MANAGING SOFTWARE EVOLUTION THROUGH MIDLEWARE AND POLICY-BASED SOFTWARE ADAPTATION FRAMEWORK

NOR HAZILAWATI AWANG

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy (Computer Science)

Faculty of Computing

Universiti Teknologi Malaysia

JANUARY 2015

Dedicated to a great family that I have.

"I sustain myself with the love of family."
Maya Angelou

ACKNOWLEDGEMENT

This challenging journey of completing my thesis would not become a reality without the help from numerous kind people around me. It is a great pleasure for me to thank these people. I would like to express my utmost gratitude to my main supervisor, Prof. Dr. Shamsul Shahibuddin for his trust in me and his vital support. I am forever indebted to Assoc. Prof. Dr. Wan Mohd Nasir Wan Kadir, my co-supervisor for his guidance, patience, and abundance of support throughout this research. Without his constant encouragement, the path would be extremely hard for me.

I would like to thank the management of HeiTech Padu Berhad for fully sponsoring my research. My special thanks go to En. Ahmad Abdul Ghani, CEO of HeiTech Academy for all his assistance, support and "just enough" pressure to keep me going. My sincere appreciation goes to Padusoft team, especially to the ex-Chief Operating Officer, En. Khairol Amin Mohd. Salleh for providing all supports I needed, from providing technical advice to allowing me to test the framework using the systems they developed.

A special thank and gratitude to my husband, Arham Rahimy Hariri for his understanding. My gratitude also goes to my parents, Tn. Haji. Awang and Pn. Hajah. Siti Zaharah for teaching me that "we have to always try the best in anything we do". Last but not least, thank you to Sarah and Hannah for keeping your mother sane throughout this meaningful journey.

ABSTRACT

Software evolution is a process that is needed in order for software to remain useful. Thus, software evolution should be properly planned and controlled to prevent its negative impact from affecting any organization. Software adaptation concept is one of the promising ways to control software evolution. In this approach, software is made adaptable to minimize the impact of change. A lot of researches on software adaptation focus on adaptability of mobile based and network application due to its context sensitivity and quality-of-service requirements. However, there is still lack of work in enterprise system domain with multiple delivery channels, which focus on adaptability of its context environment such as the changes introduced to its devices. Hence, the purpose of this research is to develop a middleware and policy-based, adaptation framework to manage negative effects of software evolution in an enterprise system. The main research focus is on the changes introduced at the device layer. The concept of policy is used to specify adaptations requirements. This research provides a framework called Middleware and Policy-Based Framework to Manage Software Evolution (MiPAF), which can be used to develop adaptive software, allowing parameterized and compositional adaptation. Furthermore, the framework can be used by client-server and web-based application. A policy language called MiPAF Policy Language (MPL) is created to be used with the framework. MiPAF is formally specified using Z Notation and the policy language is described using pseudo code. A tool is provided to assist developers in creating the policy. For evaluation of the framework, a set of runtime components were developed and implemented for Unit Trust System (UTS) Front-end and web-based UTS, two industrial-based case studies. The evaluation result shows that MiPAF excellently fulfil all the evaluation criteria described in this thesis.

ABSTRAK

Evolusi perisian adalah satu proses yang perlu kerana perisian berevolusi untuk kekal berguna. Proses ini perlu dirancang dan dikawal untuk mengelakkan kesan negatif kepada organisasi. Salah satu cara bagi mengawal kesan negatif ini ialah melalui adaptasi perisian. Kebanyakan kajian tentang adaptasi perisian tertumpu kepada bidang aplikasi mudah alih dan rangkaian kerana keperluan konteks kepekaan dan kualiti perkhidmatan. Walau bagaimanapun, masih terdapat kekurangan kajian di dalam bidang sistem perusahaan yang mempunyai pelbagai saluran penyampaian dan memfokuskan tentang adaptasi bagi pertukaran yang berlaku pada peranti. Tujuan tesis ini ialah membina satu rangka kerja adaptasi yang berteraskan konsep *middleware* dan polisi bagi mengawal kesan negatif evolusi perisian di dalam sistem perusahaan. Fokus utama kajian ialah tentang pertukaran yang berlaku pada peranti yang diguna pakai oleh sistem perusahaan. Konsep polisi digunakan untuk menyatakan keperluan adaptasi. Kajian ini menyediakan satu rangka kerja yang dinamakan *Middleware and Policy-Based Framework to Manage Software Evolution* (MiPAF). MiPAF membolehkan pembinaan perisian yang boleh diadaptasi, membenarkan adaptasi *parameterized* dan *compositional*. Rangka kerja ini boleh dimanafaatkan oleh perisian yang berasaskan pelanggan-pelayan dan juga perisian yang berasaskan sesawang. Bahasa khas untuk polisi yang dipanggil MiPAF Policy Language (MPL) dibina untuk diapplikasikan bersama dengan rangka kerja ini. MiPAF dispesifikasikan secara rasmi menggunakan *Z Notation* dan MPL diterangkan dengan menggunakan kod pseudo. Satu alat telah disediakan untuk membantu pengguna membina polisi. Untuk tujuan penilaian, komponen MiPAF telah dibina dan dilaksanakan untuk *Unit Trust System (UTS) Front-end* dan *UTS* berasas web, dua kajian kes yang diguna pakai di dalam industri. Keputusan penilaian MiPAF menunjukkan MiPAF telah memenuhi segala kriteria yang ditetapkan di dalam tesis ini dengan cemerlang.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	DECLARATION	ii
	DEDICATION	iii
	ACKNOWLEDGEMENT	iv
	ABSTRACT	v
	ABSTRAK	vi
	TABLE OF CONTENTS	vii
	LIST OF TABLES	xiii
	LIST OF FIGURES	xiv
	LIST OF ABBREVIATIONS	xvi
	LIST OF APPENDICES	xvii
1	**INTRODUCTION**	**1**
	1.1 Introduction	1
	1.2 Background of the Problem	2
	1.2.1 Issues related to software evolution	4
	1.2.2 The needs for adaptive software	5
	1.2.3 Software Evolution and Software Maintenance	6
	1.3 Statement of the Problem	7
	1.4 Objectives of the Study	8
	1.5 Scope of the Study	8
	1.6 Significance of the Study	10
	1.7 Thesis Organization	11

2	**LITERATURE REVIEW**	**13**
	2.1 Software Evolution	13
	2.1.1 Approaches in Minimizing the Effect of Software Evolution	15
	2.1.2 Why Software Adaptation Approach	17
	2.2 Software Adaptation	20
	2.3 Relationship between Software Evolution and Software Adaptation	21
	2.4 Adaptation Management	23
	2.4.1 Close-Loop Control System	23
	2.4.2 IBM Autonomic Computing	24
	2.4.3 Policy Based	26
	2.5 Related Software Adaptation Approaches	27
	2.5.1 Agent-oriented	27
	2.5.2 Architecture-based	29
	2.5.3 Component-based	31
	2.5.4 Middleware-based	32
	2.6 Comparing the Approaches	36
	2.7 Summary	37
3	**COMPARATIVE EVALUATION OF APPROACHES IN DEVELOPING ADAPTIVE SOFTWARE**	**38**
	3.1 Evaluation Criteria	38
	3.1.1 Scalability	38
	3.1.2 Context-awareness	39
	3.1.3 Performance	39
	3.1.4 Applicability	40
	3.1.5 Heterogeneity	40
	3.1.6 Dynamic Evolvability	40
	3.2 Result of the Comparative Evaluation	41
	3.2.1 Scalability	41
	3.2.2 Context-awareness	42

3.2.3 Performance 43
3.2.4 Applicability 44
3.2.5 Heterogeneity 45
3.2.6 Dynamic Evolvability 46
3.2.7 Summary of Result 46
3.3 Pros and Cons of the Current Middleware-based Approach 47
3.4 Summary 48

4 RESEARCH METHODOLOGY 49
4.1 Research Design and Procedure 49
4.2 Operational Framework 51
4.3 Instrumentation 53
4.4 Assumptions and Limitations 53
4.5 Methods for Evaluation 54
4.6 Case Study 55
4.6.1 Evaluation Steps 55
4.6.2 Evaluation Setting 57
4.6.3 Relationship between MiPAF and UTS System 58
4.6.4 Objects of the Case Study 59
4.7 Summary 62

5 THE MiPAF APPROACH 63
5.1 The Rationale behind MiPAF 63
5.2 Overview of MiPAF 67
5.2.1 Key approaches used in the Development of MiPAF 69
5.2.2 Unit Trust System Example 72
5.2.3 MiPAF Building Blocks 75
5.2.4 MiPAF Collaboration Diagram and the process of adaptation 83
5.2.5 MiPAF Policy Language 85
5.3 Summary 85

6 MiPAF DETAIL DESCRIPTION, FORMALIZATION, PROCESSES AND POLICY LANGUAGE 87

6.1 MiPAF Detail Description 88

6.1.1 Service Manager 89

6.1.2 Adaptation Manager 96

6.1.3 Service Infrastructure 104

6.1.4 Policy Repository 105

6.1.5 Device Controller 108

6.1.6 Context Monitor 111

6.2 Related processes in using MiPAF 114

6.3 Introduction to Software Process Engineering Metamodel 115

6.3.1 Conceptual Model of SPEM 116

6.4 The MiPAF Process 117

6.4.1 Requirement Phase 117

6.4.2 Analysis Phase 118

6.4.3 Design Phase 119

6.4.4 Development Phase 119

6.5 Policy Language 120

6.6 MiPAF Policy Language 120

6.6.1 MPL Ontology and Description 121

6.7 Summary 125

7 USING MiPAF IN INDUSTRIAL-BASED CASE STUDIES AND EVALUATION 127

7.1 Overview of MiPAF Runtime 127

7.2 Adaptation Requirements of UTS Front-end and Web-based UTS 128

7.3 Analysis of existing UTS Front-end printing components 130

7.4 Changes required to be done to UTS Front-end System 132

7.5 Design of the adaptation policy for UTS

Front-end using MiPAF Policy Language 132

7.5.1 Policy Editor 136

7.5.2 Implementation of Service Manager 136

7.5.3 Communication between UTS front-end and Service Manager 137

7.6 Implementation of the Adaptation Manager 139

7.6.1 Pseudo code for Adaptation Manager 139

7.6.2 Sample Codes for Storing Data in the Shared Memory 142

7.6.3 Expat XML Parser 143

7.7 Implementation of the Policy Repository 145

7.8 Implementation of the Device Controller 145

7.9 Implementation of the Context Monitor 146

7.10 Implementation of the Service Infrastructure 151

7.11 Test Scenario for Adaptability of the Industrial-Based Case Study 152

7.12 Analysis of the result 154

7.12.1 Scalability 154

7.12.2 Context Awareness 155

7.12.3 Performance 155

7.12.4 Applicability 156

7.12.5 Heterogeneity 157

7.12.6 Dynamic Evolvability 157

7.13 Summary 157

8 CONCLUSION AND FUTURE WORKS 158

8.1 Summary of the Research 158

8.2 Summary of Research Contributions 160

8.3 Future Works 162

REFERENCES 163

Appendices A-G 173-192

LIST OF TABLES

TABLE NO.	TITLE	PAGE
2.1	Criteria for comparison of selected approaches	37
3.1	Summary of result on the comparison analysis of the selected approaches to software adaptation	46
4.1	Operational Framework	54

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
1.1	High-Level View of Enterprise System Architecture	3
2.1	High Level Process in a Comprehensive, General Purpose Approach to Adaptive Software System	22
2.2	Basic Close Loop Control System	23
2.3	Functional Details of an Autonomic Manager	25
2.4	Format of SOAP Messages	35
2.5	Web Service Model	35
4.1	Research Flowchart	51
4.2	Evaluation Steps	56
4.3	Evaluation Settings	57
4.4	MiPAF Context Diagram for Case Study Implementation	59
5.1	MiPAF Context Diagram	67
5.2	High-Level Architecture of UTS	72
5.3	MiPAF Class Diagram	76
5.4	Changes Notification	81
5.5	MiPAF Collaboration Diagram	83
6.1	Communication Channels between Service Manager and Business Application	90
6.2	High-Level Component of Service Manager	91
6.3	Flow of Listener Component for Socket-based Communication	92
6.4	Flow of Information to and from Adaptation Manager	96
6.5	Components of Adaptation Manager	97
6.6	Basic Flow of Information to Adaptation Manager	98
6.7	The Flow for Adaptation Engine	99

6.8 High Level Component of Service Infrastructure 104
6.9 The Flow of Register Policy 106
6.10 High Level Components of Device Controller 109
6.11 Relationship between Context Monitor and Other MiPAF Components 112
6.12 SPEM Conceptual Model 116
6.13 Requirement Phase 118
6.14 Analysis Phase 118
6.15 Design Phase 119
6.16 Development Phase 119
6.17 MiPAF Policy Language Ontology 122
6.18 Sample of Default Policy 124
6.19 Sample of Reactive Policy 125
7.1 MiPAF Policy Language Editor 136
7.2 Sample Data Sent to the Service Manager 138
7.3 Code Snippet for Listener Implementation in C Language 139
7.4 Code Snippet for Storing Data into Named Shared Memory 142
7.5 Expat XML Parser Flow 143
7.6 Code Snippet for using Expat XML Parser to Parse XML Buffer 144
7.7 Components of Health Checker 147

LIST OF ABBREVIATIONS

ADL	–	Architecture Definition Language
AOM	–	Aspect-Oriented Modeling
AOP	–	Aspect-Oriented Programming
COM	–	Component Object Model
CORBA	–	Common Object Request Broker Architecture
CRM	–	Customer Relationship Management
DCOM	–	Dynamic Component Object Model
EJB	–	Enterprise Java Bean
ERP	–	Enterprises Resource Planning
EBNF	–	Extended Backus-Naur Notation
HTTP	–	Hyper-Text Transfer Protocol
IBM	–	International Business Machine
IPC	–	Inter-Process Communication
JADE	–	Java Agent Development Framework
MOP	–	Meta-Object Protocol
MAPE	–	Monitor, Analyze, Plan, Execute
MiPAF	–	Middleware and Policy-based Adaptation Framework
MPL	–	MiPAF Policy Language
OCL	–	Object Constraint Language
OMG	–	Object Management Group

OOP	–	Object Oriented Programming
RFID	–	Radio Frequency Identification
SOA	–	Service Oriented Architecture
SOAP	–	Simple Object Access Protocol
SPEM	–	Software Process Engineering Metamodel
SNA	–	System Network Architecture

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Initial Questionnaires to Validate Research Background	173
B	Rubric for the Evaluation Criteria	176
C	Data Type Specification for MiPAF	178
D	EBNF Specification for MiPAF Policy Language	182
E	Details on UTS Case Study	187
F	Questionnaires on MPL Editor Usability	190
G	Published Papers	191

CHAPTER 1

INTRODUCTION

This chapter consists of the introduction to this research. A brief introduction on software evolution is presented. Subsequently, background of the problem is described. Next, problem statements, objective; scope and significance of the research are discussed.

1.1 Introduction

The issue of software evolution has its root back in the 1970s based on the work done by Lehman and Belady [1]. In essence, software evolution refers to required changes in software due to changes in operating environment and/or user requirements. With the advancement of technology and dynamic business environment, changes mentioned above become more complex and difficult to manage.

Software evolution phenomenon has the potential to present huge problems to software projects as it affects all phases of software process. For long term survival, software evolution is inevitable [2]. Due to this, effective approach to software evolution has fired much interest in research community as organizations have a growing dependency on software [3].

Ideally, software should be developed with a capability to adapt itself to the changing requirements and environment. Software with this kind of capability is called

adaptive software. Changes in user requirements, the need for faster delivery of software, addition of features or bugs removal, are some of the factors driving the needs for software adaptation [4]. These needs are further emphasized currently due to the heterogeneity and complexity of computing environment; such as pervasive environment, varieties in software delivery channels and different type of networks and operating platforms.

Managing software evolution via software adaptation has been the subject of many researches such as in [4-9]. Software evolution and software adaptation are connected processes. Inter-connectedness between evolution management and adaptation management is described extensively in [10].

1.2 Background of the Problem

The role of software in an enterprise system is very important in providing operational supports in day-to-day operations. Enterprise system is referred to as intricate systems that communicate with and affect each other [11]. Example of enterprise systems are banking and financial system and postal system where various systems communicate with each other to deliver services to customers. A high level view of typical enterprise system architecture is depicted by the following Figure 1.1.

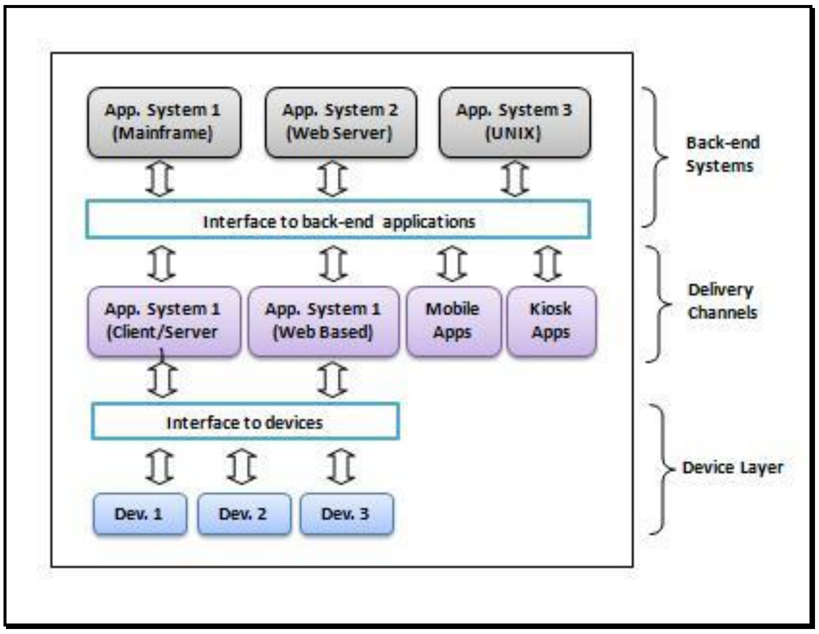
App. System 1 (Mainframe)
App. System 2 (Web Server)
App. System 3 (UNIX)
Back-end Systems
Interface to back-end applications
App. System 1 (Client/Server
App. System 1 (Web Based)
Mobile Apps
Kiosk Apps
Delivery Channels
Interface to devices
Device Layer
Dev. 1
Dev. 2
Dev. 3

Figure 1.2 High Level View of Enterprise System Architecture

In a typical environment of an enterprise system, 3 main levels exist, *Device layer*, *Delivery Channels layer*, and *Back-end Systems layer*. This architecture is observed to exist in large organizations in Malaysia such as financial institutions and public services. The observation is backed-up by a guided interview conducted with IT personnel of specific organizations and IT personnel from System Integrator (SI) Company. Please refer to Appendix A for the questionnaire used for the guided interview and analysis of the result. The description of each layer is as follows:-

- **Device Layer**

Enterprise applications, especially counter-based systems use variety of devices such as printer, biometric reader, scanner and barcode reader. These devices are logically group in Device Layer and are used by application systems at Delivery Channels.

- **Delivery Channels Layer**

The Delivery Channels level is where interaction with end users occurs. In the above diagram, different types of delivery channels exist, namely desktop-based applications that adopt client/server architecture, web-based application (thin client), mobile application and application deployed on kiosks.

- **Back-end Systems Layer**

The Back-end System level consists of various enterprise systems that need to interact with the delivery channels. The systems may include core systems of the organization such as Banking System, Enterprise Resource Planning (ERP) system, and Customer Relationship Management (CRM) system.

1.2.1 Issues related to software evolution

Enterprise systems will evolve [12-14] due to changes in user requirements or operating environment. Unplanned evolution of an enterprise system will incur high cost and risk [15]. In some organizations, the Delivery Channel Layer of an enterprise system evolves over time, but some of the devices evolve at a slower rate. For instance, although the application system at the Delivery Channel Layer is changed, the related devices are not. However, the new application system is expected to work with these legacy devices. Problems arise when the new system fail to support legacy devices due to lack of supported interfaces provided by the Application Programming Interface (API) of the devices.

Another possible problem that is foreseeable is when the Device Layer evolves. The evolution will happen when a new device is added to the Device Layer. Addition of a new device can happen in two scenarios. The first scenario is the replacement of existing devices with new devices from different vendors. Devices from different vendors come with different set of API. Therefore, applications at the Delivery Channel layer must adapt accordingly in order to use the devices.

The second scenario is the addition of new devices to support new user requirements. As the variety of devices increase, the device interfaces and protocols grows. Thus, the task of integrating these devices into the enterprise system becomes difficult and costly [16]. When any of these scenarios happen, application systems that use the device must adapt to this changes with minimal impact to its operation at lowest cost and minimum risk.

Changes in user requirements such as the need to reduce cost by sharing devices, can also lead to system evolution [13]. Less problems will be faced if the device sharing is supported by the operating system. However, for some proprietary devices, the sharing mechanism is not supported by the operating system. Examples of such devices are IBM4722 and IBM9068 passbook printers. Application systems that were designed to use device exclusively must be changed in order to adapt to this type of requirement. The problem becomes more profound if the same device needs to be shared by different applications. Worst, if the need to share the device only arises after these systems have been implemented. Changes introduced to any system at a later lifecycle are costly [17].

Application systems evolve due to technology advancement and changes in business requirements [12, 14]. As a result of the evolution, different type of client architecture existed such as thick-client, thin-client and smart-client. These client architectures require different ways of device integration. Problems will be further exacerbated when applications adopting different type of client-architecture need to share the same devices.

As the ecosystem of an enterprise system evolves over the time, the back-end systems become heterogeneous in-terms of operating platforms. These back-end systems may interact with other systems using different communication protocols such as SNA protocols, TCP/IP or SOAP. However, the evolution problems at the back-end system level are not within the scope of this research.

1.2.2 The needs for adaptive software

From the background of the problem, it is shown that there is a need for software that is adaptive in an enterprise domain. The adaptive software should support both upward scalability and downward scalability. This is due to different requirements of the application systems at delivery channels level.

The adaptive software is required due to heterogeneity of devices used by enterprise systems. The software need to adapt with the changes of devices, able to detect when a device fails and perform necessary reconfiguration. Furthermore, the adaptability of software in enterprise system must not be restricted in terms of its client architecture.

In an environment where the enterprise systems provide services to the public, performance of the system is critical. Device failure can affect performance of service delivery and in the end, may affect customer satisfaction. The enterprise system should be able to detect failed devices and adapt necessarily such as enabling users to use other devices in the network without halting the operations. The adaptive enterprise system must have an acceptable performance, based on user requirement. Apart from that, an adaptive enterprise system should not be too cumbersome to develop.

1.2.3 Software Evolution and Software Maintenance

The term software evolution and software maintenance are used interchangeably in a number of publications. Both terms revolve around changes subjected to software. However, according to Priyadarshiv and Kshivasagar [18], there are differences between software evolution and software maintenance. They argued that software maintenance comprises bug fixing activities to rectify defects in order to ensure the software meet its development purpose. The bug fixing activities happen after implementation phase and the functionalities of the software remain unchanged.

Software evolution refers to continuous changes subjected to software, which resulted in changes of one software state to a more complex and better state. Software evolution includes creation of new design which is originated from existing design, development of new functionalities or improvement of software performance.

In this research, we treat software maintenance as a subset of software evolution.

1.3 Statement of the Problem

The aim of this research is to provide an adaptation framework to manage software evolution based on compositional and parameterized adaptation approach in the domain of enterprise system. The framework will increase the adaptability of an enterprise system in the changing operating environment and user requirements. The problem statement brings about the main research problem that is:-

"How to manage software evolution by creating an adaptation framework using appropriate technique in compositional and parameterized adaptation approach?"

To answer the main research problem, a set of related questions must be addressed. The questions are as follows:-

1. What are compositional approach and parameterized approach and why these two approaches are selected?

    a. What are the state-of-the art for both approach?

    b. What are usage suitability of both approach – when, where to use?

    c. What are advantages and disadvantages of both approaches?

2. What are the main adaptability criteria for the framework to ensure it is beneficial for enterprise environment?

a. What are the main problems need to be addressed in an enterprise systems related to changes in operating environment and user requirements?

3. What are opportunities for improvement in both approaches that can be translated into the framework to manage software evolution?

4. How to make the framework easy to be used by software developers?

5. How to validate the framework to ensure its success in meeting the defined adaptability criteria?

1.4 Objectives of the Study

This research has the following objectives:-

- To investigate major techniques with respect to adaptation approaches in order to manage software evolution

- To develop a new adaptation framework in managing software evolution using compositional and parameterized adaptation approach.

- To demonstrate the applicability of the proposed approach using an industrial-based case study and the development of its supported tool.

1.5 Scope of the Study

In order to produce a new approach in managing software evolution via adaptation approach, this research is focused on the following scope:-

- Software evolution

  The focus of this research is to manage software evolution. The term *evolution* in software context refers to changes that happen to software during its lifetime [19]. This research will adopt the above definition of software evolution. To prevent a system from becoming unreliable, software evolution must be managed in software development process [15]. More explanation on software evolution can be found in Section 2.1.

- Software adaptation

  Software adaptation is a popular research area of late. There are many existing approaches to software adaptation. Researchers are looking at software adaptation from many angles such as from software architecture point of view [20, 21], component-based software development [22] and agent-oriented software engineering [23, 24].

- Two promising approaches to software adaptation are middleware and policy based approach. Middleware is said to be an important building block that impede software development [25]. Adaptive middleware on-the-other hand, allows modification to application systems when there are changes in user or operating environment [26]. Middleware can be designed to allow for separation between adaptive behavior and non-adaptive behavior in an enterprise system. This research has the interest to adopt middleware approach in developing the proposed framework. Sub-section 2.5.4 present a discussion on the topic of middleware

  Policy based approach involved the use of policy to specify the adaptation logic. Using this approach, a clear separation can be made between the business logic and adaptation specification. This separation is important since it promotes low coupling between the two, hence changes in can be implemented in a controlled manner.

- Case Study

Lehman in his work in Laws of Software Evolution coined the term E-type software [27]. E-type software can be defined as software that addresses real world problem. Therefore, the enterprise system described in the problem background is an instance of E-type software and thus, the system is subjected to software evolution.

Based on the researcher experience in developing and integrating enterprise systems, the proposed approach will be demonstrated using an industrial-based, E-type, case studies that is a Unit Trust System that has been implemented throughout Malaysia. The reason for selecting this system is that its Device Layer is always subjected to changes and there exist two type of front-end architecture or the system i.e. client-server based and web based.

1.6 Significance of the Study

Software evolution is an important issue that needs to be addressed to ensure longer life-time of implemented software thus “avoiding an early death” [12]. There is no way to prevent change in software since it has to react to the changing environment to ensure the software meets its purpose. Changes subjected to software during its post-deployment phase are not only costly but also risky. Thus, managing software evolution is one of major aspects in software development process since huge amount of project cost and effort are consumed in maintenance of existing system instead of creating a brand new system. One way to manage software evolution is via software adaptation approach.

There are other efforts towards software adaptation ranging from research specific for mobile devices [28, 29], multimedia systems [30], and network adaptability [29] to generic domain [31, 32]. However, there is lack of specific research that focuses on adaptability of an enterprise system. As technology evolves, more delivery channels and more new devices are introduced in an enterprise system. Enterprise systems must adapt to this changes to deliver required services. As the enterprise grows, more systems are added to the whole ecosystem. Existing application systems need to communicate with the new added systems at the lowest cost and lowest risk.

This study will contribute in providing an adaptation framework to overcome the above challenges faced by enterprise systems by enabling the systems to be more adaptive to the changing environments. The framework, Middleware and Policy-based Framework to Manage Software Evolution (MiPAF) is developed to enable software evolution to be planned and managed so that, the life of enterprise software can be increased and the maintenance cost can be reduced.

1.7 Thesis Organization

This thesis has the aim to develop an adaptation framework to manage software evolution. The chapters are organized as follows:-

Chapter 1: This chapter describes the background of the problem. Issues related to software evolution and the needs for software adaptation are also discussed. Problem statement, objective of the study and significance of the study are described.

Chapter 2: This chapter provides literature review on software evolution and software adaptation. It starts with the definition of software evolution and continues with discussion on existing approaches in minimizing the impact of software evolution where software adaptation is one of the approaches. Next, justification on the selection of software adaptation is presented. Software adaptation is further described and adaptation management is also discussed. Four software adaptation approaches are discussed in detail and criteria used to evaluate them are introduced.

Chapter 3: This chapter further describes the evaluation criteria introduced in Chapter 2. Comparative evaluation of the four approaches are presented based on the chosen criteria. Critical discussion is included in this chapter in order to evaluate which approach is best suited to be used in the development of MiPAF. Result of the evaluation is presented at the end of this chapter.

Chapter 4: This chapter describes research procedure, operational framework and instrumentation used to deliver the thesis objectives. Research assumptions and limitations and methods for evaluation are also described. This chapter also details up the case study to be used and approach for MiPAF evaluation using the said case study.

Chapter 5: This chapter starts with the rationale behind MiPAF and it proceeds with the overview of MiPAF. Key approaches used in MiPAF development is described. Next, MiPAF building block and collaration digram are discussed. The chapter also describes MiPAF Policy Language.

Chapter 6: This chapter describes in detail about MiPAF. Each component is presented formally using Z Notation. Next, the sub-sections in this chapter describe processes used to govern the application of MiPAF in order to manage software evolution. This chapter also includes description of MiPAF Policy Language (MPL) and ontology of MPL is also presented.

Chapter 7: This chapter discusses on the implementation of MiPAF using industrial-based case study. It focuses on adaptation requirements of the case studies, the design of adaptation policy and implementation of MiPAF runtime in order to test the adaptation behaviour of the case study. Towards the end of the chapter, case study result is analysed based on the evaluation criteria mentioned in Chapter 2.

Chapter 8: This chapter concludes the research. It provides research summary and contribution of the research. This chapter ends with suggestion for future works.

REFERENCES

1. Jayazeri, M. Species evolve, individuals age. *Eighth International Workshop on Principles of Software Evolution*. :IEEE. 2005.3-9.
2. Mens, T. and S. Demeyer, eds. *Software Evolution*. Berlin: Springer-Verlag Berlin Heidelberg, 2008.
3. Lehman, M.M. and J.F. Ramil. An Approach to a Theory of Software Evolution. in *Proceedings of the 4th International Workshop on Principles of Software Evolution.* 2001. New York:.ACM. 2001, 70-74.
4. Nary, S. and C. Lawrence. Software architecture adaptability: an NFR approach. in *Proceedings of the 4th International Workshop on Principles of Software Evolution*. 2001. New York: ACM.2001. 52-61.
5. Cazzola, P.D.W., A.M.A. Ghoneim, and P.D.G. Saake, eds. Software Evolution through Dynamic Adaptation of Its OO Design. in *Lecture Notes in Computer Science. Vol. 2975/2004.*Berlin: Springer Berlin/Heidelberg. 2004.67-80.
6. Ghoneim, A.M.A., Reflective and Adaptive Middleware for Software Evolution of Information Systems. in *Fakultät für Informatik* 2007. Germany: Otto-von-Guericke-Universität Magdeburg. 2007. 154.
7. Oreizy, P., N. Medvidovic, and R.N. Taylor. Architecture-Based Runtime Software Evolution. in *Proceedings of the 1998 (20th) International Conference on Software Engineering*. 1998. Kyoto: IEEE Computer Society.1998.177-186.
8. Perez, J., et al. Dynamic Evolution in Aspect-Oriented Architectural Models. in *Lecture Notes in Computer Science.* 2005. Berlin: Springer Berlin/Heidelberg. 2005.59-76.
9. Zhang, H., K. Ben, and Z. Zhang. A Reflective Architecture-Aware Framework to Support Software Evolution. in *Proceedings of the 9th International*

Conference for Young Computer Scientists.November 18-21. :IEEE.2008.1145-1149.

10. Oreizy, P., Medvidovic, N., Taylor, R.N., Gorlick, M.M., Heimbigner, D., Johnson, G., Quilici, A., Rosenblum, D.S., Wolf, A.L. An Architecture Based Approach to Self-Adaptive Software. *IEEE Intelligent System*.1999.14(3):54 - 62.
11. Iacob, M.-E. and H. Jonkers. Quantitaive Analysis of Enterprise Architectures. in *Interoperability of Enterprise Software and Applications*. :Springer London. 239-252;2006.
12. Godfrey, M.W. and D.M. German. The past, present, and future of software evolution. in *Frontiers of Software Maintenance.* 2008. *FoSM 2008.* Sept 28 - Oct 4, 2008. :IEEE. 2008. 129-138.
13. Hu, H. Software Evolution Based on Software Architecture. in *The Fourth International on Computer and Information Technology, 2004. CIT '04.* :IEEE. 2004. 1092-1097.
14. Roland, T.M., Software evolution: let's sharpen the terminology before sharpening (out-of-scope) tools. in *Proceedings of the 4th International Workshop on Principles of Software Evolution*. 2001. :ACM. 2001. 114-121.
15. Subramanyam, R. Position Statement: How Well Technology Supports Software Evolution. in *COMPSAC '08 Proceedings of the 32nd Annual IEEE International Computer Software and Application Conference*. 2008. Washington: IEEE Computer Society. 2008. 3
16. de Deugd, S., et al. SODA: Service Oriented Device Architecture. *Pervasive Computng*, :IEEE, 2006. 5(3): 94-96.
17. Stephens, M. and D. Rosenberg. *Extreme Programming Refactored: The Case Against XP*. :Apress. 3;2003.
18. Priyadarshi Tripathy, K.N. *Software Evolution and Maintenance*. Hoboken, New Jersey: John Wiley & Sons. 416; 2014.
19. Reiss, S.P. Evolving Evolution [software evolution]. in *Eighth International Workshop on Principles of Software Evolution.* September 5-6, 2005. :IEEE. 2005. 136-139.
20. Fayad, M.E., H.S. Hamza, and H.A. Sanchez. Towards scalable and adaptable software architectures. in *IEEE International Conference on Information Reuse and Integration, Conf, 2005. IRI -2005.* :IEEE. 102-107;2005.

21. Xiong, X. and Z. Weishi. A Framework of Software Component Adaptation. in *Algorithms and Architectures for Parallel Processing*. 2007, Berlin: Springer Berlin/Heidelberg. 2007. 153-164.

22. Holger, K., N. Dirk, and R. Andreas, A component model for dynamic adaptive systems. in *International workshop on Engineering of software services for pervasive environments: in conjunction with the 6th ESEC/FSE joint meeting*. 2007, Dubrovnik, Croatia :ACM. 2007. 21-28.

23. Qureshi, N.A. and A. Perini. An Agent-Based Middleware for Adaptive Systems. in *The Eighth International Conference on Quality Software*. 2008.

24. Seungwok, H., S. Sung Keun, and Y. Hee Yong. Dynamic Software Adaptation with Dependence Analysis for Multi-Agent Platform. in *International Conference on Computational Science and its Applications.* 2007. :IEEE Computer Society. 2007. 185-191.

25. Newcomer, E. and G. Lomow. *Understanding SOA with Web Services. :* Pearson Education, 2005.

26. Sadjadi, S.M. A Survey of Adaptive Middleware. *Technical Report, Computer Science and Engineering*. : Michigan State University. 2003.

27. Lehman, M.M. Laws of Software Evolution Revisited. in *Proceedings of the 5th European Workshop on Software Process Technology. :* Springer-Verlag. 1996. 108-124.

28. Maciel da Costa, C., M. da Silva Strzykalski, and G. Bernard. An Aspect Oriented Middleware Architecture for Adaptive Mobile Computing Applications. in *31st Annual International*. *Computer Software and Applications Conference. COMPSAC 2007. July 24-27,* 2007. : IEEE. 2007. 81-86.

29. Mukhija, A. and M. Glinz. A framework for dynamically adaptive applications in a self-organized mobile network environment. in *Proceedings of 24th International Conference on Distributed Computing Systems Workshops*. March 23-14, 2004. : IEEE. 2004. 368-374.

30. Vu Hoang, H. and H. Hoang Dang. An Application-aware Adaptive Middleware Architecture for Distributed Multimedia Systems. in *First International Conference on Communications and Electronics. ICCE '06.* Oct 10-11, 2006. :IEEE. 2006. 141-146.

31. Curry, E. *Adaptive and Reflective Middleware (Middleware for Communications).* 2004: 29-52.

32. Gjorven, E., et al. Self-Adaptive Systems: A Middleware Managed Approach. in *Lecture Notes in Computer Science*. 2006. 3996: 15-27.

33. Dictionary, M.-W.s.O. *Merriam-Webster's Online Dictionary* 2009 [cited 2009 29 January]; Available from: http://www.merriam-webster.com/dictionary/evolution.

34. Yang, H. and M. Ward. *Successful Evolution of Software Systems*: Artech House. 2002.

35. Canfora, G. Software Evolution in the Era of Software Services. in *Proceedings of 7th International Workshop on Principles of.Software Evolution. : IEEE.* 2004.

36. Bennet. K.H. , Rajlich, V.T. Software Evolution: A Roadmap. in *Proceedings of the Conference on IEEE International Conference on the Future of Software Engineering. 2000.* Limerick, Ireland: ACM. 2000. 73-87.

37. Qianxiang, W., et al. A component-based approach to online software evolution: Research Articles. *J. Softw. Maint. Evol., 2006*. 18(3): 181-205.

38. Michele Ceccarelli, L.C., Gerardo Canfora, Massimiliano Di Penta. An Eclectic Approach for Change Impact Analysis. in *2010 ACM/IEEE 32nd International Conference on Software Engineering.* 2010. Cape Town : IEEE. 2010. 163-166.

39. Vora, U. Change Impact Analysis and Software Evolution Specification for Continually Evolving Systems. in *2010 Fifth International Conference on Software Engineering Advances.* 2010. Nice, France : IEEE. 2010. 238-243.

40. Mens, T., et al. Challenges in Software Evolution. in *Eighth International Workshop on Principles of Software Evolution*. September 5-6, 2005. : IEEE. 2005. 13-22.

41. Madhavji, N.H., J.F. Ramil, and D.E. Perry. Editors. *Software Evolution and Feedback: Theory and Practice.* 2006 : John wiley and Sons. 2006.

42. Ciraci, S., P. van den Broek, and M. Aksit. A Taxonomy for a Constructive Approach to Software Evolution. *Journal of Software.* 2007. 2(2): 84-97.

43. Robbes, R., M. Lanza, and M. Lungu. An Approach to Software Evolution Based on Semantic Change. in *Fundamental Approaches to Software Engineering*. 2007. Berlin : Springer Berlin / Heidelberg. 2007. 27-41.

44. Riling, J., et al. *Story-driven Approach to Software Evolution. Software.* August, 2008. : IET. 2008. 2(4): 304 - 320.

45. Pandey, D., U. Suman, and A.K. Ramani. An Effective Requirement Engineering Process Model for Software Development and Requirements Management. in *International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom).* Oct 16 -17, 2010. :IEEE. 2010. 287-291.

46. Ferreira, M.G. and J.C.S. do Prado Leite. *Requirements Engineering with a Perspective of Software Evolution.* 2011.

47. Souza, V.S., et al. Requirements-driven software evolution. *Computer Science - Research and Development*, 2013. 28(4): 311-329.

48. Garlan, D., et al. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. *IEEE Computer Society,* 2004: 46 - 54.

49. Lundesgaard, S.A., et al. Constriction and Execution of Adaptable Applications Using an aspect-Oriented and Model Driven Approach. in *Lecture Notes in Computer Science.* 2007. : Springer Berlin/Heidelberg. 2007. 76-89.

50. Kell, S. *A Survey of Practical Software Adaptation Techniques.* J. UCS, 2008. 14(13): 2110-2157.

51. Harker, S.D.P., K.D. Eason, and J.E. Dobson. The change and evolution of requirements as a challenge to the practice of software engineering. in *Proceedings of IEEE International Symposium on Requirements Engineering*. 1993. : IEEE. 1993.

52. McKinley, P.K., et al. *Composing adaptive software.* Computer, 2004. 37(7): 56-64.

53. John, C.G. and N.T. Richard, Policy-based self-adaptive architectures: a feasibility study in the robotics domain. in *Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing system.* 2008, Leipzig, Germany : ACM. 2008.

54. Tomasz, S., S. Robert, and Z. Krzysztof. Policy-based Context-aware Adaptable Software Components for Mobility Computing. in *10th IEEE International Enterprise Distributed Object Computing Conferenc. 2006 : IEEE.* 2006.

55. Eliassen, F., et al. Evolving self-adaptive services using planning-based reflective middleware. in *Proceedings of the 3rd workshop on Adaptive and*

reflective middleware ACM International Conference Proceeding Series. 2006. : ACM. 2006. 1-6.

56. Shang-Wen Cheng ; Garlan, D.S., B. Evaluating the effectiveness of the Rainbow self-adaptive system. in *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*. 2009. : IEEE. 2009. 132-141.
57. Michael, B. Introduction to Close Loop Control. *Embedded System Design*. July 31st,2002. [cited 2010 12th march 2010]; Available from: http://www.embedded.com/story/OEG20020726S0044.
58. Team, I.A.C. An architectural Blueprint for Autonomic Computing. *Autonomic Computing White Paper*. June 2006 [cited 2010 10th March 2010]; 4th Edition:[Available from: http://www-01.ibm.com/software/tivoli/autonomic/pdfs/AC_Blueprint_White_Paper_4th.pdf.
59. Jennings, N.R., Agent-Oriented Software Engineering. in *Lecture Notes in Computer Science*. 2004: Springer Berlin/Heidelberg.2004. 4-10.
60. Henderson-Sellers, B. and P. Giorgini. *Agent-Oriented Methodolgies.* : IGI Publishing. 2005.
61. Paek, K. and T. Kim. AOM: An Agent Oriented Middleware Based on Java. in *Lecture Notes in Computer Science*. 1999. : Springer Berlin/Heidelberg. 1999.
62. Tarkoma, S. and M. Laukkanen. Adaptive Agent-Based Service Composition for Wireless Terminals. in *Lecture Notes in Computer Science*. 2003. : Springer Berlin/Heidelberg. 2003.
63. Markus, C.H., Julie, A. McCann. A survey of autonomic computing - degrees, models and applications. *ACM Comput. Survey*. 2008. 40(3): 1-28.
64. Jeff, M. and K. Jeff. Dynamic structure in software architectures. in *Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering*. 1996. San Francisco, California, United States: ACM. 1996.
65. Michel, W., L. Antonia, and L. Jose, Fiadeiro. A graph based architectural (Re)configuration language. in *Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering*. 2001. Vienna, Austria: ACM. 2001.

66. Morin, B., et al. An Aspect-Oriented and Model-Driven Approach for Managing Dynamic Variability, in *Lecture Notes in Computer Science.* 2008. :Springer-Verlag Berlin Heidelberg. 2008. 782-796.
67. Kienzle, J., et al., *Models in Software Engineering.* 2008. : Springer Berlin/Heidelberg. 2008. 1-6.
68. Bekker, C. and P. Putter. Reflective architectures: requirements for the future distributed environments. in *Proceedings of the Fourth Workshop on Future Trends of Distributed Computing Systems.* 1993. : IEEE. 1993.
69. Kurt, G., et al.Modeling of component-based adaptive distributed applications, in *Proceedings of the 2006 ACM symposium on Applied computing.* 2006, Dijon, France: ACM. 2006.
70. Batista, T., A. Joolia, and G. Coulson. Managing Dynamic Reconfiguration in Component-Based Systems. in *Software Architecture.* 2005. : Springer Berlin/Heidelberg. 2005. 1-17.
71. Frei, A., A. Popovici, and G. Alonso. Event based systems as adaptive middleware platforms. in *Workshop of the 17th Europeean Conference for Object-Oriented Programmin*g.July 21-23, 2003. : IEEE. 2003.
72. Afandi, R., J. Zhang, and C.A. Gunter. AMPol-Q: Adaptive Middleware Policy to Support QoS. *in Lecture Notes in Computer Science - Service-Oriented Computing – ICSOC 2006*, 2006. 165-178.
73. Colman, A., et al. *A*daptive application-specific middleware. in *Proceedings of the 3rd workshop on Adaptive and reflective middleware. Proceedings of the 1st workshop on Middleware for Service Oriented Computing (MW4SOC 2006) ACM International Conference Proceeding Series*. : ACM. 2006. 6-11.
74. Deitel, H.M., et al. *Web Services A Technical Introduction* : Prentice Hall. 2003.
75. Meyer-Wegener, K. Thirty Years of Server Technology - from Transaction Processing to Web Services. in *Lecture Notes in Computer Science.* 2005, :Springer Berlin / Heidelberg. 2005. 51-65.
76. San-Yih, H., et al. On Composing a Reliable Composite Web Service: A Study of Dynamic Web Service Selection. in *IEEE International Conference on Web Services, 2007, ICWS 2007.* : IEEE. 2007.
77. Noh-sam, P. and L. Gil-haeng. Agent-based Web service middleware. in *Global Telecommunications Conference, 2003. GLOBECOM'03.* :IEEE. 2003.

78. UDDI. *UDDI 101*. 2006 [cited 2009 29 January]; Available from: http://uddi.xml.org/uddi-101.

79. WSDL. *Web Service Description Language (WSDL)*. 2006 [cited 2009 29 January]; Available from: http://www.w3.org/TR/wsdl.

80. Khosrow-Pour, M.. *Encyclopedia of E-Commerce, E-Government, and Mobile*, : IGI Publishing. 2006.

81. SOAP. *Simple Object Access Protocol (SOAP)*. 2006 [cited 2009 29 January]; Available from: http://www.w3.org/TR/soap12-part1/.

82. Arora, G. and S. Kishore. *XML Web Services Professional Projects*, :Premier Press. 2002.

83. Foggon, D., et al. *Programming Microsoft .NET XML Web Services*, :Microsoft Press. 2004.

84. *Software Sustainability Institute*. 2014 [cited 2014 19/11/2014]; Available from: http://www.software.ac.uk/.

85. Gordon, I.. *Essential Software Architecture*. 2012, Heidelberg, Germany : Springer. 2012. 304.

86. Hong, J.-y., E.-h. Suh, and S.-J. Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*. 2009. 36(4): 8509-8522.

87. Floch, J., et al. Using Architecture Model for Runtime Adaptability. *IEEE Computer Society*. 2006. 23(262-70).

88. Parra, C. and L. Duchien. Model-Driven Adaptation of Ubiquitous Applications. *Electronics Communicaion of EASST*. 2008. :EASST. 2008. 11(2008).

89. Michael, C., et al. An Efficient Component Model for the Construction of Adaptive Middleware, in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*. 2001. : Springer-Verlag. 2001.

90. Clements, P., et al. The Duties, Skills, and Knowledge of Software Architects. in *The Working IEEE/IFIP on Conference Software Architecture, 2007. WICSA '07*.2007. : IEEE. 2007.

91. Vickers, B. Architecting a software architect. 2004. in *Proceedings Aerospace Conference*. 2004. : IEEE. 2004.

92. Staehli, R., F. Eliassen, and S. Amundsen. Designing adaptive middleware for reuse. in *ACM International Conference Proceeding Series; Proceedings of the 3rd workshop on Adaptive and reflective middleware*. 2004. : ACM. 2004. 80: 189-194.

93. Christine, J. and P. Dewayne. Composable context-aware architectural connectors, in *Proceedings of the 1st international workshop on Software architectures and mobility*. 2008, Leipzig, Germany : ACM. 2008.

94. Mubarak, H. Developing Flexible Software Using Agent-Oriented Software Engineering. *Software*. : IEEE, 2008. **25**(5): 12-15.

95. Cazzola, W., A. Ghoneim, and G. Saake. RAMSES: a reflective middleware for software evolution. in *Proceedings of ECOOP Workshop on Reflection, AOP and Meta-Data for Software Evolution, Oslo, Norway*. 2004. 5-63.

96. Lázaro, M., and Marcos, E. Research in Software Engineering: Paradigms and Methods. in *Proceedings of the 17th International Conference on Advanced Information System (CAiSE'05)*. 2005. 517-522.

97. Leedy, P.D.a.O., J.E. *Practical Research: Planning and Design* 8th ed. New Jersey: Prentice Hall. 2005.

98. Streb, C.K., Exploratory Case Study. Encyclopedia of Case Study Research. *SAGE Publications, Inc*, Thousand Oaks, CA: SAGE Publications, Inc.2009. 373-375.

99. Garger, J. *Using the Case Study Method in PhD Research*. 2013 20th August 2013 [cited 2014 8th July 2014].

100. Flint, S. A Conceptual Model of Software Engineering Research Approaches, in *Software Engineering Conference, 2009. ASWEC '09. Australian. April 14-17,* 2009. 229-236.

101. Potts, C. Software-engineering research revisited. *Software.* : IEEE. 1993. 10(5): 19-28.

102. Spivey, J.M. *The Z Notation : A Reference Manual*. United Kingdom: Prentice Hall International (UK) Limited. 1992.

103. Schuppenies, R. and S. Steinhauer. *Software Process Engineering Metamodel.* : OMG Group. 2002.

104. Damianou, N.C. *A Policy Framework for Management of Distributed System. Ph. D Thesis.* Department of Computing, Imperial College of Science, Technology and Medicine. University of London; 2002.

105. Keeney, J. *Completely Unanticipated Dynamic Adaptation of Software*, Ph. D Thesis. Trinity College. University of Dublin: Dublin; 2004

106. *XML Validation*. [cited 2nd February 2013]; Available from: http://www.w3schools.com/xml/xml_dtd.asp.

107. Shirrell, J. *XML: A Deeper Understanding*. [cited 2nd February 2013]. Available from: http://www.xmlbook.info

108. *Type-Length-Value*. [cited 2013 4th February]; Available from: http://en.wikipedia.org/wiki/Type-length-value#cite_note-1.

109. *TLV Page Info*. [cited 2013 4th of February]; Available from: https://cwiki.apache.org/DIRxASN1/tlv-page-info.html.

110. *Operating System Design/Processes/SharedMemory*. 2011 18th December 2011 [cited 2013 6th February]; Available from: http://en.wikibooks.org/wiki/Operating_System_Design/Processes/SharedMemory.

111. *Shared Memory*. 2005 August 2005 [cited 2013 6th February]; Available from: http://searchcio-midmarket.techtarget.com/definition/shared-memory.

112. Gouvas, P., T. Bouras, and G. Mentzas. An OSGi-Based Semantic Service-Oriented Device Architecture. *Lecture Notes in Computer Science*. : Springer Berlin Heidelberg. 2007.