *Security* should be *high*, and a fuzzy constraint could be, it should be that the QoS value of security should be *substantially larger than* a specific crisp threshold value, or *in the vicinity* of a particular threshold value, or *approximately within a given range*.

As earlier mentioned, the italicised items represent the fuzziness inherent in the elicitation process and are defined by linguistic variables, linguistic terms and linguistics hedges characterised by different membership functions. Linguistic hedges are employed to modify membership functions to further allow user naturally express their QoS aspirations. Approximation Hedges are used as constraints on QoS goals. For example, the user can express that security should be high and uses the approximation hedges to indicate that the QoS value of security should be *around*, *about* or *in the vicinity of* a specific threshold. The decision maker is often faced with the problem of selecting among a set (usually finite) of alternatives while simultaneously satisfying a set of objective criteria (goals) and observing (not violating) a set of constraints. The main contribution of Bellman and Zadeh (1970) to this theory was in recognising that a 'good' decision had to satisfy both goals and constraints and that for decision purposes, they should be treated alike. That is to say, that a 'good' decision had to satisfy some conjunctive form of goals and constraints associated with the decision-making environment. An optimal decision would be one that 'best' satisfied all the criteria in some sense.

For example, Let $X = \{x\}$, a set of alternatives. Then the fuzzy goal G is represented as a fuzzy set with the triangular membership function, denoted as:

$$\mu_G(x) = \max\left(\min\left(\frac{x-l}{m-l}, \frac{u-x}{u-m}\right), 0\right) \tag{2.13}$$

Where l, m and u respectively correspond to values lower, medium and upper values of a fuzzy set.

In the same regard, a fuzzy constraint, C, could be that "the value of x should be in the vicinity of a", representing a fuzzy set whose membership function is bell-shaped, given as:

$$\mu_C(x) = \frac{1}{(1 + (x - a)^4)} \tag{2.14}$$

Where a represents a specific constant indicated by the decision maker; the intersection of both fuzzy sets of the goal, G and constraint, C is denoted as $G \cap C$. The membership function that represents the intersection is determined by:

$$\mu_{G \cap C}(x) = \min\left[\max\left(\min\left(\frac{x - l}{m - l}, \frac{u - x}{u - m}\right), 0\right), \frac{1}{(1 + (x - a)^4)}\right] \tag{2.15}$$

A formal and more generic definition is presented next:

Definition 2.6: (Bellman and Zadeh, 1970): Suppose there are n Goals $(G_1 \ldots G_n)$ and m Constraints $(C_1, \ldots, C_m)$, then the resultant decision D is the intersection of all Goals and Constraints, denoted as:

$$D = G_1 \cap G_2 \cap \cdots \cap G_n \cap C_1 \cap C_2 \cap \cdots \cap C_m \tag{2.16}$$

Corresponding to:

$$\mu_D(x) = \min\left(\mu_{G_i}(x), \mu_{G_2}(x) \ldots, \mu_{G_n}(x), \mu_{C_1}(x), \mu_{C_2}(x), \ldots, \mu_{C_m}(x)\right) \tag{2.17}$$

The fuzzy set of alternatives is populated by the intersection of goals and constraints, better still, "a confluence of goals and constraints" according to (Bellman and Zadeh, 1970). A maximising decision is a point in the set of alternatives at which the membership function of a fuzzy decision attains its maximum value. The optimal alternative is found using a maximising decision D^* corresponding to:

$$\mu_{D^*}(x) = \arg\{\max_{x \in X} \mu_D(x)\} \tag{2.18}$$

The maximising decision is obtained from the value of x with the highest membership grade in the decision fuzzy set D.

To further explain the fuzzy decision making concept discussed in this section, a simple example is presented on how to elicit the user QoS aspiration using linguistic variables and hedges. Assume that the range of values that covers the *Availability* of a cloud service is between $0\%\ to\ 100\%$. These ranges can be further divided into three sub-ranges, which are:

i. Low Availability: 0% ~ 40%

ii. Medium Availability: 30% ~ 75%

iii. High Availability: 70% ~ 100%

The sub-ranges can be converted to linguistic terms, $Avail_{LOW}$, $Avail_{MEDIUM}$ and $Avail_{HIGH}$, and can be defined by trapezoidal membership functions.

Suppose the User's Goal and constraints on Availability are as follows:

i. Goal One (G): High *Availability*

ii. Constraint One (C): The value of availability must be close to 99%.

The membership function for the goal G is defined as follows:

$$\mu_G(x) = \max\left(\min\left(\frac{x-a}{b-a}, \quad 1, \quad \frac{d-x}{d-c}\right), 0\right) \tag{2.19}$$

Where $a = 70\%, b = 75\%,\ c = 85\%$ and $d = 100\%$, representing the trapezoidal fuzzy number of fuzzy sets $Avail_{HIGH}$.

The membership function of the Constraint C is defined as follows:

$$\mu_C(x) = \frac{1}{1 + 10(x-\psi)^2} \tag{2.20}$$

Where $\psi = 99\%$ as indicated in constraint C; the user's aspiration is said to be in the decision set of the intercession of the goal and constraint:

$$\mu_D(x) = \mu_G(x) \cap \mu_C(x) = \min\left(\mu_G(x), \mu_C(x)\right) \tag{2.21}$$

The optimal value of availability that approximates user's aspiration corresponds to any x in the support of D^*, which can be formulated as finding the value of x maximizes the intersection membership function, or equivalently:

$$\max \mu_D(x) = \max min\left(\mu_G(x), \mu_C(x)\right) \tag{2.22}$$

Using MOEA Framework, a Java library of Multi-Objective Evolutionary Algorithms, in NetBeans, the optimal solution that satisfies both the goal and constraint is $x = 98.0\%$ at $\mu_D(x) = 0.1224$. The concept of fuzzy decision is applicable in determining the approximate user QoS aspirations to evaluate cloud services and determine the best matches. Aspirations are then elicited without the user explicitly specifying actual values, but rather using natural, everyday language enabled by fuzzy set theory. Since humans naturally use and respond to fuzzy concepts, using fuzzy terms to express QoS requirements are more convenient and easier than using crisp numeric values.

2.4.5 Fuzzy Optimisation for QoS-based Service Evaluation

The next step after obtaining user's QoS requirements is to evaluate each service alternative with respect to user's QoS requirements. The evaluation forms the basis by which users can select the 'most optimal' service(s), and service selection depends on the relative importance given to each QoS attributes and QoS aspiration specified by the user (Rehman *et al.*, 2011). Since cloud services are characterised by multiple QoS attributes, utility functions can be employed to evaluate the overall quality of a given service. The utility function maps the overall performance of a cloud service into a single real score value, clarifying the goodness or usefulness of each alternative; and alternatives are ranked based on these values.

Although some MCDM approaches discussed earlier can be used to evaluate service, many of these approaches only take into consideration the priorities of user's QoS preferences, which is captured as importance weights, and do not cater for user's aspiration in the service evaluation process. For example, AHP does not consider user's ideal QoS aspiration value for each criterion; SAW and TOPSIS are used to derive performance scores and alternatives are ranked to determine the alternative with the 'best' performance without recourse to user's aspiration; moreover, these approaches are best applicable when the number of alternatives is very few. An emergent perspective is a multi-function service evaluation that is capable of considering both users' preferences and aspiration in the service evaluation process; and can be applied to evaluate a large set of alternatives, as is the case in cloud service e-marketplace. Such evaluation model can simultaneously rank service alternatives with respect to the 'ideal' alternative (based on the available QoS information) and the user's specified QoS preferences and aspiration. To this end, an SAW-based technique (Yoon and Hwang, 1995) can be combined with a proximity-based function (based on a similarity metric) to evaluate each service alternative along with their QoS performance, with respect to user's QoS requirements.

Owing to the simplicity and practicality of the SAW technique, it is popularly used to derive a performance score as a weighted sum of all QoS attributes for each alternative. The performance score forms the basis to estimate the 'goodness' of an alternative and for benchmarking each alternative against the 'ideal' alternative. The ideal alternative is defined as the alternative with the best value for all QoS criteria and usually does not quite exist (Rehman *et al.*, 2012). The application of SAW in evaluating service alternatives supposes that the alternative with the highest performance score would be selected; however, one would observe that the service alternative with the highest utility may not necessarily correspond or approximates user's QoS aspiration.

Another way to facilitate selection of cloud service is to rank alternatives in accordance to their nearness to user's QoS requirements; users can then make a selection from the ranked list (Rehman *et al.*, 2011). Similarity or distance metrics are used to determine to what extent two vectors are alike and can be applied to determine the nearness of all services available on the e-marketplace to user-defined QoS requirements (Mirmotalebi *et al.*, 2012). Proximity-based service evaluation involves a comparison between the user's requirement and all service alternatives, using a similarity metric to determine the service alternative that best matches user requirements. Based on the use of similarity computation, Rehman *et al.* (2011) identified three possible outcomes: i) Exact match with user requirements. ii) Generally lower values than the user requirement and iii) Generally higher values than the user requirement. Based on these outcomes, Rehman *et al.* (2011) noted that the use of similarity metrics would suffice for outcomes (i) and (ii), but would return dissimilarity for outcome three, in which case the QoS of the service alternative exceeds the user requirement. So, an optimal alternative would be that alternative which simultaneously maximises the utility function *as much as possible* and closely approximates similarity with user’s aspiration or *closest to* the user’s QoS requirements. Therefore, the search for an optimal alternative gives rise to multiple objective programming problems, with fuzzy goal and constraint; and can be modelled and solved as a fuzzy multi-objective optimisation problem.

A fuzzy multi-objective programming is a problem that involves two or more conflicting fuzzy objective functions that must be simultaneously optimised in the face of some set of constraints. The sources of fuzziness in the objective functions in this optimisation problem are the word phrases ‘*as much as possible*’ and ‘*closest to*’. Therefore, solving fuzzy multi-objective optimisation problems requires that both SAW and distance-based functions are transformed into a fuzzy goal and constraints based on the fuzzy decision making symmetric model proposed by Bellman and Zadeh (1970). Therefore the two conflicting goals and constraint represented as functions are: seeking an alternative with 1) highest utility and 2) nearest to user’s ideal requirements.

Traditional optimisation techniques and methods have been successfully applied for years to solve problems with a well-defined structure/configuration, sometimes known as hard systems. Such optimisation problems are usually well formulated by crisply specific objective functions and specific system of constraints, and solved by precise mathematics. Unfortunately, real world situations are often not deterministic. In the light of this, traditional models and solutions to optimisation problems do not reflect the real world actualities, as they are rigid, confining the solution space, reduces the possibility to make trade-offs, and sometimes cannot find an optimal solution (Oltean, 2004). In cases where optimisation goals and/or constraints are vaguely expressed, the optimisation problem cannot be effectively solved by formulating the problem using traditional optimisation techniques (Tang *et al.*, 2004). A better approach is to use fuzzy sets to define optimisation objectives, associating the goals and/or constraints with one or two fuzzy sets, whose membership functions will represent the corresponding fuzzy objective functions. Modelling and optimisation under a fuzzy environment are called fuzzy modelling and fuzzy optimisation (Tang *et al.*, 2004).

I. Fuzzy Modelling and Fuzzy Optimisation

Solving problems under a fuzzy environment involves two tasks: fuzzy modelling and fuzzy optimisation. The aim of fuzzy modelling is to construct a suitable model based on the peculiarity of the problem and analysis of the fuzzy information. Fuzzy optimisation aims at solving the fuzzy model ‘optimally’ using optimisation techniques and/or tools in terms of their membership functions. Six of the 7-step methodology elaborated in (Tang *et al.*, 2004), was employed to outline the application of fuzzy optimisation for the cloud service evaluation model proposed in this thesis.

II. Fuzzy Optimisation Problems: Modelling

The aspect which the fuzziness affects determines how fuzzy optimisation problems are classified (Tang *et al.*, 2004). Tang *et al.* (2004) further stated that the fuzziness affects the goals, constraints and coefficients of a fuzzy optimisation problem. Fuzziness in fuzzy goal is goals that are usually expressed vaguely, towards a specific aspiration level, which gives the target value of the objective function some flexibility e.g. the target value of the objective function $f(x, r)$ should be maximised as much as possible. The phrase, ‘as much as possible’ removes the rigidity of ‘maximise’ and gives the target value some flexibility. Fuzziness in fuzzy constraints refers to the system of constraints that gives a degree of tolerances and flexibilities through the following relational operators $\tilde{\leqq}$, $\tilde{\geqq}$ or $\cong$; Fuzzy coefficients may appear in the objective function and/or the system of constraints. Formally the fuzzy optimisation problem can be defined as:

Definition 2.7: (Tang *et al.*, 2004): Let universe $X = \{x\}$ be a set of alternatives, X_1 a subset or a fuzzy subset of X. The objective/utility function is a mapping $f : X_1 \rightarrow L(R)$, where $L(R)$ is a subset or a class of fuzzy subsets of real value set R, the feasible domain is described by a subset or a fuzzy set $C \subset X$, with a membership function $\mu_C(x) \in [0,1]$, which denotes the degree of feasibility of x. In this case, a fuzzy optimization problem may be generally expressed as:

$$f(x, r) \rightarrow \max_{x \in C} \qquad \textbf{(2.23)}$$

Where r is either a crisp constant or a fuzzy coefficient; the objective is to find the value of x that maximizes $f\ (x, r)$, and can be solved by the approaches presented in the next section.

III. Fuzzy Optimisation Problem: Solutions

Tang *et al.* (2004) have classified approaches to solving fuzzy optimisation into symmetric and asymmetric approaches. In contrast to asymmetric approaches, symmetric solution approaches handle fuzzy goals and constraints involved in the problem alike (Zimmermann, 1975). Symmetric approaches based on the fuzzy decision (Bellman and Zadeh, 1970) are approaches developed originally to deal with decision-making problems with fuzzy goals and fuzzy constraints, based on the concept of the fuzzy decision, as proposed by (Bellman and Zadeh, 1970). The fuzzy decision is defined as a fuzzy set of alternatives resulting from the intersection of the goals and the constraints. By introducing the fuzzy decision D, the solution to the fuzzy optimization problem can be interpreted as the intersection of the fuzzy goal and the fuzzy constraints, i.e. $D = G \cap C,$ where $\cap$ is a conjunctive operator, assuming different definitions and meanings in different practical application depending on the definitions of the conjunctive operator $\cap$. The membership function of the fuzzy decision is formulated as:

$$\mu_D(x) = \mu_G(x) \cap \mu_C(x), \qquad \forall x \in X \tag{2.24}$$

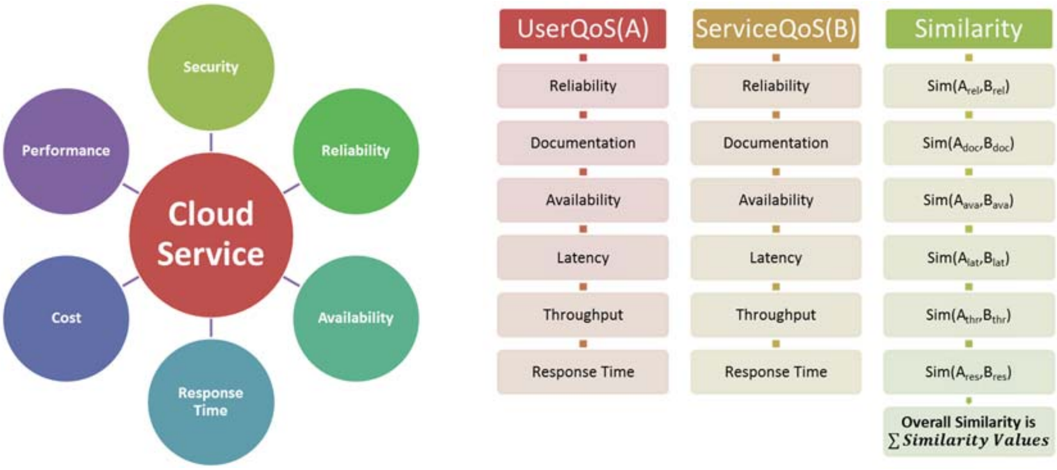
Where μ_G and μ_C are the membership functions of the fuzzy goals and the fuzzy constraints respectively, and preferences are involved. A maximizing decision x^* is then defined to be an alternative with the highest membership in the fuzzy decision D, i.e. $\mu_D(x^*) = \max \mu_D(x), \forall x \in X.$

More generally, maximising decision x^* can be determined by

$$\mu_D(x^*) = \bigcup_{x \in X} \mu_D(x) \tag{2.25}$$

IV. Utility Functions to enable Cloud Service Selection

Two utility functions based on SAW method and Euclidean metrics can simultaneously serve as objective functions in order to evaluate the performance of cloud services with respect to user requirements. SAW is one of the most popular methods of solving MCDM problems and can be used to determine the utility of alternatives. Also, the similarity is a measure of proximity between two or more objects or variables (Ayeldeen *et al.*, 2015) and it has been applied in domains that require distance computation. The notion of similarity considered here is between vectors with the same set of QoS properties, which might differ in their QoS values i.e. users' QoS requirement and service QoS description. The similarity between the user's QoS requirement and QoS description vector of a cloud service is the sum of similarities between each of the corresponding QoS attributes of the vectors (see Figure 2.13).



(a) Cloud service with QoS attributes **(b) Notion of Similarity**

Figure 2.13: Similarity Computation based on QoS Attributes

Suppose X is a vector representing values of the user's QoS aspirations; and Y is a vector of values of QoS attributes of a cloud service $s_i$belonging to service list S, such that $X = (x_1, x_2, \dots x_m)$ and $Y = (y_1, y_2, \dots y_m)$; where x_m and y_m corresponds to the value of the m^{th} QoS attribute of the users requirement and QoS attribute of the cloud Service s_i respectively, then Euclidean defined as follows:

$$EUD\ (x, y) = \sqrt{\sum_{i=1}^{m} (x_i^2 - y_i^2)} \tag{2.26}$$

Although there are several distance metrics in the literature, the Euclidean metrics is often applied to compute distance in a multidimensional space. However, the exponential Euclidean function is applied in other to reduce the effect of the value for each QoS attribute on the similarity score as the values of the QoS attributes exceeds or fall below the user's QoS requirements. Therefore, the exponential Euclidean function proposed and used in this study is given as follows:

$$eEUD(x,y) = \sqrt{\sum_{i=1}^{m} e^{(x_i^2 - y_i^2)}} \tag{2.27}$$

An emergent perspective posits that services should be evaluated on the basis that they satisfy the highest utility *as much as possible* while *closely* approximating user requirement. The fuzziness in objectives of finding an optimal alternative lies in these italicise words (as much as possible and closely). Thus, both evaluation functions, i.e. the SAW and eEUD functions (cf. Figure 2.14), are transformed into a fuzzy goal and constraints based on the fuzzy decision making Bellman *et al.*'s symmetric model (Bellman and Zadeh, 1970; Zimmermann, 2010). By representing the fuzzy goals and constraints using membership functions to represent the fuzzy goal and fuzzy constraints, the problem of finding an optimal alternative can then be translated into a linear programming model. A maximising decision among the fuzzy decision set can be achieved by solving the linear programming.

For example, let the fuzzy Goal $\tilde{G}$ and constraint $\tilde{C}$ be given as:

- Goal $\tilde{G}$: The performance score alternative should be *in the vicinity of* the ideal solution with respect to QoS preferences.
- Constraint $\tilde{C}$: The QoS values of the alternative should be *very close to* the user's aspiration with respect to QoS preferences.

Suppose, each alternative is evaluated by a SAW function described as $A_i = \sum w_j x_{ij}$, where $A_i$is the performance score of the i^{th}alternative, $w_j$is the priority weight of the j^{th} criterion as expressed by user, and x_{ij} is the QoS value of the i^{th} alternative with respect to the j^{th} criterion; φ is defined as the vector of performance scores for all alternatives given as $\varphi_i = \{A_1, A_2, \dots, A_n\}, i = 1, 2, \dots\ n$; *n* is the number of alternatives.

The goal would be represented by a bell-shaped membership function corresponding to:

$$\mu_{\tilde{G}}(\varphi_i) = \frac{1}{(1 + (\varphi_i - \rho)^4)} \tag{2.28}$$

Where, φ_i is the performance score of the i^{th} alternative, and ρ, is the performance score of the ideal alternative. The ideal alternative is the alternative with the best score for each QoS value.

Likewise, given that similarity function computes the similarity between the i^{th} alternative and the user's aspiration with respect to QoS values, based on the mapping $\boldsymbol{eEUD_i(X, s_i)}: \boldsymbol{\theta} \rightarrow [\mathbf{0}, \mathbf{1}]$, where X is a user's QoS aspiration vector, and s_i correspond to QoS description vector of a service $s_i \in S$; $\boldsymbol{0}$ indicates absolute dissimilarity and $\boldsymbol{1}$ correspond to absolute similarity; $\boldsymbol{\theta}$ is defined as a vector variable of all similarity values of user's requirement to alternatives: $\theta_i = \{eEUD(X, s_1), eEUD(X, s_2), \dots, eEUD(X, s_n)\}, i = 1, 2, \dots n$; where n corresponds to the number of services available in service directory S.

The membership function of the constraints is also bell-shaped expressed as:

$$\mu_{\tilde{C}}(\theta_i) = \left(\frac{1}{(1 + (\theta_i - 1)^2)}\right)^2 \qquad \textbf{(2.29)}$$

The elements of the fuzzy set describe by membership function $\mu_{\tilde{C}}(\theta_i)$ will have a degree of membership corresponding in extent to which θ_i is close to the real value one (1).

Therefore, the membership function of the fuzzy decision sets $\widetilde{D}$ will then be:

$$\mu_{\widetilde{D}}(\varphi_i, \theta_i) = \mu_{\tilde{G}}(\varphi_i) \wedge \mu_{\tilde{C}}(\theta_i) \qquad \textbf{(2.30)}$$

Such that:

$$\mu_{\widetilde{D}}(\varphi_i, \theta_i) = \min\ (\mu_{\tilde{G}}(\varphi_i), \mu_{\tilde{C}}(\theta_i)) \qquad \textbf{(2.31)}$$

The highest degree of the membership in $\widetilde{D}$ is given by:

$$\arg\max_{\varphi,\theta}\ (\min\ (\mu_{\tilde{G}}(\varphi_i), \mu_{\tilde{C}}(\theta_i)) \qquad \textbf{(2.32)}$$

Based on this, the equivalent in a linear programming model is:

Maximize min $(\mu_{\tilde{G}}(\varphi_i), \mu_{\tilde{C}}(\theta_i))$

Subject to:

$$\mu_{\tilde{G}}(\varphi_i) = \frac{1}{(1 + (\varphi_i - \rho)^4)} \tag{2.33}$$

$$\mu_{\tilde{C}}(\theta_i) = \left(\frac{1}{1 + (\theta_i - 1)^2}\right)^2 \tag{2.34}$$

Having formulated the optimisation model and solution approach, the problem can be solved using optimisation algorithms such as genetic algorithm (e.g. NSGAII), or swarm intelligence algorithms (e.g. Particle Swarm Optimisation [PSO] algorithm). The results obtained from the optimisation process are optimal QoS values that best approximates the user's QoS requirements with respect to the spread of QoS attributes of all service alternatives available in the service directory. The final step of evaluating the services alternative is the use of a distance-based function to rank all alternatives, according to their similarity with the optimal QoS values obtained. The ranked results are then presented to the user to make service selection decision.

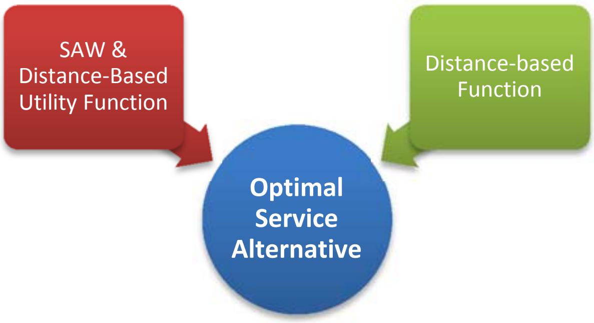


Figure 2.14: Fuzzy Multi-function Service Evaluation Model

2.4.6 Interactive GUI and Information Visualization for Ranking Results

The growing trend for personalised products and services in online shopping context requires that usability and user experience be given top priority if the vision of cloud service e-marketplace is to be realised (Riemer and Totz, 2003; Schubert and Ginsburg, 2000; Liang and Lai, 2002). Usability is a measure of how easy to use, effective a system is (i.e. did the user achieve the goal?) and efficient a system is (i.e. how long it took the user to achieve the goal?); while user experience defines the feelings of the user in utilizing the system (e.g. is the interaction satisfying, enjoyable, engaging) (De Oliveira *et al.*, 2012; Travis, 2008) The goal of pursuing usability and user experience is in the context of this research is to optimize user satisfaction (Bevan, 2009). Noteworthy is that the Graphical User Interface (GUI) is the visual medium through which the user interacts and engages the e-marketplace, and it plays a very prominent role in determining the usability and user experience in the e-marketplace environment (Van Schaik and Ling, 2008; Wong *et al.*, 2014).

Graphical User Interface is a subset of Human-Computer Interaction (HCI); HCI studies the planning and design of how humans and computers work together to effectively meet the needs of a human (Galitz, 2007). The GUI underscores input and output features; input is how a user expresses business and technical requests or requirements, whereas the output presents the result of those requests to the user (Galitz, 2007). The GUI obscures all the technical and computational processes underlying the e-marketplace operations while being a functional, enjoyable and satisfying means to explore the QoS ranking of cloud services towards making a cloud service selection. Indeed, an arbitrarily complex GUI design increases the cognitive difficulty in performing specific user-centric tasks (Galitz, 2007), consequence for which could lead to a selection of a poor or sub-optimal option or abandonment of the process altogether. Both outcomes have implications on the profitability and the perpetuity of the e-marketplace (Galitz, 2007; Liu *et al.*, 2012; Bonastre and Granollers, 2014).

I. Graphical User Interface for Cloud Service e-marketplace

In the context of cloud service e-marketplace, the large number of functionally equivalent cloud services sorted according to QoS ranking with respect to user requirements emphasises the need for an effective decision-making aid to support the exploration of cloud services. Similarly, in the regular e-commerce domain, the rate of shopping cart abandonment, dissatisfaction and frustrations experienced in many e-commerce sites due to the complexity involved in the search for commodities raises the need for user experience in online shopping (Liu *et al.*, 2012; Liang and Lai, 2002; Bonastre and Granollers, 2014). Just like one of the laws of e-commerce states that if users cannot find it, they cannot buy it either; the GUI design questions that must be answered in a cloud service e-marketplace includes:

i. How conveniently can the user express QoS requirements?

ii. How quickly can optimal results be generated?

iii. Are the results presented in the best way possible for users to understand and draw insights from?

Since the main medium of engagement in the e-marketplace environment is visual, answering these questions facilitates a GUI design that ensures the user can conveniently express QoS-based requests, for which optimal services match are found within the shortest time possible and the information is intuitively presented in a manner that is easy to understand and facilitates quality decision-making (Gui *et al.*, 2014; Galitz, 2007). Although the user experience covers all aspects of e-marketplace operations (Kuniavsky, 2003) – such as billing, payment, deploying of a service instance, and SLA monitoring, its focus in this study is how users use the GUI to request for services based on QoS requirements and to effectively explore a set of likely alternatives. An emergent perspective would be a GUI framework delineated into two, based on the support for the tasks users perform on the e-marketplace in their quest to select an optimal service alternative. These include interface design that: i) allow users to express QoS requirements and, ii) allows the visualisation and effective exploration of ranking cloud services (see Figure 2.15).

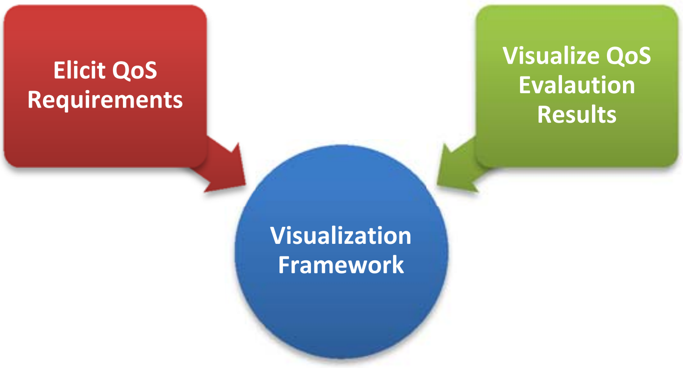


Figure 2.15: Graphical User Interface Framework

Preferable are GUI designs that are intuitive and capture user QoS requirements in a manner that is natural to the human judgement or perception. This is because the user's perception of the interface affects their attitude towards what comes out from it, and ultimately affects user satisfaction (Kuniavsky, 2003; Sundar *et al.*, 2014). Applying visualisation would in a way enable low cognitive demand in exploration by giving the user a graphical overview of the rankings and in order to understand the relationship of services to each other based on QoS attributes ranges. In addition, by interacting with this visualisation, users can then perform a trade-off analysis by filtering services according to the desired QoS factors. Such graphical depiction is more convenient and reduces cognitive overload compared to a mere textual listing of the ranking results (Almulla *et al.*, 2012; Beets and Wesson, 2010; Pleuss *et al.*, 2011; Spence, 2014; Mamoon *et al.*, 2013).

Similarly, the main drawbacks with textual representation in the domain of web service discovery were highlighted as follows: ineffective search facility and poor presentation of the web services, as textual lists, do not effectively support the user in finding suitable web services (Beets and Wesson, 2011). Earlier studies on the effect of textual/tabular representations of data as against graphical representation in decision-making contexts revealed that graphical representations performed significantly better (Coll *et al.*, 1994; Jarvenpaa, 1989; Jarvenpaa and Dickson, 1988): thus providing a preliminary basis to support the use of graphical representation to improve the user experience in cloud service selection.

II. Information Visualisation: An Overview

It has been proven that humans possess the ability to recognise the spatial arrangements of elements in a picture and decipher relationships among elements quickly and easily (Shneiderman, 1994). Such abilities enable humans to derive greater insight and comprehension of the content of a picture faster than mere text. This process leads to a more informed decision-making by capitalising on the well-developed human visual processing capability (Shneiderman, 1994). Similar to web service discovery, the application of information visualisation technique for aiding cloud service selection would improve cloud service exploration and insight into the rationale behind the ranking of cloud services with respect to user's QoS requirements (Beets and Wesson, 2011).

Information visualisation is concerned with the use of visualisation methods in assisting users to make more sense of and use large volume and complex dataset as they analyse and explore the data with a slight effort from users (Spence, 2014; Almulla *et al.*, 2012; Khan and Khan, 2011). The overarching goal of information visualisation is to communicate information in an interactively graphical or spatial manner to aid user understandability (Draper *et al.*, 2009; Beets and Wesson, 2011; Almulla *et al.*, 2012; Khan and Khan, 2011). Integrating information visualization as part of a cloud service selection framework is more beneficial compared to traditional textual listings in that users can understand relationships among data elements as they can learn more from the visualization in lesser time; users can, therefore, access to new understanding of, or knowledge about, the QoS ranking results generated by the service alternative evaluation module (Mamoon *et al.*, 2013; Beets and Wesson, 2011; Chittaro, 2006).

III. Information Visualisation: Reference Model

Several frameworks and processes to enable the design of an effective IV have been proposed in the literature (Chittaro, 2006; Card *et al.*, 1999; Adnan *et al.*, 2008; Spence, 2014; Khan and Khan, 2011); these taxonomies of information visualization processes consist of several steps and activities for turning dataset into visualizations, and can be categorized into four main modules (see Figure 2.16), which includes: Dataset, Representation (or Mapping), Organization (or presentation), and Interaction.

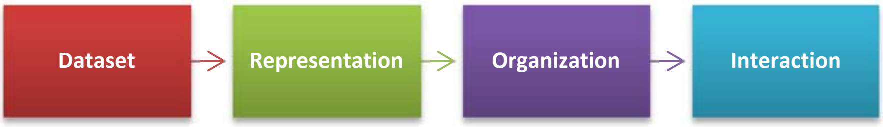


Figure 2.16: Information Visualization Reference Model
Source: Spence (2014)

a) Dataset

According to (Shneiderman, 1996), there are seven data types that are identified in the context of Information visualisation, they include:

i. **1-Dimensional datatype**-also referred to as linear data types which are organised by a single feature e.g. textual documents, alphabetical listing of items;

ii. **2-Dimensional datatype**- also referred to as planer or map data e.g. floor plans, geographic maps etc.;

iii. **3-Dimensional datatype**- representing most real-world objects;

iv. **Temporal datatype**- includes data that have timelines denoting start and finish time, e.g. project management timeline data;

v. **Multidimensional data**- correspond to most relational and/or statistical data which are usually manipulated such that items with n-attributes become points in a n-dimensional plane e.g. a list of cloud services and their multiple QoS dimensions;

vi. **Tree data type**- refers to hierarchies comprising a collection of items in which an item is linked to one parent, with exception of the root e.g. computer directories;

vii. **Network data type**- which is a generalisation of tree data type where the items or objects is linked to any number of other items.

The multi-dimensional dataset comprising a collection of cloud services in a ranked order can be presented in a table format (see Table 2.14), such that each column corresponds to service QoS attributes while each row refers to each service in the list. However, tabular representations are limited in expressing the relationships among the rankings; depending on the number of services in the ranked list and many QoS attributes to consider. To explore each of the services one after the other is cumbersome and does not readily satisfy the user's quest to understand how each service in the ranked list differs from the other.

Table 2.14: A tabular representation of cloud services with QoS properties

	Availability (%)	**Response Time(ms)**	**Reliability (%)**	**Cost($)**
Cloud service 1	78.5	450	79	205.70
Cloud service 2	99.9	320.23	90	350.45
Cloud service 3	87.92	5400	83	190.44
Cloud service 4	93.76	237.88	90	301.50
Cloud service 5	50.5	403.66	92	211.22

b) Representation

Representation (or visual mapping) refers to how to transform symbolic representation characteristic of the objects in a dataset and their interrelationship, into a graphical form using visual encoding mechanisms. This mechanism includes object's size, shape, colour, orientation (or position), and dimensionality (text, 2D, or 3D) (Chittaro, 2006; Adnan *et al.*, 2008; Moere and Purchase, 2011; Bertin, 1983). The representation must take into consideration data type, data dimensions, and the user's perceptual and cognitive abilities (Spence, 2014). The dimension of the dataset refers to the number of attributes that characterise the dataset. The way users perceive the value of data elements is rooted in how those data elements are visually encoded using size, orientation, shape, texture, and colour (Shneiderman, 1994; Bertin, 1983; Spence, 2014). The application of these encoding mechanisms (e.g. size, shape etc.), supports tasks associated with information visualisation with varying degree of suitability (Bertin, 1983). Some of the cognitive and perceptual factors to be considered include the user's perception of values and if the representation exhibits object or attribute visibility (Spence, 2014).

The concept of object and attribute visibility was first introduced by (Teoh and Ma, 2005). Teoh and Ma (2005) noted that one challenge with multi-dimensional (multi-attribute) visualisation is the multiplicity of objects and dimensions, and introduced the concept of coherence and correlation as it pertains to objects and their dimensions.

A representation is said to exhibit object coherence (or visibility) when the object is encoded as a single and compact graphical entity (e.g. a point or bubble) and the user can see all the attributes of the objects all at once. The converse of a representation possessing object coherence is when the object is represented by multiple separate visual entities (e.g. several points). Meanwhile, dimension coherence (or attribute visibility) refers to a representation in which the attribute values of the objects are distributed across each dimension, such that users can quickly see the relationships among the values of the attributes for each object (Teoh and Ma, 2005).

On the other hand, a representation satisfies object correlation when the user can immediately see the similarities among objects considering all the values of their attributes. Dimension correlation refers to a representation that allows the user to easily note the relationships among the dimensions of all objects in the dataset. In this study, the inquiry to object coherence and are concerned with a mechanism to represent cloud services in the ranked list as single coherent entities so as to enable the exploration of the relationship among alternatives. Table 2.15 contains an overview of some representations suitable for the data types as espoused by (Shneiderman, 1996).

a) Organization

Organisation (or Presentation) refers to the interface schemes that define the manner in which these representations are laid out on a screen to enable user's exploration and interaction (Adnan *et al.*, 2008; Spence, 2014; Burigat and Chittaro, 2013; Cockburn, 2009; Khan and Khan, 2011). Generally, the interface schemes facilitate sense-making, as it impacts on user's interpretation and perception of the information presented (Adnan *et al.*, 2008).

Table 2.15: Datatypes and supporting data representation

#	Datatype	Representation types	
1	1-Dimensional	■ Textual Lists	
2	2-Dimensional	■ Choropleth ■ Self-organizing Maps	■ Dot distribution map ■ Proportional symbol map ■ Cartogram
3	3-Dimensional	■ Surface and volume rendering	■ 3D Computer models
4	Temporal	■ Timeline ■ Time series ■ Gantt Chart	■ Arc diagram ■ Rose diagram (or Polar Area)
5	Multi-Dimensional	■ Tables ■ Pie chart ■ Histogram ■ Tag cloud ■ Unordered bubble chart	■ Bubble chart ■ Line chart ■ Heat map ■ Radar/spider chart ■ Parallel coordinates plot ■ Bar chart
6	Tree	■ General tree visualisation ■ Dendrogram ■ Radial tree	■ Hyperbolic tree ■ Treemap ■ Sunburst
7	Network	■ Dependency Graph/Circular hierarchy ■ Node-link diagram	■ Matrix ■ Tube map

Source: Zoss (2015)

The schemes adopted impacts on the effectiveness and ease of viewing and exploration of content in order to make more informed decisions. Information visualisation techniques, like those mentioned in Table 2.15, organises information on the screen with respect to how objects from the dataset are positioned and can be viewed on the screen per time, and the layout of the general overview of objects (Adnan *et al.*, 2008). The layout of the information on the screen affects the type of tasks that can be performed by users, as it determines the interactions users can have with the information displayed (Spence, 2014). Based on the layout of information on the display, there are three main schemes for presenting/organising information, they include Zooming, Overview+Detail and Focus+Context (Burigat and Chittaro, 2013; Cockburn, 2009; Spence, 2014; Adnan *et al.*, 2008).

i. **Zooming-** Zooming refers to the interface's ability to provide a broader overview or more detailed view by increasing or decreasing the levels of details the user can view per time (Spence, 2014; Khan and Khan, 2011). Zooming can either be geometric zoom or semantic zoom (Herman *et al.*, 2000; Spence, 2014). Geometric zoom happens when the display scales from a broader view to a fraction of the same view with only change in size, limiting what is viewable on that area of the display (e.g. zooming in and out of a geographic map). On the other hand, semantic zoom does not only change the size of the information displayed, but also its other visual properties such as information content, colours, shape, and texture (Spence, 2014; Herman *et al.*, 2000; Nestor *et al.*, 2007).

ii. **Overview + Detail-** Some studies show that user satisfaction and efficiency are enhanced when users can view and explore both contextual and detailed information at the same time ((Beard and Walker, 1990; North and Shneiderman, 2000; Hornbæk, 2001; Hornbɥk and Plaisant, 2002). The Overview+Detail (O+D) interface scheme allows both the context and detailed views to be displayed simultaneously in a separate spatial location on the screen (Adnan *et al.*, 2008; Cockburn, 2009; Hornbɥk and Plaisant., 2002). The physical separation of both views, enable the possibility of users interacting with both views separately, and actions in one view, trigger a response in the other (Cockburn, 2009). Although the O+D scheme lays a short-term memory load on users and more time is expended in visual search, some benefits of the O+D schemes include efficient navigation, with alternative views (detailed and overview) giving more control to the user. Also, users cannot 'get lost' with access to the broader view of the information space which provides task-relevant information (Beard and Walker, 1990; Plaisant *et al.*, 1994; Shneiderman, 1987; Hornbɥk and Plaisant., 2002).

iii. **Focus + Context (F+C)-** Zooming schemes provide on-demand focused and contextual information separated temporally in time, but O+D schemes present both views in co-existing in the same time in distinct spaces on the screen. F+C schemes seamlessly combines focus and context information in the same space, and focus is amplified by distorting the information space, while ensuring continuity of the focus region of interest within its surrounding context by maintaining relevant aspect of the context (Burigat and Chittaro, 2013; Spence, 2014, p. 131; Cockburn, 2009; Khan and Khan, 2011). F+C overcomes the short term memory load demand on a user by presenting all information in single coherent view, and users can easily understand and manipulate the information displayed. However, the drawbacks of distortion-oriented views like fisheye view are the misinterpretation of the underlying data (Cockburn, 2009).

b) Interactivity

Interactivity refers to the mechanisms available for making sense of the information space by navigating, exploring, organising or rearranging the information space (Adnan *et al.*, 2008; Khan and Khan, 2011). Effective exploration of the information space is determined by the method of interaction employed, the type of tasks those methods can support and the rate of response to the interaction (Adnan *et al.*, 2008; Spence, 2014; Walker *et al.*, 2016), also different interactions performs differently and are best suited for different tasks (Nestor *et al.*, 2007). The way in which users interact with the interface can take different forms, such as use of menus (drop-down, pop-up), scrolling, flipping (replacing one discreet view with the next), and direct manipulation by mouse over, single click, double click directly on the visual elements in order to initiate a response (Adnan *et al.*, 2008; Sundar *et al.*, 2014; Khan and Khan, 2011). Shneiderman (1996) has proposed seven tasks that the interaction used in information visualisation should support. The seven tasks include:

1. **Overview**: Gain an overview of the entire collection.
2. **Zoom**: Zoom in on items of interest
3. **Filter**: Filter out uninteresting items
4. **Details-on-demand**: Select an item or group and get details when needed.
5. **Relate** View relationships among items.
6. **History**: Keep a history of actions to support undo, replay, and progressive refinement.
7. **Extract**: Allow extraction of sub-collections and of the query parameters.

IV. Information Visualisation for Cloud Service Selection

The e-marketplace interface should be designed with usability and user experience intended, such that users can easily express QoS requirements and find optimal service(s) within the shortest time (Chua *et al.*, 2007). Apart from the functionality of the e-marketplace, the 'look and feel' of a graphical user interface, both for eliciting requirements and exploring results should use visual elements such as colours, shapes, layout, and typefaces, as well as support some dynamic behaviours (Chua *et al.*, 2007). In addition, the result of the ranking process is usually presented in textual formats from which the user is expected to make a selection. This approach usually demands more cognitive effort as users are expected to make sense of the results unaided. Information visualisation has been applied in the context of web service discovery and selection (Beets and Wesson, 2010; Beets and Wesson, 2011; Almulla *et al.*, 2012), in which authors reported that textual list of web services can result in time-consuming and ineffective web service discovery. The overall aim of pursuing a visualisation approach is to assist users to effectively identify and explore the expected results with respect to their QoS requirements, at the same time providing the opportunity to discover unexpected items as they gain more insight into the ranking results. An effective visualisation mechanism would allow the user to accomplish these tasks they wish to undertake with the ranking results (Walker *et al.*, 2016).

a) User Interface to Elicit User QoS Requirements

The goal of selecting a cloud service(s) based on QoS ranking produced by evaluating alternatives with respect to user's interest in and values for specific QoS attributes begins with properly articulating those requirements. Fuzzy-intuitive interfaces allow users to express their QoS requirements in a manner that capture the subjectivity inherent in those requests. A cloud service selection framework should employ fuzzy set theory to model users' preferences and aspirations for each QoS attributes and the appropriate GUI element to elicit these inputs is required.

i- Eliciting QoS Preferences using Graphical Fuzzy-AHP

There are three main implementation styles for eliciting users' QoS preference using the pairwise comparison of attributes; they include graphical, numeric and verbal representations (Millet, 1997; Forman and Gass, 2001). Numeric implementations require that users indicate preferences as a numeric ratio between two alternatives (e.g. *Security* has ¼ times more priority than *Availability*), whereas graphical approaches involve the adjustment of bar diagrams or sliders to acquire user's preferences one pair per time (Millet, 1997). Although most decision analysis systems are usually focused on the accuracy of the results, the user satisfaction of the comparison techniques and the process is also of vital consideration (Millet, 1997; Ge *et al.*, 2010).

Millet (1997) reveals that the accuracy and ease of use factors of these approaches differ with numerical and verbal approaches topping the list for accuracy, while graphical approaches topped the list for ease of use and faster completion time and optimising both accuracy and ease of use (Millet, 1997). Similar to (Cakir and Canbolat, 2008), an emergent perspective is a QoS requirements elicitation technique that embeds fuzzy-AHP into a web UI widget to improve the user experience in expressing QoS requirements while maintaining high accuracy of the results.

ii- Eliciting QoS Aspiration using Interactive Interface

The user need not express exact values for interesting QoS attributes. Applying fuzzy linguistic variable and membership functions allow the user the flexibility of expressing values for QoS attributes in imprecise terms natural to human judgment. Rather than entering some of these values as text, an intuitive GUI design should allow users to perform this task easily. For example, an interactive interface comprises of the use of drop-down menus, check boxes and text boxes. Typically, a user searching for a cloud service could articulate these requirements using fuzzy expressions as follows, '*Availability should be very high'* or '*cost around $300/month'*.

b) Information Visualisation to Display Ranking Results

The user's QoS request forms the input into the fuzzy-based multi-function utility evaluation and ranking module, which produces a top-k list of cloud services ranked according to their suitability to user's requests. The QoS ranking result forms the input dataset into the information visualisation module, represented in a graphical form for users to gain insight into the ranking results to obtain more insight into the information space, explore the results in details and compare items on the list. Discussed next are the design requirements and considerations for information visualisation with respect to representation, presentation and the interaction supported by the IV techniques.

i. **Dataset-** the QoS-based ranking result is a multi-dimensional data type that contains the values for all of the relevant QoS attributes (see Table 2.14).

ii. **Representation-** the items in the list can be visually encoded using a combination of mechanisms, comprising size, colour, and position (or orientation), into single coherent entities that exhibits object coherence and correlation; such that by sighting a cloud service representation, the user can easily make sense of its attributes compared to other services on the list. A potential information visualisation technique that suffices for the multi-dimensional data considered is the bubble graph. The bubble graph encodes each cloud service in the ranked list as a 'bubble' and explicitly shows the QoS relationships of the top ranked cloud services as well as the underlying structure of the QoS information space using *colours*, *size* and *position* (or orientation). The bubble graph can be used to visualise up to four QoS dimensions simultaneously, each dimension represented by size, colour and position (x and y coordinates), see Figure 2.17.

iii. **Presentation-** The information visualisation to support cloud service exploration and selection must be such that it lays out both the broader and more detail views on the display screen. F+C presentation style is reported inappropriate for decision-making environments because of its distorted view, as it may lead to wrong interpretations (Yang *et al.*, 2003). Other studies reported higher user satisfaction and faster task completion time of O+D styles over zoom-based presentation styles (Adnan *et al.*, 2008; Ghosh and Shneiderman, 1999). For this, O+D is considered, since the volume of information displayed must be such that does not add to the cognitive load on users; impacting negatively on user satisfaction (Pirolli *et al.*, 2003; Adnan *et al.*, 2008).

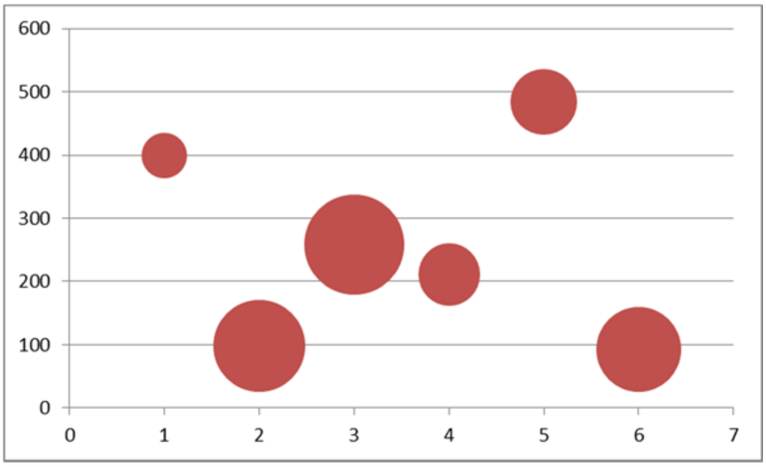


Figure 2.17: Example of Bubble Graph

iv. **Interaction-** Interactivity refers to the ability of users to engage the visualisation of ranked results in real time, making changes to visualisation parameters and viewing immediate responses in the visualisation (Khan and Khan, 2011). The information visualisation should support various interaction methods, including direct manipulation by hovering, clicking and the use of dynamic queries for advanced filtering task. This research considered interaction methods that allow users to explore the representation incrementally and dynamically using the sliders (Spence, 2014). Based on Shneiderman's Task by Data Type Taxonomy (Shneiderman, 1996), two interaction tasks were identified: They include: to gain a general overview of the ranking results (**overview**); view details of a particular selection as desired, by either a mouse click or hovering (**Details-on-demand**).

2.5 CHAPTER SUMMARY

One major challenge of operationalizing a cloud service e-marketplace is *service choice overload*; describing the complexity of decision making because of the availability of too many service alternatives which often times lead to unsatisfactory choice. Service choice overload can be minimised by using low cognitive demand decision support mechanisms for eliciting user requirements. This must be done in a way that the techniques:

i. Provides an underlying organisation combination model for ecosystem services.

ii. Combines both fuzzy QoS preference and aspiration information in the evaluation process.

iii. Employs intuitive user interface to elicit fuzzy user QoS requirements.

iv. Includes means to visualise ranking results in a way that reduces service choice overload.

Although cloud service selection techniques have been proposed in the literature, a state-of-the-art and a comparative analysis of these techniques were carried out to identify the gaps in existing approaches and to propose key requirements for a framework that suits the cloud service e-marketplace. Based on the key requirements, the emergent perspectives provided the basis to formulate a set of design considerations to guide the formulation of the cloud service selection framework.

CHAPTER THREE

METHODOLOGY

3.1 INTRODUCTION

This chapter presents the details of the methodology adopted to achieve the aim and objectives of this study. The methodology describes the proposed framework as a decision-making framework for cloud service selection in e-marketplace context. More specifically, this chapter contains insights into its strategy and underlining assumptions, process model, conceptual architectural framework, and a description of its sub-components. Furthermore, the modalities for demonstrating the plausibility of the proposed framework are presented, and this chapter concludes with a summary of its content and discussion.

3.2 PROBLEM DESCRIPTION: CLOUD SERVICE RANKING AND SELECTION

So far in this thesis, the emerging cloud service e-marketplace has been defined as a one-stop shop for cloud services, aimed at enabling the commoditization of vertical or horizontal cloud service offerings as single or composite services from a variety of providers (Menychtas *et al.*, 2014); combining services in special ways not previously thought of, enabled by the concept of a cloud ecosystem (Barros and Dumas, 2006). Functionally equivalent service offerings are differentiated by their QoS factors (e.g. availability, response time, reliability, etc.), and this information is contained in the e-marketplace service directory or catalogue (Menychtas *et al.*, 2014). It was also mentioned that services are showcased through an e-marketplace interface, on which users interact with the e-marketplace to find suitable services that satisfy user-specific QoS requirements, towards fulfilling their business objectives.

Decision making involves the selection from a collection of items based on specific interest in, and value for, the multiple attributes characterising those items. Selection is further complicated by the unavailability of properly articulated ideal points and order of preferences with respect to the underlying attributes, which must be considered in evaluating each alternative. Besides, the presentation of the result of the evaluation is another point where user satisfaction is necessary. The underlying assumptions in selection problems can be summarised as follows:

i. There exist collections of items, and the items have multiple attributes and can be represented using a data model.

ii. Users (as decision makers or information seekers) possess preferences (i.e. the order of importance of all QoS attributes) and aspiration (i.e. actual values for each QoS attribute) for the desired alternative.

iii. The selection task is to find all items that best approximates (and to what degree) the users' requirements.

Typically, cloud service selection is concerned with the performance evaluation of the set of m offerings based on user's priorities for each QoS attributes and desired QoS attribute values for the set of n QoS criteria, so that users can then choose the service(s) with the most optimal performance. On the basis of this, this study postulates improved quality of user experience during user interaction with the e-marketplace front-end by reducing the complexities of decision making through handling the subjectivity and vagueness often associated with expressing QoS preferences and aspirations. Due to the multiplicity of QoS dimensions and a large number of alternatives, cloud service selection is considered as an NP-hard problem (Jula *et al.*, 2014). Next, formal definitions describing the cloud service selection problem are presented.

3.2.1 A Set of Atomic Cloud Services

Definition 3.1 (Set of atomic services): Let $S = \{S_1, S_2, S_3 \ldots S_m\}$ be a set of m atomic services that are part of the cloud ecosystem. A combination of these atomic services creates a composite service that can satisfy complex user requirements.

3.2.2 Quality of Service (QoS) Attributes

Definition 3.2 (Set of QoS attributes): Let Q be a vector ($1 \times n$ matrix) that represents a set of QoS attributes denoted by $Q = (q_1, q_2, q_3 \ldots q_n)$, as of n components describing the QoS attributes of a service $s_i \epsilon S$.

3.2.3 The e-marketplace Cloud Services Directory

Definition 3.3 (Cloud Ecosystem Feature Model): A cloud ecosystem feature model is a sextuplet $CEFM = (F, F_O, F_M, F_{IOR}, F_{XOR}, F_c)$ consisting of features F and feature relationships in terms of parent-child and integrity constraints. F_O represents a set of parent and optional child feature pairs; F_M is a set of parent and mandatory child feature pairs; F_{IOR} and F_{XOR} are sets of pairs of child feature and their common parent feature grouped respectively into 'or' and 'alternative' groups; F_c is a set of constraints-required and excludes. A valid composition includes a set of features F combined, according to features relationships and integrity constraints F_c.

Definition 3.4 (QoS Aggregation): Let a service $s \in S$ be a valid combination composed of $a\ t\ number$ of distinct services $Z_{(1\ to\ t)}$ with n QoS attributes and acts sequentially. Let $q_i(Z_k)$ be the value of the i^{th} QoS attribute for the k^{th} distinct service. Such that the aggregated value i^{th} QoS attributes for all distinct services composed in s is given as:

$$q_i(s) = (q_i(Z_1) \bowtie q_i(Z_2) \bowtie \cdots \bowtie q_i(Z_t)) \tag{3.1}$$

Where $\bowtie$ represents the aggregation operator based on the aggregation function employed with respect to the QoS type and $t > 1$. Meanwhile, the vector Q of QoS values for a valid combination s is given as:

$$Q(s) = (q_1(s), q_2(s) \ldots q_n(s)) \tag{3.2}$$

Definition 3.5 (Services Directory): Let A be $m \times n$ Matrix that contain the QoS information of all valid composite services $s_1 \ldots S_m \in S$ generated based on definitions 3.3 and 3.4, where each element $a_{i,j}$ represents the j^{th} QoS value of the i^{th} service, while $i, j > 2$.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{n2} & \cdots & a_{mn} \end{pmatrix} \tag{3.3}$$

From A, a row vector would describe a service $s_i \in S$ with QoS attributes where each element represents the QoS attribute of composite service s_i.

3.2.4 User QoS requirements

The values of a user's QoS requirements (aspiration) are captured in a vector that corresponds to the number of QoS attributes that describes available e-marketplace services. Users QoS preferences reflect the relative importance of each QoS attribute to others and are denoted using priority weights derived from the pairwise comparison. Similar to Rehman *et al.* (2011), user requirements are defined as:

Definition 3.6 (Fuzzy Pairwise Comparison Judgment): Suppose there are n QoS attributes, and that the user can provide a set $L = \{\tilde{a}_{ij}\}$ of $m \leq n(n-1)/2$ fuzzy pairwise comparison judgments, where $i = 1,2 \ldots, n-1$; $j = 2, 3, \ldots n; j > i$, represented as triangular fuzzy numbers $\tilde{a}_{ij} = (l_{ij}, m_{ij}, u_{ij})$, a crisp priority vector $W = (w_1, w_2, \ldots, w_n)^T$ is derived such that the priority ratios $w_i/w_j$are approximately within the scopes of the initial fuzzy judgments, or $l_{ij} \lessapprox \frac{w_i}{w_j} \lessapprox u_{ij}$; where $\lessapprox$ means 'fuzzy less or equal to'.

Definition 3.7 (Fuzzy QoS Aspiration): Suppose there are n QoS attributes and there are n Goals, $G = (G_1 \ldots G_n)$ and Constraints, $C = (C_1, \ldots, C_n)$ for the QoS attributes. Then the resultant decision D_i is the intersection of each Goal and Constraint, denoted as:

$$D_i = G_i \cap C_i \tag{3.4}$$

Corresponding to:

$$\mu_{D_i}(v_i) = \min\Big(\mu_{G_i}(v_i), \mu_{C_i}(v_i)\Big) \tag{3.5}$$

Where μ_{D_i}, μ_{G_i}, and μ_{C_i} are the membership functions for decision, goal and constraint. However, the A maximizing decision is the point in the set of values at which the membership function of a fuzzy decision attains its maximum. The optimal alternative is found using a maximizing decision D^*, and its membership function corresponding to:

$$\mu_{D_i^*}(v_i) = \arg\{\max \mu_{D_i}(v_i)\} \tag{3.6}$$

Therefore, the QoS aspiration vector $V = (v_1, v_2, \dots, v_n)^T$ is obtained as the values of v_i that has the highest membership grade in the decision fuzzy set D_i.

Definition 3.8 (User QoS Requirement): The user's QoS requirement is a tuple $R = (W, V)$. Where $W = (w_1, w_2, \dots, w_n)$, and each w_i is the importance weight for i^{th} QoS attributes derived from fuzzy pairwise comparison judgment performed by the user; $V = (v_1, v_2, \dots, v_n)$, and v_i corresponds to user's desired value for the i^{th} QoS attribute obtained by fuzzy decision making process.

3.2.5 QoS Evaluation and Ranking

Users are expected to select the service(s) that most approximates their QoS requirements from the available list of alternatives based on the performance evaluation obtained from an evaluation function. First, optimal QoS values are synthesised from user's requirements (preferences and aspiration), and this information becomes the query to retrieve the most optimal set of services relevant to the user's requirements from the service directory. This was achieved by formulating a fuzzy goal and constraint of finding those QoS values that are in the vicinity of the ideal service ($\tilde{G}$), and very close to the user requirements ($\tilde{C}$). Formally, the optimal values are defined as follows.

Definition 3.9 (Optimal QoS Values): Let V^* be the optimal QoS values synthesised from user's requirements with respect to the QoS information of all services $s_i \in S$. The goal of the optimal QoS values is to find the optimal set of QoS values such that:

$$V^* = \arg\max \Psi\,(\tilde{G}, \tilde{C}), \forall\, s_i \in S \tag{3.7}$$

Where Ψ is the fuzzy multi-objective optimisation modelled as fuzzy decision making, that finds the QoS values in the *vicinity of* the service with the best QoS performance, and also *very close to* the user's requirements R.

Definition 3.10 (Optimal Cloud Service Selection): For a given user's requirements R, an optimal cloud service selection is selecting cloud service $s_i \in S$ from ranking all $s_i \in S$ such that:

$$s_i = \max_{s_i \in S} \{eEUD\ (0\ ,\ s_i)\} \tag{3.8}$$

Where $eEUD$ is a nearest neighbour ranking function that ranks all services, $s_i \in S$ according to the optimal QoS values V^*.

3.3 REQUIREMENTS FOR A CLOUD SERVICE SELECTION FRAMEWORK

Addressing some of the open issues in cloud service selection is the first step to uncovering the requirements of an effective cloud service selection technique that is suitable for an e-marketplace context. This section highlights some requirements for a service selection technique suited for an e-marketplace context based on the analysis presented in Chapter two (Section 2.4.1). The six requirements can be summarised as follows:

i. **Requirement 1: Ability to organise and compose cloud ecosystem atomic services -** A cloud e-marketplace is an ecosystem of heterogeneous services from multiple providers. There is a need to explicitly capture the cloud service attributes (functional and non-functional), and the cross-service relationships and constraints that guide the cloud service compositions in a logical and structured manner (Wittern *et al.*, 2012).

ii. **Requirement 2: Ability to elicit both QoS preferences and aspirations -** Most cloud service selection approaches unrealistically assume the user would provide perfectly crisp, precise and exact preference and aspiration information, which is not congruent with human expressions (Esposito *et al*., 2016; Sun *et al.*, 2014; Qu and Buyya, 2014). Requirement 2 is further broken into the ability to capture vagueness when users express QoS preferences and aspiration; the ability to evaluate the interdependence of the user preferences in line with the multiple QoS criteria; and the ability to evaluate services based on both the user's QoS preferences and aspirations.

iii. **Requirement 3: Ability to perform QoS-based evaluation and ranking from a large assortment of service alternatives:** The e-marketplace context requires approaches that can deal efficiently with a large number of alternatives, and considers mixed QoS data, without accruing high computational overhead (Dastjerdi *et al.*, 2015). In addition, such approaches should allow for the optimisation of specific QoS goals and should be scalable in handling multiple users simultaneously.

iv. **Requirement 4: Capture fuzziness with interactive GUI:** Users' engagement with the e-marketplace to select cloud service should be facilitated by intuitive and interactive user interfaces with which users can conveniently express requirements.

v. **Requirement 5: Visualise cloud service ranking results:** Most cloud service selection approaches presents service rankings in textual format, either in a list or tables (Beets and Wesson, 2011). This does not fully describe the implicit trade-off factors inherent in the available options, nor provide transparency into the reasoning behind the rankings, and can increase cognitive load on users (Lurie and Mason, 2007). Search or evaluation result should be innovatively presented in a way that eases understanding and reduces cognitive load (Zhang *et al.*, 2012).

vi. **Requirement 6: Take into cognizance usability and user experience factors:** The evaluations of cloud service selection approaches reported in literature focuses on the performance and accuracy of the approach in ranking services that align with user requirements. Similar to the evaluation of recommender systems, a more holistic evaluation of cloud service selection approaches should include usability and user experience dimensions.

Following the set of requirements listed above, the design agenda of the FOCUSS framework is summarised as follows:

i. Organise and compose cloud ecosystem atomic services and populate the service e-marketplace directory

ii. Elicit user fuzzy QoS preferences and aspiration;

iii. Perform QoS-based ranking and evaluation of cloud service alternatives with respect to user QoS requirements;

iv. Wrap all the underlying functionalities in a tidy graphical user interface.

3.4 OVERVIEW OF THE PROPOSED FRAMEWORK

The Fuzzy-Oriented Cloud Service Selection (FOCUSS) framework is proposed as an efficient integrated visual-rich fuzzy-based decision support that incorporates feature modelling, fuzzy set theory, fuzzy optimisation methodology, widgets and visualisation in its design for cloud service selection in cloud service e-marketplace context. The input into the FOCUSS framework is a set of cloud service alternatives derived using automated reasoning on an ecosystem model, and users QoS preferences and aspiration captured through an interactive fuzzy-based user interface. The output is a QoS ranking of services with respect to users' requirements presented using interactive bubble charts. The FOCUSS framework is proposed as a scalable approach that suffices for a large assortment of services and improves the quality of user experience in a cloud e-marketplace context. Subsequently, the process model and the conceptual architecture are presented in details, as well as, a justification showing how each component of the proposed framework satisfies the set of requirements listed in Section 3.3.

3.4.1 FOCUSS: The Process Model

The process model of the FOCUSS framework is summarised in Figure 3.1.

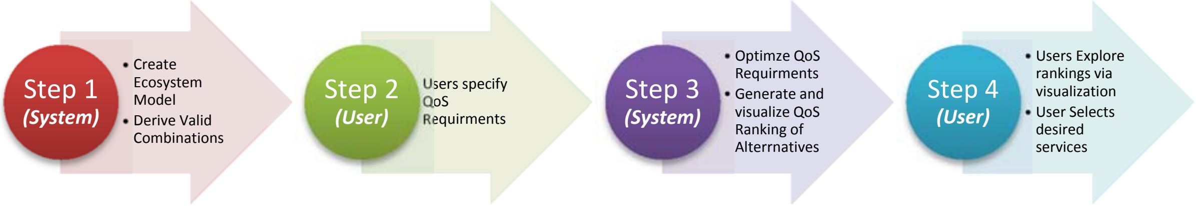

Step 1
(System)
• Create Ecosystem Model
• Derive Valid Combinations
Step 2
(User)
Users specify QoS Requirments
Step 3
(System)
• Optimze QoS Requirments
• Generate and visualize QoS Ranking of Alterrnatives
Step 4
(User)
• Users Explore rankings via visualization
• User Selects desired services

Figure 3.1: High-level Flow chart of FOCUSS Framework

A step-wise description of the workflow of the FOCUSS framework is presented as follows:

i. **Step 1:** Service providers list and register their atomic services in the ecosystem. Based on the ecosystem model, these services are organised, and formally composed in a manner that increases the value proposition of individual atomic services, and these valid combinations are stored in the service directory.

ii. **Step 2**: Users interested in using the services can specify their fuzzy QoS requirements (preferences and aspiration), using a fuzzy-based GUI.

iii. **Step 3**: Based on the specified requirements, the system first resolves the user requirements to obtain an optimal set of QoS values. The optimal QoS values are used to generate an ordered ranking of appropriate services that approximates the user's requirement, based on the QoS ranking mechanism of the proposed framework. This ranking result is visualised in a bubble chart.

iv. **Step 4**: The user can then select appropriate service(s) through exploration of the results with the capabilities provided in the visualisation and exploration GUI.

3.4.2 FOCUSS: The Conceptual Architecture

The FOCUSS framework (see Figure 3.2) consists of four modules, namely: Graphical user interface (GUI), QoS requirements processing, QoS evaluation and ranking, and Cloud ecosystem model and analysis modules. In accordance with the process model, the conceptual architecture (point 0) shows how the atomic services are combined to realise the set of composite services offered in the e-marketplace. Subjective QoS requirements are then provided by the Fuzzy-based widgets at point 1, processed by the QoS Preference Prioritizer and the QoS Aspiration Analyzer at point 2, optimised by the QoS Requirements Optimizer at point 3, while the QoS Ranking Engine ranks services in the directory at point 4. The ranked results are shown to the users via bubble graph visualisation at point 5. Each module is discussed in details subsequently.

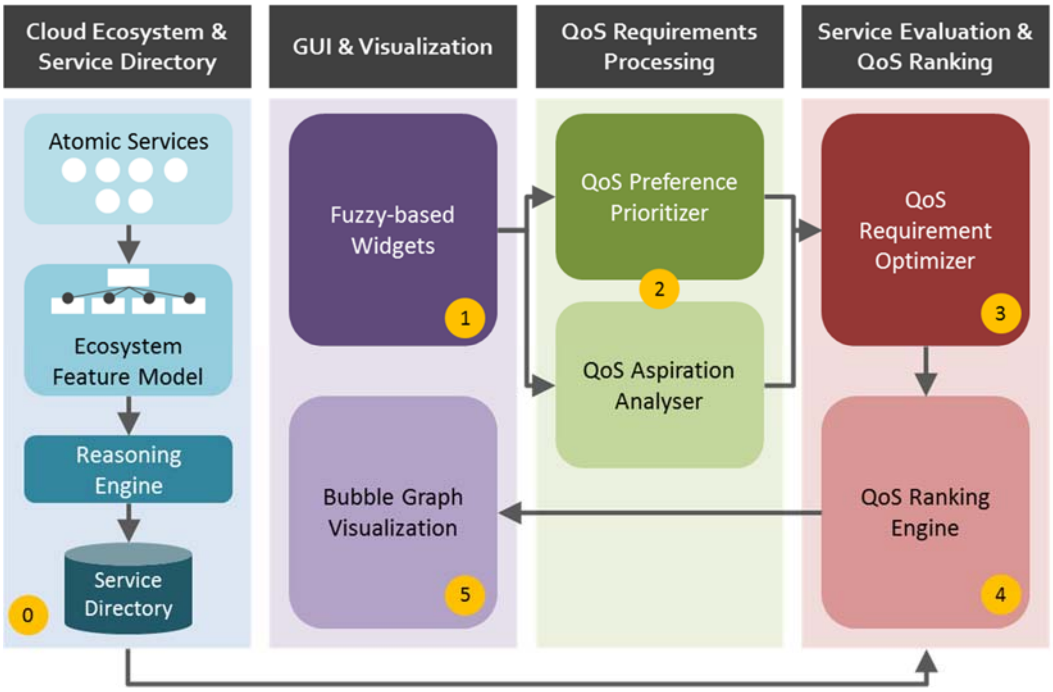

Cloud Ecosystem & Service Directory
GUI & Visualization
QoS Requirements Processing
Service Evaluation & QoS Ranking
Atomic Services
Ecosystem Feature Model
Reasoning Engine
Service Directory
0
Fuzzy-based Widgets
1
Bubble Graph Visualization
5
QoS Preference Prioritizer
2
QoS Aspiration Analyser
QoS Requirement Optimizer
3
QoS Ranking Engine
4

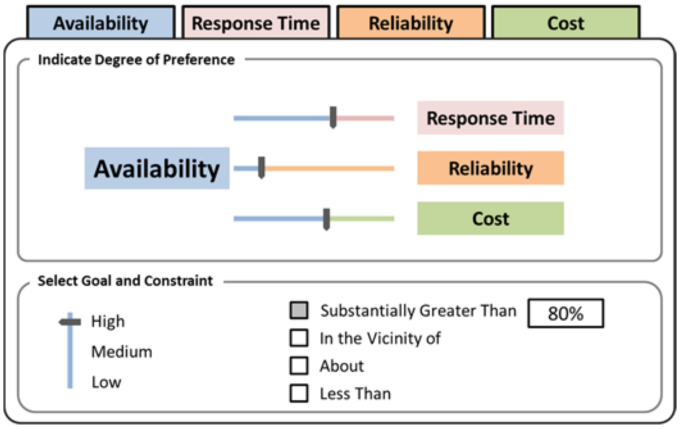
Figure 3.2: Architecture of the FOCUSS Framework

I. Graphical User Interface

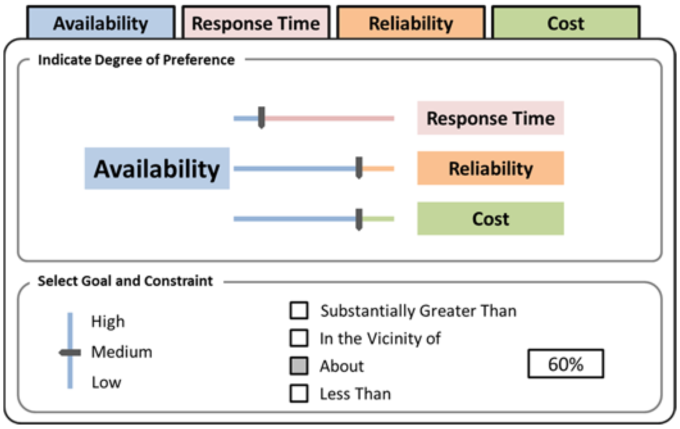
The GUI is the visual medium through which the user interacts and engages the e-marketplace, and it plays a very prominent role in usability and user experience in the e-marketplace environment (Van Schaik and Ling, 2008). The GUI module comprises Fuzzy-based Interactive GUI and Bubble graph visualisation components, which are discussed next.

a) Fuzzy-based Interactive Graphical User Interface

The fuzzy-based interactive graphical user interface consists of drop-down menus, text boxes and slider bars for eliciting users' vague preferences and aspirations under one integrated visual interface using slider bars can also enhance user experience. Users can indicate the level of preference by pairwise comparison for each QoS attribute by adjusting the slider handle left or right. The slider bar has two colour codes that correspond to the QoS attributes, and indicates the amount of preference for a QoS attribute; the lengthier colour means user prefers a QoS attributes more than the other to an extent. The positions of the slider handle are underlined by fuzzy numbers, from the fuzzified Saaty scale, and correspond to the degree of preference indicated during the pairwise comparison by the user. The QoS aspiration level is specified by selecting an option from the drop-down menu indicating linguistic values and a threshold that approximates user's QoS aspirations for a specific QoS attribute. A typical illustration of the fuzzy-enabled GUI for eliciting user's QoS preference and aspirations is shown in Figure 3.3.



(a)



(b)

Figure 3.3: Sketch of UI Design showing Availability QoS Requirements for two Users
(a) User-I expect that Availability value should be high and substantially greater than 80%
(b) User-II specifies that Availability value should be Medium and should be about 60%.

b) Bubble Chart IV Module with dynamic exploration capabilities

The FOCUSS framework incorporates the bubble graph information visualisation technique to intuitively present ranking results in a manner that is easy to understand and facilitates quality decision-making. Each ranked cloud service is represented as a bubble (**shape**), using a variety of **colours, sizes** and **x-y coordinates** to show services in the QoS information space (cf. Figure 2.17). These dimensions (colours, size and x-coordinates and y-coordinates) represents up to four QoS dimensions simultaneously. Based on the SMI QoS model for cloud services (CSMIC, 2014), four QoS attributes have been considered in this study, they include, *Cost*, *Reliability*, *Response time* and *Availability*, which also have been the basis for QoS consideration in similar approaches, for example (He *et al.*, 2012; Karim, 2013; Zeng *et al.*, 2009; Ludwig, 2012). Dynamic exploration enabled by clicking to access details of each option is the form of direct interaction that allows the users to view the details of each option almost immediately (Shneiderman, 1994; Nestor *et al.*, 2007).

II. QoS Requirements Processing Module

The user's QoS requirements elicited via the GUI are processed in the QoS Requirements Processing (QRP) module, in order to identify suitable cloud services that match those requirements. The QRP module comprises of the QoS Preference Prioritizer (QPP) and the QoS Aspiration Analyzer (QAA). An accurate elicitation of users' QoS requirements involves the interpretation of fuzzy expressions associated with evaluating service alternatives (Qu and Buyya, 2014; Esposito *et al.*, 2016; Sun *et al.*, 2014). The ability to express vague preferences or aspiration using natural linguistic terminologies enables easier and quicker expression of users' QoS requirements (Esposito *et al.*, 2016; Qu and Buyya, 2014; Gatzioura *et al.*, 2012). To the user, this means that requirements need not be stated in exact or precise terms of the service attributes (Akolkar *et al.*, 2012), and are therefore allowed some flexibility. The QPP and QAA modules are described in more details next.

a) QoS Preference Prioritizer

To prioritise user's QoS preferences, the FOCUSS framework employs Fuzzy AHP-based approach. The evaluation dimension was achieved by using fuzzy numbers and linguistic variables and employed nine fuzzy linguistic variables to define the scale for the comparison judgements values. These values are triangular fuzzy numbers (TFN) with their underlying triangular membership functions. Next, the user performs the pairwise comparison for all criteria to fill the Fuzzy comparison matrix. For example, a user's degree of importance of the cost criterion over availability can be expressed by the fuzzy number "about strongly important", i.e. $\tilde{a}_{cost,avail} = (6, 7, 8)$. The corresponding reciprocal from on the fuzzy comparison matrix becomes $\tilde{a}_{avail,cost} = (\frac{1}{8}, \frac{1}{7}, \frac{1}{6})$. The *QoS Preference Prioritizer* ensures consistency in the pairwise judgment based on a method proposed by (Csutora and Buckley, 2001), and finally derives priority weights that reflect the relative importance of each criterion to the user using the geometric mean method (Buckley, 1985). Algorithm 1 outlines the process for deriving the priority weights.

b) QoS Aspiration Analyser

The QoS Aspiration Analyser models user-desired QoS values specified using fuzzy linguistics terms and hedge membership functions (cf. Algorithm 2). For example, the use the following linguistic terminologies can be employed when expressing QoS aspiration: *"the threshold of reliability metric should low and be in the vicinity of x",* or *"cost should be cheap and in the range of a and b"* or *"Availability should be high and close to the value z"* etc., where *x, a, b,* and *z* are specific and desired QoS values for reliability, cost, and availability respectively. The fuzzy linguistic variables, '*low'*, '*in the vicinity of*', '*cheap'*, *'in the range of*', '*high'* and '*close to'* are represented using membership functions. Moreover, each QoS attributes consist of a number of membership functions, from which the user can select the ones that most approximates their intention (e.g. see

Table 3.1). Thus, the QoS Aspiration Analyser module synthesises user's QoS values based on fuzzy decision-making system, comprising of the membership functions framed as fuzzy goal and constraints. Since the linguistic terminologies describing the QoS aspiration reflect the semantic approximations of user's intent, resolving the fuzzy decision results in an optimal set of QoS values and the output of this module is a set of values that approximate user's QoS intent.

Algorithm 1: to Derive Priority Weights from Fuzzy Comparison Matrix

Input: Fuzzy Comparison Matrix M= $[\tilde{a}_{ij}]$ of n QoS attributes, and $\tilde{a}$ is a Triangular fuzzy number TFN
Output: Vector ***W*** of priority weights

```
Begin
        for each k=1 to 3
                for each i=1 to n
                        for each j=1 to n
                                r[i] ×= a[i][j]
                        end for
                        r[i]= pow (r[i], n-1)
                        TFN[i][k]=r[i]
                        tot[k]= TFN[i][k]
                        tot[k]=inverse(tot[k])
                end for
        end for
        tot=sortIncreasingOrder(tot)
        for each i=1 to n
                for each k=1 to 3
                        w[i][k]=TFN[i][k]*tot[k]
                end for
        end for
        W=normalize(w)
        Return W
        End
```

Algorithm 2: Derive QoS values from Fuzzy Aspirations

Input: Fuzzy Goals G=[g_i] and Fuzzy Constraints C=[c_i] for n QoS attributes; Let MF be membership functions
Output: Vector V of QoS Aspiration values

```
Begin
    For all i=1 to n
        V[i]= max min (MF_goal(G[i]), MF_constraint (C[i]))
    End for
    Return V
End
```

Table 3.1: Linguistic Terms for fuzzy QoS goals and constraints for Availability

Attribute	Linguistic terms of QoS Goals	Linguistic terms for QoS Constraints
Availability	High Medium Low	Substantially greater than x In the vicinity of x About x Very Close to x

III. Service Evaluation and QoS Ranking Module

Cloud services are characterised by multiple QoS attributes, and there is need to evaluate the overall performance of a given service by some utility functions with respect to users' QoS requirements. Each service alternative is evaluated and ranked in accordance with the user's QoS requirements in the service evaluation and QoS ranking module and this module comprise of the QoS requirements optimizer and the QoS ranking Engine.

a) QoS Requirements Optimizer

The QoS optimizer component computes the optimal QoS values that describe user's requirements based on the available QoS information on all the services contained in the service directory. The inputs into this component are the priority weights for each QoS attributes from the *QoS Preference Prioritizer* module and the values of the QoS attributes synthesised from the *QoS Aspiration Analyser*. Based on the collective QoS information about services in the directory, the FOCUSS framework employs two utility functions: an SAW-based function and a distance metric, exponential Euclidean distance metric- eEUD (2.27), to evaluate the performance of each service alternative. These functions have been discussed in Section 2.4.5. The SAW function is used to evaluate performance utility of each alternative in order to determine the QoS properties of the services alternative with the highest utility, with respect to a user's preferences, while eEUD (see Section 2.4.5) is used to identify those QoS properties of the alternative that is closest to users' requirements, with respect to both preferences and aspiration. The returns from the two functions are used to construct the optimal QoS properties drawn from user's requirement; this is based on the assumption that users always seek to maximise utility subject to their personalised QoS requirements. Hence, the optimal QoS properties are those which simultaneously maximise the utility function as much as possible and closely approximate similarity with user's aspiration. The conflicting nature of functions gives rise to a multiple objective decision-making problems, which the *QoS Requirement Optimizer* models and solves as a fuzzy multi- objective optimisation problem. The objective functions are transformed into a fuzzy goal and constraint and also solved by fuzzy decision making (Bellman and Zadeh, 1970). The resultant output (optimal QoS properties) forms the input into the QoS Ranking Engine.

b) QoS Ranking Engine

The output from the *QoS Requirement Optimizer* forms the basis for ranking the alternatives in the service directory. The main technique used in this module is the nearest neighbour ranking algorithm, based on the eEUD metrics (2.27), that identifies the nearest neighbours to the optimise QoS requirements. The output is the QoS ranking of the alternatives, and *top-k* items become the dataset fed into the bubble chart visualisation. Algorithm 3 outlines the process of the QoS Ranking Engine.

Algorithm 3: Rank Services in Directory based on Optimized QoS Requirements

Input: Vectors V and W representing the QoS aspiration values and priority weights; the service directory A; Let MF be membership functions
Output: Top-k List of services R

```
Begin
    For each item $a_i$ in A do
        perfScore[i] ← Saw ($a_i$, W,V)
        simScore[i] ← eEUD($a_i$ ,W, V)
    For all i=0 to sizeOf(A)
        $MF_d$ ← arg max min ($MF_{saw}$(perfScore[i]),$MF_{eEUD}$(simScore[i]))
    OP ← Evaluate $MF_d$ to return the optimal QoS values that approximate user
    requirements W and V
    For all items $a_i$ in A
        L[i]=eEUD(O, a)
    R ← Rank all items in L according to most similar to O and return top-k
    Return R
End
```

IV. Cloud Ecosystem and Service Directory Module

As part of the FOCUSS framework, a directory of available services is created based on the combinations of atomic services in a systematic manner. The directory of services serves as the baseline for the selection process proper. The cloud ecosystem and service directory module consist of the cloud ecosystem feature model, the reasoning engine and the service directory.

a) Cloud Ecosystem Feature Model

Information about the participating atomic services in the ecosystem, which includes QoS properties, is collected and modelled using Variability Modelling techniques. Noting that the cloud ecosystem structure is analogous to the fundamental principles of software product line engineering (SPLE) (Berger *et al.*, 2014), one of the variability modelling techniques used in the SPLE is used to effectively structure the hierarchical interrelationships among ecosystem atomic services.

The cloud ecosystem feature model, based on *extended feature model* (Benavides *et al.*, 2006), is employed in the FOCUSS framework to organise the services participating in the ecosystem; then the model is transformed into a constraint satisfaction problem based on some mapping rules, and this forms the formal basis to enable automated reasoning on the ecosystem feature model. An automated reasoning process called generate all products, is used to populate the e-marketplace service directory. At least more than one atomic services are composed to form valid combinations, therefore, the QoS properties of the constituent services are aggregated to determine the overall QoS values for the valid combination.

i- Mapping Cloud Ecosystem Feature Model to Constraint Programming

Table 3.2 contains the rules for mapping the Cloud Ecosystem Feature Model (CEFM) into a constraint satisfaction problem using constraint programming.

ii- QoS aggregation functions

Usually, the overall QoS properties of the composite services, conceptualised into a business process, are determined by the QoS attributes of constituent services and their composition relationships. There are four basic composition patterns that inform the arrangement of constituent services in a business process (Mohabbati *et al.*, 2011; Bouanaka and Zarour, 2013; He *et al.*, 2012; Yu and Lin, 2005). They include:

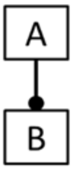
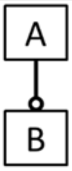
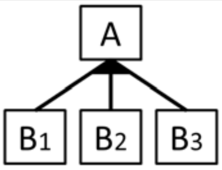
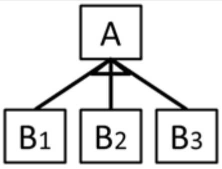
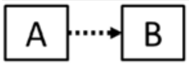
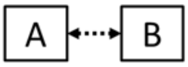
i. **Sequential**: A sequential pattern describes an activity (or services) in the business process that executes after another activity has concluded execution. In other words, the services are executed one after the other.

ii. **Parallel**- In a parallel pattern, all the branches are executed at the same time.

iii. **Conditional (or branch):** Only one branch, with a set of activities, is selected for execution in the branch pattern.

iv. **Loop:** In a loop pattern, an activity in the business process is executed for $(n > 0)$ times.

Table 3.2: Rules for Mapping Cloud Ecosystem Feature Model into CSP

RELATIONSHIPS IN CEFM	CSP MAPPING
 Mandatory	$A = B$
 Optional	$if(A = 0)$ $B = 0$
 OR	$if(A > 0)$ $Sum\ (B_1, B_2 \dots B_n) in\ (1 \dots n)$ $else$ $B1 = 0, B2 = 0 ... B_n = 0$
 Alternative	$if(A > 0)$ $Sum\ (B_1, B_2 \dots B_n) in\ (1 \dots 1)$ $else$ $B1 = 0, B2 = 0 ... B_n = 0$
 Requires	$if(A > 0)$ $B > 0$
 Excludes	$if(A > 0)$ $B = 0$

However, the sequential composition pattern was used in this research. The sequential pattern is the fundamental pattern, as the other patterns (i.e. parallel, conditional and loop), can be reduced or converted to the sequential pattern (Yu and Lin, 2005; Alrifai *et al.*, 2010). Based on the nature of QoS attribute, different aggregation functions can be applied (Yu and Lin, 2005). However, for the purpose of this study, the FOCUSS framework considers only the summation and multiplication aggregation functions (see Table 3.3):

i. **Summation**: In summation aggregation function, the values of a QoS attributes are summed up (e.g. cost and response time). The overall *cost* for a valid combination service should be a summative total of the cost of all constituent services.

ii. **Multiplication**: Multiplication function implies that the aggregate is a product of all the values of a QoS attribute of all the constituent services (e.g. availability).

The QoS aggregation rules for the four QoS properties considered in the FOCUSS framework (i.e. cost, response time, availability and reliability) are given in Table 3.3.

Table 3.3: Aggregation Functions Used in the FOCUSS Framework

AGGREGATION TYPE	QOS ATTRIBUTE	AGGREGATION FUNCTION
Summation	Cost	$q_i(s) = \sum_{j=1}^{t} q_i(Z_j)$
	Response Time	
Multiplication	Availability	$q_i(s) = \prod_{j=1}^{t} q_i(Z_j)$
	Reliability	

Source: Yu and Lin (2005)

The multiplication aggregation functions are non-linear functions. In order to make all aggregate functions to be linear ones, these functions were transformed using $\log\ (q_i(s)) = \log\left(\prod_{j=1}^{t} q_i(Z_j)\right) = \sum_{j=1}^{t} \log(q_i(s))$, a logarithmic function used for such purposes (Li *et al.,* 2014).

b) Reasoning Engine

The FOCUSS framework employs Choco (Jussien, 2008), a general purpose constraint solver, for reasoning on the cloud ecosystem feature model in order to derive useful information from the model, case in point, all valid combinations of constituent services. Choco solver employs, by default, a backtracking approach to find solutions. The search is ordered as an enumeration tree and traversed using a Depth-First Search (DFS) algorithm augmented with variable and value selection heuristics. The solver determines the satisfiability of the CSP, and if a CSP is satisfiable, then solutions can be obtained. The solver searches for a solution in a CSP, using its search strategy to generate all the possible combinations of values for each variable in the CSP and certifies that they correspond to a solution of the CSP. Table 3.2 shows the rule for mapping constructs in the cloud ecosystem feature model into CSP. The corresponding CSP representation of the model is read by the reasoning engine and performs automated analysis of the CSP representation to generate all valid service combinations with aggregated QoS information based on the aggregation functions listed in Table 3.3.

c) Service Directory

The service directory indexes all the QoS information about the collection of valid combination services generated by all products operations on the cloud ecosystem feature model. Based on Definition 3.3 and Algorithm 4, the service directory is modelled as case base and stored in a relational database with columns and rows representing the QoS attributes and the QoS values for each valid combination service.

Algorithm 4: Populate Cloud service directory with Composite Services

Input: Cloud Ecosystem Feature Model (CEFM) of atomic services with *n* number of QoS attributes
Output: Service Directory A

```
Begin
  S ← reasoningEngine(CEFM) generate sets of valid composite services based on constraints
  for each composite_service C in S
    for each i=1 to n
        for each atomic_service a in C
            $A_s[i]$ = aggregate (QoS(a[i]))
        end for
    end for
  end for
  Return A
End
```

3.4.3 Justification for the FOCUSS Framework

To satisfy *Requirement 1*, the FOCUSS framework uses a cloud ecosystem feature model (CEFM) that is based on *extended feature model* (Benavides *et al.*, 2006) to organise the atomic services that are participating in the ecosystem. The CEFM is transformed into a formal representation as a constraint satisfaction problem and one of the automated reasoning operations performed on the formal representation of the CEFM generate all valid combinations is used to populate the e-marketplace service directory.

The FOCUSS framework fulfils *Requirement 2* by employing fuzzy set theory to elicit QoS preferences and aspirations while taking into consideration both users' preferences and aspiration. To determine the user's preferences on QoS attributes, the advantage of pairwise comparisons to derive priority weights of each QoS attribute from comparison matrices far outweighs direct and arbitrary assignment of weights (Javanbarg *et al.*, 2012). The result of each pairwise comparison is a numerical value denoting the estimated ratio between the weights of any two criteria; and the weights are crisp values obtained from Saaty's scale (Saaty, 1980). The AHP method proposed by (Saaty, 1980) provides some measure of flexibility in judgment by ensuring intermediate values in the Saaty's discreet scale (Cakir and Canbolat, 2008).

On the other hand, human judgment is shrouded with impression and vagueness. In most practical cases, and users might be reluctant or unable to assign exact numerical values in comparison judgements (Mikhailov and Tsvetino, 2004). Comparison judgement using crisp numerical values lacks the flexibility and robustness required to effectively capture the vague perception inherent in human judgement, and sometimes, lead to unsatisfactory decisions (Cakir and Canbolat, 2008; Javanbarg *et al.*, 2012; Mikhailov and Tsvetino, 2004). User's claim about the relative importance of the QoS criteria can to delineated comparison ratios as fuzzy numbers (Cakir and Canbolat, 2008; Tajvidi *et al.*, 2014; Mikhailov and Tsvetino, 2004).

Similarly, expressing QoS aspiration also benefits from the flexibility provided by employing fuzzy set theory, where rather than expressing the value of QoS attributes using exact crisp values, linguistic variables defined by membership functions can be used together with hedges.

Cloud services e-marketplaces are characterised by a large set of services, which are most times functionally equivalent. The FOCUSS framework fulfils *Requirements 3*, by employing a fuzzy multi-objective optimisation mechanism that accurately evaluates and rank a large set of services in accordance with user's QoS requirements.

Requirement 4 is satisfied by the FOCUSS framework as it integrates fuzzy-based web widgets for eliciting vague preferences and aspirations in an integrated visual interface. GUI designs that can intuitively capture these requests are naturally desirable. Indeed, the user's perception of the interface affects attitude to what comes out of it, and ultimately affects user satisfaction (Kuniavsky, 2003; Sundar *et al.*, 2014). Estimating relative pairwise comparison can be made numerically, graphically, or linguistically (Forman and Gass, 2001). However, a graphical and linguistic approach further reduces cognitive load on the user and is easier than expecting the user to enter crisp numeric ratios. The choice of slider bars is motivated by the study performed by (Millet, 1997), which shows that interaction or engagement is better off using slider by giving the user the opportunity to adjust and interact directly with the elements on the screen.

The FOCUSS framework fulfils *Requirement 5* by including the bubble graph as a visualisation mechanism for improving the understanding of the rationale for the rankings of cloud services based on the user's requirements. Most cloud service ranking and selection systems are black boxes, providing a list of ranked cloud services with no transparency into the reasoning behind the ranking results (Chen *et al.*, 2013). Arguably, confidence in the ranking results would be enhanced if users are privy to the knowledge of the underlying rationale. The graph explicitly would show the relationships of the top ranked cloud services as well as the underlying structure of the QoS space by using bubbles, colours, and size in a spatial arrangement. This exploratory mechanism provides valuable insight into the QoS information space and enables an improved understanding of how each service in the ranking relates to others in the QoS information space overall performance evaluations of cloud services (Chen *et al.*, 2013).

To validate the FOCUSS framework, an illustrative case study is undertaken to demonstrate the practicality of the FOCUSS framework, while controlled experiments are performed to assess the ranking accuracy and scalability of the FOCUSS framework. Apart from the performance and accuracy evaluations which are predominant in literature, user studies were carried out to ascertain the suitability of the FOCUSS framework in the e-marketplace context; thus ensuring that the FOCUSS framework fulfils *Requirement 6*.

3.5 ASSUMPTIONS

The underlying assumptions underpinning the proposed FOCUSS framework are highlighted as follows:

1. It is assumed that there would be no failure on the part of any of the services and that all services are available in every given instance.
2. The QoS information given is accurate and reflects the true performance of the services.
3. All valid combinations would be deployed on host e-marketplace infrastructure.
4. There are service composition realisation or actuator mechanisms that concretise valid combinations for onward cloud deployment for the user.
5. Every other aspect of the e-marketplace is functional, as this study is only concerned with the aspect selecting cloud services from a large pool in an e-marketplace context.

6. The number of QoS dimensions considered is limited to four, given the multivariate constraints imposed by the bubble graph visualisation.

7. Noting the dynamic nature of the cloud computing domain, correctly evaluating the performance of cloud service should be an on-going concern, a major assumption is that the QoS properties of the atomic services remain unchanged during the decision process, that is, all $z_i \in Z$ are constant. This assumption reduces cloud into a decision problem without uncertainty (Rehman *et al.*, 2011).

8. It is assumed that providers have correctly specified their QoS requirements; however, a consistent update of the QoS information about the services, based on monitoring benchmark values from third party services is required (Ruiz-Alvarez and Humphrey, 2011).

3.6 CHAPTER SUMMARY

In this chapter, several models describing the Fuzzy-Oriented Cloud Service Selection Framework (FOCUSS) were presented. The FOCUSS framework is presented as an integrated, feature-based and visual-rich fuzzy-based decision making framework for cloud service selection in cloud service e-marketplace context and attempts to answer the research questions posed in this study. The automated analysis of cloud ecosystem feature models populates the service directory, while the fuzzy theories are employed to elicit user QoS requirements via interactive GUI, through which ranking of service alternatives can also be explored. The practical demonstration and validation of the FOCUSS framework will be discussed in the subsequent chapters.

CHAPTER FOUR

IMPLEMENTATION

4.1 INTRODUCTION

The previous chapter contained the description of the FOCUSS framework proposed in this study for service selection in cloud service e-marketplace. In this chapter, the details of the implementation of the FOCUSS framework are presented first, using some software tools, technologies and middleware frameworks. Next, this chapter contains details of an illustrative case study of a Cloud-based Customer Relationship Management Software-christened Customer Relationship Management as a Service (CRMaaS). Based on a GUISET use case, the CRMaaS provides a scenario through which the practical application of the FOCUSS framework was demonstrated.

4.2 IMPLEMENTATION DETAILS

In order to realise the FOCUSS framework and demonstrate its applicability, a set of technological tool has been identified. These tools were categorised into different functional areas- Integrated Development Environment (IDE), Front-end Components, and Back-end components- Java was the primary programming language used to implement components of the FOCUSS framework. The tools used to implement the components of the FOCUSS framework are presented as follows:

4.2.1 Integrate Development Environment: NetBeans 8.1

NetBeans 8.1 is a free open-source cross-platform integrated software development platform written in Java and allows applications to be developed from a set of modular software components called modules, which can be extended by third parties. Apart from supporting developments in Java, the NetBeans IDE also supports other languages, such as PHP, C/C++, XML and HTML5. The NetBeans IDE is cross-platform and runs on platforms supporting a compatible JVM, including Microsoft Windows. The NetBeans complete bundle provides complete tools for Java EE, SE and ME standards, including Web profile, Enterprise Java Beans (EJB), Servlets technologies, Java Persistence API, web services, and annotations. NetBeans also supports the JavaServer Pages (JSP) and includes GlassFish and Apache Tomcat servers. With NetBeans IDE, desktop, mobile and web applications can quickly be developed in Java, as well as HTML5 applications with HTML, JavaScript, and CSS. Furthermore, the NetBeans IDE provides drivers for the Java DB, MySQL, Oracle, and PostgreSQL database servers, as well as other JDBC drivers. NetBeans 8.1 IDE served as the umbrella environment for the implementation of the components of the FOCUSS framework.

4.2.2 Front-end Web-based UI

To achieve improved user experience in the FOCUSS framework, the graphical user interface components were realised using a combination of front-end technologies, languages and framework, which are presented subsequently.

a) JavaServer Pages (JSP)

JavaServer Pages (JSP) is a technology that is used to create dynamically generated web pages based on HTML, XML or other document types using Java. JSP files are deployed and run on a compatible web server that possesses a Servlet container (e.g. Apache Tomcat or Jetty). JSP is considered high-level abstractions of Java Servlets and are translated into Servlets at runtime. With JSP, Java codes and predefined actions are commingled with markup languages (e.g. HTML) and are executed by a Java Virtual Machine (JVM) that interacts with the server's host operating system to provide an abstract, platform-neutral environment.

b) Hyper Text Markup Language

HyperText Markup Language (HTML) is the standard markup language for creating and presenting web pages. Published in October 2014, HTML5 is the fifth and current version of the HTML standard, and improves on previous HTML standards with support for the multimedia, and provides API for complex web applications.

c) Cascading Style Sheets

Cascading Style Sheets (CSS) is a stylesheet language used for determining how a document written in a markup language is presented in a web browser. Together with HTML and JavaScript, CSS is employed to create visually engaging and appealing web pages and user interfaces for web applications and mobile applications. The main concept of employing CSS is to separate the content of a document from its presentation, and as such improve content accessibility, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate *.css* file, and reduce complexity and repetition in the structural content. Apart from being used to create the visual appearance of web pages.

d) JavaScript

JavaScript (JS) is a high-level, dynamic, untyped, and interpreted programming language used together with HTML, and CSS to create web based contents. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language that supports a variety of programming paradigms such as object-orientation, imperative, and functional programming. It possesses API for manipulating text, arrays, dates and regular expressions. JS is a client-side programming language used to dynamically alter the content of an HTML document. JavaScript was heavily employed in the implementation of the front-end components of the FOCUSS framework.

e) BootStrap Framework

The FOCUSS framework employs BootStrap 3.3.6 (bootstrap.com), a free and open-source HTML, CSS and JS framework for creating and styling the web user interface. BootStrap supports responsive web UI design, in that it adapts dynamically to the characteristics of the device in use. It comprises design templates for layout, forms, buttons, navigation and other UI components and provides a consistent appearance for formatting text, tables and HTML form elements. BootStrap allows the use of modal windows to reduce on-screen clutter, coloured buttons to dictate functionality, and tabbed panes, to help split the system into smaller segments.

BootStrap is compatible with many modern browsers such as Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, and Opera. The Jquery JavaScript library was used to manipulate the HTML. Jquery is fast, light, and is a collection of feature-rich JavaScript library, that greatly simplifies document traversal and manipulation, and event handling.

f) Google Chart Visualization

Google Charts provide customizable JavaScript classes for visualising data on web pages. The Google Chart JavaScript libraries expose a variety of chart types including line charts, treemaps, scatter plot, bubble graph, etc. The Charts are rendered in an HTML5/SVG technology that works across browser types. The Charts are populated from data sources such as a database or directly from a web page. The FOCUSS framework employs the Bubble Chart from the Google Chart types to visualise the QoS ranking of Cloud service alternatives with respect to users' QoS requirements.

4.2.3 Back-end Components

a) Glassfish Web Server

This is a fully compliant implementation of the Java EE 5 platform. It provides the necessary middleware infrastructure support for all the Java APIs. The Application Server includes a number of Java EE tools that are not part of the Java EE 5 platform but are provided as an additional support to the developer.

b) Java Servlet Technology and Java Classes

A Java Servlet is a Java objects deployed in a web container and used to extend the functionalities of a server. Servlets support hosted applications that comply with the request-response communication model, and are the Java equivalent of dynamic web technologies like PHP and ASP.NET. The web container in which the Servlets are hosted handles the Servlet lifecycle and maps an URL to specific servlets while ensuring that the URL requester possesses the correct access rights. Servlets and Java classes were used to encapsulate the business logic of the FOCUSS framework. The business logic is the code that fulfils the purpose of the application; For example, a method in a Java class implements the business logic *derivePriorityWeights*. When *derivePriorityWeights* is invoked, the QoS preferences of the users based on the Fuzzy comparison matrix would be performed and the vector of the priority weights is returned for further processing. Other components responsible for the core business logic of the FOCUSS framework include the QoS Prioritizer, QoS Analyzer, QoS Requirement Optimizer, The QoS Ranking Engine, and the Reasoning Engine.

c) Choco Constraint Library and Solver

Choco (Jussien *et al.*, 2008) is a Free and Open-Source Software dedicated to Constraint Programming. It is a Java library used to describe hard combinatorial problems as Constraint Satisfaction Problems (CSP) and solved using constraint programming techniques. Problems are modelled in a declarative way by stating the set of constraints that need to be satisfied in every solution, and Choco solver solves the problem by alternating constraint filtering algorithms with a search mechanism. Choco supports four types of variables (Integer, Boolean, Set and Real), many global constraints, solution search strategies, explanation-based engine, that enables conflict-based back jumping, dynamic backtracking and path repair. The FOCUSS framework employed Choco 2.1.5 to describe the cloud ecosystem feature model that organises the participating services in the cloud ecosystem and describes their relationship with each other. Choco solver uses, by default, a backtracking approach to finding solutions. The search is ordered as an enumeration tree and traversed using a Depth-First Search (DFS) algorithm augmented with variable and value selection heuristics. The model provides a template for valid combinations of services based on some imposed constraints.

d) The MOEA Framework

The MOEA Framework (moeaframework.org) is a free and open source Java library of Multi-Objective Evolutionary Algorithms (MOEAs) and other general-purpose multiobjective optimisation algorithms. The framework supports genetic algorithms, differential evolution, particle swarm optimisation, genetic programming, and grammatical evolution to formulate and solve multiobjective optimisation problems. New problems are formulated in the MOEA Framework using decision variable(s) encoded as any of binary, strings, real-valued numbers, and permutations. After the definition, problems can then be optimised using the MOEA algorithms available in the framework. Specifically, the MOEA Framework comprises the following algorithms NSGA-II, NSGA-III, ε-MOEA, GDE3, PAES, PESA2, SPEA2, IBEA, SMS-EMOA, SMPSO, OMOPSO, CMA-ES, and MOEA/D.

e) MySQL Database

MySQL is the most popular open-source relational database management system (RDBMS) for web-based application and is a central component of the widely used

LAMP open source web application software stack (and other "AMP" stacks). The latest MySQL version 5.7.11 was employed in implementing the service directory of the FOCUSS framework. The MySQL database server was integrated into the NetBeans 8.1 IDE via the MySQL Connector/J Java Database Connectivity API. The API allows SQL commands to be invoked from Java programming language methods. The connector is used in an enterprise bean when there is a need for a session bean to access the database. The connector can also be used from a Servlet or a JSP page to access the database directly, bypassing the enterprise bean. The summary of the technologies employed to implement the FOCUSS framework is presented in Table 4.1.

Table 4.1: Summary of Tool Support to Realise the FOCUSS Framework

#	Module	Language	Technology/Library/ Framework
1	GUI: Front-end	Java	JSP
		HTML CSS JavaScript	BootStrap 3.3.6
2	GUI: Visualization	JavaScript	Google Chart API
3	QoS Aspiration Analyzer	Java	MOEA Framework 2.9
4	QoS Preference Prioritizer	Java	Servlet and Java Class
6	QoS Requirements Optimizer	Java	MOEA Framework 2.9, Servlet
7	QoS Ranking Engine	Java	Servlet and Java Class
8	CEFM	Java	Choco 2.1.5 Servlet and Java Class
9	Reasoning Engine	Java	
10	Service Directory	SQL	MySQL 5.1.17, Java Servlet
Integrated Development Environment			**NetBeans 8.1**
Web Application Server			**Glassfish Web Server**

4.3 Illustrative Case Study

In this section, a cloud ecosystem and e-marketplace scenario is presented to demonstrate the practical application of the FOCUSS framework. As an illustrative case study, the GUISET project was considered. GUISET is envisioned as both an enabling infrastructure and a suite of on-demand services. The primary motivation for the GUISET project is economic advantages of enterprise clusters over the stand-alone organisation. These advantages include resource sharing, cost reduction, and the ability to compete with larger firms (Braun, 2005). As a cloud computing model, GUISET is aimed at offering affordable e-enabling and “appliance-like” technology services through the Internet to lower the total cost of ownership. The GUISET infrastructure would provide small businesses with business-relevant services on a pay-as-you-go basis. These services are aimed at e-enabling the activities of under-resourced local Very Small Software

Enterprises (VSSE) and provide the platform for these VSSE to participate in the global market of e-services in an ecosystem environment. VSSE can leverage the capabilities of the GUISET infrastructure and e-marketplace platform to trade value-added services relevant to other enterprises that are also part of the GUISET ecosystem. The relevance of pursuing an e-marketplace for CRMaaS initiative is to provide a viable platform where local VSSE can readily participate in provisioning services on the global scale. The application of the FOCUSS framework will facilitate easy discovery of services offered by local VSSE. Ultimately, this will promote the profitability of the VSSEs and multiply their economic returns and impact. Even though many local VSSE are characterised by meagre budgets, they still contribute directly and indirectly to the national GDP, through employment generation and wealth creation (Venesaar and Loomets, 2006; Hamwele, 2005). This contribution can be sustained and possibly increased by participation in an ecosystem exposed via a cloud e-marketplace. Based on a GUISET use case, a cloud-based Customer relation management software, called Customer Relationship Management as a Service (CRMaaS), serves as an illustration of cloud ecosystem and e-marketplace scenario in order to validate the framework proposed in this research.

4.3.1 Customer Relationship Management

Customer Relationship Management (CRM) refers to ways by which companies coordinate and analyse user interactions and data all through the lifecycle of a customer. These ways may include technology, people and organisational strategies deployed to collect user information about personal data, purchase history, preferences, and concerns across different channels, through which the organisation engages with the user. These channels may include phone conversations, emails, social media, etc. Customers information are consolidated into the CRM database and the organisation utilises this data to improve business relationships so as to achieve user retention and increased sales. Traditional on-premise CRM software puts the burden of administration and maintenance on the organisation, however, employing cloud-based solutions outsources these services to a third party, leaving the organisation to focus on its core business, particularly when technological expertise and budget is limited.

4.3.2 Customer Relationship Management as a Service (CRMaaS)

Customer Relationship Management as a Service (CRMaaS) is a cloud ecosystem of CRM solutions for Small and Medium Enterprises (SME) delivered through the GUISET cloud e-marketplace. The components that make up the CRMaaS ecosystem includes: Contact Management, Database, Marketing and Social media analysis (see Figure 4.1). The CRMaaS solution is realised by the participation of various service providers in the ecosystem. One or many providers can contribute one or more of the following range of services to the ecosystem with differentiated QoS factors. The description of each module is as follows:

i. ***Contact Management Service*****:** Tool to manage user contacts and communication; including appointment management, task management and scheduling, communication (SMS, email),

ii. ***Cloud Database:*** Cloud-based Relational Database Management System (RDBMS) to store user information, including user personal data, purchase history, preferences etc.

iii. ***Marketing Service:*** Tools for communicating with users; including email marketing, text message marketing, social media marketing etc.

iv. ***Social Media Analytics:*** Tool that monitors conversations on social media and analyses feedbacks, capturing user sentiments.

v. ***Cloud Platform:*** The valid combinations derived would require a cloud platform on which to run.

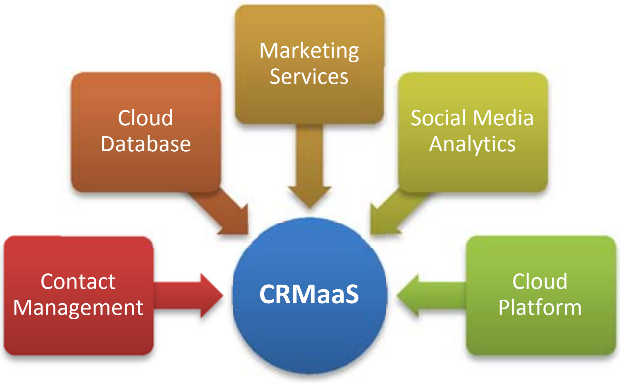


Figure 4.1: High-level Structure of the components of a CRMaaS

An instance of the CRMaaS offering is a combination of any/all of these services to create a complete CRM solution. In the GUISET e-marketplace, multiple variants of CRMaaS solutions exist and are differentiated by QoS factors that are relevant to any SME. The SME can then search for and consume CRM solution that aligns with their specific aspiration and preferences. Furthermore, the multi-tenant nature of the CRMaaS allows for multiple SMEs (tenants) to be hosted per time, each having a variant view of the CRMaaS that suits specific requirements. Therefore, the e-marketplace service directory contains a set of *m* CRM solutions that can be evaluated along *n* decision criteria with respect to an SME's preferences and aspirations. The cloud service selection in this context is concerned with the evaluation of the set of *m* offerings based on the preferences and aspirations on the set of *n* criteria (Sahri *et al.*, 2014). Having expressed requirements, which are converted to a search query, the e-marketplace platform generates search results in form ranking of complete CRM solutions that approximate the requirements expressed. An SME that requires a complete cloud-based CRM solution for managing its customer relationship processes in a bid to improve the business relationship and increase the bottom-line can find the most appropriate solution via the GUISET e-marketplace. Two examples of such SME are as follows*: A micro-finance bank* and a newly opened *on-line drug store*, and these examples are used throughout the use case.

In the following paragraph, high-level scenario descriptions of their requirements are presented.

i. **Case One: Microfinance Bank-** A microfinance bank (MFB) provides microfinance services such as savings, loans, domestic funds transfer, and other financial services to under-resourced, micro, small and medium enterprises to enable them to conduct or expand their businesses. The operations of an MFB are time-critical and data sensitive; thus they require a solution that is stable with little or no unpredictable issue. MFB may also require that the solution should be of excellent performance that must be available and highly reputable, as their operations involve sensitive user information. The micro-finance bank requires a reliable solution that meets all its requirements and has made adequate budgetary provisions to offset the cost.

ii. **Case Two: Online Drug Store-** On the other hand, a new online drug store set up to expand an existing brick-and-mortar drugstore. The online drugstore allows existing and prospective users to purchase and pays for over-the-counter medication online. The owner of the drug store prefers a low-priced reliable CRM solution that can handle basic customer relationship management processes. Also, being a small start-up, the owner is less keen on reliability, and based on current cash flow realities, is constrained by the amount of funds that can be spent on the CRM solution.

4.4 PRACTICAL DEMONSTRATION OF THE FOCUSS FRAMEWORK

This section presents how the FOCUSS framework is used both to set up a cloud ecosystem for realising the CRM software and create the e-marketplace platform that enables the designated information technology officers of the MFB and the drug store to search for and select the appropriate CRM solution that is based on their specific requirements.

4.4.1 CRMaaS Ecosystem Model and Reasoning Engine

Based on the components of the CRMaaS presented in Figure 4.1, more than one candidate cloud service, among others, would suffice in fulfilling each of contact management, database, marketing, social media analysis and cloud platform on which the CRMaaS runs. Table 4.2 contains the list of all the constituent services that can fulfil each component, together with the values of the QoS attributes, and are part of the CRMaaS ecosystem. The QoS attributes considered in this example includes availability and reliability, measured in percentages (%); response time measured in milliseconds (ms), while the cost is measured in Dollars/month ($/Month).

Table 4.2: Candidate Cloud Services to realize CRMaaS Components

CRMaaS Components	Candidate Services	QoS Values			
		Availability (%)	Response Time (ms)	Reliability (%)	Cost ($/Mon)
Contact Management	CM1	90	--	90	30.50
	CM2	95	--	67	29.99
	CM3	70	--	40	25.50
	CM4	99	--	79	34.99
Cloud Database	CD1	89	100.22	60	13.50
	CD2	79	50.54	75	20.50
	CD3	97	120.34	80	50.00
Marketing	M1	99	--	--	55.50
	M2	91	--	--	59.99
Social Media Analysis	SMA1	90	200.45	88	49.99
	SMA2	95	138.56	90	50.00
	SMA3	85	125.45	79	45.67
Platform	P1	99	300.45	70	199.99
	P2	99	423.10	75	149.99

The candidates services for each CRMaaS component is given as follows (Table 4.2): Contact management (CM1, CM2, CM3, CM4); Cloud Database (CD1, CD2, CD3); Marketing (M1, M2); Social Media Analysis (SMA1, SMA2, SMA3); Platform (P1, P2). The values of the QoS properties were populated by randomly generated numbers.

Figure 4.2 shows the feature model of the CRMaaS cloud ecosystem without the QoS attributes annotated in the diagram. The model logically structures and describes the relationship among the atomic services. The rules guiding the combination of these candidate services are contained in Table 4.3, while the CEFM that models the relationships and constraints is presented in Figure 4.2. All CRMaaS components are mandatory; however, each candidate service is an alternative to other candidate services within the same component group.

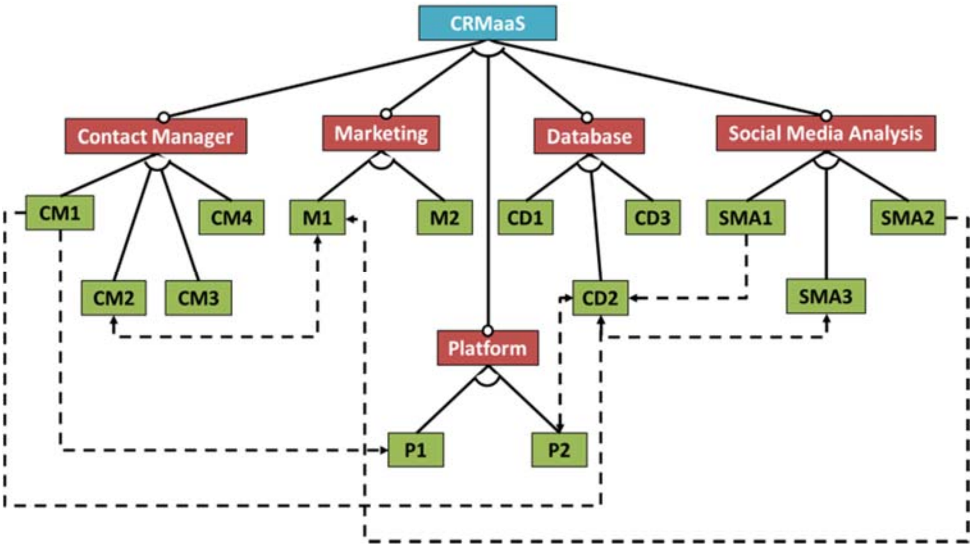


Figure 4.2: High-Level Feature Model of CRMaaS Cloud Ecosystem (Without QoS Attributes)

Table 4.3: Require and Exclude Constraints

CM1	REQUIRES	P1
CM1	REQUIRES	CD1
CM2	EXCLUDES	M1
SMA1	REQUIRES	CD2
CD2	EXCLUDES	P2
SMA2	REQUIRES	M1
SMA3	EXCLUDES	CD2

From the model, it is obvious that cloud database CD2 cannot run on cloud platform P2, therefore no valid combination would contain both cloud database CD2 and platform P2. The feature model is transformed into constraint programming based on the rules for the mapping of feature model to CSP. The Java code for the constraint programme for the model in Figure 4.2 is contained in Appendix D. A total of 38 valid combinations that form actual CRMaaS instances that can be offered to users were obtained from the automated reasoning on the CSP model that defines the Cloud Ecosystem Feature Model (see Table 4.4). The QoS properties of the valid combination were computed based on the aggregation functions described in Section 3.2.

Table 4.4: List of Valid combinations based on CRMaaS Cloud Ecosystem Model

Service_ID	Constituents Services	Aggregate QoS Values*			
		Availability (%)	Response Time (ms)	Reliability (%)	Cost ($/Mon)
S1	CM4 CD3 SMA3 M2 P2	98.68	668.89	75.73	340.64
S2	CM3 CD3 SMA3 M2 P2	97.16	668.89	72.78	331.15
S3	CM4 CD3 SMA3 M2 P1	98.67	546.24	75.43	390.64
S4	CM3 CD3 SMA3 M2 P1	97.16	546.24	72.48	381.15
S5	CM4 CD1 SMA3 M2 P2	98.29	648.77	74.48	304.14
S6	CM3 CD1 SMA3 M2 P2	96.79	648.77	71.53	294.65
S7	CM4 CD1 SMA3 M2 P1	98.29	526.12	74.19	354.14
S8	CM3 CD1 SMA3 M2 P1	96.79	526.12	71.23	344.65
S9	CM2 CD3 SMA3 M2 P2	98.49	668.89	75.02	335.64
S10	CM2 CD3 SMA3 M2 P1	98.49	546.24	74.72	385.64
S11	CM2 CD1 SMA3 M2 P2	98.11	648.77	73.77	299.14
S12	CM2 CD1 SMA3 M2 P1	98.11	526.12	73.47	349.14
S13	CM4 CD3 SMA3 M1 P2	99.03	668.89	75.73	336.15
S14	CM3 CD3 SMA3 M1 P2	97.53	668.89	72.78	326.66
S15	CM4 CD3 SMA2 M1 P2	99.51	682	76.3	340.48
S16	CM3 CD3 SMA2 M1 P2	98.01	682	73.34	330.99
S17	CM4 CD3 SMA3 M1 P1	99.03	546.24	75.43	386.15
S18	CM3 CD3 SMA3 M1 P1	97.53	546.24	72.48	376.66
S19	CM4 CD3 SMA2 M1 P1	99.51	559.35	76	390.48
S20	CM3 CD3 SMA2 M1 P1	98.01	559.35	73.04	380.99
S21	CM4 CD1 SMA3 M1 P2	98.66	648.77	74.48	299.65
S22	CM3 CD1 SMA3 M1 P2	97.15	648.77	71.53	290.16
S23	CM4 CD1 SMA2 M1 P2	99.14	661.88	75.05	303.98
S24	CM3 CD1 SMA2 M1 P2	97.63	661.88	72.1	294.49
S25	CM4 CD1 SMA3 M1 P1	98.66	526.12	74.19	349.65
S26	CM3 CD1 SMA3 M1 P1	97.15	526.12	71.23	340.16
S27	CM4 CD1 SMA2 M1 P1	99.14	539.23	74.75	353.98
S28	CM3 CD1 SMA2 M1 P1	97.63	539.23	71.8	344.49
S29	CM1 CD1 SMA3 M2 P1	97.88	526.12	74.75	349.65
S30	CM1 CD1 SMA3 M1 P1	98.24	526.12	74.75	345.16
S31	CM1 CD1 SMA2 M1 P1	98.73	539.23	75.32	349.49
S32	CM4 CD2 SMA1 M2 P1	98.02	551.35	75.62	360.46
S33	CM3 CD2 SMA1 M2 P1	96.52	551.35	72.67	350.97
S34	CM2 CD2 SMA1 M2 P1	97.84	551.35	74.91	355.46
S35	CM4 CD2 SMA2 M1 P1	98.62	489.46	75.72	360.98
S36	CM3 CD2 SMA2 M1 P1	97.12	489.46	72.76	351.49
S37	CM4 CD2 SMA1 M1 P1	98.39	551.35	75.62	355.97
S38	CM3 CD2 SMA1 M1 P1	96.88	551.35	72.67	346.48

*The QoS aggregation is performed using the functions listed in Table 3.3

4.4.2 Fuzzification of QoS Information of Services in Service Directory

The QoS information about the services offered through the e-marketplace was fuzzified by representing three ranges of QoS values with linguistic variable and underlying membership functions. The range of QoS values for Availability QoS is broken into four, namely: Very High, high, medium and Low. The range of Reliability is Very high, high, Average and Low, while that of Response time is Low, Acceptable and below Average. The linguistic values for Cost QoS are Premium, Standard, Moderate and Cheap. Table 4.5 shows the QoS attributes, the linguistic variables and the membership function used to represent each QoS attribute.

Table 4.5: QoS Attributes, fuzzy sets and underlying membership function

QoS Attribute	Fuzzy sets	Membership Function
Availability	Very High, High, Medium, Low	Trapezoidal Membership Function
Response Time	Low, Acceptable, Below Average	
Reliability	Very High, High, Average, Low	
Cost	Premium, Standard, Moderate, Cheap	

Based on the available QoS information of all services in the service directory (see Table 4.4), Figure 4.3 shows the range of values under each linguistic variable for each QoS attribute and the membership function diagram used in this case study.

Linguistic Variable: Availability

Linguistic Term	QoS Value Range
Very High	90% -- 100%
High	70% -- 95%
Average	60% -- 85%
Low	50% -- 75%

Linguistic Variable: Response Time

Linguistic Term	QoS Value Range
Low	200ms – 560ms
Acceptable	500ms – 790ms
Below Average	700ms – 1000ms

Linguistic Variable: Reliability

Linguistic Term	QoS Value Range
Very High	90% -- 100%
High	70% -- 95%
Average	60% -- 85%
Low	50% -- 75%

Linguistic Variable: Cost

Linguistic Term	QoS Value Range
Premium	370$ -- 500$
Standard	280$ -- 400$
Moderate	190$ -- 300$
Cheap	100$ -- 200$

Figure 4.3: Linguistic Variables for QoS attributes

Apart from the QoS range, users are also allowed to express some form of constraints to qualify whatever linguistic term they select.

Table 4.6 shows the various linguistic hedges and their associated membership functions. These constraints include: In the Vicinity of x, and very close to x, where x is a QoS value specified by the user.

Table 4.6: Linguistic Hedges and Membership Functions for each QoS Attributes

LINGUISTIC HEDGES FOR QOS VALUE	MEMBERSHIP FUNCTION
x is In the vicinity of a	$\mu_C(x) = \frac{1}{(1 + (x - a)^4)}$
x Very close to a	$\mu_{\tilde{C}}(x) = \left(\frac{1}{1 + (x - a)^2}\right)^2$
x Substantially Higher than a	$\mu_c(x) = (1 + (x - a)^{-2})^{-1}$
x Substantially Lower than a	$\mu_c(x) = (1 + (a - x)^{-2})^{-1}$
x Approximately between a and b	$\mu_c(x) = (1 + a(x - 6)^{-5})^{-1}$
a and b are actual QoS values specified by the user	

4.4.3 Eliciting User Requirements

Based on the two instances of the MFB and an online drug store discussed earlier, the user performs a pairwise comparison of all QoS attributes to enable the system to determine the relative importance of each QoS attributes to the user. In addition, the user specifies QoS aspirations using the linguistics terms and hedges for QoS values described in the previous section. Table 4.7 and Table 4.8 show the QoS priorities and aspirations for MFB respectively; while Table 4.9 and Table 4.10 contain the QoS priorities and aspirations for the ODS respectively. An example of how Availability QoS requirements are expressed using the FOCUSS GUI for MFB and ODS are shown in Figure 4.4 and Figure 4.5.

Table 4.7: QoS Pairwise comparison for MFB

QoS Attribute	Fuzzy Judgement	QoS Attribute
Availability	Extremely more important than	Response Time
Availability	Extremely less important than	Reliability
Availability	Somewhat Less important than	Cost
Response Time	About equal	Reliability
Response Time	About equal	Cost
Reliability	Somewhat more important than	Cost

Table 4.8: QoS Aspiration for MFB

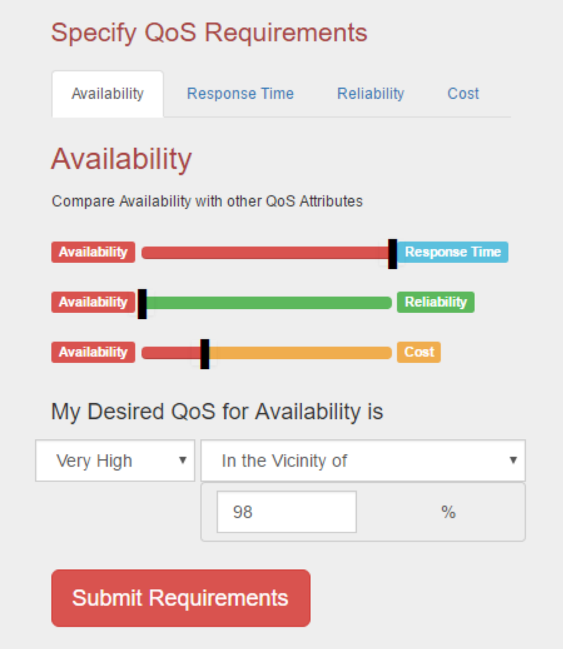
QoS Attribute	Goal	Hedges/Constraints
Availability	Very High	In the Vicinity of 98%
Response Time	Low	Very close to 400ms
Reliability	Very High	In the Vicinity of 75%
Cost	Premium	In the Vicinity of 400$

Table 4.9: QoS Pairwise comparison and Aspiration for Online Drug Store

QoS Attribute	Judgement	QoS Attribute
Availability	About Equal	Response Time
Availability	About Equal	Reliability
Availability	Extremely Less important than	Cost
Response Time	About Equal	Reliability
Response Time	Extremely less Important than	Cost
Reliability	Extremely less Important than	Cost

Table 4.10: QoS Aspiration for Online Drug Store

QoS Attribute	Goal	Constraints
Availability	High	In the Vicinity of 90%
Response Time	Acceptable	In the Vicinity of 600ms
Reliability	High	Very close to 70%
Cost	Cheap	In the vicinity of 250$



Specify QoS Requirements

Availability

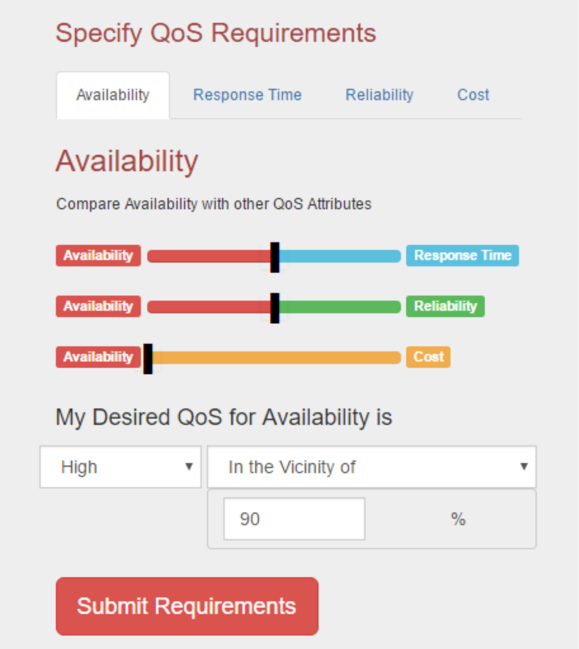
Compare Availability with other QoS Attributes

My Desired QoS for Availability is

Very High ▾ | In the Vicinity of ▾

98 %

Figure 4.4: Availability QoS Requirements for Microfinance Bank in FOCUSS GUI



Specify QoS Requirements

Availability

Compare Availability with other QoS Attributes

My Desired QoS for Availability is

High ▾ | In the Vicinity of ▾

90 %

Figure 4.5: Availability QoS Requirements for Online Drug Store in FOCUSS GUI

The GUI employs a dual colour coded slider bars that correspond to the colour code for the two QoS attributes being compared. When the slider bar is in the middle (i.e. the length of either colour in the slider bar are equal), then the underlying fuzzy comparison scale is ‘about equal’. Furthermore, there are eight steps on either side of the midpoint of the slider bar corresponding to the other scales in the fuzzy Saaty pairwise comparison scale.

4.4.4 QoS Requirements Processing

I. QoS Prioritization

The fuzzy prioritisation method, based on Geometric Mean Method (Buckley, 1985) was applied to derive crisp weights representing the relative importance of each QoS attributes from the fuzzy pairwise comparison matrix. Based on the Geometric Mean Method (Buckley, 1985), the crisp weights from the fuzzy pairwise comparison for MFB and ODS are shown in Table 4.11 and Table 4.12 respectively. These tables show that the order of relative importance of the QoS attributes for MFB is as follows Reliability>Cost>Availability>Response Time; while the most important QoS attribute to ODS is cost and the other QoS attributes have equal weights.

Table 4.11: Priority Weights and Order of Relative Importance for QoS attributes (MFB)

QoS Attributes	Priority Weight	Importance
Availability	0.12993	3
Response Time	0.12967	4
Reliability	0.53100	1
Cost	0.20939	2

Table 4.12: Priority Weights and Order of Relative Importance for QoS attributes (ODS)

QoS Attributes	Priority Weight	Importance
Availability	0.0950	2
Response Time	0.0950	2
Reliability	0.0950	2
Cost	0.7152	1

II. QoS Analyser

Applying the concept of fuzzy decision making discussed in Section 2.4, QoS values were synthesised from users' fuzzy estimations by finding the element with the highest membership function from the intersection set of the fuzzy sets selected by users to denote their desired QoS aspirations. Table 4.13 and Table 4.14 show how QoS aspirations were synthesised from representing the fuzzy sets for MFB and ODS respectively.

Table 4.13: Synthesised QoS Aspiration for Microfinance Bank

QoS Attribute	Linguistic Term	Linguistic Hedges	Synthesised QoS Values
Availability	Very High	In the Vicinity of 98%	98.49%
Response Time	Low	Very close to 400ms	489.46ms
Reliability	Very High	In the Vicinity of 75%	75.43%
Cost	Premium	In the Vicinity of 400$	390.64$/Month

Table 4.14: Synthesised QoS Aspiration for Online Drug Store

QoS Attribute	Linguistic Term	Linguistic Hedges	Synthesised QoS Values
Availability	High	In the Vicinity of 90%	97.12%
Response Time	Acceptable	In the Vicinity of 600ms	559.35ms
Reliability	High	Very close to 70%	72.1%
Cost	Cheap	In the vicinity of 250$	290.16$/Month

Table 4.15: Completely elicited QoS requirements of MFB and ODS

QoS Attributes	Microfinance Bank		Online Drug Store	
	QoS Weight	QoS Values	QoS Weight	QoS Values
Availability	0.1242	98.49	0.0950	97.12
Response Time	0.1237	489.46	0.0950	559.35
Reliability	0.5798	75.43	0.0950	72.1
Cost	0.1724	390.64	0.7152	290.16

4.4.5 QoS-based Ranking of Service Alternatives

I. QoS Requirements Optimizer

Table 4.15 shows a summary of priority weights and QoS values obtained from the users. These inputs are fine-tuned according to the values of the QoS attributes of available services in the service directory. Optimized QoS requirement is obtained by finding those QoS values that are the most ideal, and closest to user's requirements. The FOCUSS framework utilises an SAW-based and exponential Euclidean distance function (eEUD) described in Section 2.4 for this purpose, by optimising the fuzzy goals very close to both the most ideal QoS values, and user's requirements. For this case study, each service alternative is evaluated with respect to user's weight of importance using SAW function, and the similarity of each service QoS attributes to a combination of user's preference weights and aspiration values are performed with the eEUD function. Using MOEA framework, the optimal QoS values that satisfy both the fuzzy goal and constraint are obtained as being very close to the service alternatives with the best performance and closest to user requirements. Table 4.16 shows a comparison of the initial QoS requirements and the final QoS requirements with respect to user's priority weights and QoS aspiration.

Table 4.16: Comparison of Initial QoS Requirements and Optimised QoS values

QoS Attributes	Microfinance Bank			Online Drug Store		
	Initial Requirements		Optimized Requirements	Initial Requirements		Optimized Requirements
	Weight	*Values*		*Weight*	*Values*	
Availability	0.1242	98.49	98.5	0.0950	97.12	97
Response Time	0.1237	489.46	489.5	0.0950	559.35	559
Reliability	0.5798	75.43	75.4	0.0950	72.1	72
Cost	0.1724	390.64	390.6	0.7152	290.16	290.2

II. Service QoS Ranking

Having obtained the optimised QoS requirements, the final stage is to rank the services in the service directory using these requirements. This is performed using flat memory technique in case retrieval to find the k-nearest neighbours using the eEUD function to the optimised requirements as shown in Table 4.16. Table 4.17 and Table 4.18 show the 10 most suitable CRM services with QoS values that match the optimised requirements MFB and ODS respectively.

Table 4.17: Top ten Services that match optimal requirements for MFB

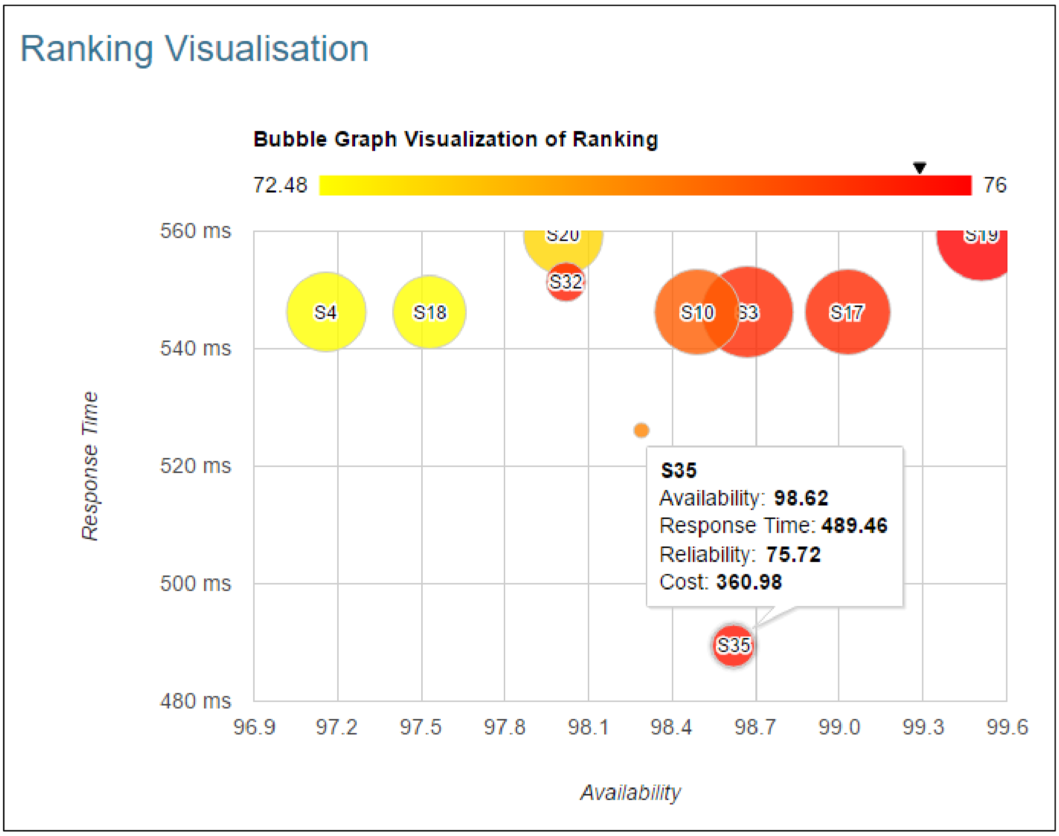
SERVICE RANK	SERVICE_ID	AVAILABILITY (%)	RESPONSE TIME(MS)	RELIABILITY (%)	COST ($/MONTH)
1	S3	98.67	546.24	75.43	390.64
2	S17	99.03	546.24	75.43	386.15
3	S10	98.49	546.24	74.72	385.64
4	S35	98.62	489.46	75.72	360.98
5	S19	99.51	559.35	76	390.48
6	S4	97.16	546.24	72.48	381.15
7	S18	97.53	546.24	72.48	376.66
8	S20	98.01	559.35	73.04	380.99
9	S7	98.29	526.12	74.19	354.14
10	S32	98.02	551.35	75.62	360.46

Table 4.18: Top ten Service Alternatives to Optimal Requirements for ODS

SERVICE RANK	SERVICE_ID	AVAILABILITY (%)	RESPONSE TIME(MS)	RELIABILITY (%)	COST ($/MONTH)
1	S22	97.15	648.77	71.53	290.16
2	S6	96.79	648.77	71.53	294.65
3	S11	98.11	648.77	73.77	299.14
4	S21	98.66	648.77	74.48	299.65
5	S24	97.63	661.88	72.1	294.49
6	S5	98.29	648.77	74.48	304.14
7	S23	99.14	661.88	75.05	303.98
8	S14	97.53	668.89	72.78	326.66
9	S2	97.16	668.89	72.78	331.15
10	S16	98.01	682	73.34	330.99

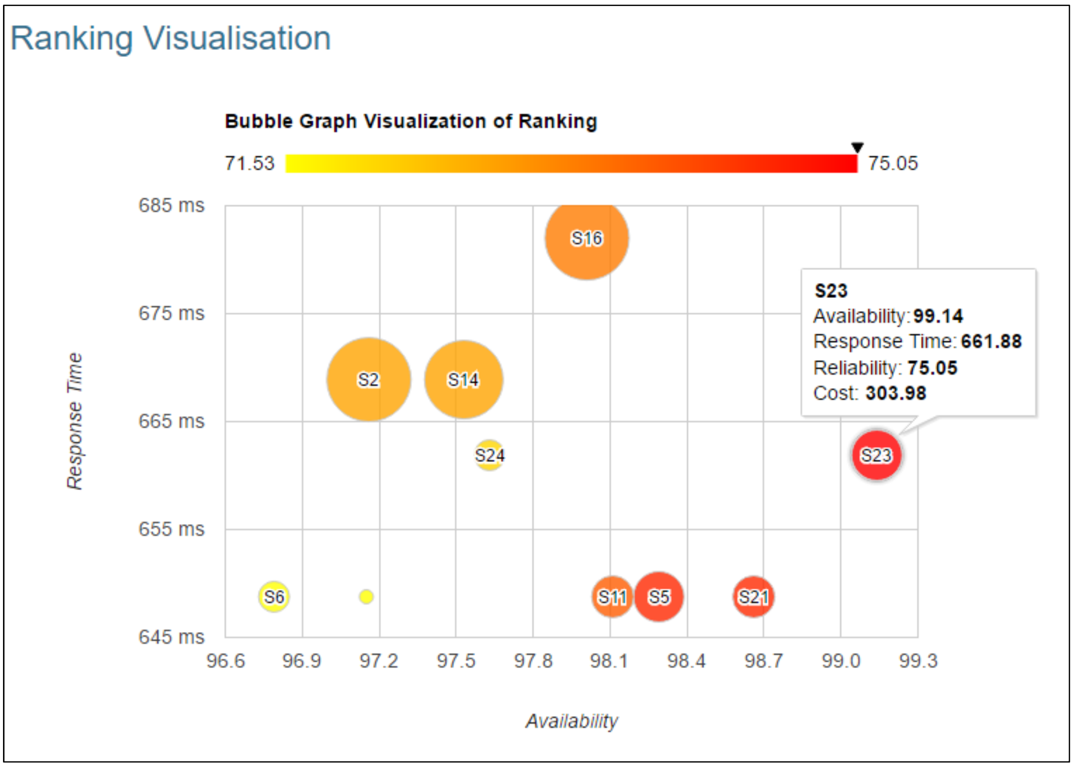
III. Visualising the Ranking

To enable further analysis, the results shown in Table 4.17 and Table 4.18 are then visualised using a bubble chart, from which the user can explore the relationships among the ranked alternatives. The MFB or ODS can then select the most satisfactory service that best satisfies their respective requirements. Figure 4.6 and Figure 4.7 shows the bubble graph for data contained in Table 4.17 and Table 4.18 respectively. Figure 4.8 and Figure 4.9 shows the complete GUI for QoS requirements elicitation and the tabular and bubble graph visualisation.


Ranking Visualisation

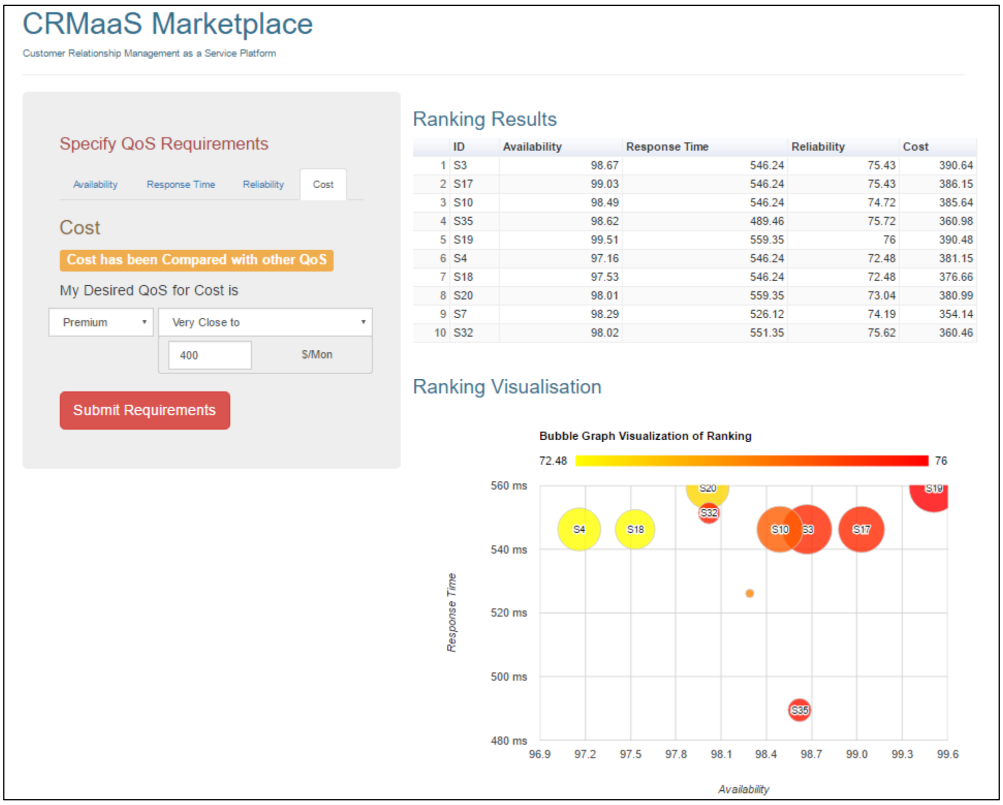
S35
Availability: **98.62**
Response Time: **489.46**
Reliability: **75.72**
Cost: **360.98**

Figure 4.6: Bubble Graph for Ranked Services for MFB Requirements
On mouse over, the details for Service_ID 35 is shown.


Ranking Visualisation

S23
Availability: **99.14**
Response Time: **661.88**
Reliability: **75.05**
Cost: **303.98**

Figure 4.7: Bubble Graph for Ranked Services for ODS Requirements
One mouse hover, the details of the Service_ID 23 is shown



CRMaaS Marketplace

Customer Relationship Management as a Service Platform

Specify QoS Requirements

Cost

Cost has been Compared with other QoS

My Desired QoS for Cost is

Premium ▾ Very Close to ▾

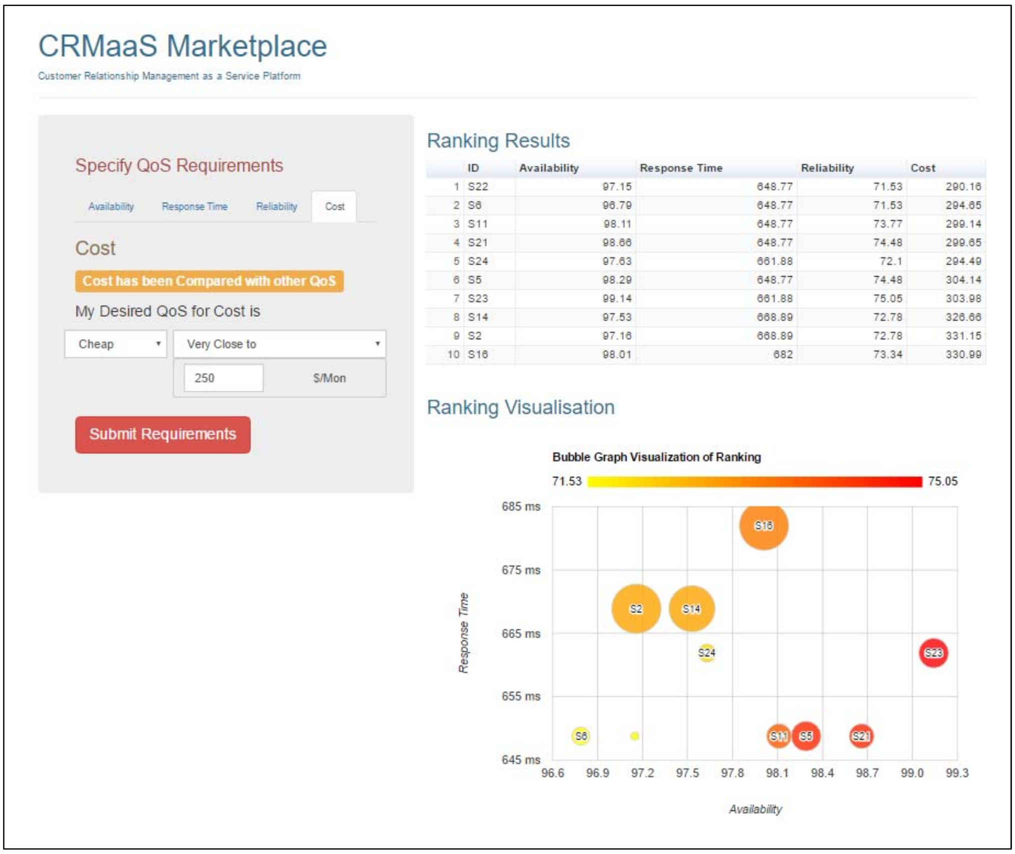
400 $/Mon

Ranking Results

	ID	Availability	Response Time	Reliability	Cost
1	S3	98.67	546.24	75.43	390.64
2	S17	99.03	546.24	75.43	386.15
3	S10	98.49	546.24	74.72	385.64
4	S35	98.62	489.46	75.72	360.98
5	S19	99.51	559.35	76	390.48
6	S4	97.16	546.24	72.48	381.15
7	S18	97.53	546.24	72.48	376.66
8	S20	98.01	559.35	73.04	380.99
9	S7	98.29	526.12	74.19	354.14
10	S32	98.02	551.35	75.62	360.46

Ranking Visualisation

Figure 4.8: Complete GUI Showing Requirements, Table and Bubble Graph (MFB)



CRMaaS Marketplace

Customer Relationship Management as a Service Platform

Specify QoS Requirements

Cost

Cost has been Compared with other QoS

My Desired QoS for Cost is

Cheap ▾ Very Close to ▾

250 $/Mon

Ranking Results

	ID	Availability	Response Time	Reliability	Cost
1	S22	97.15	648.77	71.53	290.16
2	S6	96.79	648.77	71.53	294.65
3	S11	98.11	648.77	73.77	299.14
4	S21	98.66	648.77	74.48	299.65
5	S24	97.63	661.88	72.1	294.49
6	S5	98.29	648.77	74.48	304.14
7	S23	99.14	661.88	75.05	303.98
8	S14	97.53	668.89	72.78	326.66
9	S2	97.16	668.89	72.78	331.15
10	S16	98.01	682	73.34	330.99

Ranking Visualisation

Figure 4.9: Complete GUI Showing QoS Requirements, Table and Bubble Graph (ODS)

4.5 CHAPTER SUMMARY

This chapter contains a demonstration of the utility of the proposed FOCUSS framework, by identifying the tool support base to realise the framework. The framework was further validated via an illustrative case study of a Customer Relationship Management as a Service (CRMaaS) e-marketplace that comprises the coming together of various atomic services to realise a cloud ecosystem of CRM services. The demonstration of how these services are combined was carried out following a structured organisation with respect to constraints guiding their combination. Also demonstrated is how the users' (a microfinance bank and an online drug store) requirements would be elicited and how the framework would rank available alternative which is then presented to the users through a bubble graph visualisation. The validation performed shows that the FOCUSS framework is a viable approach for cloud service ranking and selection in cloud service e-marketplace. In the next chapter, the empirical evaluation of the FOCUSS framework is presented.

CHAPTER FIVE

EVALUATION

5.1 INTRODUCTION

The previous chapter contains details of the prototype implementation as a proof of concept to validate the proposed FOCUSS framework with an illustrative case study. This chapter contains a quantitative and qualitative evaluation of the FOCUSS framework. In the following sections, the methodology employed for evaluating the FOCUSS framework is discussed, which include the experimental designs for FOCUSS evaluation broken down into design parameters and test cases, as well as the analysis and discussion of results. Specifically, this chapter presents the evaluation procedures employed to validate the scalability, accuracy, and user experience of the FOCUSS framework using descriptive and inferential statistics on data obtained from three experiments. The results of the evaluation reported in this chapter serve as a justification of the performance of the FOCUSS framework in line with the aim and objectives set forth in this thesis. This chapter concludes with the summary of major themes discussed therein.

5.2 PERFORMANCE AND USABILITY EVALUATION

Having validated that the FOCUSS framework can be used to bring a variety of atomic services together to form an ecosystem, from which valid combinations are determined, the FOCUSS framework is evaluated with respect to its QoS-ranking performance and efficiency, as well as user experience. A major aim of an evaluation is to show via experimentations the performance differences when approaches or systems are compared to each other with respect to some given factors. Moreover, it is also needful to understand the factors that contribute to the differences in performance. In order to determine the differences in performance, the trends of two or more test collections of reasonable size are observed for consistency across the different data; after which statistical significance test is performed to validate the fact that any observable difference is not due to chance.

There are three empirical evaluation methods in software engineering; they include case studies, surveys, and experiments (Wohlin *et al.*, 2012). Experiments are more beneficial in that they can be employed to answer specific questions by setting up a direct comparison between the treatments of interest. In experiments, the biases and errors in comparison are minimised, and the ability to control the factors enables stronger inferences to be made about the difference in the results, providing a better basis to make stronger inferences about causation (Oehlert, 2010). A design of the experiment or experimental design is defined as a series of trials in which a number of individual experimental units and responses are measured, which can be analysed to quantify and compare the effects of the treatments, with which a cause-effect inference can be established (Oehlert, 2010).

The evaluation of the FOCUSS framework is carried out via three experiments; comprising two simulation-based evaluations and a user study (see Figure 5.1). Using simulation experiments, the FOCUSS framework was evaluated for computational efficiency (scalability), as well as QoS-based ranking accuracy, whereas user studies were carried out to access the user experience dimension of the FOCUSS framework. Efficiency measures scalability of the FOCUSS QoS-ranking mechanism by considering how the number of available functionally equivalent service alternatives in the service directory affects execution time for producing top-k ranked results. Accuracy measures the degree to which the FOCUSS framework ranks available service alternatives according to the QoS requirements of a user, as measured against a well-known benchmark. The user studies were carried out to estimate the ease of use and degree of the user experience of the FOCUSS framework. After the data generated from the experiments were collected, inferential statistical tests were performed to analyse the results of the experiments for statistical significance. The performance evaluation experiments are presented in subsequent sections.

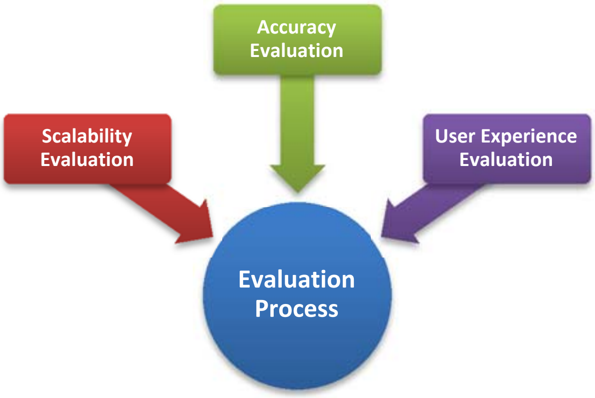


Figure 5.1: Evaluation Process for the FOCUSS framework

5.3 EXPERIMENT-1: SCALABILITY EVALUATION

5.3.1 Experiment Goal and Hypothesis

The main goal of experiment-1 is to determine the computational efficiency of the FOCUSS QoS-based ranking module. In order to achieve this, a simulation was undertaken to determine the scalability of the FOCUSS QoS-based ranking module by varying the number of services alternatives and measuring the execution time to produce a top-k rank of services. Hence, the null hypothesis is stated as follows:

> **H_0**: *The performance in terms of the execution time of the FOCUSS framework in producing a rank of top-k services scales linearly with increase in service alternatives.*

5.3.2 Experiment Dataset

Since there are no publicly available cloud services dataset, the QoS values of web services from a publicly available real-world datasets, the QWS dataset (Al-Masri and Mahmoud, 2007), was adopted instead; web services shares many similarities with cloud services (Sun *et al.*, 2014) and the QWS dataset has been used in similar studies involving cloud services, For example, (He *et al.*, 2012; Jahani *et al.*, 2014). The QWS dataset comprises QoS information for 2,507 web services resulting from the evaluation of one user with the measurements of nine QoS attributes. The nine QoS attributes of the QWS dataset include response time, availability, and throughput, the likelihood of success, reliability, compliance, best practices, latency, and documentation. For the purpose of this experiment, the information of three QoS attributes (availability, response time and reliability) was selected from the QWS dataset, and since the QWS dataset did not contain values for cost, uniformly distributed values for cost was randomly generated in the interval 10 to 500; the randomly generated values correspond to values between \$10 - \$500 per/month as cost of the services. To simplify the scope of experiment-1, 4 QoS attributes (Availability, Response time, Reliability and Cost) were considered, as the case in similar studies, for example (Zhao *et al.*, 2014; Ye *et al.*, 2011; Ludwig, 2012).

5.3.3 Simulation Parameters and Protocol

The scalability of the QoS-based ranking mechanism of the FOCUSS framework is measured by the execution time in milliseconds. The execution time is the time it takes to produce top-k services as the number of service alternatives increases. The number of services alternatives (n) was increased from 50 to 1000 based on the QoS dataset outlined in the previous section.

To achieve variation in the QoS data and the number of cloud services used in this experiments, the first 50 services was selected as the first case, then the next 100, then the next 350, and then next 750, then the next 1000. In all, a total of 2150 services (50+100+350+750+1000), together with their QoS information (including cost), were taken from the 2507 services contained the QWS dataset. The descriptive statistics (minimum value, maximum value, mean and standard deviation) of QoS information for 4 QoS attributes of the test datasets (n=50, 100, 350, 750 and 1000 services) are shown in Table 5.1, Table 5.2, Table 5.3, Table 5.4, and Table 5.5.

Table 5.1: Descriptive Statistics for Dataset, n=50

QoS Attribute	Min	Max	Mean	Std. Deviation
Availability (%)	18.00	100.00	79.18	18.71
Response time (ms)	49.43	3321.40	328.39	519.55
Reliability (%)	53.00	83.00	68.92	8.10
Cost ($/month)	111.63	496.01	289.90	126.03

Table 5.2: Descriptive Statistics for Dataset, n=100

QoS Attribute	Min	Max	Mean	Std. Deviation
Availability (%)	23.00	100.00	78.59	19.12
Response time (ms)	42.50	4207.50	436.02	651.13
Reliability (%)	42.00	83.00	69.71	8.32
Cost ($/month)	100.28	498.21	323.74	112.53

Table 5.3: Descriptive Statistics for Dataset, n=350

QoS Attribute	Min	Max	Mean	Std. Deviation
Availability (%)	9.00	100.00	82.05	17.567
Response time (ms)	42.50	4637.61	419.42	624.70
Reliability (%)	42.00	89.00	69.93	8.32
Cost ($/month)	100.79	497.86	301.64	115.82

Table 5.4: Descriptive Statistics for Dataset, n=750

QoS Attribute	Min	Max	Mean	Std. Deviation
Availability (%)	8.00	100.00	80.55	18.98
Response time (ms)	40.00	4758.00	390.03	596.04
Reliability (%)	33.00	89.00	70.14	8.70
Cost ($/month)	103.20	499.54	293.91	115.58

Table 5.5: Descriptive Statistics for Dataset, n=1000

QoS Attribute	Min	Max	Mean	Std. Deviation
Availability (%)	7.00	100.00	81.70	18.57
Response time (ms)	37.00	4989.67	370.94	531.43
Reliability (%)	33.00	89.00	69.32	8.80
Cost ($/month)	101.50	499.90	299.39	119.50

To manage the scope of the experiment, the value of k was fixed at 20, and equal distribution for priority weights are assumed, such that the weight for each QoS attribute is equal to $1/q$ (where q is the number of QoS criteria been evaluated); the value of q is equal to 4 (Availability, Response time, Reliability and Cost).

The simulation experiment was conducted by running the FOCUSS QoS-based ranking algorithm 30 times against a set of QoS requirements, and computing the average execution time (in milliseconds) it took to produce a ranking of top-20 ($k = 20$) services in the dataset with respect to the QoS requirements. The QoS-based ranking algorithm of the FOCUSS framework was implemented with Java programming language in NetBeans 8.1 IDE. The simulation experiments was conducted on Lenovo PC running Windows 10 Home single language edition with the following specifications: Intel Pentium CPU N3540 at 2.16GHz 2.16GHz processor and 4.00GB RAM on 64-bit Operating System, x64-based processor.

The summary of the parameters for simulation experiments is presented in Table 5.6.

Table 5.6: Summary of Parameters for Simulation Experiment-1

Metric	Execution Time (in milliseconds)
Top–k (k)	20
Number of QoS attributes (q)	4
Number of Alternatives (*n*)	50, 100, 350, 750, and 1000
Priority Weight (*w*)	1/q (corresponding to [0.25, 0.25,0.25,0.25])
Number of trial runs (*t*)	30

5.3.4 Results and Analysis

The results for simulation experiment are summarised in the descriptive statistics contained in Table 5.7 and depicted by the line graph in Figure 5.2. A simple linear regression was performed to determine the relationship between the numbers of service alternatives (n) and mean execution time, and also to test for the statistical significance of the scalability of the FOCUSS QoS-based ranking module as the number of service alternative increases. The simple linear regression was used to test the null hypothesis defined in Section 5.3.

Table 5.7: Execution Time for Ranking Top-20 Services vs. Number of Services

#Alternatives	Range(ms)	Min(ms)	Max(ms)	Mean(ms)	Std. Deviation(ms)
50	79.00	312.00	391.00	336.87	19.80
100	87.00	312.00	399.00	340.47	23.98
350	126.00	312.00	438.00	342.80	27.86
750	94.00	312.00	406.00	344.63	22.83
1000	78.00	328.00	406.00	349.43	19.40

As shown in line graph in Figure 5.2, the trendline shows a linear relationship between the number of alternatives and the mean execution time. The regression equation and statistics are given as follows: $y = 2.928x + 334.06$, $R^2 = .967$, $F(1,3) = 119.085$, $p < .05$; (where y = mean execution time, and x = number of service alternatives).

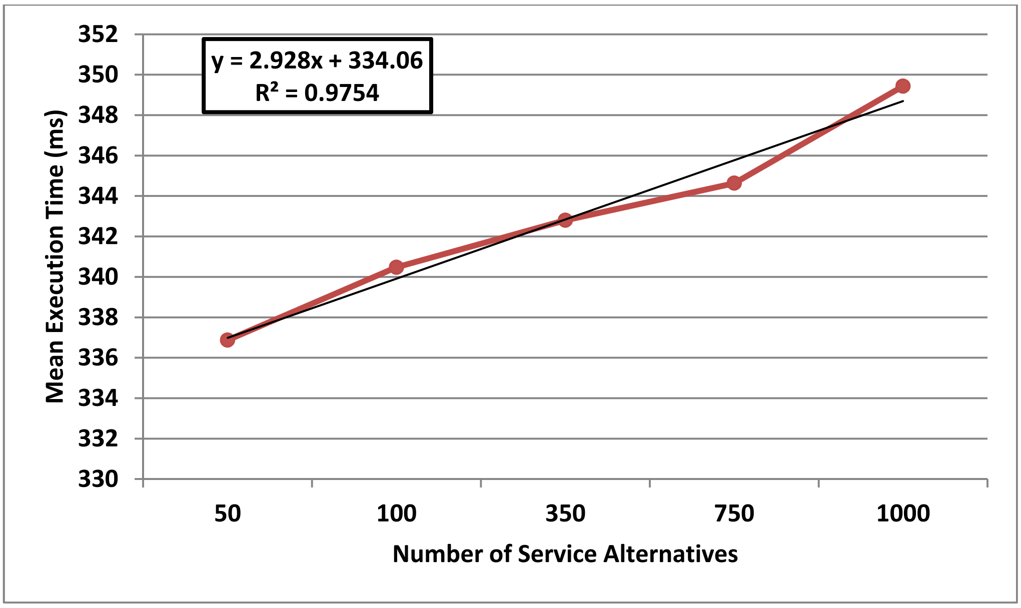


Figure 5.2: Average Execution Time to Rank Services vs. Number of Services

5.3.5 Discussion

The adjusted R^2 value from the regression analysis is $R^2 = 0.967$; and connotes that 96.7% of the variation in the time required to produce the top-20 rank is significantly explained by the number of service alternatives available. Consequently, since the p-value ($p = .002$) is less than the alpha value ($p < .05$), the indication is that the QoS-based ranking mechanism of the FOCUSS framework is timely efficient and linearly scalable as can also be observed from Figure 5.2. On the basis of this, the null hypothesis (**H$_0$**) is accepted that the performance in terms of execution time of the FOCUSS framework in producing a rank of top-k services scales linearly with increase in service alternatives.

5.4 EXPERIMENT-2: RANKING ACCURACY EVALUATION

The main approach employed for the design of Experiment-2 is comparative. It is important that such experiment is planned so that data is collected to enable comparison between the FOCUSS framework and other methods. This was achieved by first establishing the metrics, on the basis of which these methods are compared. The data generated from applying these metrics were then used to determine which method(s) performs better or comparable. The design and execution of experiment 2 is carried out in the following stages: 1) statement of the goal of the experiment; 2) Statistical design; 3) Data collection; 4) Data validation; 5) Data analysis; 6) Experiment execution, and 7) Interpretation of results. In the next sections, the application of each stage to the design, execution and analysis of experimental results conducted in this study are presented.

5.4.1 Experiment Goal and Hypothesis

The main goal of the experiments was to find out the effect of fuzzy-based QoS requirements, as implemented in the FOCUSS framework, on both the QoS-based ranking accuracy compared to other methods that accept numeric QoS requirements. In other words, the experiments aim to find out whether a QoS-based ranking method $M_1(q)$ that accepts fuzzy QoS requirements, i.e. q = linguistic query as inputs, performs considerably well as compared to a QoS-based ranking method $M_2(q)$ that accepts exact numeric QoS requirements, i.e. q=numeric query as inputs, on cloud service datasets of varying sizes. The design of the experiments involves the following important outcomes:

i. Determining the impact of QoS requirement input type (linguistic or numeric) on the accuracy of QoS-based ranking results, thus justifying the proposal of applying linguistic descriptors to approximate numerical QoS requirements.

ii. Determining the impact of the number of top-k ranked services in the set $[3, 5, 10, 15, 20]$ on the ranking accuracy and performance ranking performance. Ranking order is important as most users would usually consider the top k results (Mirmotalebi *et al.*, 2012).

iii. Determining the effect of the number of service alternatives in the set $[50, 100, 350, 750, 1000]$ on the ranking accuracy.

iv. Deciding, whether method $M_2(numeric)$ is better than method $M_1(linguistic)$ for QoS-based ranking of cloud services in cloud service e-marketplace context.

The goal of this experiment is to compare the ranking accuracy of the FOCUSS framework against other QoS-based ranking techniques using TOPSIS as the benchmark; based on the null hypothesis:

> **H_0:** *There will be no significant difference between the ranking performances of a method that accepts exact numeric values as QoS requirement and those that use linguistic descriptors to approximate values for QoS requirements.*

5.4.2 Experiment Dataset

The same adapted QoS dataset described in Experiemnt-1 that contains QoS information on Availability, Response time, Reliability and Cost for 2150 services and the 5 grouping of the 2150 services into sets of 50, 100, 350, 750 and 1000 services were also used in Experiment-2. Test cases for user's QoS requirements were generated following the conceptualization of QoS requirements as a collection of QoS preference and aspiration. The user's QoS aspiration was randomly generated following uniform distributions from intervals with lower and upper bounds corresponding to the worst and best QoS values respectively, of each of the 5 datasets of services collected derived from the QWS dataset.

For example, since the first set contained 50 services, the maximum and minimum values for each QoS attributes were identified and these values formed the basis for generating 5 random QoS requirements (Queries) for hypothetical users. While details of the datasets and corresponding user queries are contained in Appendix B, Table 5.8 shows the descriptive summary of the dataset, n=50 and the five (5) QoS requirements randomly generated for it (i.e. dataset, n=50), denoted as Query1 to Query5.

Table 5.8: Minimum Values, Maximum Values and Five Test Queries for Dataset (n=50)

	Availability	Response Time	Reliability	Cost
Min	18	49.43	53	111.63
Max	100	3321.4	83	496.01
Query1	24.66	492.69	62.1	197.92
Query2	90.79	1608.38	59.64	341.7
Query3	46.99	377.46	61.34	160.98
Query4	96.74	1279.35	71.9	466.13
Query5	60.17	346.89	74.89	152.97

5.4.3 QoS-based Ranking Methods Evaluated

The experimental units of this experiment are the methods whose ranking accuracies are compared. The units to which the treatments are applied include TOPSIS, weighted distance (WD) (Rehman *et al.*, 2011), Exponential Weighted Distance (eWD) (Rehman *et al.*, 2011). The justification for selecting these methods is that they are closest to the ranking principle underlying FOCUSS, in that they all considered both user's aspiration and priority of QoS attributes in the ranking of cloud services and can be applied to a large collection of service alternatives. Of these methods, the TOPSIS method was selected as a baseline for comparison. Apart from the fact that it was used in similar studies such as Sun *et al.* (2014) and Chamodrakas *et al.* (2011), the rationale for selecting TOPSIS as the benchmark is premised on the similarity of its fundamental principle to that of the FOCUSS framework. In TOPSIS, the best alternative has the shortest Euclidean distance from the ideal solution, at the same time farthest from the worst solution; this is very similar to the underlying principle behind the FOCUSS framework and the methods with which it is compared.

The FOCUSS framework utilises an exponential Euclidean distance metrics that estimate the proximity of all alternatives to the optimised QoS requirements derived from user's QoS requirements. The optimised QoS requirements are determined by those QoS values closest to the most optimal solution in the collection while maintaining closeness to initial user's requirements. In order to make the comparison suitable, the original TOPSIS fundamental notion of what constitutes the ideal solution was modified and set to the user's requirements. This is reasonable as an ideal solution to a user's requirement would be those alternatives with values closest to the user's requirements, and farthest from the worst solution.

The eWD and WD methods (Rehman *et al.*, 2011) both compute the similarity between two vectors representing user requirements criteria and each service's QoS properties, and the best service is one whose QoS properties best match user requirements. The WD approach is a sum of the weighted difference between the QoS values specified by user and service's QoS properties. The similarity for each service alternative compared with user's requirements is computed using:

$$Sim_{wd}(UserReq, Ser) = \sum_{i=1}^{n} w_i * (UserReqVect_i - ServiceDesVect_i) \quad \textbf{(5.1)}$$

Where n is the number QoS attributes. The similarity values of all the service alternatives are then sorted and the lower the better.

The similarity values used to rank services using the eWD is computed using the exponential weighted difference between QoS vectors of user requirements and service alternatives, the formula is as follows:

$$Sim_{ewd}(UserReq, Ser) = \sum_{i=1}^{n} e^{-w_i*(UserReqVect_i - ServiceDesVect_i)} \quad \textbf{(5.2)}$$

Two versions of FOCUSS ranking algorithms were implemented (FOCUSS_lin and FOCUSS_num). FOCUSS_lin is the original FOCUSS method that uses fuzzy linguistic descriptor to represent QoS requirements from users, while FOCUSS_num, following the same ranking principle of the original FOCUSS method, utilises numeric QoS values. Similarly, versions of eWD and WD to process queries expressed using fuzzy linguistic descriptors were also considered. Consequently, the six methods involved in the simulation experiments are listed in Table 5.9.

Table 5.9: Methods Evaluated in Experiment-2

QoS Version	Method	Method ID
Linguistic	Exponential Weighted Difference Metric	eWD_lin
	FOCUSS	FOCUSS_lin
	Weighted Difference Metric	WD_lin
Numeric	Exponential Weighted Difference Metric	eWD_num
	FOCUSS	FOCUSS_num
	Weighted Difference Metric	WD_num

5.4.4 Evaluation Metrics

A number of measures from the domain of information retrieval for measuring the ranking performance of ranking algorithms were identified and used to evaluate the performance of the FOCUSS framework compared to other QoS-based ranking methods discussed earlier, using TOPSIS as the benchmark. These metrics have been used for evaluation QoS-based ranking in similar studies, for example, (Qu and Buyya, 2014; Sun *et al.*, 2014; Qu *et al.*, 2014; Mirmotalebi *et al.*, 2012). Precision and recall are popular retrieval evaluation metrics in Information Retrieval, but cannot be applied in this evaluation because they are single-value metrics based on the whole list of service alternatives relevant to a QoS requirement (query) and do not consider the order or ranking of those services in the retrieved list. However, metrics such as Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP), Spearman Rank Coefficient (SRC) and Kendall Tau Rank Coefficient (KRC) are applicable to measure the ranking performance of QoS-based ranking algorithms. Unlike in Zanakis *et al.* (1998), only the rank-order produced by the methods are being evaluated by the metrics not the values or rating underscoring the rankings.

I. Normalized Discount Cumulative Gain (NDCG)

Since the value of top-k ranked service varies in this experiment, the ranking performance of the QoS-based techniques compared is measured by normalising the cumulative gain at each top-k position for each query (or user QoS requirement). This is achieved by sorting list of services by relevance, producing the maximum possible Discount Cumulative Gain (DCG) till position k, also referred to as the Ideal DCG (IDCG) till that position. Normalized Discount Cumulative Gain (NDCG) at positions corresponding to value of top-k is applied to measure whether the FOCUSS framework can still rank most satisfactory services at the top. The relevance scores (rel_i) used in computing the NDCG are performance values obtained by the TOPSIS method in response to a query. For a query, the normalized discounted cumulative gain, or NDCG, is computed mathematically as:

$$NDCG_k = \frac{DCG_k}{IDCG_k} \tag{5.3}$$

And

$$DCG_k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2(i)} \tag{5.4}$$

While $IDCG_k$ corresponds to the ideal DCG at position k.

II. Mean Average Precision (MAP)

As earlier stated the precision and recall metric is best applicable considering the whole list of relevant services to a query. In order to measure ranking performance for a ranked sequence of services, the precision and recall at every position in the ranked sequence of services are computed to plot a precision-recall curve. The precision-recall curve is created by plotting precision $p(r)$ as a function of recall r. The Average Precision (AveP) is computed as the average value of $p(r)$ over the interval from $r = 0$ to $r = 1$, such that:

$$\boldsymbol{AveP} = \frac{\sum_{i=1}^{k}(\boldsymbol{P(i)} \times \boldsymbol{rel(i)})}{\boldsymbol{k}} \tag{5.5}$$

Where i is the rank in the sequence of the return services, k is the number of top-k services returned; $P(i)$ is the precision at the rank i in the list, given as:

$$P(i) = \frac{\#relevant\ service\ retrieved\ @\ i}{\#\ relevant\ services\ @\ i} \tag{5.6}$$

And $rel(i)$ is an indicator function, such that compared to ranking produced by the TOPSIS method, $rel(i) = 1$ if the service at rank i is a relevant service, and $rel(i) = 0$ otherwise.

The MAP is the average of precision values at the ranks where there are relevant services to the user QoS query. The mean is obtained by averaging over several queries. The total number of queries used in this experiment is 5. Therefore, Mean Average Precision is defined by:

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{5.7}$$

Where Q is the number of queries; for this experiment, the value of Q is 5 according to the experiment design (cf. Section 5.4).

III. Spearman Rank Correlation Coefficient

Spearman's Rank Correlation (SRC) coefficient, also known as, Spearman's rho is used to measure the rank correlation between two variables, by using a monotonic function to describe the relationship between those variables. A perfect correlation QoS-based ranking of service alternatives produced by two methods has a Spearman correlation of +1, while the Spearman correlation of -1 when the ranking is extremely dissimilar.

Given a list of top-k service alternatives, produced by method M_1, and M_2, the list of top-k are converted to ranks $rg\ M_1$, and $rg\ M_2$; therefore, Spearman rho, ρ, is computed as:

$$\rho = 1 - \frac{6\sum d_i^2}{k(k^2-1)} \tag{5.8}$$

Where $d_i$is the difference between the ranks computed as $d_i = rg(M_1) - rg(M_2)$.

IV. Kendall Tau Coefficient

Kendall Rank Correlation (KRC) coefficient is also known as Kendall's tau coefficient and is denoted as $\boldsymbol{\tau}$ is used to measure the ordinal association between two variables. The Kendall correlation between two variables will be high when the top-k list produced by two methods has a correlation value of 1, and low with a correction value of -1. Any pair of observation between the top-k items produced by two methods, M_1 and M_2 are concordant, if the position of an item produced by $M_1$is in the same position of that item in the list produced method M_2, and discordant otherwise. The Kendall tau coefficient is computed as follows:

$$\tau = \frac{(C-D)}{\frac{k(k-1)}{2}} \tag{5.9}$$

Where *C* = Concordant pairs; *D* = Discordant pairs; *k* is the number of top-k items produced by the methods.

The metrics were implemented as Java methods in NetBeans, while the indicators for metrics used to evaluate the methods are as follows: a perfect agreement between a QoS-based method and TOPSIS, in terms of top-k items produced, would be signalled by the following: NDCG=1, MAP=1, SRC=1, and KRC=1.

5.4.5 Experiment Design

The evaluation methodology employed in the study is based on a simulation modelling technique similar to Chamodrakas *et al.* (2011). Simulation is a widely-used research method to study and analyse complex scenarios and to gain insights into performance and scalability for large-size problem instances. Moreover, it helps to evaluate the generalizability of the results. Experimental designs indicate how to vary the settings of the factors or independent variables to see if and how they impact on the response variable or dependent variable (Sanchez, 2005). For the experiments conducted in this study, a factorial design was selected as a suitable design for the simulation experiments (Sanchez, 2005).

Factorial designs are represented more concisely as a^k where k is the number of factors under investigation at a levels with a total of a^k design points. Also, factorial designs can be written such that different set of factors are investigated at different number of levels. As a case in point, a design with two factors, with 2 and 3 levels for each factor will be written as 2×3 design (also called crossed design). Every column in the design matrix corresponds to a factor, and the entries within the column correspond to settings or treatment for this factor. However, each row also represents a particular combination of factor levels, and which is referred to as a design point. Repeating the whole design matrix is called the replication of the design, and given n design points and b replications, the total number of tests becomes $N_{test} = n * b$.

Some benefits of a factorial design include 1) Ability to examine all possible combinations of factors levels for each of the factors, which is useful in identifying important interaction effect. 2) They are also orthogonal designs, such that the pairwise correlation between any two factors is equal to zero. Experiment-2 follows a 3-way factorial design and was inspired by the works of Chamodrakas *et al.* (2011) and Zanakis *et al.* (1998).

5.4.6 Simulation Parameters and Protocol

The response variable for Experiment-2 is the ranking performance in terms of accuracy, measured by four accuracy metrics (NDCG, MAP, SRC and KRC). The factors considered are- the number of top-k ranked services (top-k), the number of service alternatives (alternatives) and the QoS requirements input type (query). There are six factor levels for the top-k results corresponding to k= 3, 5, 7, 10, 15, and 20; while there are also five factor levels for alternatives- [50, 100, 350, 750, 1000]. The input types are either numeric or linguistic, corresponding to two factors. Equal distribution for priority weights are assumed, such that the weight for each QoS attribute is equal to $1/q$ (where q is the number of QoS criteria been evaluated); the value of q is equal to 4 (Availability, Response time, Reliability and Cost). For each combination, the trials were performed five times using the five QoS requirements shown in the Appendix B, after which the average for each combination case was taken. In all, the total number of solutions generated is equivalent to: 5 QoS queries × each combination, which comprise 5 levels for alternative × 6 levels for top-k × 6 levels for methods × 4 evaluation metrics = 3600 solutions. The average of the 3600 solutions produced 720 data points which are then analysed using the Kruskal-Wallis test, the non-parametric equivalent of the Analysis of variance (ANOVA) method. Table 5.10 shows the summary of the responses, factors and factor levels for Experiment-2.

Table 5.10: Summary of Experiment Variables, levels, methods, and metrics

#Service Alternatives (n)	**Top-k (k)**	**QoS Preference weight (w)**	**Methods to be compared* (m)**	**Evaluation Metrics (e)**	**#QoS Attributes (q)**	**#Queries per Trial runs (t)**
50 100 350 750 1000	3 5 7 10 15 20	Uniform Distribution (1/q)	FOCUSS_lin FOCUSS_num eWD_lin eWD_num WD_lin WD_num	NDCG MAP Kendall Tau Spearman rho	4	5
*TOPSIS is the Benchmark method used for comparison Total solutions = n × k × m × e × t = 5 × 6 × 6 × 4 × 5 = **3600 solutions**						

The protocol followed in Experiment-2 is outlined below:

i. The methods were implemented with Java programming language in NetBeans 8.1 IDE. The simulation experiments were conducted on the same PC specification in section 5.3.3.

ii. The first step in each approach was to normalise the decision matrixes (comprising datasets, *n = 50, 100, 350, 750, 1000*) using vector normalisation so as to keep the values within [0, 1].

iii. Five QoS requirements were generated for which each method generated a ranking of cloud services from the decision matrix. The queries were also normalised using vector normalisation method.

iv. Each design point was repeated 5 times with each QoS query generated in point (ii) above.

v. Each evaluation metric (*m*) was applied to measure the accuracy performance of each method on the basis of each trial run.

vi. The value from the evaluation metrics was recorded in an Excel worksheet (See Appendix B)

vii. A total of 3600 data points were collected (720 data items per QoS query).

viii. The average values from all metrics for all methods, resulting in 720 data point, were analysed for significance and meaningfulness using Kruskal-Wallis test in SPSS software package.

5.4.7 Results and Analysis

The descriptive analyses of the results are presented in the next section, while the results are tested for statistical significance using the non-parametric equivalent of ANOVA (Kruskal-Wallis test), along with the relevant post hoc analysis tests.

I. Descriptive Statistical Analysis

The mean and median ranking accuracy produced by the four metrics employed in this simulation experiments for all six methods are contained in Table 5.11.

Table 5.11: Median and Mean Ranking Accuracy for Methods by Metrics

Methods	Median Accuracy				Mean Accuracy			
	NDCG	MAP	SRC	KRC	NDCG	MAP	SRC	KRC
eWD_num	0.941406	0.837057	0.362688	0.324128	0.935858	0.812477	0.362688	0.324128
eWD_lin	0.939955	0.806438	0.398779	0.356178	0.930129	0.772053	0.398779	0.356178
FOCUSS_num	0.982181	0.866667	0.67503	0.64148	0.96989	0.854511	0.67503	0.64148
FOCUSS_lin	0.981735	0.866667	0.691848	0.659023	0.966899	0.863404	0.691848	0.659023
WD_num	0.544211	0.623264	-0.11192	-0.09716	0.561568	0.657504	-0.11192	-0.09716
WD_lin	0.553056	0.554167	-0.10259	-0.09436	0.566099	0.634281	-0.10259	-0.09436

Meanwhile, Figure 5.3 and Figure 5.4 show that the FOCUSS_lin is closer to TOPSIS than the other five methods. The only exception is the NDCG result for FOCUSS_num. The next closer method to TOPSIS for all other metrics is FOCUSS_num. The vital discovery from the results of the experiment is that both versions of the FOCUSS ranking algorithms (FOCUSS_num and FOCUSS_lin) produced better ranking results than other approaches compared, and clearly outperforms other methods, particularly for the SRC and KRC metrics. The next best set of methods is eWD, and outperforms WD methods; eWD_lin produced results closer to TOPSIS than eWD_num only in SRC and KRC, while eWD_num is better with NDCG and MAP. However, it can also be observed that the two versions of eWD produced better results than versions of WD; WD_num produced worse results in all metrics than WD_lin, except for MAP where the results for WD_num is than WD_lin.

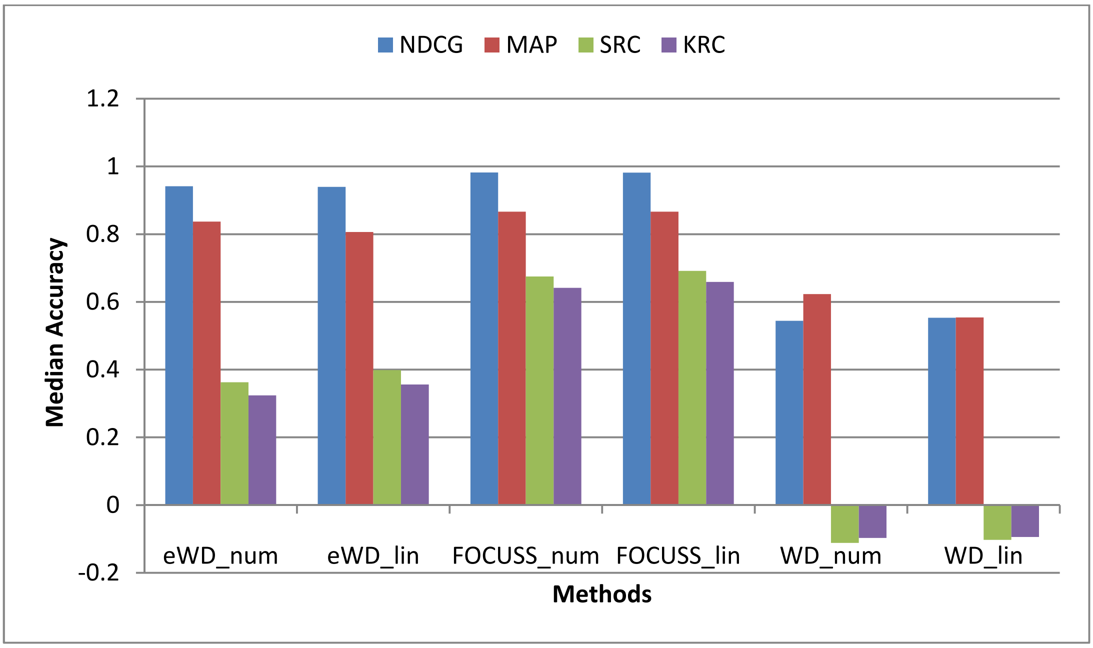


Figure 5.3: Median Ranking Accuracy for all Six Methods by each Metric

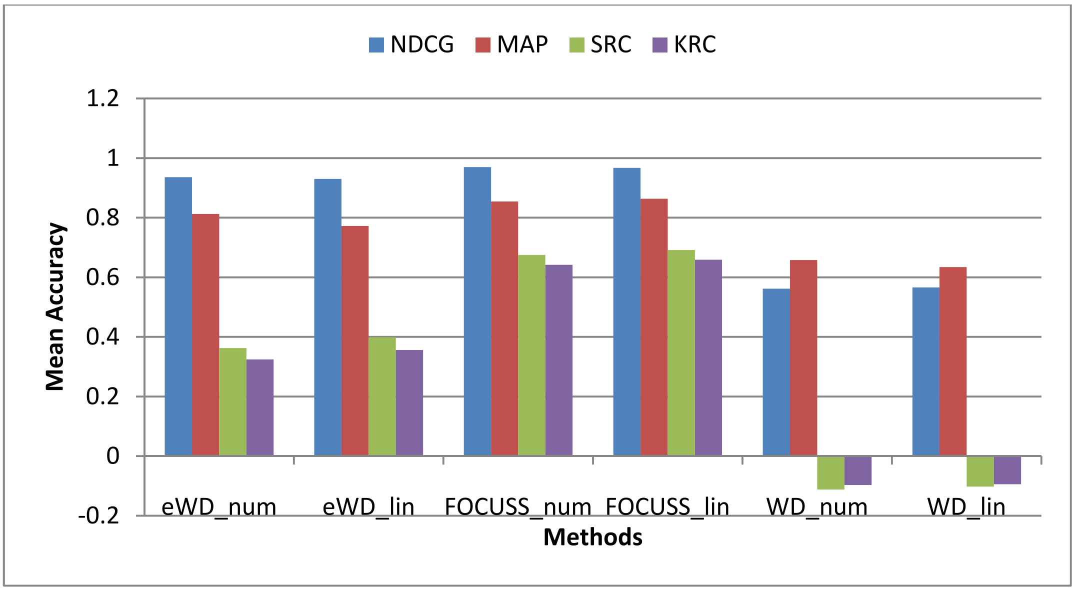


Figure 5.4: Mean Ranking Accuracy for all Six Methods by each Metric

The QoS input type did not considerably affect the ranking accuracy, with only marginally differences (less than 0.02) in the median accuracy scores across the evaluation metrics as shown in Table 5.12 and depicted in Figure 5.5.

Table 5.12: Median Accuracy based on QoS Input Type (Linguistic and Numeric)

Input-Type	NCDG	MAP	SRC	KRC
Numeric	0.935964	0.828724	0.340000	0.303810
Linguistic	0.927880	0.798562	0.316429	0.261203

Apart from the analysis of the descriptive statistics, there is still the need to further determine the significance of the results using inferential statistics which is presented in the next section.

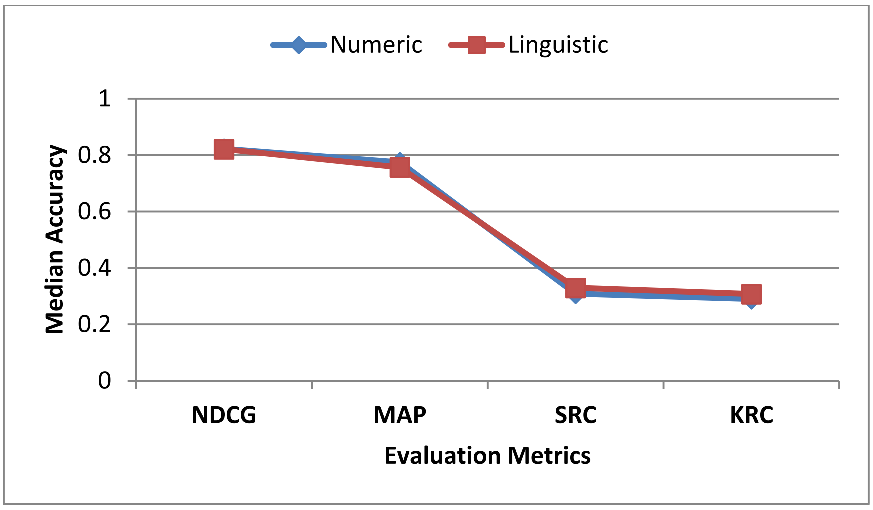


Figure 5.5: Median Accuracy for Numeric and Linguistic QoS Requirements

II. Inferential Statistical Analysis

The initial consideration was to use parametric ANOVA for analysis of results, and preliminary tests were conducted to ensure that the underlying assumption for ANOVA was not violated. The test included checks for normality, linearity, univariate, homogeneity of variance-covariance matrices, and it was observed that these assumptions were violated. The violations were due to the inherent random structure of the simulation experiments. Therefore, the non-parametric alternative, the Kruskal-Wallis test, was used instead as the statistical procedure to investigate the ranking accuracy of the various methods compared in the simulation experiments. Non-parametric tests do not make assumptions about the normality of the distribution of variables.

The four dependent variables (the accuracy metrics) are NDCG, MAP, SRC, and KRC, while the five independent variables are: methods (method), the number of alternatives (size), the number of top-k results (top-k) and QoS input type (input_type). The statistical computations were performed using the SPSS statistical application package. The Kruskal-Wallis test allows for the comparison of the scores on some continuous variable for three or more groups; after the scores have been converted to ranks, the mean rank for each group is then compared. The significance level chosen in the analysis is 95% (α = 0.05) as a standard benchmark; therefore, p-value < α is considered statistically significant. Mann-Whitney U tests on pairwise statistical comparisons were performed as a post hoc follow-up tests to identify the method(s) that are statistically significantly different from the others.

a) Kruskal-Wallis Test

According to the non-parametric Kruskal-Wallis test, only the grouping variable, method, showed significant difference across all accuracy metrics used. The results of other grouping variables (the number of alternatives, the top-k results obtained and the QoS input type) did not show a significant difference in all accuracy metrics, except for MAP (see Table 5.13). From Table 5.13, it is obvious that there is no significant difference in the ranking performance produced by the metrics for all methods with numeric QoS inputs and those with linguistic inputs; for the metrics, $p > 0.05$. In addition, the number of alternatives (size) did not affect significantly the ranking accuracy obtained from the metrics, neither did the number of top-k ranked services; as the p-values for size (NDCG, p=.06; SRC, p=.056; KRC, p=.084) and top-k (NDCG, p=.142; SRC, p=.991; KRC, p=.987) are greater than 0.05; except for MAP in both cases, where the p-values < 0.05 for both variables size and top-k.

Table 5.13: Summary of Kruskal-Wallis Test on Ranking Accuracy

	NDCG			MAP			SRC			KRC		
	χ^2	*df*	**Sig.**	χ^2	*df*	**Sig.**	χ^2	*df*	**Sig.**	χ^2	*df*	**Sig.**
Method	128.89	5	**.000**	30.784	5	**.000**	127.114	5	**.000**	129.818	5	**.000**
Size	9.051	4	**.060**	49.036	4	**.000**	9.228	4	**.056**	8.207	4	**.084**
Top-k	8.269	5	**.142**	46.807	5	**.000**	.520	5	**.991**	.624	5	**.987**
Input_Type	.003	1	**.958**	.571	1	**.450**	.273	1	**.602**	.178	1	**.673**

The grouping variable, *method*, confirms that there is a statistically significant difference in the accuracy performance of the six methods when compared, NDCG [χ2 (5, N=180) =

128.89, p < 0.05]; MAP [χ2 (5, N=180) = 30.78, p < 0.05]; SRC [χ2 (5, N=180) = 127.11, p < 0.05]; SRC [χ2 (5, N=180) = 129.82, p < 0.05]. An examination of the mean ranks was done as an indication of the ranking accuracy of the methods compared. Higher mean rank suggests better accuracy and vice versa. FOCUSS_num method recorded a higher mean rank (M=137.02) on NDCG than the other five methods, closely followed by FOCUSS_lin (M=136.85). The method with the lowest mean rank is WD_num (M=30.75). The mean ranks for other methods includes: eWD_lin (M=103.52), eWD_num (M=103.12), and WD_lin (M=31.75).

Similarly, an inspection of the mean ranks for MAP, SRC and KRC reveals that FOCUSS_lin had the highest mean (M=114.88; M=140.27; M=141.08 respectively), and closely followed by FOCUSS_num (M=112.45; M=136.90; M=138.45 respectively).Table 5.14 summarises the mean ranks of the six methods along four metrics. Overall, FOCUSS_lin outperformed other methods on all metrics, except for FOCUSS_num, that performed better than FOCUSS_lin on the NDCG metric.

Table 5.14: Mean Ranks for each Accuracy Metrics for all methods

METHODS	MEAN RANKS			
	NDCG	MAP	SRC	KRC
eWD_num	103.12	101.47	97.73	96.95
eWD_lin	103.52	88.8	102.57	101.47
FOCUSS_num	**137.02**	112.45	136.9	138.45
FOCUSS_lin	136.85	**114.88**	**140.27**	**141.08**
WD_num	30.75	66.37	30.78	31.18
WD_lin	31.75	59.03	34.75	33.87

b) Mann-Whitney U Tests

Although, the Table 5.14 tells us that the methods differed according to the accuracy metrics used, but does not reveal how the methods differed. Pairwise comparisons of the methods were carried out using the Mann-Whitney test. Meanwhile, both versions of WD evidently performed worse than versions of eWD and FOCUSS, and there is no significant difference in the QoS input types, therefore, the pairwise comparisons were limited to versions of eWD and FOCUSS. More specifically, the following five pairs were considered for a follow-up test, and they include FOCUSS_lin Vs. eWD_lin; FOCUSS_lin Vs. eWD_num; FOCUSS_num Vs. eWD_lin; FOCUSS_num Vs. eWD_num; FOCUSS_lin Vs. FOCUSS_num. The summary of the Mann-Whitney U follow-up tests are presented in Table 5.15, Table 5.16, Table 5.17, Table 5.18, and Table 5.19.

i- FOCUSS_lin Vs. eWD_lin

According to the Mann-Whitney U test for pairwise comparison between FOCUSS_lin and eWD_lin, the U-statistics (cf. Table 5.15) revealed that there is statistically significant difference between ranking accuracy on all metrics for the FOCUSS_lin method and the eWD_lin method based on the following U-statistics ($U = 212.00\ [Z = -3.521], p < 0.01$), produced by the NDCG metrics. Statistical significance in difference was also recorded for SRC ($U = 195.000\ [Z = -3.765], p < 0.01$); and KRC ($U = 182.000\ [Z = -3.965], p < 0.01$), but the MAP metrics did not produce statistical significance ($U = 301.500\ (Z = -2.201), p > 0.01$).

The descriptive statistics showed that FOCUSS_lin (NDCG mean rank = 38.43; MAP mean rank=35.45; SRC mean rank = 38.98; KRC mean rank = 39.43) scored higher on NDCG, MAP, SRC and KRC respectively than eWD_lin (NDCG mean rank = 22.57; MAP mean rank=25.55; SRC mean rank = 22.02; KRC mean rank = 21.57); with a difference of between 9 to 17 points. Furthermore, the difference between accuracy of FOCUSS_lin and eWD_lin method was somewhat large on all accuracy metrics used: NDCG (r = -0.45); SRC (r = -0.49); KRC (r = -0.51), except for MAP, with medium effect in the difference (r = -0.28).

Table 5.15: Mann-Whitney Test Results (FOCUSS_lin Vs eWD_lin)

Metric	Method	N	Mean Rank	U	Z	p-value	Sig (0.01)
NDCG	eWD_lin	30	22.57	212.000	-3.521	0.000	Significant
	FOCUSS_lin	30	38.43				
	Total	**60**					
MAP	eWD_lin	30	25.55	301.500	-2.201	0.028	Significant
	FOCUSS_lin	30	35.45				
	Total	**60**					
SRC	eWD_lin	30	22.02	195.500	-3.765	0.000	Significant
	FOCUSS_lin	30	38.98				
	Total	**60**					
KRC	eWD_lin	30	21.57	182.000	-3.965	0.000	Significant
	FOCUSS_lin	30	39.43				
	Total	**60**					

ii- FOCUSS_lin Vs. eWD_num

The pairwise comparison between FOCUSS_lin and eWD_num methods (cf. Table 5.16) revealed statistical significant difference between both methods on NDCG ($U = 202.000\ [Z = -3.669], p < .05$), SRC ($U = 153.500\ [Z = -4.386], p < .05$) and KRC ($U = 141.000\ [Z = -4.571], p < .05$) and MAP ($U = 364.000\ [Z = -1.277]$, $p < .05$). However, the mean rank results showed that the accuracy of FOCUSS_lin is higher than of eWD_num for all metrics: NDCG (Mean rank =38.77 Vs. 22.23), MAP (Mean rank = 33.37 Vs. 27.63), SRC (Mean rank = 40.38 Vs. 20.62) and KRC (Mean rank = 40.80 Vs. 20.20).

Table 5.16: Mann-Whitney Test Results (FOCUSS_lin Vs eWD_num)

Metric	Method	N	Mean Rank	U	Z	p-value	Sig (0.01)
NDCG	eWD_num	30	22.23	202.000	-3.669	0.000	Significant
	FOCUSS_lin	30	38.77				
	Total	**60**					
MAP	eWD_lin	30	27.63	364.000	-1.277	0.202	Insignificant
	FOCUSS_num	30	33.37				
	Total	**60**					
SRC	eWD_lin	30	20.62	153.500	-4.386	0.000	Significant
	FOCUSS_num	30	40.38				
	Total	**60**					
KRC	eWD_lin	30	20.20	141.000	-4.571	0.000	Significant
	FOCUSS_num	30	40.80				
	Total	**60**					

iii- FOCUSS_num Vs. eWD_lin

The pairwise comparison between FOCUSS_num and eWD_lin methods (see Table 5.17) showed statistical significant difference between FOCUSS_num and eWD_lin methods on NDCG ($U = 204.000\ [Z = -3.637], p < 0.01$), SRC ($U = 212.000\ [Z = -3.519], p < 0.01$) and KRC (U=184.000 [Z=-3.927], p < 0.01), except for MAP ($U = 308.500\ [Z = -2.095], p = 0.036$). Besides, the mean rank descriptive statistics showed that the accuracy of FOCUSS_num is higher than of eWD_lin for all four metrics: NDCG (Mean rank = 38.70 Vs. 22.30), MAP (Mean rank = 35.22 Vs. 25.78), SRC (Mean rank = 38.43 Vs. 22.65) and KRC (Mean rank = 39.35 Vs. 21.65). The effect size is as follows is large for NDCG (r= -0.47), SRC (r = -0.45), KRC (r= -0.51), and medium effect for MAP (r = -0.27).

Table 5.17: Mann-Whitney Test Results (FOCUSS_num Vs eWD_lin)

Metric	Method	N	Mean Rank	U	Z	p-value	Sig (0.01)
NDCG	eWD_lin	30	22.30	204.000	-3.637	0.000	Significant
	FOCUSS_num	30	38.70				
	Total	**60**					
MAP	eWD_lin	30	25.78	308.500	-2.095	0.036	Significant
	FOCUSS_num	30	35.22				
	Total	**60**					
SRC	eWD_lin	30	22.57	212.000	-3.519	0.000	Significant
	FOCUSS_num	30	38.43				
	Total	**60**					
KRC	eWD_lin	30	21.65	184.000	-3.927	0.000	Significant
	FOCUSS_num	30	39.35				
	Total	**60**					

iv- FOCUSS_num Vs. eWD_num

The pairwise comparison between FOCUSS_num and eWD_num methods (see Table 5.18) suggested that there is statistical significant difference between the accuracy of both methods judging from all metrics. The U statistics includes: NDCG (U=188.000 [Z=-3.874], p < 0.01), MAP (U=387.000 [Z= -0.934], p =.350), SRC (U=152.000 [Z=-4.406], p < 0.01) and KRC (U=134.000 [Z=-4.666], p < 0.01). Furthermore, the mean ranks for both methods showed that the accuracy of FOCUSS_num is higher than of eWD_num for all four metrics: NDCG (Mean rank = 39.23 Vs. 21.77), MAP (Mean rank = 32.60 Vs. 28.40), SRC (Mean rank = 40.43 Vs. 20.57) and KRC (Mean rank = 41.02 Vs. 19.98), to a large effect (r = -0.5; r = -0.57; r = -0.6,) for NDCG, SRC and KRC respectively, and small effect for MAP (r = -0.12).

Table 5.18: Mann-Whitney Test Results (FOCUSS_num Vs eWD_num)

Metric	Method	N	Mean Rank	U	Z	p-value	Sig (0.01)
NDCG	eWD_num	30	21.77	188.000	-3.874	0.000	Significant
	FOCUSS_lin	30	39.23				
	Total	**60**					
MAP	eWD_lin	30	28.40	387.000	-0.934	0.350	Insignificant
	FOCUSS_num	30	32.60				
	Total	**60**					
SRC	eWD_lin	30	20.57	152.500	-4.406	0.000	Significant
	FOCUSS_num	30	40.43				
	Total	**60**					
KRC	eWD_lin	30	19.98	134.500	-4.666	0.000	Significant
	FOCUSS_num	30	41.02				
	Total	**60**					

v- FOCUSS_lin Vs. FOCUSS_num

The pairwise comparison between FOCUSS_lin and FOCUSS_num methods (see Table 5.19) suggested that there is no statistical significant difference between the accuracy of both methods. The U statistics includes: NDCG (U=188.000 [Z=-3.874], p > 0.01), MAP (U=387.000 [Z= -0.934], p > 0.01), SRC (U=152.000 [Z=-4.406], p > 0.01) and KRC (U=134.000 [Z=-4.666], p > 0.01).

Table 5.19: Mann-Whitney Test Results (FOCUSS_lin Vs FOCUSS_num)

Metric	Method	N	Mean Rank	U	Z	p-value	Sig (0.01)
NDCG	FOCUSS_num	30	30.08	437.500	-0.185	0.853	Insignificant
	FOCUSS_lin	30	30.92				
	Total	**60**					
MAP	FOCUSS_num	30	29.97	434.000	-0.238	0.812	Insignificant
	FOCUSS_lin	30	31.03				
	Total	**60**					
SRC	FOCUSS_num	30	29.03	406.000	-0.652	0.515	Insignificant
	FOCUSS_lin	30	31.97				
	Total	**60**					
KRC	FOCUSS_num	30	29.08	407.500	-0.630	0.529	Insignificant
	FOCUSS_lin	30	31.92				
	Total	**60**					

In addition, the mean ranks of the accuracy of both versions of FOCUSS methods showed just marginal difference, as FOCUSS_lin was slightly higher than FOCUSS_num by a maximum of one point across all metrics: NDCG (Mean rank = 30.92 Vs. 30.08), MAP (Mean rank = 31.03 Vs. 29.97), SRC (Mean rank = 31.97 Vs. 29.03) and KRC (Mean rank = 29.08 Vs. 29.08). The effect size is very low with r = -0.02, r = -0.03, r = -0.08, r=-0.08 for NDCG, MAP, SRC and KRC respectively.

5.4.8 Discussion

The Kruskal-Wallis test revealed that the number of alternatives, size and QoS input type did not affect the accuracy metrics considered, but the methods produced distinguishable accuracy results. Judging by the results from both the descriptive and inferential statistical analysis, the two versions of the FOCUSS methods (FOCUSS_num and FOCUSS_lin) produce better accuracy results on all four metrics considered and were in all cases closer to the benchmark metric (TOPSIS) than the other four methods compared in this experiments.

Furthermore, the significantly higher mean rank of the FOCUSS_lin methods indicates that FOCUSS_lin produces more accurate rankings than other methods. In addition, an important discovery from the experiments is that expressing QoS requirements using linguistic terms did not compromise the accuracy of the ranking method. This discovery is aligned with the results of Sun *et al.* (2014), proving that there is no significant difference in the rankings produced by methods that accept linguistic QoS requirements as input and those that accept numeric QoS requirements (see Figure 5.5).

Although, the versions of FOCUSS had higher accuracy compared to other methods evaluated, there exist only a marginal but insignificant difference between the ranking produced by FOCUSS_lin and FOCUSS_num (see Table 5.19). On the basis of the results obtained from experiment-2, the null hypothesis (H_0) in section 5.4 is hereby accepted. The hypothesis H_0 states that there will be no significant difference between the ranking performances of a method that accepts exact numeric values as QoS requirement and those that use linguistic descriptors to approximate values for QoS requirements.

5.5 EXPERIMENT-3: USER EXPERIENCE EVALUATION

Experiment-3 is a controlled user study designed to evaluate the user experience of the bubble graph visualisation integrated as part of the FOCUSS framework compared to traditional tabular format. The use of visualisation techniques is expected to reduce the cognitive load of the user by aiding the completion of user tasks accurately and time efficiently (Sebrechts *et al.*, 1999). The use of a controlled experiment is well suited for the answering how one visualisation format technique compares to another (Lam *et al.*, 2012). In this experiment, a “head-to-head” comparison was carried out on both visualisation formats (Lam *et al.*, 2012). The effectiveness of the visualisations was measured quantitatively using time and accuracy metrics, while subjective assessment of the visualisations was carried out by soliciting participants’ feedback via the use of a usability questionnaire.

5.5.1 Experiment Goal and Hypothesis

The objective of Experiment-3 is to determine the differences in quality of user experience for users exploring the list of top-k alternatives produced by QoS-based service ranking methods. It has been argued in this study that information visualisation is a viable means to explore top-k alternatives as this gives the user the flexibility of performing trade-off analysis much more easily compared to a traditional top-k list presented with text in a tabular format. The objects studied in this controlled experiment are bubble graph visualisation and textual-tabular visualisation of a list of top-k services. The purpose of the experiment is to evaluate the two visualisation formats in representing a list of ranked top-k services in the context of a cloud service e-marketplace, with respect to the quality of user experience of both formats from a researcher's viewpoint. The user experience is measured in terms of how quickly and accurately users identify their preferred alternatives.

To this end, the formulated null hypothesis to be rejected is as follows:

1. **H_0:** *There is no significant difference in task completion time of bubble graph visualisation and tabular visualisation of the list of top-k service alternatives.*
2. **H_0:** *There is no significant difference in perceived effectiveness, perceived efficiency, and perceived correctness of bubble graph visualisation and tabular visualisation of the list of top-k service alternatives.*

5.5.2 Experiment Instrumentation

Wohlin *et al.* (2012) identified mainly three types of instruments used for an experiment: objects, guidelines and measurement instruments; and these three instruments were utilised in Experiment-3. Next, each instrument is described in more details.

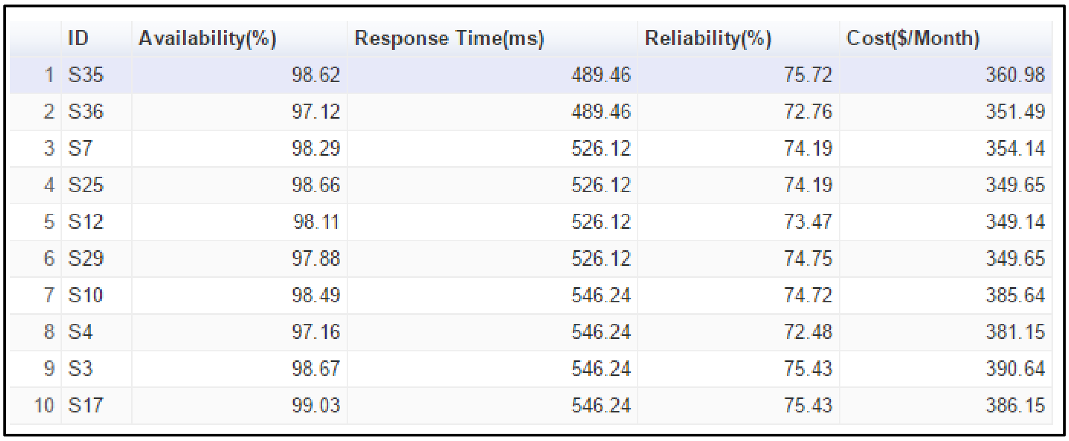
I. Object

The object instrument is the prototype implementations of two hypothetical CRMaaS e-marketplace with the result page implementing either bubble graph visualisation or Tabular visualisation. The hypothetical CRMaaS e-marketplace was accomplished in Java programming language, as a WAR file deployed on glassfish server and container running on the same PC configuration presented in Section 5.3.3.

The QoS requirements for four QoS attributes (Availability, Response Time, Reliability and Cost) presented in Table 5.20 produced the list of top-20 services that served as input data for both visualisation formats. These predefined QoS requirements were inputted via the UI component of the hypothetical CRMaaS e-marketplace prior to the commencement of the tasks by participants. Participants accessed the visualisation formats via two different tabs on a web browser. Figure 5.6 and Figure 5.7 show the bubble graph visualisation and the tabular listing of the top-10 alternatives respectively, based on the QoS requirements presented in Table 5.20.

Table 5.20: QoS requirements used in Experiment-3

QoS Attributes	Preference	Aspiration	
		Goal	Constraints
Availability	• Somewhat less important than response time • Somewhat More important than reliability • Extremely more important than cost	Very High	In the vicinity of 99%
Response Time	• Somewhat less important than reliability • Very more important than cost	Low	Very Close to 400
Reliability	Very less important than cost	High	In the vicinity of 70%
Cost	-	Premium	In the vicinity of 500$/Month



	ID	Availability(%)	Response Time(ms)	Reliability(%)	Cost($/Month)
1	S35	98.62	489.46	75.72	360.98
2	S36	97.12	489.46	72.76	351.49
3	S7	98.29	526.12	74.19	354.14
4	S25	98.66	526.12	74.19	349.65
5	S12	98.11	526.12	73.47	349.14
6	S29	97.88	526.12	74.75	349.65
7	S10	98.49	546.24	74.72	385.64
8	S4	97.16	546.24	72.48	381.15
9	S3	98.67	546.24	75.43	390.64
10	S17	99.03	546.24	75.43	386.15

Figure 5.6: Tabular listing of top-k services from Table 5.20 requirements

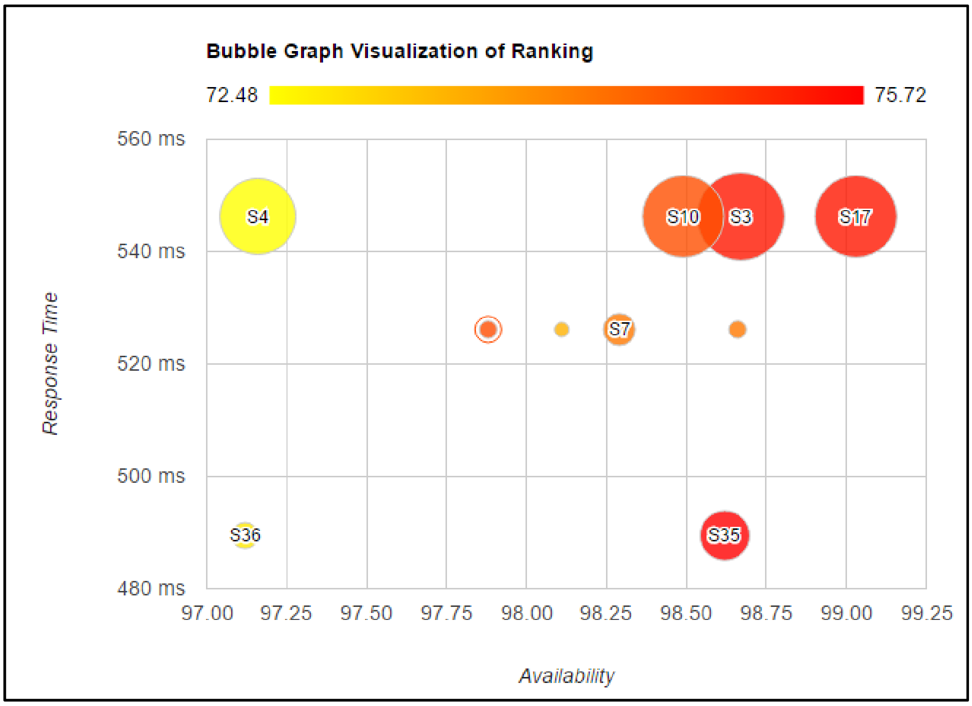
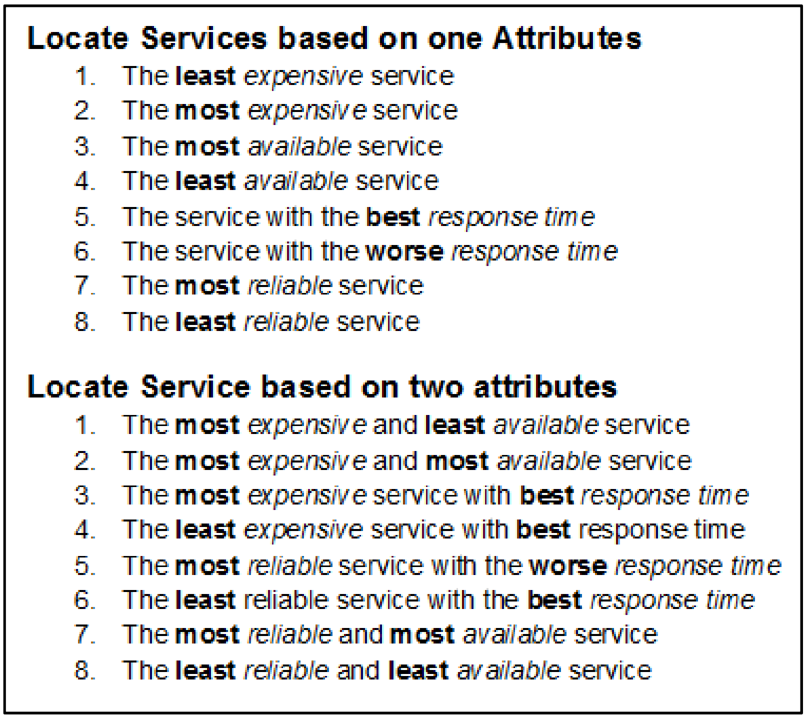


Figure 5.7: Bubble Graph Visualisation of Top-10 Services

II. Guidelines

Guidelines are used to provide guidance to each participant for the experiments, and it contains the descriptions and outline of specific tasks each participant is expected to complete. The tasks were based on the taxonomy of user's tasks proposed by Valiati (2005) and elaborated in Pillat *et al.* (2005).

Although, the taxonomy describes seven user tasks (locate, compare, configure, infer, determine, identify, and visualise), locate tasks were defined for this experiment, as they represent the decision-making scenarios in a cloud service e-marketplace. Locate tasks refers to finding specific information in the visualisation relating to data items, dimensions, properties, values etc. (Pillat *et al.*, 2005). The starting point of a *Locate* task is the participant exploring the visualisation and ends with the participants identifying the desired information (Pillat *et al.*, 2005). A total of sixteen (16) tasks were defined and documented in the guideline for this experiment (see Figure 5.8).



Locate Services based on one Attributes

1. The **least** *expensive* service
2. The **most** *expensive* service
3. The **most** *available* service
4. The **least** *available* service
5. The service with the **best** *response time*
6. The service with the **worse** *response time*
7. The **most** *reliable* service
8. The **least** *reliable* service

Locate Service based on two attributes

1. The **most** *expensive* and **least** *available* service
2. The **most** *expensive* and **most** *available* service
3. The **most** *expensive* service with **best** *response time*
4. The **least** *expensive* service with **best** response time
5. The **most** *reliable* service with the **worse** *response time*
6. The **least** reliable service with the **best** *response time*
7. The **most** *reliable* and **most** *available* service
8. The **least** *reliable* and **least** *available* service

Figure 5.8: List of 16 'Locate' User Tasks used in Experiment-3

III. Measurement

The measurement instrument is used to collect data from participants. Two measurement instruments were used in this experiment; they include the task performance survey (sample is in Figure 5.9) and a post-experiment questionnaire. The task performance survey instrument was employed to capture time to complete it, while the post-experiment questionnaire, a customization of the Post-Study-Satisfaction-User Questionnaire (PSSUQ), was used to elicit user experience.

PSSUQ (Lewis, 1992) is a popularly used instrument used in conducting usability studies in the literature, and it consists of 19 items, from which only 15 relevant questions were selected for this study. These 15 items were specifically adapted for evaluating participant's impression of the visualisation formats used in this context of this research. Participants were required to rate each item in the post-experiment questionnaire on a 7-point Likert scale according to the following scale (7-Excellent and 1-Poor). The sample questions of the modified PSSUQ for both Table and Bubble graph visualisation formats are presented in Figure 5.10, the complete instrument is contained in Appendix C.

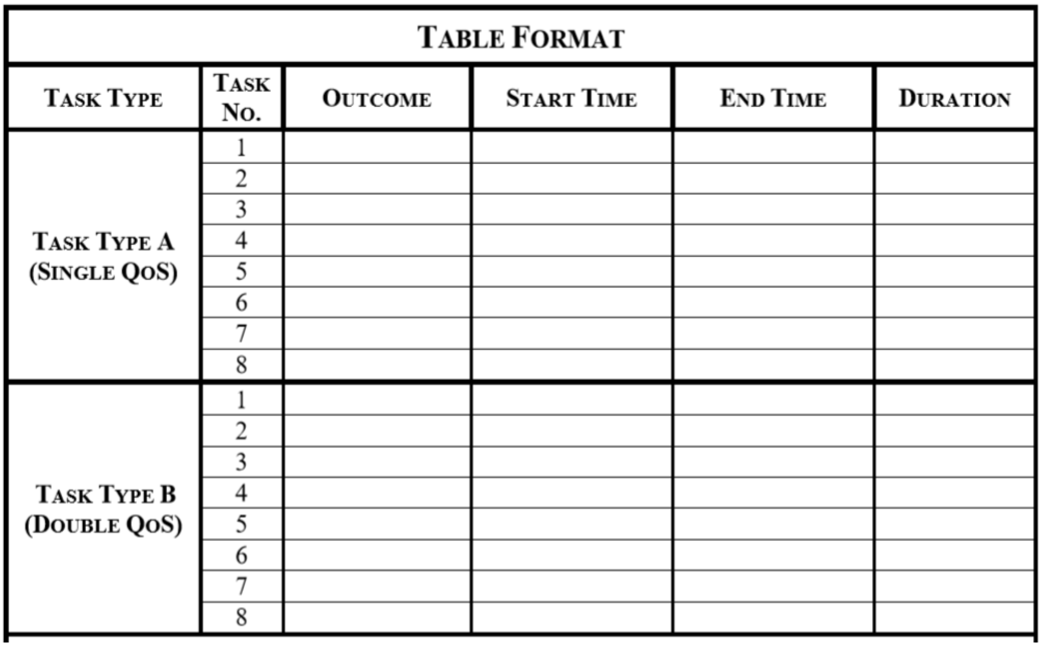
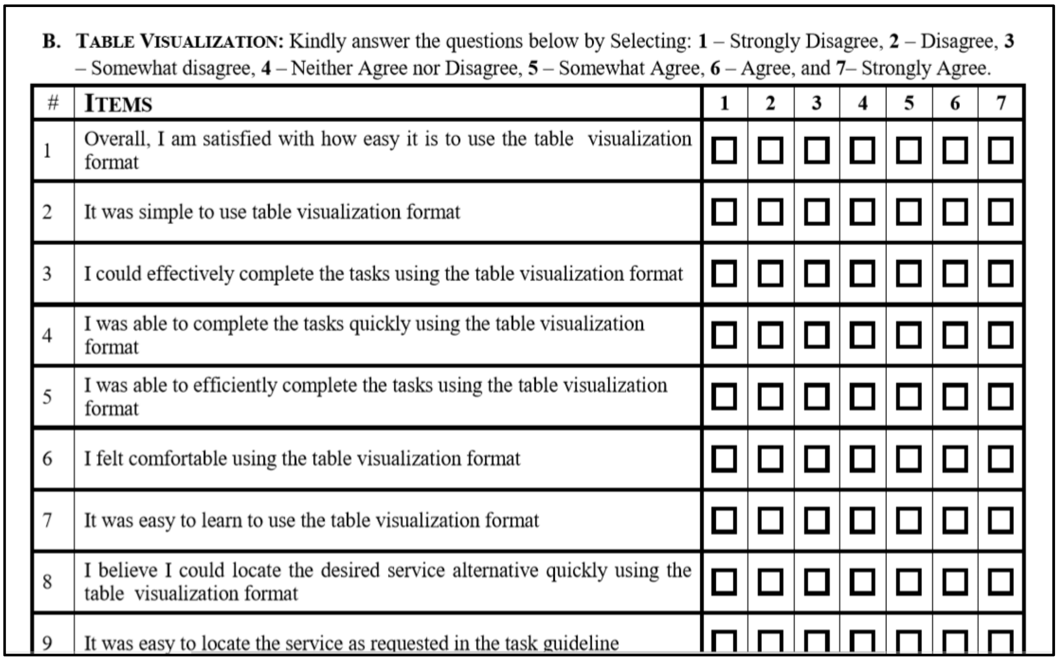


TABLE FORMAT

TASK TYPE	TASK NO.	OUTCOME	START TIME	END TIME	DURATION
TASK TYPE A (SINGLE QOS)	1				
	2				
	3				
	4				
	5				
	6				
	7				
	8				
TASK TYPE B (DOUBLE QOS)	1				
	2				
	3				
	4				
	5				
	6				
	7				
	8				

Figure 5.9: Task performance Survey Instrument



B. TABLE VISUALIZATION: Kindly answer the questions below by Selecting: **1** – Strongly Disagree, **2** – Disagree, **3** – Somewhat disagree, **4** – Neither Agree nor Disagree, **5** – Somewhat Agree, **6** – Agree, and **7**– Strongly Agree.

#	ITEMS	1	2	3	4	5	6	7
1	Overall, I am satisfied with how easy it is to use the table visualization format	□	□	□	□	□	□	□
2	It was simple to use table visualization format	□	□	□	□	□	□	□
3	I could effectively complete the tasks using the table visualization format	□	□	□	□	□	□	□
4	I was able to complete the tasks quickly using the table visualization format	□	□	□	□	□	□	□
5	I was able to efficiently complete the tasks using the table visualization format	□	□	□	□	□	□	□
6	I felt comfortable using the table visualization format	□	□	□	□	□	□	□
7	It was easy to learn to use the table visualization format	□	□	□	□	□	□	□
8	I believe I could locate the desired service alternative quickly using the table visualization format	□	□	□	□	□	□	□
9	It was easy to locate the service as requested in the task guideline	□	□	□	□	□	□	□

Figure 5.10: Sample of Modified Post-Study-Satisfaction-User-Questionnaire

5.5.3 Experiment Design and Protocol

The independent variable of the study is the visualisation format and it has two levels: Bubble Graph Visualisation and Tabular Visualization. The dependent variables are the speed of task completion and user satisfaction in performing defined tasks with the two visualisation formats. Task completion time was achieved by tracking the overall completion time in seconds, and the aggregated user satisfaction scores from participants' feedback using the modified PSSUQ instrument. The objects evaluated are the bubble graph and tabular visualisations of a list containing top 10 services based on the requirements shown in Table 5.20. Although all QoS dimensions are important, we considered four QoS dimensions, for the purpose of this study, to represent the attributes of the services: Availability, Response Time, Reliability and Cost. Figure 5.6 shows the tabular listing of 10 cloud services, their QoS attributes and corresponding bubble graph visualisation is shown in Figure 5.7.

The legend of the bubble graph is as follows: x-axis represents availability measured in percentage, y-axis the response time of the services in milliseconds, the colours of the bubble represent the reliability, and the darker colour signifies higher reliability. The cost is represented by the size of the bubble, as bigger bubbles signify higher cost.

The task guidelines contain the tasks that participants are expected to complete. The starting point of a locate task is the participant exploring the visualisation and ends with the participants identifying the desired information. Figure 5.8 shows the list of 16 tasks grouped into two categories (eight tasks in each category) to represent the subset of activities users undertake in a cloud service e-marketplace.

A completely randomised design was selected for experiment-3. The participants were randomly divided into two groups; the first group is assigned to use the bubble graph visualisation and then the tabular visualisation in that order, while the second group used the tabular visualisation and then the bubble graph visualisation in a reverse order to the first group. Experiment-3 can be described as one factor with two treatment type of experiment, to which a paired comparison design (or cross-over design) is applied; the same number of participants started with both visualisations formats to have a balanced design (Wohlin *et al.*, 2012; Oehlert, 2010). The experiment was run offline (i.e. not in a real cloud service e-marketplace context). Because the experiments involve multiple subjects (participants) and objects (bubble graph and tabular visualisation formats), it was designed as a blocked subject-object study (Wohlin *et al.*, 2012; Oehlert, 2010).

A total of 10 persons participated in the experiment, comprising 7 males and 3 females, ranging in age from 20 to 25. The participants were undergraduate students studying computer, engineering and mathematics-related courses. Participants were taken through a ten minutes tutorial session where the purpose and the process of the experiments were made known. Participants were given a tutorial on the use of both visualisation formats to complete sample tasks and allowed to complete some preliminary tasks to ascertain their ability to perform the main tasks defined for the experiments. As soon as participants were comfortable with the process, they were presented with copies of guideline containing tasks to be completed. The tasks involved using the bubble graph and tabular visualisations, and the task performance survey instrument. The tasks guideline outlined 16 tasks (see Figure 5.8) grouped into two categories (eight tasks in each category) to represent the subset of activities users undertake in a cloud service e-marketplace. The tasks were grouped according to levels of complexity ranging from locating services by both one to two QoS criteria. The experiment administrator recorded the time it took each participant to complete each task with the aid of a stopwatch. Upon completion of the tasks, a post-experiment survey was conducted in order to capture participants' impressions of both visualisation formats.

5.5.4 Results and Analysis

Quantitative and subjective data were generated and analysed. While the quantitative data collected via the task performance questionnaire were used to measure the speed of task completed, the qualitative data was collected via the post-experiment questionnaire completed by the participants and analysed. The next section describes the results of Experiment-3 in more details.

I. Task Completion Time (Speed)

Overall, the use of bubble graph had faster completion time with a median completion time of 10 seconds compared to 15 seconds for tabular visualisation. The magnitude of the difference in completion time is demonstrated by U-statistics from Mann-Whitney test, with U=8619.500, z=-4.983, p =.000. Furthermore, there was also a significant difference in the speed between the tasks in both task types (U=6139.500, z=-7.996, p=.000) with a median completion time of 9 seconds for tasks in category A, while the tasks in category B took a median time of 19 seconds. Figure 5.11 shows the median completion time for tasks in category A and category B using the bubble graph and tabular visualisation types. For the bubble graph visualisation, it took 7.5 seconds to complete category A tasks, and 13.5 seconds to complete category B tasks; whereas the same tasks were completed in 10 seconds and 26 seconds for category A and B tasks respectively using the tabular visualisation.

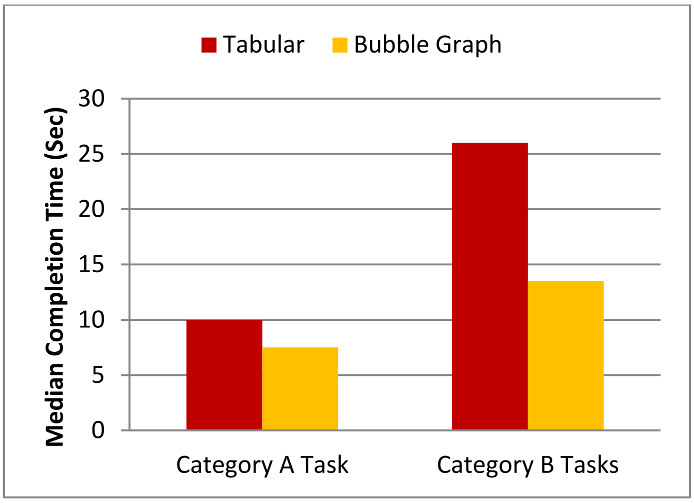


Figure 5.11: Median Time to Locate Services by Task Type

II. User Experience

Figure 5.12 presents the results from the post-study questionnaire. The goal of the post-experiment questionnaire was to capture user's impression and examine perceived quality of user experience of the visualisation formats. For the bubble graph visualisation, most of the questions received median scores from 6 on a 1-7 Likert scale. The highest score of 6 was received for the more general questions like "The bubble graph visualisation was easy to use" and "Overall, I was satisfied with the bubble visualisation format". In contrast, the tabular visualisation had slightly lower median scores of not more than 5.

A Wilcoxon Signed Rank Test revealed a statistically significant difference in the user experience of bubble graph (md=6.00) and tabular visualization (md=5.00), with z = – 5.237, p=0.018 ($p < .05$), with a medium effect size (r = .30). Furthermore, feedbacks were sought from the participants to ascertain perceived ease, speed and accuracy using both visualisation formats for both categories of tasks involving one and two QoS attributes.

Participants were asked to indicate which visualisation types were easier, faster and produced the most accurate result for both task categories. For exploration based on single QoS attributes, 70% of the participants confirmed that the bubble graph was easier to use compare to 30% who said the table was easier; 80% of participants said they performed the task faster than using table (20%), while 10% reported that the use of bubble graph was less accurate than tabular (90%) as shown in Figure 5.13. Also, while performing exploration using two QoS attributes, 80% and 90 % of the participants reported that the bubble graph was easier and faster respectively; while 70% said the use of bubble graph was more accurate than the tabular visualisation (30%) as depicted in Figure 5.14.

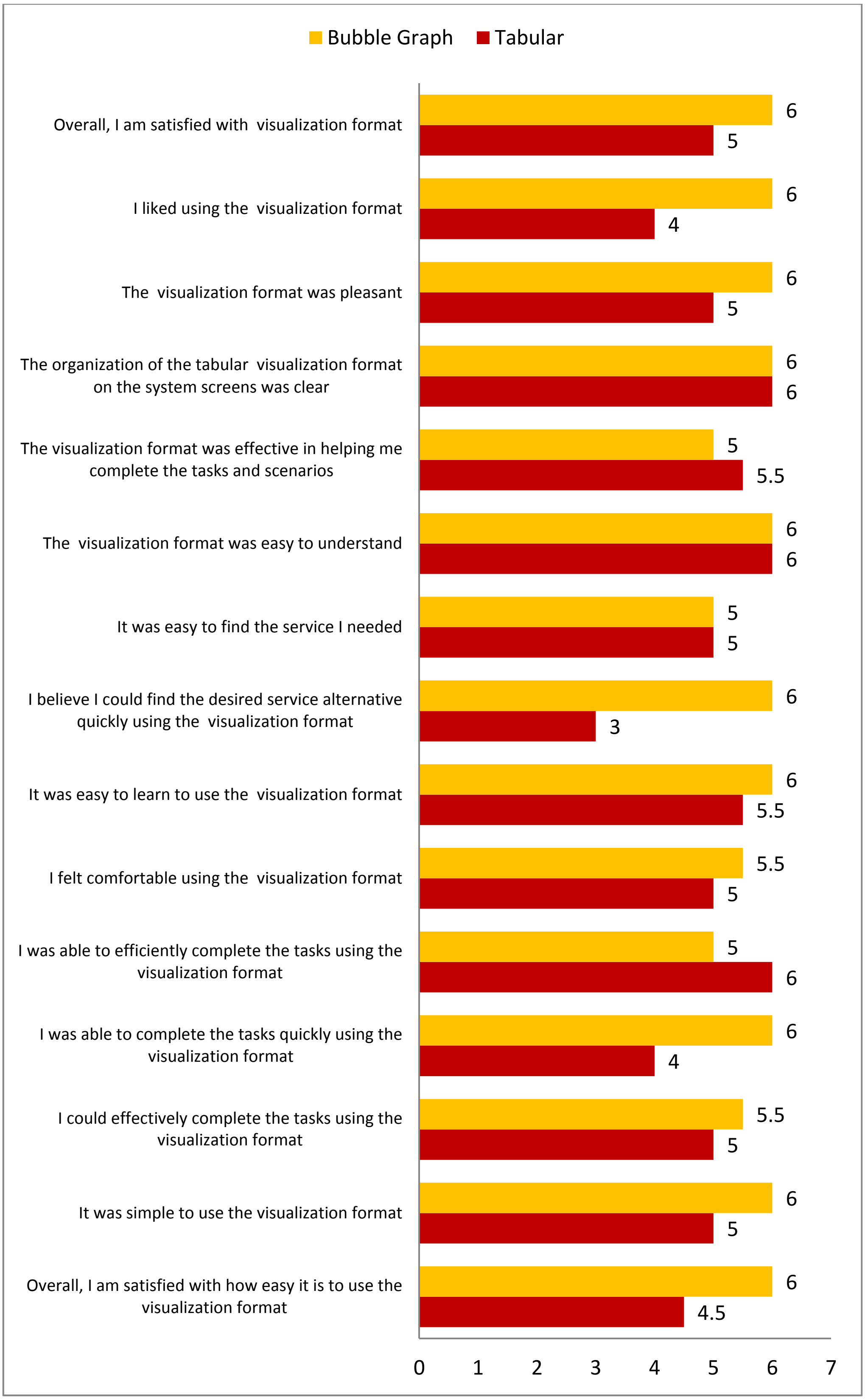


Figure 5.12: Median score for Post-Study Questionnaire

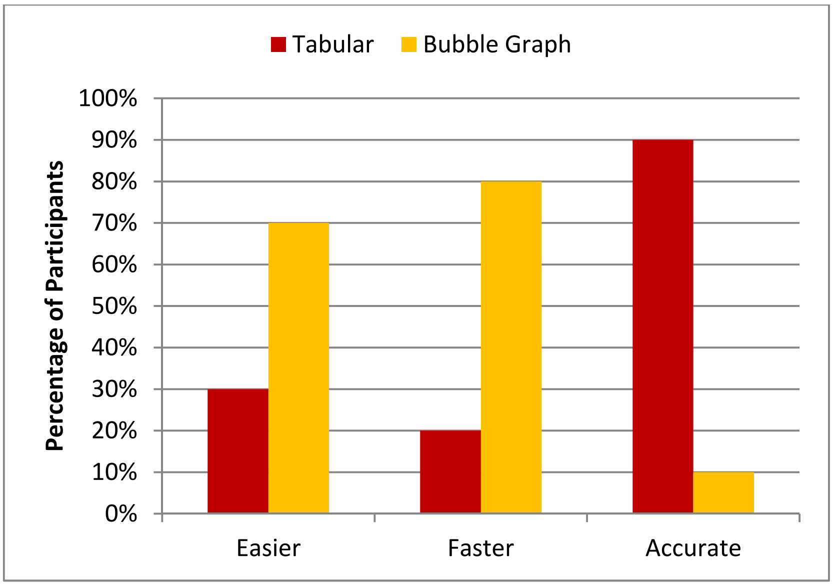


Figure 5.13: Perceived User satisfaction (Single QoS Attribute)

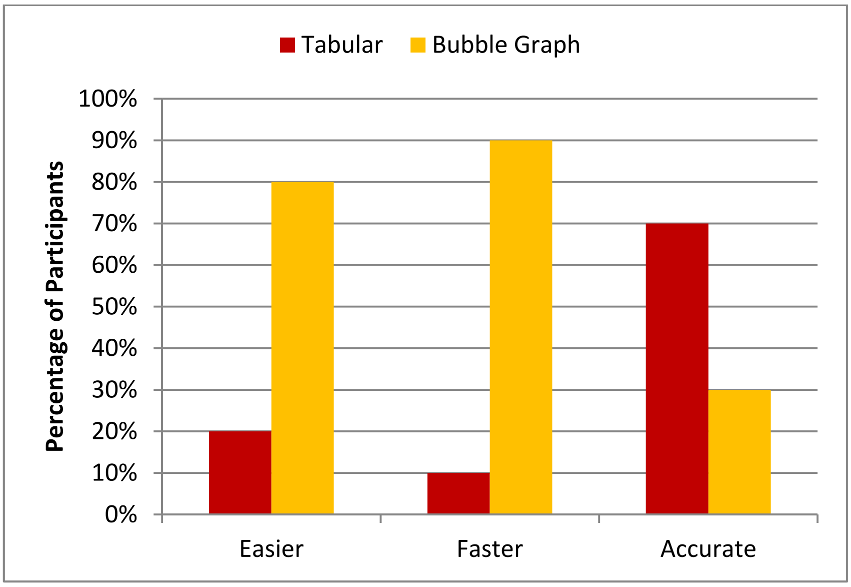


Figure 5.14: Perceived User satisfaction (Double QoS Attributes)

5.5.5 Threats to Validity

There are threats to experimental evaluations which often affect the validity of the results. The possible threats to the validity of the results obtained from this experiment were carefully investigated- including threats to internal, conclusion, external and construct validities. Internal validity refers to the causal relationship between the visualisation types and the effectiveness in performing locate task in terms of task completion time, as well as the perceived ease, speed and accuracy of the visualisation types. The threat to internal validity of our results is in the selection of the participants. Although the participants were selected at random, they all had the required knowledge background to act competently while participating in the experiment (Wohlin *et al.,* 2012).

Threats to the conclusion validity affect the ability to arrive at the correct conclusion about relations between the treatment and the outcome of an experiment (Wohlin *et al.*, 2012). The concern is the extent to which, we can generalise based on the experiment, particularly considering the number of participants (10) and their level of experience. It is possible to have obtained a different result with a bigger group and more experienced subjects. However, from the experiment, the subjects who were students have similar computing background as technology officers, who would normally make such decisions for many organisations. They also showed that they had potential to make rational decisions as regards the tasks assignment, although they are not adept as real technology experts, but good enough to form a valid opinion on the suitability of bubble graph and tabular visualisations. Also, for a first-rate evaluation experiment, 10 is an acceptable sample size in order to obtain a valid first impression (Turner *et al.*, 2006).

To ensure construct validity all participants performed exactly the same tasks based on the same set of instructions thus minimising any mono-method bias (Lam *et al.*, 2012). Therefore, there are no serious threats to validity for our conclusions from the experimental evaluation.

5.5.6 Discussion

Realising the vision of a true cloud service e-marketplace in the face of the growing trend for personalised products and services requires that user satisfaction and user experience be given top priority. The overall goal of this study is to simplify cloud service selection while optimising user experience and satisfaction in the decision-making process. Just like one of the laws of e-commerce states that if users cannot find it, they cannot buy it either; integrating information visualization in the User Interface (UI) design of e-marketplace provides the mechanisms for user to, in the shortest possible time and through the easiest means, find a cloud service that meets their requirements.

Humans possess the ability to recognise the spatial arrangements of elements in a picture and decipher relationships among elements quickly and easily. Such abilities enable humans to derive greater insight and comprehension from the content of a picture faster than mere text. This process leads to a more informed decision-making by capitalising on the well-developed human visual processing capability. This study posits that applying information visualisation technique for aiding cloud service selection would improve cloud service exploration, and proposed a visualisation framework to allow users compare cloud services with respect to their requirements.

The factors observed and measured in the experiments carried out were the speed at which the tasks were completed and the ease at which it was done. Generally, a faster completion time and greater ease in carrying out the assigned tasks meant the higher quality of user experience for a particular visualisation format. Although the use of table can be enhanced to include the ability to sort, the extra activity of sorting introduces additional complexity for the user when making a comparison. However, the bubble visualisation requires users just gazing at the visualisation (display) and with minimal interaction with the display (e.g. hovering), can gather more insight about the various alternatives. The task completion results show that bubble graph will drastically reduce the time it takes to find the most suitable service.

Consequent to the feedback provided by the participants, the results clearly indicates that using the tabular visualization to complete the tasks took longer time, and hence was more difficult, which, based on the feedback provided by the participants, was a result of the tabular formats could not adequately support participants in performing the required tasks. The bubble graph was the faster of the two formats evaluated (with a median time of 10 Seconds) particularly for more complex tasks involving two or more QoS attributes.

Based on the results obtained from this experiment, the two hypothesis set was rejected. Specifically, H_0 which states that there is no significant difference in task completion time of bubble graph visualization over the tabular visualization was rejected as the results showed significant difference in completion time (see Figure 5.11); in the same vein, H_0 was also rejected because there was a significant difference in user experience of bubble graph visualization compared to tabular visualization based on participants' feedback.

5.6 CHAPTER SUMMARY

In this chapter, the quantitative and qualitative experiments carried out to evaluate the FOCUSS framework were reported. This chapter described the experimental protocols followed to validate the scalability, ranking accuracy, user experience of the FOCUSS framework using descriptive and inferential statistics on data obtained from three experiments. The results obtained showed that the FOCUSS framework is viable for cloud service ranking and selection in cloud service e-marketplace contexts. The next chapter contains the summary, a highlight of the contribution to knowledge and the conclusion of this thesis.

CHAPTER SIX

CONCLUSION AND RECOMMENDATIONS

6.1 INTRODUCTION

This chapter highlights the summary and contribution of this thesis; and also contains recommendations for future work directions.

6.2 SUMMARY

Service providers leverage cloud ecosystems and cloud e-marketplaces to increase the business value of their services to reach a wider range of service users. A cloud services ecosystem is an environment that host heterogeneous cloud service offerings from different providers and affords the opportunity for collaborations. The cloud e-marketplace extends the concept of an e-marketplace and is an online platform that manages the distribution and trading of cloud services. On this platform, service providers enlist services with the purpose of integration with other services to form composite services for users to purchase. However, the growing popularity of cloud services requires cloud e-marketplaces that optimise user experience by enabling the composition of atomic services that satisfy complex user requirements beyond what atomic services can provide, while considering that the user's QoS requirements are elicited in ways akin to subjective human expressions. In addition, the user experience on such platforms can also be enhanced by showcasing a ranked result of services that match the user's QoS requirements via intuitive means that reduces the cognitive load of the users.

This study addresses the problem of service choice overload in cloud service e-marketplaces, which impacts negatively on user experience. So far the following has been accomplished in this study in line with the research questions and objectives of this study:

OBJECTIVE ONE: *To formulate an integrated service selection framework that will improve the quality of user experience in a cloud service e-marketplace.*

A review of the literature reveals that existing cloud selection approaches do not currently provide the sophistication to optimise user experience in the e-marketplace. Through the analysis of the state-of-the-art studies, a set of requirements was identified for a cloud service selection framework that would suffice in an e-marketplace context. Therefore, the study filled the existing gap in literature by proposing the FOCUSS framework as an integrated cloud service selection framework that incorporates mechanisms to address the existing gaps in cloud service selection literature, which are: 1) the need to compose atomic services on the fly to satisfy complex users' requirements; 2) the need to allow users the flexibility of expressing QoS requirements; both preferences and aspirations, and to be able to do so with subjective descriptors that are more akin to human judgment; and 3) the need to reduce choice overload by showing only the top best services in a manner that facilitates easy comparison for effective decision-making. These identified gaps formed the basis for the design of the FOCUSS framework, which comprises of four main components, namely; Cloud ecosystem and service directory, GUI and Visualisation, QoS requirement processing, and Service Evaluation and QoS ranking.

OBJECTIVE TWO: ***To design models and algorithms that will enable the components of the service selection framework.***

An assortment of models and algorithm were employed in the realisation of the components of the FOCUSS framework. Each component is described thus:

i. **Cloud Ecosystem and Service Directory:** The framework uses the extended feature model notations, to model the Cloud Ecosystem Feature Model (CEFM) that organises and formally compose atomic services to populate the service directory. The composite services are able to satisfy user complex requirements beyond what atomic services can handle. The CEFM is mapped as a constraint satisfaction problem and the Choco-based reasoning engine reasons with a Depth-First search algorithm to derive all valid mappings. Possible combinations of atomic services that can be generated from the pool of atomic services are made available in the e-marketplace based on former composition approaches.

ii. **GUI and Visualization:** The framework integrates fuzzy-based web interface widgets comprising sliders, drop-down menus and text boxes for eliciting vague QoS preferences and aspirations, while bubble graph visualisation is employed to improve understanding of the relationship among the ranked services. Users can indicate preferences by pairwise comparison for each QoS attribute by adjusting the slider handle. The slider bar has two colour codes that correspond to the QoS attributes and indicates the level of preference for a QoS attribute. Besides, humans derive better insight from a picture faster than mere text; therefore user experience is improved by the use of information visualisation. More specifically, the FOCUSS framework proposed the use of bubble graph visualisation to simplify decision making by showing how each service in the ranked results relates to others.

iii. **QoS Requirements Processing:** The QoS Requirements Processing module comprises the QoS Preference Prioritizer (QPP) and the QoS Aspiration Analyser (QAA) sub-modules. The QPP module ensures consistency in the pairwise judgment and uses the geometric mean method to derive priority weights. To prioritise user's QoS preferences, the QPP employs a Fuzzy AHP-based approach. The QAA module synthesises user's QoS values based on fuzzy decision making, comprising membership functions framed as fuzzy goal and constraints. Since the linguistic terminologies describing the QoS aspiration reflect the semantic approximations of user's intent, resolving the fuzzy decision results in an optimal set of QoS values that approximate user's QoS intent.

iv. **Service Evaluation and QoS Ranking:** The Service Evaluation and QoS Ranking modules consist of two sub-modules: the QoS Requirement Optimizer (QRP) and the QoS Ranking Engine (QRE). The QRP component computes the optimal QoS values that describe user's requirements based on the QoS information on all the services in the service directory. The inputs into this component are the priority weights and the value of QoS attributes. The framework defines two utility functions to evaluate the performance of each service alternative with respect to user's requirement. The output from the QRP forms the basis for ranking the services in the directory. The main technique used in this module is the nearest neighbour algorithm and the ranked output is fed into the bubble graph visualisation.

OBJECTIVE THREE: ***To implement a prototype of the service selection framework and demonstrate its plausibility***

The study used an illustrative case study of a Customer Relationship Management as a Service (CRMaaS) e-marketplace to demonstrate the plausibility of the FOCUSS framework. The envisioned CRMaaS ecosystem involves multiple atomic service providers who collaborate to provide CRM solutions, while prospective small businesses can purchase CRM solutions in the e-marketplace. The components that make up the CRMaaS ecosystem includes: Contact Management, Database, Marketing and Social media analysis.

The use cases featured a scenario of how two organisations, a Microfinance bank and online drug store, select appropriate services matching their respective QoS requirements from the CRMaaS e-marketplace. The illustrative case study described the whole process from ecosystem feature modelling, constraint-based reasoning, QoS aspiration and requirement specification, and visualisation of ranking results.

OBJECTIVE FOUR: ***To evaluate the framework in terms of its performance and usability attributes.***

An evaluation, comprising simulation experiments and user studies was performed to ascertain the performance and usability of the FOCUSS framework. First, the result of the scalability experimental simulation confirmed the performance of the FOCUSS framework in terms of the time it takes to rank top-k services. A linear regression analysis of data collected from the simulation shows that the FOCUSS framework scales linearly with an increase in service alternatives in terms of performance, thus confirming the computational efficiency of the FOCUSS QoS-based ranking module. The second simulation experiment tested the ranking accuracy of two versions of the FOCUSS ranking algorithm compared to existing methods and tested the hypothesis that there will be no significant difference between the ranking performances of a method that accepts exact numeric values as QoS requirement and those that use linguistic descriptors to approximate values for QoS requirements. Judging by the results from both the descriptive and inferential analysis, the two versions of the FOCUSS methods (FOCUSS_Num and FOCUSS_Lin) produce more accurate results on all four metrics considered and were in all cases closer to the benchmark metric (TOPSIS) than the other two methods used in this simulation. Finally, a user study was undertaken to ascertain the usability attributes of the visualisation component of the FOCUSS framework. The summary of the results of the user study showed that the use of bubble graph recorded higher accuracy, faster completion time and greater ease in carrying out the assigned tasks; thus corresponding to the higher quality of user experience.

6.3 CONTRIBUTIONS TO KNOWLEDGE

This study contributes to the general research areas of cloud service selection and decision making as it applies to cloud service e-marketplaces. More specifically, the main contribution of this study caters for the observed limitations in the existing cloud service selection approaches by enabling the 1) formal composition of atomic services to satisfy complex user requirements beyond what atomic services can deliver; 2) elicitation and processing of subjective user QoS requirements in ranking cloud services; 3) presentation of search results in a visually intuitive way that aids in better decision making.

To this end, the Fuzzy-Oriented Cloud Service Selection (FOCUSS) framework was formulated to improve the user experience in a cloud service e-marketplace. FOCUSS is an integrated framework for cloud service ranking and selection, proposed as an efficient integrated visual-rich fuzzy-based decision support that incorporates a feature modelling, constraint-based reasoning, fuzzy decision making, fuzzy optimisation and visualisation in its design for cloud service selection in cloud service e-marketplace context. More concretely the FOCUSS framework:

a) **Satisfies complex user requirements beyond what atomic services can deliver.** Currently, users are constrained to make choices only from a set of predefined atomic services, or at best, manually configure their desirable features and QoS requirements in order to realise their complex requirements given that they have deep knowledge of the service domain. FOCUSS employs constraint-based reasoning on the feature model to formally compose atomic services to fulfil complex user requirements.

b) **Elicits and processes subjective user QoS preferences and aspirations.** Without proper articulation of requirements, cloud service selection can be overwhelming, and leads to service choice overload; more so that user requirements, broken into QoS aspiration and QoS preferences, are often shrouded in vagueness and subjectivity. Contrary to existing approaches in which either vague QoS preferences or aspirations are considered, FOCUSS elicit user QoS requirements in a way that captures the vagueness inherent in both the users' QoS preferences and aspirations and optimises these QoS inputs dimensions to identify suitable services options.

c) **Presents search results in a visually intuitive way that aids in better decision making.** The search results from many cloud service e-marketplace are usually presented as an unordered list of icons representing the services that best fit users' keyword-based queries. The drawback of such presentation mechanisms is that users are not able to immediately discriminate among the cloud services for easy decision making. FOCUSS simplifies decision making as users can identify services that best fit their requirements quicker and easier compared to tabular formats.

6.4 CONCLUSION

The popularity of cloud computing has led to the proliferation of services that are commoditized and traded via cloud e-marketplaces. The benefits of employing cloud-based services compel many enterprises, particularly small businesses, to migrate over to the cloud (Budniks and Didenko, 2014; Ross and Blumenstein, 2015; Sultan, 2011). Realising the vision of a true cloud service e-marketplace in the face of the growing trend for personalised products and services requires that user satisfaction and user experience be given top priority. An organisation's resolution to adopt a new cloud service requires decision support in navigating the vast plethora of services (Qu and Buyya, 2014; Saripalli and Pingali, 2011). Without proper articulation of requirements, cloud service selection in the face of so many choices can be overwhelming and leads to service choice overload. Decision support becomes essential because cloud service selection involves the consideration of multiple QoS attributes, which are compared to a variety of services; often based on QoS requirements that are vague or subjective in nature.

The overall goal of this study was to simplify cloud service selection while optimising user experience in the decision-making process. For this, an integrated fuzzy-oriented framework was proposed to facilitate an enhanced user experience in cloud e-marketplaces through the formal composition of atomic services to satisfy complex user requirements, elicitation and processing of subjective user QoS requirements, and presentation of search results in a visually intuitive way that aids users' decision making. To do this, an integration of key concepts such as constrained-based reasoning on feature models, fuzzy pairwise comparison of QoS attributes, fuzzy decision making, and information visualisation have been used. Results from experiments performed showed that the FOCUSS framework is scalable; ranks services using subjective descriptors and optimises the user experience in cloud service e-marketplace.

6.5 Recommendations for Future Work

The study provides possibilities for further research in tandem with the dynamism of the cloud computing landscape and user experience dimensions. The issues related to future works identified are as follows:

a) *Managing the Heterogeneity of QoS Information*

Although QoS are measurable non-functional attributes that describe and distinguish services and forms the basis for service selection (Chen *et al.*, 2013; Abdelmaboud *et al.*, 2015), QoS properties are usually heterogeneous in nature, covering both quantitative and qualitative (or categorical) attributes. Besides, the Service Measurement Index (SMI) defines seven main metrics to measure QoS of cloud services, which includes Accountability, Agility, Assurance, Financial, Performance, Security and Privacy, and Usability; including multiple attributes under each categories and have values that are either quantitative or qualitative (or categorical) in nature. For example, response time is measured using quantitative numeric values (in milliseconds), while security and user friendliness or ecosystem friendliness are measured based on qualifier tags such as good, high etc. Many cloud service selection frameworks have only considered quantitative attributes, for example Rehman *et al.* (2011), Jung *et al.* (2013) and Mirmotalebi *et al.* (2012), based on the assumption that all QoS attributes are quantitative in nature, such approaches are limited and cannot suffice to handle the heterogeneous QoS model of cloud services, to cover for both quantitative and qualitative (or categorical) QoS dimensions. To effectively achieve a QoS-based ranking of cloud services in cloud service e-marketplaces, there is need to consider both the quantitative and qualitative QoS dimensions that characterise cloud services and rank cloud services accurately with respect to user requirements. To achieve this, heterogeneous similarity metrics that combines quantitative and qualitative dimensions, such as the Heterogeneous Euclidean Overlap Metric and Heterogeneous Value Difference Metric (Wilson and Martinez, 1997) can be employed for QoS-based ranking to enable the selection of services in cloud service e-marketplace.

b) Managing the Fuzzy Nature of QoS information

Since the cloud computing landscape is characterised by dynamism, the correct and accurate evaluation of the QoS performance of cloud services should be a constant. The objective evaluation of cloud services sourced from service monitoring or benchmark third party services (Ruiz-Alvarez and Humphrey, 2011) or subjective feedback assessment from other users would constantly alter the QoS information of cloud services in the ecosystem. Hence a means to update aggregated QoS information and a constant update of the QoS information of the services is required. The automated and dynamic update is activated as new services join, or exit the ecosystem and also when there is an adjustment in the QoS information of a service. So we must find a means to capture and manage the uncertain nature of the QoS information of services using a fuzzy number or interval numbers according to the QoS history of services.

c) Managing the Size of Cloud Ecosystem feature model

To further increase the business value of their services, more service providers will likely participate in cloud ecosystems. Consequently, as the size of the cloud ecosystem increases, the potential number of composite service formally or incidentally composed will also increase. Retrieval of services in response to user queries and requirements will be enhanced by efficient storage of these composite offerings with multiple QoS attributes. One challenge with the plethora of services is managing the storage of a large number of services. In realising the FOCUSS framework, a relational database management system (RDBMS) was employed to store the list of service. The efficiency of retrieval will be reduced with the use of RDBMS. Since the service registration phase usually occurs offline together with the derivation of valid composition, a plausible approach is to apply case retrieval nets (Lenz and Burkhard, 1996). Each service can be referred to as a case, while the case retrieval nets are employed to manage the large size of the resultant service compositions and provide efficient retrieval compared to traditional RDBMS.

d) Serendipity and Diversity in Service Selection

Existing approaches elicit user's QoS requirements preference *a priori*. A priori elicitation suggests that QoS requirements are specified at the onset while the system generates and present services that satisfy the user requirements. Similarity-based retrieval based on a priori requirements elicitation cannot address the *Stonewalling* and *Diversity problem* of recommendation (Bridge, 2001). Stonewalling refers to a scenario where the system respects all the preference of the consumer and yet no recommendation is returned (e.g. system returns 'No Match Found!"). Diversity problem arises when the system returns a number of similar services, and the similarities among the services are so close without any diversity. In both scenarios, the user is expected to start the search all over again from scratch, since the recommendation system does not remember nor consider previously specified preferences.

e) Group Decision Making Scenarios

The GUISET project is designed to enable a cluster of SMEs to use technology in their business by lowering the initial cost of acquisition and maintenance. A cluster of SME comprises at least two SME, whose inputs matter in the selection of cloud services for their businesses. In such scenarios, it is also possible to include group decision-making scenarios in the quest for cloud service selection; this is particularly true considering the unique model of the GUISET project in which the prospective users are a cluster of SMME who require cloud services to e-enable their business activities. Each stakeholder in an SMME cluster should make sufficient input in the decision-making process to select a cloud service. The selection takes into cognisance all stakeholders' QoS requirements and aggregates the requirements to produce a single service. The scope of this research was mainly focused on a single user organisation, and there is a value in expanding it to address and incorporate inputs from users in a group. Although one major challenge with group decision making is conflicting user requirements, this challenge can be solved by finding the Pareto optimal services that match all the requirements (Yu, 2014). Also, the best set of services matching the group QoS requirements can be obtained by employing regression analysis to determine the QoS values of the services that with the least contradictions among users QoS requirement. The QoS values solution can then be mapped to utility functions that can be used to evaluate all services in the cloud e-marketplace.

REFERENCES

Abdelhamid, R. (2012). A decision support system for performance evaluation. *Int. J. Comput. Appl (Special Issue on Computational Intelligence and Information Security)*, **1:** 1-8.

Abdelmaboud, A., Jawawi, D. N., Ghani, I., Elsafi, A., and Kitchenham, B. (2015). Quality of service approaches in cloud computing: a systematic mapping study. *Journal of Systems and Software*, **101**: 159-179.

Abraham, A., Jain, L. C., and Goldberg, R. (2005). *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications.* London: Springer-Verlag. ISBN:1852337877.

Adnan, W. A., Daud, N., and Noor, N. L. (2008). Expressive information visualization taxonomy for decision support environment. *Third International Conference on Convergence and Hybrid Information Technology* (pp. 88-93). IEEE.

Afshari, A., Mojahed, M., and Yusuff, R. M. (2010). Simple additive weighting approach to personnel selection problem. *International Journal of Innovation, Management and Technology*, **1 (5):** 511-515.

Akingbesote, A., Adigun, M., Jembere, E., Othman, M., and Ajayi, I. (2014). Determination of optimal service level in cloud e-marketplaces based on service offering delay. *International Conference on Computer, Communications, and Control Technology (I4CT)* (pp. 283-288). Langkawi, Kedah, Malaysia: IEEE.

Akolkar, R., Chefalas, T., Laredo, J., Peng, C.-S., Sailer, A., Schaffa, F., et al. (2012). The future of service marketplaces in the cloud. *IEEE Eighth World Congress on Services (SERVICES)* (pp. 262-269). IEEE.

Al-Masri, E., and Mahmoud, Q. H. (2007). QoS-based discovery and ranking of web services. *IEEE 16th International Conference on Computer Communications and Networks (ICCCN)* (pp. 529-534). IEEE.

Almulla, M., Yahyaoui, H., and Almatori, K. (2012). Visualization of real-world web services based on fuzzy logic. *Proceedings of 2012 IEEE Eighth World Congress on Services* (pp. 330-335). IEEE.

Alrifai, M., Skoutas, D., and Risse, T. (2010). Selecting skyline services for QoS-based web service composition. *Proceedings of the 19th international conference on World wide web* (pp. 11-20). ACM.

Al-Shammari, S., and Al-Yasiri, A. (2014). Defining a metric for measuring QoE of SaaS cloud computing. *Proceedings of PGNET*, (pp. 251-256).

AppExchange. (2015). *AppExchange*. Retrieved September 9, 2015, from Business app store from Salesforce: https://appexchange.salesforce.com/

Aruldoss, M., Lakshmi, T. M., and Venkatesan, V. P. (2013). A survey on multi criteria decision making methods and its applications. *American Journal of Information Systems*, **1 (1):** 31-43.

Avram, M.-G. (2014). Advantages and challenges of adopting cloud computing from an enterprise perspective. *Procedia Technology*, **12:** 529-534.

Ayeldeen, H., Shaker, O., Hegazy, O., and Hassanien, A. E. (2015). Distance similarity as a CBR technique for early detection of breast cancer: an Egyptian case study. In J. Mandal, S. Satapathy, S. Kumar, P. Sarkar, A. Mukhopadhyay (Eds.): *Information Systems Design and Intelligent Applications* (pp 449-456).

Ayhan, M. B. (2013). A fuzzy AHP approach for supplier selection problem: A case study in a Gear motor company. *International Journal of Managing Value and Supply Chains*, **4 (3):** 11-23.

Baek, S., Kim, K., and Altmann, J. (2014). Role of platform providers in service networks: The case of Salesforce.com AppExchange. *IEEE 16th Conference on Business Informatics (CBI)* (pp. 39-45). IEEE.

Bai, Y., and Wang, D. (2006). Fundamentals of fuzzy logic control – fuzzy sets, fuzzy rules and defuzzifications. In Y. Bai; H. Zhuang, D. Wang (Eds.): *Advanced Fuzzy Logic Technologies in Industrial Applications* (pp. 17-36).

Bakos, Y. (1998). The emerging role of electronic marketplaces on the Internet. *Communications of the ACM*, **41 (8):** 35-42.

Baranwal, G., and Vidyarthi, D. P. (2014). A framework for selection of best cloud service provider using ranked voting method. *IEEE International Advance Computing Conference (IACC)* (pp. 831-837). IEEE.

Barros, A. P., and Dumas, M. (2006). The rise of web service ecosystem. *IT Professional*, **8 (5):** 31-37.

Bass, L., and Kazman, R. (2003). *Software architecture in practice.* M.A: Addison-Wesley. ISBN: 0321815734.

Batory, D., Benavides, D., and Ruiz-Cortes, A. (2006, December). Automated analysis of feature models: challenges ahead. *Communications of the ACM*, **49 (12):** pp. 45-47.

Beard, D. V., and Walker, J. Q. (1990). Navigational techniques to improve the display of large two-dimensional spaces. *Behaviour and Information Technology*, **9 (6):** 451-466.

Beets, S., and Wesson, J. (2010). *Can Information Visualization techniques be used to support web service discovery.* Retrieved May 17, 2015, from Proceedings of the 2010 Southern Africa Telecommunication Networks and Applications Conference (SATNAC): http://www.satnac.org.za/proceedings/2010/papers/progress/Beets%20FWIP%20495.pdf

Beets, S., and Wesson, J. (2011). Using Information Visualization to support web service discovery. *Proceedings of the South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment* (pp. 11-20). ACM.

Bellman, R. E., and Zadeh, L. A. (1970). Decision making in fuzzy environment. *MonogemenlScience*, **17 (4):** 141-164.

Benavides, D., Ruiz–Corte´s, A., Trinidad, P., and Segura, S. (2006). A survey on the automated analyses of feature model. *Jornadas de Ingenier´ıa del Software y Bases de Datos*, 367-376.

Benavides, D., Segura, S., and Ruiz-Cortes, A. (2010). Automated analysis of feature models 20 years later: A literature review. *Information Systems*, **35 (6):** 615-636.

Benayoun, R., Roy, B., and Sussman, B. (1966). ELECTRE: Une méthode pour guider le choix en présence de points de vue multiples. *Note de travail*, no:49.

Benlachgar, A., and Belouadha, F.-Z. (2013). Review of software product line models used to model cloud applications. *ACS International Conference on Computer Systems and Applications (AICCSA)* (pp. 1-4). IEEE.

Berger, T., Pfeiffer, R.-H., Tartler, R., Dienst, S., Czarnecki, K., Wasowski, A., et al. (2014). Variability mechanisms in software ecosystems. *Information and Software Technology*, **56 (11):** 1520-1535.

Bertin, J. (1983). *Semiology of graphics: diagrams, networks, maps.* University of Wisconsin press. ISBN: 0299090604.

Bevan, N. (2009). What is the difference between the purpose of usability and user experience evaluation methods? *Proceedings of the Workshop UXEM'09 (Interact 09):* (pp. 1-4).

Bollen, D., Knijnenburg, B. P., Willemsen, M. C., and Graus, M. (2010). Understanding choice overload in recommender systems. *Proceedings of the fourth ACM conference on Recommender systems* (pp. 63-70). ACM.

Bonastre, L., and Granollers, T. (2014). A set of heuristics for user experience evaluation in e-commerce websites. *The Seventh International Conference on Advances in Computer-Human Interactions* (pp. 27-34). Achi.

Bosch, J., and Bosch-Sijtsema, P. (2010). From integration to composition: on the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, **81 (3):** 67-76.

Bouanaka, M. A., and Zarour, N. (2013). An approach for an optimized web service selection based on skyline. *International Journal of Computer Science Issues*, **10 (1):** 412-418.

Bouyssou, D. (1996). Outranking relations: do they have special properties? *Journal of Multi-Criteria Decision Analysis*, **5 (2):** 99-111.

Brans, J.-P., Vincke, P., and Mareschal, B. (1986). How to select and how to rank projects: The PROMETHEE method. *European journal of operational research*, **24 (2):** 228-238.

Braun, P. (2005). Small Business Clustering: Accessing Knowledge through Local Networks. *Unpublished paper presented at The CRIC Cluster conference: Beyond Cluster- Current Practices and Future Strategies.* Ballarat.

Bridge, D. (2001). Product recommendation systems: a new direction. *Procs. of the Workshop Programme at the Fourth International Conference on Case-Based Reasoning*, (pp. 79-86).

Buckley, J. (1985). Fuzzy hierarchical analysis. *Fuzzy Sets and Systems*, **17 (3):** 233-247.

Budniks, L., and Didenko, K. (2014). Factors determining application of cloud computing services in Latvian SMEs. *19th International Scientific Conference Economics and Management 2014* (pp. 74 – 77). Riga, Latvia: Elsevier.

Burigat, S., and Chittaro, L. (2013). On the effectiveness of Overview+Detail visualization on mobile devices. *Personal and ubiquitous computing*, **17 (2):** 371-385.

Buyya, R., Yeo, C. S., and Venugopal, S. (2008). Market-oriented cloud computing. *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC'08)* (pp. 5-13). IEEE.

Cakir, O., and Canbolat, M. S. (2008). A web-based decision support system for multi-criteria inventory classification using fuzzy AHP methodology. *Expert Systems with Applications*, **35 (3):** 1367–1378.

Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). *Readings in information visualization: using vision to think.* Morgan Kaufmann. ISBN: 1-55860-533-9.

Cavalcante, E., Batista, T., Lopes, F., Rodriguez, N., de Moura, A. L., Delicato, F. C., et al. (2012). Optimizing Services Selection in a Cloud Multiplatform Scenario. *IEEE Latin America Conference on Cloud Computing and Communications (LATINCLOUD)* (pp. 31-36). IEEE.

Chamodrakas, I., Leftheriotis, I., and Martakos, D. (2011). In-depth analysis and simulation study of an innovative fuzzy approach for ranking alternatives in multiple attribute decision making problems based on TOPSIS. *Applied Soft Computing*, **11 (1):** 900--907.

Chang, D.-Y. (1996). Applications of the extent analysis method on fuzzy AHP. *European Journal of Operational Research*, **95 (3):** 649-655.

Chen, X., Zheng, Z., Liu, X., Huang, Z., and Sun, H. (2013). Personalized QoS-Aware web Service recommendation and visualization. *IEEE Transactions on Services Computing*, **6 (1):** 35-47.

Chernev, A., Böckenholt, U., and Goodman, J. (2015). Choice overload: A conceptual review and meta-analysis. *Journal of Consumer Psychology*, **25 (2):** 333–358.

Chittaro, L. (2006). Visualizing information on mobile devices. *Computer*, **39 (3):** 40-45.

Choi, C. R., and Jeong, H. Y. (2014). A broker-based quality evaluation system for service selection according to the QoS preferences of users. *Information Sciences*, **227:** 553-566.

Chou, S.-Y., Chang, Y.-H., and Shen, C.-Y. (2008). A fuzzy simple additive weighting system under group decision-making for facility location selection with objective/subjective attributes. *European Journal of Operational Research*, **189 (1):** 132-145.

Chua, F. F., Yuan, H., and Kim, S. D. (2007). A visualization framework for web service discovery and selection based on quality of service. *The 2nd IEEE Asia-Pacific Service Computing Conference* (pp. 312-319). IEEE.

Cockburn, A., Karlson, A., and Bederson, B. B. (2009). A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys (CSUR)*, **41 (1):** 1-31.

Coll, R. A., Coll, J. H., and Thakur, G. (1994). Graphs and tables a four-factor experiments. *Communications of the ACM*, **37 (4):** 76-87.

Cox, E. (2005). *Fuzzy Modeling and genetic algorithms for data mining and exploration.* Elsevier Inc. ISBN: 9780121942755.

CSMIC. (2014, July). *Service Measurement Index Framework Version 2.1.* Retrieved September 17, 2015, from Cloud Services Measurement Initiative Consortium: http://csmic.org/downloads/SMI_Overview_TwoPointOne.pdf

Csutora, R., and Buckley, J. J. (2001). Fuzzy hierarchical analysis: the Lambda-Max method. *Fuzzy Set. Syst.*, **120 (2):** 181-195.

Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., and Wąsowski, A. (2012). Cool features and tough decisions: A comparison of variability modeling approaches. *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems (VaMoS '12)* (pp. 173-182). NY, USA: ACM.

Czarnecki, K., Helsen, S., and Eisenecker, U. (2005). Formalizing cardinality□based feature models and their specialization. *Software Process: Improvement and Practice*, **10 (1):** 7-29.

Dastjerdi, A. V., Garg, S. K., Rana, O. F., and Buyya, R. (2015). CloudPick: a framework for QoS-aware and ontology-based service deployment across clouds. *Software: Practice and Experience*, **45 (2):** 197-231.

Dastjerdi, A., and Buyya, R. (2011). A taxonomy of QoS management and service selection methodology for cloud computing. In L. Wang, R. Ranjan, J. Chen, and B. Benatallah, *Cloud computing: methodology, systems, and applications* (pp. 109-131). Boca Raton: CRC Press. ISBN: 9781439856413.

De Oliveira, R., Cherubini, M., and Oliver, N. (2012). Influence of Usability on Customer Satisfaction: A Case Study on Mobile Phone Services. *International Workshop on the Interplay between User Experience and Software Development*, (pp. 14-19).

Deelstra, S., Sinnema, M., and Bosch, J. (2005). Product derivation in software product families: a case study. *Journal of Systems and Software*, **74 (2):** 173-194.

Ding, S., Yang, S., Zhang, Y., Liang, C., and Xia, C. (2014). Combining QoS prediction and customer satisfaction estimation to solve cloud service trustworthiness evaluation problems. *Knowledge-Based Systems*, **56 (1):** 216-225.

Draper, G. M., Livnat, Y., and Riesenfeld, R. F. (2009). A survey of radial methods for information visualization. *IEEE transactions on visualization and computer graphics*, **15 (5):** 759-776.

Ehrgott, M., Waters, C., Kasimbeyli, R., and Ustun, O. (2009). Multiobjective programming and multiattribute utility functions in portfolio optimization. *INFOR: Information Systems and Operational Research*, **47 (1):** 31-42.

Elfaki, A. O., Abouabdalla, O. A., Fong, S. L., Johar, M. G., Aik, K. L., and Bachok, R. (2012). Review and future directions of the automated validation in software product line engineering. *Journal of Theoretical and Applied Information Technology*, **42 (1):** 75-93.

Esposito, C., Ficco, M., Palmieri, F., and Castiglione, A. (2016). Smart loud storage service selection based on fuzzy logic, theory of evidence and game theory. *IEEE Transactions on Computers*, **65 (8):** 2348-2362.

Forman, E. H., and Gass, S. I. (2001). The analytic hierarchy process-an exposition. *Operations research*, **49 (4):** 469-486.

Galitz, W. O. (2007). *The essential guide to user interface design: an introduction to GUI design principles and techniques.* John Wiley and Sons. ISBN:0470053429.

Garcıa-Galán, J. R.-C. (2013). Migrating to the cloud: a software product line based analysis. *3rd International Conference on Cloud Computing and Services Science (CLOSER):* (pp. 416-426).

Garg, S. K., Versteeg, S., and Buyya, R. (2013). A framework for ranking of cloud computing services. *Future Generation Computer Systems*, **29 (4):** 1012-1023.

Garg, S. K., Versteeg, S., and Buyya, R. (2011). SMICloud: A framework for comparing and ranking cloud services. *Fourth IEEE International Conference on Utility and Cloud Computing (UCC)* (pp. 210-218). IEEE.

Gartner. (2016). *Public Cloud Services, Worldwide, 2011-2016, 4Q12 Update*. Retrieved 2016, from Gartner: www.gartner.com/resId=2332215

Gatzioura, A., Menychtas, A., Moulos, V., and Varvarigou, T. (2012). Incorporating business intelligence in cloud marketplaces. *IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA)* (pp. 466-472). IEEE.

Ge, M., Delgado-Battenfeld, C., and Jannach, D. (2010). Beyond accuracy: evaluating recommender systems by coverage and serendipity. *Proceedings of the fourth ACM conference on Recommender systems* (pp. 257-260). ACM.

Ghosh, P., and Shneiderman, B. (1999). *Zoom-only vs. overview-detail pair: a study in browsing techniques as applied to patient histories.* University of Maryland Technical Report CS-TR-4028.

Gui, Z., Yang, C., Xia, J., Huang, Q., Liu, K., Li, Z., et al. (2014). A service brokering and recommendation mechanism for better selecting cloud services. *Plos One*, **9 (8):** e105297.

Hamwele, T. (2005, December 17). *SARPN - Namibia.* Retrieved May 15, 2012, from Southern African Regional Poverty Network (SARPN) Website: http://www.sarpn.org/documents/d0001692/P2029-SMEs_Tuwilika_Oct2005.pdf

Han, S.-M., Hassan, M. M., Yoon, C.-W., and Huh, E.-N. (2009). Efficient service recommendation system for cloud computing market. *Proceedings of the 2nd international conference on interaction sciences: information technology, culture and human* (pp. 839-845). ACM.

Haug, A., Hvam, L., and Mortensen, N. H. (2011). The impact of product configurators on lead times in engineering-oriented companies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **25 (2):** 197-206.

Haynes, G. A. (2009). Testing the boundaries of the choice overload phenomenon: The effect of number of options and time pressure on decision difficulty and satisfaction. *Psychology and Marketing*, **26 (3):** 204-212.

He, Q., Han, J., Yang, Y., Grundy, J., and Jin, H. (2012). QoS-driven service selection for multi-tenant SaaS. *IEEE 5th international conference on Cloud computing (cloud)* (pp. 566-573). IEEE.

Herman, I., Melancon, G., and Marshall, M. S. (2000). Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on visualization and computer graphics*, **6 (1):** 24-43.

Hornbæk, K., and Frøkjær, E. (2001). Reading electronic documents: The usability of linear, fisheye, and overview+detail interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 293-300). ACM.

Hornbцk, K. B., and Plaisant, C. (2002). Navigation patterns and usability of overview+detail and Zoomable user interfaces for maps. *Transactions on Computer Human Computer Interaction*, **9 (4):** 362-389.

Hubaux, A., Jannach, D., Drescher, C., Murta, L., Mannisto, T., Czarnecki, K., et al. (2012). Unifying software and product configuration: A research roadmap. *Proceedings of the Workshop on Configuration (ConfWS'12):* (pp. 31-35). Montpellier, France.

Hvam, L., Henrik Mortensen, N., and Riis, J. (2008). *Product Customization.* Springer Science and Business Media. ISBN: 978-3-540-71449-1.

Hwang, C., Lai, Y., and Liu, T. (1993). A new approach for multiple objective decision making. *Computers and Operational Research*, **20 (8):** 889–899.

Hwang, L., and Yoon, K. (1981). *Multiple attribute decision making: methods and applications.* New York: Springer-Verlag. ISBN: 978-3-642-48318-9.

Iyengar, S. S., and Lepper, M. R. (2000). When choice is demotivating: Can one desire too much of a good thing? *Journal of personality and social psychology*, **79 (6):** 995-1006.

Jahani, A., Khanli, L. M., and Razavi, S. N. (2014). W_SR: A QoS based ranking approach for cloud computing service. *Computer Engineering and Applications Journal*, **3 (2):** 55-62.

Jahanshahloo, G., Lotfi, F. H., and Izadikhah, M. (2006). Extension of the TOPSIS method for decision-making problems with fuzzy data. *Applied Mathematics and Computation*, **181 (2):** 1544–1551.

Jarvenpaa, S. L. (1989). The effect of task demands and graphical format on. information processing strategies. *Management Science*, **35 (3):** 285-303.

Jarvenpaa, S.-L., and Dickson, G. W. (1988). Graphics and managerial decision making: Research-based guidelines. *Communications of the ACM*, **31 (6):** 764-774.

Javanbarg, M. B., Scawthorn, C., Kiyono, J., and Shahbodaghkhan, B. (2012). Fuzzy AHP-based multicriteria decision making systems using particle swarm optimization. *Expert Systems with Applications*, **39 (1):** 960–966.

Javed, B., Bloodsworth, P., Rasool, R. U., Munir, K., and Rana, O. (2016). Cloud market maker: An automated dynamic pricing marketplace for cloud users. *Future Generation Computer Systems*, **54:** 52-67.

Jula, A., Sundararajan, E., and Othman, Z. (2014). Cloud computing service composition: A systematic literature review. *Expert Systems with Applications*, **41 (8):** 3809–3824.

Jung, G., Mukherjee, T., Kunde, S., Kim, H., Sharma, N., and Goetz, F. (2013). CloudAdvisor: A recommendation-as-a-service platform for cloud configuration and pricing. *203 IEEE Ninth World Congress on Services (SERVICES)* (pp. 456-463). IEEE.

Jussien, N., Rochart, G., and Lorca, X. (2008). Choco: an open source java constraint programming library. *CPAIOR'08 Workshop on Open-Source Software for Integer and Contraint Programming (OSSICP'08*, (pp. 1-10).

Kang, J., and Sim, K. M. (2010). Cloudle: a multi-criteria cloud service search engine. *2010 IEEE Asia-Pacific Services Computing Conference (APSCC)* (pp. 339-346). IEEE.

Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, S. (1990, November). Feature–Oriented Domain Analysis (FODA) Feasibility. *Technical Report CMU/SEI-90-TR-21* . Software Engineering Institute, Carnegie Mellon University.

Karataş, A. S., Oğuztüzün, H., and Doğru, A. (2012). From extended feature models to constraint logic programming. *Science of Computer Programming*, **78 (12):** 2295-2312.

Karim, R. a. (2013). An end-to-end QoS mapping approach for cloud service selection. *Proceedings of IEEE Ninth World Congress on Services* (pp. 341-348). IEEE.

Khadka, R., Saeidi, A., Jansen, S., Hage, J., and Helms, R. (2011). An evaluation of Service frameworks for the management of service ecosystems. *PACIS 2011 proceedings*, (Paper 93).

Khan, M., and Khan, S. (2011). Data and information visualization methods, and interactive mechanisms: A survey. *International Journal of Computer Applications*, **34 (1):** 1-14.

Knijnenburg, B. P., and Willemsen, M. C. (2009). Understanding the effect of adaptive preference elicitation methods on user satisfaction of a recommender system. *Proceedings of RecSys'09*, (pp. 381-384).

Kuniavsky, M. (2003). *Observing the user experience: a practitioner's guide to user research.* Morgan kaufmann. ISBN: 0123848695

Kwon, H.-K., and Seo, K.-K. (2013). A decision-making model to choose a cloud service using fuzzy AHP. *Advanced Science and Technology Letters*, **35:** 93-96.

Lam, H., Bertini, E., Isenberg, P., Plaisant, C., and Carpendale, S. (2012). Empirical studies in information visualization: Seven scenarios. *IEEE Transactions on Visualization and Computer Graphics*, **18 (9):** 1520-1536.

Lenz, M., and Burkhard, H.-D. (1996). Case retrieval nets: Basic ideas and extensions. *Annual Conference on Artificial Intelligence* (pp. 227-239). Springer.

Lewis, G. (2011). *Architectural Implications of Cloud Computing.* Retrieved March 17, 2012, from SEI-CMU.

Lewis, J. (1992). Psychometric evaluation of the post-study system usability questionnaire: The PSSUQ. *Proceedings of the Human Factors Society 36th Annual Meeting* (pp. 1259–1263). Santa Monica, CA: Human Factors Society.

Li, H., and Jeng, J.-J. (2010). CCMarketplace: a marketplace model for a hybrid cloud. *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research* (pp. 174-183). IBM Corp.

Li, J., Zheng, X.-L., Chen, S.-T., Song, W.-W., and Chen, D.-r. (2014). An efficient and reliable approach for quality-of-service-aware service composition. *Information Sciences*, **269:** 238-254.

Liang, T.-P., and Lai, H.-J. (2002). Effect of store design on consumer purchases: an empirical study of on-line bookstores. *Information and Management*, **39 (6):** 431–444.

Liang, T.-P., Lai, H.-J., and Ku, Y.-C. (2006). Personalized content recommendation and user satisfaction: Theoretical synthesis and empirical findings. *Journal of Management Information Systems*, **23 (3):** 45-70.

Liu, L., Chen, M., and Huang, B. (2012). Analysis of user experience at B2C e-commerce website. *Proceedings of the 2012 2nd International Conference on Computer and Information Application (ICCIA 2012)* (pp. 2-4). Atlantis Press.

Ludwig, S. A. (2012). Clonal selection based genetic algorithm for workflow service selection. *2012 IEEE Congress on Evolutionary Computation* (pp. 1-7). IEEE.

Lurie, N., and Mason, C. (2007). Visual Representation: implications for decision making. *Journal of Marketing*, **71 (1):** 160-177.

Ma, H., and Hu, Z. (2014). Cloud service recommendation based on trust measurement using ternary interval numbers. *Proceedings of International Conference on Smart Computing (SMARTCOMP)* (pp. 21-24). IEEE.

Mamoon, M. H., El-Bakry, H. M., Salama, A. A., and Mastorakis, N. (2013). Visualization of retrieved information: A survey. *WSEAS International Conference. Proceedings. Recent Advances in Computer Engineering Series* (pp. 152-166). WSEAS.

Martens, B., Teuteberg, F., and Gräuler, M. (2011). Design and implementation of a community platform for the evaluation and selection of cloud computing services: A Market Analysis. *ECIS 2011 PROCEEDINGS.* (Paper No: 215).

Mell, P., and Grance, T. (2011, September). *The NIST Definition of Cloud Computing.* Retrieved October 26, 2012, from National institute of standards and technology website: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Menychtas, A., Gomez, S. G., Giessmann, A., Gatzioura, A., Stanoevska, K., Vogel, J., et al. (2011). A marketplace framework for trading cloud-based services. *Proceedings of the 8th international conference on Economics of Grids, Clouds, Systems, and Services* (pp. 76-89). Springer-Verlag.

Menychtas, A., Vogel, J., Giessmann, A., Gatzioura, A., Garcia Gomez, S., Moulos, V., et al. (2014). 4CaaSt marketplace: An advanced business environment for trading cloud services. *Future Generation Computer Systems*, **41:** 104–120.

Mikhailov, L. (2003). Deriving priorities from fuzzy pairwise comparison judgments. *Fuzzy Sets and Systems*, **134 (3):** 365–385.

Mikhailov, L., and Tsvetino, P. (2004). Evaluation of services using a fuzzy analytic hierarchy process. *Applied Soft Computing*, **5 (1):** 23-33.

Millet, I. (1997). The effectiveness of alternative preference elicitation methods in the analytic hierarchy process. *Journal of Multi-Criteria Decision Analysis*, **6 (1):** 41-51.

Mirmotalebi, R., Ding, C., and Chi, C.-H. (2012). Modeling user's non-functional preferences for personalized service ranking. In *Service-Oriented Computing* (pp. 359-373). Berlin Heidelberg: Springer-Verlag.

Moere, A. V., and Purchase, H. (2011). On the role of design in information visualization. *Information Visualization*, **10 (4):** 356-371.

Mohabbati, B., Gašević, D., Hatala, M., Asadi, M., Bagheri, E., and Bošković, M. (2011). A Quality Aggregation Model for Service-Oriented Software Product Lines Based on Variability and Composition Patterns. *International Conference on Service-Oriented Computing* (pp. 436-451). Springer.

Mu, B., Li, S., and Yuan, S. (2014). QoS-Aware cloud service selection based on uncertain user preference. *International Conference on Rough Sets and Knowledge Technology* (pp. 589-600). Springer International Publishing.

Nestor, D., O'Malley, L., Healy, P., Quigley, A., and Thiel, S. (2007). Visualisation techniques to support derivation tasks in software product line development. *Proceedings of the 2007 conference of the center for advanced studies on Collaborative research* (pp. 315-325). IBM Corp.

North, C., and Shneiderman, B. (2000). Snap-together visualization: Evaluating coordination usage and construction. *Int'l Journal of Human-Computer Studies special issue on Empirical Studies of Information Visualization*, **53 (5):** 715-739.

O'Hagan, M. (1993). *A fuzzy decison maker.* Retrieved June 10, 2015, from Proceedings of Fuzzy Logic '93 (Computer): https://pdfs.semanticscholar.org/09ba/3f0a63b5932a6a47f8f9b4840baeee763f40.pdf

OASIS. (2007, April 11). *Web Services Business Process Execution Language*. Retrieved February 27, 2011, from OASIS: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html#_Toc164738475

Obata, T., and Ishii, H. (2003). A method for discriminating efficient candidates with ranked voting data. *European Journal of Operational Research*, **151 (1):** 233-237.

Oehlert, G. W. (2010). *A first course in design and analysis of experiments.* New York: W.H Freeman. ISBN: 0-7167-3510-5.

Oltean, G. (2004). Multiobjective fuzzy optimization method. *Scientific Bulletin of the Politechnica University of Timisoara, Trans. on Electronics and Communications*, **49 (63):** 220-225.

Pajic, D. (2014). Browse to search, visualize to explore: Who needs an alternative information retrieving model? *Computers in Human Behavior*, **39:** 145-153.

Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2007). Service-Oriented Computing: State of the art and research challenges. *Computer*, **40 (11):** 38 - 45.

Papazoglou, M., and van den Heuvel, W.-J. (2011). Blueprinting the cloud. *IEEE Internet Computing*, **15 (6):** 74-79.

Patiniotakis, I., Verginadis, Y., and Mentzas, G. (2014). Preference-based cloud service recommendation as a brokerage service. *Proceedings of the 2nd International Workshop on CrossCloud Systems* (pp. 1-5). ACM.

Pillat, R. M., Valiati, E. R., and Freitas, C. M. (2005). Experimental study on evaluation of multidimensional information visualization techniques. *Proceedings of the 2005 Latin American conference on Human-computer interaction* (pp. 20-30). ACM.

Pirolli, P., Card, S. K., and Van Der Wege, M. M. (2003). The effects of information scent on visual search in the hyperbolic tree browser. *ACM Transactions on Computer-Human Interaction (TOCHI)*, **10 (1):** 20-53.

Plaisant, C., Carr, D. A., and Shneiderman, B. (1994). *Image browsers: Taxonomy, guidelines, and informal specifications.* Technical Report CS-TR-3282 Dept. of Computer Science at Univ. of Maryland.

Pleuss, A., Rabiser, R., and Botterweck, G. (2011). Visualization Techniques for Application in Interactive Product Configuration. *Proceedings of the 15th International Software Product Line Conference, Volume 2* (p. 22). CM.

Pu, P., Faltings, B., Chen, L., Zhang, J., and Viappiani, P. (2011). Usability guidelines for product recommenders based on example critiquing research. In *Recommender Systems Handbook* (pp. 511-545). US: Springer.

Qaisar, E. J. (2012). Introduction to Cloud Computing for Developers-Key concepts, the players and their offerings. *Information Technology Professional Conference (TCF Pro IT):* (pp. 1-6). IEEE.

Qian, H., Zu, H., Cao, C., and Wang, Q. (2013). CSS: Facilitate the cloud service selection in IaaS platforms. *Proceedings of International Conference on Collaboration Technologies and Systems (CTS)* (pp. 347-354). IEEE.

Qu, C., and Buyya, R. (2014). A cloud trust evaluation system using hierarchical fuzzy inference system for service selection. *28th International Conference on Advanced Information Networking and Applications* (pp. 850-857). IEEE.

Qu, L., Wang, Y., Orgun, M., Liu, L., and Bouguettaya, A. (2014). Context-aware cloud service selection based on comparison and aggregation of user subjective assessment and objective performance assessment. *IEEE International Conference on Web Services (ICWS)* (pp. 81-88). IEEE.

Quinton, C., Duchien, L., Heymans, P., Mouton, S., and Charlier, E. (2012). Using feature modelling and automations to select among cloud solutions. *Proceedings of the Third International Workshop on Product LinE Approaches in Software Engineering* (pp. 17-20). IEEE.

Quinton, C., Haderer, N., Rouvoy, R., and Duchien, L. (2013). Towards multi-cloud configurations using feature models and ontologies. *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds* (pp. 21-26). ACM.

Quinton, C., Romero, D., and Duchien, L. (2014). Automated selection and configuration of cloud environments using software product Lines principles. *IEEE 7th International Conference on Cloud Computing (CLOUD)* (pp. 144-151). IEEE.

Rabiser, R., Wolfinger, R., and Grunbacher, P. (2009). Three-level customization of software products Using a product line approach. *42nd International Conference on System Sciences HICSS'09* (pp. 1-10). Hawaii: IEEE.

Raman, R., Livny, M., and Solomon, M. (1998). Matchmaking: Distributed resource management for high throughput computing. *The Seventh International Symposium on High Performance Distributed Computing, 1998. Proceedings.* (pp. 140-146). IEEE.

Rehman, Z. u., Hussain, F., and Hussainz, O. K. (2011). Towards multi-criteria cloud service selection. *Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* (pp. 44-48). IEEE.

Rehman, Z., Hussain, O. K., and Hussain, F. K. (2012). IaaS cloud selection using MCDM methods. *e-Business Engineering (ICEBE): 2012 IEEE Ninth International Conference on* (pp. 246-251). IEEE.

Rehman, Z., Hussain, O. K., and Hussain, F. K. (2014). Parallel cloud service selection and ranking based on QoS history. *International Journal of Parallel Programming*, **42 (5):** 820-852.

Riemer, K., and Totz, C. (2003). The many faces of personalization-an integrative economic overview. In M. Tseng, and F. Piller, *The Customer Centric Enterprise* (pp. 35-50). Springer.

Rimal, B. P., Jukan, A., Katsaros, D., and Goeleven, Y. (2011). Architectural requirements for cloud computing systems: An enterprise cloud approach. *Journal of Grid Computing*, **9 (1):** 3-26.

Ross, P. K., and Blumenstein, M. (2015). Cloud computing as a facilitator of SME entrepreneurship. *Technology Analysis and Strategic Management*, **27 (1):** 87-101.

Roy, B. (1991). The outranking approach and the foundations of ELECTRE methods. *Theory and decision*, **31:** 49-73.

Ruiz-Alvarez, A., and Humphrey, M. (2011). An automated approach to cloud storage service selection. *Proceedings of the 2nd international workshop on Scientific cloud computing* (pp. 39-48). ACM.

Saaty, T. L. (1990). *Decision making for leaders. The Analytic Hierarchy Process for decisions in a complex world.* RWS publications. ISBN: 096203178X.

Saaty, T. L. (1980). *The analytic hierarchy process: planning, priority setting, resources allocation.* New York: McGraw. ISBN: 0070543712.

Saaty, T. (1988). What is the analytic hierarchy process? In G. Mitra, H. Greenberg, F. Lootsma, M. Rijkaert, and H. Zimmermann (Eds.): *Mathematical Models for Decision Support* (pp. 109-121). Springer.

Saaty, T., and Sodenkamp, M. (2010). The analytic hierarchy and analytic network measurement processes: the measurement of intangibles. In *Handbook of Multicriteria Analysis* (pp. 91-166). Springer.

Sahri, S., Moussa, R., Long, D. D., and Benbernou, S. (2014). DBaaS-Expert: A recommender for the selection of the right cloud database. *International Symposium on Methodologies for Intelligent Systems* (pp. 315-324). Springer.

Salesforce.com. (2000-2015). Retrieved September 9, 2015, from Salesforce.com Website: http://www.salesforce.com/

Sanchez, S. M. (2005). Work smarter, not harder: guidelines for designing simulation experiments. *Proceedings of the 37th conference on Winter simulation* (pp. 69-82). Winter Simulation Conference.

Saripalli, P., and Pingali, G. (2011). MADMAC: Multiple attribute decision Methodology for adoption of clouds. *IEEE International Conference on Cloud Computing (CLOUD)* (pp. 316-323). IEEE.

Schäfer, R. (2001). Rules for using multi-attribute utility theory for estimating a user's interests. *Ninth Workshop Adaptivitat und Benutzermodellierung in Interaktiven Softwaresystemen*, (pp. 8-10).

Scheibehenne, B., Grifeneder, R., and Todd, P. (2010). Can there ever be too many options? A meta-analytic review of choice overload. *Journal of Consumer Research*, **37 (3):** 409-425.

Schubert, P., and Ginsburg, M. (2000). Virtual communities of transaction: The role of personalization in electronic commerce. *Electronic Markets*, **10 (1):** 45-55.

Schulz-Hofen, J. (2007). Web service middleware - an infrastructure for near future real life web service ecosystems. *IEEE International Conference on Service-Oriented Computing and Applications* (pp. 261-270). IEEE.

Schwartz, B. (2004). *Paradox of choice.* New York: Ecco. ISBN: 149151423X.

Sebrechts, M. M., Cugini, J. V., Laskowski, S. J., Vasilakis, J., and Miller, M. S. (1999). Visualization of search results: a comparative evaluation of text, 2D, and 3D interfaces. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 3-10). ACM.

Shezi, T., Jembere, E., Adigun, M., and Nene, M. (2012). Analysis of Open Source Enterprise Service Buses toward Supporting Integration in Dynamic Service Oriented Environments. *International Conference on e-Infrastructure and e-Services for Developing Countries*, (pp. 115-125).

Shneiderman, B. (1987). *Designing the user interface.* Addison Wesley. ISBN: 0201694972.

Shneiderman, B. (1994). Dynamic queries for visual information seeking. *IEEE Software*, **11 (6):** 70-77.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. *Proceedings of IEEE Symposium on Visual Languages* (pp. 336--343). IEEE.

Soltani, S., Asadi, M., Gasevic, D., Hatala, M., and Bagheri, E. (2012). Automated planning for feature model configuration based on functional and non-functional requirements. *Proceedings of the 16th International Software Product Line Conference-Volume 1* (pp. 56-65). ACM.

Song, H., Qi, Y., Tian, X., and Xu, D. (2007). Navigating and visualizing long lists with fisheye view and graphical representation. *Second Workshop on Digital Media and its Application in Museum and Heritages* (pp. 123-128). IEEE.

Spence, R. (2014). *Information visualization: An introduction* (3rd ed.). Springer. ISBN: 3319073400.

Sultan, N. A. (2011). Reaching for the "cloud": How SMEs can manage. *International journal of information management*, **31 (3):** 272-278.

Sun, C.-C. (2010). A performance evaluation model by integrating fuzzy AHP and fuzzy TOPSIS methods. *Expert Systems with Applications*, **37:** 7745–7754.

Sun, L., Dong, H., Hussain, F. K., Hussain, O. K., and Chang, E. (2014). Cloud service selection: State-of-the-art and future research directions. *Journal of Network and Computer Applications*, **45:** 134-150.

Sun, L., Dong, H., Hussain, F. K., Hussain, O. K., Ma, J., and Zhang, Y. (2014). A hybrid fuzzy framework for cloud service selection. *IEEE International Conference on Web Services (ICWS)* (pp. 313-320). IEEE.

Sundar, S. S., Bellu, S., Oh, J., Xu, Q., and Jia, H. (2014). User experience of on-screen interaction techniques: An experimental investigation of clicking, sliding, zooming, hovering, dragging, and flipping. *Human--Computer Interaction*, **29 (2):** 109-152.

Sundareswaran, S., Squicciarini, A., and Lin, D. (2012). A brokerage-based approach for cloud service selection. *Fifth International Conference on Cloud Computing* (pp. 558-565). IEEE.

Tajvidi, M., Ranjan, R., Kolodziej, J., and Wang, L. (2014). Fuzzy cloud service selection framework. *IEEE 3rd International Conference on Cloud Networking (CloudNet)* (pp. 443-448). IEEE.

Takeda, H., Hamada, S., Tomiyama, T., and Yoshikawa, H. (1990). A cognitive approach of the analysis of design processes. *Design theory and methodology-DTM*, **27:** 153-160.

Tang, J., Wang, D., Fung, R. Y., and Yung, K.-L. (2004). Understanding of fuzzy optimization: theories and methods. *Journal of Systems Science and Complexity*, **17 (1):** 117-136.

Teoh, S. T., and Ma, K.-L. (2005). Hifocon: Object and Dimensional Coherence and Correlation in Multidimensional Visualization. In G. Bebis, R. Boyle, D. Koracin, and B. Parvin (Eds.): *Advances in Visual Computing* (pp. 235-242). Springer.

Toffler, A. (1970). *Future shock.* New York: Amereon Ltd. ISBN: 0553277375.

Torfi, F., Farahani, R. Z., and Rezapour, S. (2010). Fuzzy AHP to determine the relative weights of evaluation criteria and fuzzy TOPSIS to rank the alternatives. *Applied Soft Computing*, **10 (2):** 520–528.

Townsend, C., and Kahn, B. E. (2014). The “visual preference heuristic”: the influence of visual versus verbal depiction on assortment processing, perceived variety, and choice overload. *Journal of Consumer Research*, **40 (5):** 993-1015.

Travis, D. (2008, March 3). *Measuring satisfaction: Beyond the usability questionnaire*. Retrieved September 20, 2015, from User Focus Website: http://www.userfocus.co.uk/articles/satisfaction.html

Triantaphyllou, E. (2013). *Multi-criteria decision making methods: a comparative study.* Springer Science and Business Media. ISBN: 0792366077.

Turner, C., Lewis, J., and Nielsen, J. (2006). Determining usability test sample size. In W. Karwowski, *International Encyclopedia of Ergonomics and Human Factors* (pp. 3084-3088). Boca Raton: CRC Press.

Valiati, E. (2005). *Taxonomia de Tarefas para Técnicas de Visualização de Informações Multidimensionais. Porto Alegr.* Retrieved October 7, 2016, from Porto Alegre, PPGC/UFRGS Technical Report: http://www.inf.ufrgs.br/~carla/papers/EValiati.pdf

Van Laarhoven, P., and Pedrycz, W. (1983). A fuzzy extension of Saaty’s priority theory. *Fuzzy sets and Systems*, **11 (1-3):** 229-241.

Van Schaik, P., and Ling, J. (2008). Modelling user experience with web sites: Usability, hedonic value, beauty and goodness. *Interacting with Computers*, **20:** 419-432.

Venesaar, U., and Loomets, P. (2006). The Role of entrepreneurship in economic development and implications for SME policy in Estonia. *14th Nordic Conference on Small Business Research.* Sweden: Unplublished Paper.

Vigne, R., Mach, W., and Schikuta, E. (2013). Towards a smart webservice marketplace. *IEEE 15th Conference on Business Informatics (CBI)* (pp. 208-215). IEEE.

Vigne, R., Mangler, J., Schikuta, E., and Rinderle-Ma, S. (2012). A structured marketplace for arbitrary services. *Future Generation Computer Systems*, **1 (28):** 48-57.

Walker, J., Borgo, R., and Jones, M. W. (2016). TimeNotes: A study on effective chart visualization and interaction techniques for time-series data. *IEEE transactions on visualization and computer graphics*, **22 (1):** 549-558.

Wang, S., Liu, Z., Sun, Q., Zou, H., and Yang, F. (2014). Towards an accurate evaluation of quality of cloud service in service-oriented cloud computing. *Journal of Intelligent Manufacturing*, **25 (2):** 283-291.

Wang, T. C., Lee, H. D., and Chang, M. (2007). A fuzzy TOPSIS approach with entropy measure for decision making problem. *IEEE International Conference on Industrial Engineering and Engineering Management* (pp. 124–128). IEEE.

Whaiduzzaman, M., Gani, A., Anuar, N. B., Shiraz, M., Haque, M. N., and Haque, I. T. (2014). Cloud service selection using multicriteria decision analysis. *The Scientific World Journal*, **2014:** Article ID 459375. DOI:10.1155/2014/459375.

White, J., Schmidt, D. C., Benavides, D., Trinidad, P., and Ruiz–Cortés, A. (2008). Automated diagnosis of product-line configuration errors in feature m. *Proceedings of 12th International Software Product Line Conference* (pp. 225 - 234). IEEE Computer Society.

Wilson, D. R., and Martinez, T. R. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, **6:** 1-34.

Wittern, E., Kuhlenkamp, J., and Menzel, M. (2012). Cloud service selection based on variability modeling. *Proceedings of the 10th international conference on Service-Oriented Computing*, (pp. 127-141).

Wohlin, C. R., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering.* Springer Science and Business Media. ISBN: 3642290434.

Wong, W., Bartels, M., and Chrobot, N. (2014). Practical eye tracking of the ecommerce website user experience. In C. Stephanidis, and M. Antona (Eds.): *Universal Access in Human-Computer Interaction. Design for All and Accessibility Practice* (Vol. 8516, pp. 109-118). Springer International Publishing.

Yager, R. (1977). Multiple objective decision-making using fuzzy sets. *International Journal of Man-Machine Studies*, **9:** 375-382.

Yang, C. C., Chen, H., and Hong, K. (2003). Visualization of large category map for internet browsing. *Decision support systems*, **35 (1):** 89-102.

Yang, C., and Chen, B. (2004). Key quality performance evaluation using fuzzy AHP. *Journal of the Chinese Institute of Industrial Engineers*, **21 (6):** 543–550.

Ye, Z., Zhou, X., and Bouguettaya, A. (2011). Genetic algorithm based QoS-aware service compositions in cloud computing. *International Conference on Database Systems for Advanced Applications* (pp. 321-334). Springer.

Yoon, K. (1987). A reconciliation among discrete compromise situations. *Journal of Operational Research Society*, **38:** 277–286.

Yoon, K. P., and Hwang, C.-L. (1995). *Multiple attribute decision making: an introduction.* Sage publications. ISBN: 0803954867.

Yu, Q. (2014). CloudRec: a framework for personalized service recommendation in the cloud. *Knowledge and Information Systems*, **43 (2):** 417-443.

Yu, T., and Lin, K.-J. (2005). Service selection algorithms for composing complex services with multiple QoS constraints. *Proceedings of the International Conference on Service-Oriented Computing-ICSOC 2005*, (pp. 130–143).

Yu, Z., and Zhang, L. (2014). QoS-aware SaaS services Selection with interval numbers for group user. *Journal of Software*, **9 (3):** 553-559.

Zadeh, L. A. (1974). The concept of a linguistic variable and its application to approximate reasoning. *Learning systems and intelligent robots*, **8 (9):** 1-10.

Zanakis, S. H., Solomon, A., Wishart, N., and Dublish, S. (1998). Multi-attribute decision making: A simulation comparison of select methods. *European journal of operational research*, **107 (3):** 507-529.

Zeng, W., Zhao, Y., and Zeng, J. (2009). Cloud service and service selection algorithm research. *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation* (pp. 1045-1048). ACM.

Zhang, L.-J., and Zhou, Q. (2009). CCOA: Cloud computing open architecture. *IEEE International Conference on Web Services* (pp. 607-616). IEEE.

Zhang, M., Ranjan, R., Haller, A., Georgakopoulos, D., and Strazdins, P. (2012). Investigating decision support techniques for automating cloud service selection. *IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. 759-764). IEEE.

Zhang, M., Ranjan, R., Nepal, S., Menzel, M., and Haller, A. (2012). A declarative recommender system for cloud infrastructure services selection. *9th International Conference on Economics of Grids, Clouds, Systems, and Services* (pp. 102-113). Berlin, Germany: Springer-Verlag.

Zhao, X., Wen, Z., and Li, X. (2014). QoS-aware web service selection with negative selection algorithm. *Knowledge and Information Systems*, **40 (2):** 349-373.

Zheng, Z., Ma, H., Lyu, M. R., and King, I. (2011). QoS-aware web service recommendation by collaborative filtering. *IEEE Transactions on Services Computing*, **4 (2):** 140-152.

Zhu, K., Shang, J., and Yang, S. (2012). *The triangular fuzzy AHP: Fallacy of the popular extent analysis method.* Retrieved May 16, 2015, from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2078576

Zimmermann, H.-J. (1975). Description and optimization of fuzzy system. *International Journal of General System*, **2 (1):** 209-215.

Zimmermann, H.-J. (2010). Fuzzy Set Theory. *Wiley Interdisciplinary Reviews: Computational Statistics*, **2 (3):** 317-332.

Zoss, A. (2015, December 8). *Introduction to data visualization: Visualization types.* Retrieved February 17, 2016, from Duke University Library: http://guides.library.duke.edu/datavis/vis_types

APPENDIX A: LIST OF PUBLICATIONS SO FAR FROM THE THESIS

Ezenwoke, A., Olawande, D., and Adigun, M. (2017). Towards a Fuzzy-Oriented Framework for Service Selection in Cloud e-Marketplaces. *The 7th International Conference on Cloud Computing and Services Science* (*CLOSER2017*).

Ezenwoke, A., Olawande, D., and Adigun, M. (2017). Towards a Constraint-based Approach for Service Aggregation and Selection in Cloud e-Marketplaces. *Future Technologies Conference* (*FTC2017*).

Ezenwoke, A., Olawande, D., and Adigun, M. (2017). A visualization framework for service selection in cloud e-marketplace. *13th IEEE World Congress on Services (SERVICES2017).*

Appendix B: Data used in Experiments

B.1. Queries used in Experiment-2

Table B1.1: Minimum QoS Values, Maximum QoS Values and 5 Test Queries for Dataset, n=100

	Availability	Response Time	Reliability	Cost
Min	23	42.5	42	100.28
Max	100	4207.5	83	498.21
Query1	92.33	1478.33	53.41	467.4
Query2	52.49	3580.85	46.15	116.64
Query3	90.89	3346.46	67.05	177.94
Query4	77.94	3855.53	64.64	299.05
Query5	40.83	898.46	66.6	453.86

Table B1.2: Minimum QoS Values, Maximum QoS Values and 5 Test Queries for Dataset, n=350

	Availability	Response Time	Reliability	Cost
Min	9	42.5	42	100.79
Max	100	4637.61	89	497.86
Query1	58.89	1587.48	81.58	323.47
Query2	38.88	1790.82	48.16	453.56
Query3	16.25	3889.45	69.64	463.36
Query4	92.17	4247.19	76.43	479.14
Query5	59.83	909.44	79.12	332.55

Table B1.3: Minimum QoS Values, Maximum QoS Values and 5 Test Queries for Dataset, n=750

	Availability	Response Time	Reliability	Cost
Min	8	40	33	103.2
Max	100	4758	89	499.54
Query1	21.69	2140.04	81.38	318.1
Query2	24.6	3079.98	34.01	456.6
Query3	22.97	2846.15	76.23	412.06
Query4	94.91	2551.7	35.14	131.65
Query5	10.86	2651.62	67.95	316.65

Table B1.4: Minimum QoS Values, Maximum QoS Values and 5 Test Queries for Dataset, n=1000

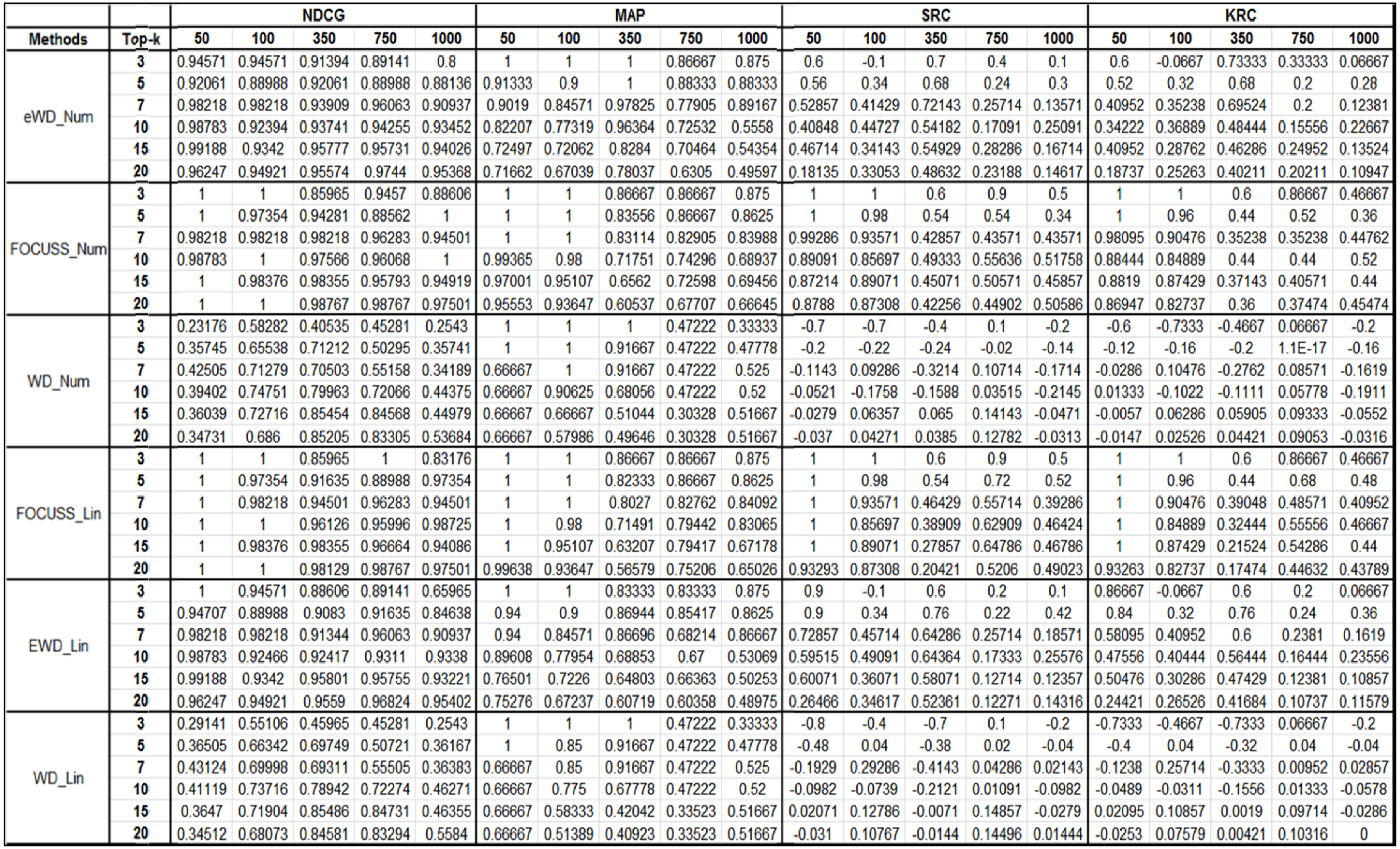
	Availability	Response Time	Reliability	Cost
Min	7	37	33	101.5
Max	100	4989.67	89	499.9
Query1	52.23	960.51	44.33	200.64
Query2	13.66	903.83	50.41	433.1
Query3	23.07	3984.38	63.61	142.07
Query4	70.26	1853.95	62.41	377.11
Query5	14.28	1292.89	84.57	440.49

B.2. Data obtained for Exeperiment-1

Table B2.1: Execution time in milliseconds for top-10 rankings

#Trials	50 Services	100 Services	350 Services	750 Services	1000 Services
Trial1	359	313	344	406	360
Trial2	343	360	328	390	328
Trial3	313	312	328	376	359
Trial4	344	343	339	344	344
Trial5	328	328	359	344	359
Trial6	343	378	340	344	328
Trial7	368	376	344	328	359
Trial8	328	328	312	328	344
Trial9	333	328	375	359	344
Trial10	391	344	312	359	344
Trial11	313	313	328	328	391
Trial12	328	328	391	344	343
Trial13	312	329	329	328	343
Trial14	313	313	344	328	343
Trial15	313	359	328	360	344
Trial16	336	328	438	375	359
Trial17	344	328	344	344	344
Trial18	328	399	359	359	329
Trial19	328	375	328	344	329
Trial20	360	359	329	344	328
Trial21	312	312	344	313	359
Trial22	359	328	390	338	344
Trial23	344	391	344	328	359
Trial24	343	343	313	313	328
Trial25	336	343	323	328	343
Trial26	328	328	328	312	359
Trial27	328	328	328	375	406
Trial28	328	344	312	344	391
Trial29	375	328	328	328	329
Trial30	328	328	375	328	343
Mean Execution time	**336.8666667**	**340.4667**	**342.8**	**344.6333**	**349.4333**

B.3. Data obtained from simulation Experiment-2



		NDCG					MAP					SRC					KRC				
Methods	Top-k	50	100	350	750	1000	50	100	350	750	1000	50	100	350	750	1000	50	100	350	750	1000
eWD_Num	3	0.94571	0.94571	0.91394	0.89141	0.8	1	1	1	0.86667	0.875	0.6	-0.1	0.7	0.4	0.1	0.6	-0.0667	0.73333	0.33333	0.06667
	5	0.92061	0.88988	0.92061	0.88988	0.88136	0.91333	0.9	1	0.88333	0.88333	0.56	0.34	0.68	0.24	0.3	0.52	0.32	0.68	0.2	0.28
	7	0.98218	0.98218	0.93909	0.96063	0.90937	0.9019	0.84571	0.97825	0.77905	0.89167	0.52857	0.41429	0.72143	0.25714	0.13571	0.40952	0.35238	0.69524	0.2	0.12381
	10	0.98783	0.92394	0.93741	0.94255	0.93452	0.82207	0.77319	0.96364	0.72532	0.5558	0.40848	0.44727	0.54182	0.17091	0.25091	0.34222	0.36889	0.48444	0.15556	0.22667
	15	0.99188	0.9342	0.95777	0.95731	0.94026	0.72497	0.72062	0.8284	0.70464	0.54354	0.46714	0.34143	0.54929	0.28286	0.16714	0.40952	0.28762	0.46286	0.24952	0.13524
	20	0.96247	0.94921	0.95574	0.9744	0.95368	0.71662	0.67039	0.78037	0.6305	0.49597	0.18135	0.33053	0.48632	0.23188	0.14617	0.18737	0.25263	0.40211	0.20211	0.10947
FOCUSS_Num	3	1	1	0.85965	0.9457	0.88606	1	1	0.86667	0.86667	0.875	1	1	0.6	0.9	0.5	1	1	0.6	0.86667	0.46667
	5	1	0.97354	0.94281	0.88562	1	1	1	0.83556	0.86667	0.8625	1	0.98	0.54	0.54	0.34	1	0.96	0.44	0.52	0.36
	7	0.98218	0.98218	0.98218	0.96283	0.94501	1	1	0.83114	0.82905	0.83988	0.99286	0.93571	0.42857	0.43571	0.43571	0.98095	0.90476	0.35238	0.35238	0.44762
	10	0.98783	1	0.97566	0.96068	1	0.99365	0.98	0.71751	0.74296	0.68937	0.89091	0.85697	0.49333	0.55636	0.51758	0.88444	0.84889	0.44	0.44	0.52
	15	1	0.98376	0.98355	0.95793	0.94919	0.97001	0.95107	0.6562	0.72598	0.69456	0.87214	0.89071	0.45071	0.50571	0.45857	0.8819	0.87429	0.37143	0.40571	0.44
	20	1	1	0.98767	0.98767	0.97501	0.95553	0.93647	0.60537	0.67707	0.66645	0.8788	0.87308	0.42256	0.44902	0.50586	0.86947	0.82737	0.36	0.37474	0.45474
WD_Num	3	0.23176	0.58282	0.40535	0.45281	0.2543	1	1	1	0.47222	0.33333	-0.7	-0.7	-0.4	0.1	-0.2	-0.6	-0.7333	-0.4667	0.06667	-0.2
	5	0.35745	0.65538	0.71212	0.50295	0.35741	1	1	0.91667	0.47222	0.47778	-0.2	-0.22	-0.24	-0.02	-0.14	-0.12	-0.16	-0.2	1.1E-17	-0.16
	7	0.42505	0.71279	0.70503	0.55158	0.34189	0.66667	1	0.91667	0.47222	0.525	-0.1143	0.09286	-0.3214	0.10714	-0.1714	-0.0286	0.10476	-0.2762	0.08571	-0.1619
	10	0.39402	0.74751	0.79963	0.72066	0.44375	0.66667	0.90625	0.68056	0.47222	0.52	-0.0521	-0.1758	-0.1588	0.03515	-0.2145	0.01333	-0.1022	-0.1111	0.05778	-0.1911
	15	0.36039	0.72716	0.85454	0.84568	0.44979	0.66667	0.66667	0.51044	0.30328	0.51667	-0.0279	0.06357	0.065	0.14143	-0.0471	-0.0057	0.06286	0.05905	0.09333	-0.0552
	20	0.34731	0.686	0.85205	0.83305	0.53684	0.66667	0.57986	0.49646	0.30328	0.51667	-0.037	0.04271	0.0385	0.12782	-0.0313	-0.0147	0.02526	0.04421	0.09053	-0.0316
FOCUSS_Lin	3	1	1	0.85965	1	0.83176	1	1	0.86667	0.86667	0.875	1	1	0.6	0.9	0.5	1	1	0.6	0.86667	0.46667
	5	1	0.97354	0.91635	0.88988	0.97354	1	1	0.82333	0.86667	0.8625	1	0.98	0.54	0.72	0.52	1	0.96	0.44	0.68	0.48
	7	1	0.98218	0.94501	0.96283	0.94501	1	1	0.8027	0.82762	0.84092	1	0.93571	0.46429	0.55714	0.39286	1	0.90476	0.39048	0.48571	0.40952
	10	1	1	0.96126	0.95996	0.98725	1	0.98	0.71491	0.79442	0.83065	1	0.85697	0.38909	0.62909	0.46424	1	0.84889	0.32444	0.55556	0.46667
	15	1	0.98376	0.98355	0.96664	0.94086	1	0.95107	0.63207	0.79417	0.67178	1	0.89071	0.27857	0.64786	0.46786	1	0.87429	0.21524	0.54286	0.44
	20	1	1	0.98129	0.98767	0.97501	0.99638	0.93647	0.56579	0.75206	0.65026	0.93293	0.87308	0.20421	0.5206	0.49023	0.93263	0.82737	0.17474	0.44632	0.43789
EWD_Lin	3	1	0.94571	0.88606	0.89141	0.65965	1	1	0.83333	0.83333	0.875	0.9	-0.1	0.6	0.2	0.1	0.86667	-0.0667	0.6	0.2	0.06667
	5	0.94707	0.88988	0.9083	0.91635	0.84638	0.94	0.9	0.86944	0.85417	0.8625	0.9	0.34	0.76	0.22	0.42	0.84	0.32	0.76	0.24	0.36
	7	0.98218	0.98218	0.91344	0.96063	0.90937	0.94	0.84571	0.86696	0.68214	0.86667	0.72857	0.45714	0.64286	0.25714	0.18571	0.58095	0.40952	0.6	0.2381	0.1619
	10	0.98783	0.92466	0.92417	0.9311	0.9338	0.89608	0.77954	0.68853	0.67	0.53069	0.59515	0.49091	0.64364	0.17333	0.25576	0.47556	0.40444	0.56444	0.16444	0.23556
	15	0.99188	0.9342	0.95801	0.95755	0.93221	0.76501	0.7226	0.64803	0.66363	0.50253	0.60071	0.36071	0.58071	0.12714	0.12357	0.50476	0.30286	0.47429	0.12381	0.10857
	20	0.96247	0.94921	0.9559	0.96824	0.95402	0.75276	0.67237	0.60719	0.60358	0.48975	0.26466	0.34617	0.52361	0.12271	0.14316	0.24421	0.26526	0.41684	0.10737	0.11579
WD_Lin	3	0.29141	0.55106	0.45965	0.45281	0.2543	1	1	1	0.47222	0.33333	-0.8	-0.4	-0.7	0.1	-0.2	-0.7333	-0.4667	-0.7333	0.06667	-0.2
	5	0.36505	0.66342	0.69749	0.50721	0.36167	1	0.85	0.91667	0.47222	0.47778	-0.48	0.04	-0.38	0.02	-0.04	-0.4	0.04	-0.32	0.04	-0.04
	7	0.43124	0.69998	0.69311	0.55505	0.36383	0.66667	0.85	0.91667	0.47222	0.525	-0.1929	0.29286	-0.4143	0.04286	0.02143	-0.1238	0.25714	-0.3333	0.00952	0.02857
	10	0.41119	0.73716	0.78942	0.72274	0.46271	0.66667	0.775	0.67778	0.47222	0.52	-0.0982	-0.0739	-0.2121	0.01091	-0.0982	-0.0489	-0.0311	-0.1556	0.01333	-0.0578
	15	0.3647	0.71904	0.85486	0.84731	0.46355	0.66667	0.58333	0.42042	0.33523	0.51667	0.02071	0.12786	-0.0071	0.14857	-0.0279	0.02095	0.10857	0.0019	0.09714	-0.0286
	20	0.34512	0.68073	0.84581	0.83294	0.5584	0.66667	0.51389	0.40923	0.33523	0.51667	-0.031	0.10767	-0.0144	0.14496	0.01444	-0.0253	0.07579	0.00421	0.10316	0

Figure B3.1: Methods versus ranking accuracy by four metrics