

**A SIZE METRIC-BASED EFFORT ESTIMATION METHOD FOR SERVICE
ORIENTED ARCHITECTURE SYSTEMS**

SAMSON WANJALA MUNIALO

**A Thesis Submitted in Partial Fulfillment of the Requirement of the Degree of
Doctor of Philosophy in Information Technology of Masinde Muliro University of
Science and Technology**

SEPTEMBER, 2020

DECLARATION

This thesis is my own original work prepared with no other than the indicated sources and support and has not been presented elsewhere for a degree or any other award.

Samson Wanjala Munialo

DATE

SIT/LH/003/2015

CERTIFICATION

The undersigned certify that they have read and hereby recommend for acceptance of Masinde Muliro University of Science and Technology a Thesis entitled “**A Size Metric-Based Effort Estimation Method for Service Oriented Architecture Software Systems.**”

Prof. Geoffrey Muchiri Muketha

DATE

Department of Computer Science

School of Computing and Information Technology

Murang’a University of Technology

Dr. Kelvin Kabeti Omieno

DATE

Department of Information Technology and Informatics

School of Computing and Information Technology

Kaimosi Friends University College

DEDICATION

I dedicate this thesis to my wife Hellen Wasike and my sons Gerrard Munialo and Warren Wachie.

ACKNOWLEDGEMENT

I thank the almighty God for granting me the opportunity of developing this thesis and for providing me with ample environment to enable me to progress in my PhD studies.

I would like to express my gratitude to my supervisors Prof. Geoffrey Muchiri Muketha and Dr. Kelvin Kabeti Omieno for extending their knowledge and experience in guiding and shaping this thesis. I thank them for their great support, patience, motivation and constructive comments that enabled this thesis to develop in depth and in content.

I thank my wife Hellen, my sons Gerrard and Warren for their encouragement and patience. I also thank my loving parents for their continued support and prayers.

I acknowledge my fellow PhD students Mary Mwadulo and Amos Chege for their encouragement and support as we walked together through the PhD journey. I also thank members of staff school of Computing and Informatics Masinde Muliro University of Science and technology for their support that motivated me to progress smoothly in my study. I also acknowledge my friends and my colleagues in School of Computing and Informatics, Meru University and Science and Technology for their encouragement.

ABSTRACT

Service Oriented Architecture (SOA) is one of the recent software development paradigms that enable alignment of business processes into integrated services within and outside organizations regardless of the heterogeneity of technologies used. Determining the scope, effort and cost of SOA systems is important to facilitate the planning and eventually successful implementation of software projects. A number of methods have been proposed to estimate effort of building SOA projects. Despite the fact that these methods are promising, the problem of measuring SOA size and estimating SOA effort still remains largely unresolved mainly because there is limited attempt in using Unified Modeling Language (UML) size metrics to define size-based attributes for estimating SOA development effort. To address this problem, a set of size metrics were defined and effort estimation method that is based on the size metrics was developed. To automate the computation of the metric and the method, a static analysis tool that uses deep learning techniques to detect UML arrows and recognize text was constructed. The automated tool deep learning techniques were each subjected to validity checks based on datasets of 100 operation names and 100 arrow head images. Briand's theoretical validation was used to test the validity of the designed size metrics and they were found to be mathematically sound. Experimental research design was employed to sampled SOA systems to test variables used in the study and the accuracy of the proposed effort estimation method and implementation automated tool. A survey involving experts from the industry was carried out to replicate and validate the experiment done by students and to determine the appropriateness of the proposed size metrics, SOA development effort factors and the implementation automated tool. The experiment was based on a sample of 15 students' SOA projects developed by Meru University of Science and Technology students while the survey involved 20 programmers from the industry. Descriptive statistics such as Mean magnitude of relative error (MMRE) and Magnitude of Error (MRE) were used to test SOA effort estimation accuracy while linear regression analysis tested relationship among variables identified in the study. Result from the experiment revealed that the proposed metrics and method are more accurate and there is a correlation between size attributes and SOA size and between SOA size and SOA development effort. Response from the survey showed that the proposed metrics and effort factors are valid and they have influence on size and effort respectively. Findings from this study were meant to provide a basis for future software engineering researchers to develop more effective and more accurate size metrics and effort estimation methods.

TABLE OF CONTENTS

DECLARATION.....	ii
DEDICATION.....	iii
ACKNOWLEDGEMENT.....	iv
ABSTRACT.....	v
LIST OF FIGURES.....	xiv
LIST OF TABLES.....	xvi
ACRONYMS AND ABBREVIATIONS.....	xix
DEFINITION OF OPERATIONAL TERMS.....	xxi
CHAPTER ONE : INTRODUCTION.....	1
1.1 Background to the Study.....	1
1.2 Problem Statement.....	4
1.3 Objectives.....	5
1.3.1 General Objective.....	5
1.3.2 Specific Objectives.....	5
1.4 Research Questions.....	5
1.5 Hypotheses.....	5
1.6 Scope of the Study.....	6
1.7 Significance of the Study.....	7
1.8 Limitations of the Study.....	8
1.9 Assumptions of the Study.....	8
1.10 Contribution of the Thesis.....	9
1.11 Organization of the Thesis.....	9
CHAPTER TWO : LITERATURE REVIEW.....	11
2.1 Introduction.....	11
2.2 Basic Concepts of SOA.....	11
2.2.1 Evolution of SOA.....	12
2.2.2 A Typical SOA System.....	13
2.2.3 SOA layered Architecture.....	14
2.2.4 SOA Characteristics.....	15

2.2.5 Web Services	16
2.2.5.1 SOAP Web Service Standard.....	16
2.2.5.2 REST Web Service Architecture	17
2.2.6 SOA Development Methodologies	18
2.2.6.1 Service Oriented Architecture Framework (SOAF)	18
2.2.6.2 Service Oriented Modeling Architecture	19
2.2.6.3 Service Oriented Architecture Modeling Language	19
2.3 Existing Software Size Metrics.....	20
2.3.1 Traditional Software Size Metrics	20
2.3.1.1 Source Line of Code (SLOC)	21
2.3.1.2 Function Point Analysis (FPA).....	21
2.3.1.3 Story Points	23
2.3.1.4 Use Case Points (UCP)	24
2.3.1.5 Object Points	25
2.3.1.6 Object – Oriented Size and Complexity Metrics	26
2.3.2 Existing SOA Complexity and Size Metrics	27
2.3.7.1 Weighted Service Interface Count (WSIC)	27
2.3.7.2 Number of Services (NOS).....	28
2.3.7.3 COSMIC-SOA Metrics.....	28
2.4 Existing Effort Estimation Methods	30
2.4.1 Traditional Effort Estimation Methods.....	30
2.4.1.1 Expert Judgment	30
2.4.1.2 Analogy.....	31
2.4.1.3 Price-to-Win.....	31
2.4.1.4 Bottom-up and Top-up.....	32
2.4.1.5 Wideband Delphi	32

2.4.1.6 Constructive Cost Model (COCOMO)	33
2.4.1.7 Artificial Neural Network Effort Estimation Methods	36
2.4.1.8 Fuzzy Logic Effort Estimation Methods.....	37
2.4.2 Existing SOA Cost Estimation Methods.....	38
2.4.2.1 SMAT-AUS Scope, Cost and Effort Estimation Framework for SOA	38
2.4.2.2 SOA Cost Estimation for Customization to Packaged Applications	39
2.4.2.3 Effort Estimation for Web Service Composition.....	41
2.4.2.4 Software Cost Estimation Framework for SOA using Divide-and-conquer	42
2.4.2.5 Service Point Estimation Model for SOA Based Projects	43
2.4.2.6 Requirements Based Model for SOA Systems Effort Estimation	45
2.4.2.7 Estimating Size and Effort of Business Process SOA Applications.....	46
2.4.2.8 Phased Approach for Effort Estimation for SOA projects.....	47
2.5 Existing Automated Tools to Interpret design artifacts	49
2.6 Theoretical Framework	50
2.6.1 Software Effort Estimation Factors	51
2.6.1.1 Size Factor	51
2.6.1.2 Technical Factors	52
2.6.1.3 Personnel Factors	52
2.6.1.4 Requirements Factors.....	53
2.6.1.5 Service Type Factors.....	53
2.6.2 COCOMO 2.0 Model.....	54
2.7 Conceptual Framework	56
2.8 Identified Gaps.....	57
2.9 Chapter Summary	59
CHAPTER THREE: RESEARCH METHODOLOGY	60
3.1 Introduction.....	60

3.2 Research Philosophy	60
3.3 Research Design.....	61
3.4 Target population	63
3.5 Sampling and Sample size	64
3.6 Data Collection Instruments	65
3.7 Reliability and Validity of Data Collection Instruments	66
3.7.1 Validity	66
3.7.2 Reliability.....	67
3.8 Data Collection Procedure	67
3.9 Data Analysis	68
3.10 Ethical Issues	69
3.11 Chapter Summary	69
CHAPTER FOUR : DESIGN OF SIZE METRICS FOR SOA	71
4.1 Introduction.....	71
4.2 SOA Size Attributes.....	71
4.2.1 Weighted Operation Count (WOC)	72
4.2.2 Service Dependency Count (SDC)	74
4.2.3 Weighted Message Count	77
4.2.4 Weighted Service Count (WSC).....	80
4.3 Application of defined SOA size metrics in a Purchase order SOA Application.....	81
4.3.1 Purchase Order Business Processing Modeling.....	81
4.3.2 UML Diagram for Purchase Processing System	84
4.3.3 Application of SOA Size Metrics	87
4.3.3.1 WOC for Purchase Order System	87
4.3.3.2 SDC for Purchase Order System.....	89
4.3.3.3 WMC for Purchase Order System	90
4.3.3.4 WSC for Purchase Order	91
4.4 Theoretical Validation of the Proposed Metrics	91

4.4.1 Overview of Briand's.....	91
4.4.2 Results.....	92
4.5 Chapter Summary	95
CHAPTER FIVE: DESIGN OF SOA EFFORT ESTIMATION METHOD	96
5.1 Introduction.....	96
5.2 SOA Size.....	97
5.3 Service Type Factors (STF).....	99
5.3.1 Service Construction Type.....	99
5.3.1.1 Available Service.....	99
5.3.1.2 Migrated Service.....	100
5.3.1.3 New Service.....	101
5.3.2 SOA Architectural Style (SA)	102
5.3.2.1 SOAP.....	102
5.3.2.2 REST.....	102
5.3.2.3 Comparison between REST and SOAP Effort Factor	103
5.4 SOA Effort Multiplier Factors (EMF).....	104
5.4.1 Product factors	106
5.4.1.1 Database Complexity and size	107
5.4.1.2 Database complexity and Database size fuzzy effort multiplier.....	109
5.4.1.3 User Interface Complexity.....	114
5.4.1.4 Integration complexity.....	114
5.4.1.5 User inteface and Integration fuzzy logic effort multiplier	115
5.4.2 Service Development Environment	118
5.4.2.1 Web Service Development Tool Support	118
5.4.2.2 Web Service Infrastrucure Capabilities	119
5.4.2.3 Web service development tool and intrastructure fuzzy effort multiplier	120

5.4.3 Requirement Factors	123
5.4.3.1 Requirements Elicitation.....	123
5.4.3.2 Business Value.....	126
5.4.3.3 Security Requirements	128
5.4.4 Personnel Factors	130
5.4.4.1 Web Service Experience	130
5.4.4.2 Application Experience.....	132
5.4.4.3 Programming Experience.....	134
5.3.4.4 Team Cohesion	135
5.5 Effort Estimation Method Example	136
5.6 Chapter Summary	137
CHAPTER SIX : IMPLEMENTATION OF SIZE METRICS AND EFFORT ESTIMATION TOOL (SOA-SMET)	139
6.1 Introduction.....	139
6.2 Requirements of SMET	139
6.3 Architecture of SMET.....	140
6.3.1 Automated UML Feature Extraction Component.....	141
6.3.1.1 Service Operation Names Extraction.....	142
6.3.1.2 Service Operation Classification.....	144
6.3.1.3 Arrow Head Detection	145
6.3.2 Manual Entry/Display	147
6.3.3 Business Logic Layer.....	149
6.3.4 Data Layer.....	152
6.3.5 SMET Output.....	152
6.4 Chapter Summary	153

**CHAPTER SEVEN : AN EMPIRICAL VALIDATION OF PROPOSED METRICS,
EFFORT ESTIMATION METHOD AND AUTOMATED TOOL154**

7.1 Introduction.....	154
7.2 Empirical Validation Strategy.....	154
7.3 Context Definition	156
7.4 Experimental Preparation.....	156
7.5 Experimental Planning.....	157
7.5.1 Hypotheses	158
7.5.2 Threats to Validity	159
7.6 Experimental Operation.....	159
7.7 Experiment Results	162
7.7.1 SOA Size Metrics Validation Results.....	162
7.7.1.1 SOA Size Metrics descriptive analysis	163
7.7.1.2 Function Point Size Metric Descriptive Analysis	164
7.7.1.3 Correlation between Size Metrics and SOA Size	166
7.7.2 Effort Estimation Method Validation Results.....	168
7.7.2.1 Proposed Effort Estimation Method Descriptive Analysis	169
7.7.2.2 COCOMO Effort Estimation Method Descriptive Analysis	171
7.7.2.3 Correlation between Size Metrics and SOA Size	173
7.7.3 Automated Implementation Tool Accuracy Level	174
7.8 Expert Opinion Survey	175
7.8.1 Survey Preparation and Planning.....	176
7.8.2 Demographic Summary of the Respondents.....	177
7.9 Survey Results	178
7.9.1 Response on SOA Size Metric Validation.....	178
7.9.1.1 Experts’ response on service internal structure influence on SOA size	178
7.9.1.2 Experts’ response on influence of service dependency on SOA size	179

7.9.1.3 Experts’ response on influence of data movement on SOA size	179
7.9.1.4 Experts’ response on WSC and SOA size effect on effort	180
7.9.2 Response on Effort estimation factors	180
7.9.2.1 Experts’ response on influence of service type on SOA effort.....	180
7.9.2.2 Response on influence of SOA Effort Multiplier Factors (EMF) to Effort.....	181
7.9.3 Response on the Validity and Appropriateness of the Implemented Tool	182
7.10 Chapter Summary	183
CHAPTER EIGHT: SUMMARY, CONCLUSION AND RECOMMENDATIONS	185
8.1 Summary	185
8.1.1 Defining Metrics for SOA Size.....	185
8.1.2 Developing an Effort Estimation Method for SOA Projects	186
8.1.3 Automating the SOA Metrics and Effort Estimation Method	186
8.2 Conclusion	187
8.2.1 Defined SOA Size Metrics.....	187
8.2.2 Effort Estimation Method for SOA.....	188
8.2.3 Automated Implementation Tool.....	188
8.3 Recommendations for Future Work.....	188
REFERENCES.....	190
APPENDICES	196

LIST OF FIGURES

Figure 2.1: SOA Model	13
Figure 2.2: SOA n-tier layers	14
Figure 2.3 COCOMO 2.0 model variables relationship	55
Figure 2.4: Conceptual framework showing SOA effort estimation variables.....	57
Figure 4.1: Service interface diagram.....	74
Figure 4.2: UML Diagram showing dependency among services.....	76
Figure 4.3: UML sequence diagram showing data movement	79
Figure 4.4: Purchase order process Business Processing Modeling Notation.....	82
Figure 4.5: UML interface diagram representing purchase order process services.....	85
Figure 4.6: UML sequence diagram representing purchase order process services	86
Figure 5.1 : Fuzzy Effort estimation method model	106
Figure 5.2: Triangular Membership function.....	110
Figure 5.3: Database complexity fuzzification	111
Figure 5.4: Database size fuzzification.....	111
Figure 5.5 : User interface complexity fuzzification	115
Figure 5.6: Integration complexity fuzzification	116
Figure 5.7: Development tool fuzzification	120
Figure 5.8: Infrastructure capabilities fuzzification.....	121
Figure 5.9: Requirements elicitation factor	124
Figure 5.10: business value fuzzification.....	127
Figure 5.11: security requirements fuzzification	129
Figure 5.12 Web service developers' experience fuzzification.....	131

Figure 5.13 Application experience fuzzification.....	133
Figure 5.14: Programming experience fuzzification	135
Figure 6.1: SMET architecture	141
Figure 6.2: Taxi Service UML interface diagram.....	142
Figure 6.3: Highlighted operation names detected by EAST detector	143
Figure 6.4: Group of text recognized by Tesseract OCT.....	144
Figure 6.5: WOC arrow classification by ResNet50 CNN	146
Figure 6.6: WMC arrow classification by ResNet50 CNN	146
Figure 6.7 SMET Manual entry interface of SOA size attributes.....	147
Figure 6.8: SMET Effort factors input interface.....	148

LIST OF TABLES

Table 2.1 Function Point Complexity weights	22
Table 2.2: UCP Actors classification.....	24
Table 2.3 UCP Use cases classifications	24
Table 2.4: Classification of objects weight.....	26
Table 2.5: COCOMO Complexity factor weights	34
Table 2.6 Context Effort Factors	41
Table 2.7: Comparison between orchestration and choreography.....	42
Table 3.1: Research Method per objective	62
Table 3.2: Statistical analyses for each research objective	69
Table 4.1: Service Operation weight	73
Table 4.2 Weighted Service dependency weights	76
Table 4.3 Weighted message type arrows.....	78
Table 4.4: WOC for Invoice service	88
Table 4.5: WOC for Production service	88
Table 4.6: WOC for shipping service	89
Table 5.1: Intermediate COCOMO Effort coefficients	97
Table 5.2: Effort distribution among development phases	100
Table 5.3: Effort factors for SOA service types.	101
Table 5.4: Service architectural style weights	103
Table 5.5 SOA Effort multiplier Factors (EMF)	105
Table 5.6 Database complexity factor	107
Table 5.7 Database size factor	108

Table 5.8: Summary of database complexity and database size rules	112
Table 5.9 User Interface complexity.....	114
Table 5.10 Integration complexity.....	114
Table 5.11: Summary user interface complexity and integration complexity	116
Table 5.12 Web Service development tool support.....	119
Table 5.13 Infrastructure capabilities factor	120
Table 5.14: Summary of automated and infrastructure capabilities rules	121
Table 5.15: Requirements elicitation effort factors	124
Table 5.16: Business value effort factor	126
Table 5.17 Security Requirements effort multiplier	128
Table 5.18 Web service developer’s experience effort multiplier.....	130
Table 5.19 Application experience effort multiplier.....	132
Table 5.20 Programming experience effort multiplier	134
Table 5.21 Team Cohesion factor	135
Table 5.22 Purchase order SOA application EMF.....	136
Table 7.1: Data Analysis for the proposed SOA size metrics.....	163
Table 7.2 Descriptive analysis for the proposed SOA size metrics	164
Table 7.3: Data Analysis for Function point analysis.....	164
Table 7.5: Correlation between WOC metrics and SOA size.....	166
Table 7.6: ANOVA analysis correlation between WOC metrics and SOA size	167
Table 7.7: Correlation between SDC metrics and SOA size	167
Table 7.8: Correlation between WMC metrics and SOA size	167
Table 7.9: ANOVA correlation analysis between WMC metrics and SOA size.....	168
Table 7.10: Effort Estimation Analysis based the proposed method.....	170

Table 7.11: COCOMO Effort Estimation Method descriptive analysis	171
Table 7.12: Correlation between SOA Size and SOA development effort.....	173
Table 7.13: ANOVA correlation between SOA size and SOA development effort.....	173
Table 7.15 Experts' experience in Software development	177
Table 7.16: Response on service internal structure attributes influence on SOA size	178
Table 7.17: Experts' response on weights assigned to service internal structure.....	178
Table 7.18: Experts' response on weights assigned to service internal structure.....	179
Table 7.19: Experts' response on WMC attributes and weights.....	179
Table 7.20: Experts' response on influence of service type to SOA development effort...	180
Table 7.21: Experts' response on influence of EMF on SOA development effort.....	182
Table 7.22: Experts' response on appropriateness of implementation tool.....	183

ACRONYMS AND ABBREVIATIONS

API – Application Programming Interface

BPMN – Business Process Modeling notations

COCOMO - Constructive Cost Model

COM – Component Object Model

CORBA – Common Object Request Broker Architecture

COSMIC-FFP -Common Software Measurement International Consortium-Full function
Points

CORBA - Common Object Request Broker Architecture

CRM - Customer Relationship Management system

DCOM - Distributed Component Object Model

EAF - Effort adjustment factor

FC – Function Count

FP – Function Point

FPA – Function Point Analysis

FUR – Functional User Requirements

GQM – Goal/ Question/ Metric

HTTP – Hypertext Transfer Protocol

ISO – International Organization for Standardization

JSON – JavaScript Notation

MMRE - Mean magnitude of relative error

MRE – Magnitude of Relative Error

NOP – Number of objects points

OMG – Object Management Group

PHP – Hypertext Preprocessor

REST – Representational State Transfer

RMI – Remote Method Invocation

RPC – Remote Procedure Call

SDC – Service Dependency Count

SOA – Service Oriented Architecture

SOAF – Service Oriented Architecture Framework

SOAML – SOA Modeling language

SOAP – Simple Object Access Protocol

SOMA – Service Oriented Modeling Architecture

SLOC - Source line of codes

UDDI – Universal Description Discovering and Integration

UCP – Use Case Point

UFP – Unadjusted Function Point

UML – Unified Modeling Language

URL – Universal Resource Locator

WBS – Work Breakdown Structure

WMC – Weighted Message Count

WOC – Weighted Operation Count

WSDL – Web Service Description Language

WSC – Weighted Service Count

DEFINITION OF OPERATIONAL TERMS

Effort – Exertion of physical or mental power to complete a task or a project.

Project Size – The scope of a project that describes how big the project is.

Service Oriented Architecture: SOA is a paradigm for organizing and utilizing services made available to consumer services and applications over a distributed network.

Service: A service is a discoverable and self-contained software entity that interacts with applications and other services for the purpose of achieving business objectives.

Software Development Effort estimation: is the process of predicting the amount of human effort required to develop software, expressed in person-hours or person-month.

Software size metrics: Software size metric is a standard of measure of a degree to which a software system or processes possesses some size properties.

CHAPTER ONE

INTRODUCTION

1.1 Background to the Study

Software effort estimation is the process of predicting human effort required to build a software project. The bulk of the cost of software development is due to human effort estimated in terms of person-months (Borade & Khalker, 2013). Reliable effort estimation enables adherence to schedule and budget for successful resource allocation and software project implementation. The main reason for software effort estimation is to help software developers and managers to answer the question “how much effort is required to build a software project?”

The ability to measure size and estimate software effort precisely contributes to better management of IT project. Demand for more functionality, higher reliability and higher performance has resulted to higher competitiveness among software developers (Coelho & Basu, 2012). To stay competitive, software developers need to deliver software products on time, within the budget and to the agreed level of quality. Most projects fail due to planning issues such as cost, effort, time and requirements specifications. A study on software projects in 2012 shows that 43% of projects were challenged and 18% failed due to over budget, late delivery and less than required features (Standish, 2013).

One of the key indicators to be considered when estimating software development effort is software project size. Other indicators include environmental factors, technical factors and human factors. The size of the software project determines the scope and is modeled as the main input when estimating development effort. Software size metrics is a standard of

measure of a degree to which a software system possesses size based properties (Coelho & Basu, 2012). The goal of using size metrics in software engineering is to obtain objective and quantifiable measurements which may have valuable applications in estimating software development effort. Software developers and Software project managers must tell “how big is the software project” before estimating software development effort.

Software developers and software project managers have had the interest of estimating accurately the size, effort and cost of developing software products. Earlier effort estimation methods were based on software lines of codes or function points to estimate the size (Litoriya & Kothari, 2013). However, demand for new functionalities and inclusion of new features such as software re-use, distributed systems and iterative development established a need for new software size, effort and cost estimation methods (Prokopova & Silhavy, 2015).

Furthermore, most software size metrics and software effort estimation methods are not automated due to the fact that they do not use artifacts such as Unified Modeling Language to expose software attributes that are relevant to software size (Harizi, 2012). UML not only provide a view of the system for design purpose but also reveal the scope or size of a software system. Use of diagrams such as UML to reveal attributes that determine software size provides an opportunity to capture attributes that are relevant to computing software size and effort from UML automatically. Automation of software size attributes extraction from UML provides a more efficient way of computing software size and software development effort.

Service oriented Architecture (SOA) is an example of a popular paradigm for developing distributed systems that provides a challenge to existing software effort estimation techniques. SOA consists of service providers which are elements that offer services to be used by other service users. The need for agility, cost-effectiveness and efficiency, adaptability and legacy leverage in the rapidly changing business environment has led many organizations to migrate to SOA applications. SOA other benefit include clear separation of services from implementation which allows service upgrades to occur without overhaul of the entire system and less impact on service users (Farrag & Moawad, 2014). SOA is an important aspect of organization's IT infrastructure as it links all applications and services that support business processes. SOA is a paradigm shift in designing information systems where services corresponding to business functions are published in form of standard interface to be discovered by other services.

Estimating SOA systems development effort is difficult because they consist of integration among services within and outside the organization regardless of heterogeneous technology and programming language over a distributed network. Secondly, there are different types of services including new, migrated and discovered and factors that affect SOA effort are different from other software applications. Furthermore, SOA embraces the principle of software reuse and enabling modification of legacy systems to suit today's business needs. A number of research studies have attempted to introduce effort estimation methods for SOA. However, existing SOA effort estimation methods do not rely on size metrics in their effort estimation.

1.2 Problem Statement

Over decades, researchers have attempted to introduced software size metrics and effort estimation methods. Traditional software size metrics such as Source Line of Code (SLOC) and Function Point (Albrecht, 1983) are challenged when dealing with SOA applications due to SOA architectural difference when compared to other software paradigm. This prompted researchers to introduce size metrics specifically for SOA but still they did not capture all key SOA size attributes (Zhang & Li, 2009; Hirzalla, Cleland-Huang & Arsanjani, 2009; Elhag & Mohamad, 2014; COSMIC, 2010). On the other hand, existing traditional effort estimation methods such as Constructive Cost Models (Boehm, 1981; Boehm, 2000), Artificial Neural Network Effort Estimation Methods (Bawa & Chawla, 2012; Rijwani & Jain, 2016) and Fuzzy Logic Effort Estimation Methods (Thamarai & Murugavalli, 2015; Patra & Rajnish, 2018) are also unable to estimate effort for SOA due to SOA effort factors which are different from traditional software effort factors. In an attempt to estimate SOA development effort, researchers have proposed SOA effort estimation methods (Obrien, 2009; Akkiraju & Hendrik, 2010; Li & Liam, 2010; Farrag & Moaward, 2014; Li & Keung, 2010; Gupta, 2013; Verlaine, Jureta & Faulkner, 2014; Mishra & Kumar, 2014). Despite the fact that these methods are promising, the problem of estimating SOA development effort still remains largely unresolved mainly because there is limited attempt in using size metrics and key SOA effort factors to estimate SOA development effort.

1.3 Objectives

1.3.1 General Objective

The main objective of the study was to define a suite of size-based metrics and then use them as the basis of developing an effort estimation method for SOA systems.

1.3.2 Specific Objectives

- i. To define a suite of size metrics to measure size attributes of SOA software systems.
- ii. To develop an effort estimation method for SOA systems based on the size metrics.
- iii. To implement a static analysis tool that automates computing of the size and estimating effort for developing SOA systems.

1.4 Research Questions

- i. How to define a suite of metrics for measuring the size attributes of SOA systems?
- ii. How to estimate SOA development effort effectively based on the size metrics?
- iii. How to implement the proposed metrics and effort estimation method into a static analysis tool for SOA software systems?

1.5 Hypotheses

This research study conceptualized the following 4 alternative hypotheses statements.

1. There is a correlation between SOA size attributes and SOA size.
2. There is a correlation between SOA size and SOA development effort.
3. The proposed SOA size metrics and SOA effort estimation method are more accurate as compared to existing metrics and methods.
4. The proposed SOA automated tool deep learning techniques are accurate in extracting UML text and images.

1.6 Scope of the Study

Over the years SOA communication methods included DCOM (Distributed Component Object Model), RMI (Remote Method Invocation) and CORBA (Common Object Request Broker Architecture). These communication methods were specific to a particular platform and operating system. With the development in internet communication, web service is currently the most common SOA communication infrastructure due to availability, interoperability and affordability of internet infrastructure. Therefore, this study focused on web services which offer services over the web.

A web service can be developed using any programming language such as JAVA, ASP, PHP, Python and Visual Basic among others. However, this study focused on web services developed using PHP programming languages rather than examining the entire population of programming languages platform and projects. SOA infrastructure developments were excluded from this study on the basis of their availability from the industry. It is cheaper to acquire a ready-made standard infrastructure that has been developed and tested by the industry rather than developing new infrastructure.

Research investigation was based on a controlled laboratory experiment in the context of undergraduate students in computer science and Information Technology. Furthermore, a survey was done in the context of industry programmers to replicate and validate the experiment. Participants and projects to be used in the laboratory experiment and survey were sampled from a population of participants and projects.

1.7 Significance of the Study

Software size and effort estimation are critical components of software project management. The proposed size metrics and SOA effort estimation method will enable Software developers and software project managers to plan and implement software projects successfully. A well-defined software size metrics and effort estimation method will help project managers not to underestimate or overestimate SOA software projects (Rijwani & Jain, 2016). Underestimating leads to under-scoping, setting too short schedule and under-staffing (Shivakumar, Balaji & Ananthakumar, 2016). On the other hand, overestimating leads to over-scoping, over-staffing and over-costing. Cost overruns increased from an average of 56% in 2004 to 59% in 2012 in sampled software projects while time overruns increased from 71% in 2010 to 74% in 2012 (Standish, 2013).

Existing software effort estimation methods are less accurate given the fact that software engineering industry is evolving rapidly. Since 1960's software development paradigm has evolved from procedural language to object oriented language to internet programming to component based and SOA among others. This evolution requires similar evolution in software size and effort estimation methods to cater for new attributes introduced by new software development paradigms. This study introduced SOA size metrics and effort estimation method that will contribute in improving estimation accuracy in software industry. Furthermore, software engineer researchers will use this model's framework to develop more accurate metrics and methods for SOA and other types of software architecture.

1.8 Limitations of the Study

Due to inability to access SOA projects developed in the industry and unavailability of datasets based on UML artifacts in the industry, the study research investigation was based on SOA projects developed by students in a controlled laboratory experiment in a university setup. In order to overcome the challenges that may arise from the use of students' SOA projects, programmers from the industry participated in a survey to replicate the controlled experiment to improve the research validity. Secondly, research on SOA size metrics and effort estimation methods is still at infancy and thus there is limited publication on validation and calibration of existing SOA size metrics and effort estimation methods providing fewer opportunities for comparison with the metrics and method proposed in this study. Therefore, this study research results were compared with the industry accepted effort estimation error margin of up to 25%.

1.9 Assumptions of the Study

The study assumed that participants in the laboratory experiment and survey answered the questions in an honest and candid manner. This was enhanced by explaining to participants how anonymity and confidentiality was preserved. Participants were allowed to withdraw from the study at any time with no negative consequences to their withdrawal. Secondly, the study assumed that participants had sincere interest in participating in the research and they did not have any other motives. Lastly, the study assumed that the samples were representative of the population and participants were to experience similar phenomenon of the study.

1.10 Contribution of the Thesis

This thesis has made the following contributions:

- a) The proposed size metrics for SOA can contribute to knowledge in the area of software engineering in the field of software metrics. The metrics can provide foundation to future researchers in software metrics who may want to implement or extend the proposed size metrics. The proposed size metrics can contribute to practice in project management where software size is a critical component when estimating scope, effort, cost and time taken to develop a SOA application system.
- b) The developed method can contribute positively to knowledge and to practice in the area of software project management where project scope, effort and cost estimation are key aspects that determine project successful implementation.
- c) The tool can enhance metric computations and software method estimation by project managers working with SOA software systems.
- d) Provided empirical evidence that the proposed metrics and effort estimation method for SOA are valid and more accurate.

1.11 Organization of the Thesis

This thesis is organized into 8 chapters. The first chapter introduces the thesis by detailing the background of the study, problem statement, research objectives and research questions, scope of the study, significance of the study and limitations of the study. In addition, the chapter also provides details on contribution and structure of the thesis.

The second chapter reviews literature on basic concepts of SOA including description of web services, SOA characteristics and SOA development methodologies. The chapter

further reveals analysis of existing software size metrics, existing effort estimation methods and discusses the theoretical framework and conceptual framework.

The third chapter discusses how the research was carried out including research philosophy, research design, target population, sampling and data collection. The chapter also describes data collection instruments' reliability and validity and ethical issues.

The fourth chapter proposes a suite of SOA size metrics including Weighted Operation Count (WOC), Service Dependency Count (SDC), Weighted Message Count (WMC) and Weighted Service Count (WSC). The chapter also provides detailed theoretical validation of the proposed suite of size metrics for SOA.

The fifth chapter proposes SOA effort estimation method with a detailed analysis of SOA development effort estimation factors. The chapter also provides detail analysis of application of fuzzy logic to SOA software development effort factors to give more accurate and realistic results.

The sixth chapter provides a detailed design of SOA size metrics and effort estimation implementation tool including the requirements of the tool and the tool architecture design.

The seventh chapter provides empirical analysis details on how laboratory experiment and expert survey was planned and conducted. The chapter also shows detail of statistics analysis by comparing the proposed size metric and effort estimation method with existing size metric and effort estimation method. The last chapter concludes the thesis by highlighting achievement made by the research study, contribution to knowledge and practice and recommendation for future work.

CHAPTER TWO

LITERATURE REVIEW

1.

2.1 Introduction

This chapter presents a detailed analysis of basic SOA concept, existing software size metrics, SOA development effort factors and existing software effort estimation methods including their challenges and strengths with regard to SOA. This chapter further provides a conceptual framework detailing relationship among variables discussed in this study and a theoretical framework which entails the theory behind this research study.

2.2 Basic Concepts of SOA

SOA is a software system comprising of various communicating services working in synergy to achieve a defined objective. A service thus can be viewed as a reusable component that represents business processes such as order form, foreign exchange conversions or tax calculations. A service is a coarse-grained, discoverable and self-contained software entity that interacts with applications and other services through a loosely coupled, asynchronous, message based communication model (Johnston & Kelly, 2002; Chindove et. al, 2017). SOA defines an interaction model between functional units, in which the consumer of the service interacts with the service provider to find out a service that matches its needs through a registry (Chindove et. al, 2017).

SOA is defined as ‘a paradigm for organizing and utilizing distributed capabilities that may be under control of different ownership domains’. It is an environment of services made available to consumers over a distributed network (Hussain, Muhammad & Ahmed, 2010).

It consists of a set of business-aligned services that support a flexible and dynamic configuration to end-to-end business processes (Siddiqui & Tyagi, 2016).

2.2.1 Evolution of SOA

Software architecture of a computer system is the structure of the system, which comprise of the software components and interaction among them. From earlier computing age to present time, software architectures have evolved rapidly from fulfilling basic functionalities to affecting human life by providing corporate agility and operational efficiency. This result in utilization of shared application functionalities, reuse services and resources (Farrag & Moawad, 2014).

Improvement in hardware technologies, operating systems and networking enabled developers to gain more benefits by building more complex and composite software systems. Earlier computing systems used monolithic programs based on procedural software architecture which did not encourage modification. Structural design was later introduced to decompose a large program into manageable modules for easy modification (Seth, Singla & Aggarwal, 2012). Thereafter, object oriented programming was presented to enable encapsulation, information hiding, inheritance and polymorphism. Object oriented programming concept gave rise to component based architecture and service oriented architecture which allows building of more complex software in a distributed network and maximize utilization of resources and applications reuse (Asha, Kavana & Parvathy, 2017).

With regard to networking, earlier computers were single user machines not connected in a network. Currently networking technology has developed tremendously from local area

networking to wide area networking, from client-server architecture to n-tier architecture and from centralized server systems to distributed systems. These developments have contributed immensely to development of more complex systems such as service oriented architecture that fits well in the current networking technologies (Domdouzis et. al, 2016).

2.2.2 A Typical SOA System

The main elements of a service oriented architecture systems are services and service infrastructure (Bianco, Lewis, Meison & Simanta, 2011). SOA is an architectural style that defines an interaction model between 3 main functional units in which the consumer of the service interacts with the service provider by searching for a service that meets its requirements through a registry (Kubasell, 2006) as illustrated in Figure 2.1.

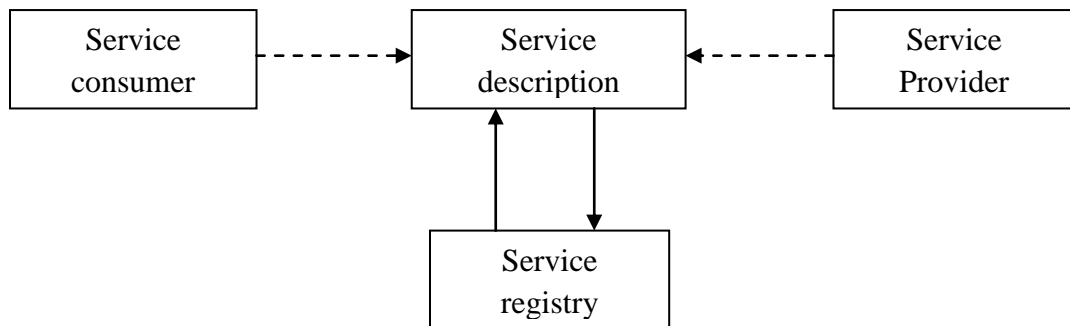


Figure 2.1: SOA Model (Adopted from Kubasell, 2006)

Service consumer is an entity that searches for a service to execute a required function by discovering a service through a registry. Secondly, *Service registry* is a directory which is accessed by consumers to enable location of service providers. Thirdly, *service provider* is a network addressable entity that accepts and execute request from consumers by providing the service description to fulfill consumer's requirements.

2.2.3 SOA layered Architecture

The early systems were based on 2-tier architecture where a client was connected directly to the database without any logical layer in between. Later on, 3-tier architecture was introduced with business logic layer in between presentation layer and data layer. The 3-tier approach isolates code implementation from the client and provides a platform for sharing data and concurrency access (Sharbanoo, Ali & Mehran, 2012). SOA is based on n-tier architecture in which services are layered on top of components that supports certain functionalities and provides quality of service (Seth, Singla & Aggarwal, 2012) as shown in Figure 2.2.

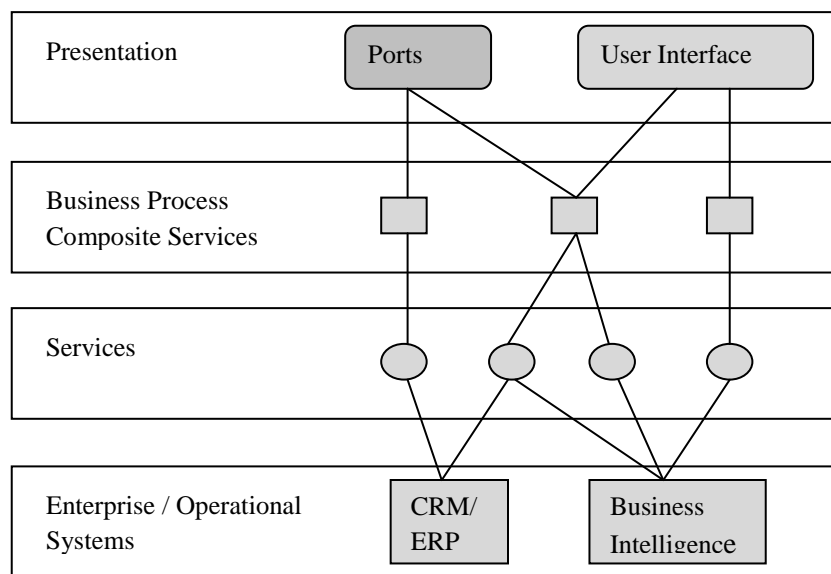


Figure 2.2: SOA n-tier layers (Adopted from Sharbanoo, Ali & Mehran, 2012).

The SOA n-tier layers are operational/enterprise layer, service layer, business process composition layer and presentation layer. Operational layer contains existing applications such as Enterprise Resource Planning (ERP), Supply Chain Management (SCM), Customer Relationship Management (CRM) and other applications which provide background services. Each of these systems are self-contained which their own databases and

implementation infrastructure. Enterprise component layer provides functions and requirements including management, availability and load balancing to services. The third layer is the service layer which contains the actual services. The fourth layer is the business process composition layer where services are composed into a single application through service orchestration and choreography which supports specific use cases and business processes. Lastly the presentation layer provides a link between users and the services and composite applications (Sharbanoo, Ali & Mehran, 2012).

2.2.4 SOA Characteristics

SOA characteristics includes clear separations of service interface from implementation, loosely-coupled, coarse grained, interoperability and location transparent. Clear separation of services interfaces from implementation allows service upgrades to occur without impact on system users. Secondly, services are loosely-coupled software entities with minimal level of dependency that facilitates software re-use (Farrag & Moawad, 2014). Loosely coupled refers to defining interfaces in such a way that they are independent of each other implementation such that replacing a component will have less effect on the system (Svanidzaite, 2014). Thirdly, course-grained services means that services can encapsulate and perform complete business logic such that services are discovered dynamically at run-time through a consumer searching a registry. Furthermore, services are interoperable with the ability to communicate with each other independent on the platform and programming language. Lastly, services are location transparent where clients of service don't have pre-knowledge on the position of service.

2.2.5 Web Services

An example of SOA are the web services technology which are the most commonly used SOA systems that provide a platform to link services developed in different programming languages running in different platforms using internet protocols (Mumbaikar & Padiya, 2013). For example, an application written in Java running on Linux communicates with an application outside the organization written in .NET or PHP running on Windows platform via internet standards. An exposed service provides a basic API by which a service can be invoked.

Before web services, SOA distributed technologies such as RMI (Remote Method Invocation), CORBA (Common Object Request Broker Architecture) and DCOM (Distributed Component Object Model) were only able to operate on a specific platform. RMI was specific for java Runtime environment which allows invocation of a method in a different address. On the other hand, DCOM was a component based model specifically made for Windows applications relying on RPC to enable communication among different application in different addresses. Similarly, OMG's CORBA enabled a client to request an object in a server. To link applications using the three technologies one required gateways to link and enable communication between two different technologies (Frantisek & Stal, 1998). In contrast, web services use internet standards and styles such as SOAP (Simple Object Access Protocol) or REST (Representational State Transfer) to communicate and interoperate among services (Dudhe & Sherekar, 2014).

2.2.5.1 SOAP Web Service Standard

Simple Object Access Protocol (SOAP) is a W3C standard that allows message exchange among services in a distributed environment (Dudhe & Sherekar, 2014). SOAP uses XML

to define an object to exchange messages regardless of different languages and platforms used. SOAP standard protocols include WSDL (Web service description language) and UDDI (Universal Description, Discovery and Integration). WSDL is a standard that uses XML to describe web services. A web service defines its methods in WSDL document, input/output parameters for each method, data types, transport protocol and the URL where a service is hosted. On the other hand, UDDI enables web services to publish details about their organization and web services to the registry. It also provides a way of finding a service via the registry also known as service discovery (Belqasmi, Singh, Ban melhem & Glitho, 2012).

2.2.5.2 REST Web Service Architecture

REST is an architectural style for communication among services across a distributed environment. REST web service respond to request made by consumer service via HTTP request for resources (Mumbaikar & Padiya, 2013). In this case, resources are the building blocks for web services such as a database record. REST architecture uses HTTP GET, PUT, POST and DELETE methods to access and manipulate resources. REST messages are smaller, perform better and consume less bandwidth as compared to SOAP standard. REST models data into a resource and enables identification of a resource through resource URI (Belqasmi, Singh, Ban melhem & Glitho, 2012).

Through internet SOAP standard or REST architectural style, a web service is simply accessed by its URL exposed online. Consumer web services only need to know the URL of the provider services, data types and methods to call the provider service. Provider web services publish their functions while hiding their implementation details from the client. Resources are accessed through XML of JSON messages that have standard meaning

making it easier to exchange information via internet (Mumbaikar & Padiya, 2013). Introduction of SOAP standard and REST architectural style provided a window for growth development of SOA projects due to their simplicity in application as compared to earlier SOA standards.

2.2.6 SOA Development Methodologies

SOA development life cycle is different from other traditional software development lifecycle due to SOA's development objective which is to implement IT solutions based on business requirements and processes. The most common SOA development methodologies are Service Oriented Architecture Framework (SOAF)(Erradi, Anand & Kulkarini, 2006), Service Oriented Modeling and Architecture (SOMA) (Arsanjani et al, 2008) and Service Oriented architecture Modeling Language (SoaML) (Amsdon, 2010). They all advocated for developing SOA based on business modeling process in order to add value to business requirements.

2.2.6.1 Service Oriented Architecture Framework (SOAF)

SOAF methodology consists of 5 main phases namely information elicitation, service identification, service definition, service realization, roadmap and planning. It combines the top-down modeling of an existing business process with a bottom-up analysis of existing applications (Erradi, Anand & Kulkarini, 2006). Information elicitation phase entails the analysis of the current business processes "as-is" model and proposed business process "to-be" model. Candidate services are identified to implement the "to-be" business model while Process-to-Application model (PAM) is done to analyze existing legacy application assets to discover which application can be transformed to suit into SOA implementation (Erradi, Anand & Kulkarini, 2006). SOAF Process-to-Application model

(PAM) provides a blueprint for identifying significant SOA size attributes when dealing with legacy applications. However, SOAF “to-be” business model is at a higher level of abstraction which cannot show a detailed service attributes which can assist in identifying SOA size attributes.

2.2.6.2 Service Oriented Modeling Architecture

SOMA provides a guideline on how to use business model details as input to define services. It emphasize on SOA principles to solve business problems by designing SOA based on business aligned goals and strategy. SOMA has 3 phases namely: Identification, Specification and Realization (Arsanjani et al, 2008). The main objective of identification phase is to capture exhaustively list of services that are potential candidates for exposure. Secondly, specification phase includes checking the requirements against what services can provide and to design services in detail. Lastly, Realization phase provide guidance on how to translate architectural decisions and designs to service realization and eventually implementation (Arsanjani et al, 2008).

2.2.6.3 Service Oriented Architecture Modeling Language

SOAML is an Object Management Group (OMG) standard that is intended to bridge the gap between business requirements and IT solutions. It is an extension of UML with an aim of supporting SOA modeling (Amsdon, 2010). SOAML proposed 5 main phases namely: Service Identification, Service Specification, Service Realization, Service composition and Service Implementation. In SOAML, the most important phase in identifying SOA attributes is the Service specification phase which is modeled by defining each service in detail. It includes details on services interfaces, the role played by the interfaces, functional capabilities and inputs/outputs details and communication protocols.

Service specification phase is modeled using SOAML interface diagram and sequence diagram (Amsdon, 2010).

2.3 Existing Software Size Metrics

Over the decades, software development process has transformed from structured design such as waterfall to new approaches such as agile, component based, software re-use and service oriented architecture. This transformation resulted to increase in software size and complexity. Consequently, software effort factors have also changed over time due to evolution in software practices (Sharma, Bajpai & Litoriya, 2012). Software size has been the main software effort factor or indicators since early 1980's with the introduction of Lines of Codes and Function Point analysis metrics to measure software size. Later software paradigms such as Object Oriented programming led software size metrics researchers' to shift focus from lines of codes to modules internal structure and relationship among modules as the main attributes to measure. With the introduction of SOA, researchers introduced size metrics specifically for SOA including Number of service Count, Service interface count (Zhang, Li, 2009; Hirzalla, Cleland-Huang & Arsanjani, 2009; Elhag, Mohamad, 2014) and Functional Size measurement method for SOA (COSMIC, 2010) built on the foundation of earlier programming architectures. This study classify software size metrics into traditional size metrics and SOA based size metrics.

2.3.1 Traditional Software Size Metrics

Traditional software size metrics were constructed to measure size of software applications without considering the software type or design methodology. Traditional software size metrics identified in this study include Lines Source of codes (SLOC), Function Points

Analysis (FPA) (Albrecht, 1983), Story Points (Greening, 2003), Use Case Points (Karner, 1993) and Object Points.

2.3.1.1 Source Line of Code (SLOC)

SLOC was the earliest size metric used to measure the size of a program by counting the number of lines of a program's code (Albrecht & Gaffney, 1983). The goal was to measure the amount of intellectual work put into program development (Khatibi & Jalawi, 2010; Prokopova & Silhavy, 2015). SLOC can also be used to measure number of errors, defects and documentation pages. One major limitation of SLOC is that, it is applied in procedural programming languages and it works at the coding phase of program development cycle. Currently it is not a suitable metrics due to rise in automated generated codes. SLOC is dependent on programming language platform and therefore it is inadequate when dealing with heterogeneous systems developed using different programming platforms such as SOA (Prokopova & Silhavy, 2015).

2.3.1.2 Function Point Analysis (FPA)

Function Point Analysis (FPA) presented by Albrecht (1983) measures number of functionalities in a software application. Software functionality is not directly related to the number of lines of codes, a skilled developer may use less SLOC to develop functionality than unskilled programmer. Furthermore, a programmer whose productivity is measured in SLOC may tend to include unnecessary codes (Coelho & Basu, 2012). It is based on this principle that functionality of application software is the key driver of the application software size which will eventually make a major contribution to software development effort (Arnuphaptrairong & Suksawasd, 2017).

FPA is independent on programming language and thus can work with different programming languages. FPA can be applied at requirement specification and design phases of software development process. Function point measure is arrived at by counting the number of five basic software components including external inputs, external outputs, external inquiries, logical internal files and external interfaces. Each of the 5 function component is weighed by a respective complexity level ranging from low, average to high as indicated in Table 2.1 then summed up to give Unadjusted Function (UFP) (Albrecht & Gaffney, 1983).

Table 2.1 Function Point Complexity weights

Function type	Low	Average	High
Internal logical file	7	10	15
External interface file	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6

(Albrecht, 1983)

Therefore, Unadjusted Function Points (UFP) is,

$$UFP = \sum_{i=1}^5 \sum_{j=1}^3 N_{ij} W_{ij}$$

Where N is the number of function type and W is the weight of a function type.

UFP is then adjusted with 14 complexity factors (CF)

$$\text{Total CF (TCF)} = 0.65 + (\text{sum of factors})/100$$

$$\text{Adjusted Function Point} = \text{UFP} * \text{TCF}$$

Another advantage of FPA is that non-technical user can easily understand the metric. Function point work best when requirements are well defined and there is certainty on the structure of system to be developed.

A number of function Points metrics were introduced since 1985 including Mark II Function points, 3D Function Points, COSMIC full Function Points, De-Marco Function Points and Feature Function Points (David, 2006) and International Function Point User Group (IFPUG). However, tradition Function point metrics versions do not capture SOA features such as service dependency, operations and message movement. This prompted adjustment of traditional Function Point by researchers to take SOA features into consideration (Mahmood, Ilahi, Ahmad & Ahmad, 2012). Mahmood et al introduced a Function Point version calibrated to meet SOA features demand by adjusting data communications, distributed data processing, performance and heavily used configuration factors. Based on case studies involving three projects, their proposed function point metrics returned more accurate results as compared to the traditional function point method. However, service internal structure, service dependency, service types were not captured in their modified FPA and detailed analysis of the method was not documented.

2.3.1.3 Story Points

A Story Point measures the size of a story or a feature in agile software development. It is relative in nature in that a story that is assigned 2 points requires twice the effort of a story that takes 1 Point. A story relies on analogy where the developer must have experience on the type of application software being developed. A Story Point is based on the effort, complexity and inherent risk in developing a feature (Greening, 2003; Cohn, 2005; Coelho

& Basu, 2012). Agile team estimate the effort and duration required to deliver a feature based on story points. Story Point only deals with applications developed based on agile software development method as it represents size of a feature to be developed in agile process.

2.3.1.4 Use Case Points (UCP)

Use Case Point is suitable for applications built using object oriented paradigm. It calculates unadjusted software application size (unadjusted UCP) by counting the number of use cases and the number of actors resulting from users' requirements specifications. A use case diagram shows interaction between different users and the systems. Use case diagram provides an opportunity to measure software size at an early stage of software development (Karner, 1993; Azzeh, 2013; Kirmani & Wahid, 2015). Actors and Use cases are classified and weighted based on their complexity as shown in Table 2.2 and Table 2.3.

Table 2.2: UCP Actors classification

Actor type	Description	Weight
Simple	Actor interaction with API	1
Average	Actor interaction with Protocol driven interface	2
Complex	Actor interaction with GUI	3

(Karner, 1993)

Table 2.3 UCP Use cases classifications

Use case	Number of transactions	Weight
Simple	Less than or equal to 3	5
Average	Between 4 and 7	10
Complex	Greater than 7	15

(Karner, 1993)

Unadjusted Actor Weight (UAW) = Number of actors * Weights

Unadjusted Use case weight (UUCW) = Number of use cases * Weights

Unadjusted Use Case Point (UUCP) = UAW + UUCW

The study identified 13 Technical Factor (TF) and 8 Environment Factors (EF) which are scored from 0 to 5 then multiplied and added to specific constants to get Technical Complexity Factor (TCF) and Environmental Complexity Factor (ECF) as shown below (Karner, 1993).

$$TCF = 0.6 + (0.01 * TF)$$

$$EFC = 1.4 + (-0.03 * EF)$$

Use Case Points (UCP) is computed by finding the product of UUCP multiplied by TCF and ECF.

$$UCP = UUCP * TCF * ECF$$

UCP is appropriate for measuring the size of Object Oriented Application at an early age and it is simple to implement (Azzeh, 2013). However, it requires adjustments in the classification of use cases due to the advancement in technology and factors need to be included apart from environmental and technical factors (Kirmani & Wahid, 2015). Furthermore, it is not suitable for SOA due to its inability to capture SOA attributes.

2.3.1.5 Object Points

Object points measures the size of software based on number and complexity of objects (Borade & Khalker, 2013). The objects are screens, reports and 3GL components. The steps for estimating development effort using object point include counting the number of objects, classification of objects (simple, medium, average) and assigning weights to objects with regard to complexity as shown in Table 2.4.

Table 2.4: Classification of objects weight

Object type	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3 GL components			10

(Borade & Khalker, 2013)

Object point is determined by adding all the weights of object instances to get object point count. Estimate percentage re-use is finally used to compute the overall object points (NOP) where,

$$NOP = (Object\ Point) * (100\% \text{ reuse})/100$$

Furthermore, developers' productivity is weighted from low to highest then effort is estimated by dividing net object point by productivity (Borade & Khalker, 2013). Object point only considered Third Generation Languages (3GL) and Fourth Generation Languages (4GL) factors and thus cannot apply to current programming paradigms such as OOP, Component-based and SOA.

2.3.1.6 Object – Oriented Size and Complexity Metrics

Object-Oriented programming share a number of properties with SOA and Component based systems. Properties that are common in these paradigms are separation of tasks into methods or operations, cohesion and dependency properties. Similarities between Object Oriented applications and SOA properties prompted a number of SOA metric researchers to adopt Object Oriented Metrics to measure SOA. Traditional Object-Oriented Metric that was adopted and configured to measure SOA complexity attribute is Weighted Method Count (WMC). WMC measures software complexity by counting the number of weighted

methods based on Cyclomatic Complexity (McCabe, 1976; Chidember & Kemerer, 1998; Hirzalla, Cleland-Huang & Arsanjani, 2009). Other metrics adopted by SOA researchers from Object-Oriented are coupling and cohesion metrics. Furthermore, Object-Oriented design tools such as UML is widely used to represent SOA design. A case in point is Service Oriented Architecture Modeling Language (SoaML) (Amsden, 2010) which is an extension of UML design tool.

2.3.2 Existing SOA Complexity and Size Metrics

Existing SOA complexity and size metrics include Weighted Service Interface count (WSIC), Number of Services (NOS) metrics and COSMIC-SOA metrics.

2.3.7.1 Weighted Service Interface Count (WSIC)

WSIC was proposed to measure the number of exposed interfaces or operations as defined in the Web Service Description Language (WSDL) documents (Hirzalla, Cleland-Huang & Arsanjani, 2009; Elhag & Mohamad, 2014). However, the metric was not validated empirically and it did not take into consideration other related attributes. WSIC returns the number of operations in a service based on the hypothesis that the higher the number of service operations the more complex a SOA application will be. They observed that the amount of work needed to develop and test a service operation increases with rise in the number of operations. WSIC provided an insight on the relevance of operations as an attribute in determining SOA complexity. However, no literature so far has revealed the empirical analysis to verify the metrics reliability. Furthermore, WSIC metric is a not a size metric but a complexity metric.

2.3.7.2 Number of Services (NOS)

Number of services metric is a simple count of services contained in a SOA system (Zhang & Li, 2009; Hirzalla, Cleland-Huang & Arsanjani, 2009; Elhag & Mohamad, 2014).

Number of services metric is a measure of a system's complexity based on the hypothesis "The higher the number of services the more complex a SOA system becomes.". Factors that contribute to system's complexity as a result of number of services include increase in the number of operations, increase in number of integrations and dependencies and need to provide better governance and infrastructure to support SOA application.

Number of services metric provided a foundation for developing more SOA metrics that rely on number of services. However, according to this research study knowledge, there is no literature on empirical validation done on NOS metric. In addition, the metric simply counts the number of services disregarding the fact that services are different and therefore should be assigned weights in relation to their complexity or size.

2.3.7.3 COSMIC-SOA Metrics

COSMIC is a consortium of software measurement professionals that was started in 1998. COSMIC-FFP (Common Software Measurement International Consortium-Full function Points) was introduced purposely to measure functional size of software. COSMIC-FFP (2003) is a software size estimation method approved by ISO (ISO/IEC 19761:2003). COSMIC measurement methods involves applying a set of models, principles, processes and rules to measure functional user requirements of a given software which will result to the function size of software (COSMIC, 2015). The group has published several methods which include COSMIC-FFP for web application and COSMIC-SOA for SOA based projects.

COSMIC was originally introduced for business and real-time applications which are characterized with large amount of data movement. COSMIC-SOA was presented to handle service applications over distributed network. With data exchange among service providers and users being the concept behind SOA, it matched COSMIC philosophy of sizing software in relation to amount of data exchange which prompted the introduction of COSMIC-SOA specifically for SOA application systems which also embraces data movement principle (COSMIC-SOA, 2010).

COSMIC principle states that the main programming efforts are dedicated towards handling of data movements from/to the storage and users. Therefore, the number of data movement provides insight into the system size (Martino & Gravino, 2009). COSMIC methods require a definition of the context model for a specific application with clear boundaries separating the software from its operating environment. Each data movement crossing the boundary is counted to give the full function points.

The data movements are classified as Entry, Exit, Read and write. One advantage of COSMIC is the ability to estimate software size of big projects by counting the amount of data. However, COSMIC methods focus more on data movement rather than considering other SOA size indicators. Furthermore, data movement alone does not recognize the fact that a service that is more complex in design, with more operations and dependencies is bigger than a service that is less complex, with fewer operations and dependencies. Data movement only caters for one aspect of SOA size.

2.4 Existing Effort Estimation Methods

This study classified existing effort estimation methods as traditional effort estimation methods and SOA effort estimation methods.

2.4.1 Traditional Effort Estimation Methods

Research in software cost estimation has been around for several years now, although it is still at its infancy due to changes in software environments and development methods. Traditional software effort estimating include expert judgment, Analogy, top-down, bottom-up, price-to-win, Wideband Delphi, Source line of codes (SLOC), Object points, Function-Point Analysis (Albrecht, 1983), Constructive Cost Model-I (COCOMO-I) (Boehm, 1981), Constructive Cost Model-II (COCOMO-II) (Boehm, 2000), Artificial neural Network (ANN) methods and Fuzzy logic methods.

2.4.1.1 Expert Judgment

Expert Judgment technique is the most frequently applied effort estimation method where experts are responsible for estimating the size and effort of a software (Khatabi & Jawawi, 2010). This method is based on the project manager experience in similar software projects (Borade & Khalker, 2013). Expert judgment is prone to human errors and biasness and its success is grounded on expert judgment (Bhalerao & Ingle, 2009). Expert experience may differ from one expert resulting to varying estimates on the same type of project. However, it is helpful in small and medium sized software project and when the development teams and software attributes have not experienced significant changes as compared to previous projects. However, this method cannot be used to estimate a large and complex software project such as SOA.

2.4.1.2 Analogy

Analogy technique estimation is done according to the actual effort of one or more completed projects that are similar to the new project to be estimated (Khatabi & Jawawi, 2010; Borade & Khalker, 2013). Estimation can be done at the total project level or at sub system level. The strength of estimation by analogy is that the estimate is based on actual project experience and estimation can be done in the absence of an expert. However, it does not take into consideration the extent of other relevant effort factors in the previous project such as the environment and functions which may differ with new project cost factors (Kumari & Pushkar, 2013). In addition, a lot of past information about past projects is required whereas in some situations there may be no similar projects developed in the past to compare with. Most SOA projects are unique to the organization and the types of services offered depend on unique organization's needs providing a challenge when comparing with past projects' data to estimate software development effort.

2.4.1.3 Price-to-Win

Price-to-Win estimation method is based on customer budget instead of software parameters or features. Example is when a customer is willing to pay for 6 persons-month and the project estimate is 8 persons-month then estimation is done as per the customer ability to pay. This may cause delays and force developers to work overtime (Kumari & Pushkar, 2013). Price-to-win method helps in getting the contract but it generally causes effort, cost and time overruns. Furthermore, price-to-win may demotivate developers due to set budget that is client centered as compared to consensus between developers and the client. In addition, price-to-win may compromise the quality of a project due to developers' need to fit into the client's budget.

2.4.1.4 Bottom-up and Top-up

Bottom-up estimation method estimates by separating each software component then summed to give the overall estimate for the product. It is possible only when the requirements and design of the system are known at an early stage of software development (Sharma, Bajpai & Litoriya, 2012). While top-down method establish an overall estimate for the project then the software project is sub-divided into its functional components which are then estimated based on the overall estimate (Sharma, Bajpai & Litoriya, 2012).

The design and requirements must be well defined to partition software to its component. Bottom-up and top-down methods may also apply to SOA systems by sub-dividing a system into services which are the basic components of SOA. SOA design may include different service types and legacy systems considerations becoming difficult to sub-divide to functional components. Bottom up and top-up method only enable sub-division of a project into smaller quantifiable sub-projects but there is no documented evidence to show how the smaller sub-projects size and software development effort are computed then summed up to give the overall result.

2.4.1.5 Wideband Delphi

Wideband Delphi method is an estimation technique where effort and cost are estimated centered on team consensus. It is done by getting advices from experts who have extensive experiences in similar projects. Wideband Delphi technique was introduced by Barry Boehm and John Farquher in 1970s. It uses work breakdown structure as the basis for estimating project size, effort and cost (Gandomani, Wei & Binhamid, 2014; Stellman &

Greene, 2005). This method emphasizes on consultations, communication and interaction among participants.

Participants include customer representatives and technical team members involved in development of the software product. Each member estimates for each task and identify changes and missing assumptions in work breakdown structure. Members with high or low estimates are asked to justify, and then members revise the estimates. The cycle repeats until when estimators agree on the estimates. The coordinator collects estimates from team members and assembles the tasks and estimates into a single final task list.

Wideband Delphi depends on team members experience and agreement among members and thus it is not appropriate method when applied to a software project that is unfamiliar to members (Stellman & Greene, 2005). Furthermore, it is a preferred method when requirements are well defined and therefore, cannot work for software development methodologies where requirements are not clear. However, it encourages collaboration and it is simple to apply. Even though Wideband Delphi estimates are consensus-based, experts may be biased, optimistic or pessimistic in their estimation.

2.4.1.6 Constructive Cost Model (COCOMO)

COCOMO model proposed by Boehm (1981) used parametric to compute and estimate software development effort. Due to COCOMO methods popularity various studies have extended COCOMO framework to develop cost estimation methods with an aim of improving software estimation accuracy. The 4 original COCOMO methods were simple COCOMO (Boehm, 1981), Intermediate COCOMO (Boehm, 2000), Detailed COCOMO and COCOMO II.

Basic COCOMO computes software effort and cost as a function of program size expressed in thousands lines of codes (KLOC) using the formula:

$$Effort = a(KLOC)^b$$

Where a and b are effort coefficient and economy of scale constants respectively which are assigned weights according to software project complexity and size as shown in Table 2.5.

Table 2.5: COCOMO Complexity factor weights

Model	Effort coefficient (A)	Economy of scale constants (B)
Organic (Small)	3.2	1.05
Semi-detached(Average)	3.0	1.12
Embedded (Large)	2.8	1.20

With the advancement in software development methods and environment, basic COCOMO was not able to capture all relevant effort factors in its estimation. Therefore, intermediate COCOMO was released to include emerging software attributes in their computation of software estimates.

Intermediate COCOMO also used Kilo lines of codes as in basic COCOMO but it included Effort adjustment factors (EAF) which are subjective assessment of products, hardware, personnel and project attributes (Boehm, 2000; Kumari & Pushker, 2013). EAF considered a set of four factors, with each factor having a number of attributes. The complexity factors were hardware, personnel, project and product with 17 attributes rated on a 6 point scale that ranges from very low to very high. The intermediate COCOMO model takes the form

$$EFFORT = a * (KLOC)^b * EAF.$$

Later on, detailed COCOMO was introduced to incorporate all characteristics of intermediate COCOMO on each step of software development process (Analysis, Design, coding and testing). The 17 attributes were used at each stage of development cycle to estimate software development effort (Boehm, 2000; Kumari & Pushker, 2013; Borade & Khalker, 2013).

In addition, COCOMO-II was introduced in 1997 as an extension of intermediate COCOMO. COCOMO II used Thousands lines of code (KLOC) or Function point to measure software size. Furthermore, Effort Adjustment Factors (EAF) were increased by 5 to 22 attributes (Boehm, 2000; Kumari & Pushker, 2013). The Usage of COCOMO II was very wide and its results were more accurate compared to previous versions of COCOMO.

Tansey & Stoulia (2007) attempted to use COCOMO II to estimate the cost of developing and reusing SOA services. They concluded that COCOMO II has a number of coefficients that capture some of SOA attributes. However, COCOMO II is inadequate when estimating effort required when reusing a service and could not capture all SOA attributes such as service type factor. They proposed that COCOMO II should be extended to accommodate new characteristics of SOA.

All COCOMO versions captured a wide range of parameter when estimating the cost of a project. So far COCOMO methods are the most popular methods and the most validated method with clear results. The use of COCOMO requires clear and well defined requirements (Basha & Dhavachelvan, 2010). However, SOA attributes are not included among COCOMO complexity factors that determine software effort. Therefore, all versions of COCOMO are inadequate in relation to estimating SOA effort.

2.4.1.7 Artificial Neural Network Effort Estimation Methods

Artificial Neural Network (ANN) effort estimation methods were proposed with an aim of acquiring facts from previous software projects and use the facts to predict software development effort more accurately (Rijwani & Jain, 2016; Bilgaiyan, Sagnika, Mishra, Madhabunenda & Das, 2017). Neural networks are simulation of human biological nervous system with mathematical functions for prediction and estimation (Park & Bark, 2008). There are several types of neural networks used by researchers to estimate effort including back propagation algorithm and feed forward network algorithm such as Radial basis function neural network, Generic regression, wavelet neural network among others.

Neural networks use a function that works on identified software project attributes as input to the function to give a predictive output. Neural network requires training of the method using past software project data and use of trial and error to attain a more accurate estimation. After training ANN method, weights are modified appropriately to give an expected output (Bawa & Chawla, 2012). Based on trained project data, neural network method use a function to compute on the project attributes as input then predict development effort more accurately (Rijwani & Jain, 2016).

Neural network methods are preferred when there is enough previous project data to train the ANN method. Secondly, they are able to use training datasets to give more accurate prediction. In addition, ANN methods are able to model complex relationship between project effort attributes and effort (Bawa & Chawla, 2012). However, according to this research knowledge there is no enough data on exiting SOA projects to use as training data to ANN.

2.4.1.8 Fuzzy Logic Effort Estimation Methods

Various research on Software development effort estimation have incorporated fuzzy logic in their estimation methods to yield more accurate results as compared to traditional algorithmic methods (Ziauddin et al, 2013; Ahlawat & Chawla, 2015; Thamarai & Murugavalli, 2015; Patra & Rajnish, 2018; Kaur, Narula, Wason & Jain, 2018). They proposed fuzzy effort estimation model based on existing arithmetic methods such as COCOMO II. Their main objective was to develop estimation methods that are more representative of human thinking and perception with regard to effort estimation.

Fuzzy estimation methods take values assigned to traditional methods such as COCOMO II effort multiplier factors then convert the values into linguistic values through fuzzification membership functions including Triangular and Trapezoidal membership functions (Thamarai & Murugavalli, 2015; Patra & Rajnish, 2018; Kaur, Narula, Wason & Jain, 2018). Fuzzy sets derived from fuzzification represents membership functions which correlates fuzzy sets to degree of membership in the interval [0,1].

Most research studies on fuzzy logic effort estimation employed Mamdani Inference Engine to apply IF..THEN rules on crisp values entered into the system to give a fuzzy output in relation to the input (Ziauddin et al, 2013; Ahlawat & Chawla, 2015; Thamarai & Murugavalli, 2015). The summed output were then defuzzified by Center of gravity method to give crisp data as the final output. They used MATLAB to implement their proposed fuzzy effort estimation methods. Through experiments, they proved that fuzzy logic effort estimation methods yielded more accurate results when compared to traditional methods such as COCOMO II. However, according to this study's knowledge no research to date has proposed fuzzy logic effort estimation method for SOA applications.

2.4.2 Existing SOA Cost Estimation Methods

Attempts to estimate SOA effort and cost have also been discussed in various studies although there has been no evidence of validation or calibration of frameworks published in recent literature. Existing SOA frameworks include: Estimating the cost of development customization to packaged applications using SMART-AUS scope, cost and effort estimation framework for SOA (O'Brien, 2009), SOA Cost Estimation for Customization to Packaged application (Akkiraju & Geel, 2010), Effort Estimation for Web Service composition (Li & Liam, 2010), Phased effort estimation of legacy systems migration to SOA (Farrag & Moaward, 2014), Software Cost Estimation Framework for SOA Systems using Divide-and-Conquer (Li & Keung, 2010), Service Point Estimation Model for SOA Based Projects (Gupta,2013), Requirements Based Model for SOA Systems Effort Estimation (Verlaine, Jureta & Faulkner, 2014) and Estimating Development Size and Effort of Business Process SOA Applications (Mishra & Kumar, 2014).

2.4.2.1 SMAT-AUS Scope, Cost and Effort Estimation Framework for SOA

SMAT-AUS framework recognized types of SOA projects as key inputs when determining scope, cost and effort of Service oriented Architecture (SOA) projects (O'Brien, 2009). The framework identified different SOA project types including service mining, service development, application development, service integration, SOA infrastructure, SOA governance and SOA architecture analysis (Li & Keung, 2010).

For each of the project type, the study proposed a template that will capture details about existing applications (legacy systems) to migrate to SOA, existing services to be mined and services to be developed. The study also recommended identification of cost factors that

are specific to a SOA project type and factors that are common across more than one project type (O'Brien, 2009).

The study identified Technical and social factors that influence SOA development effort. Technical factors identified in the framework include hardware and software issues that impact SOA development effort. On the other hand, social factors presented in the framework were factors that deal with people including communication among people, developers' skills, organization's structure and development teams. SMAT-AUS framework was validated based on a case study involving one SOA project. However, the paper only provided a framework and proposals with no detail or metrics to measure identified factors. Furthermore, the paper excluded service size as one of the cost factors in the proposed framework.

2

2.4.2.2 SOA Cost Estimation for Customization to Packaged Applications

Akkiraju & Geel (2010) introduced a model to estimate effort involved in developing SOA systems by taking into account SOA advanced features such as business object management, load balancing, web server management and web page management. This model estimates effort and cost using business process model by counting business objects. Inputs to the model include number of process steps, number of user roles, packaged application landscape, legacy application landscape and size of message object.

Akkiraju & Geel (2010) applied Service Oriented Modeling Architecture (SOMA) to conduct process decomposition to arrive at granular process steps. With the use of SOMA, they were able to identify business objects. Based on Artifact-centric approach, inputs and outputs flowing in and out of the process steps were identified to capture all business

objects whose state changes are significant to the process (Kumaran, Liu & Wu, 2008). This approach works well when process model artifacts are revealed by BPMN (Business Processing Modeling notations). In cases where business processes have not been documented using BPMN notations, linguistic analysis approach is used to parse the business processes stated in English sentences. Linguistic text analysis approach use noun identification, tokenization and verb identification mechanisms to identify business objects.

Other inputs to the model included service interface count and user interface count. Service interface counts were defined as interfaces that are exposed by the application whose complexity was taken into consideration. On the other hand, user interface was defined as count of user roles which represents a web page with features for creating, reading, updating and deleting operations (Akkiraju & Geel , 2010). All these inputs were entered into an effort estimation engine that is based on work breakdown structure (WBS). The time required for each activity was captured then summed up to obtain overall person-month. The model was validated using a laboratory experimentation that was based on three projects.

One advantage of estimating by considering business objects as the key input to the model is the ability to estimate effort at an early stage of software development. However, at an early stage, key service attributes such as structural attributes, message movement and dependency attributes cannot be captured. Furthermore, key factors that influence SOA development effort such as service type, personnel factors, requirement factors and product factors were not used in the method.

2.4.2.3 Effort Estimation for Web Service Composition

Li & Liam (2010) proposed an effort classification matrix for web service composition with regard to context and technology aspects of service composition. The method defined qualitative effort estimation hypotheses to identify effort factors that influence web service composition. They identified two classifications of effort factors as context and Technology dimensions. Context dimension included Pattern, Semiotics, Mechanism, Design time and Run-time as shown in Table 2.6. On the other hand, Technology dimension included Workflow based, Model Driven and AI planning.

Table 2.6 Context Effort Factors

Context factor	Type
Pattern	Orchestration and Choreography
Semiotics	Service Discovery and Matchmaking
Mechanism	SOAP based and RESTful
Design-time	Manual, Semi-Automatic and Automatic
Run-time	Dynamic and Static Composition

(Li & Liam, 2010)

They applied a set of service composition effort estimation hypotheses to generate a checklist to qualitatively define composition effort factors. Special symbols and rules were used to assign weights to each effort factors (Li & Liam, 2010). For example, they used $E_{(F-H)}$ to represent effort influenced by factor F when using hypothesis H. A weight/Score of S was used to set $E_{(F-H)}$ to flag effort according to different factors scores. For instance, they compared orchestration and choreography as shown in Table 2.7.

Table 2.7: Comparison between orchestration and choreography

Hypothesis	Comparison	Score
H3	$E_{For-H3} < F_{Fch-H3}$	$S(E_{For-H3}) = 1,$ $S(E_{Fch-H3}) = 2$
H5	$E_{For-H3} < F_{Fch-H3}$	$S(E_{For-H5}) = 1,$ $S(E_{Fch-H5}) = 2$
Total	$E_{For} < F_{Fch}$	$S(E_{For}) = 1,$ $S(E_{Fch}) = 2$

By associating effort factors weights and hypotheses, they were able to classify all effort factors and develop a matrix that compares different factors in relation to a set of hypotheses. They eventually constructed an effort estimation checklist table that can be used by developers to apply expert judgment to qualitatively judge and compare effort factors (Li & Liam, 2010). However, they focused on qualitative analysis with no emphasis on empirical analysis. The research is based on service composition disregarding other aspects of web service development phases.

2.4.2.4 Software Cost Estimation Framework for SOA using Divide-and-conquer

A framework for costing SOA using work breakdown structure provided an approach that follows the work breakdown structure principle by decomposing SOA into sub-problems (services) until a service cannot be sub-divided further. Effort of each service is estimated then recomposed to form the overall effort and cost (Li & Keung, 2010). The study classified services into available service (service discovery), migrated service (service migration), new service (service development) and combined service (integrated).

The study proposed an algorithm based on divide and concur technique to separate a SOA project into the four types of services. They proposed different sets of metrics for each service type and the total cost is calculated through the service integration metrics (Li & Keung, 2010). A case study based on one SOA project was used to show the framework applicability. This study revealed a framework for classifying SOA by using an algorithm to sub-divide a SOA system into distinct and smaller services. However, the study focus on a framework to decompose SOA project with no regard to SOA key attributes contributing to SOA development effort such as SOA size and other key attributes.

2.4.2.5 Service Point Estimation Model for SOA Based Projects

A technique to calculate size of SOA projects and estimate SOA development effort was proposed (Gupta, 2013). The model takes service operation as the unit of measurement whose complexity forms the basis of computing service size. Therefore size and effort are computed at service level rather than project level. The Model considered 3 dimensions to take into account when computing SOA size and effort. The three dimensions are Functional Complexity, Quality of service (QoS) and service development environment.

Functional complexity includes invocation data, business logic and downstream integration. On the other hand, Quality of Service are non-functional requirements for services which are classified into operational objectives and implementation objectives. Operational objectives include response time, data load, concurrency and security while implementation objectives include interface type, reusability and testability. Whereas functional complexity and QoS applies at a service level, service development environment attributes applies at project level. Service development environment attributes include

knowledge of business domain, knowledge of technology, team dynamics, service governance, requirements stability and tool support (Gupta, 2013).

The model involves 4 steps which include complexity identification, project sizing, effort estimation and cost estimation. Complexity identification entails identifying attributes, allocating weights to identified attributes based on their complexity and summing up the weights to give the overall score. For instance, Functional Complexity Factor (FCF) attribute classify operations type as simple, average and complex operations assigned weights of 3, 5 and 7 respectively(Gupta, 2013). Other factors computed based on weights are Integration Complexity Factor (ICF), Technical Complexity Factors (TCF) or QoS factors and Environment Complexity Factors (ECF). Complexity factors were used to compute SOA project size by multiplying summed complexity factors weights with set constants.

$$\text{Where, } TCF = 1.0 + (0.01 \times TCS)$$

$$ECF = 0.7 + (0.01 \times ECS)$$

TCS is Technical Complexity score and ECS is Environment Complexity score.

Therefore, Size of a service operation (SOS) = USP x TCF

USP is Unadjusted Service Point and therefore, Size of Service Interface (SIS) is,

$$SIS = \sum_i^n SOS \times ECF$$

Development effort depends on productivity of a team. Productivity level varies from one team to another and from one programming language to another. Development effort is productivity multiplied with SOA size (Gupta, 2013).

$$\text{Effort (E)} = P \times S, \text{ Where P is productivity and S is SOA size.}$$

The model provided a clear and detailed analysis of SOA attributes focusing on service internal structure complexity, technical complexity and environment complexity. It provided an insight into SOA attributes, weights and computation. However, so far there is no literature revealing theoretical validation and empirical validation of the model to determine the model's accuracy in relation to other existing models (Gupta, 2013).

2.4.2.6 Requirements Based Model for SOA Systems Effort Estimation

Software complexity factors were considered as key aspects that determine SOA application development effort (Verlaine, Jureta & Faulkner, 2014). They defined service structural complexity as attributes that involves software structural design attributes that influence service development effort. Structural complexity included Input/ Output complexity (IOC), Functional Requirements Complexity (FRC), Data Store Complexity (DSC), Non- functional Requirements Complexity (NFRC), Design Constraint Imposed Complexity (DCI), Interface Complexity (IFC), System Feature Complexity (SFC) and Software Deployment Location Complexity (SDLC). Service structural complexity factors were allocated weights based on their impact on service development effort to compute Requirement-Based Complexity (Verlaine, Jureta & Faulkner, 2014).

$$\text{Requirement-Based Complexity (RBC)} = (\text{PC} + \text{DCI} + \text{SFC}) \times \text{SDLC}.$$

Where $\text{PC} = \text{IOC} \times \text{RC}$ values, IOC includes the sum of IC, OC and DSC while RC is the sum of FRC and NFRC.

The model was validated theoretically based on 5 complexity properties of software complexity measurement process (Kitchenham et al, 1995). Technical Complexity Factors

(TCF) were used to adjust the model's computation. TCF are factors that influence development effort allocated weights of 0 (non-influence) to 5 (strong influence).

$$\text{TFC Value} = 0.65 + 0.001 \times \sum \text{DI} \text{ Where DI is the degree of influence.}$$

$$\text{Adjusted Requirement –Based Complexity} = \text{RBC} \times \text{TFC Value}$$

Software development effort is related to productivity of development staff. Productivity of staff is the ratio between the number of lines of codes and required time. Productivity varies from one programming language to another. For example the value for J2EE productivity of staff is 46 as proved by empirical research on 2190 software projects.

$$\text{Final effort} = (\text{Adjusted RBC} \times L) / P$$

Where L is the number of codes of lines per function point and P is productivity of staff.

The Model revealed relevant factors that contribute to SOA development effort, the metrics were validated theoretically and a case example was used to implement the metrics. Results from the theoretical validation and case study were encouraging. They considered structural complexity metrics when computing SOA size and effort but they didn't include service dependency and movement of data as key size attributes.

2.4.2.7 Estimating Size and Effort of Business Process SOA Applications

Business Process Modeling Notations (BPMN) constructs were used as the basis for computing development effort of business process SOA applications (Mishra & Kumar, 2014). They used modeling tools such as IBM webSphere and BPMN2 to construct BPMN model which eventually generated BPEL codes where size of business process is estimated using metrics. BPMN model constructs that were used to estimate SOA size include

number of processes, number of associated tasks, number of partnerLink elements, number of input and output variables, number of operations, number of XPath queries, number of events, number of associated links and number of mapped BPEL properties.

Based on BPMN artifacts, size of a process, total process size, total number of tasks, mapped BPEL properties and task size are computed to give SOA size. After estimating size, COCOMO model was used to estimate development effort based on medium-sized project COCOMO semi-detached type of project constants. MMRE was used to validate the model's accuracy with regard to provided datasets from existing projects. The model approach towards SOA size measurement is different from other approaches. The model only focuses on BPMN and BPEL models artifacts and other key SOA features such as service type that were not captured in the model.

2.4.2.8 Phased Approach for Effort Estimation for SOA projects

Farrag & Moawad (2014) proposed a model to estimate effort to migrate from legacy systems to SOA systems. They identified and analyzed key factors contributing to effort and cost of developing SOA systems (Farrag & Moawad, 2014). The identified factors were distributed among SOA development phases including Requirement, Design, Development, Testing and integration. The study further classified services into four categories namely: Available service, migrated service, new service and Composite service. They focused on migrated service in which they identified three strategies employed when migrating from legacy systems to SOA. The strategies are wrapping, re-engineering and replacement.

Wrapping is a strategy where interface is built to wrap existing legacy systems. It is preferred in situations where legacy systems codes are too expensive, limited time to rewrite and are of high business value that it becomes a risk to develop a new solution from scratch. On the other hand, re-engineering is a strategy of adjusting the legacy system by adding new functionalities. Meanwhile, replacement is a strategy of removing the old applications and replacing with a new SOA system (Farrag & Moawad, 2014).

Using the identified factors spread over SOA development phases, the study analyzed key factors against the three migration strategies which they applied weights ranging from 1(low) to 3(high) depending on effort required for a factor in relation to a migration strategy. For example, Obsolete legacy system technology is one of the factor in design phase weighted 3 when applying wrapping, 2 when applying re-engineering and 1 when replacing. This is done for all factors then a percentage effort per development phase is calculated (Farrag & Moawad, 2014). The study emphasis is on the cost of migrating from legacy systems to SOA applications. Furthermore, the study only focused on other key factors disregarding the size of services which is critical when estimating SOA effort.

In a different study, Farrag & Moawad (2016) proposed an effort estimation method that considers four types of web service namely new service, migrated service, available and composed service. For available service, effort is based on integration phase and testing phase effort. Secondly, they identified factors that influence migrated and new service distributed among all phases of SOA development cycle. Identified effort factors for migrated services were allocated weights of 1(low influence) to 3(high influence) while new service was allocated weights ranging from 0 (no influence) to 5 (very high influence). The total weights for all factors were summed up to give Total Degree of Influence (TDI).

$$\text{Value Adjusted Factor (VAF)} = (\text{TDI} \times 0.01) + 0.05$$

$$\text{Adjusted FP count} = \text{Unadjusted FP count} \times \text{VAF}$$

To convert function point count to effort in man-hour, a productivity factor of 8 per function point was used.

$$\text{Estimated total effort} = \text{Adjusted FP count} * 8$$

Two SOA projects were used to test the accuracy of their proposed SOA effort estimation method in an experiment which returned Relative errors of 3.66% to 19.08% (Farrag & Moawad, 2016). The proposed effort estimation method returned encouraging results with regard to SOA. However, they used only 2 projects from the same organization in the experiment. Furthermore, the aspect of SOA size factor was not captured.

2.5 Existing Automated Tools to Interpret design artifacts

Various research studies have proposed automated tools to read and interpret design artifacts such as UML and Computer Aided Design (CAD) (Karasneh & Chaudron, 2013; Intwala, Kharade, Chaugule & Magikar, 2016). Karasneh & Chaudron (2013) presented a tool to read UML class diagram to an XML file. The tool provided a platform to upload UML images, use geometrical technique to detect rectangles with attribute names, detect existence of relationship among class rectangles and extract attribute names using Microsoft Office Document Imaging (MODI) Optical Character Reader (OCR). The tool accuracy was validated based on various UML diagrams including large diagrams (Karasneh & Chaudron, 2013). However, the tool is challenged when dealing with various styles of representing UML diagram in existence currently. In addition, the tool was not able to establish the type of relationship among class interface rectangles.

Inwala et al. (2016) introduced a geometrical tool to detect arrow symbols contained in CAD image. The tool allows upload of CAD images which are converted to grayscale, then to binary using OTSU thresholding. The tool was required to detect two types of arrows based on the shape of the arrow head. Based on morphological technique, the tool was able to detect solid arrows using Black Top Hat morphology and detect line based arrows using White Top Hat morphology. The tool made use of contour detection and area checks to enhance arrow detection. The tool was tested by subjecting it with different types of CAD diagrams and it returned encouraging results. However, when different types of arrows with varied line thickness, area and contour features are used, the tool is challenged due not implementing the tool with machine learning techniques.

In the past few years, machine learning and deep learning techniques have become critical and effective techniques in detecting patterns in images. These techniques include Artificial Neural Network (ANN), Decision tree, Naïve Bayes (Dong-Chul, 2016), Support Vector machine (SVM) (Thai, Hai & Thuy, 2012) and Convolutional Neural Network (CNN) (Sultana, Sufian & Dutta, 2018). These techniques are more efficient and are able to capture varied shapes, thickness and style of shapes and pattern if training was done exhaustively.

2.6 Theoretical Framework

Theoretical framework presents the underlying theories that support a research study to validate the existence of a research problem (Swanson & Chermack, 2013). Over the years, various studies have looked into the study of software size metrics and software development effort estimation. In this study, key independent variables are SOA size,

service type and Effort Multiplier Factors (EMF) while the dependent variable is SOA development effort.

2.6.1 Software Effort Estimation Factors

Various researchers have identified and classified effort factors differently based on the type of software architecture and period when a factor was recognized as a key indicator contributing to software development effort.

2.6.1.1 Size Factor

One common factor identified by various studies is software size attribute. The first method to introduce software size factor was Function Point Analysis (Albrecht, 1983). Later on COCOMO versions also embraced the use of software size factor as the key input when computing software development effort. Basic COCOMO used Lines of codes (LOC) to measure software size. Apart from Lines of codes, other versions of COCOMO were designed to allow the use of Function Point and Object Point metrics to compute size for effort estimation. Despite the fact that size is a critical factor when estimation effort, various SOA based effort estimation methods miss to capture all size attributes but capture one or two size attributes. Service interface count which takes into account service internal structure was captured by various studies (Obrien, 2009; Li & Keung, 2010; Gupta, 2013; Farrag & Moaward, 2014) while service dependency attribute of size was captured by (Obrien, 2009; Li & Liam, 2010; Farrag & Moaward, 2014). Message count was first considered by COSMIC (2015) and later on various research studies used message count attribute to compute SOA size (Li & Liam, 2010; Li & Keung, 2010; Gupta, 2013). However, according to this study's knowledge, no research study considered the three key SOA size metrics as a whole to compute size.

2.6.1.2 Technical Factors

Most researchers have identified technical factors also known as environmental factor as a contributor to software development size. Technical factor encompasses hardware and software issues that may affect development effort. Technical factors were identified in all other research on software effort estimation with variance in the number of technical sub-attributes and type of sub-attributes. Albrecht (1983) identified 10 technical factors related to hardware and software configuration issues. On the other hand, Use Case Point (UCP), Karner (1993) identified 14 technical attributes that contribute to effort. Furthermore, Boehm (2000) classified technical attributes into two namely product attributes and computer attributes with 7 factors. With advancement of technology, hardware technical factors influence on effort has reduced tremendously.

2.6.1.3 Personnel Factors

Another common factor mentioned in most research on effort estimation is personnel factors also referred to as human factors. Boehm (2000) identified 5 personnel attributes which include analyst capability, application experience, programmer capability, virtual machine experience and programming language experience. Use case point (UCP) captured 8 personnel factors which were grouped under environmental factors. Most recent research studies on effort estimation have identified personnel factors as critical determinant of development effort. Common personnel attributes in literature include programming experience, application experience, system analyst experience and team cohesion (Boehm, 2000; Karner, 1993; Gupta, 2013; Kuan, 2017).

2.6.1.4 Requirements Factors

Software Effort estimation Researchers have also identified requirement issues as factors that determine software development effort. Gupta (2013) identified quality of service factors where he featured factors related to requirement factors such as security objectives. Requirements factors captured in recent literature include business agility, business value and integration with other businesses (Farrag, Moawad & Imam, 2016). Requirement factor was also identified as one of the attributes under environment factors in Use Case Points (Karner, 1993). However, requirements factors including security factors and business value factors were not captured.

2.6.1.5 Service Type Factors

With the introduction of SOA effort estimation methods, service type factors became key attributes that determines SOA development effort. Service type is classified into new service, migrated service and available service (Li & Keung, 2010; Farrag, Moawad & Imam, 2016; O'Brien, 2009). This is informed by the fact that it will take more effort to develop a service from scratch than to discover and use available effort. Service type can also be classified as SOAP and REST based depending on the technology used to develop a service (Belqasmi & Glitho, 2012). Relatively, REST service takes less effort to develop as compared to SOAP service when all factors are held constant. However, based on this study literature, no research study so far has captured both construction type and service development architecture. Construction type is categorized as new, migrated and available service while development architecture is classified as REST and SOAP.

2.6.2 COCOMO 2.0 Model

The theory that best explains the relationship among variables in the area of software effort estimation is the COCOMO 2.0 model (Boehm et al., 1995). COCOMO 2.0 provided a template that considered various types of software development paradigm rather than a specific type of software development method. The COCOMO 2.0 model consider software size, Software reuse and re-engineering factor and effort multiplier factors to estimate software development effort.

COCOMO 2.0 allows use object points or function point or source lines of codes to measure software size (Boehm et al., 1995). After computing size, the model includes other parameters constants to take care of economy of scale and effort coefficient. The size factor discussed in COCOMO provided the foundation towards the definition of size metrics proposed in this study (Boehm et al., 1995). However, COCOMO model is not focused on SOA and thus it does not capture SOA size attributes such as operations count, dependency count and data movement as defined in this study.

Another component that is included in the COCOMO 2.0 model is software reuse and reengineering to compute the amount of percentage of effort required when reusing software (Boehm et al., 1995). The aspect of software reuse is vital in recent software development paradigms including SOA. It is in this backdrop that this study and other SOA effort estimation methods included service type factor which captures new service, migrated service and available service (O'Brien, 2009; Li & Keung, 2010; Farrag, Moawad & Imam, 2016). However, COCOMO 2.0 defined software reuse and reengineering factor as an input to software size but in this research study it is an input to software development effort.

Lastly, COCOMO 2.0 revealed a list of factors known as effort multiplier cost drivers that have significant impact on software development effort. COCOMO 2.0 effort cost drivers include product factors, platform factors, personnel factors and project factors. However, with rapid evolution in software and hardware technologies, influence of these factors to effort has continued to vary with time. For instance, storage space is currently not a significant factor due to improvement in storage technology. Therefore, these factors vary depending on the software paradigm or time when the model was proposed. In this study, these types of factors are classified as product factors, service development environment, requirement factors and personnel factors. The COCOMO 2.0 model revealing the relationship among variables is as shown in Figure 2.3.

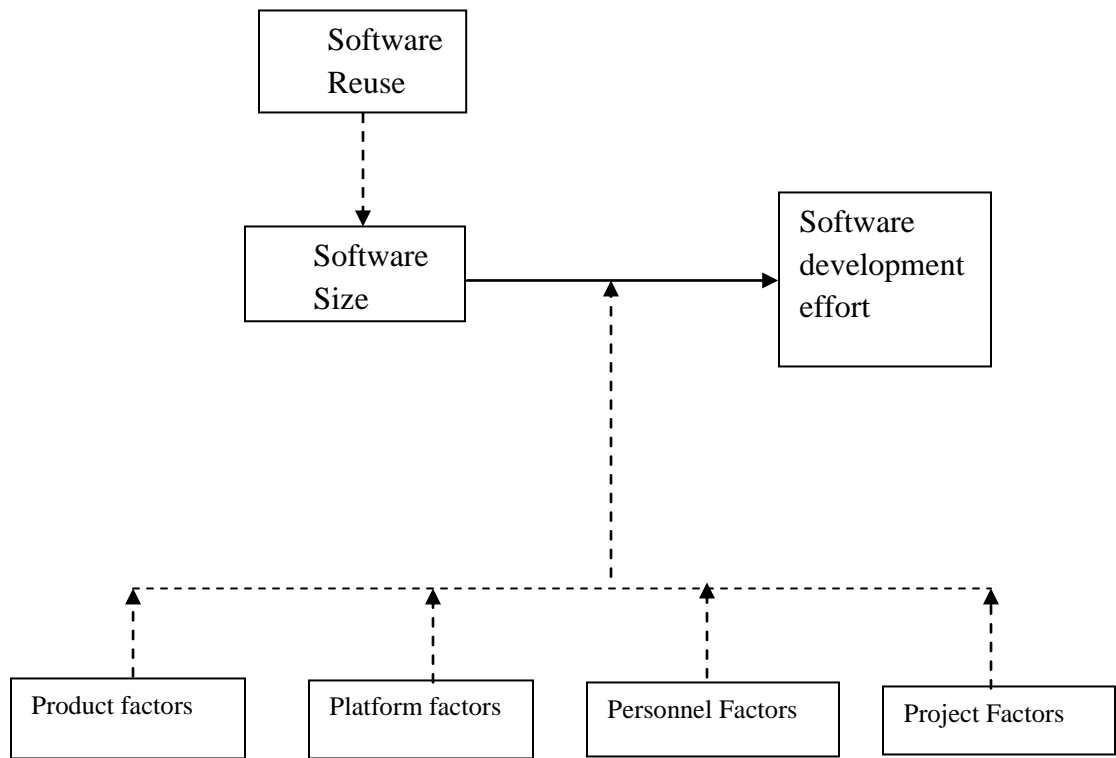


Figure 2.3 COCOMO 2.0 model variables relationship

Figure 2.3 shows an overview of COCOMO 2.0 variables with software size being the main independent variable and software development effort is the dependent variable while software reuse and effort multiplier factors including product, platform, personnel and project factors are moderating variables. This model provided the foundation to development of most software effort estimation methods including the method proposed in this study.

2.7 Conceptual framework

The focus of this study was to develop a SOA effort estimation method based on size metrics and other factors such as service type and SOA effort factors. This study defined SOA size attributes including weighted operations count, service dependency count and weighted message count attributes as independent variables to SOA size which is considered as the dependent variable. On the other hand, SOA size is independent variable in relation to SOA development effort as the dependent variable. Weighted operations count attribute provided an analysis of SOA internal structure to determine SOA size, Service dependency count focuses on relationship and dependency among service while Message count reveals messages exchange between services.

This study, defined SOA size and Service Type Factor (STF) as independent variables to SOA development effort while Effort Multiplier Factors (EMF) as moderating variables. EMF defined in this study includes 12 factors classified as Product factors, Requirement factors, Environment factors and Personnel factors. A conceptual framework showing relationship among variables in the study is as shown in Figure 2.4.

Independent Variable

Dependent Variable

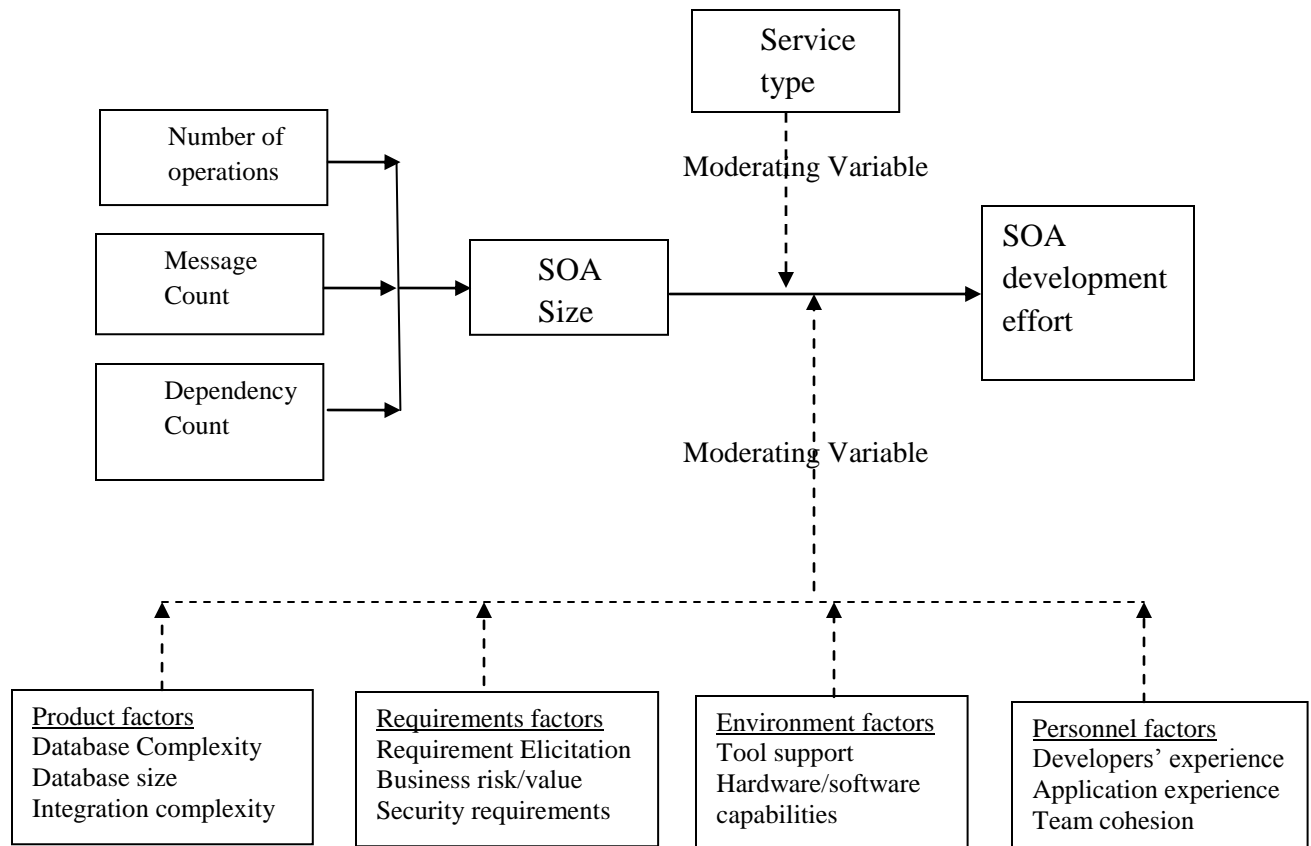


Figure 2.4: Conceptual framework showing SOA effort estimation variables

2.8 Identified gaps

It is clear from literature that existing effort estimation methods are inadequate when estimating SOA as compared to other applications due to SOA complexity. A number of SOA cost estimation frameworks were introduced due to demand for a more accurate SOA effort estimation method. For instance, estimating the cost of development customization to packaged applications using SOA framework (Akkiraju & Geel, 2010) captured SOA cost factors exhaustively by taking into consideration the entire project life cycle but classification of service type was not done.

Phased effort estimation of legacy systems migration to SOA framework (Farrag & Moaward, 2014) and SMART-AUS scope, cost and effort estimation framework for SOA (Obrien, 2009) proposed a wide scope of effort estimation factors based on SOA projects. The methods covered the whole project life cycle. However, they disregarded service size which is a key factor when estimating development effort..

Functional Size measurement method for SOA (COSMIC, 2010) is based on size estimation to estimate SOA development effort. COSMIC provides a detailed analysis of message count as a way of measuring SOA size. COSMIC was adopted by ISO as SOA measurement method (ISO 19761). However, they focused on data movement as an aspect of software size disregarding size indicators such as service internal structure complexity.

BPMN and BPEL models artifacts provided means of estimating effort at an early phase of software development (Mishra & Kumar, 2014; Akkiraju & Geel, 2014). However, BPMN and BPEL artifacts cannot reveal aspect of SOA internal structure, dependency and message count which are critical for computing SOA size. Li & Liam (2010) proposed a method that analyzes effort factors qualitatively but with no empirical analysis and computation which implementation of the method into a tool is rendered practically difficult. This prompted researchers to propose methods that consider structural complexity as a key input to service size and effort (Vaerlaine, Jureta & Faulkner, 2014; Gupta, 2013). However, service structural complexity attributes were not based on any artifacts such as UML. Furthermore, service dependency and data movement were not captured.

2.9 Chapter Summary

Based on literature review, different situations and development environment determine the appropriate software development effort method to use. There are situations where accuracy in software development effort estimation is critical then in this case a more accurate method should be employed, in other instance, winning a contract is important therefore, price-to-win becomes the most appropriate method to estimate software development effort. (Borade & Khalker, 2013).

Over the years software developers have had the interest of estimating accurately the size, effort and cost of developing SOA systems. However, traditional size metrics and effort estimation methods are inadequate when estimating effort for SOA due to their inability to capture SOA features in their estimation. Estimating SOA systems development effort is difficult because they comprise of integration among services within and outside the organization regardless of heterogeneous technology. This prompted introduction of SOA effort estimation methods which still they do not make full use of size metrics to estimate SOA software development effort.

CHAPTER THREE

RESEARCH METHODOLOGY

2

3.1 Introduction

The purpose of this chapter was to describe in detail the research philosophy, research design, population, data collection, sampling, analysis, reliability and validation techniques used in this study.

3.2 Research Philosophy

Research philosophy is about the development of knowledge, the nature of knowledge and important assumptions on the ways the researcher view the world (Saunders, Lewis & Thornhill, 2012). Research philosophy provided a guide on the way data was gathered, analyzed and used to generate knowledge. One of a research philosophy view is Epistemology which describes value systems for different types of research.

Epistemology constitutes “what the researcher think what is important in the area of study”. It is what the researcher value in an area of study. Important aspects of epistemology are pragmatism (Both observable and subjective phenomena), positivism (deals with observable phenomena), realism (focus on explaining within a context) and interpretivism (subjective meaning and social phenomena) (Saunders et al., 2012).

Positivism refers to natural science that emphasize on observable and quantifiable phenomena (Saunders, Lewis & Thornhill, 2012). In this philosophy, quantitative phenomena are of great value within a study. Positivism believes that reality is stable and can be observed from an objective view point to describe a phenomenon. It involves

hypothesis to be tested to confirm or refute a theory. This research study adopted positivism philosophy with a view of working with quantitative and measurable phenomena. This research proposed SOA size metrics and effort estimation methods which are as a result of related attributes/variables which were validated empirically.

There are two types of research approach namely inductive research approach and deductive research approach. Inductive approach starts from a specific view to a general view of a phenomenon while deductive moves from general to specific view of a phenomenon (Saunders, Lewis & Thornhill, 2012). This research study adopted deductive research approach. The study started from a wide variety of factors contributing to SOA development effort, this research study narrowed down to specific factors for SOA software development effort.

3.3 Research Design

Research design is a plan on what needs to be done to achieve the research objectives. This research study used both exploratory and descriptive research. The objective of exploratory research in this study was to identify key variables that influence SOA effort while descriptive research was used to gather experimental results and expert opinion to accurately describe effort estimation factors (Kothari, 2004). Briand's properties theoretical framework was used to validate each size metrics theoretically.

Research investigation was based on a controlled laboratory experiment to validate the proposed SOA size metrics and effort estimation method and to test the accuracy of deep learning techniques used in the automated tool. Experiment was appropriate for this research to provide manipulation and high control of variables under study (Kothari, 2004).

Another advantage of using experiment is the ability to perform statistical analysis using hypothesis testing methods and opportunities for replications (Wohlin, Roneson, Host, Ohlsson, Regnell & Wesslenal, 2000). Experiment was used to validate the proposed SOA size metrics, effort estimation method and automated tool techniques empirically. Research design methods and analysis used to confirm the achievements of each objective are as shown in Table 3.1.

Table 3.1: Research Method per objective

Objective	Research Method
i. To define a suite of size metrics that will be used to measure the identified SOA effort estimation indicators.	Briand’s properties theoretical framework was used to validate Size metrics theoretically. Experimentation and survey were used to validate the metrics empirically.
ii. To develop an effort estimation method for SOA systems based on the size metrics.	Experimentation and survey were used to validate the SOA effort estimation method.
iii. To implement the size metrics and effort estimation method into analysis tool for SOA systems	Experiment was used to validate the accuracy of deep learning techniques. Expert survey was conducted to determine the validity of the implemented tool

Survey enables more variables to be examined as compared to a laboratory experiment (Kothari, 2004). In this study, a survey was carried out to replicate and validate the experiment done by students and to validate more variables. The survey objective was to collect experts’ opinion on the proposed SOA size metrics, effort estimation method and

the implementation automated tool. Results from the experiment and survey were analyzed to reveal the relationship among variables used in the study and to test the accuracy of the proposed metrics, effort estimation method and automated tool.

3.4 Target population

The target population included all third year undergraduate students in Meru University of Science and Technology taking computing related programs including BSc. Computer Science (76), BSc. Information Technology (61), BSc. Computer Technology (45), BSc. Computer Security and Forensics and Bachelor (34) and Bachelor of Business Information Technology (58). Third year computing students were selected due to their knowledge in Programming and Systems analysis and design. On the other hand, Fourth year students were not selected because their focus is on completing their course and they also undertake an academic project in their final year.

Bachelor of Science in Computer Science was selected based on the fact that a course named “CCS 3353 Research Methodology and Group Project” was done at this level which provided an environment to engage them in the experiment as compared to other third year computing students in the University. It was also convenient to meet a class during the allocated time in the timetable rather than assemble students from various groups. Selected students participated in developing web services, tested the proposed size metrics, effort estimation method and implemented tool for the metrics and method. University environment was selected due to convenience, availability and accessibility of resources and control of subjects and participants in the experiment (Kothari, 2004).

To validate existing deep learning techniques used to build the proposed automated tool, this study identified dependency arrows and possible class attribute names to train and test deep learning techniques. Efficient and Accurate Scene Text (EAST) detector (Zhou, 2015), Tesseract OCR with Long Short-Term Memory (LSTM) (Deepa & Lalwani, 2019) and Multi-class SVM required text depicting class attributes names to test their accuracy while ResNet50 CNN (He, Zhang, Ren & Sun, 2016) required arrows with varied arrow heads representing type of relationship to validate ResNet50 CNN detection accuracy.

The second phase involved a survey with a view of capturing expert opinion on the proposed size metrics and effort estimation method. A population of 40 programmers was identified as potential participants in the study out of which the sample was selected. Programmers were selected based on their previous web-service development work and profile. The significance of carrying out this survey was to subject the proposed size metrics and SOA effort estimation method parameters to industry experts and complement the laboratory experiment.

3.5 Sampling and Sample size

A sample of 15 web service projects were used in the experiment. The 15 web services developed by 15 groups of BSc. Computer Science students were subjected to the proposed size-metrics to determine the metrics validity. The projects were also exposed to the proposed estimation method to compare the estimated value with the actual estimation. Non-probability purposive sampling was used to sample all BSc. Computer Science students to participate in this study in the laboratory experiment in the university environment. The study focused on PHP web services with SOAP and RESTful communication protocols.

Datasets were prepared for the purpose of training and testing deep learning techniques used in this study. EAST detector and Tesseract OCT (LSTM) did not require any training while Multi-class SVM was trained using a dataset of 1200 operation names and ResNet50 CNN was trained with a datasets of 900 arrow head images for UML interface and 900 arrow head images for UML sequence diagram. Each technique was validated using 100 images of varying characteristics to determine the techniques accuracy. All datasets used in this study were created by extracting from various sources due to non-existence of class diagram dependency arrows and class attributes names datasets online in one repository.

On the other hand, 27 practicing programmers were selected to participate in the survey based on their response on their experience determined by the number of years they have worked with Web services and their knowledge in PHP-SOAP and PHP REST web services. Furthermore, 20 practicing programmers were selected randomly through simple probability sampling from the group of 27 programmers selected. The selected developers were requested via a questionnaire to give their opinions on the proposed SOA size metrics and effort estimation method.

3.6 Data Collection Instruments

The study used primary data which were collected via questionnaire. The questionnaire contained structured and closed questions. The questionnaire was piloted with a group of 5 programmers to validate the feasibility of the survey questionnaires. Questionnaires were delivered physically and uploaded online to the respondents to reach a wider scope. Clear instructions were provided to respondents on the need to answer questions, the purpose of the questionnaire and to assure them on confidentiality. Collected data was checked for discrepancies, inconsistencies and outliers for correction before analysis.

A questionnaire in Appendix 2 was provided to students to record data concerning SOA project in the experimentation. Students were required to record data on web service projects they developed and record results upon exposing developed web service projects to the proposed metrics and method. On the other hand, questionnaire in appendix 1 was distributed to software industry web service software developers to validate the experiment results. Data collected from industry developers included opinion on relevance of identified SOA size attribute, Effort estimation factors and to test the validity of the proposed metrics, proposed estimation method and implementation tool.

3.7 Reliability and Validity of Data Collection Instruments

Research instruments were tested to confirm their validity and reliability.

3.7.1 Validity

The results are said to be valid if they represents the population adequately away from the experiment setting (Wohlin et al, 2000). The research instrument is said to be valid if it measured what it is supposed to measure. Validity threats included construct validity, content validity, internal validity and external validity. A pilot survey involving 5 programmers in the university was done to test the content and the structure of the instrument.

Content validity is the extent to which a measuring instrument provides adequate coverage for the research study (Kothari, 2004). This study ensured that research instrument questions cover all variables, and give answers to all research questions under study. Thirdly, internal validity was enhanced by designing instruments to measure what the study aims to measure. On the other hand, external threat in this study was as a result of

using students as subjects in the research experiment. External threat was reduced by validating the experiment with a survey based on programmers in software engineering industry.

3.7.2 Reliability

Reliability is the degree at which different rates give consistent estimates of the same phenomenon (Wohlin et al, 2000). A measuring instrument is reliable if it provides consistent results (Kothari, 2004). Reliability threat includes inconsistency results when measuring SOA size and estimating effort for sampled SOA projects. The threats was checked by comparing results of the proposed size metrics and proposed effort estimation method with existing metrics and existing effort estimation method.

3.8 Data Collection Procedure

A pilot survey was carried out with questionnaires issued to 5 programmers to determine the adequacy of the expert survey instrument. Feedback from the 5 programmers helped to improve the survey instrument validity and reliability. Secondly, to identify 20 programmers to participate in the study, 50 simple questionnaires were issued to 50 programmers online asking them if they have ever worked with SOA applications. This was meant to ensure that the research only deals with programmers who have experience with SOA applications. Out of which 27 programmers responded positively with regard to having participated in developing SOA application. A sample of 20 programmers was selected randomly to participate in the study. This study used questionnaires to give respondents adequate time to understand the metrics to enable them fill questionnaires correctly. Questionnaires were hand delivered to the programmers because the study required explaining to programmers on the metrics and effort estimation method. Annex

documents were attached to the questionnaire for further explanation. Students who participated in the laboratory experiment were also guided on how to fill the questionnaire. They were guided on how to participate in developing the applications to be used in the experiment and how to record the experiment results. Outlier and inconsistency data caused by erroneous recording or misunderstanding by students were discarded to ensure data collected reflects the actual results.

3.9 Data Analysis

Data collected was analyzed using statistical techniques which include descriptive and inferential statistics. Descriptive statistics was used to calculate mean, minimum and maximum value, variance and Magnitude of Relative Error (MRE) while linear regression analysis was used to test the correlation among variable.

The main descriptive analysis to be employed was Magnitude of relative error (MRE) and Mean Magnitude of Relative Error (MMRE) to show the deviation in effort estimation between the actual effort and the effort estimated by the proposed estimation method.

$$\text{MRE} = (\text{Actual effort} - \text{Estimated Effort})/\text{Actual effort}$$

Other descriptive statistics included tabulation of data then finding the mean, standard deviation and median time and effort used to complete the project under study. Descriptive analysis was also applied to compute the accuracy of deep learning techniques in the implementation tool. Linear regression analysis was used to test if there is relationship between SOA size and SOA project development effort and relationship between size attributes and SOA size. Statistical analyses for each research objective are as shown in Table 3.2.

Table 3.2: Statistical analyses for each research objective

Objective	Statistical Analysis tools
i. To define a suite of size metrics that will be used to measure the size attributes of SOA software systems	Linear regression analysis Descriptive statistics
ii. To develop an effort estimation method for SOA systems based on the size metrics.	Magnitude of Relative Error (MRE). Mean Magnitude of Relative Error (MMRE) Linear regression analysis
iii. To implement a static analysis tool that computes the size and estimate effort of SOA software systems.	Descriptive statistics

3.10 Ethical Issues

Research ethics were observed when conducting the research to ensure consistency and valid contribution to the industry. Data collected was secured for confidentiality and participants in the laboratory experiment were informed on the confidentiality issues and guided on the need to record the actual result. The research proposal was approved as per letter from the Directorate of Postgraduate studies, Masinde Muliro University of Science and Technology as attached in Appendix 3. An authorization letter by National Commission of Science Technology and Innovation (NACOSTI) in Appendix 4 was provided.

3.11 Chapter Summary

This chapter discussed research methodology techniques employed in this study to guide on the methods and instruments used to collect and analyze data. The chapter described the

direction this research study took with regard research design, target population, sampling technique, data collection, reliability and validity of research instrument and ethical issues.

CHAPTER FOUR

DESIGN OF SIZE METRICS FOR SOA

4.1 Introduction

This chapter provides a detailed analysis of identified SOA size attributes and their respective metrics. Software metrics defined in this chapter are for measuring and qualifying identified attributes relevant for computing SOA size. This chapter provides the solution to the first specific objective in this study which was to define a suite of SOA size metrics. Theoretical validation was used to test the metrics' constructive validity.

4.2 SOA Size Attributes

Since the introduction of Object Oriented Programming to date, software applications have been organized and developed in modular manner. Currently, analysis of software attributes focuses on module internal structure, module interactions and relationships among modular software. Recent software development paradigm such as SOA and component based have also adopted a modular approach by considering internal structures, interaction and relationship among services when analyzing software attributes. In fact, some of Object Oriented Programming metrics are applicable to Service Oriented architecture (Elhag & Mohamad, 2014). SOA size attributes are well captured when analyzing interaction of components within a SOA application in both static design level and dynamic run-time level (Marsyahariani, Daud & Kadir, 2014).

This study considered SOA internal structure, data movement, interaction and relationship among services as key parameters for defining SOA size metrics. The level of abstraction is based on UML static (design-time) design and run-time (dynamic) design. Static metrics

refers to artifacts of software that are taken from design phase level while dynamic metrics are derived when designing the system run-time model (Marsyahariani, Daud & Kadir, 2014). SOA being an architectural style that enables development of services that are modular and integrated can be represented at different levels of abstractions using UML diagram and other extensions of UML such as SOAML (Amsden, 2010).

This study defines three metrics namely Weighted Operation Count (WOC), Service Dependency Count (SDC) and Weighted Service Count (WSC) metric at static level exposed through SOAML and UML class diagram. On the other hand, Weighted Message Count (WMC) metrics is defined at run-time level exposed through UML sequence diagram. Furthermore, SOA size attributes in this study are classified into two categories that is service level metrics and system level metrics. WOC is a service level metric while SDC, WMC and WSC metric are systems level metrics.

4.2.1 Weighted Operation Count (WOC)

Weighted Operation Count (WOC) metric evaluates the internal structure of an individual service by counting the number of operations or methods contained in a service based on their complexity. WOC takes into consideration the number of operations, operations' complexity and operations' parameters to determine the size of a service.

WOC is defined as a set of operations and a set of parameters contained in an operation.

$$\text{Therefore, } WOC(S) = \langle O, P \rangle \text{ Where } O \in S \wedge P \in O$$

S denotes a service, O is a set of operations and P is a set of parameters in an operation.

WOC is based on the hypotheses that the more the number of operations and complexity of operations the greater the size of a service. Consequently, it takes more effort to construct a

service with more and complex operations as compared to a service with fewer and simple operations. WOC metric takes into account the number of parameters in an operation as an indicator of more processes to be done by the operation. The WOC metric counts the number of operations weighted according to their complexity and the number of parameters in an operation as shown in Equation 4.1.

$$WOC(S) = \langle O, P \rangle = \sum_{i=1}^n (O_i + P_i) \quad (4.1)$$

This research study adopted weights of 2 to simple operation, 3 to average operation and 4 to complex operation as shown in Table 4.1. The weights are based on the ratio of number of lines contained in sampled simple, average and complex web services. In addition, a weight of 1 is allocated to each parameter contained in an operation.

Table 4.1: Service Operation weight

Operation type	Examples	Weight
Simple (SO)	Get/ Write operation, Arithmetic Calculations, Simple decision making process	2
Average	Operations based on simple algorithm e.g. Searching, sorting.	3
Complex	Operations based on intelligence techniques, decision support algorithm.	4

WOC is a service level metric captured at static level design based on SOAML service interface diagram which reveals operations and parameters graphically. A sample of SOAML service interface diagram showing service operations and parameters is as shown in Figure 4.1.

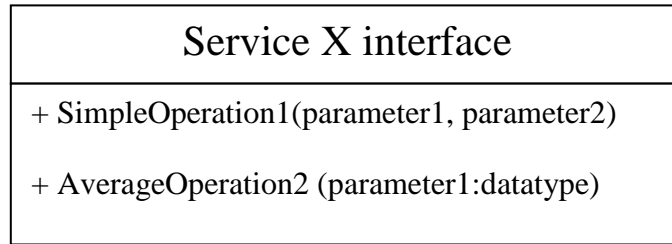


Figure 4.1: Service interface diagram

The proposed Weighted Operations Count (WOC) considers a service interface X in figure 4.1, with the sum of operations $O_1 \dots O_n$. Service X contains two operations namely +simpleOperation1() with 2 parameters and +averageOperation2() with one parameter.

$$WOC(S) = \langle O, P \rangle = \sum_{i=1}^n (O_i + P_i)$$

$WOC(X) = \text{simpleOperation1}() + \text{averageOperation2}()$ web service points

simpleOperation1() = $O + P = 2 + 2 = 4$ web service points

averageOperation2() = $O + P = 3 + 1 = 4$ web service points

$WOC(X) = 4+4 = 8$ Web service points

The size of service X based on its internal structure as revealed by SOAML service interface diagram is 8 web service points according to WOC metrics.

4.2.2 Service Dependency Count (SDC)

Service dependency also known as coupling is the degree of interaction and extent of dependency between services (Sharma, Shewandayn & Bhukya, 2017). This study identified Service Dependency attribute as an indicator of web service size measurement.

A service oriented principle states that services should be “loosely coupled” meaning the nature of interaction should be limited to solely exposing operations for the purpose of

interaction with other services. The study defined Service Dependency Count (SDC) metric to count the number dependencies among services.

SDC captures dependency attributes from UML static design class diagram at system level. SDC focuses on direct dependency which is dependency that exists between a service provider and a service consumer. Based on this study hypothesis, a service with more interaction will have more configurations to link to other services as compared to a service that is linked to fewer services. Implying that a service is bigger in size when it depends on more services or services depends on it and more effort is spent when configuring a service dependency.

SDC considers the type of dependency determined by the depth of relationship between services. The type of dependency based on how deep services are related is known as service composition (Hirzalla, Cleland-Huang & Arsanjani, 2009). Service composition is a collection of related services that take part in solving a specific business function. For service composition to be formed, at least two participant services must be present to complete its functionality (Hirzalla, Cleland-Huang & Arsanjani, 2009). Services that are not in composition are said to be atomic in which case they do not require other services to complete a business process. Dependency attributes according to UML representation are classified based on service composition as atomic point-to-point message exchange, lighter aggregation and strong composition. SDC considers the number of dependency among service multiplied by a weight according to the type of composition between services. Dependencies are allocated weights of 1, 2 and 3 for atomic, lighter aggregation and strong composition respectively as shown in Table 4.2.

Table 4.2 Weighted Service dependency weights

Composition type	Weight (Points)
Atomic (point-to-point dependency)	1
Lighter aggregation	2
Strong composition	3

SDC is defined as a set of types of dependencies among services as shown in Equation 4.2.

$$SDC = \sum_{i=1}^n a + \sum_{i=1}^n g + \sum_{i=1}^n t \quad (4.2)$$

Where a is a set of atomic dependency, g is a set of lighter aggregation dependency and t is a set of strong aggregation dependency.

Atomic dependency is indicated by a dotted arrow in UML diagram while lighter aggregation and strong composition are denoted by a light diamond arrow and a dark diamond arrow respectively as shown in UML class diagram in Figure 4.2.

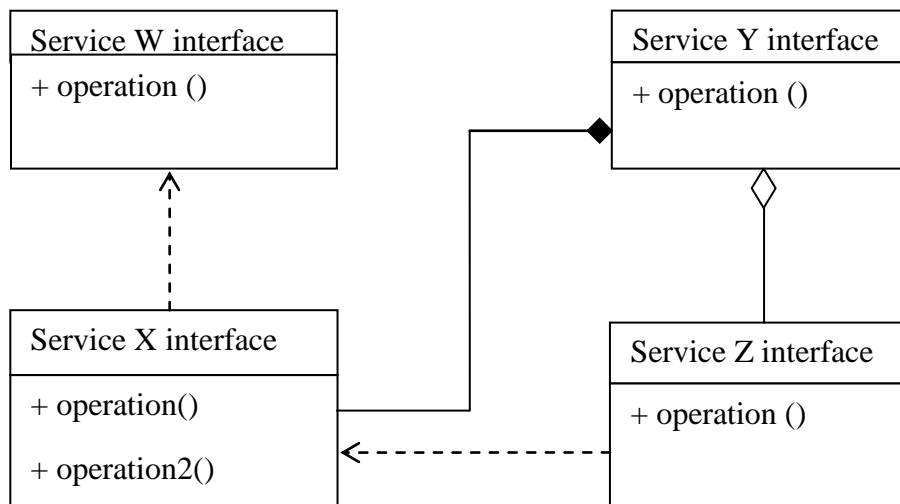


Figure 4.2: UML Diagram showing dependency among services

Figure 4.2 represents 4 service interfaces with X service interface depending on W service interface in an atomic dependency interaction as indicated by dotted arrow. Secondly, service X depends on service Y in a strong composition relationship denoted by a dark

diamond arrow. Service Z depends on service Y in a lighter aggregation relationship denoted by a light diamond arrow and service Z depends on X in an atomic dependency relationship.

The proposed Service Dependency Count (SDC) considers a service interface S, with dependencies $D_1 \dots D_n$ identified in the static service interface UML diagram.

Therefore SDC for the SOA application is, $SDC = \langle S, D \rangle$

$$SDC = \sum a + \sum g + \sum t = 2 + 2 + 3 = 7 \text{ web service points}$$


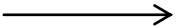

Figure 4.2 reveals 2 atomic (a) dependencies weighted 1 point each, one lighter aggregation (g) dependencies with a weight of 2 points and one strong composition (3) dependency with a weight of 3 points. SDC provides a measurement of SOA size based on interaction among services taking into account the service and type of interaction as stipulated in UML interface diagram.

4.2.3 Weighted Message Count

Weighted Message Count (WMC) represents movement of data groups between services, databases and other applications. Weighted Message Count takes into consideration the type of message call which are classified as synchronous, asynchronous and reply messages. In this regard, data movement specification is linked to the design of information model which is represented by UML sequence diagram. WMC is a system level metric that simply counts the number of messages from and to services. Based on this study hypothesis, there is a relationship between the number of messages in an application and the size of the SOA application because it takes a process to produce a message.

Furthermore, more weight is assigned to synchronous message call because it requires coordination of events to enable message movements in unison. On the other hand, asynchronous message call is assigned lesser weight due its simplicity in design, it does not return a value and no coordination is required to facilitate data movement as compared to synchronous message call. Lastly, reply message carries much lesser weight based on the fact that reply messages are based on conditional tests that will provide error messages or acceptance messages. Weighted Message Count (WMC) assign weights to data movement based on the type of message exchange as shown in Table 4.3.

Table 4.3 Weighted message type arrows

Message type	UML arrow line	Weight
Synchronous		3
Asynchronous		2
Reply message		1

From Table 4.3, each type of message to and from a service is represented by a specific arrow line assigned different weights in relation to the type of message.

Therefore, $WMC(S) = \langle M \rangle$

In this case, M represents a set of messages. M is made up of three types of messages that is synchronous, asynchronous and reply message. Based on UML sequence diagram, data movement to and from a service are identified as shown in Figure 4.3.

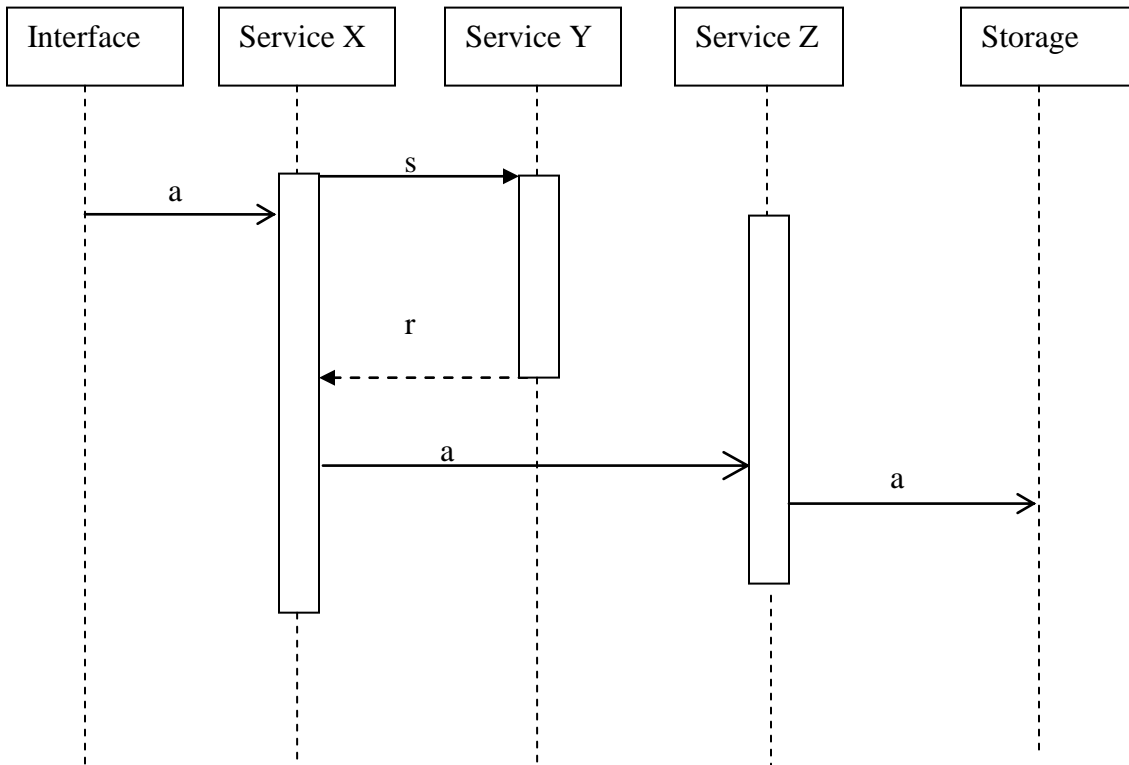


Figure 4.3: UML sequence diagram showing data movement

Figure 4.3 shows five services with horizontal arrows indicating data movement among service. The diagram shows messages labeled a , s and r representing three asynchronous messages, one synchronous message and one reply message respectively.

Equation to compute WMC is as shown in Equation 4.3.

$$WMC = \sum_{i=1}^n s + \sum_{i=1}^n a + \sum_{i=1}^n r \quad (4.3)$$

a = asynchronous message s = synchronous messages r = reply message

Given that asynchronous has a weight of 2, synchronous has a weight of 3 and reply message has a weight of 1.

$$WMC = (a * 3) + (s * 1) + (r * 1) = 6+3+1=10 \text{ Web service points}$$

4.2.4 Weighted Service Count (WSC)

Weighted Service Count (WSC) metric simply sums output derived from WOC, SDC and WMC.

Where,

- WOC is a set of all weighted operations and parameters contained in the operations.
- SDC is a set of all dependencies between a service and other services weighted with regard to the type of composition.
- WMC is a set of all messages to and from a service weighted according to the type of message.

WSC is a systems level metric whose attributes are derived from UML static design diagram. WSC returns size of a SOA system measured in web service points.

Therefore, $WSC = \langle WOC, SDC, WMC \rangle$

According to WSC hypothesis, the more the number of weighted services operations, dependencies and messages contained in a SOA application, the bigger and more complex the SOA application will be resulting to more effort required to build and integrate services. WSC is computed as in Equation 4.4.

$$WSC = WOC + SDC + WMC \quad (4.4)$$

WSC for the SOA application in UML diagrams 4.1, 4.2 and 4.3 is,

$$WSC = 8 + 7 + 10 = 25 \text{ web service points}$$

A case example of application the proposed SOA size metrics in a purchase order SOA application is elaborated in the next section.

4.3 Application of defined SOA size metrics in a Purchase order SOA Application

4.3.1 Purchase Order Business Processing Modeling

Based on SoaML design methodology, design of a SOA system starts with business process modeling which involves capturing the business design from an understanding of business requirements and objectives (Amsden, 2010). The business requirements and objectives are translated into business process specification using Business Process Modeling Notation (BPMN). Services are then identified from the business processes and service specifications are captured through UML diagram to identify methods or operations, data movement and relationship among services.

This study identified Purchase order process as a case example to illustrate the applicability of the proposed SOA size metrics. It involves a consortium of companies with the need to align their purchase order processes to business requirements. A typical Purchase order business processes includes managing purchase order, production scheduling, inventory management, shipping and invoicing. The order process starts by receiving and processing the purchase order which includes item description, item quantity and customer details. The purchase order provides information to calculate the price of items, process production schedule and shipping details. Invoice is then prepared by including the total cost of items, production cost and shipping cost. The identified purchase order business processes based on BPMN is as shown in Figure 4.4.

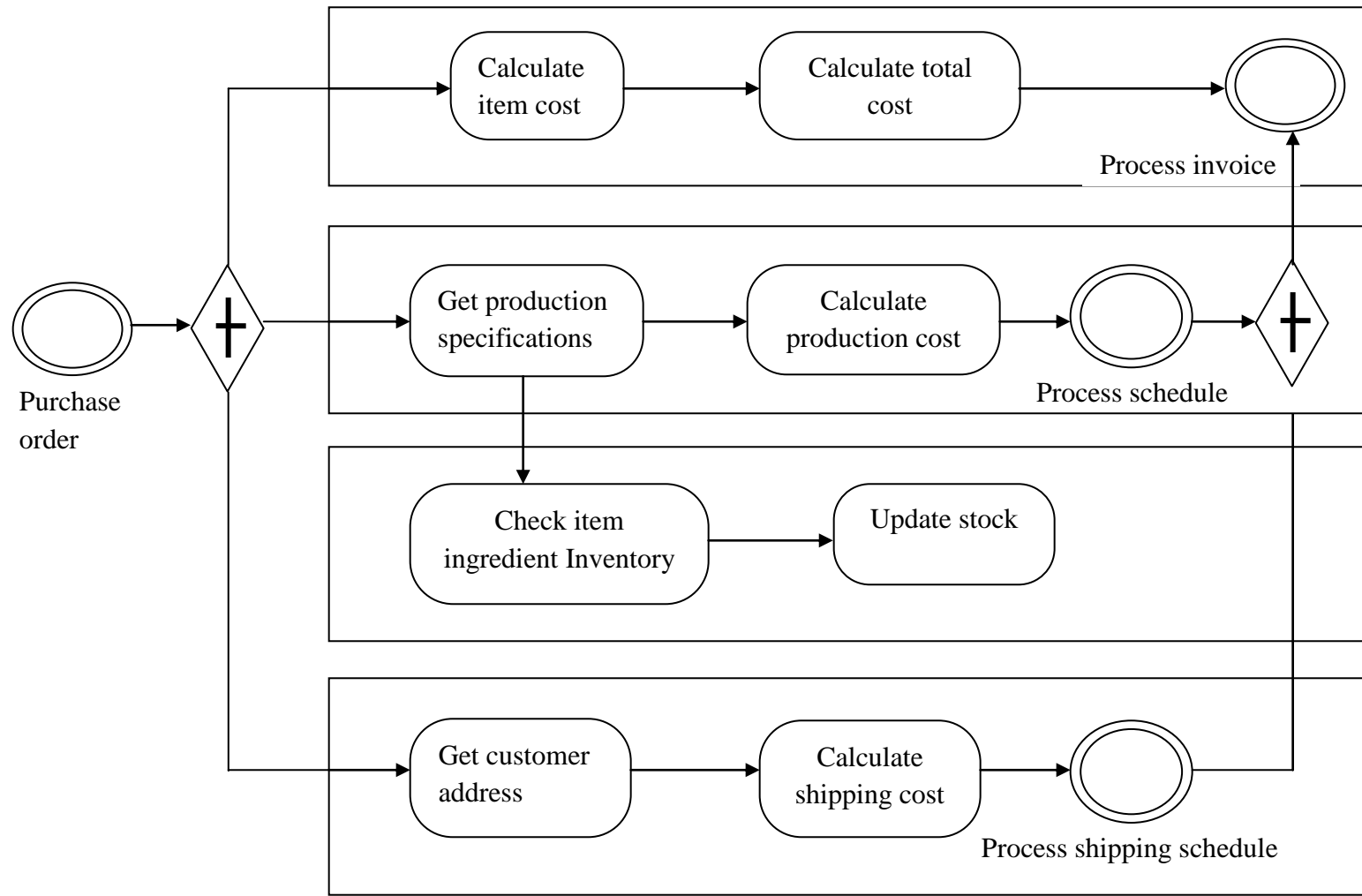


Figure 4.4: Purchase order process Business Processing Modeling Notation

Based on BPMN representing purchase order for a consortium of companies in Figure 4.4, processing of invoice, production schedule, inventory and shipping schedule processes are done at different departments and companies with the main aim of processing invoice to customers. Purchase order process triggers three processes which include preparation of invoice, production schedule process and shipping schedule process each with a number of sub-processes or sub-tasks.

The ultimate goal of the system is to process invoice which rely on other processes output. First of all, input from purchase order process include list of items requested and their quantities to calculate the total items cost. Secondly, input from purchase order process engages the production department to specify the specifications for the items ordered and check in the inventory the availability of ingredient to produce the ordered item. Once a request is made in the inventory, the stock level is updated and a production schedule and production cost are processed and the result is used to update the invoice. Lastly, the third process triggered by the purchase order process is shipping process which receives customer address details to calculate the distance to ship items, shipping cost and provide a shipping schedule. Invoice process captures production cost and shipping cost from production and shipping processes respectively then add to items cost to give the total cost. In some instance, currency conversion process provides functionality that enables foreign currency exchange.

Based on purchase order case example, BPMN diagram in Figure 4.4 helps to break the entire process into a smaller, structured and easy to understand processes which can be adopted by systems developers. However, BPMN level of abstraction does not provide crucial details for software design and measurement. The best lower level of abstraction

that provides design and measuring details is UML diagram. UML interface diagram provides interface diagrams showing number of operations and relationships while UML sequence diagram shows movement of data among different processes or services. SOA size metrics proposed in this study rely on UML design framework to identify SOA attributes relevant for size measurement.

4.3.2 UML Diagram for Purchase Processing System

To measure the size of purchase order process, Business Process Modeling Notation in Figure 4.4 was converted to UML diagram for a detailed and lower abstraction. UML diagram is used to expose number of services, number of operations, number of parameters, relationship among services and data movements among services in a SOA system (Amsden, 2010). These attributes provide key variables when applying metrics proposed in this study. A UML interface diagram representation of the purchase order process in Figure 4.5 shows attributes to use when calculating Weighted Operation Count (WOC) and Service Dependency Count (SDC). On the other hand, UML sequence diagram in Figure 4.6 represents data movement among services which is a key attribute when calculating Weighted Message count (WMC).

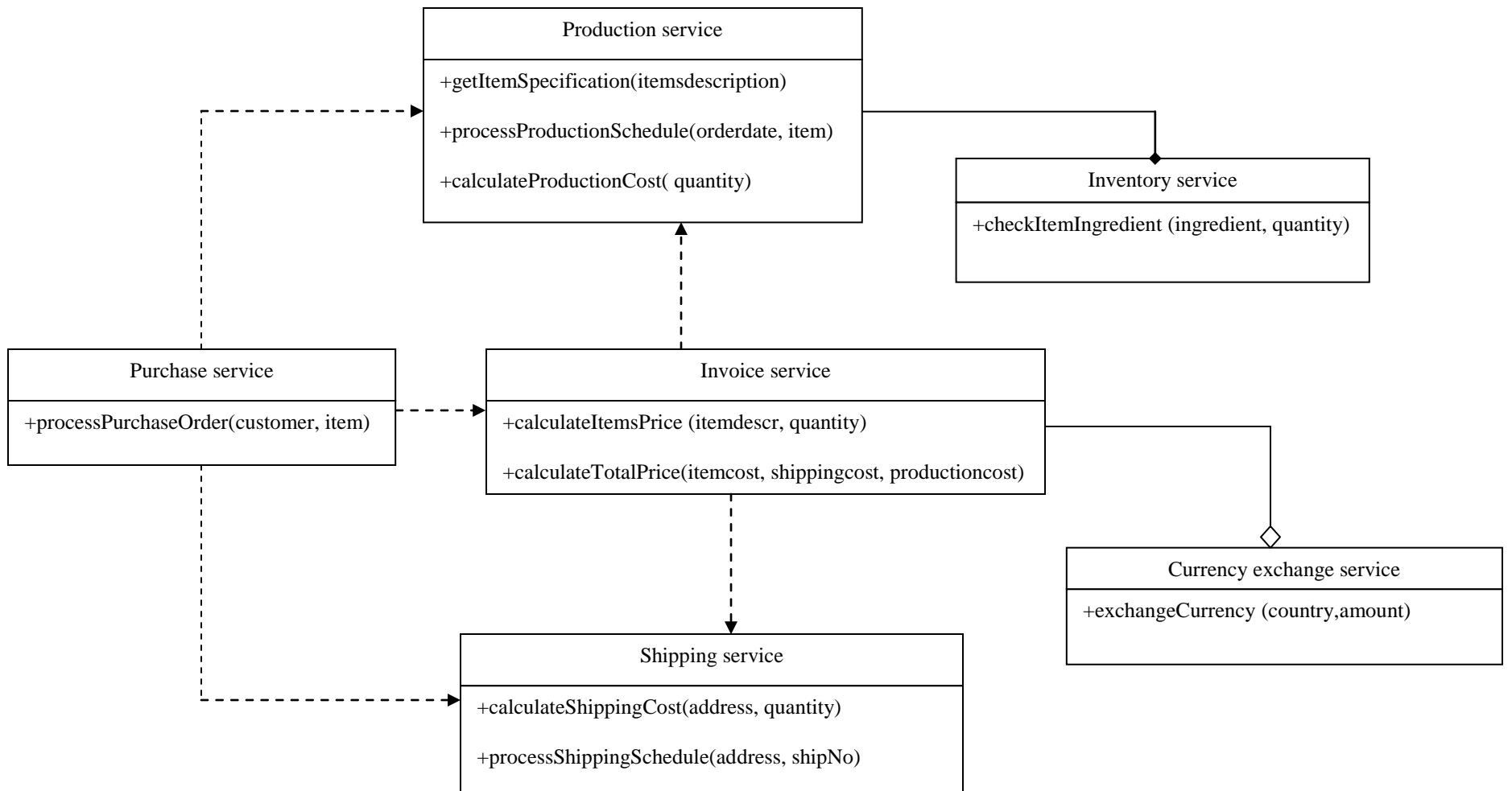


Figure 4.5: UML interface diagram representing purchase order process services

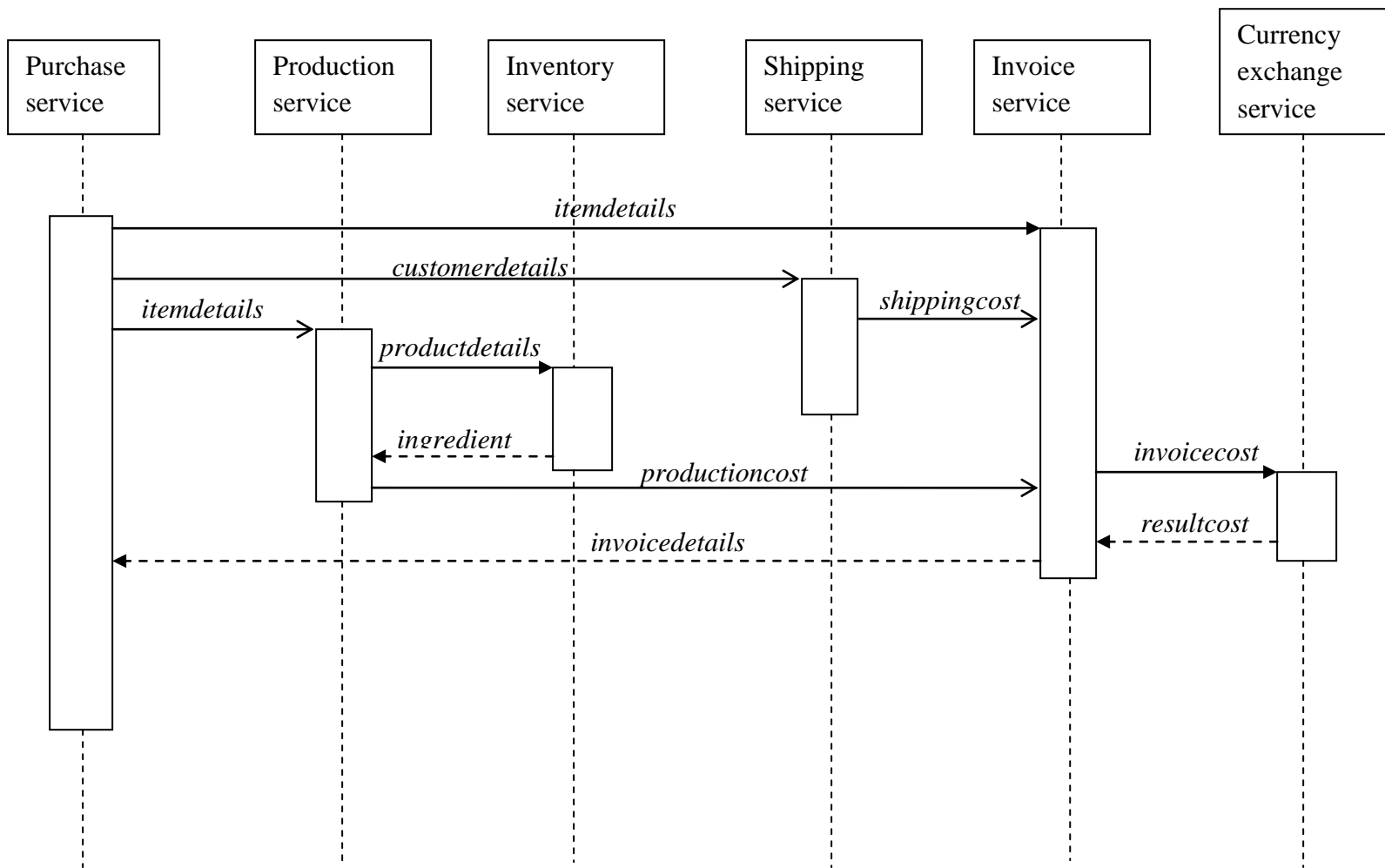


Figure 4.6: UML sequence diagram representing purchase order process services

WOC considers the number of operations contained in each service, complexity of each operation and the number of parameters as key attributes when determining the size of a service with regard to a service internal structure. Based on SOAML service interface diagram representing the purchase order SOA system in Figure 4.5, WOC metric measures the size of a service by counting the number of service operations and parameters.

Secondly, SDC measures the size of a SOA system by considering the relationship among services. SDC counts the number dependencies and type of dependency between services. Based on the UML interface diagram in Figure 4.5, dependencies are indicated by using different types of arrows linking consumer services with provider services with different types of arrows depicting different type of dependency.

Thirdly, WMC counts the number of message movements from a service based on the type of message. Details of message movements are well captured by the UML sequence diagram in Figure 4.6 showing different types of horizontal arrow lines representing different types of messages weighted accordingly. Lastly WSC will take the results of WOC, SDC and WMC for the entire SOA application system to give the final SOA size measure for the purchase order SOA system.

4.3.3 Application of SOA Size Metrics

4.3.3.1 WOC for Purchase Order System

a) **WOC for Invoice service** - Invoice service in UML diagram in Figure 4.5 has two operations with simple arithmetic calculation classified as simple operations each allocated a weight of 2. The first operation has 2 parameters and the second operation has 3 parameters as shown in Table 4.4.

Table 4.4: WOC for Invoice service

Operation	Type	Weight	Number of Parameters	Total weight
<i>calculateItemsPrice (itemdescr, quantity)</i>	Simple	2	2	4
<i>calculateTotalPrice(itemcost, shippingcost, productioncost)</i>	Simple	2	3	5
Total weight (Web service points)				9

$$\text{WOC (Invoice service)} = \sum_{i=1}^n (O_i + P_i)$$

$$= (2 + 2) + (2 + 3) = 9 \text{ web service points}$$

b) WOC for Purchase service - Purchase service hosts one operation classified as a simple operation with two parameters. Therefore,

$$\text{WOC (Purchase service)} = 2 + 2 = 4 \text{ web service points}$$

c) WOC for Production service - Production service contains operations and number of parameters weighted as shown in Table 4.5.

Table 4.5: WOC for Production service

Operation	Type	Weight	Number of Parameters	Total
<i>getItemSpecification(itemsdescription)</i>	Simple	2	1	3
<i>processProductionSchedule(orderdate,item)</i>	Average	3	2	5
<i>CalculateProductionCost (quantity)</i>	Simple	2	1	3
Total weight (Web service points)				11

$$\text{WOC (Production service)} = 11 \text{ web service points}$$

d) WOC for shipping service - Shipping service contain 2 operations and 4 parameters weighted as shown in Table 4.6.

Table 4.6: WOC for shipping service

Operation	Type	Weight	Number of Parameters	Total
<i>calculateShippingCost(address, quantity)</i>	Simple	2	2	4
<i>processShippingSchedule(address, shipNo)</i>	Average	3	2	5
Total weight				9

WOC (Shipping service) = 9 web service points

e) WOC for Inventory service: Inventory service has one operation classified as simple operation with two parameters. Therefore, WOC (inventory) = 2 + 2 = 4 web service points.

f) WOC Currency exchange service

Currency exchange service has one operation with a simple operation having 2 parameters.

Therefore, WOC = 2 + 2 = 4 web service points.

Total WOC for the entire purchase order SOA system is the total sum of all the services

Total WOC = WOC (invoice) + WOC (purchase) + WOC (production)+ WOC (shipping)

+ WOC(inventory) + WOC (currency exchange)

= 9 + 4 + 11 + 9+ 4 + 4 = **41 web service points**

4.3.3.2 SDC for Purchase Order System

SDC for the entire purchase order SOA system is captured at system application level rather than individual service. It entails counting the number of arrow types representing dependency in UML service interface diagram. In this case, UML interface diagram in

Figure 4.5 reveals five atomic dependencies, one lighter aggregate dependency and one strong composition dependency. To compute SDC, the number of type of dependency is considered.

Therefore, SDC (purchase order SOA) = <S, D>

SDC (purchase order SOA) =

$$SDC(Purchase\ order) = \sum_{i=1}^n a + \sum_{i=1}^n g + \sum_{i=1}^n t$$

$$= 5 + 2 + 3 = \mathbf{10\ web\ service\ points}$$

Where a represents atomid dependency, g represents lighter aggregation dependency and t represents strong composition.

4.3.3.3 WMC for Purchase Order System

WMC considers amount of data movement in a SOA system depicted by UML sequence diagram at system level. According to the sequence UML diagram in Figure 4.6, horizontal solid arrow head with continuous line depicts synchronous message, horizontal line arrow head with continuous line represent asynchronous message and horizontal dotted arrow lines represent reply messages. Figure 4.6 reveals 3 synchronous, 3 asynchronous and 3 reply messages. Therefore WMC for purchase order SOA system is,

$$WMC(Purchase\ order) = \sum_{i=1}^n s + \sum_{i=1}^n a + \sum_{i=1}^n r$$

$$WMC (Purchase\ order) = 9 + 6 + 3 = \mathbf{18\ web\ service\ points}$$

Given that asynchronous has a weight of 2, synchronous has a weight of 3 and reply message has a weight of 1.

4.3.3.4 WSC for Purchase Order

To calculate the size of Purchase Order (PO) system involves summing WOC, SDC and WMC to compute WSC.

$$\begin{aligned}\text{WSC (Purchase Order)} &= \text{WOC (PO)} + \text{SDC (PO)} + \text{WMC (PO)} \\ &= 41 + 10 + 18 \\ &= \mathbf{69 \text{ web service points}}\end{aligned}$$

The result reflects the size of purchase order SOA system based on attributes revealed by UML interface diagram and sequence diagram.

4.4 Theoretical Validation of the Proposed Metrics

4.4.1 Overview of Briand's

Metrics development involves 2 stages which include metrics definition and metrics validation (Muketha, Ghani & Selamat, 2010). Metric definition is the actual design of the metrics through identification of key factors and their contribution to the metric. On the other hand, metrics validation is determining the validity of software metrics with respect to the domain under research. There are two types of software metrics validity namely theoretical validity and empirical validity (Srinivasan & Devi, 2014).

In this study, Briand's size properties (Briand, Morasca & Basili, 1991) formed the basis of theoretical validation while survey and experimentation documented in Chapter seven provides a framework for empirical validation to the proposed metrics. Briand, Morasca & Basili(1991) proposed a rigorous mathematical framework based on precise mathematical concept with regards to software size, length and complexity measurements. They defined a concept that takes into consideration a system as an entity that has elements and relationships among elements.

According Briand's size properties framework, the size of a system is a function of size (S) containing sets of elements (E) and sets of relationship among elements (R). The framework defines three fundamental size properties that determine the validity of a software metrics. The three properties include non-negativity, null value and module additivity summarized as follows:

Size Property 1: Non-negativity – The size of a system S is non-negative.

$$S = \langle E, R \rangle \text{ where } S \geq 0$$

Size Property 2: Null value – The size of a system (S) is null if elements (E) is empty.

$$S = \langle E, R \rangle \text{ where } S = 0 \text{ if } E = \emptyset$$

Size Property 3: Additivity – The size of a system (S) is equal to the sizes of its elements.

$$S = \sum (E)$$

4.4.2 Results

i) Weighted Operation Count (WOC) Theoretical Validation

Based on Briand's size property framework, a software size metric should satisfy non-negativity, null value and additivity properties to confirm a metric theoretical validity. In this regard, $WOC = \langle O, P \rangle$ is non-negative given that the size of service operations and parameters cannot be negative. Given a service $S = \langle O, P \rangle$ where $O \in S \wedge P \in O$. $WOC(S)$ involves counting the number of operations and parameters which in this case it cannot return a negative value. Therefore, $WOC(S) \geq 0$ satisfying Briand's size 1 property which states that the size of a system is non-negative.

Secondly, according to Briand's size 2 property, service must have an operation for its size to count. According to WOC, a service size is determined by the number and complexity of

operations and parameters. $WOC(S) = \langle O, P \rangle$ such that if $O = \emptyset$, then $WOC(S) = \emptyset$ conforming to Briand's size 2 property which states that the size of a system is null if it has empty modules (E).

Thirdly, Briand's size 3 additivity property requires that the size of a system should be equal to the total size of all modules. With WOC case, the size of a service is equivalent to the sum of the size of all weighted operations and parameters contained in a service. The size of a service (S) according to WOC is not greater than the size of all operations contained in a service.

$$WOC(S) = M_1 + M_2 + \dots + M_n. \text{ Where } M_1 = \langle O_1, P_1 \rangle, M_2 = \langle O_2, P_2 \rangle \text{ and } M_n = \langle O_n, P_n \rangle$$

Where M represents a set of operations and parameters, O represents weighted operations and P represents parameters. WOC metric meets Briand's size 3 additivity property which demands the sum of a system to be equal to the sum of all modules.

ii) Service Dependency Count (SDC) Theoretical Validation

SDC theoretical validation is based on Briand's property framework to confirm the metric validity. SDC conforms to Briand's size 1 property given that SDC cannot return a negative value as it involves adding weighted operation count and dependencies.

$$SDC(S) = \langle D \rangle \text{ where } D \in S.$$

Therefore, $SDC(S) \geq 0$ and $D \geq 0$ where S represents sets of services and D represents sets of dependencies then the value of S cannot be negative. Consequently, the value of adding dependencies cannot return negative values.

Secondly, SDC meets Briand's size 2 null value property because when there is no dependency (D), SDC will return a null value. When $D = \emptyset$ then $SDC = \emptyset$. Thirdly, the

value of SDC a is equivalent to the sum of all services and dependencies of services. $SDC(S) = D_1 + D_2 + \dots + D_n$ conforming to Briand's size 3 additivity property which states that the size of a SOA System (S) is equivalent to the sizes of its elements.

iii) Weighted Message Count (WMC) Theoretical Validation

Weighted Message Count (WMC) considers the amount of message exchange among services as indicator of size. WMC counts the number of weighted messages to determine the size of a service. In this regard, If M represents message originating from a service, WMC (M) cannot return a negative value therefore,

$$WMC = \langle M \rangle \geq 0$$

WMC (M) satisfy Briand's non-negativity property given that the size of a service is non-negative as it results from summing messages from a service which cannot be negative. Secondly, WMC metric returns a null value if there is no message originating from a service. Therefore,

$$\text{If } M = \emptyset \text{ then } WMC(A) = \emptyset.$$

which conforms to Briand's Null value property which states that the size of a system (S) is null if element (M) is empty. Thirdly, Briand's size 3 property demands that the size of a system should be equal to the sizes of its elements. WMC meets the size 3 property requirements given that, the size of a service S is equal to the sum of the sizes of its messages. $S = \langle M \rangle$ is equal to the size of $S_1 = \langle M_1 \rangle, S_2 = \langle M_2 \rangle \dots S_n = \langle M_n \rangle$.

iv) Weighted Service Count (WSC) Theoretical Validation

Weighted Service Count provides a framework for summing up results from WOC, SDC and WMC. Based on Briand's size 1 non-negativity property, WSC cannot return a negative value because all the WSC ingredients cannot return a negative value. Secondly

$WSC = \emptyset$ IF $WOC \wedge SDC \wedge WMC = \emptyset$ conforming to the size 2 property. Lastly the size of a SOA application is equivalent to the sum of all services WOC, SDC and WMC conforming to Briand's size 3 property.

$WSC (A) = WOC(S_1), SDC(S_1), WMC(S_1) + WOC(S_2), SDC(S_2), WMC(S_2) \dots\dots$
 $WOC(S_n), SDC(S_n), WMC(S_n)$

The size of a SOA application is a result summation of all services WOC, SDC and WMC.

4.5 Chapter Summary

Theoretical validation of the SOA size metrics proves the validity of the metrics in measuring SOA size. The metrics derived from UML diagram provided a framework for identifying key attributes relevant for measuring SOA size. The metrics were designed and applied to a purchase systems example to show the metrics applicability. To establish construct validity, the metrics were subjected to Briand's property framework to establish if the metrics are structurally valid. Later on in this study, an empirical validation will be carried out to prove that the metrics results are consistent with the predicted results.

CHAPTER FIVE

DESIGN OF SOA EFFORT ESTIMATION METHOD

5.1 Introduction

This chapter highlights key attributes for determining SOA development effort. In this context, effort is determined by considering how many programmers are needed to accomplish a task and for how long. The effort of developing an application is largely mental activities which require human activities to complete a task or a project (Li,O'Brien, 2010; Rijwani & Jain, 2016). The unit of measuring software development effort is person-day or person-month. Due to SOA structural differences as compared to other software architecture, SOA development factors or attributes differ from other software development effort attributes.

This study introduced attributes that are specific to SOA in addition to existing software development effort attributes revealed by previous research studies. This study estimates SOA development effort for all development phases including requirement specification, software architecture phase, software construction phase and testing phase because Software developers engage with the system from requirement specifications to software testing (Farrag, Moawad, Imam, 2016). Three main factors identified in this study that contribute to SOA development effort are SOA size, service type and Effort multiplier Factors (EMF). Effort estimation process is relative and uncertain in nature which necessitated the use of fuzzy logic in the proposed effort estimation method to represent SOA Effort multiplier Factors (EMF) in a more representational and accurate manner.

5.2 SOA Size

SOA size is the main attribute that determines SOA development effort. There is a positive relationship between software size and software development effort. The bigger the size of software the more effort is required to develop the software. Most effort estimation methods estimate software development effort as a function of size. They compute effort by considering software size multiplied by proven and tested constants (Boehm, 2000; Albrecht, 1983; Kuan, 2017). This study effort estimation method is based on COCOMO II constant A which represents effort coefficient and scale (exponential) factor B to account for relative economies of scale as shown in Equation 5.1 (Boehm, 2000).

$$\text{Therefore, Effort (PM)} = A * (\text{Size})^B \quad (5.1)$$

This study focused on small and medium sized projects developed in a familiar stable environment with a relatively small team of developers. Small and medium size projects were used due to their availability and they constitute the majority of projects. In this regard, this study used intermediate COCOMO organic coefficient (Boehm, 2000) in Table 5.1 to compute software development effort as a function of SOA size. The size metric from chapter 4 that serves as an input to the proposed method is Weighted Service Count (WSC) which is computed by summing WOC, SDC and WMC.

Table 5.1: Intermediate COCOMO Effort coefficients

Project Type	Coefficient constant (A)	Exponential Scale factor (B)
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

SOA effort is computed at SOA application system level having computed WSC for the entire SOA application. For instance, based on Purchase order SOA application system discussed in chapter 4, the total WSC for the entire SOA application was 69 web service points. Therefore, effort for developing Purchase order SOA application when considering effort as a function of size is,

$$\text{Effort (X)} = A * (\text{WSC})^B$$

This study web service point is closer to Function Points with regard to taking key functional aspects when computing SOA size. COCOMO metrics take KLOC (Kilo Lines of Codes) as the software size input to effort function rather than Function Points. Therefore, Function point must be converted to KLOC to compute software development effort. The same principle applies to web service points computed in this study which is converted to KLOC to be used in effort computation. Based on previous research and validation on function point to KLOC conversion, programming languages including PHP, Java, Perl, JavaScript and C++ Function Point are each equivalent to 53 Lines of Codes (LOC) (Anders, 2018). Therefore, this study adopted 53 LOC per web service point to convert web service point to LOC when computing SOA application development effort.

Therefore, for purchase order SOA application with 69 web service points,

$$\text{LOC} = 53 \times 69 = 3657 \text{ LOC} = 3.657 \text{ KLOC}$$

Effort for developing Purchase order SOA application system as a function of size is,

$$\text{Effort} = (3.2 * (3.657)^{1.05}) = 12.486 \text{ persons per month}$$

Effort as a function of size alone is not enough to estimate SOA development effort effectively. The above effort of 12.486 persons per month is based on Purchase order SOA application size before multiplying with Service Type Factors (STF) and 12 Effort multiplier Factors (EMF) also known as cost drivers. EMF which include product factors,

service development environment factors, requirements factors and personnel factors provide a positive or negative fraction effect on effort.

5.3 Service Type Factors (STF)

This study classified Service type into Service Construction type (SC) and Service Architectural Style type (SA). STF have great effect on SOA development effort especially at service coding or construction phase. For example if a service is discovered type the amount of effort at coding phase is near to zero. STF is determined at service level because different service may have different service type.

5.3.1 Service Construction Type

Service Construction types (SC) is classified into three types based on how the service was realized. The three types are available service, migrated service and new service. Available services is a service that exist to be discovered by consumer services, on the other hand, migrated service is a services that is converted from legacy system to web service application while a new service is built from scratch. The type of service construction type determines greatly the amount of effort required to build a service (Li & Keung, 2010).

5.3.1.1 Available Service

Available service is a service that already exists provided by a third party or from an existing system in the organization which only requires to be discovered. This type of service constitutes minimal construction effort with more effort focused on the requirement specification, design, testing and integration. Effort at requirement specification phase, design, testing phase and integration phases remain the same as effort to develop new service. Consequently, effort at construction phase is minimal to near zero (Farrag,

Moawad, Imam, 2016). Therefore, the amount of effort required to enable discovery of a service is a fraction of the overall effort which is a summation of effort at requirements phase, design phase, testing phase and integration phase excluding effort at coding phase. Based on previous research on effort distribution, more effort is at construction phase with lesser effort at requirement phase as shown in the Table 5.2 (Farrag, Moawad, Imam, 2016).

Table 5.2: Effort distribution among development phases

Phase	Estimated effort (%)
Requirements and Analysis	16
Design	15
Development	40
Testing	22
Integration	7
Total Effort	100

(Farrag, Moawad, Imam, 2016)

According to Table 5.2, effort to discover an available service consist of requirement specification effort, Design effort, testing effort and integration effort which is equivalent to 60% of the entire effort. Therefore, based on Purchase order SOA application, if invoice service is available service then Service Construction Factor (SC) for service (invoice) is 0.6 effort factor based on SC.

5.3.1.2 Migrated Service

Migrated service is created through wrapping or modifying an existing legacy system. Its creation involves effort in all software development phases and a fraction of effort at construction phase. At construction phase, the main effort involved is reengineering a

software module into a service. This requires considerable amount of effort especially effort of understanding the structure of the legacy system and effort to deal with compatibility issues between the legacy system programming language and the service language. This effort is estimated at 50% of effort at construction phase which is equivalent to 20% of overall effort across all phases and 80% of the entire effort across all phases. Therefore, if shipping service in Purchase order SOA application discussed in chapter 4 is a migrated service then effort for developing SC (shipping service) is 0.8 effort factor based on SC.

5.3.1.3 New Service

New service is built from scratch meaning that Effort is required in all the development phases represented by an effort factor of 100% which is equivalent to 1. Development of a new service has no effect on development effort because developers are engaged throughout the development cycle. Therefore taking an example of inventory service in chapter 4 as a new service then SC (Shipping service) is 1 effort factor. Details of service construction type weights are as shown in Table 5.3.

Table 5.3: Effort factors for SOA service types.

Service Type	Available service	Migrated service	New service
Service type Effort Factor	0.6	0.8	1.00

To compute SC for the entire application, SC for all services in a SOA application are multiplied to get Total Service Construction (TSC)

5.3.2 SOA Architectural Style (SA)

SOA architectural style defines that protocols and style for developing web services. The two most common communication architectural style or protocols used in SOA applications are REST (Representational State Transfer) and SOAP (Simple Access Protocol). Basically SOAP and REST are not directly comparable given that SOAP is a protocol that make use of WS* technologies while REST is an architectural style designed to communicate via HTTP protocol (Li & O'Brien, 2010). However, they are comparable in terms of the amount of effort required to build a service using either REST or SOAP.

5.3.2.1 SOAP

SOAP technology exposes a service through a method's logic interface and sends data from one service to another based on a standardized set of message patterns. SOAP relies on XML, WSDL, UDDI, HTTP and WADL standards to interpret, discover, automate and integrate services (Belqasmi & Glitho, 2012) . SOAP is preferred when developing heavy weight applications and is suitable when dealing with more quality of service requirements (Li & O'Brien, 2010). Its Quality of service is based on message layer and functional components located in machines remotely accessed via API.

5.3.2.2 REST

On the other hand, REST uses a consistent interface to access identified resources based on data access method. REST allows a wide variety of data formats such as JSON and XML (Belqasmi & Glitho, 2012). It is faster, easy to integrate with legacy applications and uses less bandwidth enabling faster development of services as compared to SOAP. REST is suitable for light-weight and client driven applications. Currently it is the most common technology for developing web service applications (Li & O'Brien, 2010).

5.3.2.3 Comparison between REST and SOAP Effort Factor

When associating complexity involved in developing a SOAP application and a REST application, it takes more effort to develop a SOAP web application as compared to REST web service application of the same size. So far there is no publication that has compared the two development protocols quantitatively with regard to effort estimation. However, Li & O'Brien (2010) compared effort based on REST and SOAP architectural styles qualitatively. The composition of web service using REST is different from SOAP composition hence effort employed must be different. Through qualitative analysis, REST and SOAP were compared in relation to following hypothesis (Li & O'Brien, 2010):

H1: Increase of information in a service increase effort used to develop the service.

H2: Increase of difficulty of technique used to develop a service increase service development effort.

Based on the two hypotheses, services built using SOAP technology have more information and are difficult to build as compared to REST services. The qualitative analysis assigned a factor of 2 to SOAP and a factor of 1 to REST services effort (Li & O'Brien, 2010). But because our research study adopted COCOMO models' function when defining effort factors, therefore, in this study a factor of 0.2 is the difference between SOAP and REST. SOAP was allocated a weight of 1.2 and REST was allocated a weight of 1.00 as shown in Table 5.4

Table 5.4: Service architectural style weights

Service Type	REST	SOAP
Effort Factor weight	1.00	1.2

Therefore, if invoice service in chapter 4 is a REST service, then SA factor is 1 of development effort while if shipping service is a SOAP service then SA (Shipping) is 1.2 of development effort. STF is computed as shown in Equation 5.2.

Service Type factor (STF) = Service construction type factor * Service architecture factor

$$STF = \prod_{i=1}^n SC * \prod_{i=1}^n SA \quad (5.2)$$

Effort after including STF is shown in Equation 5.3. For instance, if STF for purchase order SOA application is 1.2 of effort,

$$\text{Effort (Purchase order)} = STF * A * (\text{service size})^B \quad (5.3)$$

$$\text{Effort (Purchase order)} = 1.2 * (3.2 * (3.657)^{1.05}) = 14.98 \text{ persons per month}$$

From the above example, SFT has a tremendous impact on software development effort which may be a decreasing or increasing effect on SOA development effort.

5.4 SOA Effort Multiplier Factors (EMF)

SOA development effort is also determined by effort factors also known as cost drivers which are proportional to the amount of effort employed and whose values either increase or decrease effort. Apart from SOA size factor and STF discussed in this study, other 12 SOA effort factors also known as Effort multiplier Factors (EMF) were introduced in this study. EMF have an effect of either increasing or decreasing the amount of development effort. In most cases when EMF is normal it is assigned a value of 1 which has no effect on software development effort. On the other hand, EMF that is assigned a value that is less than one has a decreasing effect on software development effort while EMF with a value greater than 1 has an increasing impact on development effort. EMF is applied at the SOA system application level when computing development effort. EMF were grouped into 4

categories namely Product factors, Environment factors, Requirement specification factors and Personnel factors as shown in Table 5.5.

Table 5.5 SOA Effort multiplier Factors (EMF)

S/N	SOA Effort Factor categories	SOA Effort factors
1.	Product factors	Database complexity & size User interface complexity Integration complexity
2.	Development environment factors	Development tool support Infrastructure capabilities
3.	Requirements specification factors	Requirement elicitation Business risk/value Security requirements
4.	Personnel factors	Web service experience Application experience Programming experience Team cohesion

This research study EMF processing is based on fuzzy logic for the purpose of accurate estimation and to provide a realistic way of representing effort attributes. Fuzzy logic concept provides a platform to represent fuzzy sets, convert crisp data to linguistic variables and use IF..THEN rules fed into an inference system to give a more accurate output (Patra & Rajnish, 2018).

Converting SOA EMF crisp values to fuzzy sets enables smooth transition from one category to another. For example, in real sense complexity value cannot be precisely 0.8 but a value that can also be less than 0.8 or greater than 0.8 e.g. 0.84 or 0.78 and so on. Meaning there is no clear boundaries among various categories when representing categories in fuzzy sets.

The process of fuzzy logic involves a fuzzy membership function used to relate a point in the effort attributes fuzzy sets as a value within [0,1] to determine a degree of membership of an identified attribute value (Thamarai & Murugavali, 2016). Fuzzy rules are then applied on fuzzy sets to determine the estimated effort given sets of conditions. Lastly, fuzzification converts the output from inference control system to total effort multiplier as crisp output which is multiplied with the result of the product of SOA size and service factors to estimate the final SOA development effort as shown in Figure 5.1.

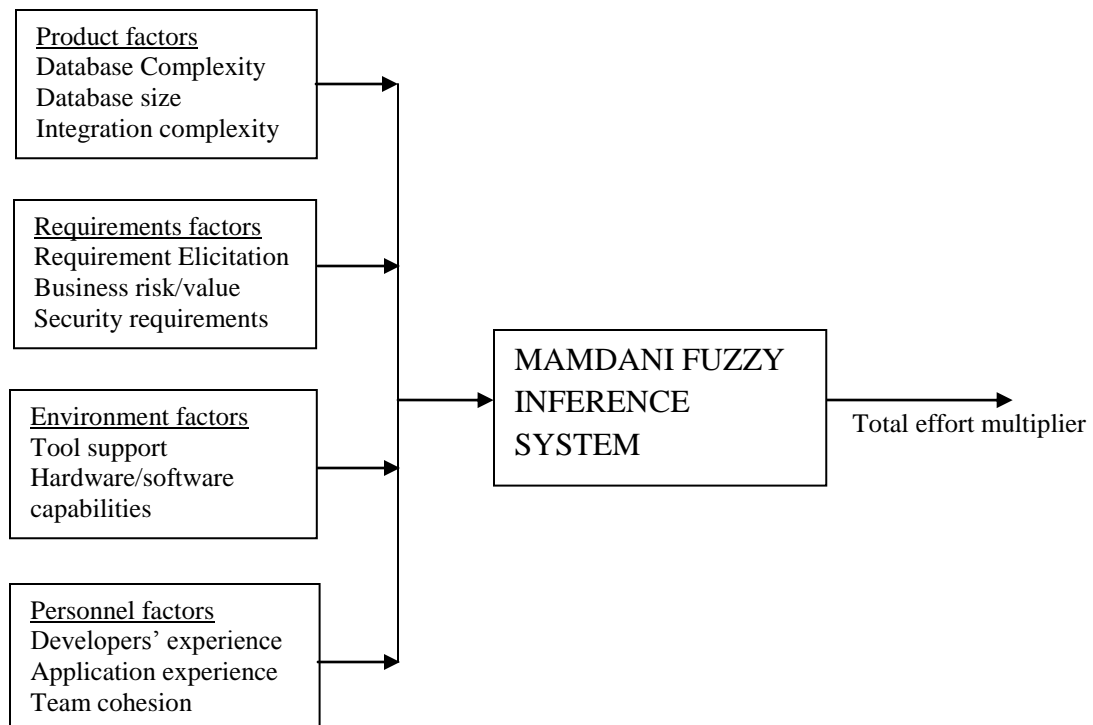


Figure 5.1 : Fuzzy Effort estimation method model

5.4.1 Product factors

Product factors or infrastructure complexity (Gupta, 2013) include elements that add functional value to SOA application with regard to product structure. They constitute the functional aspects of an application including storage of information used by SOA

application, allow data entry and deal with integrating SOA application with other applications. Product factors proposed in this study are database complexity, database size, interface complexity and integration complexity. The complexity and size of product factors have a great impact on SOA development effort. This study adopted COCOMO DATA factors classified as normal, high and very high with values of 1.00, 1.09 and 1.19 respectively for database complexity and database size values. This study rounded off the COCOMO values to 1.00, 1.1 and 1.2. Furthermore, COCOMO DATA factor ranking of low valued at 0.94 was not considered in this study's product factor. This is based on the argument that product factor value cannot be less than one. The values were then represented as 3 fuzzy set namely normal, high and very high linguistic variables.

5.4.1.1 Database Complexity and size

Since 1980s to date database has become a core component in any software application. Database design shows the data management structure for the entire application design. Database complexity or Data access complexity (Gupta, 2013) includes database constraints and commands that affects the complexity of a service. Considerable effort must be put when designing a database for successful implementation of an application systems design. This study proposed Database complexity product factors with value 1 representing normal, 1.1 for High and 1.2 for Very high as shown in Table 5.6

Table 5.6 Database complexity factor

Database Complexity	Normal	High	Very High
Weight Factor	1.00	1.10	1.20

Database complexity is determined by complexity features included in the database management system integrated with the SOA application. For example, a database system that only contains tables and queries with no additional components is classified as Normal. Secondly, a table that requires views and security features and other constraints is classified as high complexity and very high complexity is a database with all other features such as procedures, roles, access rights and database transactions.

Another factor to consider when evaluating database influence on SOA development effort is database size factor. Database size factor simply counts the number of tables contained in a database to determine classification of database factor into Normal, High and Very High as shown in Table 5.7. The more the number of tables contained in a database the more effort is required to design, integrate the database with SOA application and implement the database. In this study, when tables in a database are less than 50, effort factor is classified as normal, while between 50 and 100, effort factor is classified as high and above 100 is classified as very high.

Table 5.7 Database size factor

Database Size	Normal	High	Very high
Weight Factor	1	1.10	1.20

Database complexity and size not only considers Database management systems but also storage in other applications storage structure such as text files. For more accuracy and representational, database complexity and database size factors values are converted to fuzzy sets.

5.4.1.2 Database complexity and Database size fuzzy effort multiplier

Fuzzy logic provides a better way of representing data in fuzzy sets to express data that is unclear and vague in nature. A case in point is complexity of a product which may be subjective from one developer from the other. Secondly, representing data in a class or category provides a wider representation. For example database complexity High value is represented as a range from 1.00 to 1.20 in fuzzy sets rather than a crisp value. Fuzzy logic processes include Initialization, Fuzzification, Inference system and Defuzzification (Ziauddin et al, 2013; Ahlawat & Chawla, 2015; Thamarai & Murugavalli, 2015; Patra & Rajnish, 2018; Kaur, Narula, Wason & Jain, 2018).

i) Initialization: Initialization is the process of defining linguistic variables which are words from a natural language replacing crisp values (Ahlawat & Chawla, 2015). Linguistic variables in this case include database complexity and database size. Each linguistic variable has normal, high and very high linguistic values.

Then, Database size = {normal, high, very high}

Database complexity = {normal, high, very high}

ii) Fuzzification: It is a technique of using membership function to convert crisp data to fuzzy values. It determines the degree to which inputs belong to a particular fuzzy set. There are different types of membership functions including Triangular, Trapezoidal and Gaussian (Thamarai & Murugavalli, 2015). This study used Triangular membership function to convert effort multiplier factors to fuzzy sets as indicated the equation 5.1.

$$f(\mu) = \begin{cases} 0 & \text{for } x < a \\ \frac{x-a}{b-a} & \text{for } a \leq x < b \end{cases} \quad 109$$

$$\frac{c-x}{c-b} \quad \text{for } a \leq x < b$$

$$0 \text{ for } x > c$$

Equation 5.1 : Triangular membership function

The equation takes value x as input and compute membership of x in relation to the fuzzy sets. Triangular membership function considers 4 values including x , a , b and c when computing membership function based on the locations a triangle shaped chart as shown in Figure 5.2.

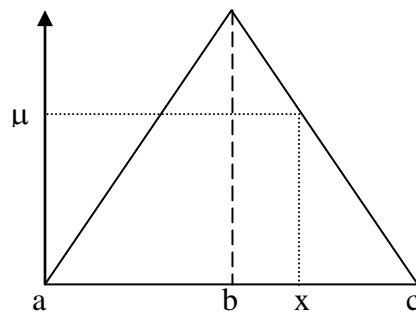


Figure 5.2: Triangular Membership function

According to Figure 5.2, the function μ is a value from 0 to 1 $[0, 1]$ indicating the degree of membership of a particular value x in a fuzzy set represented by the triangle.

Database complexity with three fuzzy sets has 3 triangular shaped charts representing 3 fuzzy sets. The value of crisp data x_1 at any point will results to a function (μ) as shown in Figure 5.3.

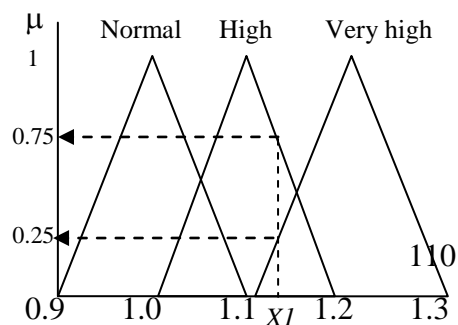


Figure 5.3: Database complexity fuzzification

Given crisp input x_1 at 1.125 in Figure 5.3,

$$\mu_{(x_1 = \text{Normal})} = 0 \qquad \mu_{(x_1 = \text{High})} = 0.75 \qquad \mu_{(x_1 = \text{very high})} = 0.25$$

Figure 5.3 shows that crisp value x_1 is at 0 degree of membership at normal fuzzy set, $\mu = 0.75$ degree of membership at High fuzzy set and $\mu = 0.25$ degree of membership at $\mu =$ Very high degree of membership. Result from the function shows fuzzy logic ability to represent vague and continuous data into a more accurate manner. Triangular membership function is preferred when the range within a set is not high. Database size factor with the same number of fuzzy sets is represented in Figure 5.4 when crisp input value y_1 .

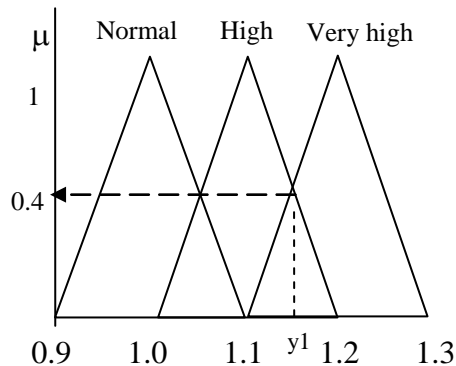


Figure 5.4: Database size fuzzification

Given crisp input $y_1 = 1.15$,

$$\mu_{(y_1 = \text{Normal})} = 0 \qquad \mu_{(y_1 = \text{High})} = 0.5 \qquad \mu_{(y_1 = \text{Very High})} = 0.5$$

Figure 5.4 shows the degree of membership for each database size fuzzy sets when y_1 is 1.15. Results from fuzzy membership function provide inputs to inference engine built with IF .. THEN structure to evaluate the degree of membership output category.

iii) Fuzzy Inference System: It is a fuzzy logic component that evaluates rules in the rule base to determine the outcome of set conditions. Fuzzy inference system employed in this study is Mamdani System which is the most popular inference engine used by Software effort estimation researchers (Patra & Rajnish, 2018; Kaur, Narula, Wason & Jain, 2018). The following rules apply to database complexity and database size factors with regard to SOA development effort as summarized in table 5.5. The results of IF rules were strengthened by calculating W_n for each rule using AND operator in the antecedent. The AND operator is equivalent to MIN function while OR operator is equivalent to MAX function. Database complexity and database size rules were combined using the AND operator to give the expected effort as summarized in Table 5.8.

Table 5.8: Summary of database complexity and database size rules

SIZE	COMPLEXITY		
	Normal	High	Very high
Normal	Normal	High	High
High	High	High	Very High
Very high	High	Very high	Extremely very High

For example when database complexity is high and size is high the effort multiplier is high each allocated a value where Normal = 1, High = 1.1, Very High = 1.2 and Extremely very high = 1.3. The AND operator which is equivalent to MIN was used to compute W_n as follows:

$$W1 = \text{MIN}(\text{Normal}, \text{Normal}) = \text{MIN}(0,0) = 0$$

$$W2 = \text{MIN}(\text{Normal}, \text{High}) = \text{MIN}(0,0.5) = 0$$

$$W3 = \text{MIN}(\text{Normal}, \text{Very high}) = \text{MIN}(0,0.5) = 0$$

$$W4 = \text{MIN}(\text{High}, \text{Normal}) = \text{MIN}(0.75,0) = 0$$

$$W5 = \text{MIN}(\text{High}, \text{High}) = \text{MIN}(0.75,0.5) = 0.5$$

$$W6 = \text{MIN}(\text{High}, \text{Very High}) = \text{MIN}(0.75, 0.5) = 0.5$$

$$W7 = \text{MIN}(\text{Very high}, \text{Normal}) = \text{MIN}(0.25, 0) = 0$$

$$W8 = \text{MIN}(\text{Very high}, \text{High}) = \text{MIN}(0.25, 0.5) = 0.25$$

$$W9 = \text{MIN}(\text{Very high}, \text{Very high}) = \text{MIN}(0.25, 0.5) = 0.25$$

After computing the value of W_n , the next step is to convert fuzzy data sets to crisp output data using a process known as defuzzification.

iv) Defuzzification – Defuzzification is the process of converting output data to crisp output value using a defuzzification method. There are a number of different methods used to defuzzify logic data sets including center of gravity (COG), Center of sums method (COS), Center of Area (COA), weighted average method and Maxima methods. This study employed center of gravity defuzzification method to convert fuzzy sets to crisp data. The crisp data is multiplied with all other crisp output from all factors to give the Total Effort multiplier (TEM).

$$\text{Center of gravity (service type)} = \frac{\sum_{i=1}^n \mu_i \cdot W_i}{\sum_{i=1}^n \mu_i}$$

$$COG = \frac{W1.X1+W2.X2.....Wn.Xn}{W1+W2.....+Wn}$$

$$COG = \frac{(0.5X1.1) + (0.5X1.2) + (0.25X1.2) + (0.25X1.3)}{0.5 + 0.5 + 0.25 + 0.25}$$

$$= 1.183 \text{ of SOA application development effort}$$

Based on database complexity input $x_1 = 1.125$ and database size input $y_1 = 1.15$, the result of COG is a crisp output 1.183 of SOA application development effort which falls in very high effort category coinciding with COCOMO very high effort multiplier value.

5.4.1.3 User Interface Complexity

User interface provides a link between the web service application and the user to enable the user to interact with the system functionalities (Verlaine, Jureta & Faulkner, 2014). Interface carries both functional and non-functional features of the service application. User interface design may be a simple form design or a complex interface design with multimedia interface. Varied amount of effort is required for different types of interface based on complexity as shown in Table 5.9.

Table 5.9 User Interface complexity

Complexity	Normal	High	Very high
Factor	1.00	1.10	1.20

5.4.1.4 Integration complexity

SOA principle is based on integration among services, integration between web services applications and existing legacy applications, integration between web service and database systems and integration between web services within an organization and integration with services outside the organization provided by a third party. Integration effort is inherently the amount of effort used to configure a service to integrate with other resources. Integration with other services is classified as normal, integration with database system outside the application is classified as high while integration with legacy system and services outside the organization is classification as very high as shown in table 5.10.

Table 5.10 Integration complexity

Integration complexity	Normal	High	Very high
Factor	1.00	1.10	1.20

5.4.1.5 User interface and Integration fuzzy logic effort multiplier

i) Initialization – Initialization considers user interface complexity and integration complexity as linguistic variables with the following values linguistic values:

User interface complexity = {normal, high, very high}

Integration complexity = {normal, high, very high}

ii) Fuzzification – Triangular membership function was used to determine the degree to membership given crisp inputs. Fuzzy sets representation for user interface complexity is shown in Figure 5.5.

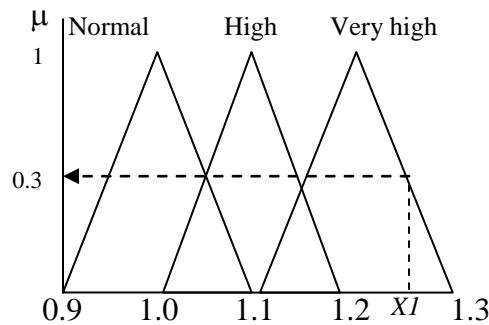


Figure 5.5 : User interface complexity fuzzification

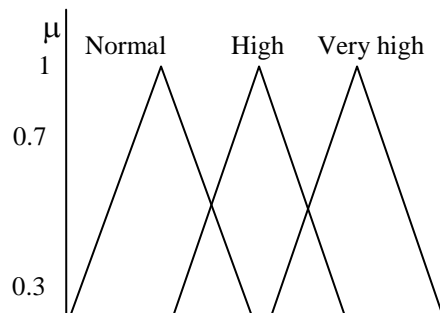
Given examples of crisp input x_1 at 1.27,

$$\mu_{(x_1 = \text{Normal})} = 0$$

$$\mu_{(x_1 = \text{High})} = 0$$

$$\mu_{(x_1 = \text{very high})} = 0.3$$

Considering crisp input $x_1 = 1.27$ in Figure 5.5, x_1 has $\mu = 0$ degree of membership in normal user interface fuzzy set, $\mu = 0$ degree of membership in High user interface fuzzy set and $\mu = 0.3$ degree of membership in very high fuzzy set. On the other hand, integration complexity fuzzy set representation is as shown in Figure 5.6.



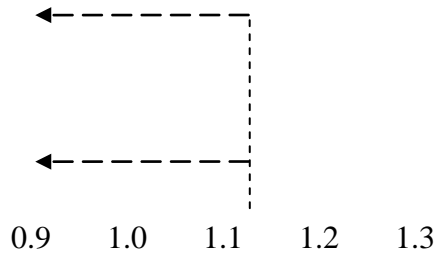


Figure 5.6: Integration complexity fuzzification

Given an example of crisp input $y_1 = 1.13$,

$$\mu_{(y_1 = \text{Normal})} = 0 \quad \mu_{(y_1 = \text{High})} = 0.7 \quad \mu_{(y_1 = \text{Very High})} = 0.3$$

Based on Figure 5.6 example when crisp input $y_1 = 1.13$, degree of membership for the crisp input is 0 for normal fuzzy set, 0.7 for high fuzzy set and 0.3 for very high fuzzy set.

iii) Fuzzy Inference System – Mamdani inference system was used to give the outcome given conditions set by the crisp input. User interface combined with integration complexity in relation to web development effort is summarized in Table 5.11.

Table 5.11: Summary user interface complexity and integration complexity

Integration	User Interface		
	Normal	High	Very high
Normal	Normal	High	High
High	High	High	Very High
Very high	High	Very high	Extremely very High

Table 5.11 illustrates web service development effort result when combining user interface and integration factors. For example when user interface is normal and integration is normal then effort is normal allocated a value of 1. Other development effort weights are High =1.1, Very High = 1.2, Extremely very high = 1.3.

The AND operator in conjunction with W_n was computed to give the following results:

$$W1 = \text{MIN} (\text{Normal}, \text{Normal}) = \text{MIN} (0, 0) = 0$$

$$W2 = \text{MIN} (\text{Normal}, \text{High}) = \text{MIN} (0, 0.7) = 0$$

$$W3 = \text{MIN} (\text{Normal}, \text{Very high}) = \text{MIN} (0, 0.3) = 0$$

$$W4 = \text{MIN} (\text{High}, \text{Normal}) = \text{MIN} (0, 0) = 0$$

$$W5 = \text{MIN} (\text{High}, \text{High}) = \text{MIN}(0, 0.7) = 0$$

$$W6 = \text{MIN} (\text{High}, \text{Very High}) = \text{MIN}(0, 0.3) = 0$$

$$W7 = \text{MIN} (\text{Very high}, \text{Normal}) = \text{MIN} (0.3, 0) = 0$$

$$W8 = \text{MIN} (\text{Very high}, \text{High}) = \text{MIN} (0.3, 0.7) = 0.3$$

$$W9 = \text{MIN} (\text{Very high}, \text{Very high}) = \text{MIN} (0.3, 0.3) = 0.3$$

Defuzzification –center of gravity defuzzification method was used.

$$\text{Center of gravity (service type)} = \frac{\sum_{i=1}^n \mu_i W_i}{\sum_{i=1}^n \mu_i}$$

$$COG = \frac{(0.3 \times 1.2) + (0.3 \times 1.3)}{0.3 + 0.3}$$

$$= 1.25 \text{ of SOA application development effort}$$

The result of COG is a crisp output 1.25 which a very high effort multiplier value to be included in the final computation of development effort.

5.4.2 Service Development Environment

Service development environment include hardware and software required to support development and implementation of SOA application. Service development environment determines the amount of effort with regard to efficiency, constraint and capability of available hardware and software tools. This study focused on tool support and Hardware/software capabilities.

5.4.2.1 Web Service Development Tool Support

Current programming practice involves the use of frameworks and tools that automate web service development as compared to developing a service code by code. Less effort is spend when developing a web service supported by tools and framework rather than writing codes from scratch. IDE developers have incorporated assistants and help techniques that enhances and speed up program design and development. The productivity of a software development team is directly proportional to the development tools employed by programmers in developing the web service (Gupta, 2013).

This study classified software development tools with regard to development effort based on tool usefulness, tool integration with other applications, tools flexibility and collaboration capabilities properties, presence of assistants and Presence of libraries. COCOMO II ranked tool support factor from extra low, very low, low, normal, high, very high and extra high support with an average distance of 0.12 between the categories. This study adopted COCOMO weights but a departed from COCOMO II in the categorization of Software development tool. This study classified software development tools into three categories namely lowly automated, normal and highly automated. The categorization is based on existing development tools for programming languages used to develop web

service applications. Tools that support only coding and compilation are classified as lowly automated, tools with code line assistant and user friendly is classified as normal while fully automated is classified as highly automated. Web service development tool support categories and allocated weights are as shown in Table 5.12

Table 5.12 Web Service development tool support

Tool support	Lowly Automated	Normal	Highly automated
Weight Factor	1.10	1	0.9

5.4.2.2 Web Service Infrastrucure Capabilities

Web services Infrastructure are components that include hardware, networking and software components that enhance smooth development of web service applications. Infrastructure capabilities include hardware, networking and software infrastructure capacity and capability to host, execute and test web services. Web service infrastructure capabilities factor has an impact on web service software development effort. When facilities have low capacity and capabilities to host and enable service development more effort is required as compared to when facilities are capable (Tarawneh, 2011).

Infrastructure capabilities or technical factors are the presence of standard hardware and software infrastructure (Tarawneh, 2011). Hardware in this case includes storage infrastructure, processor and hardware configuration issues. Networking infrastructure includes data communication infrastructure, server and network configuration issues. On the other hand, software capabilities include software integration issues, operating systems compatibility and configuration issues. Table 5.13 shows classification of web service infrastructure capabilities with regard to service development effort.

Table 5.13 Infrastructure capabilities factor

Infrastructure	Very low	Low	Normal
Factor range	1.2	1.1	1

5.4.2.3 Web service development tool and infrastructure fuzzy effort multiplier

i) Initialization: The 2 linguistic variables used in fuzzy logic computation are development tool and infrastructure while their linguistic values are as shown below:

Development tool = {lowly automated, normal, highly automated}

Infrastructure = {very low, low, normal}

ii) Fuzzification – Triangular membership function was used to determine the degree to which development tool crisp input value belong to a particular fuzzy set is represented in Figure 5.7.

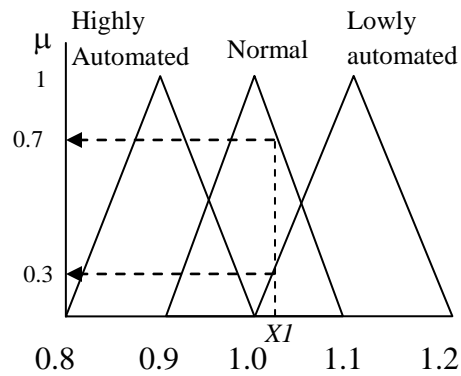


Figure 5.7: Development tool fuzzification

Given examples of crisp input $x_1 = 1.03$,

$$\mu_{(x_1 = \text{Highly automated})} = 0$$

$$\mu_{(x_1 = \text{Normal})} = 0.7$$

$$\mu_{(x_1 = \text{Lowly automated})} = 0.3$$

Figure 5.7 shows web service development tool degree of membership of value $x_1=1.03$ which is $\mu = 0$ in highly automated fuzzy set, $\mu = 0.7$ in Normal fuzzy set and $\mu = 0.3$ in Lowly automated fuzzy set. Web service infrastructure fuzzy sets and crisp input y_1 and degree of membership μ are shown in Figure 5.8

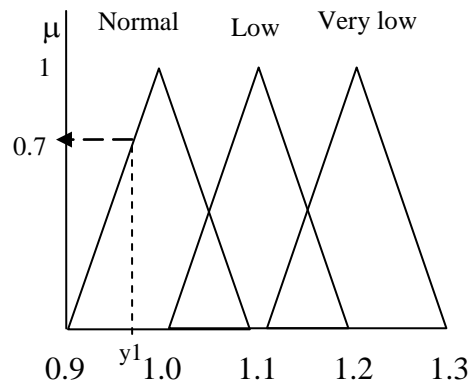


Figure 5.8: Infrastructure capabilities fuzzification

Given an example of crisp input $y_1=0.97$,

$$\mu_{(y_1 = \text{Normal})} = 0.7 \quad \mu_{(y_1 = \text{Low})} = 0 \quad \mu_{(y_1 = \text{Very Low})} = 0$$

Based on Figure 5.8 degree of membership (μ) in Normal set is 0.7 and 0 in Low and Very low sets respectively.

iii) Fuzzy Inference System: Mamdani inference rules combining web service development tool support and infrastructure capabilities for $y_1=0.97$ in relation to development effort are summarized in table 5.14.

Table 5.14: Summary of automated and infrastructure capabilities rules

Infrastructure capabilities	Tool support		
	Highly Automated	Normal	Lowly automated
Normal	Low	Normal	High

Low	Normal	High	Very High
Very low	High	Very high	Extremely very High

Table 5.14 shows the result of effort as Low = 0.9, Normal = 1, High =1.1, Very High = 1.2 and extremely very high = 1.3 when web service tool support and infrastructure capabilities are analyzed. The AND operator which uses MIN was used to compute Wn as follows:

$$W1 = MIN (Highly automated, Normali) = MIN (0, 0.7) = 0$$

$$W2 = MIN (Highly automated, Low) = MIN (0, 0) = 0$$

$$W3 = MIN (Highly Automated, Very low) = MIN (0, 0) = 0$$

$$W4 = MIN (Normal, Normal) = MIN (0.7, 0.7) = 0.7$$

$$W5 = MIN (Normal, Low) = MIN (0.7, 0) = 0$$

$$W6 = MIN (Normal, Very Low) = MIN(0.7, 0) = 0$$

$$W7 = MIN (Lowly automated, Normali) = MIN (0.3, 0.7) = 0.3$$

$$W8 = MIN (Lowly Automated, Low) = MIN (0.3, 0) = 0$$

$$W9 = MIN (Lowly automated, Very low) = MIN (0.3, 0) = 0$$

iv) Defuzzification: Center of gravity was used to convert fuzzy data sets to crisp output also known as effort multiplier.

$$Center\ of\ gravity\ (service\ type) = \frac{\sum_{i=1}^n \mu \cdot W}{\sum_{i=1}^n \mu}$$

$$COG = \frac{(1 \cdot 0.7) + (1.1 \cdot 0.3)}{0.7 + 0.3}$$

= 1.03 of SOA application development effort

The result of COG is a crisp output 1.03 of SOA application development effort which has normal influence on development effort. This result is eventually used to compute the final web service application effort.

5.4.3 Requirement Factors

Requirements are demands or desires or needs defined by stakeholders outlining what must be provided or accomplished by software developers (Hassan & Salman, 2012). Without requirements, you cannot measure success or failure of system development and implementation (Micheal & Boniface, 2014). Critical issues at requirement specification include business value and security requirements to be incorporated in the system. Requirement factors proposed in this study include Requirement elicitation factors, business risk and value and security requirements.

5.4.3.1 Requirements Elicitation

Requirement elicitation is a process of capturing stakeholders' needs and demands. Requirement elicitation provides a framework for ensuring software product compliance with users' needs and demands. It plays a vital role in determining project success and failure (Bormane, Grzibovska, Bervisa & Grabis, 2016). Requirements specification also determines the amount of effort to use in designing, building, testing and implementing a system (Hassim, 2017).

When requirements are clear and unambiguous less effort is used to develop an application as compared to when requirements are unclear and ambiguous. This study proposed 4 classification of requirements elicitation effort factors including very ambiguous, ambiguous, clear and very clear requirements as illustrated in Table 5.15 provides an

analysis of effort multipliers based on requirements elicitation. Weight factor for very ambiguous is 1.30 with an interval of 0.15 all the way to Very clear requirements at 0.85.

Table 5.15: Requirements elicitation effort factors

Requirements elicitation	Very ambiguous requirements	ambiguous requirements	Clear requirements	Very Clear requirements
Description	Programmers unable to envision what is expected.	Requirements are not very clear.	Programmers understand the requirements. Fewer revisions	Provides a clear view of what is expected.
Weight Factor	1.30	1.15	1	0.85

The weights allocated in table 5.15 are converted into fuzzy sets based on the following processed:

i) Initialization: The linguistic variable in this case is requirements elicitation with 4 values namely very ambiguous, ambiguous, clear and very clear as shown below.

$$\text{Requirements elicitation} = \{\text{very ambiguous, ambiguous, clear, very clear}\}$$

ii) Fuzzification – Triangular membership function was used to determine the degree of membership of input x belongs to a particular fuzzy set is represented in Figure 5.9

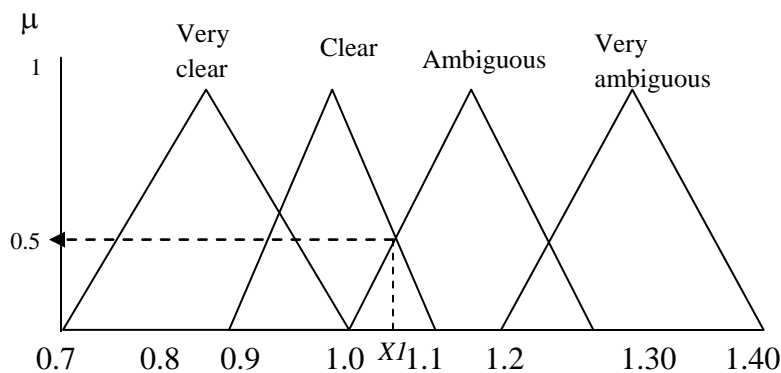


Figure 5.9: Requirements elicitation factor

Given examples of crisp input x_1 at 1.05,

$$\mu_{(x_1 = \text{Very clear})} = 0 \quad \mu_{(x_1 = \text{Clear})} = 0.5 \quad \mu_{(x_1 = \text{Ambiguous})} = 0.5$$

$$\mu_{(x_1 = \text{Very ambiguous})} = 0$$

According to Figure 5.9, crisp data $x_1 = 1.05$ falls under clear requirements fuzzy set by 0.5 degree of membership and 0.5 degree of membership in ambiguous requirements fuzzy set.

iii) Fuzzy Inference System – Rules for requirements elicitation are not combined with other antecedent as in previous examples, therefore simple IF will establish the rules as follows:

IF (requirements = Very clear) THEN effort = Low

IF (requirements = clear) THEN effort = Normal

IF (requirements = ambiguous) THEN effort = high

IF (requirements = very ambiguous) THEN effort = very high

The result from the above rules are effort multipliers including Low = 0.9, Normal = 1, High = 1.1 and very High = 1.2 of SOA development effort.

$$\text{Center of gravity (service type)} = \frac{\sum_{i=1}^n \mu_i W_i}{\sum_{i=1}^n \mu_i}$$

$$COG = \frac{(0.5 \cdot 1.0) + (0.5 \cdot 1.1)}{0.5 + 0.5}$$

$$= 1.05 \text{ of SOA development effort}$$

The result of COG is a crisp output of 1.05 which has a normal influence on SOA development effort. The value is multiplied with other modifiers then the product is multiplied with the effort size function and service type to give the final effort.

5.4.3.2 Business Value

Business value is the perception by the organization on the impact of software product in relation to organization's improvement, survival and image. Business risk is also related to business value in the sense that a system whose failure will have great impact to an organization is valued more. Business risk in relation to software development is a possible negative event that may occur in a business as a result of software implementation failure (Benaroch, Appari, 2010).

More effort is required to build a system that is highly valued and is of high risk to the organization as compared to a system that is lowly valued and low risk (Farrag, Moawad, Imam, 2016). A case in point is an e-business enterprise such as Amazon relies on e-commerce platform for its survival and thus the application is of great value and the risk of failure has a great impact to the organization. Business risk or value in relation to SOA development effort is rated from very low to very high as shown in table 5.16.

Table 5.16: Business value effort factor

Business value	Very low value	Low value	Normal	High value	Very high value
Weight	0.70	0.85	1	1.15	1.30

Based on the relevance and risk that comes with the application system, business value attributes description include; application software an organization can do without classified as very low, application to perform non-functional business classified as low, application to perform a core business process classified as normal, critical system classified as high and very critical system as very high. The weights allocated are converted into fuzzy sets through fuzzy logic process as shown below:

i) Initialization: Linguistic variable in this case is business value with the following linguistic values.

Business value = {very low, low, normal, high, very high}

ii) Fuzzification: Business value fuzzy logic sets are represented using Triangular membership function as shown in Figure 5.10.

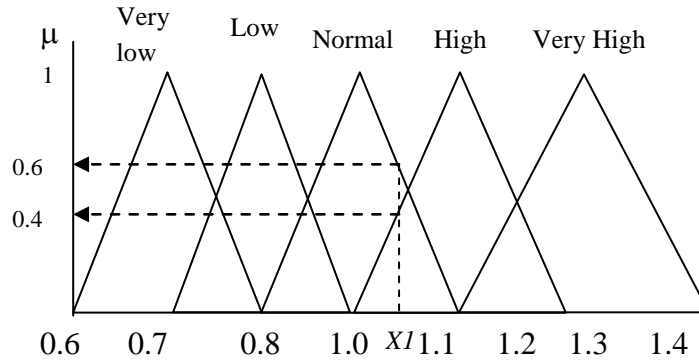


Figure 5.10: business value fuzzification

Given examples of crisp input x_1 at 1.04,

$$\mu_{(x_1 = \text{Very low})} = 0 \quad \mu_{(x_1 = \text{low})} = 0 \quad \mu_{(x_1 = \text{Normal})} = 0.4$$

$$\mu_{(x_1 = \text{High})} = 0.6 \quad \mu_{(x_1 = \text{Very High})} = 0$$

Figure 5.10 shows input crisp value $x_1 = 1.16$ which falls in normal and high fuzzy sets.

The degree of membership (μ) in normal set is 0.4 and high is 0.6 respectively.

iii) Fuzzy Inference System rules: Mamdani inference system IF rules were used to evaluate conditions as follows:

IF (Business value = Very low) THEN effort = very low

IF (business value = low) THEN effort = Low

IF (business value = normal) THEN effort = normal

IF (business value = high) THEN effort = high

IF (business value = Very high) THEN effort = very high

Values representing impact of business value of development effort are very low = 0.8, low = 0.9, normal = 1, high =1.1 and very high = 1.2 .

$$Center\ of\ gravity\ (business\ value) = \frac{\sum_{i=1}^n \mu_i \cdot w_i}{\sum_{i=1}^n \mu_i}$$

$$COG = \frac{(0.6 \times 1.1) + (0.4 \times 1.0)}{0.6 + 0.4}$$

$$= 1.06\ \text{of development effort}$$

Based on the above COG crisp result, 1.06 has normal impact on development effort.

5.4.3.3 Security Requirements

Security requirement is a security condition or capability needed by stakeholders to ensure confidentiality, integrity, availability, authenticity and authorization of an application system (Assal & Chiasson, 2018). For an organization where security is a priority such as financial institutions, they will put more emphasis on demanding applications that are not vulnerable to threats. The degree of security requirements in the system determines the amount of effort required to develop the application. Therefore, system with no security features is classified as very low, low security features is classified as low, confidentiality and authenticity security features is classified as normal, biometric features is classified as high while use of algorithm and encryption is classified as very high. Security requirements factor is categorized as shown in Table 5.17.

Table 5.17 Security Requirements effort multiplier

Security requirements	Very low	Low	Normal	High security features	Very high security features
Factor	0.70	0.85	1	1.15	1.30

Application of fuzzy logic to security requirements is as the following processes.

i) Initialization: Security requirement is the linguistic variable with the following linguistic values:

Security requirements = { very low, low, normal, high, very high }

ii) Fuzzification: Figure 5.11 illustrates crisp input at $x = 1$ is fully low security requirement with 1 degree of membership. In all other sets apart from low set, $x = 1$ has 0 degree of membership which gives a direct result that need no further computation.

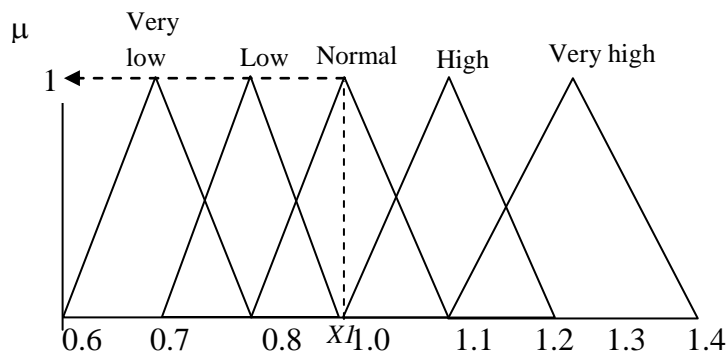


Figure 5.11: security requirements fuzzification

Given examples of crisp input x_1 at 1.0

$$\mu_{(x_1 = \text{Very low})} = 0 \quad \mu_{(x_1 = \text{low})} = 1 \quad \mu_{(x_1 = \text{Normal})} = 0$$

$$\mu_{(x_1 = \text{High})} = 0 \quad \mu_{(x_1 = \text{Very High})} = 0$$

iii) Fuzzy Inference System rules: The rule that applies in this case is IF security requirements = low which has low impact on SOA application development effort.

IF (Security requirements = Very low) THEN effort = very low

IF (Security requirements = low) THEN effort = Low

IF (Security requirements = normal) THEN effort = normal

IF (Security requirements = high) THEN effort = high

Where Very low = 0.8, Low = 0.9, Normal = 1, High =1.1, Very High = 1.2

$$Center\ of\ gravity\ (service\ type) = \frac{\sum_{i=1}^n \mu_i W}{\sum_{i=1}^n \mu}$$

= 1 X 1 = 1 of development effort which has no effect on the value of effort.

5.4.4 Personnel Factors

Personnel factors are attributes and behavior of personnel developing a SOA application system. Personnel factors take into consideration the experience of programmers with the programming language, experience of developers with the application, experience of developers with the architecture e.g. SOA and team cohesion. People or personnel factors are personnel attributes that contributes to SOA development effort (Tarawneh, 2011). Personnel factors proposed in this study include web service development experience, Programming language experience, application experience and team cohesion.

5.4.4.1 Web Service Experience

Developers' experience in web service application is determined by how long developers have worked with web service applications since when they started developing web service applications. The more experienced a web service developer is the less effort the developer will use to develop a web service system as compared to inexperienced web service developer (Kuan, 2017). Table 5.18 shows classification of web service developers based on the number of months and years and weights allocated to each category.

Table 5.18 Web service developer's experience effort multiplier

Personnel factors	Very low	Low	Normal	High	Very High
developer's	0 to 6 months	6 to 9 months	1 year to 2	2 years to 4	4 years and

experience			years	years	above
Factor	1.42	1.17	1	0.86	0.70

This study adopted COCOMO developers' experience weights whose values were then converted to fuzzy sets as follows:

i) Initialization: Linguistic variable in this case is Developers' experience with the following linguistic values.

Web service experience = {very low, low, normal, high, very high}

ii) Fuzzification: Web service experience factor value is converted to fuzzy sets as in Figure 5.12

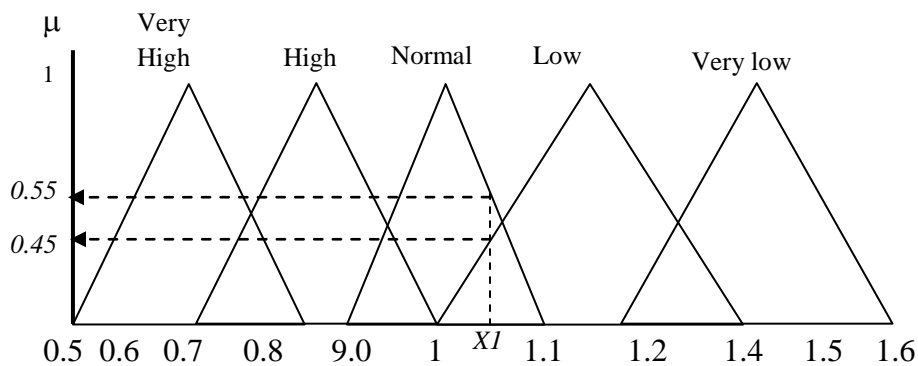


Figure 5.12 Web service developers' experience fuzzification

Given examples of crisp input x_1 at 1.45

$$\mu_{(x_1 = \text{Very low})} = 0 \quad \mu_{(x_1 = \text{low})} = 0.45 \quad \mu_{(x_1 = \text{Normal})} = 0.55$$

$$\mu_{(x_1 = \text{High})} = 0 \quad \mu_{(x_1 = \text{Very High})} = 0$$

Crisp input $x_1 = 1.45$ has a degree of membership (μ) 0.45 in low set and 0.55 in normal logic set. In other logic sets in Figure 5.12, x_1 has 0 degree of membership.

iii) Fuzzy Inference System rules- The following rules were generated to display the outcome of set conditions.

IF (Web service experience = Very low) THEN effort = very low

IF (Web service experience = low) THEN effort = Low

IF (Web service experience = normal) THEN effort = normal

IF (Web service experience = high) THEN effort = high

IF (Web service experience = Very high) THEN effort = very high

Values for effort are very low = 1.4, low = 1.2, normal = 1, high = 0.9 and very high = 0.7

$$\text{Center of gravity (service type)} = \frac{\sum_{i=1}^n \mu_i W_i}{\sum_{i=1}^n \mu_i}$$

$$= (0.55 \times 1) + (0.45 \times 1.1) / (0.55 + 0.45) = 1.05 \text{ which has a normal impact on}$$

software development effort.

5.4.4.2 Application Experience

This factor defines the programmers experience with the type of application. A programmers knowledge on an application determines the amount of effort spend when developing a web service. For instance, a programmer who is not familiar with a Banking application will spend more effort to develop a web service as compared to a programmer who is familiar with the application. This study adopted COCOMO weights in classifying application experience based on average developers' application experience on a particular application type in number of months and years as shown in Table 5.19.

Table 5.19 Application experience effort multiplier

Application	Very low	Low	Normal	High	Very High
-------------	----------	-----	--------	------	-----------

Experience					
Application Experience	0 to 6 months	6 to 9 months	1 year to 2 years	2 years to 4 years	4 years and above
Factor	1.30	1.10	1	0.90	0.80

This study adopted COCOMO developers' application experience weights which were rounded off to the nearest 0.10th whose values were then converted to fuzzy sets as follows:

- i) Initialization: Linguistic variable in this case is application experience with the following linguistic values. Application experience = {very low, low, normal, high, very high}
- ii) Fuzzification: Application experience factor fuzzy sets are as shown in Figure 5.13

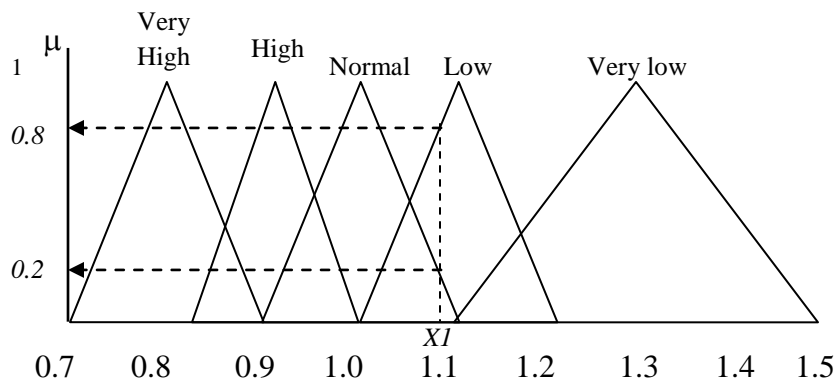


Figure 5.13 Application experience fuzzification

Given examples of crisp input x_1 at 1.08

$$\mu_{(x_1 = \text{Very low})} = 0 \quad \mu_{(x_1 = \text{low})} = 0.8 \quad \mu_{(x_1 = \text{Normal})} = 0.2$$

$$\mu_{(x_1 = \text{High})} = 0 \quad \mu_{(x_1 = \text{Very High})} = 0$$

Values assigned to impact of application experience to effort are very low = 1.3, low = 1.1, normal = 1, high = 0.9 and very high = 0.8

This will affect the following 2 rules,

IF (application experience = low) THEN effort = low

IF (application experience = Normal) THEN effort = Normal

$$\text{COG} = (0.8 \times 1.1) + (0.2 \times 1.0) / (0.8 + 0.2) = 1.08 \text{ of development effort.}$$

5.4.4.3 Programming Experience

Programming experience factor is a measure of how long developers have worked with a programming language. An experienced developer with a particular language understands the language syntax, libraries and other issues regarding the language including the programming environment. Therefore, the developers experience in a particular language determines the amount of effort used to develop a web service application. This study adopted COCOMO programming experience as shown in Table 5.20.

Table 5.20 Programming experience effort multiplier

Application Experience	Very low	Low	Normal	High	Very High
Application Experience	0 to 6 months	6 to 9 months	1 year to 2 years	2 years to 4 years	4 years and above
Proposed factor	1.30	1.10	1	0.90	0.80

This study adopted COCOMO developers' application experience weights which were rounded off to the nearest 0.10th whose values were then converted to fuzzy sets as follows:

i) Initialization: Linguistic variable in this case is application experience with the following linguistic values.

$$\text{Application experience} = \{\text{very low, low, normal, high, very high}\}$$

ii) Fuzzification: Application experience factor value was converted to fuzzy sets as in Figure 5.14.

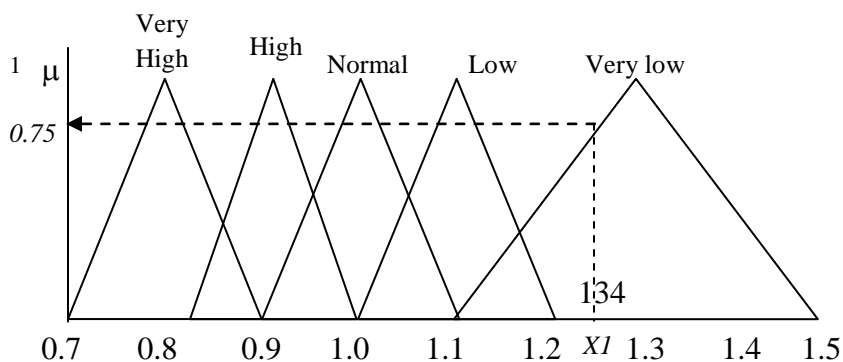


Figure 5.14: Programming experience fuzzification

Given examples of crisp input x_1 at 1.25

$$\mu_{(x_1 = \text{Very low})} = 0.75 \quad \mu_{(x_1 = \text{low})} = 0 \quad \mu_{(x_1 = \text{Normal})} = 0$$

$$\mu_{(x_1 = \text{High})} = 0 \quad \mu_{(x_1 = \text{Very High})} = 0$$

Values assigned to impact of application experience to effort are very low = 1.3, low = 1.1, normal = 1, high = 0.9 and very high = 0.8

IF (application experience=very low) THEN effort = very low
= $0.75 \times 1.3 = 0.975$ of development effort.

5.3.4.4 Team Cohesion

Team cohesion factor takes into consideration the team members shared vision, teamwork and consistency of members' objectives. SOA involves integration of services which requires team dynamics and collaborations (Gupta, 2013). Software development is a teams' effort that requires coordination among members with regard to integrating service developed by different programmer. A team that is cohesive encourages seamless communication among members and willingness to accommodate other members. Team cohesion is rated from low to very high based on the level of team cohesiveness. The more a team is cohesive the lesser effort is spent to develop a web service application system as compared to a team that is less cohesive. Table 5.21 shows team cohesion factor weights and description.

Table 5.21 Team Cohesion factor

Personnel factors	Very low	Low	Normal	High	Very High
Team Cohesion	Highly intolerable team with irreconcilable objectives	Intolerable team with irreconcilable objectives	accommodate opinions & reconcilable objectives	Consistency of objectives and	Shared long term vision and objectives
Factor	1.30	1.15	1	0.85	0.70

Applying fuzzy logic to a value $x = 1$ then,

$$\mu_{(x1 = \text{Very low})} = 0 \quad \mu_{(x1 = \text{low})} = 0 \quad \mu_{(x1 = \text{Normal})} = 1$$

$$\mu_{(x1 = \text{High})} = 0 \quad \mu_{(x1 = \text{Very High})} = 0$$

This will have an impact on one IF rule that is .IF (Team Cohesion=Normal) THEN effort = very low = $1 \times 1 = 1$ of development effort.

5.5 Effort Estimation Method Example

The proposed SOA effort estimation method predicts effort for SOA application software based on size metrics proposed in chapter 4 in web service points. The total effort for developing the entire system is calculated by taking into consideration the SOA application size, product of STF and product of EMF. Each of these factors have significant impact on SOA development effort. Given purchase order SOA application discussed in chapter 4 with size of 3.657 KLOC and STF = 1.2, The Final Effort is computed by considering the product of EMF.

Assuming EMF values for each factor was computed as shown in Table 5.22 Therefore final effort is computed as shown in Equation 5.4.

$$\text{Final Effort (Purchase order)} = \text{STF} * A * (\text{Application size})^B * \prod_{11}^n \text{EMF} \quad (5.4)$$

Table 5.22 Purchase order SOA application EMF

S/N	SOA Effort Factor categories	SOA Effort factors	Rate	Factor value
1	Product factors	Database complexity & size (DC)	Small	1.15
		User interface complexity (UIC)	Simple	1
		Integration complexity (IC)	Simple	1
2	Service development environment	Development tool support (DT)	Normal	1.15
		Infrastructure capabilities (FC)	Low	1.15
3	Requirements specification factors	Requirement elicitation (RE)	Clear	0.85
		Business risk/value (BR)	Low	0.85
		Security requirements (SR)	Normal	1
4	Personnel factors	Web service experience (SE)	Low	1.15
		Application experience (AE)	Very low	1.30
		Programming Experience (PE)	Very low	1.30
		Team cohesion (TC)	Normal	1

$$\begin{aligned}
 * \prod_{11}^n EMF &= DC * UIC * IC * DT * HS * RE * BR * SR * DE * AE * PE * TC \\
 &= 0.8 * 1 * 1.15 * 1 * 1 * 1.15 * 1.15 * 0.85 * 0.85 * 1 * 1.15 * 1.30 * 1 * 1 = \mathbf{1.314}
 \end{aligned}$$

Therefore, Final Effort (Purchase order) = STF * A * (service size)^B * $\prod_{12}^n EMF$

$$\text{Effort (Purchase order)} = 1.2 * (3.2 * (3.657)^{1.05}) * 1.314 = \mathbf{19.683 \text{ persons per month}}$$

In the above example, effort is estimated as a function of service size then multiplied by the product of service factors and product of effort multiplier factors value to give the final effort. Effort computed before including EMF was 14.98 persons per month but after inclusion of EMF product, final effort was 19.683 persons per month. This shows that the product of EMF has a great impact on development effort.

5.6 Chapter Summary

SOA Effort estimation method provides predictions on the amount of effort to use when developing a SOA system. Effort estimated in this case constitutes effort across all phases of software development from requirements specification to Implementation. Effort is a

key component derived from SOA size with an aim of accurate estimation of effort, cost and scheduling. Empirical validation was employed to the proposed effort estimation method to ascertain its validity and accuracy.

CHAPTER SIX

IMPLEMENTATION OF SIZE METRICS AND EFFORT ESTIMATION TOOL (SOA-SMET)

6.1 Introduction

This chapter provides a detailed description of a prototype tool named SOA Size Metrics and Effort Estimation Tool (SOA-SMET) based on the proposed SOA size metrics and effort estimation method. The chapter describes elements of the tool's development process from requirements specification and prototype architecture and implementation.

6.2 Requirements of SMET

The main objective of SOA size measure and effort estimation prototype tool is to allow entry of SOA size and effort attributes, compute SOA size and estimate effort then display the result. Requirements of SOA size metrics and effort estimation tool includes:

- i) Allow entry of SOA size attributes which includes number of weighted service operations, number of parameters contained in an operation, number of weighted dependency, number of weighted messages and number of weighed services. The tool provides two options of entering size attributes which include:
 - a) To automated feature extraction from UML using deep learning techniques. These techniques include EAST detector to detect service operations names, Tesseract OCR to recognize service operations names and Multi-class SVM to classify operations into simple, average and complex. On the other hand, ResNet50 CNN is used to detect arrows depicting dependency and message exchange among service interfaces.

- b) The other method for entering SOA size attributes is through direct entry by manual entry of values via form text box elements.
- ii) Use algorithms to compute Weighted Operation Count (WOC), Service Dependency Count (SDC), Weighted Message Count (WMC) and Weighted Service Count (WSC).
- iii) Allow users to select service types and SOA factors that have impact of SOA development effort.
- iv) Use an Algorithm to compute Effort for SOA application based on SOA size, product of service type factor and EMF product.
- v) Display the result of WOC, SDC, WMC and WSC computation for each service.
- vi) Display effort estimated for the entire SOA application.

6.3 Architecture of SMET

Software architecture is an organization of different software components to enhance seamless information sharing among software components (Cao, Wei & Qin, 2013). In this study, SMET was constructed based on automated UML Feature Extraction component, manual SOA size attributes entry, SOA size metrics computation and effort computation components as shown in Figure 6.1. The automated UML feature extraction component consists of machine learning and deep learning techniques to detect and recognize text, classify text and detect UML dependency and message exchange arrows. On the other hand, manual SOA size attributes value entry component, allow manual input of SOA size attribute values via the keyboard and mouse click. Upon entry of values automatically or manually, the tool computed SOA size metrics and SOA development effort.

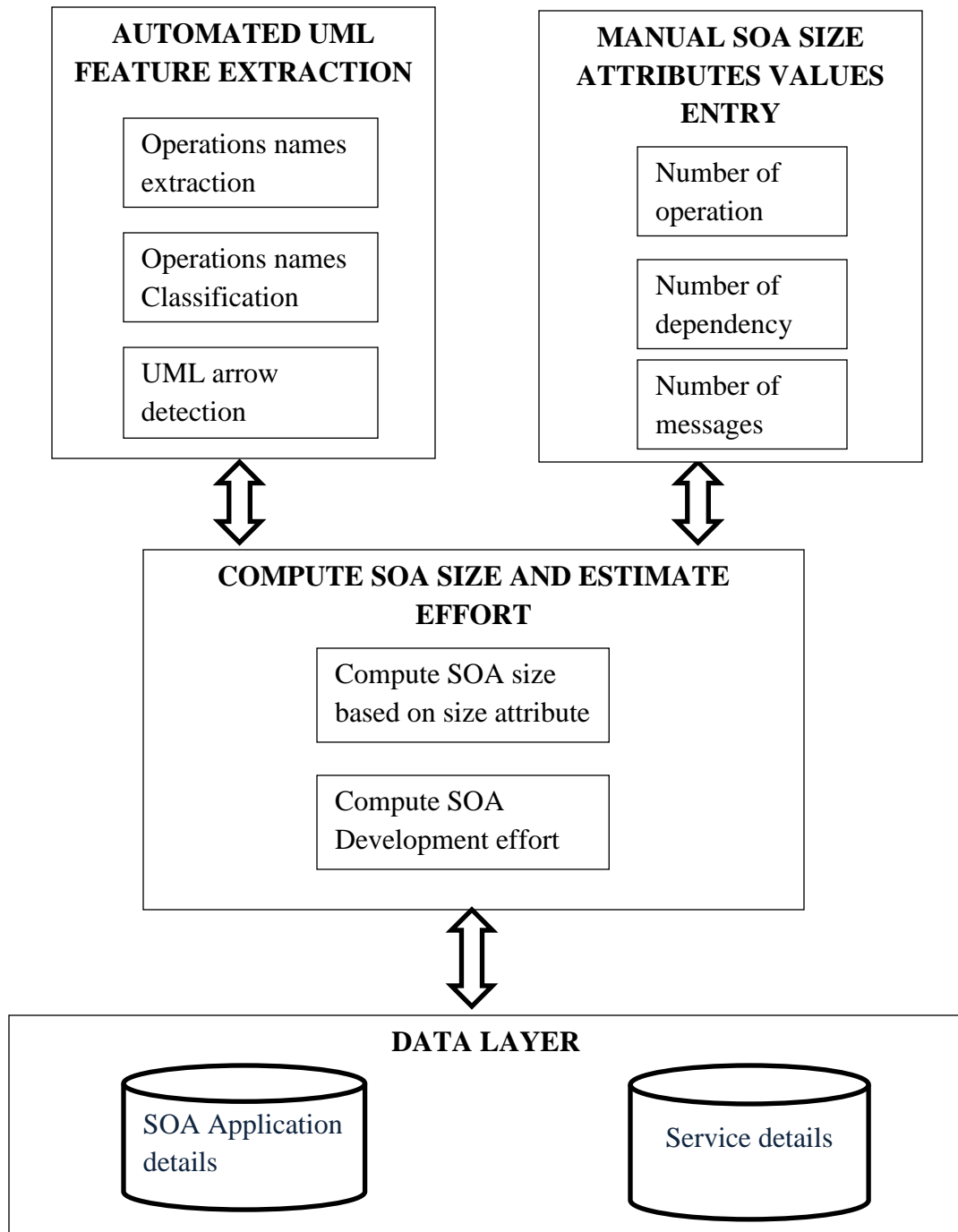


Figure 6.1: SMET architecture

6.3.1 Automated UML Feature Extraction Component

The automated UML feature extraction component of the prototype tool provides a platform for detecting UML service interface operations, service dependency and message

exchange arrows automatically. One of the input components for SOA-SMET is the UML image recognizer which enables uploading of UML service interface diagram and UML sequence diagram images in picture file formats including bmp, gif, png and jpg. Operation names text from UML diagram that represent SOA attributes are extracted through deep learning text detection, text recognition and text classification. In addition, dependency and message exchange arrow head are also detected using deep learning technique.

6.3.1.1 Service Operation Names Extraction

Text embedded into an image can easily be manipulated once extracted from the image. Text extraction involves 2 steps which includes text detection and text recognition. Text detection determines the presence of text in an image while text recognition identifies the text detected. The tool was required to extract service operation names for the purpose of classifying the names as simple or average or complex to compute Weighted Operation Count (WOC). For instance, given a UML diagram in Figure 6.2, the task was to extract operation names listed in lower rectangles representing service operation names.

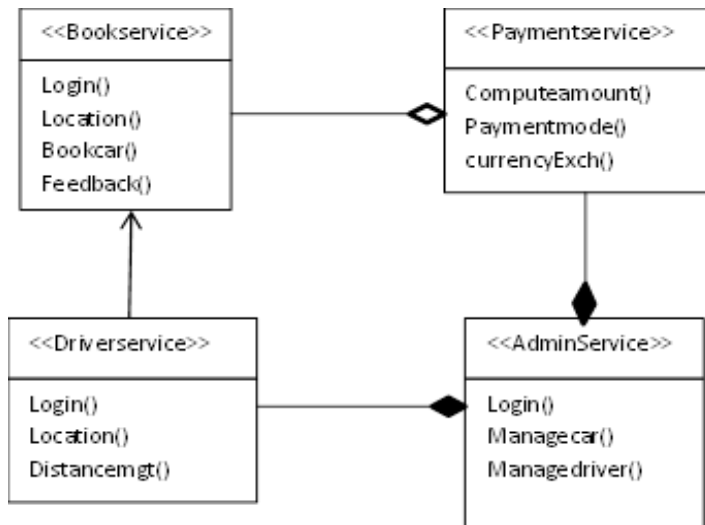


Figure 6.2: Taxi Service UML interface diagram

This study used an existing deep learning technique known as Efficient and Accurate Scene Text detection (EAST) pipeline to identify operation names contained in UML diagram images. EAST text detector is a Fully Convolution Network (FCN) deep learning technique which is efficient and effective when dealing with different types of text including text of different shapes and texts with different shades and fades (Zhou et al., 2015). This study used python and openCV library to implement EAST detector. The implementation utilized feature maps to determine the probability of regions containing text in an image and defined coordinates of text bounding box by highlighting text as shown in Figure 6.3.

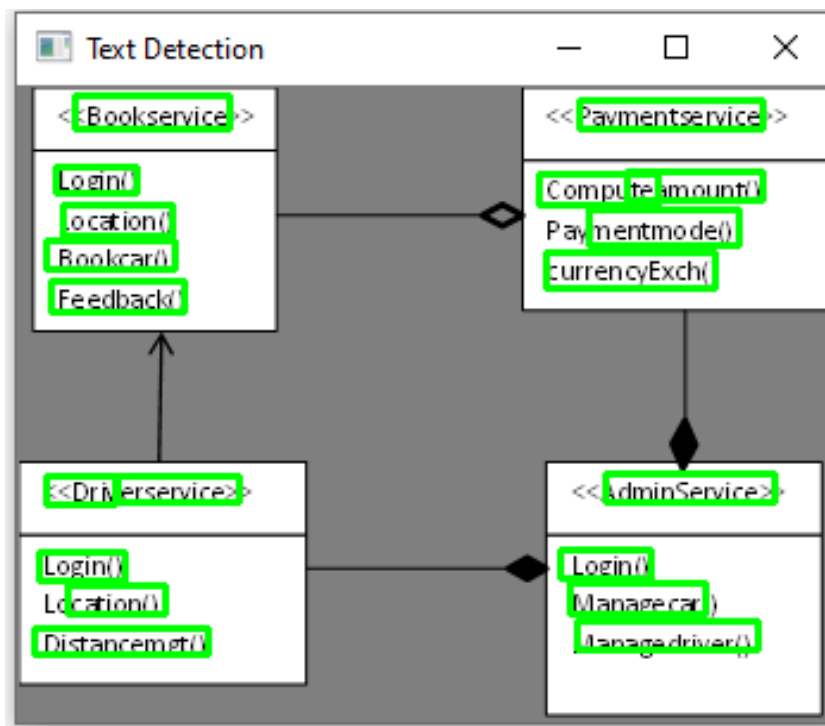


Figure 6.3: Highlighted operation names detected by EAST detector

The second step in text extraction from UML image is text recognition which takes the identified text by EAST detector and store the text into an array. This study used an

existing technique known as Tesseract OCR with an inbuilt deep learning technique referred to as Long Short Term Memory (LSTM) to extract text more efficiently and accurately. This study implemented Tesseract OCR using Python programming language which facilitated storage of extracted text into arrays. Python also enabled separation of service interface name and operation names with the use of conditional structure “IF” to discard names with symbols << or >> and only allow names that ends with brackets which in this case are service operation names in the lower rectangle of UML interface diagram. The sample result of Tesseract OCR is as shown in Figure 6.4.

```
RECOGNIZED TEXT
=====
Bookcar ()
Distancemeti)
Feedback) ,
Location ()
Login ()
```

Figure 6.4: Group of text recognized by Tesseract OCT

6.3.1.2 Service Operation Classification

The ultimate objective of text extraction and classification is to determine the type of service operation which includes simple, average and complex operation. This study used a machine learning technique known as Multi-class Support Vector Machine (SVM) to analyze a group of text and classify them accordingly. First of all, SVM was exposed to a wide variety of possible operation names for the purpose of training the model. In this context, a dataset of 1200 operation names with 3 categories of simple, average and

complex operation names was used to train the SVM model and a dataset of 100 operation names was used to test the model. This study sourced operation names from various repositories to create a datasets for training and testing purpose.

6.3.1.3 Arrow Head Detection

This study employed an existing deep learning technique known as ResNet50 CNN (He, Zhang, Ren & Sun, 2016) to detect types of arrows in UML service interface diagram and UML sequence diagram to determine the type of dependency or composition and type of message exchange among services respectively. Convolutional Neural Network (CNN) is a deep learning technique made up of neurons with learnable weights and biases. CNN is trained with datasets to analyze and classify patterns (Tripathi & Kumar, 2019). The objective of ResNet50 CNN in this study was to detect and classify UML interface arrows into atomic, lighter aggregation and strong composition and UML sequence diagram arrows into synchronous, asynchronous and reply arrows.

A dataset of 900 UML interface arrow images with the three defined categories was used to train the model and 100 arrow images were used to test the model. The same applied to UML sequence diagram arrows which the study used a dataset of 900 arrow images to train the model and 100 images to test the model. The study constructed the arrow images and sourced from different repositories due to non-availability of specific host for arrows for UML diagram online. Python programming language supported by Tensorflow and Keras frameworks were used to implement the model. A sample result after uploading UML interface image to the model is as shown in Figure 6.5.

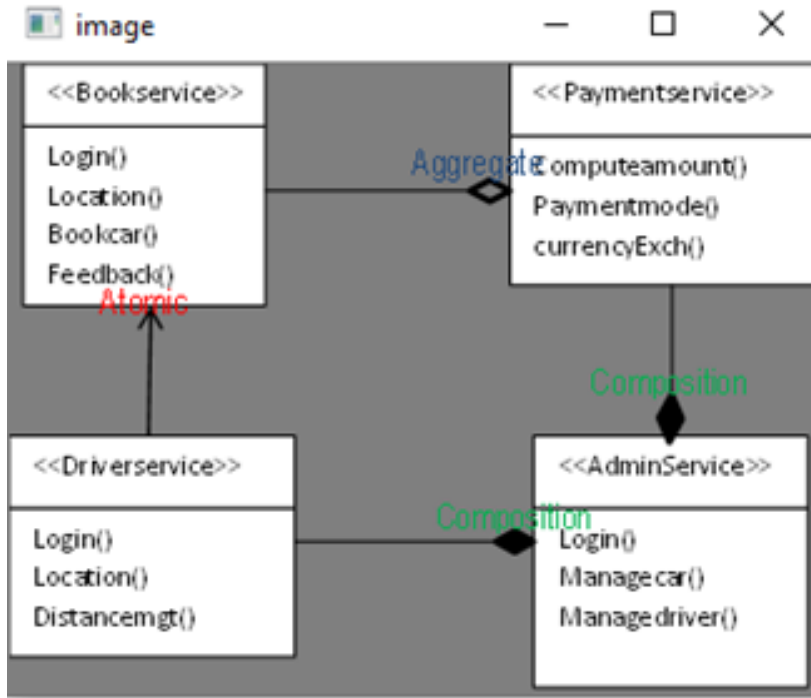


Figure 6.5: WOC arrow classification by ResNet50 CNN

The same principle was applied to UML sequence diagram where arrow heads were classified as synchronous, asynchronous and reply represented as *s*, *a* and *r* as shown in Figure 6.6.

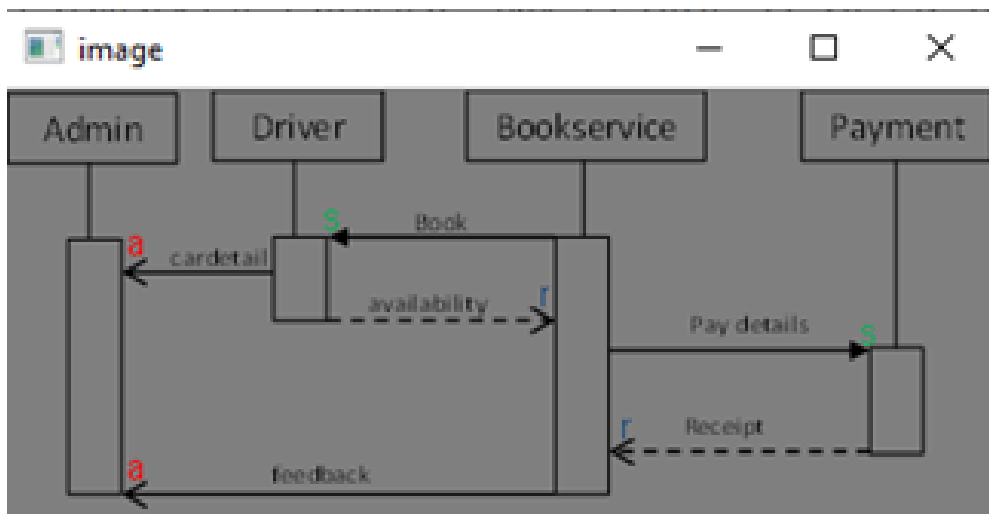


Figure 6.6: WMC arrow classification by ResNet50 CNN

Once the respective arrow types are captured, python is able to count the number of each arrow type using a counter.

6.3.2 Manual Entry/Display

Apart from reading UML image automatically, a session of computing SOA size may also start with the user capturing SOA size attributes through entering into the tool the size attributes values via a form to compute WOC, SDC and WMC as in Figure 6.7. Direct input through entering size attributes via a form provided an alternative to automated UML feature extraction. It is used in situations where UML diagram is not clear or when it is handwritten or an image cannot be loaded due to various reasons.

SOA SIZE METRICS AND EFFORT ESTIMATION (SMET)
Fill in the form with estimates that pertain to SOA Application

Web Service Computation Algorithm

SOA General Information

Web service application name: Number of Services:

Number of operations in a service (Service No: #1)

Service Name:

Service Operation type: Simple, Average, Complex, No. of parameters

Enter number of operation(s) in a service:

Number of fan-out dependency from a service

Service fan-out dependency type: Atomic, Lighter aggregation, Strong composition, Fan in

Enter number dependency type:

Number of Messages from a service

Message type: Synchronous, Asynchronous, Reply Message

Enter number of message type:

Action Buttons

Figure 6.7 SMET Manual entry interface of SOA size attributes

Furthermore, effort estimation factors such as service type factors and Effort multiplier factors can only be entered via the form elements and cannot be entered automatically. Service types are selected among provided options while effort multiplier efforts are entered through a slide element due to fuzzy logic application that requires continuous data input as shown in Figure 6.8.

The interface is divided into four main sections, each containing sliders for different effort factors:

- Product Factors:**
 - Database complexity: Slider with markers for Normal, High, and Very High.
 - Database Size: Slider with markers for Normal, High, and Very High.
 - User Interface Complexity: Slider with markers for Normal, High, and Very High.
 - Integration Complexity: Slider with markers for Normal, High, and Very High.
- Service Development Environment:**
 - Development Tool: Slider with markers for Low Automated, Normal, and High Automated.
 - Infrastructure Capabilities: Slider with markers for Very Low, Low, and Normal.
- Requirements factors:**
 - Requirement Elicitation: Slider with markers for Very Ambi, Ambig, Clear, and Very Clear.
 - Business Value: Slider with markers for Very Low, Low, and Normal.
 - Security Requirement: Slider with markers for Very Low, Low, and Normal.
- Personnel factors:**
 - Web service developers Experience: Slider with markers for Very Low, Low, and Normal.
 - Application Experience: Slider with markers for Very Low, Low, and Normal.
 - Team Cohesion: Slider with markers for Very Low, Low, and Normal.

Figure 6.8: SMET Effort factors input interface

6.3.3 Business Logic Layer

Business logic layer also known as the middle tier layer links the presentation layer with the data layer. In this tool, components in this layer receive data from automated UML feature extraction component or Manual Entry component, process the data then send result to data layer. This layer performs control of data functionalities and data manipulation through arithmetic, logical and conditional expressions (Cao, Wei & Qin, 2013).

In this study, user-side interactive scripting known as JavaScript and Common Gateway Interface known as php scripting language were used to build the business logic of the proposed prototype. JavaScript was employed to enable data manipulation, validation and controls at the user-side browser before data is send to the server. On the other hand, data send from the form is received by php, manipulated by php then send to a database management system.

The SOA-SMET prototype business logic performs computation on Service operations attributes to give WOC, Service dependency attributes to give SDC and Message movement attributes to give WMC. The tool then compute SOA size by summing WOC, SDC and WMC. Furthermore, SMET estimates effort for the entire SOA application after including Service Type Factors (STF) and Effort Multiplier Factors (EMF).

SMET Algorithms - Algorithm provide a step by step procedure in simple English showing the processes involved including input, computation flows and output. SMET algorithms provide detailed procedure of how metrics are computed and effort estimated. SMET algorithm includes SOA size metrics algorithm and Effort estimation algorithm.

Where size attributes input in this case is either through SMET UML recognizer or direct input via text elements.

i) SOA Size metrics Algorithm

START

Input SOA application name (Appname)

Input number of service contained in an application (N)

Count=0 ; Atotal =0; Stotal = 0;

Weighted Operation count (WOC)

Count number of simple operation(s) contained in a service (SO)

Count number of average operation(s) contained in a service (AO)

Count number of complex operation(s) contained in a service (CO)

Count number of parameters contained in a service (P)

Weighted Operation Count (WOC) = $2\sum SO + 3\sum AO + 4\sum CO + \sum P$

Write (WOC)

Weighted Service Dependency count (SDC)

Count number of atomic dependencies (A)

Count number of lighter aggregation dependencies (G)

Count number of strong composition dependencies (T)

Service Dependency Count (SDC) = $\sum A + 2\sum G + 3\sum T$

Write (SDC)

Weighted Message count (WMC)

Count number of synchronous messages from a service (S)

Countnumber of Asynchronous messages from a service (AS)

Count number of reply messages (R)

Weighted Message Count (WMC) = $3\sum S + 2\sum AS + \sum R$

Write (Sname+WMC)

Weighted Service Count (WSC)

Weighted service count (WSC) = WOC+SDC+WMC

Write (WSC)

ii) SOA Effort Estimation method algorithm

Service type factors

WHILE (count<n)

Count=count+1;

Select Service construction (SC) type[Available/migrated/New]

If service=available SC=0.6

else if service=migrated SC=0.8

else if service=New SC=1

Select Service communication protocol (SP) type[SOAP/REST]

If service=SOAP SP=1.2 else if service=REST SP=1

Service type factor (STF) = SC * SP

Total STF = Stotal*STF // Compute total service type factor for the entire application

End while

Effort multiplier Factors (EMF) computation

x=0; f=1;

X++

Select Service Effort multiplier Factors ratings

EMF = f*Factor

End while

Display EMF

SOA application size computation

SOA Application effort (SAE) = $3.2 * (WSC*53)^{1.05} * STF * EMF$

Display (Appname + SAE)

STOP

The algorithm show the processes involved in SMET which are structured in nature. The processes include arithmetic operations, logical operations and control operations. The algorithm shows how the tool calculates SOA size also referred to as WSC. The algorithm multiply service construction type factor with service architectural type factor to give service type factor (STF). STF for all services are multiplied then used to estimate effort together with the product of Effort multiplier Factors (EMF) to compute the estimated effort.

6.3.4 Data Layer

Data layer provides database management functionality responsible for modeling data to ensure optimization of data access, data consistency and data security (Cao, Wei & Qin, 2013). This layer is concerned with storage, indexing and relational modeling of a database which forms the prototype storage backbone. Data layer receives result from business logic layer after computation and provides data to the same for further analysis and manipulation.

SMET data layer model identified two main entities namely SOA application entity and Service entity. The SOA application entity stores details of a SOA application while service entity represents individual service details. The two entities are related in a one-to-many relationship with the application entity being the parent and service entity the child entity.

6.3.5 SMET Output

SMET output is a simple display of results of SOA size metrics and effort estimation method computation. Results are presented as a report detailing the name of the application, number of services, results of WOC, SDC, WMC and WSC. To compute

WOC, SDC and WMC metrics, the user clicks compute button for each metric and the result is displayed via the provided text box element. Output from computed WOC, SDC and WMC provide input to WSC which is computed automatically to give the size of a service.

WSC is multiplied by constants by Service type Factor (STF) and by Effort multiplier Factors (EMF) to estimate effort for developing SOA application. User selects service construction type (SC) and service communication protocol type (SP) options through the two drop-down menus provided in the effort estimation form interface to compute service type factor (STF). Lastly, users select EMF by moving the slider to a specified EMF factor rating.

6.4 Chapter Summary

This chapter provided details of SMET prototype tool implemented to capture SOA size attributes to compute size and estimate SOA effort. The tool provides an automatic interface for capturing UML diagram images through feature extraction and manual entry of SOA size attributes via a form. The prototype tool then computes SOA size and estimate effort for SOA based on values entered or captured automatically. The results of the tool size computation and estimation are stored in a database.

CHAPTER SEVEN

AN EMPIRICAL VALIDATION OF PROPOSED METRICS, EFFORT ESTIMATION METHOD AND AUTOMATED TOOL

7.1 Introduction

This chapter provides a detailed analysis of SOA size metrics, effort estimation method and automated tool empirical validation based on a laboratory experiment and a survey. The experiment involved 15 SOA based projects developed by university students. The survey was used to gather experts' opinion on the validity of the proposed SOA size metrics, proposed effort estimation method and the proposed automated prototype tool. The main objective of empirical validation was to test correlation between size attributes and SOA size, test correlation between SOA size and SOA development effort, test the accuracy of the SOA size metrics, SOA development effort estimation method and the implementation prototype tool.

7.2 Empirical Validation Strategy

The 15 SOA based projects were exposed to the proposed size metrics and function points analysis metrics to compare the results of the two approaches. From the proposed size metrics, development effort for each SOA based project was computed and compared with COCOMO effort estimates and the actual effort to determine the proposed effort estimation method accuracy. The implementation tool was also tested to determine the accuracy of deep learning techniques used in the tool. Lastly expert opinion via questionnaire were used to test the validity of each attribute, validity of size metrics and effort factors proposed in the study with regard to their influence to SOA effort estimation effort. Expert also validated the appropriateness of the implementation tool.

Experimentation Empirical validation preparation started by guiding 15 groups of students on how to develop software requirement specification (SRS) document and Software Design Document (SDD) for web service. The web service projects' SDD provided design artifacts such as UML diagram, Entity Relation Diagram (ERD) and Data Flow Diagram (DFD) which revealed key attributes and dimensions used to measure web service size. The groups were tasked to use the design artifacts to develop web service projects as they record the time each student worked on the project from requirement specification phase to integration phase.

The experiment involved subjecting web service projects design UML artifacts to the proposed SOA size metrics to measure web service size. The projects DFD and ERD were then exposed to Function point size metrics to enable comparison of the proposed metrics and function point analysis metric results for the purpose of testing the accuracy of the proposed metrics. Analysis of experiment results provided details on the relationship among variables proposed in the study. Furthermore, the accuracy of deep learning techniques used in the implementation automated prototype tool was tested based on testing datasets.

Secondly, the proposed effort estimation method was used to estimate effort spend to develop each SOA based project in persons per day. The estimated effort was then compared with COCOMO estimated effort and also compared with the actual effort spend to determine the proposed method's validity. Lastly, a survey was used to capture experts' opinions and analyses of the opinions were done to determine the validity of variables used in the study.

7.3 Context Definition

The laboratory experiment was intended to test the achievement of objective 1 and 2 of this study by measuring the accuracy of the proposed SOA size metrics and effort estimation method as compared to function point analysis and COCOMO II respectively. The experiment involved 15 web service based projects developed by 15 groups of 3rd year undergraduate Computer Science students from Meru University of Science and Technology. Apart from one group with 6 students, all other 14 groups had 5 students per group which is a total of 76 students who participated in the experiment.

7.4 Experimental Preparation

The subjects used in this study were 76 students in 3rd year BSc. Computer Science students who were all selected based on the fact that they were taking a course named CCS 3353 Research Method and Group project as part of their third year second semester course. Before this experiment the participants had knowledge in Systems Analysis and design, Fundamentals of computer programming, Object oriented programming, IT project management and Internet application programming which provided relevant background knowledge to this experiment.

The subjects being students, had no experience in the industry and had no knowledge of SOA based project development before this experiment. This challenge was addressed by training the subjects in SOA projects requirement specifications, systems design, SOA construction, testing and integration. More emphasis was on training subjects how to develop DFD, ERD, UML interface and UML sequence diagrams.

Experimental objects included 15 SOA based projects Systems design artifacts modeled using DFD, ERD, UML interface and UML sequence diagrams. DFD and ERD revealed SOA size attributes as input to Function point analysis while UML interface and UML sequence diagrams revealed attributes to enter into the proposed SOA size metrics. Other materials used in the experiment include laptops, lecture notes, web server, text editors and system design tools.

7.5 Experimental Planning

The goal of this experiment was to determine the relationship between size attributes and SOA size. The second goal was to test the relationship between SOA size and SOA development effort, the third goal was to determine the accuracy of the proposed SOA size metrics, the fourth goal was to determine the accuracy of SOA effort estimation method and lastly to determine the accuracy of deep learning techniques used in the automated implementation tool.

Variables in this experiment included SOA size attributes, SOA size, SOA development effort factors and SOA development effort. This study was set to check 2 sets of relationships. The first relationship was between SOA size attributes as independent variables and SOA size as the dependent variable. The second set of relationship was between SOA effort factors as independent and SOA effort as the dependent variable. Correlation among variables in this study was determined by testing relationship between size attributes and SOA size and between SOA size with effort for developing SOA. The accuracy of the proposed SOA size metrics and Effort development method were determined by comparing the results of the proposed SOA size metrics with Function point analysis metric results. The accuracy of the proposed SOA effort estimation method was

compared with COCOMO effort estimation result and also compared with actual effort used to develop the 15 projects.

Planning for the implementation automated tool involved preparing the datasets for machine learning and deep learning techniques used in the tool. Due to unavailability of datasets for UML service operation names and arrows depicting service dependency and data movement, this study embarked on collecting images from various sources. Sources that provided datasets for this study included random identification of operation names and arrows used in various UML images from various sources accessed online.

7.5.1 Hypotheses

This research study conceptualized the following 4 hypotheses statement to guide experimentation analysis.

Null hypothesis H_0 : There is no correlation between size attributes and SOA size

Alternative hypothesis H_1 : There is a correlation between SOA size attributes and SOA size.

Null hypothesis H_0 : There is no correlation between SOA size and SOA development effort.

Alternative hypothesis H_1 : There is a correlation between SOA size and SOA development effort.

Null hypothesis H_0 : The proposed SOA size metrics and effort estimation method are less accurate as compared to existing metrics and methods.

Alternative hypothesis H_1 : The proposed SOA size metrics and effort estimation method are more accurate as compared to existing metrics and methods.

Null hypothesis H_0 : The proposed SOA automated tool deep learning techniques are not accurate in extracting UML text and images.

Alternative hypothesis H_1 : The proposed SOA automated tool deep learning techniques are accurate in extracting UML text and images.

7.5.2 Threats to Validity

Validity threats in the experiment included construct validity, internal validity and external validity. Construct validity was ensured by subjecting metrics to theoretical validity tests. On the other hand, internal validity threat was as a result of inconsistencies and errors in students SOA based project design artifacts. To reduce this threat, the design artifacts had to undergo a thorough cleaning by removing and correcting inconsistencies and errors. Given that experiment involved students with no industrial experience, the proposed metrics and methods were subjected to expert opinion to reduce external validity. Lastly, to ensure all possible operation names and UML arrows are captures, images sources for training and testing purposes were collected exhaustively from various sources online to capture a wide range of images.

7.6 Experimental Operation

Experiment process started from 4th March 2019 where students were required to developed software requirement specification (SRS) document for SOA based project as they record the actual time taken to gather user requirements and develop the document. Each group submitted their SRS documents by 8th March 2019 which were verified and errors corrected before design phase. The corrected SRS documents were then re-submitted on 13th March 2019 and verified again before the next phase.

Upon verification of SRS documents, the groups were guided on how to develop a SOA based Software design document (SDD). The design phase ran from 18th March 2019 to 29th March 2019 with each group recording the time spent by each individual student to work on SDD. After correcting and verifying SDD, construction of SOA based projects progressed from 2nd April 2019 to 19th April 2019 after which integration and testing were done from 24th April 2019 to 3rd May 2019 with developers recording the time taken in each phase. Based on the final SOA based projects and documentations that were presented and submitted by each group, design artifacts were used as inputs to the proposed SOA size metrics and Function point analysis metrics.

UML interface diagram and UML sequence diagram revealed SOA size attributes as input to the proposed SOA size metrics. The attributes included number of operations per service measured by Weighted Operation Count (WOC) metrics, Number of dependencies measured by Service dependency Count (SDC) and Data movement measured by Weighted Message Count (WMC). On the other hand, attributes entered to Function point analysis were derived from DFD and ERD diagrams. The DFD and ERD attributes included Internal Logic File(ILF), External Interface File (EIF), External Input (EI), External Output (EO) and External Query (EQ).

Based on size attributes captured from design artifacts , SOA size was computed for each project and actual time used to develop the projects were recorded for further analysis. To compute SOA effort estimation, students recorded the type of services contained in each of project in relation to Service Architectural (SA) and Service Construction (SC) types. In addition, EMF details were captured from each group including product factors, service development environment factors, requirement factors and personnel factors. Due to

inability by students to capture their EMF effectively, these factors were captured through observation and experience with students as they interact with their projects. For instance, students could not capture their experience, team cohesion and other factors with sincerity and honesty. With all factors gathered from the 15 SOA based projects, effort for developing each the project was computed.

On the other hand, to develop deep learning models for the automated tool, datasets were required to train and test the existing techniques. In this regard, UML dependency or composition arrow types were collected from various sources, then they were grouped in two folders namely training folder and testing folder. In each of the folder, three sub-folders named atomic, aggregation and composite were created to store their respective arrow types with each folder holding 300 specific arrows. The same procedure was done for UML sequence diagram arrows grouped into asynchronous, synchronous and reply sub-folders with 300 arrow type in each folder. A GPU machine was used to train and test the ResNet50 CNN technique to come up with the model.

In the same principle, possible operation names were collected for training and testing the SVM technique for classifying operation names into simple, average and complex operations. However, EAST detector and Tesseract OCR (LSTM) did not require training but they were also tested for accuracy in relation to the UML interface operation names. Testing for EAST detector and Tesseract OCR involved 100 different operation names contained in UML diagrams.

7.7 Experiment Results

This section provides a detailed analysis of SOA size metrics validation, SOA development effort estimation method validation and SOA implementation prototype tool validation results. SOA size metrics validation involves comparison with Function Points Analysis results while effort estimation validation includes comparison with COCOMO results and the actual development effort.

7.7.1 SOA Size Metrics Validation Results

Data was collected from the 15 SOA based projects in two sets which include data for the proposed SOA metrics and data for Function point analysis to compare the two metrics results accuracy in relation to SOA. Table 7.1 represents results of the proposed SOA size metrics based on 15 SOA application projects. The Table shows the values of WOC, SDC, WMC and SOA size (WSC) in web service points. These values were captured from UML service interface and UML sequence diagrams designed by students for SOA projects.

Based on size metrics values, WOC contributes more to SOA size an average of 70% of SOA size while SDC and WMC contribute 30% of total SOA size. However, in projects that involve linking with services contained in legacy systems and services outside the organization will impact positively to SDC and WMC. Being student SOA project, dependency was only within the project itself rather than outside impacting negatively to the value of SDC and WMC.

Table 7.1: Data Analysis for the proposed SOA size metrics

Project ID	Project Name	WOC	SDC	WMC	SOA size (WSC)
1	Online Carpool System	31	7	6	44
2	Online doctors' appointment system	24	3	3	30
3	SACCO management system	32	4	6	42
4	Online Event & Catering system	25	7	6	38
5	Bus service online reservation system	27	5	5	37
6	Online furniture purchase system	27	4	7	38
7	Construction Material online purchase systems	29	4	7	40
8	Prime freelance systems	30	7	7	44
9	Real Estate online property management system	28	7	6	41
10	Tourism and accommodation online system	23	6	3	32
11	Apartment rental online system	27	6	6	39
12	Online Horticulture Sales Information system	30	8	3	41
13	CDF disbursement management system	25	5	3	33
14	Online Pharmaceutical management system	27	5	6	38
15	Online Event ticketing system	24	5	3	32

7.7.1.1 SOA Size Metrics descriptive analysis

According to Table 7.1, details captured from each project shows the projects were small in size and developed in a predicted environment. Being small projects developed by students, SOA size for all projects were less than 50 web service points with the biggest project having 44 web service points and the smallest project had 30 web service points.

Having used projects developed by students in this study, the time duration and scope of SOA based projects used in this experiment are closer to each other as indicated in the standard deviation. Summary of descriptive statistics for SOA size experiment results are as shown in Table 7.2.

Table 7.2 Descriptive analysis for the proposed SOA size metrics

Descriptive analysis	WOC	SDC	WMC	SOA size (WSC)
Maximum	32	8	7	44
Minimum	23	3	3	30
Mean	27.26667	5.6	5.2	37.93333
Standard deviation	2.737743	1.45733	1.641718	4.415341

Based on Table 7.2 descriptive statistics, average size of projects developed by students was 37.9333 and standard deviation was 4.415341. The maximum WOC was 32 while the minimum SDC and WMC were 3 and 3 respectively. The same pattern is revealed when computing maximum value, mean and standard deviation.

7.7.1.2 Function Point Size Metric Descriptive Analysis

Function Point Analysis was used as a benchmark to the proposed SOA size metrics. Function was selected based on its popularity with the industry and has been calibrated several times. Based on DFD and ERD design artifacts from the 15 projects FP analysis was computed project size as in Table 7.3.

Table 7.3: Data Analysis for Function point analysis

Proj. ID	Project Name	ILF	EIF	EI	EO	EQ	UFP
-----------------	---------------------	------------	------------	-----------	-----------	-----------	------------

1	Online Carpool System	12	0	12	3	3	30
2	Online doctors' appointment system	7	0	5	3	3	18
3	SACCO management system	12	0	6	4	3	25
4	Online Event & Catering system	9	0	4	3	4	20
5	Bus service online reservation system	12	0	6	3	3	24
6	Online furniture purchase system	12	0	5	4	3	24
7	Construction Material purchase systems	14	0	5	3	4	26
8	Prime freelance systems	11	0	7	4	3	25
9	Real Estate online property management system	12	0	4	3	3	22
10	Tourism and accommodation system	9	0	7	2	3	21
11	Apartment rental online system	11	0	7	3	6	27
12	Online Horticulture Sales Information system	16	0	6	7	6	35
13	CDF disbursement management system	10	0	7	3	3	23
14	Online Pharmaceutical management	13	0	5	3	4	25
15	Online Event ticketing system	9	0	7	2	3	21

However, whereas the proposed size metrics are focused on web service projects, Function Point analysis is for all types of application software. This study used FP analysis to compare with the proposed SOA size metrics accuracy. Table 7.3 shows details of the 15 SOA based projects attributes based on Function point measurement. Because the projects are relatively small in size, the unadjusted function point (UFP) returned is low. For all the SOA based projects used in the experiment no project was linked to an outside application for the purpose of external storage and thus they all returned 0 for External Interface file (EIF). Table 7.4 shows a descriptive statistics for the 15 web service project subjected to Function point analysis.

Table 7.4: Descriptive analysis for Function Point

Descriptive	ILF	EIF	EI	EO	EQ	UFP
-------------	-----	-----	----	----	----	-----

statistics						
Maximum	16	0	12	7	6	35
Minimum	7	0	4	2	3	18
Mean	11.2666667	0	6.2	3.333333	3.6	24.4
Standard deviation	2.25092574	0	1.934647	1.175139	1.055597	4.188419

As compared to the proposed SOA metrics, Function point returned lower mean points of 24.4 due to the fact that, Function point analysis focuses more on the structured design with no points for integration, dependencies and data movement. The accuracy of the proposed SOA metrics and Function point analysis can be determined when using their results to compute development effort. Lastly, there is a high relationship between the results returned by the proposed SOA metrics and function points analysis results with a coefficient of 0.654 which indicates there is a correlation between the value of size computed by the proposed metrics and Function Point analysis.

7.7.1.3 Correlation between Size Metrics and SOA Size

The three metrics that contribute to SOA size namely WOC, SDC and WMC were each tested for correlation with SOA size based on linear regression analysis. Based on Table 7.5 regression analysis, there is a high degree of correlation between WOC metric and SOA size as indicated by the value of R at 0.912, R² at 0.831 and a p-value of 0.000 as shown in Table 7.6.

Table 7.5: Correlation between WOC metrics and SOA size

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	0.912 ^a	0.831	0.818	1.884

a. Predictors: (Constant), WOC

Table 7.6: ANOVA analysis correlation between WOC metrics and SOA size

Model		Sum of Squares	Df	Mean Square	F	Sig.
1	Regression	226.794	1	226.794	63.900	0.000 ^b
	Residual	46.140	13	3.549		
	Total	272.933	14			

a. Dependent Variable: Size

b. Predictors: (Constant), WOC

Secondly, linear regression analysis was used to test the correlation between SDC and SOA size but the relationship was not significant as shown in Table 7.7 where the R value is 0.494 and R^2 value is 0.244. This lack of significance was as a result of using student SOA projects in the experiment with minimum aspect of service dependency among services as compared to real life services in organizations that are linked to services within and outside the organization.

Table 7.7: Correlation between SDC metrics and SOA size

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	0.494 ^a	0.244	0.186	3.983

a. Predictors: (Constant), SDC

Based on Table 7.8 linear regression analysis, there is a significant correlation between WMC and SOA size as indicated by R value at 0.731 and R^2 value at 0.534. In addition, WMC is a key indicator of SOA size as shown in Table 7.9 p-value of 0.002.

Table 7.8: Correlation between WMC metrics and SOA size

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	0.731 ^a	0.534	0.498	3.129

a. Predictors: (Constant), WMC

Table 7.9: ANOVA correlation analysis between WMC metrics and SOA size

Model		Sum of Squares	Df	Mean Square	F	Sig.
1	Regression	145.647	1	145.647	14.875	0.002 ^b
	Residual	127.286	13	9.791		
	Total	272.933	14			

a. Dependent Variable: Size

b. Predictors: (Constant), WMC

WOC and WMC metrics are significantly correlated to SOA size with coefficients of 0.912 and 0.731 respectively. The null hypothesis that there is no significant correlation between the size attributes and SOA size was rejected and alternative hypothesis that there is significant correlation between size attributes and SOA size was accepted. However, the relationship between SDC and SOA size was not as significant as WOC and WMC in relation to SOA size.

7.7.2 Effort Estimation Method Validation Results

Software development effort is a function of size multiplied by other effort factors. In this study size of the 15 SOA based projects were computed by the proposed SOA size metrics. The measured size and SOA development effort factors were then used to estimate effort for each project based on intermediate COCOMO for organic projects given that each of the SOA based projects are small, predictable and in a stable environment. SOA development effort factors used in this study include Service Type Factors (STF) and Effort multiplier Factors (EMF). Therefore,

$$\text{Effort (SOA application)} = \text{STF} * A * (\text{SOA service size})^B * \prod_{12}^n \text{EMF}$$

Where a = 3.2 , b = 1.05 for organic project (Small-scale projects)

Service type (STF) is factor determined by the type of service which includes Service Construction (SC) and Service Architectural style (SA). SC is classified as available, migrated and new service while SE is classified as REST and SOAP. STF was computed by multiplying the value of each of 15 web service projects SC and SA. In this regard, STF is first determined per service then the product of all services STF is computed for the entire SOA project. On the other hand, total EMF was computed by multiplying product factors, service development environment factors, requirements specification factors and personnel factors for each project. In this case, EMF is computed at the system level by considering the entire SOA application.

7.7.2.1 Proposed Effort Estimation Method Descriptive Analysis

The product of EMF per project was multiplied to SOA size and product of STF to compute effort estimation for each project as shown in Table 7.10. SOA sizes for the 15 projects which were computed in web service points had to be converted to KLOC to be used in the effort estimation based on COCOMO principle. Therefore, SOA size computed for the 15 projects were multiplied by 53 which is the constant used for java, C++, Perl and PHP to convert to LOC. LOC was then converted to KLOC by dividing by 1000.

Table 7.10: Effort Estimation Analysis based the proposed method

ID	Project Name	Size	STF	EMF	Estimated Effort (P/M)	Actual Effort (P/M)	MRE
1	Online Carpool System	44	1.2	1.294	12.813	9.54	-0.343
2	Online doctors' appointment system	30	1	1.294	7.142	6.32	-0.130
3	SACCO management system	42	1.2	1.125	10.611	8.84	-0.200
4	Online Event & Catering system	38	1	1.176	8.322	8.12	-0.025
5	Bus service online reservation system	37	1.2	1.294	10.682	8.31	-0.285
6	Online furniture purchase system	38	0.8	1.125	6.368	7.14	0.108
7	Construction Material online purchase systems	40	1	1.294	9.661	7.06	-0.368
8	Prime freelance systems	44	1	1.238	10.213	8.43	-0.212
9	Real Estate online property management	41	1	1.294	9.914	7.86	-0.261
10	Tourism and accommodation online system	32	1	1.294	7.643	6.21	-0.231
11	Apartment rental online	39	0.8	1.294	7.526	6.53	-0.152
12	Online Horticulture Sales Information system	41	1	1.294	9.914	8.84	-0.122
13	CDF disbursement management system	33	1	1.294	7.894	6.23	-0.267
14	Online Pharmaceutical management system	38	0.864	1.238	7.565	6.62	-0.143
15	Online Event ticketing	32	1	1.238	7.310	5.73	-0.276
Mean Magnitude of Relative Error							-0.194

According to Table 7.10, effort factors had tremendous effect on final development effort due to personnel factors where each project had a weight of 1.29 for SOA application experience given that students had no experience in SOA applications and its development

techniques. On database complexity, database size, hardware/ software capabilities, business risk/value and security requirements were awarded 1 for each project due to similarity in students' projects based on these factors. Integration complexity had a weight of 1.15 for all projects in the study due to connectivity to databases. Factors that experience variance among different projects in the experiment are interface complexity, development tool support and requirements elicitation.

The most common measures for effort estimation methods accuracy according to literature are Magnitude of Relative Error (MRE) and Mean Magnitude of relative error (MMRE).

$$MRE = \frac{y - \bar{y}}{y}$$

Where y is actual effort and \bar{y} is the estimated effort.

$$MMRE = \frac{1}{n} \times \sum_{i=1}^n MRE_i$$

Where n is the number of projects and MRE_i is for each project.

The accuracy of the proposed effort estimation method was -0.194 MMRE which is within the acceptable margin of -0.25 and +0.25. Therefore the accuracy of the proposed effort estimation method as revealed in the experiment shows that the proposed SOA effort estimation method is more accurate when dealing with SOA based applications.

7.7.2.2 COCOMO Effort Estimation Method Descriptive Analysis

The 15 web service projects were also subjected to COCOMO II effort estimation method as shown in Table 7.11 to enable comparison with the proposed effort estimation method.

Table 7.11: COCOMO Effort Estimation Method descriptive analysis

ID	Project Name	Size FP	AEF	Estimate d Effort (P/M)	Actual Effort (P/M)	MRE
1	Online Carpool System	30	1.27	7.01	9.54	0.2652
2	Online doctors' appointment system	18	1.24	4	6.32	0.3671
3	SACCO management system	25	1.20	5.47	8.84	0.3812
4	Online Event & Catering system	20	1.29	4.65	8.12	0.4273
5	Bus service reservation system	24	1.21	5.28	8.31	0.3646
6	Online furniture purchase system	24	1.20	5.24	7.14	0.2661
7	Construction Material online purchase systems	26	1.21	5.75	7.06	0.1856
8	Prime freelance systems	25	1.17	5.33	8.43	0.3677
9	Real Estate online property management	22	1.19	4.74	7.86	0.3969
10	Tourism and accommodation online system	21	1.20	4.55	6.21	0.2673
11	Apartment rental online	27	1.34	6.62	6.53	-0.0138
12	Online Horticulture Sales Information system	35	1.30	8.44	8.84	0.0452
13	CDF disbursement management system	23	1.33	5.55	6.23	0.1091
14	Online Pharmaceutical management system	25	1.24	5.65	6.62	0.1465
15	Online Event ticketing	21	1.26	4.78	5.73	0.1658

According to Table 7.11, COCOMO was used because it is the most documented and validated Software development effort estimation method. COCOMO takes software size and Effort Adjustment Factors (EAF) to compute effort the 15 web service projects.

Where Effort based on COCOMO II = $A * (\text{Size})^B * \prod_{17}^n EAF$

A = 3.2 and B = 1.05 and EAF is a product of 17 effort adjustment factors. Size is expressed in KLOC after converting size in Function point to KLOC by multiplying FP size with 53 and divide by 1000.

Based on Table 7.10 and Table 7.11 results, the proposed SOA effort estimation method is more accurate when compared to COCOMO given that COCOMO returned a higher MMRE of 0.2495 while the proposed SOA effort estimation method returned MMRE of -0.194.

7.7.2.3 Correlation between Size Metrics and SOA Size

The SOA size computed by the proposed metrics was correlated with the estimated effort for SOA as shown in Table 7.12 based on linear regression analysis.

Table 7.12: Correlation between SOA Size and SOA development effort

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics				
					R Change	F Change	df1	df2	Sig. F Change
1	0.716 ^a	0.513	0.475	1.2899	0.513	13.684	1	13	0.003

a. Predictors: (Constant), size Metrics

As shown in Tables 7.12 and 7.13, regression analysis results were significant with an R value of 0.716, R² value of 0.513, and a p-value of 0.003. This implies that there is a significant correlation between SOA size and SOA development effort. Therefore, the null hypothesis that there is no significant correlation between the SOA application size and SOA development effort was rejected and alternative hypothesis that there is significant correlation between SOA size and development effort was accepted.

Table 7.13: ANOVA correlation between SOA size and SOA development effort

Model		Sum of Squares	Df	Mean Square	F	Sig.
1	Regression	22.769	1	22.769	13.684	0.003 ^b
	Residual	21.631	13	1.664		
	Total	44.400	14			

a. Dependent Variable: effort

b. Predictors: (Constant), size Metrics

7.7.3 Automated Implementation Tool Accuracy Level

The third objective required this study to develop an automated tool to improve on efficiency and accuracy of capturing SOA size attributes. Having used deep learning techniques to extract features from UML diagram, testing the accuracy of techniques used in the tool before implementation was a requirement. First of all, the accuracy of extracting text from UML diagram by EAST detector and Tesseract OCR was tested based 100 different operation names. Secondly, operation names classification accuracy by SVM was tested with 100 operation names and ResNet50 CNN was also tested with 100 different service composition arrow heads and 100 arrow heads for message exchange arrows. Table 7.14 shows validation of each technique used to extract data from UML diagram when subjected to testing datasets.

Table 7.14: UML extraction techniques validation

Models	Training dataset	Testing dataset	Average accuracy
EAST detector	-	100	96.4%
Tesseract OCR	-	100	95.8%
Multi-class SVM	1200	100	93.1%
ResNet50 CNN (UML Interface)	900	100	97 %
ResNet50 CNN (UML sequence)	900	100	97.4%

Based on the analysis in Table 7.14, EAST detector, Tesseract OCR, Multi-class SVM and ResNet50 CNN returned encouraging result after validation. Issues that led to inability to capture names included faded text, unclear text shape and spacing issues that made some text to look like separated texts. On the other hand, ResNet50 CNN was affected by arrows whose shapes were not clear. Based on these results, the deep learning techniques were accurate as in alternative hypothesis,

Based on objective one, two and three of this study that required us to develop of a size metrics, development effort estimation method and develop an implementation tool, this study achieved the three objectives based on validation of the metrics, estimation method and implementation tool. However, due to the use of students as subjects and their projects as objects, there was need to seek opinions from experts which the study went further to collect expert opinions through a survey.

7.8 Expert Opinion Survey

A survey was used to gather expert opinions on the validity of the proposed SOA size metrics, proposed SOA effort estimation method and the proposed implementation tool. Expert opinion survey was also meant to complement laboratory experiment done by students. Preparation and planning was done appropriately before the survey was conducted to ensure validity and reliability of the instrument. The survey was conducted successfully and data from the survey was analyzed with a view to validate the proposed metrics and effort estimation method.

7.8.1 Survey Preparation and Planning

A pilot study was conducted based on five programmers to determine the adequacy of the survey instrument and gather feedback on how to improve the instrument. The five programmers were provided with annex documentation with details on the proposed metrics and method. They were given one week to read and understand the metrics and the method before answering the questionnaire. Feedback from the five programmers helped in improving the questionnaire content and context.

To identify the 20 programmers to participate in the study, 40 simple questionnaires were sent to programmers to indicate if they have ever worked with SOA applications including web services. Out of the fifty questionnaire 46 were returned with 27 responded positively having engaged in SOA applications before while 19 said they had never participated in developing SOA applications before. Random sampling was used to select 20 programmers out of 27 who had worked with SOA applications before.

The sampled programmers were taken through the proposed metrics, proposed effort estimation method and the implementation tool. They were required to ask questions where they did not understand the questionnaire. Upon satisfactory understanding of the proposed metrics, method and implementation tool, the sampled programmers were issued with questionnaires accompanied with annex documentation describing in detail the proposed metrics, estimation method and the implementation tool.

Threats to validity – Conclusive validity was under threat due to a relatively small sample of 20 experts who participated in the survey. However, the type of data required from experts did not require a bigger sample to validate the metrics. Furthermore, internal

validity was reduced by explaining and demonstrating to the experts the proposed tool to enable them to understand the proposed metrics, method and tool to instill objectivity when answering the questionnaire.

7.8.2 Demographic Summary of the Respondents

All the 20 questionnaires were returned successfully with no outlier data and all required data were entered. Before starting the survey, sampled experts were asked to provide their demographic details including their knowledge and experience in software development, SOA development and their highest level of academic qualifications. Based on academic qualification, 2 of the respondents had MSc. Degree in computing related field and the remaining 18 respondents had BSc. Degree in computing related field. Summary of respondents' experience in software development and experience in SOA application development is as shown in Table 7.15.

Table 7.15 Experts' experience in Software development

Experience	Below 1 year	Between 1 and 3 years	Above 3 years
Software development	4	12	4
SOA application development	8	7	5

According to Table 7.15, 16 respondents had over 1 year experience in software development and 12 respondents had over 1 year experience working with SOA applications. This confirmed that the experts had enough experience to assist in validating SOA size metrics and effort estimation method. However, 8 experts had less than 1 year experience in SOA development which disadvantaged the study but this is as a result of prevailing circumstance in the region.

7.9 Survey Results

7.9.1 Response on SOA Size Metric Validation

7.9.1.1 Experts' response on service internal structure influence on SOA size

Sampled experts responded on the relevance of internal structure attributes used in WOC metric to SOA size as shown in Table 7.16.

Table 7.16: Response on service internal structure attributes influence on SOA size

Attribute	Strongly Agree	Agree	Disagree	Strongly Disagree
Number of Operations/Methods	10	10	0	0
Operation's complexity	17	3	0	0
Number of parameters	13	7	0	0

Result in Table 7.16 shows that 10 respondents strongly agreed and 10 of the respondents agreed that the number of operation in a service has influence on service size. Secondly, 17 respondents strongly agreed that operation has influence on SOA size. Lastly, 13 respondents strongly agreed that number of parameters is a factor when measuring size. All respondents also agreed on the weights assigned to each service internal structure attribute when measuring the size of a service as shown in Table 7.17.

Table 7.17: Experts' response on weights assigned to service internal structure.

Attribute	Weights	Strongly Agree	Agree	Disagree	Strongly Disagree
Simple operation	2	4	16	0	0
Average operation	3	4	14	2	0
Complex operation	4	11	9	0	0
Parameter	1	9	11	0	0

According to Table 7.17, Experts agreed on the weights assigned to WOC size attributes in relation to classification of operations based on complexity and parameters. However, 2 experts disagreed on the weights assigned to average in relation to simple operation.

7.9.1.2 Experts' response on influence of service dependency on SOA size

All sampled experts reported that service dependency attribute contributes to SOA size with 75% of the respondents strongly agreeing and 25% agreeing with the attribute influence. They also concurred that the weights allocated to service dependency attributes are relevant to enable measurement of SOA size as shown in Table 7.18.

Table 7.18: Experts' response on weights assigned to service internal structure.

SDC Attribute	Weights	Strongly Agree	Agree	Disagree	Strongly Disagree
Atomic dependency	1	9	11	0	0
Lighter Aggregation	2	10	10	0	0
Strong composition	3	5	15	0	0

7.9.1.3 Experts' response on influence of data movement among services on SOA size

All sampled experts agreed that data movement among services attributes contributes to SOA size as indicated in Table 7.19.

Table 7.19: Experts' response on WMC attributes and weights

WMC Attribute	Weights	Strongly Agree	Agree	Disagree	Strongly Disagree
Synchronous	3	15	5	0	0
Asynchronous	2	8	11	1	0
Reply	1	7	13	0	0

According to Table 7.19 most experts confirmed that WMC attributes and their weights are relevant. However, one expert disagreed on asynchronous attribute but he did not give the reason for contrary opinion. The research maintained the weight of two on the basis of no reason given and only one expert had contrary opinion.

7.9.1.4 Experts’ response on WSC and SOA size effect on effort

SOA size which is basically the sum of WOC, SDC and WMC is a factor identified in this study which affects SOA development effort. Respondents also believed WSC is equivalent to SOA size where 15 strongly agreed. On the other hand, 10 respondents strongly agreed and 10 agreed that SOA size has influence on SOA development effort.

7.9.2 Response on Effort estimation factors

7.9.2.1 Experts’ response on influence of service type on SOA development effort

Another variable proposed in this study that influence SOA development effort is Service Type Factor (STF). Selected experts were asked to rate the influence of STF to SOA development effort to validate this research. Most experts sampled agreed that STF contribute immensely to SOA development effort and they also agreed on the allocated STF weights used to multiply with SOA size as shown in Table 7.20.

Table 7.20: Experts’ response on influence of service type to SOA development effort

Service type	Weight	Strongly agree	Agree	Disagree	Strongly disagree
Available service	0.6	4	16	0	0
Migrated service	0.8	10	10	0	0
New service	1	4	16	0	0

SOAP	1.3	8	12	0	0
REST	1	9	11	0	0

7.9.2.2 Response on influence of SOA Effort Multiplier Factors (EMF) to Effort

Expert confirmed that SOA EMF identified in this study are relevant when included in the method to estimate SOA effort. According to Table 7.21, all sampled experts agreed that EMF are relevant in estimating SOA effort. Factors which experts strongly agreed that they are relevant to effort estimation include service developers' experience (85%), SOA application experience (80%), Hardware/software capabilities (80%) and database complexity (75%). On the other hand, most experts did not agree strongly with the fact that requirement elicitation (30%) and business risk/value (25%) contribute immensely to SOA development effort. With no expert disagreeing with any of the proposed SOA effort factors, this research adopted all the factors. Response of EMF is based on expert experience, environment where they operate from and their personal intuition or feeling about a factor. For instance, an ambiguous user requirement may not be that ambiguous to another expert based on previous experiences on what ambiguity is in their respective context.

Table 7.21: Experts’ response on influence of EMF on SOA Development Effort

SOA effort factor	Description	Strongly agree	Agree	Disagree	Strongly disagree
	Database complexity	15	5	0	0
Product factors	Database size	10	10	0	0
	Integration complexity	11	9	0	0
Service development environment factors	Development tool support	13	7	0	0
	Hardware/Software capabilities	16	4	0	0
Requirements specification factors	Requirement elicitation	6	14	0	0
	Business risk/value	5	15	0	0
	Security requirements	14	6	0	0
Personnel factors	Service developers’ experience	17	3	0	0
	SOA Application experience	16	4	0	0
	Team cohesion	14	6	0	0

7.9.3 Response on the Validity and Appropriateness of the Implemented Tool

Respondents were asked to rate the appropriateness of the tool features, tool interface design and tool performance. Based on tool main features 70% of the respondents strongly agreed that the features included in the tool are appropriate and 30% agreed on the appropriateness of the implementation tool features. Secondly, 60% of the respondents strongly agreed that the tool interface design were appropriate, 30% strongly agreed and

10% disagreed. Lastly, 80% of the respondents strongly agreed on the tool performance and 20 percent agreed as shown in Table 7.22

Table 7.22: Experts’ response on appropriateness of implementation tool

Implementation tool appropriateness	Strongly Agree	Agree	Disagree	Strongly Disagree
Tool Features	14	6	0	0
Tool Interface design	12	8	0	0
Tool performance	16	4	0	0

According to Table 7.22 analysis, experts agreed on the appropriateness of the implementation to with regard to tool features, tool interface design and tool performance.

7.10 Chapter Summary

This research study involved 76 students who developed 15 SOA based applications used in this research laboratory experiment. All size metrics proposed in this study showed strong correlation with size. The experiment further revealed that the proposed metrics are more accurate when compared with function point analysis metrics when dealing with SOA based applications. The experiment also tested the accuracy of the proposed effort estimation method which was proved to more accurate as compared to the existing effort estimation methods. Due to the fact that the laboratory experiment in this study was done by students, there was need to subject the proposed metrics to the industry for further validation. In this regard, this research conducted a survey involving 20 sampled experts to validate the proposed metrics and effort estimation method. Based on the expert survey results, selected experts confirmed that the proposed size metrics and effort estimation method are relevant and valid for SOA based applications. The general observation made in the laboratory experiment and survey is that the proposed metrics which includes WOC,

SDC, WMC and WSC are valid metrics for measuring SOA size. Furthermore, factors used to estimate effort including SOA size, service type and SOA effort factors proposed by this study are relevant and valid when estimating SOA effort.

CHAPTER EIGHT

SUMMARY, CONCLUSION AND RECOMMENDATIONS

8.1 Summary

This research study examined existing literature of Software size metrics and Software effort estimation methods with a view of identifying existing gaps. The study then identified attributes that contribute to SOA size and defined SOA size metrics that rely on the identified size attributes to measure SOA size. The study further identified factors that contribute to SOA development effort and proposed a SOA effort estimation method based on the factors. The study implemented a tool to compute SOA metrics and effort estimation method. Experiment and survey methods were used to validate the metrics, effort estimation method and the tool.

The main objective of this research was to define a suite of size-based metrics and then use them to develop an effort estimation method for SOA systems. The summary of this research study is illustrated based on the three main objectives set by this study which were to define a suite of size metrics to measure the size attributes of SOA software systems, to develop an effort estimation method for SOA systems based on the size metrics and to implement a static analysis tool that computes the size and estimate effort of SOA software systems.

8.1.1 Defining Metrics for SOA Size

Existing software metrics and effort estimation methods analysis provided this research study with the opportunity to identify the gap in the industry that motivated this study to propose new SOA metrics and effort estimation method. Based on the identified gaps in

literature, this research identified SOA size attributes that are relevant in designing SOA size metrics. The research study proposed WOC, SDC, WMC and WSC size metrics in relation to the identified size attributes. The metrics were validated theoretically by Briand's theoretical properties to determine the proposed size metrics' structure validity. The proposed metrics were subjected to a laboratory experiment based on web service projects developed by 3rd year University students taking BSc. Computer Science. Expert opinions were gathered through a survey and data analyzed to validate the metrics empirically.

8.1.2 Developing an Effort Estimation Method for SOA Projects

This study identified key factors including SOA size, Service Type Factors (STF) and SOA Effort Multiplier Factors (EMF) which contribute to SOA development effort. The study proposed effort estimation method for SOA applications to fill in the gap that existed in the industry. The proposed method was exposed to empirical validation through a laboratory experiment to ascertain the proposed method accuracy as compared to an existing method. The proposed method was also subjected to experts to give their opinion on the relevance of the identified effort factors and the validity of the proposed SOA effort estimation method.

8.1.3 Automating the SOA Metrics and Effort Estimation Method

This study automated the proposed SOA size metrics and effort estimation method into a tool. The tool provides a platform to capture size attributes and compute WOC, SDC, WMC and WSC automatically then display the results. The implemented tool computes SOA size upon input of SOA size attributes through deep learning UML text and image extraction and also offers manual size attributes input via a form. The tool computes

development effort factors to estimate effort required to develop a SOA application. The tool was validated by testing the accuracy of underlying deep learning techniques and through a survey which captured experts' response on the tool appropriateness.

8.2 Conclusion

The research study successfully carried out a literature review on existing SOA size metrics and effort estimation methods which resulted to gaps identification that eventually prompted the design of the proposed metrics, proposed effort estimation method and the implementation automated tool.

8.2.1 Defined SOA Size Metrics

This research study contributed to the knowledge of software metrics and software project management by introducing the proposed metrics into the field of software engineering. The theory development in this study was as a result of the gap identified in literature review. Literature review developed in this study contributed immensely to the area of software size metrics which will enable future researchers to develop more software metrics. Theories on development of SOA size metrics in this study, revealed how SOA architectural difference from other software provided an opportunity to identify SOA attributes that contribute to size. These will contribute greatly to theory of software size in relation to attributes identification and development of size metrics. This study's contribution to practice include a new SOA size metrics that will be used to measure SOA application size to allow project managers and programmers to determine the scope of SOA based software application.

8.2.2 Effort Estimation Method for SOA

Literature review on Software effort estimation revealed the gaps and challenges of existing software effort estimation methods. This revelation is a contribution to researchers in software effort estimation. The design of effort estimation method documented in this study is a contribution to theory in Software project management. The design reveals the process of identifying software effort factors and how to compute effort which is a great contribution to theory. The new effort estimation method for SOA will contribute to practice by enabling project managers and developers to plan on the effort required, cost and time schedule for implementing a SOA project.

8.2.3 Automated Implementation Tool

Lastly, project managers and SOA developers will use the automated implementation tool compute SOA size and development effort more efficiently and accurately by simply uploading a UML diagram representing SOA attributes for the tool to compute SOA size and eventually estimate effort.

8.3 Recommendations for Future Work

This study recommends future work based on the three research objectives to enable future researchers to improve on software metrics, estimation methods and implementation tools for computing software size and effort.

This study recommends more software size metrics to be designed to capture emerging issues and attributes in the dynamic software engineering industry. The industry is dynamics with regard to changes in software attributes that contribute to software size, new software development methods, changes in SOA programming languages and changes in

programming platforms and architectures. All these changes require a review of metrics to capture new issues in the industry. There is also need to define metrics that measure size of multi-architectural applications such as an application that has a component of SOA and Component based elements.

Research in software effort estimation is still at infancy due to emerging issues that affect software development effort. First of all, there is no software effort estimation method that returns a Relative Margin of Error that is zero, there is always a margin of error which researchers should strive to improve. Secondly, there is need to capture more factors that contribute to software development effort due to emerging issues in software development. Lastly, with the introduction of new software architecture and software development methods, there is need to develop effort estimation methods that meet the needs of different architecture and software development methods.

This study only automated the aspect of entering SOA size attributes into the tool to compute SOA size. This research study recommends an improvement to the implementation tool by adding a feature that automatically detect effort estimation features such as service type factors, database complexity, integration factors, infrastructural factors and interface complexity. Automating capturing of effort factors will reduce subjectivity and errors and improve on the speed of estimating effort.

The research recommends further validation of the SOA size metrics and SOA development effort estimation method in laboratory experiments or case studies through the use of industry based projects including medium-scale and large-scale SOA projects.

REFERENCES

- Ahlawat, D. & Chawla, R. (2015). Software Development Effort Estimation using Fuzzy Logic framework: An implementation. *International Journal on Advanced Computer Theory & Engineering*.
- Ahmed, N.A. & Ahmed A.H. (2012). Enabling Complexity Use Case Function point on SOA. *2013 International conference on Computing, Electrical and Electronic Engineering*.
- Akkiraju, R. & Geel, V.H. (2010). Estimating the Cost of developing customization to packaged Applications Software Using SOA. *IEEE International Conference on web services*.
- Albrecht, A.J. & Gaffney, G.E. (1983). Software Function, Source lines of Codes, and Development Effort Prediction. *A Software Science Validation, IEEE Trans Software Engineering*.
- Amsden, J. (2010). Modeling with SoaML, The Service-Oriented Modeling Language. Part I. Service Identification, *IBM*.
- Anders, L. (2018). Function Point Analysis FPA on Team Planning Website Based on PHP and MYSQL. *Journal of Information Technology & Software Engineering*.
- Arnuphaptrairung, T. & Suksawasd, W. (2017). An Empirical Validation Application Effort Estimation Model. *Proceeding of International Multi-conference of Engineers and Computer Scientists, Hong Kong*.
- Asha, R.N., Kavana, M.D. & Parvathy, S.J. (2017). Object-Oriented Programming for Enhanced Programming Modularity. *International Journal for Science Research & Development*. Vol. 5. Issue 09.
- Assal, H. & Chiasson, S. (2018). Security in the Software Development Lifecycle, *Advanced Computing systems Association*.
- Azzeh, M. (2013). Software Cost Estimation on Use case points for Global software development. *5th International Conference on Computer Science and Information Technology, IEEE*.
- Basha, S. & Dhavachelvan, P. (2010). Analysis of Empirical Software Effort Estimation Model. *International Journal of Computer Science and Information Security*.
- Bawa, A. & Chawla, R. (2012). Experimental Analysis of Effort Estimation Using ANN. *International Journal of Emerging trends & Technology in Computer Science*.
- Belqasmi, F., Singh, J., Ban melhem, S. & Glitho, R. (2012). SOAP-based vs RESTful Web services for Multimedia Conferencing: A Case study.
- Benaroch, M. & Appari, A. (2010). Financial Pricing of Software Development Risk Factors. *IEEE*.

- Bhalerao, S. & Ingle, M. (2009). Incorporating Vital factors in agile estimation through algorithmic method. *International Journal of Computer Science and Computer applications*.
- Bianco, P., Lewis, G., Merson, R., & Simanta, S. (2011). Architecting Service Oriented Systems: *Software Engineering Institute, Carnegie Mellon University*.
- Bilgaiyan, S., Sagnika, S., Mishra, S. & Das, M. (2017). A systematic Review on Software Cost Estimation in Agile Software Development. *Journal of Engineering Science and Technology Review*.
- Boehm, B.W. (1981). *Software Engineering Economics, Prentice Hall*.
- Boehm, B.W., Clark, B., Horowitz, E. & Westland, C. (1995). Cost Models for future software life cycle processes: COCOMO 2.0. *J.C. Baltzer AG, Science Publishers*.
- Boehm, B.W. et al. (2000). *Software Cost Estimation with COCOMO: Prentice-Hall*.
- Borade, J.G., & Khalker, V.R. (2013). Software Effort and Cost Estimation Techniques, *International Journal of Advanced Research in Computer Science and Software Engineering*.
- Bormane, L., Grzibovsha, J., Bervisa, S. & Grabis, J. (2016). Impact of Requirements Elicitation Processes on success of Information System Development Projects. *Information Technology and Management Science*.
- Briand, L.C., Morasca, S., & Basili, C.H. (1991). Property – Based Software Engineering Measurement, *IEEE Transactions on Software Engineering*.
- Cao, J., Wei, J. & Qin, Y. (2013). Research and Application of the Four-Tier Architecture. *International Conference of Education Technology and Information Systems*.
- Cao, L. (2008). Estimating Agile Software Project Effort: An empirical study. *Association of Information Systems AIS Electronic Library(AISEL), Americas Conference on Information Systems*.
- Chidamber, S, R., Kemerer, C, F. (1998). Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis, *IEEE*.
- Chindove, H., Seymour L. F., & Van Der Merwe F. I. (2017). Service-oriented Architecture: Describing Benefits from an Organisational and Enterprise Architecture Perspective. vol. 3, no. *Iceis*. pp. 483–492.
- Coelho, E., Basu, A. (2012). Effort Estimation in Agile Software Development using Story Points. *International Journal of Applied Information Systems*.
- Cohn, M. (2006). *Agile Estimating and Planning: Pearson Education*.
- COSMIC. (2010). *Guideline for Sizing SOA Software, v1.0 : The Common Software Measurement International Consortium (COSMIC)*
- COSMIC. (2015). *Guideline for Sizing SOA Software, v4.0 : The Common Software Measurement International Consortium (COSMIC)*

- Deepa, R. & Lalwani, K.N. (2019). Image Classification and Text Extraction using machine learning. *Proc. 3rd International Conference of Electronic Communication Aerosp. Technology*.
- Domdouzis, K., Andrews, S. & Akhgar, B. (2016). Application of a New Service-Oriented Architecture (SOA) Paradigm on the Design of a Crisis Management Distributed System. *International Journal of Distributed Systems and Technologies*. Vol 7. Issue 2.
- Dong-Chul, P. (2016). Image Classification Using Naïve Bayes Classifier. *International Journal of Computer Science Electronics Engineering*. Vol 4.
- Dudhe, A. & Sherekar, S. (2014). Performance Analysis of SOAP and REST Mobile web service in cloud environment. *International Journal of Computer Applications*.
- Elhag, M, A., Mohamad, R. (2014). Metrics for Evaluating the Quality of Service Oriented Design. *IEEE*.
- Farrag, A. E. & Moawad, R. (2014). Phased Effort Estimation of legacy Systems Migration to SOA. *International Journal of Computer and Information Technology*.
- Farrag, A. E. , Moawad, R. & Imam, I. (2016). An Approach for Effort Estimation of SOA projects. *Journal of Software*.
- Frantisek, P. & Stal, M. (1998). An architectural view of distributed Objects and Components in CORBA, Java RMI, COM/DCOM, *Springer*.
- Gandomani, T., Wei, T., & Binhamid, K. (2014). Software Cost Estimation Using Expert Estimates, Wideband Delphi and Planning Poker Technique. *International Journal of Software Engineering and its applications*.
- Gupta, D. (2013). Service point Estimation Model for SOA Based Projects, *Service Technology Magazine*.
- Harizi, M. (2012). The Role of Class Diagram in Estimating Software Size. *International Journal of Computer Applications*. Volume 44.
- Hassan, S. & Afsar, S. (2012). Software Engineering: Factors Affecting Requirement Prioritization. *Global Journal of Computer Science and Technology Software and Data Engineering*.
- Hassim, W.B. H.W. (2017). A Review on effective Requirement Elicitation Techniques. *International Journal of Advances in Computer Science and Technology*.
- Hirzalla, M., Cleland-Huang, J., & Arsanjani, A. (2009). A metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures. *ACM*.
- Hussain, S., Muhammad, & S., Ahmed, S. (2010). Mapping of SOA and RUP: DOA as a case study. *Journal of Computing*.
- Inwala, A.M., Kharade, K., Chaugule, R. & Magikar, A. (2016). Dimensional arrow detection from CAD drawings. *Indian Journal of Science and Technology*. Vol. 9.
- Karasneh, B. & Chaudron, M.R.V. (2013). Extracting UML Models from images. *5th International conference on Computer Science and Information Technology (CSIT)*.

- Karner, G. (1993). Metrics for Objectivity. University of Linkoping. Sweden.
- Kaur, I., Narula, S.G., Wason, R. & Jain, V. (2018). Neuro Fuzzy – COCOMO II Model for Software Cost Estimation. *Institute of Computer Applications and Management*.
- Khatibi, V., & Jawawi, D.N. (2010). Software Estimation Methods: A review. *Journal of Emerging Trends in Computing and Information Sciences*.
- Kitchenham, B., Pfleeger, S.L. & Fenton, N.E. (1995). Towards a Framework for Software Measurement Validation, *IEEE Transaction on Software Engineering*.
- Kirmani, M. & Wahid, A. (2015). Use Case Point and e-Use Case Point method of Software Effort Estimation: A critical performance comparison. *International Journal of Computer Application*.
- Kubasell, M. (2006). Cashing Optimization in Service Oriented Architecture: *Masarykiana University*
- Kumari, S., & Pushkar, S. (2013). Performance Analysis of software cost Estimation methods: A Review. *International Journal of Advanced Research in Computer Science and Software Engineering*.
- Kothari, C.R. (2004). Research Methodology: Methods and Techniques, 2nd Edition: *New Age International (P) Ltd Publishers*.
- Kuan, S.W. (2017). Factors on Software Effort Estimation. *International Journal of Software Engineering & Application*.
- Li, Z. & Keung, J. (2010). Software Cost Estimation Framework for Service-Oriented Architecture Systems Using Divide-and-Conquer Approach. *5th IEEE International Symposium on Service Oriented System Engineering*.
- Li, Z. & O'Brien, L. (2010). Towards Effort Estimation for Web Service Compositions Using Classification Metrics, *IEEE*
- Litoriya, R. & Kothari, A. (2013). An Efficient Approach for Agile Web Based Project Estimation: AgileMOW. *International Journal of Computer Science and Computer applications*.
- Mahmoud, K., Ilahi, M., Ahmed & B., Ahmed, S. (2012). “Empirical Analysis of Function points in service oriented in Service oriented architecture (SOA) Applications”, : *Industrial Engineering letters*.
- Marsyahariani, N., Daud, N. & Kadir, W.M.N. (2014). Static and Dynamic Classification for SOA Structural Attributes Metrics. *8th Malaysian Software Engineering Conference*.
- Martino, S., Gravino, C. (2009). Estimating Web Application using COSMIC-FFP Method. *International Journal of Computer and Applications*.
- McCabe, T, J. (1976). A Complexity Measure. *IEEE*.
- Micheal, A. & Boniface, A. (2014). Inadequate Requirements Engineering Process: A key factor for poor Software development in Developing Nations: A case study. *International Journal of Computer and Information Engineering*.

- Mishra, S. & Kumar, C. (2014). Estimating Development Size and Effort of Business Process SOA Applications. *2nd International Conference on Systems and Informatics*.
- Mumbaikar, S. & Padiya, P. (2013). Web services Based on SOAP and REST principles. *International Journal of Scientific and Research Publications*.
- Muketha, G.M., Ghani, A. & Selamat, M. (2010). A Survey of Business Process Complexity Metrics, *Information Technology Journal*.
- O'Brien, L. (2009). A framework for scope, cost and effort estimation for Service oriented Architecture (SOA) projects, Australian Software Engineering Conference. *IEEE*.
- Park, H. & Back, S. (2008). An empirical validation of Neural Network Model for Software Effort Estimation, Expert Systems with Applications. *Elsevier*.
- Patra, P. & Rajnish, K. (2018). A Fuzzy based Parametric Approach for Software Effort Estimation. *International Journal of Modern Education and Computer Science*.
- Prokopova, Z. & Silhavy, P. (2015). Algorithmic Method for Effort Estimation. *Programming and Computer Software Society*.
- Rijwani, P. & Jain, S. (2016). Enhanced Software Effort Estimation Using Multilayered Feed Forward Artificial Neural Network Technique. *Science Direct, Elsevier*.
- Seth, A., Agarwal, H., & Singla, A. (2010). Testing and Evaluation of Service Oriented Systems. *International Journal of Engineering Research and Application*.
- Seth, A., Singla, A.R. & Aggarwal, H. (2012). Service-Oriented Architecture Adoption Trends: A Critical Survey. *Springer-Verlay Berlin Heidelberg*. Pp 164-175.
- Saunders, M., Lewis, P. & Thornhill, A. (2012). Research Methods for Business Students, *6th Edition, Pearson Education Limited*.
- Shamsoddid-Mutlah, E. (2012). A survey of Service Oriented Architecture systems testing. *International Journal of Software Engineering and applications*.
- Sharbanoo, M., Ali M., Merran M. (2012). An Approach for Agile SOA development using Agile Approach. *International Journal of Computer Science*.
- Sharma, N., Bajpai, A., & Litoriya, R. (2012) Software Effort Estimation. *International Journal of Computer Science and Applications*.
- Sharma, T.N., Bhardwaj, A., & Kherwa, G.R. (2012). Analysis of various Models of Software Cost Estimation. *International Journal of Engineering Research and Application*.
- Sharma, V., Shewandayn, B. & Bhukya N. (2017). Measuring Usability of Web services using Coupling Metrics. *International Journal of Advanced Research in Basic Engineering Science and Technology*.
- Shivakumar, N., Balaji, N. & Ananthakumar, K. (2016). A Neuro Fuzzy Algorithm to Compute Software Effort Estimation. *Global Journal of Computer Science and Technology: Software Engineering & Data Engineering*.

- Siddiqui Z. A. & Tyagi K. (2016). A critical review on effort estimation techniques for service-oriented-architecture-based applications. *International Journal of Computer Application*. vol. 7074.
- Srinivasan, K. & Devi, T. (2014). Software Metrics Validation Methodologies in Software Engineering. *International Journal of Software Engineering & Applications*.
- Stellman, A., & Greene, J. (2005). Applied Software Project management: *O'Reilly Media*.
- Svanidzaite, S. (2014). An Approach to Service Oriented Architecture development methodology: SOUP Comparison with RUP and XP, *Computational Science and Techniques*.
- Sultana, F., Sufian, A. & Dutta, P. (2018). Advancement in image classification using convolutional Neural Network. *Proc. – 2018 4th IEEE International conference Computer Intelligence and communication Networks*.
- Swanson, R.A. & Chermack, T.J. (2013). Theory Building in Applied Disciplines. Berrett-Koehler publisher. San Francisco.
- Tarawneh, H. (2011). A suggested Theoretical Framework for Software Project Success. *Journal of Software Engineering and Applications*.
- Tensey, B. & Stoulia, E. (2007). Valuating Software Service Development: Integrating COCOMO II and Real Options Theory. *The First International Workshop on the Economics of Software and Computation*.
- Thai, L.H., Hai, T.S. & Thuy, N.T. (2012). Image Classification Using Support Vector Machine and Artificial Neural Network. *International Journal of Information Technology and Computer Science*. Vol 2.
- Thamarai, I. & Murugavali, S. (2016). Analogy Based Software Effort Estimation Based Differential Evolution and Hybrid Fuzzy logic and Firefly Algorithm. *Asian Journal of Information Technology*.
- The Standish group, (2013)., Chaos Manifesto: Think big, Act small, *The Standish Group International*.
- Tripathi, S. & Kumar, R. (2019). Image classification using small Convolutional Neural Network. *9th International Conference on Cloud computing*.
- Verlaine, B., Jureta, J.I. & Faulkner, S. (2014). A Requirements – Based Model for Effort Estimation in Service – Oriented Systems. *Springer*
- Wohlin, C., Runeson, P., Host, M., Ohlsson, M., Regnell, B. & Wesslen, A. (2000). Experimentation in Software Engineering. *The Kluwer International series in Software Engineering*.
- Zhang, Q., Li, X. (2009). Complexity metrics for Service-Oriented Systems, *Second International Symposium on Knowledge Acquisition and Modeling*.
- Zhou, X. et al. (2017). EAST:An Efficient and Accurate Scene Text Detector. *IEEE*.
- Ziauddin, Kamal, S., Khan, S. & Nasir, A.J. (2013). A Fuzzy Logic Cost Estimation Model. *International Journal of Software Engineering and Its Applications*.

APPENDICES

APPENDIX 1: QUESTIONNAIRE TO EXPERTS SOA SIZE METRICS AND EFFORT ESTIMATION METHOD VALIDATION BY EXPERTS

This questionnaire is part of a study that aims to develop a SOA size metrics, an effort estimation method and an automated implementation tool for the metric and effort estimation method. The main objective of this questionnaire is to determine if the proposed SOA size metrics and SOA effort estimation method are valid. You have been chosen to participate in this study due to your knowledge and experience in software development especially SOA applications such as web service applications.

Please note that any identifying information you provide is purely for the purpose this study, it will remain confidential and will never be shared with a third party. If you have any query, contact me via email address sammunialo@gmail.com or telephone 0721452484. Kindly familiarize yourself with the attached SOA size metrics and estimation method (Annex 1, 2, 3 and 4) before responding to the questionnaire.

PART A - PERSONAL DETAILS

- i) What is the Name of your organization
- ii) Kindly indicate your highest level of academic qualifications.....
.....
- iii) How long have you worked as a programmer/developer/system designer?
 Less than 1 Year 1-3 Years Over Three Years
- iv) How long have you worked with API/web service as a developer/designer?
 Less than 1 Year 1-3 Years Over Three Years

PART B- SOA SIZE METRICS

In this section, we want to capture your approval rating of the correctness of proposed SOA metrics underlying theory, structure and assumptions for achieving its intended purpose. This study considered SOA internal structure, data movement, dependency among services and number of services as key parameters for defining SOA size metrics. (See Annex 1)

- i) SOA internal structure – This study proposed Weighted Operation Count (WOC) metric to measure service size based on service internal structure. WOC takes into account the number of operations/methods, operation’s complexity and number of parameters as key attributes influencing the size of a web service. The metric assigns a weight to an operation/method in a service based on its complexity and counts number of parameters then sums the weights of all operations in a service.

- a) **To what extent do you agree on the influence of the following service internal structure attributes to service size and development effort?**

SN	Component	Strongly agree	Agree	Disagree	Strongly disagree
1	Number of operations/methods				
2	Operations complexity				
3	Number of parameters in an operation				

b) If you don't believe in some or all of these attributes, recommend appropriate alternative attributes that contribute to service size based on service internal structure.

.....

.....

.....

.....

c) To what extend do you agree on the following categorization and weights of operations/methods based on their complexity?

SN	Attribute	Weight	Strongly agree	Agree	Disagree	Strongly disagree
1	Simple operation	2				
2	Average operation	3				
3	Complex operation	4				

d) If you don't believe in some or all of the above categorization and weights, recommend appropriate alternative categorization and weights that contributes to operations complexity.

.....

.....

.....

.....

.....

ii) Dependency among services - Service dependency also known as coupling which is the degree of interaction and extent of dependency between services. This study identified Service dependency attribute as an indicator of web service size measurement. The study defined Service Dependency Count (SDC) Metrics to count the number of dependencies between services as represented in UML interface diagram. SDC takes into consideration different types of dependencies as attributes contributing to SOA size. Dependency is classified into three namely atomic, lighter aggregation and strong composition based on the depth of dependency also known as service composition (See annex 2).

a) **To what extend do you agree that dependency between services influence web service size?**

Strongly agree Agree Disagree strongly disagree

b) **To what extend do you agree on the following categorization and weights of web service fan-out dependency?**

SN	Attribute	Weights	Strongly agree	Agree	Disagree	Strongly disagree
1	Atomic dependency	1				
2	Lighter Aggregation	2				
3	Strong composition	3				

c) **If you don't believe in some or all of the above categorization and weights, recommend appropriate alternative categorization and weights that contributes to size based on web service dependency.**

.....

.....

.....

iii) Data movement among services – This attribute is measured by Weighted Message Count (WMC) which takes into account the amount of data groups moving between services, databases and other applications. Weighted Message Count categorizes messages as synchronous, asynchronous and reply messages. In this regard, data movement specification is linked to the design of information model which is represented by UML sequence diagram (See annex 3).

a) **To what extent do you agree that the amount of data movement among services influence web service size?**

Strongly agree Agree Disagree strongly disagree

b) **To what extent do you agree on the following categorization and weights of data movement types among services?**

SN	Attribute	Description	Weights	Strongly agree	Agree	Disagree	Strongly disagree
1	Synchronous	It requires coordination of events between the sender and receiver to enable message movements in unison.	3				
2	Asynchronous	It does not return a value and no coordination is required with the receiver service to facilitate data movement	2				
3	Reply message	Reply messages are based on conditional tests that will provide error messages or acceptance messages.	1				

c) **If you don't believe in some or all of the above categorization, recommend appropriate alternative categorization that contributes to size based on web service dependency.**

.....

.....

.....

.....

iv) SOA size: Weighted Service Count (WSC) metric is used to sum the output derived from WOC, SDC and WMC then sum for all services to get the overall SOA size.

a) To what extent do you agree that the number of services influence web service application size?

Strongly agree Agree Disagree Strongly disagree

PART B- EFFORT ESTIMATION METHOD VALIDATION

In this section we want to capture your approval rating of effort estimation factors proposed in this study. Effort is determined by considering how many programmers are needed to accomplish a task and for how long measured in person-day or person-month. This study estimates SOA development effort for all development phases including requirement specification, software architecture phase, software construction phase and testing phase. Effort is determined by SOA size and other key factors/ cost drivers which are proportional to the amount of effort employed and whose values either increase or decrease effort.

i) SOA size – This study identified SOA size as the main attribute that determines SOA development effort.

a) To what extent do you agree that the size of SOA influence SOA development effort?

Strongly agree Agree Disagree strongly disagree

ii) Service type – It is defined by how the service was developed or realized. This study classified Service type into service construction type and service communication protocol. Service construction types include discovered, migrated and new service while service communication protocols include SOAP and REST.

a) To what extent do you agree that service type influence SOA development effort?

Strongly agree Agree Disagree strongly disagree

b) To what extent do you agree on the following categorization and weights of service construction types which influence SOA development effort?

Service construction type	Description	Weight	Strongly agree	Agree	Disagree	Strongly disagree
Available service	Existing service to be discovered.	0.6				
Migrated service	Service created from an existing legacy system	0.8				
New service	Service built from scratch	1				

c) If you don't believe in some or all of the above categorization and weights, recommend appropriate alternative categorization that contributes to operations complexity.

.....

.....

.....

.....

d) To what extent do you agree on the following categorization service communication types and their weights which influence SOA development effort?

Service communication protocol type	Description	Weight/rate	Strongly agree	Agree	Disagree	Strongly disagree
SOAP web service	SOAP is a communication protocol that sends data from one service to another based on a standardized set of message patterns.	1.2				
REST web service	Uses a consistent interface to access identified recourses based on data access method	1				

e) **If you don't believe in some or all of the above categorization and weights, recommend appropriate alternative categorization that influence to SOA effort.**

.....

.....

.....

.....

iii) Other Service development effort factors – Other service development effort factors include SOA product factors, Service development environment factors, Requirement specification factors and Personnel factors.

a) **To what extent do you agree on the following categorization factors which influence SOA development effort?**

SOA effort factor	Description	Strongly agree	Agree	Disagree	Strongly disagree
Product factors	Database complexity				
	Database size				
	Integration complexity				
Service development environment factors	Development tool support				
	Hardware/Software capabilities				
Requirements specification factors	Requirement elicitation				
	Business risk/value				
	Security requirements				
Personnel factors	Service developer's experience				
	SOA Application experience				
	Team cohesion				

b) If you don't believe in some or all of the above categorization, recommend appropriate alternative categorization that influence to SOA effort.

.....

.....

.....

PART C: APPROPRIATENESS OF THE IMPLEMENTATION TOOL

Based on the implementation tool demonstration rate the tool appropriateness with regards to tool features, tool interface design and tool performance.

- a) **To what extent do you agree on the appropriateness of the following implantation tool aspects?**

Implementation tool appropriateness	Strongly Agree	Agree	Disagree	Strongly Disagree
Tool Features				
Tool Interface design				
Tool performance				

- b) **If you disagree, indicate any suggestion on modifications or improvement to the implementation tool.**

.....

.....

.....

Thank you for your participation.

Your contribution to this research study will go a long way to improve SOA size metrics and effort estimation method.

APPENDIX 2: QUESTIONNAIRE TO STUDENTS ON WEB SERVICE PROJECTS EXPERIMENTATION

This questionnaire is part of the study that aims to develop a SOA size metrics and an effort estimation method based on the size metrics. The main objective of this questionnaire is to determine if the proposed metrics and method are valid as compared to existing metrics and effort estimation methods with regard to SOA. You have been chosen to participate in this study to provide details about your project in relation to size and effort factors. Please note that any identifying information you provide is purely for the purpose this study, it will remain confidential and will never be shared with third party.

PART A – WEB SERVICE APPLICATION DETAILS

- i) Project ID: _____
- ii) Project Name: _____
- iii) Start Date: _____ End Date: _____
- iv) Number of developers involved in the system _____
- v) Indicate the total hours spend in the following phases of project development.

Phase	Number of hours
Requirements and Analysis	
Design	
Development	
Testing	
Integration	
Total hours	

PART B- SOA SIZE DETAILS

In this section, we want to capture details of your SOA application project to be input to the proposed SOA size metrics. This study considered SOA internal structure, data movement and dependency among services as key inputs into the proposed metrics. (For more details on the attributes see annex 1, 2, 3 and annex 4)

- i) **List services included in the application as per the UML interface diagram.**

ID	Service name	ID	Service name
1		4	
2		5	
3		6	

- ii) **Indicate the number of size attributes in the table below for each service listed above.**

NUMBER OF WEB SERVICE SIZE ATTRIBUTES							
Application Name: _____							
Service Internal Structure per service (See annex 1 for more details)							
No	Attribute	Services ID					
		1	2	3	4	5	6
1	Number of simple operations						
2	Number of average operations						
3	Number of Complex operations						
4	Number of parameters						
Service dependency in the application (See annex 2 for more details)							
1	Number of atomic dependency						
2	Number of lighter aggregation						
3	Number of Strong composition						
Data movement among services in the application (See annex 2 for more details)							
1	Number of Synchronous messages						
2	Number of Asynchronous messages						
3	Number of reply messages						

PART C- SOA DEVELOPMENT EFFORT FACTORS

In this section we want to capture factors that influence SOA development effort. These factors include size, service type, product factor, service development environment factors, requirement specification factor and personnel factors (See annex 4 for more details and clarifications).

- i) For each service in your project, indicate (tick) the type of service in relation to Service Construction (SC) type and Service Architectural type.

SERVICE TYPE FACTOR							
Application Name: _____							
Service Construction type per service (See annex 3 for more details)							
		Services ID					
No	Attribute	1	2	3	4	5	6
1	Discovered service						
2	Migrated Service						
3	New service						
Service architectural type per service (See annex 3 for more details)							
4	SOAP						
5	REST						
Add more details if services are more than 6:							

- ii) **Rate the web service application based on the following effort factors based on the SOA application development (See annex 4)**

SOA effort factor	Description	Normal	High	Very High
Product factors	Database complexity			
	Database size			
	Integration complexity			

SOA effort factor	Description	Very low	Low	Normal	High	Very High
Service development environment factors	Development tool support					
	Hardware/Software capabilities					
Requirements specification factors	Requirement elicitation					
	Business risk/value					
	Security requirements					
Personnel factors	Service developer's experience					
	SOA Application experience					
	Team cohesion					

Thank you participating in this study

**APPENDIX 3: PROPOSAL APPROVAL LETTER FROM BOARD OF
POSTGRADUATE STUDIES OF MMUST**



**MASINDE MULIRO UNIVERSITY OF SCIENCE AND TECHNOLOGY
(MMUST)**

Tel: 0702597360/61
: 0733120020/22
E-mail: deansgs@mmust.ac.ke
Website: www.mmust.ac.ke

P.O Box 190
50100 Kakamega
KENYA

Directorate of Postgraduate Studies

Ref: MMU/COR: 309079

31st January, 2018

Samson Wanjala Muniolo
STPLIF003/2013
P.O. Box 190-50100
KAKAMEGA

Dear Mr. Muniolo,

RE: APPROVAL OF PROPOSAL

I am pleased to inform you that the Directorate of Postgraduate Studies has considered and approved your Ph.D proposal entitled: *"A Size Metric-Based Effort Estimation Method For Service-Oriented Architecture Software Systems"* and appointed the following as supervisors:

1. Prof. Geoffrey Muchiri - Department of Information Technology- Murang'a University
2. Dr. Kelvin Kabeti Omicau - Department of Computer Science-MMUST

You are required to submit through your supervisor(s) progress reports every three months to the Director of Postgraduate Studies. Such reports should be copied to the following: Chairman, School of Computing and Informatics Committee and Chairman, Department of Information Technology. Kindly adhere to research ethics consideration in conducting research.


It is the policy and regulations of the University that you observe a deadline of three years from the date of registration to complete your Ph.D thesis. Do not hesitate to consult this office in case of any problem encountered in the course of your work.


We wish you the best in your research and hope the study will make original contribution to knowledge.

Yours Sincerely,

Dr. Benedict Alafa
FOR DIRECTOR DIRECTORATE OF POSTGRADUATE STUDIES


APPENDIX 4: RESEARCH PERMIT FROM NACOSTI


REPUBLIC OF KENYA


NATIONAL COMMISSION FOR
SCIENCE, TECHNOLOGY & INNOVATION

Ref No: **811609** Date of Issue: **29/July/2019**


RESEARCH LICENSE




This is to Certify that Mr. Samson Muniato of Masinde Muliro University of Science and Technology, has been licensed to conduct research in Meru, Nairobi on the topic: SIZE METRIC-BASED EFFORT ESTIMATION METHOD FOR SERVICE-ORIENTED ARCHITECTURE SYSTEMS for the period ending : 29/July/2020.

License No: **NACOSTI/P/19/31**

811609
Applicant Identification Number


Director General
NATIONAL COMMISSION FOR
SCIENCE, TECHNOLOGY &
INNOVATION

Verification QR Code



**NOTE: This is a computer generated License. To verify the authenticity of this document,
Scan the QR Code using QR scanner application.**