WEB DEVELOPMENT PRODUCTIVITY IMPROVEMENT THROUGH OBJECT-ORIENTED APPLICATION FRAMEWORK

MOHAMMAD NURUZZAMAN

MASTER OF SCIENCE
UNIVERSITI UTARA MALAYSIA
2014

812302

Awang Had Salleh
Graduate School
of Arts And Sciences

Universiti Utara Malaysia

PERAKUAN KERJA TESIS / DISERTASI
(Certification of thesis / dissertation)

Kami, yang bertandatangan, memperakukan bahawa
(We, the undersigned, certify that)

MOHAMMAD NURUZZAMAN

calon untuk Ijazah — MASTER
(candidate for the degree of)

telah mengemukakan tesis / disertasi yang bertajuk:
(has presented his/her thesis / dissertation of the following title):

"WEB DEVELOPMENT PRODUCTIVITY IMPROVEMENT THROUGH OBJECT-ORIENTED APPLICATION FRAMEWORK"

seperti yang tercatat di muka surat tajuk dan kulit tesis / disertasi.
(as it appears on the title page and front cover of the thesis / dissertation).

Bahawa tesis/disertasi tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan, sebagaimana yang ditunjukkan oleh calon dalam ujian lisan yang diadakan pada : *10 Mac 2014.*
That the said thesis/dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on:
March 10, 2014.

Pengerusi Viva: *(Chairman for VIVA)*	Assoc. Prof. Dr. Wan Rozaini Sheik Osman	Tandatangan *(Signature)* [illegible]
Pemeriksa Luar: *(External Examiner)*	Dr. Nazean Jomhari	Tandatangan *(Signature)* [illegible]
Pemeriksa Dalam: *(Internal Examiner)*	Dr. Azida Zainol	Tandatangan *(Signature)* [illegible]
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	Dr. Azham Hussain	Tandatangan *(Signature)* [illegible]
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	Assoc. Prof. Hatim Mohamad Tahir	Tandatangan *(Signature)* [illegible]

Tarikh:
(Date) *March 10, 2014*

PERMISSION TO USE

ABSTRAK

Kebanyakan aplikasi web yang digunakan untuk komersial dan industri adalah kompleks, sukar untuk dilaksanakan, berisiko untuk diselanggara dan memerlukan pemahaman yang mendalam tentang keperluan untuk penyesuaian. Pasaran perisian pada masa ini lebih berdaya saing, maka produktiviti telah menjadi perhatian utama dalam industri pembangunan perisian. Tujuan kajian ini adalah untuk mereka bentuk dan membangunkan satu kerangka aplikasi untuk mempercepatkan produktiviti pembangunan web melalui teknologi berorientasikan objek. Ini akan membenarkan penyesuaian, mengguna semula rekabentuk dan menjana kod secara automatik untuk membantu meningkatkan produktiviti sebagai kejayaan meyelesaikan masalah yang diberi. Kajian ini menggunakan *Systematic Literature Review (SLR)* untuk mengenalpasti sumber kerumitan dan faktor pengeluaran. Metodologi pembangunan tangkas (*Agile*) telah digunakan untuk mereka bentuk kerangka dan ianya telah disahkan dengan data empirikal dari dua projek komersial. Penemuan kajian mendapati bahawa Kerangka Aplikasi Berasaskan Objek (*OOAF*) mempunyai faktor ketara yang mempengaruhi produktiviti dan secara dramatik meningkatkan produktiviti yang lebih tinggi berbanding pendekatan tradisional. Ia telah memenuhi keperluan semasa dengan mengurangkan kerumitan, usaha-usaha pembangunan dan mempercepatkan produktiviti pembangunan web. Kajian ini menyumbang dalam bidang kejuruteraan perisian, khususnya dalam bidang peningkatan produktiviti perisian dan penyesuaian perisian. Ini akan membawa kepada masa pembangunan yang lebih cepat kepada industri perisian.

Katakunci: Kerangka Aplikasi Berasaskan Objek, Pembangunan Web, Produktiviti Perisian, Metrik Perisian, Pengukuran Produktiviti.

ABSTRACT

Most of the commercial and industrial web applications are complex, difficult to implement, risky to maintain and requires deep understanding of the requirements for customization. As today's software market is more competitive, productivity has become a major concern in software development industry. The aim of this research is to design and develop an application framework for accelerating web development productivity through object-oriented technology. It allows customization, design reuse and automatic code generation to support productivity improvement as a breakthrough solution for the given problem. This research employed systematic literature review (SLR) to identify the source of complexity and productivity factors. Agile development methodology was used to design the framework and it was validated by empirical data from two commercial projects. Results showed that object-oriented application framework (OOAF) has significant factors that affect productivity and dramatically improve higher productivity over traditional approach. It has fulfilled the current needs by reducing complexities, development efforts and accelerates web development productivity. This research contributes in the area of software engineering, specifically in the field of software productivity improvement and software customization. These will lead to faster development time for software industries.

Keywords: Object-oriented Application Framework, Web Development, Software Productivity, Software Metrics, Measuring Productivity.

ACKNOWLEDGEMENT

I would like to express my gratefulness to everyone contributed in completing this dissertation. This dissertation would not been possible without the guidance and encouragement from academic, industry and personal supporters.

It was my pleasure to study under Dr. Azham Bin Hussain's supervision. It is not enough to say him that thank you very much for his guidance to help me to achieve my goal. Without his valuable support, my thesis would not have been possible. I would like to express my thanks to my co-supervisor Assoc. Professor Hatim Mohd Tahir for his comments which help improving my work. This dissertation could not been completed without the intellectual; challenges, resources, departmental data and professional support provided by *ENT Broadband Sdn Bhd, SK United Packaging Sdn Bhd* and *Accenture Malaysia.*

Most importantly, I would like to thank my family, who provided me with love and patience during this journey. To my mom and dad, friends and colleagues for their invaluable life lessons and encouragement to accept the next challenge. To my wife, Fatin Nurul Aini Binti Ghazali for her unconditional patience and constant support.

Finally, I would like to thank all my friends, university staffs, the direct and indirect supports helped me completing my dissertation in time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

3GL	Third Generation Language
4GL	Fourth Generation Language
AAF	Adaption Adjustment Factor
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
B2B	Business to Business
BPM	Business Process Management
C2C	Customer to Customer
CBSE	Component-Based Software Engineering
CBD	Component-Based Development
CM	Code Modified
COTS	Component of the Shelf
COCOMO	Component Cost Model
DM	Design Modified
D&D	Design and Development
D2D	Domain to Domain
ESLOC	Equivalent Source Line of Code
FP	Function Point
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
JSF	Java Server Face
JSP	Java Server Page
J2EE	Java 2 Enterprise Edition
JDHTML	Java Dynamic Hyper Text Markup Language
KSLOC	Kilo (Thousand) Source Line of Code
LOC	Line of Code
MD	Man Days
MVC	Model View Controller

MFC	Microsoft Foundation Class
MSC	Multi-media Super Corridor
MSLOC	Modified Source Line of Code
NSLOC	New Source Line of Code
OO	Object Oriented
OOP	Object Oriented Programming
OOM	Object Oriented Method
OOT	Object Oriented Technology
OOAF	Object Oriented Application Framework
OOSAD	Object Oriented System Analysis and Design
QA	Quality Assurance
RDBMS	Rational Database Management System
RSLOC	Reuse Source Line of Code
SDLC	Software Development Life Cycle
SLR	Systematic Literature Review
SLOC	Source Line of Code
UML	Unified Modeling Language
UFP	Unadjusted Function Point
VOS	Visual Object Sharing
WM	Work Modified
XSLT	Extensible Stylesheet Language Transformations
XML	Extended Markup Language

CHAPTER ONE
INTRODUCTION

This chapter presented an overview of the thesis. It described the problem statement and continues with research questions, objectives, scope and contribution. At the end, it presents the organization of the thesis.

1.1 Background

Object-Oriented (OO) based web application development is not easy, mapping user requirements into a function is complex, customization requires deep understanding and risky to maintain. Technology for completely integrated user interface, reuse design, customization environment and implementation is still immature in the area of web engineering. It is different from traditional web development as it focuses on visual elements (Kaur & Singh, 2008). OO software development method includes requirements analysis, system design, development, testing and documentation that enable web engineers to repeat Software Development Life Cycle (SDLC) phases and avoid possible failure of current ubiquitous web. This revolution makes easier for web engineers to develop software packages and also made a significant impact to working on it.

Previously, most of the developed web applications were procedure-oriented. It is an ever-growing complexity due to an exponential increase in software size. It also make it unsuitable to reuse and customize based on user preferences. Considering this effort has pushed legacy applications into the new OO-based web application development. There are numerous recurring efforts, particularly in the user interface design and coding phase (Pardo Leite, Yu & Liu, 2005; David, 2012). An approach is needed to accomplish web application customization and reuse design for improving development productivity. The necessity of an OO framework designer become obvious due to complexity, difficulties to implement, hard to reuse and requires deep understanding of requirements for customization. The idea behind this approach is just reuse it, do not need to develop anything that already exists. It increases greater consistency of operation, reduced development time, reduce complexity, easier maintenance, smaller resulting code size, increase productivity and reliability of the web applications (Frakes & Kyo Kang, 2005). Reuse has a computable and significant impact on reducing development efforts and improving software quality (Kung-Kui Lau & Zheng Wang, 2007).

Web engineers often proclaim *"Do not reinvent, the inverse of reinvention is reuse"*. It is said that easier to proclaim than it is to achieve. The projects which were fail due to quality of application framework, poor design, lack of clarification of requirements, developer's skill and administration issues. Therefore, it will be significantly important to examine current problems in Object-Oriented Application Framework (OOAF) development and looking for alternate solutions. There are several factors need to take into consideration when designing OOAF such as requirements simplicity, complexity in object relationship, classes collaboration and complication in using an application framework. To escalate web application development productivity, this research proposed OOAF as an alternate solution for web application development to overcome the above mentioned problems in OOAF.

1.2 Research Motivation

I was motivated into this research after I started working at Daifuku, a global industrial material handling company. My role was as a Development Team Leader since 2010-2012. Experiences and knowledge I have gathered from Daifuku can be accumulated into a strong research problem. I found that web developers/ engineers faces three intractable problems—lack of clarification in requirements, complexity of the domain and increased development efforts. This work compliments ongoing research problem in the area of software engineering. In particularly, I am also interested in analyzing and developing new methods and tools for guiding software development decisions for finding better way to incorporate these concepts into education.

1.3 Problem Statement

My research addressed that complexity is an ever-present obstacles in web application (software) development and three intractable problems in—requirements clarification, domain complexity and development efforts. This is due to customization techniques, complex relationships among the objects, collaboration, size of classes and the buildup of low-level in detail. The predictable result is that run-time errors found in developed web applications. This is crucial and significantly more expensive to correct software defects once they have reached the end user compared with earlier in the development process (Hevner et al, 2005). Because of this increasing cost of correcting defects and the need for software development productivity and quality become main concern.

Complexity destroys modularity, reduce productivity and software quality (Ondrej, Jiri & Jan, 2012; Clarke & Walker, 2001). As a result, web engineering team requires great efforts to developing today's web applications. A new OOAF is essential to support more effective and rapid development of applications with lower efforts. According to Alvaro, Santana de Almeida & Romero de Lemos (2010); Sudhakar, Farooq & Patnaik (2012) application framework development by integrating component-based software engineering (CBSE) and object-oriented technology (OOT) could achieve web application customization and increase web development productivity.

This research proposed visual object sharing technique to support customizable and reusable OOAF architecture. A big picture behind the approach to reduce development efforts, complexities and increase productivity through visual object sharing technique in OOAF has been demonstrated.

1.4 Research Questions

i. What are the complexities and productivity factors to achieve reuse design and easier customization in OO-based web application development?

ii. How to design and develop OOAF that could support design reuse and existing web applications customization?

iii. How to evaluate the proposed OOAF, can it increase development productivity for web engineers and developers?

1.5 Research Objectives

This research achieved the following objectives:-

i. To identify sources of software complexity and productivity factors in the area of web application customization, reusability and productivity improvement.

ii. To design and develop a new OOAF to produce customizable and reusable software components that accelerate web application development productivity.

iii. To evaluate web application development productivity through software metrics.

1.6 Significance of the Study

This research has led to develop a OO application framework (OOAF) for developing web applications. This research is significant due to—Firstly, it can expose whether OOAF is applicable to develop web application. Secondly, explore new strategies for development productivity. Finally, data that were collected can be used for further studies for measuring software productivity through software metrics.

At the end, this research has fulfilled the current needs by reducing complexities, development efforts and improving productivity through OOAF. These attributes are extremely valuable and required for an optimized business process to satisfy customer and business.

1.7 Research Scopes and Delimitations

To solve the problems of this research, the scope of the research need to be defined so that this research becomes not too wide, more focus and go to right direction. This research limited to OOAF and it will not evaluate whether suggested OOAF is suitable or not in economical point of view. However, it has provided a complete development procedure for web engineers to implement OO-based web application as well as detail description of how to apply customization and reuse it using the proposed application toolkit.

At the end of the research, a complete evaluation has been conducted on the proposed OOAF and it emphasized on reuse design, coding and customization to discover pros and cons of the OOAF.

1.8 Research Outcome

This research involves multiple-disciplinary research including object-oriented framework, component-based software engineering (CBSE), web engineering, software productivity and customization. The contribution of this research covered as follows:

i. Design and develop an OOAF to produce customizable and reusable software components that accelerate web application development.

ii. Increase web development productivity and minimize development efforts by reusability and easier customization.

1.9 Thesis Outline

The content of the thesis is organized into six chapters. Chapter 1 has briefly drawn the overview of the research background, motivation, problem statement, objectives, research contribution, limitation and outcome. Chapter 2 covered literature review on the area of the research. It gave theoretical analysis on existing work and OO application framework that have been carried out. Chapter 3 explained the research approach used in this study. It described how this research was conducted, and systematic steps to guide readers in the process of completing the research. Chapter 4 showed analysis and design of the OOAF. Chapter 5 showed data collection and the result of productivity measurement. Chapter 6 summarized the research and provided recommendations for future research on productivity improvement.

1.10 Summary

There has significant progress been made on software reuse and customization. One of the most important issues is how to make best use of reusable and easily customizable web application system. Another is better representation technique for software artifacts. This chapter addresses background of the application framework, relevant problems and proposed a solution. There is a clear need for much more empirical work on reuse and customization. Research is needed to be explored more on identify and validate measures of reusability and customization.

CHAPTER TWO
LITERATURE REVIEW

Object-Oriented Technology (OOT) becomes more important due to it is a key for developing OO application. It promises greater productivity, lower development efforts, easier customization and higher reusability. However, researchers report that this promise is difficult to accomplish. This chapter described framework concept, key problems on why framework promises failed, productivity factors, previous related works and productivity development strategy through OOAF.

2.1 Framework Concept

In an OO software environment, a framework is a "*reusable software component, including reuse of analysis and design*" (Stoev & Dimov, 2008). According to Wallace and Bruce (2011); Fayad, Hamza and Yi Chen (2005) - "*a framework is more than a class hierarchy*". It depends on Hollywood Principle: - "*Don't call us, we'll call you*". It says that web developers handle it by applying inheritance, polymorphism or generalization methods so that developers spend fewer efforts in coding and spend more efforts on the business specific problems. An application could be implemented flexibly and within shortest time-frame through framework.

2.1.1 Application Framework

An application framework also known as "Toolkit" that allows for the creation of application. It consists of framework used by software engineers that provides a fundamental structure to support development of an application for a specific environment. An application framework or toolkit acts as a tool to supply the structure and templates for constructing an application. It becomes popular with the rise of GUI. Web engineers and developers found that to create a user interface (UI) with less effort application framework proved to be a good solution (Liang & Shimomura, 2009). It provides a standard framework with underlying pre-defined code structure. The intention of designing application framework is to lessen the issues faced entire software development life cycle (SDLC). Web engineers usually use OOP techniques to implement framework, whereby unique parts of application framework can simply inherit from pre-existing classes in the framework. The advantages of using application framework includes extensibility, simplicity, easier customization, code and design reuse. These advantages lead to lower cost of development, reduction of errors, reduce web development effort, increase quality and rapid application development.

2.1.2 Object-Oriented Application Framework

Object-Oriented Application Framework (OOAF) is set of libraries. It is designed to help web engineers to solve problems and build applications. The aim is to improve the overhead related with common activities in SDLC. According to Williams, Szyperski, and Wittenberg (2012), OO application framework organize and interconnect software components, generates runtime structure and manages execution of the application.

Object-oriented application framework or OO toolkit acts as a tool to supply the structure and templates for constructing an application. By using OO techniques while implementing the framework, pre-existing classes can be used to build the application easily. The primary benefits of OOAF are componentization, extensibility for customization, reusability, modularity and inversion of control.

According to Carlos and Pedro (2002) - "*OO application framework is a reusable, semi-complete application, which produces custom applications and collaborates to carry out a set of responsibilities*". OOAF provides reusability technique by utilizing the domain knowledge and experiences. It supports to avoid recreating common solutions to the application, decrease development time, cost and improve web application quality such as design pattern. Design pattern recurring solution for design and development problems.

Object-oriented application framework typically provides core functionality and underlying pre-define code structure to most of the application like session management, security, caching, interface template and data persistence. By using an appropriate framework, web engineers can save countless of hours. This is achieved through reuse design and code that share across different modules of an application (Riehle & Thomas, 2005).

Table 2.1 shows comparison between existing frameworks for developing web applications. From the current literature, numerous research works has done in the area of software customization, reusability and productivity improvement of web application development. Every single of them is dealing with dissimilar concern, concept and point of view (Malavolta, 2010). It is agreed that universal framework cannot exist due to different domains.

Table 2.1: Existing Web Application Framework Comparison

Project	Lang.	Ajax	Toolkit	Security	Design Reuse	Customize Module	Form Validation	DB Migration
Struts	Java	Yes	No		No	Hard	Yes	
Spring	Java	Yes	No	Acegi	No	Hard	Bean Validation	
JSF	Java	Yes	Yes	Yes	No	Moderate	integration	
GWT	JScript	Yes	Yes	Yes	No	Hard	Bean Validation	via Java
ZK	Java, Zuml	jQuery	No	Spring	No	Hard	Client, Server	Hibernate, Spring
ASP. NET	ASP	Yes	Yes	ASP	No	Moderate	plug-ins	Entity
Catalyst	Perl	REST, JSON	No	External	Html Template	Hard	Html Handler	
CakePHP	PHP	jQuery	No	ACL	No	Moderate	Yes	Yes
Joomla	PHP	Mootool	No	Yes	No	Moderate	Yes	No
Zend	PHP	Yes	No	ACL	No	Moderate	Yes	Yes

(Source: http://www.wikipedia.org/web_application_toolkit.htm)

2.2 Factors Effecting Productivity

Software productivity measure or define is a complex process. Most of the software productivity studies are inadequate and misleading. Productivity development should focus not only on the efficient development but also should emphasize the quality and value of application developed (Trendowicz & Munch, 2009). There are several factors that impact the clarification of productivity. For example, if the product output has defects and need to rework or bugs fix, it will decrease the productivity.

Most of the researchers related to web application development productivity focused on the study of an individual developer's efficiency. In addition, for large project development team often works with new tools, client participation, requirements and faced developer turnover, unclear goals, complexity, communication between team members that affect software productivity (Premraj, Kitchenham, Shepperd & Forselius, 2005; Sudhakar, Farooq & Patnaik, 2012). Thus, team size and team activities also important factor for project success. Research shows those strong skills team has higher productivity whereby weak skills team has lower productivity overall assigned tasks (Nwelih & Amadin, 2008). Additionally, complexity raises the team size and it reduces the productivity (Hernández-López et al., 2011). Their research also showed that productivity varies from C2C, B2B, working environment and software development process. A company with different business sector could accomplish different level of productivity.

2.3 Productivity Development Concept

Software productivity improvement is a critical part of the software engineering process. It is not hypothetical. In economical point of view, productivity defines as the ratio of produced goods for labor consumed to producing it (Nwelih & Amadin, 2008; Jones, 1996; Erne, 2011). From this assumption, software development productivity is the ratio between amount of the effort and expenditures of producing the software product. Productivity in its simplest form is product output divided by the efforts input (Wagner & Ruhe, 2008; Maxwell, 2001; Boehm, 1999).

$$Development\ Productivity = \frac{output}{input}$$

The output variable can be function point (FP) or lines of code (LOC) and the input variable can be person days. For example, lines of code per unit time (SLOC/h) or some variant of function points per unit time (FP/h). In that sense, the IEEE defines the productivity as the relationship of an output and its corresponding input primitive. Moreover, it is sometimes also known as efficiency.

According to Nwelih and Amadin (2008); and Erne (2011) explained the definition as software productivity consist of people and complexities of both software. It can be measured by dividing cost of software development efforts with software size.

$$Development\ Productivity = \frac{size\ (output)}{efforts\ (input)}$$

2.3.1 Input Variables

It is desirable to differentiate the developer according to their skills. To avoid simple effort calculation, it is necessary to include other input factors such as management effort, materials used, tangible/intangible variables.

2.3.2 Output Variables

Traditionally, the outputs of software development are measured by function points (FP) or lines of source code (LOC). These two measures are highly correlated, but FP approach takes the complexity into account. To incorporate both the quantity and quality aspects, we proposed considering code reuse, complexity, functionality and length. Reuse is measured by the estimated percentage of reused object points; complexity is measured using the big-O notation, functionality by the function points count and length by taking into consideration then density of comments (Nwelih & Amadin, 2008; Erne, 2011).

2.4 Related Works

It became clear that a number of different peoples and groups have performed similar research. Their findings, argument, conclusions, considerations and recommendation have been study in this research.

Object-oriented application framework designing and developing is complex. Presented work in this chapter can be considered only a portion of the overall work. There are quite a number of application frameworks such as Microsoft Foundation Classes (MFC), IBM Small Talk, End Point, Fox Toolkit, Java Dynamic Hyper Text Markup Language (JDHTML), Spring, Qt and Java Server Face (JSF).

According to Hneif and Lee (2011) introduced challenging problems in OO application development by showing a framework study on manufacturing application. Their research intended to provide evidence to support reuse. The research revealed that this domain is complex, dynamic and large. They implemented OOAF with OOT but evalution was not discuss. Several limitations must be considered in interpreting in Hneif and Lee study findings. We agree that- "*framework-based software engineering is the idea of constructing software system based on the integration of resuable components rater than developing softaware from scratch*".

Riehle and Thomas (2005) publication on framework design is more clear and complete. He introduced role modeling for OOAF design that indicates a substantial enhancement over current practices. Three case studies used to validate how role modeling performs in real-life environment. Each case study compares with traditional framework and role modeling framework design. However, each of the cases works from a different perspective. Their study combined the strengths of class-based modeling with role modeling while leaving out their weaknesses. Thus, it is an evolutionary of current methods that preserves existing investments. Finally, it is first comprehensive method for framework design.

Weiss and Heidenbluth (2012) introduced demand-driven software customization. This customization only focuses on customer's needs. It required empirically analyzes the benefit of the various adoptions from customer's point of view. They have conducted few surveys and presented a large study that deals with general questions of customization and analyzed the starting point of software customization. The survey points out that especially customization options, which adapt the functionality, increase the usability and enable controls are great importance for future software implementation. Hence, their results enable competitive advantages by implementing customization options that meet customer needs.

One interesting finding in our study contradicts one of the literatures. According to Lapouchnian (2011) presented requirements-driven approach for applying customizable and adaptive technique to the application. Requirement models used to capture the problems unpredictability, leading to application design that support similar functionality. This can be customized on the basis of user preferences at application deployment time. It can also be used at runtime to support customization if the running application is considered to be unacceptable. The contributions of Lapouchnian's study included systematical design of a framework. Three corresponding design views are: configurationally, behavioral and architectural view. The framework is also applied to business process management (BPM) customization.

There are wide ranges of literatures about software productivity trying to concentrate on market demands within shorter time and at the same time maintain higher quality products. However, there are still lots of unsolved issues. Indeed, there is no ultimate solution that can solve all of these issues related with development productivity. Eme (2011) presented how tools can be useful and when to use in software process. According to Boehm et al., (2009) contributed strategies on quality management by organizing development team. Another study demonstrated a model for motivation that is really worthy to read (Chiang & Mookerjee, 2004); others suggested incentive and reward strategy (Furtado, Aquido & Meira, 2009; Zhuge, 2008).

With respect to productivity measurement and modeling Premraj, Twala, Forselius and Mair (2004) pointed that reuse components should not be included. Their study mentioned that to construct a good productivity measure, size must be related to effort. In addition, COTS software projects come from four dissimilar facts. First, it needs to be considered alternative approach of measuring productivity. It is emphasis on problem of defining productivity measures. Secondly, it builds simple effort estimation techniques to improve productivity. Sentas, Angelis, Stamelos and Bleris (2005) have given more complex approach, which uses ordinal regression to access productivity and reliability of the given software products. Thirdly, it analyzes development practices and software productivity by benchmarking or making international comparison. Finally, it discovers the most important factors that improve software development productivity.

Paiva, Barbosa, Lima and Albuquerque (2010) conducted a study on same data set to construct productivity benchmark. The study found that, the most important variables are company and business sector. They investigated dataset in more depth. They divide the dataset into distinct business sector and analyzed it. Each business sector was identified into a variable, which manipulating productivity. Finally, the variable used to build productivity benchmark equations. It is agreed that productivity level differs between company-to-company and domain-to-domain.

In summary, it has to be self-proclaimed that the current application framework research and development is still so far from being conclusive. Further studies must be anticipated for better web application customization must be implemented and improve web application development productivity.

2.5 Summary

Web application development productivity can be enhanced by reuse design, code, minimizing rework through adopting comprehensive development standards and practices. This chapter described overview of OO framework, productivity and customization concepts. Finally, this chapter described factors effecting productivity and related works.

CHAPTER THREE
RESEARCH APPROACH

Object-oriented framework based web engineering is the idea of constructing applications based on reusable components rather than developing from scratch. The primary purpose of this research is to reduce web application development efforts in order to improve productivity by introducing OOAF. Therefore, this chapter identified the necessary steps, theoretical study and evaluates benchmarks that support the proposed OOAF.

3.1 Research Approach

The research approach of this research is divided into five steps as shown in Figure 3.1 and detail research activities shown in Appendix A. Figure 3.1 also shows which research approach has been taken to accomplish research objectives.

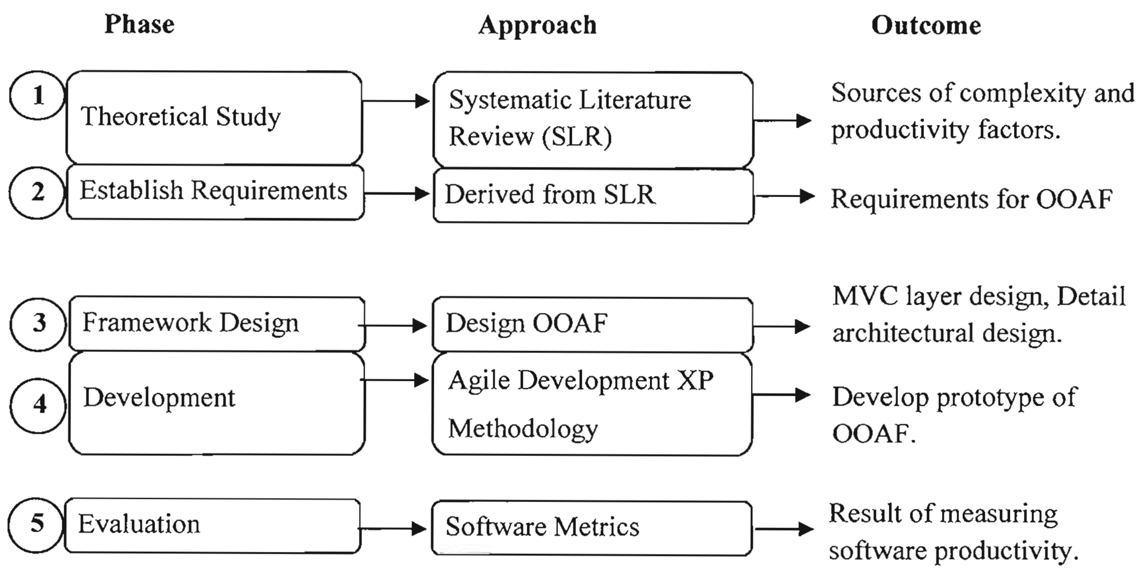

Phase
Approach
Outcome
1
Theoretical Study
Systematic Literature Review (SLR)
Sources of complexity and productivity factors.
2
Establish Requirements
Derived from SLR
Requirements for OOAF
3
Framework Design
Design OOAF
MVC layer design, Detail architectural design.
4
Development
Agile Development XP Methodology
Develop prototype of OOAF.
5
Evaluation
Software Metrics
Result of measuring software productivity.

Figure 3.1: Research Approach

3.1.1 Theoretical Study

In the initial stage, board range of study was required in many different aspects of current application frameworks and architectures. This includes of identifying new features of web development, past and current trends. To get good foundation of knowledge and understanding of the requirements, development tools, strength and weaknesses of the web application this research review previous literature, collect ideas, issues and articles related to OOAF.

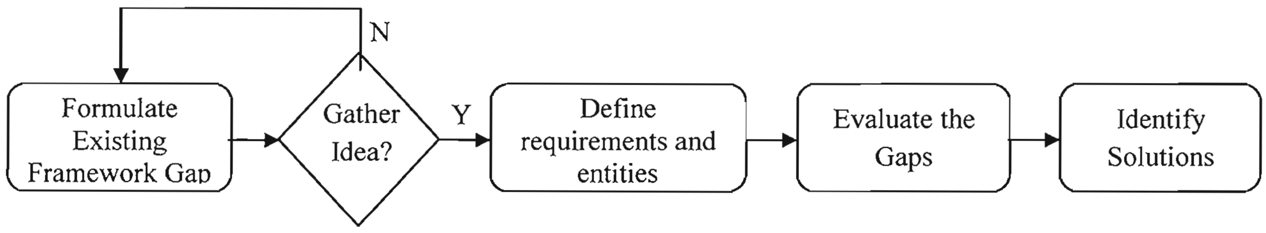

Formulate Existing Framework Gap
Gather Idea?
N
Y
Define requirements and entities
Evaluate the Gaps
Identify Solutions

Figure 3.2: Process of Theoretical Framework

Figure 3.2 shows the process of theoretical framework. This phase adopted from systematic literature review (SLR) by Kitchenham (2004) and divided into two subsections—theoretical study and empirical study. The theoretical study focused on the current problems in OOAF, processes and techniques to design extensible, customizable, reusable components and barriers in productivity improvement. The empirical study focused on case study to examine OOAF in the actual working environment. Finally, estimated the development productivity and software metrics.

This research analyzed concepts of web application design, common practices and assessment benchmarks. In addition, thoughtful study on the existing OOAF, issues interrelated to application framework design strategies; customization and reusability techniques that simplify to implement an OOAF has carried out. This research found that the current OOAF do not provide design reuse with underlying structure generation. Hence, it is vital to realize obstacles and current practices by web developers. Thus, the aim is to undertake all of the mentioned complications by presenting visual object sharing technique in OOAF. This research showed that the proposed technique reduced domain complexity, provide design reuse, easier customization and increase productivity of web application development.

3.1.1.1 Identify Source of Complexity

Some of the problems related to OO framework have been described in Chapter 1. Research showed that, complexity arises due to unclear, complex and instability requirements. These led to increase number of classes, complex object collaboration between objects, complex relationship among objects and classes. The predictable result is crucial and fielded with unexpected errors.

Most of the projects failed due to complex requirements, poor design, difficult to learn tools used, hard to develop and inadequate documentation (Muller et al., 2009). Thus, it is worthy to discover the problems in OOAF and search for new solutions. Previous OO frameworks are developed by traditional object-oriented programming (OOP) which do not share design reuse and easier customization techniques.

Current OOP are good at describing of an object, but it is static interface. It is difficult for developers to learn the two-way pattern of a framework by reading it. As a replacement, developers discuss with experts and read other documents. Design pattern is one of the approaches to improve design (Christiaans & Almendra, 2010).

Another approach is to elaborate the interactions between objects and its constraints. This research presented better way to express and develop OOAF.

3.1.1.2 Identify Productivity Factors

There is various factor impacted web application development productivity. Table 3.1 identified the most important factors manipulating productivity. We analyzed the factors that have minimum five citations and list down under three categories. Whether study become not ambiguous we have taken most cited factors from Table 3.1. To narrow down the research scope we skipped one factor named as "*Process*" and selected important factors which have minimum 20 citations in product category.

Table 3.1: *Software Productivity factors analysis*

Category	**Factors**	**Number of Citations**			
		1990-1999	**2000-2009**	**2010-Jul'13**	**Total**
Product	Reuse, Customization	7	8	6	21
	Software Size	8	6	6	20
	Software Complexity	6	9	6	21
	Team's Skills	4	11	3	18
People	Turnover	0	5	3	8
	Team Communication	1	5	3	9
	Motivation	1	5	2	8
	Management Quality	2	4	2	8
Project	Process	5	15	7	27
	Team Size	4	10	2	16

Client Participation	1	3	2	6
Tools, Methods used	6	11	8	25
Dev. Environment	3	5	3	11
Programming Language	4	8	5	17
Requirement Stability	6	8	4	18

We have chosen 4 factors that effects software productivity improvement—reuse/ customization technique, software size, complexity and tools/ methods used.

3.1.2 Establish Requirements

It is not small undertaking to design and develop an OO application framework. There are many considerations to be addressed. Effective requirements gathering and design can mean the difference between success and failure. Few questions needed to answer to identify requirements as shown in Table 3.2.

Table 3.2: *Requirements Analysis*

No	Question	Answer
1	What kind of OOAF needs to build?	For building web applications
2	What kind of data source required?	File-based and Rational Database
3	What languages are suitable to design and develop OOAF?	.NET used for design User Interface and Java 7 used in framework level.
4	What kind of user interface required?	JSP user interface with HTML5

3.1.3 Design OOAF

Web application design is hard. The design of reusable, customizable and extensible OOAF is even harder. The proposed application framework design is Model-View-Controller (MVC) pattern as shown in Figure 3.3. Design stage divided into two sections: MVC layer-based design and detail architectural design. Layer-based design outlined high-level strategy and road map for the solution of the problem. MVC Layer-based design proposed for modularity, reusability and extensibility.

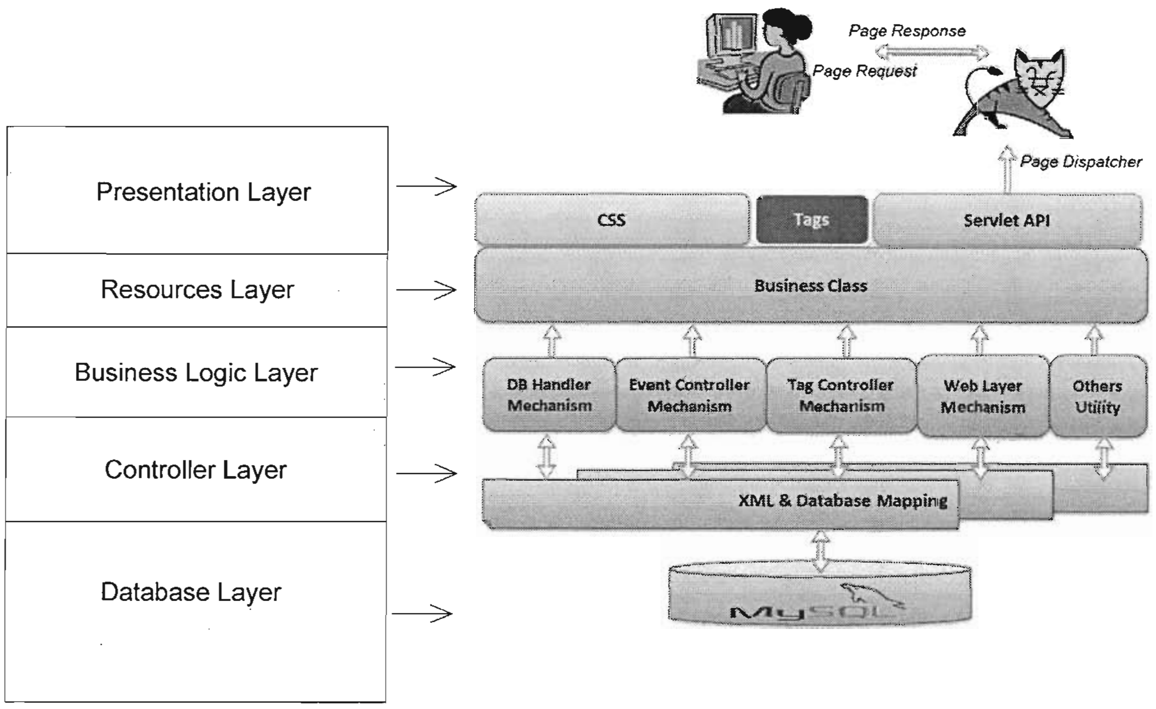


Figure 3.3: Layer-based (MVC) Conceptual Architectural Design

Detail architectural design identified objects, models it and described in the target implementation language as shown in Chapter 4. It also constructed core functionalities of the OOAF as shown in Figure 4.7 and Appendix B. It identified problems; solutions of the given problem and solutions suitable to the problem.

This study applied design pattern to solve design problems and blueprint of the proposed application framework for developing web applications. Design of user interface, object interaction, relationship and visual object sharing technique applied to produce robust structure or desire behavior. In design phase, effort directed to visual object sharing technique which generates underlying structure, event handler, and codes so that web engineers will able to focus on implementing value-added services.

3.1.3.1 Design Proposed Productivity Improvement Technique

There are number of ways to improve reusability such as inheritance, polymorphism and delegation. There have been few efforts to develop web applications through visual objects. Therefore, this research aim is to enhance web application customization and reusability via visual object sharing technique. In our proposed technique, "visual object" is the core element which plays the key role of customization and reusability of web application design. In this sense, the proposed OOAF (sometimes refer as a toolkit) is object-centric rather than application-centric.

This research presented visual object sharing technique for customization and design reuse to accelerate software development productivity as shown Figure 3.4. Visual object sharing (VOS) technique plays as a fundamental role. It facilitates customization, extensibility and reusability. The problem comes up when developed application become large in size. It is very difficult to work with such framework due to many classes, line of codes and complex relationship between objects.

"*Hierarchical representation is one of the most effective techniques*". It enables the complex and large classes divided into multiple and smaller classes. It leads to grow number of classes as a result web application become complex. Therefore, visual object sharing technique used to minimize the current mentioned problems. The proposed visual object sharing technique defines how a visual object shares same behavior by changing relationship between them and how object collaborates with other objects in a class. By visual object sharing, each class has instance of an object and a single class cannot provide this technique. Thus, proposed visual object sharing technique can deliver an endless flexibility to decrease web application development complexity as well as software crisis. By integrating hierarchical representation with visual object sharing technique, customizable and extensible OO application framework can be achieved.

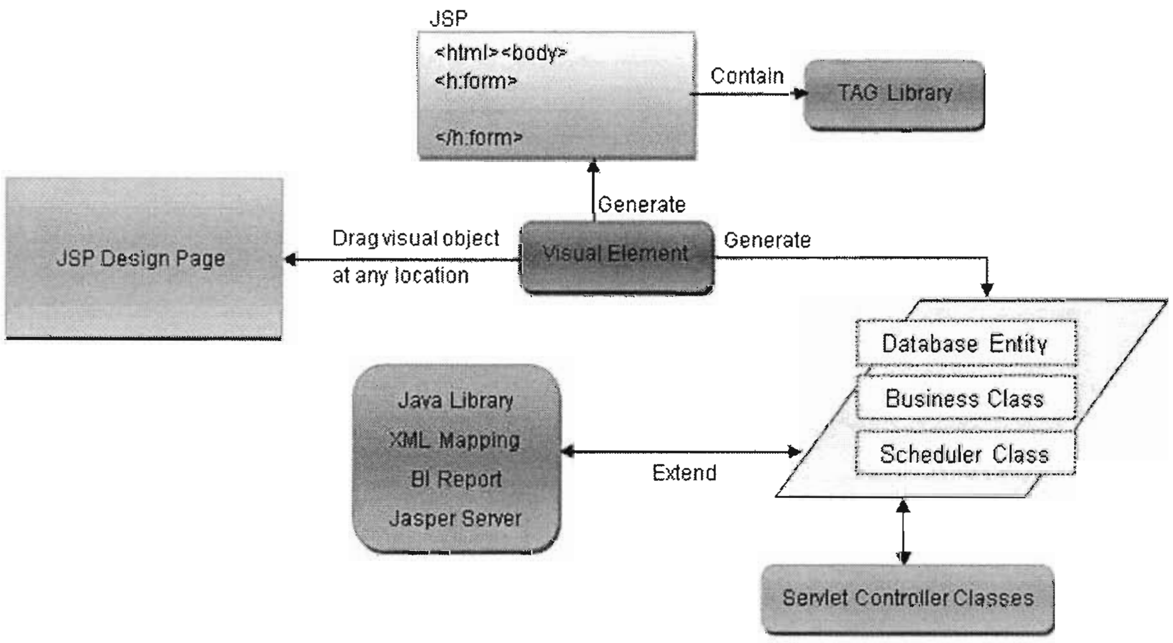


Figure 3.4: Visual Object Sharing Technique

The concept of visual object is based on GUI elements to construct web applications. Each visual object has its own features, template key that handle directly and separately. It is visualize using tag library, AJAX and set of functionality defined in meta-data and XSLT schema. It generates all underlying data and application codes by meta-data based on the conditions and parameters selected by the web engineer. In visual object sharing technique, a number of modules share one visual object by calling template key id. Template key establish relationships between JSP/Servlet and other java classes. Without visual object sharing, a module own independent components that appear as the same code or elements in different modules in the application. Additionally, each visual object has its own inversion of control and it can be invoked outside of the class. It is also responsible for instantiating and disposing method invocations from the caller. Web engineers can create, customize, reuse and organize visual objects according to users/customers specific need.

The technique for customization of a visual object or its feature requires some necessary steps. Firstly, each visual object to be customized initially requires defining its behavior and events followed by parameterization and objecting delegation. It does not require any naming convention. The technique allows this task to be performed using a graphical user interface. Secondly, template key used to search visual object from resource repository and visualize or wrap it with existing visual object so that reuse can be built over the same set of object resources. Thirdly, it automatically generates all underlying data, application code and XML mapping based on the changes made by the web developer. The application code generates against the data stored in objects defined by meta-data. These features reduce coding complexity, development time and increases reusability as well as productivity and software quality.

The proposed OOAF adequately addresses layered-based presentation technique needed for flexible customization and integrates existing tools or application modules into a single environment. This technique provides separation of business logic from presentation layer and database layer. Furthermore, this technique also has the ability to extract visual object and related resources from existing application modules without source code modification. This leads to extensibility, flexible integration and a customization environment.

To develop a web module using proposed OOAF web developer should drag and drop objects into working window only. Rest of the jobs such as HTML coding, parameters, business class, controller class, database connectivity, and event handlers will be created by proposed framework itself. Each object has its own properties with an identification key. New object behavior could define by web developers or they can customize object properties. This could be applied in the similar way to other web elements or objects. Figure 3.4 shows visual object sharing technique on how it interacts with other elements such as Java classes, JSP file, XML and other objects. In addition, each module will abstract its objects. Once an instance of a module creates then it generates instance of object. In addition, if an instance of a module destroys, object instance will destroy instantly.

3.1.4 Development

Development of OO application framework is even harder than design. New software components, objects, classes followed the architectural strategy of the OOAF. This stage prepared inventive OOAF interface design concept as shown in Figure 3.5.

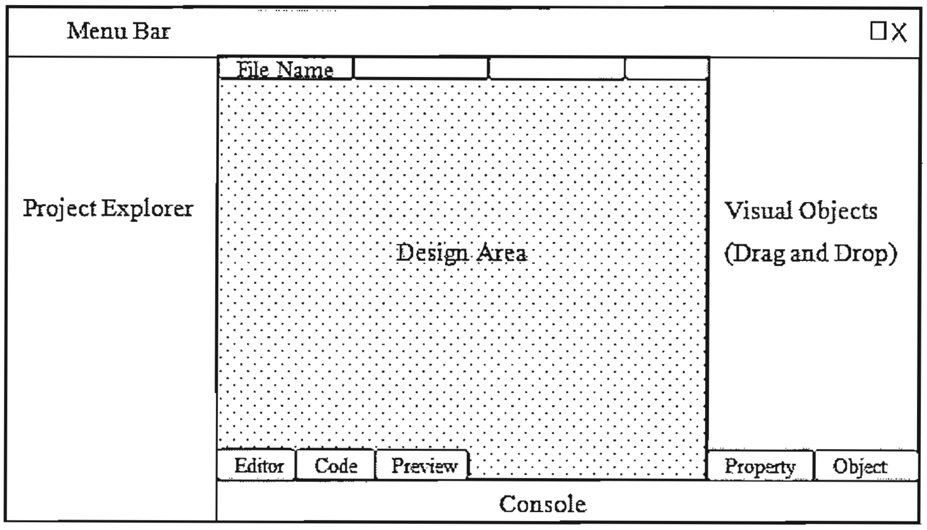


Figure 3.5: Preliminary Interface Design of OOAF

This research adopted agile development methodology (Extreme Programming) for faster requirements gathering, design and development. The combination of Java and .NET used to develop the proposed OOAF. VS2010 as .NET IDE, "Eclipse" as Java IDE, JDK-1.7, and .NET 4.0 as GUI designer used in development environment. We focused on using OOP because it has essential features for developing OOAF.

3.1.5 Evaluation

The aim of this stage was to estimate productivity of the OOAF toolkit in the actual working environment. A key element for any estimation or measurement is to know what is needed to be measured because without this element it is impossible to establish a measurement (Hernández-López, Colomo-Palacios, García-Crespo & Cabezas-Isla, 2011; Tangen, 2005). We considered software size, function complexity, effort required and process to build the software.

Knowledge gained from the development phase merged into a case study and requested web engineers to develop modules and observed its performance on individual. In this stage, study focused on novice and professional web engineers who asked to design and develop "Automatic Jobsheet Processing and Invoice Management System" through the proposed OOAF and traditional method. Then questionnaire was provided to collect data from two different projects. The questionnaire attached in Appendix C. One project was developed by proposed OOAF, and another project developed by structure method. Both of the projects were similar in terms of domain, product size, programming language, development environment and system requirements. Chapter 5 will discussed more on data collection and data analysis.

3.2 Productivity Measurement Concept

Software products are intangible. Thus, measuring productivity is one of the great challenges. In software engineering, productivity is usually measured using a product size ratio between the efforts required to produce the product (Andreou & Tziakouris, 2007). For example, lines of code per unit time (SLOC/t) or some variant of function points per unit time (FP/t).

According to Kitchenham and Mendes (2004) certain measures of productivity are imperfect; they are still used because of the ease to take them. Parallel to its definition, there are factors (Wagner & Ruhe, 2008) that affect the measurement result and dependent on the environment and organization so we analyzed it case-by-case basis (Paiva, Barbosa, Lima & Albuquerque, 2010). At the developer's level, some of these factors are: motivation, commitment, experience, skills and work environment. However, the literature focuses primarily on other factors such as experience, programming language, requirements stability, complexity, reuse and size (Gummesson, 1992). In addition, before any attempt to measure web developer's productivity, it was necessary to identified what should be captured in this measure. So with a definition of productivity is possible to determine this need. In this research, we have selected as most relevant factors which have minimum of 20 citations from product category as shown in Table 3.1.

3.2.1 Measurement Procedure

Productivity improvement is major concern in software industry. Software development teams gradually reduce their production costs and increase productivity. To improve productivity, it is necessary to understand how to measure it.

There are numerous works from different perspectives on how to measure software productivity. Few of them introduce new metrics and few present basic concepts (Bandi, Vaishnavi & Turk, 2003). Therefore, in order to have a systematic and effective analysis, it is necessary to setup our own assumptions in clear fashion. This research made an assumption as shown in Eq. (1-4) in defining the relationship of size, productivity and effort. Chapter 5 will discussed more about it.

This research showed the actual man days used as a multiplicative function of the primary production correspondence. The software development productivity is generally measured as shown in Eq. (1)—

$$Productivity = \frac{size\ (output)}{efforts\ (input)} \quad (1)$$

Where, *size* is considered in total LOC and *efforts* are considered in man-days. In the COCOMO model, the complexity as a driver during the assessment of human effort, rating value from 0.70 to 1.65 (Boehm & Ricardo, 2008). This rating represents the complexity degree from "*very low*" to "*very high*". The relationship between productivity and complexity is shown in Eq. (2)—

$$productivity = \frac{size}{a \times size^{E}} \quad (2)$$

In Eq. (2), productivity is a function of size with adjustment from exponent factor *E*, where *E* is complexity and α is coefficient of regression, which collects the cost driver (Jorgensen, Indahl & Sjoberg, 2003). Based on regression analysis, many factor prediction models have a format as shown in Eq. (3) where, *K* is the coefficient product of some other factors and α is the regression coefficient.

$$effort = K \times (complexity)^{a}$$

Taking Eq. (3) into Eq. (1), the following equation Eq. (4) is obtained—

$$productivity = Size/(K \times complexity^{a}) \quad (4)$$

Equation (2) and (4) indicate that productivity declines exponential when complexity increases. On the other hand, Gill and Kemerer (2001) proposed a significant negative linear relationship between productivity and cyclical complexity. All of these results verified the statement that "*reducing complexity creates greater efficiency and results in higher productivity*".

3.3 Summary

A board range of study is required to get good foundation of knowledge, research approach, strength and weakness of the web application development. This chapter described the research approach adopted from various methodologies for necessary steps, processes and evaluation criteria that support this research. This chapter also identifies productivity factors, elaborated proposed productivity improvement technique and productivity measurement procedure.

CHAPTER FOUR
APPLICATION DESIGN

4.1 Introduction

The process of building interactive software system is difficult task, subtle and complex. Generally, it comprises tens of thousands of lines and hard to ensure that it could provide effective solution. It is agreed that design of an OO application framework could meet this challenge. Many approaches exist that try to deal with the arising issues. Some address what kind of software is to be developed and how it looks like which includes human factors, user interface issues. Other approaches address how system can be built which includes advance techniques, visual composition and component ware. Only few approaches try to integrate and bridge gaps between analysis, design and implementation techniques. This is the best approach that involves reusable software artifacts in each SDLC phases. Thus, this research design and implemented such an OOAF to speed-up commercial web applications development and customization.

In this chapter, we have presented layer-based architecture to describe proposed OOAF. This research also indicates that layer-based architecture and separation of concerns approach reduces application design complexity to develop domain-oriented OO system.

4.2 Design OO Application Framework

In a simplest term, design is creative activity to solving problems. This required planning, novelty, rigor, decision-making and it cannot be solved in a single stroke

(Cross, Christiaans & Dorst, 2007; Baker & Van der Hoke, 2009). Our proposed and developed OOAF named as– “Zaman Toolkit (Z)” for creating extensible and reusable modules for building web applications. The design stage divided into two sections: architectural design and detail application design. Architectural design outlined high-level strategy and road map for the solution of the problem. Detail design identified objects, described in the target implementation language and constructed core functionalities of the OOAF. Application design which prepared inventive OO toolkit design concept has discussed in this section.

4.3 MVC Layers Design

The design goal of the OOAF is layer-based (MVC) with reusable, interactive and robust UI elements. The architecture of the OOAF divided into five (5) layers as presentation layer, resources layer, business logic layer, controller layer, database layer as shown in Figure 3.3. Each layer has a specific function and set of modeling activities. The following section presented high-level design of the each of the layers.

4.3.1 Presentation Layer

Proposed OOAF provides a set of UI elements and design templates that encapsulated visual presentation, event handling, code generation. Visual objects define in presentation layer with associate component model, server-site events and object stateful data. They form the foundation of custom web application development.

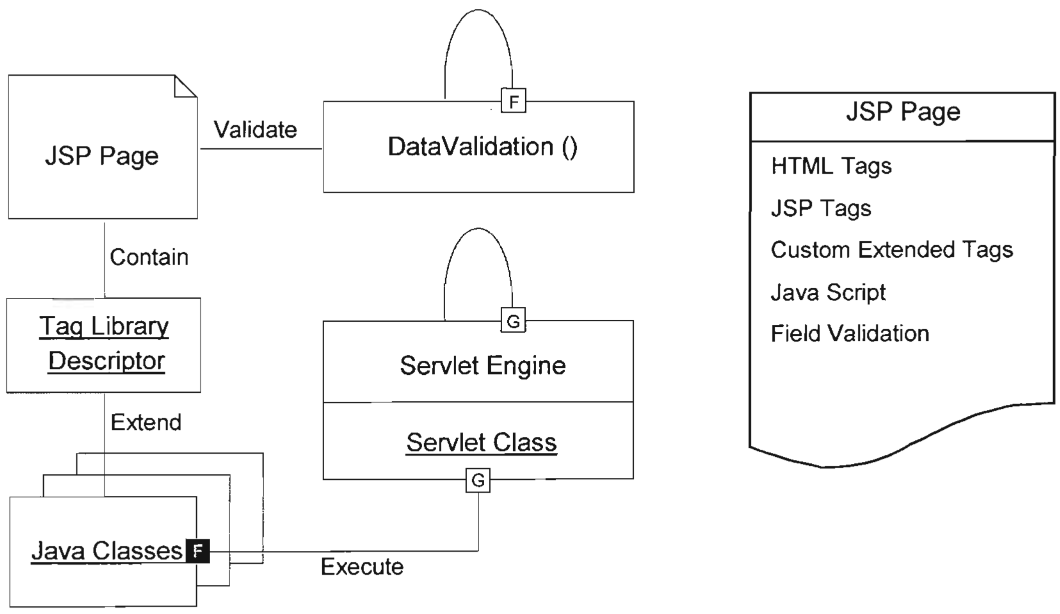


Figure 4.1: Presentation Layer Architecture

Figure 4.1 shows presentation layer architecture that used Java Server Pages (JSP) API from Sun Microsystems. In general, OOAF write web pages using a mixture of HTML, JSP tags, Java classes, java script and own custom markup tags. These pages are served, rendered from a servlet engine or JSP-compatible web application server. In addition, if an invalid input is given, presentation layer has data validation technique before it sent to the controller.

The presentation layer displays the visual objects or UI elements in JSP pages. The presentation layer calls an object instance and renders its attributes to the JSP pages. When object instance changes, presentation layer automatically redraws the visual object and transforms the object state. When a page forwarding request comes from servlet engine it's destroy all object instances for current page. Page forward mapping should be either in RDBMS or a XML format. It provides "*loose coupling*" between visual objects and presentation layer that provides faster web development in OO environment.

4.3.2 Resources Layer

On top of the presentation layer, resources layer is responsible for both sharing of resources and the enforcement of visual object access based on the available resources. Since access to resources is most often highly security-critical, the resources layer has to provide important visual object properties.

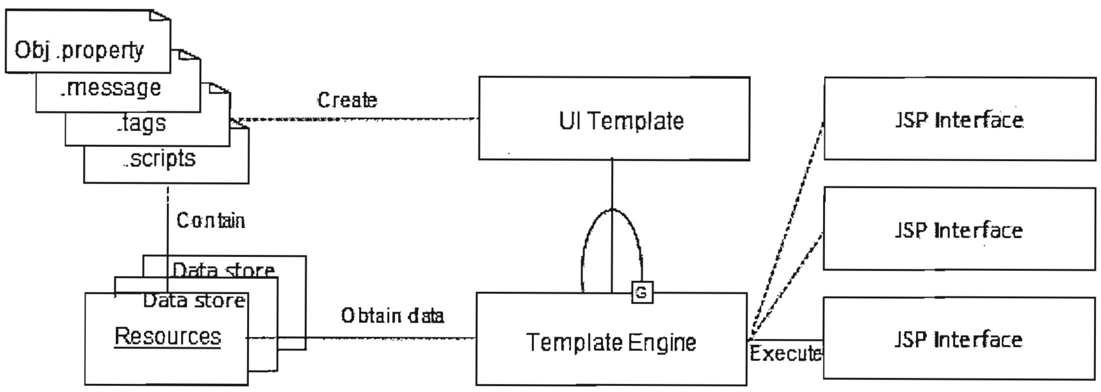


Figure 4.2: Resources Layer Architecture

The resources layer also contains template engine or service agent which is liable for communicating other services to retrieve resources as shown in Figure 4.2. The resources layer should encapsulate all the code that deals with external resources (such as databases or other services), without leaking any implementation detail to higher layers.

4.3.3 Business Logic Layer

Model-View-Controller (MVC) based system develops and implement only in Java/Servlet. If UI change, then OOAF generates relevant HTML code, meta-data, XML files, event handling and underlying structure.

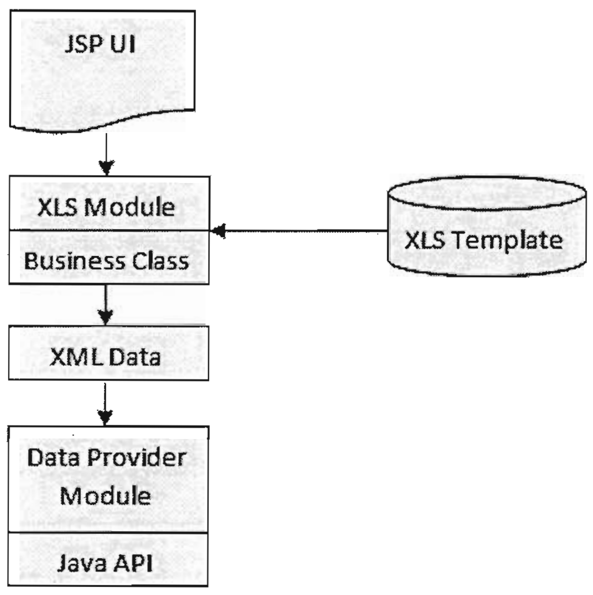


Figure 4.3: Business Logic Layer Architecture

Similarly, if the business logic changes, then only visual object's event codes need to change. An application develops with MVC methodology should easier to customize web application. Figure 4.3 shows business logic layer of proposed OOAF. Logic layer concerns with business logic only. A web designer who knows nothing about Java can concentrate on the look and feel of the UI layout. Whereby, web developers can focus on the core logic and Java Beans. A change to the user interface only concern changes to the relevant JSP only.

4.3.4 Controller Layer

In OOAF architectural design, system flow is intermediated by controller layer. It delegates the request to the controller technique or handler. A controller receives input from presentation layer, initiate visual object's instance and executes action as shown in Figure 4.4. If an invalid input is sent to the handler, the object instance notifies to handler to forward an error and notifies to re-input.

The controller layer consists of java classes and interfaces that provide the runtime environment for the components used within an OOAF. It is responsible for handling application level events such as switching, dispatching events to UI modules, authentication, state management, error processing, log on and log off.

Proposed OOAF controller technique has a specialized view since it is integrated with visual object sharing technique. It is actually one or more visual UI elements in JSP and therefore model can inject what it should display. The controller could add necessary parameterization so that the JSP event controller can observe the input.

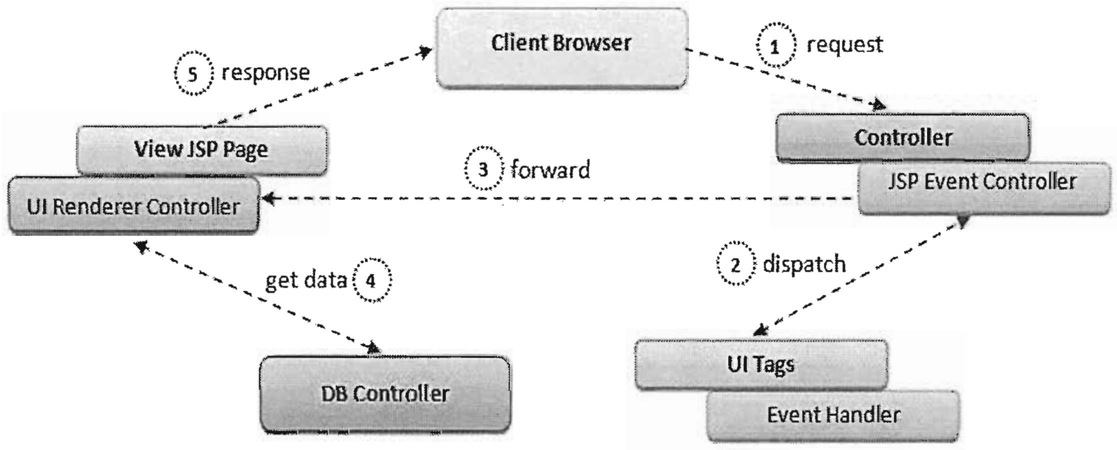


Figure 4.4: Controller Layer Architecture

As shown in Figure 4.4, controller adopts the request from client browser or presentation layer and dispatches UI elements or event handlers. UI element represents object state or business logic. It notifies any observer when data changes. Every request passes via controller who retrieves visual object values. The visual object sends the result back to the controller. The controller will take the result and forward to JSP.

4.3.5 Database Layer

The database layer provides and stores information. It is very crucial and internal layer that is protected from user's view. There are no directly accesses to database from the upper layers. Its access is routed through the database layer as shown in Figure 4.5.

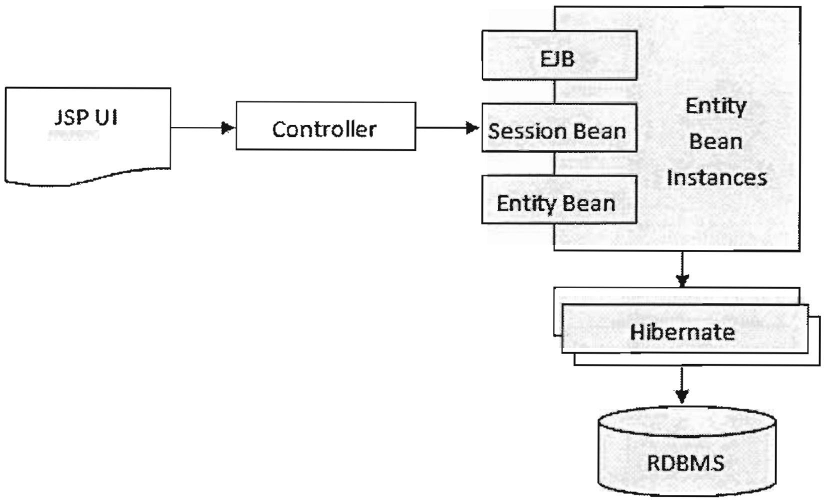


Figure 4.5: Database Layer Architecture

All query subjects in this layer are imported from the data source. Because these query subjects point directly to the database, actions such as join, relationship, or renaming of query items cannot be done. Future model changes caused by schema changes are made in this layer. All other layers are unaffected by the schema changes. The ability to leverage code generation tools is one of the keys to a flexible architecture. Proposed OOAF comes with this feature. It produced Java sources, object-relational mapping and configuration XML files.

4.4 Detail Architectural Design

After high-level design we focused on detail design which is the process of defining the lower-level components, modules and interfaces. We verified detailed designs in design reviews and level-by-level. We used OO design method to define module processing and divided into four (4) categories- conceptual architecture, module architecture, class architecture and presentation architecture. Figure 4.6 shows the relationships among these architectural layers.

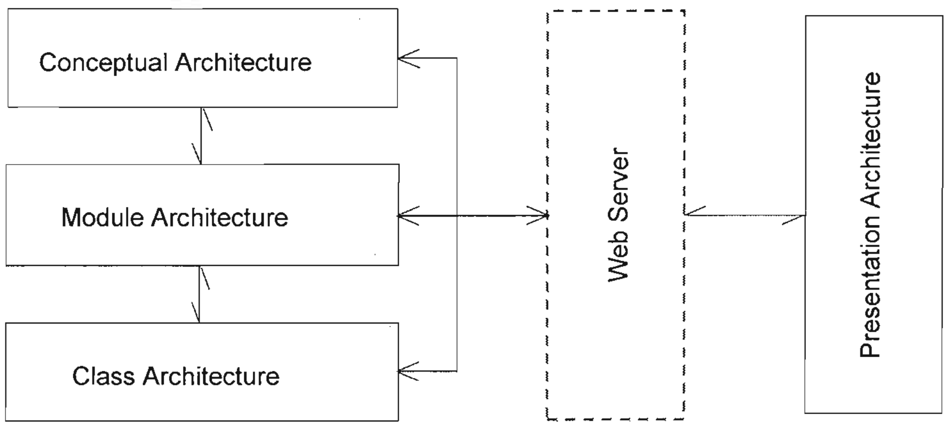


Figure 4.6: Relationship among the Architecture Layers

4.4.1 Conceptual Architecture Design

The conceptual architecture is very high-level structure of an application. It describes the major design elements and relationships among them. Figure 4.7 demonstrated what developed OOAF can generate, how the data acquisition is connected, underlying structure and java classes and mapping files.

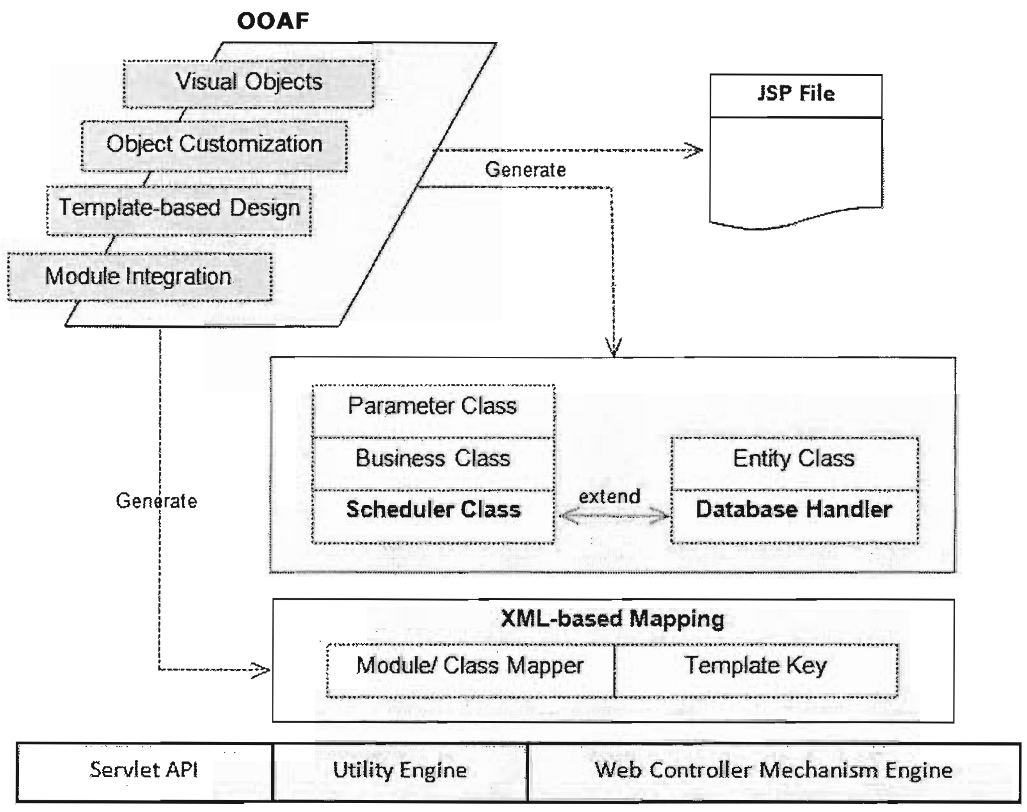
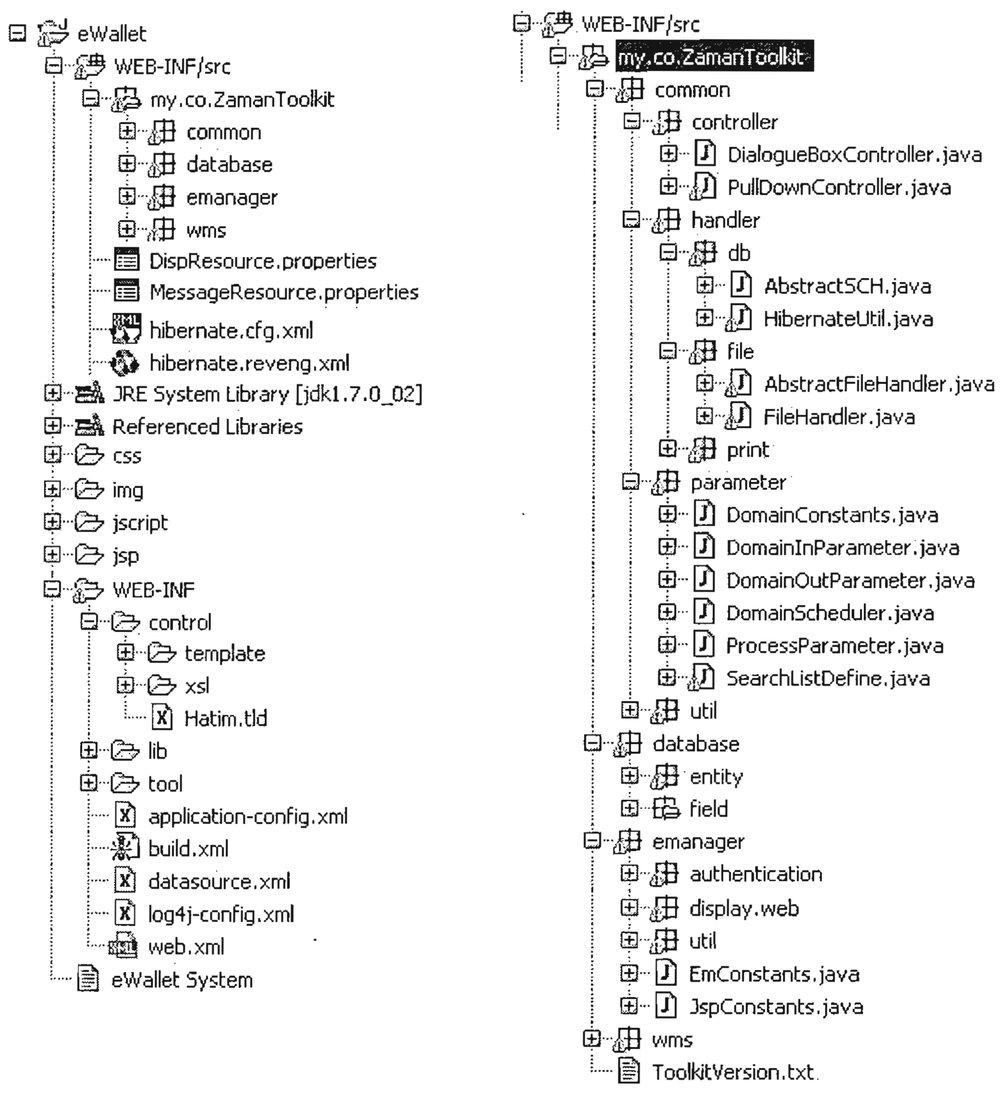


Figure 4.7: Conceptual Architecture Design

Figure 4.7 showed that developed OOAF contains UI elements, visual objects, template design and object customization technique. Upon interface design completion, OOAF automatically generates JSP file, java classes with event handlers, business class, entity class, database handler, relevant data elements and XML mapping files.

4.4.2 Module Architecture Design

The OOAF's module architecture encompasses into two structures- functional decomposition and UI layer. Functional decomposition captures the way system is logically decomposed into subsystem, modules, file management, handler, controller, parameter, resources and abstract program units as shown in Figure 4.8. It captures visual object interrelationships in terms of exported and imported interface.



```
eWallet
  WEB-INF/src
    my.co.ZamanToolkit
      common
      database
      emanager
      wms
    DispResource.properties
    MessageResource.properties
    hibernate.cfg.xml
    hibernate.reveng.xml
  JRE System Library [jdk1.7.0_02]
  Referenced Libraries
  css
  img
  jscript
  jsp
  WEB-INF
    control
      template
      xsl
      Hatim.tld
    lib
    tool
    application-config.xml
    build.xml
    datasource.xml
    log4j-config.xml
    web.xml
  eWallet System
```

```
WEB-INF/src
  my.co.ZamanToolkit
    common
      controller
        DialogueBoxController.java
        PullDownController.java
      handler
        db
          AbstractSCH.java
          HibernateUtil.java
        file
          AbstractFileHandler.java
          FileHandler.java
        print
      parameter
        DomainConstants.java
        DomainInParameter.java
        DomainOutParameter.java
        DomainScheduler.java
        ProcessParameter.java
        SearchListDefine.java
      util
    database
      entity
      field
    emanager
      authentication
      display.web
      util
      EmConstants.java
      JspConstants.java
    wms
    ToolkitVersion.txt
```

Figure 4.8: Module Architecture of OOAF Application

4.4.3 Class Architecture Design

The class architecture is used to organize the source code into packages, directories and libraries. This facilitates system building, installation, configuration management and minimizes dependencies among sub-projects to enforce import/export constraints specified in the module architecture. Figure 4.9 shows the sample class diagram of visual element (FreeTextBox) and JSP Page.

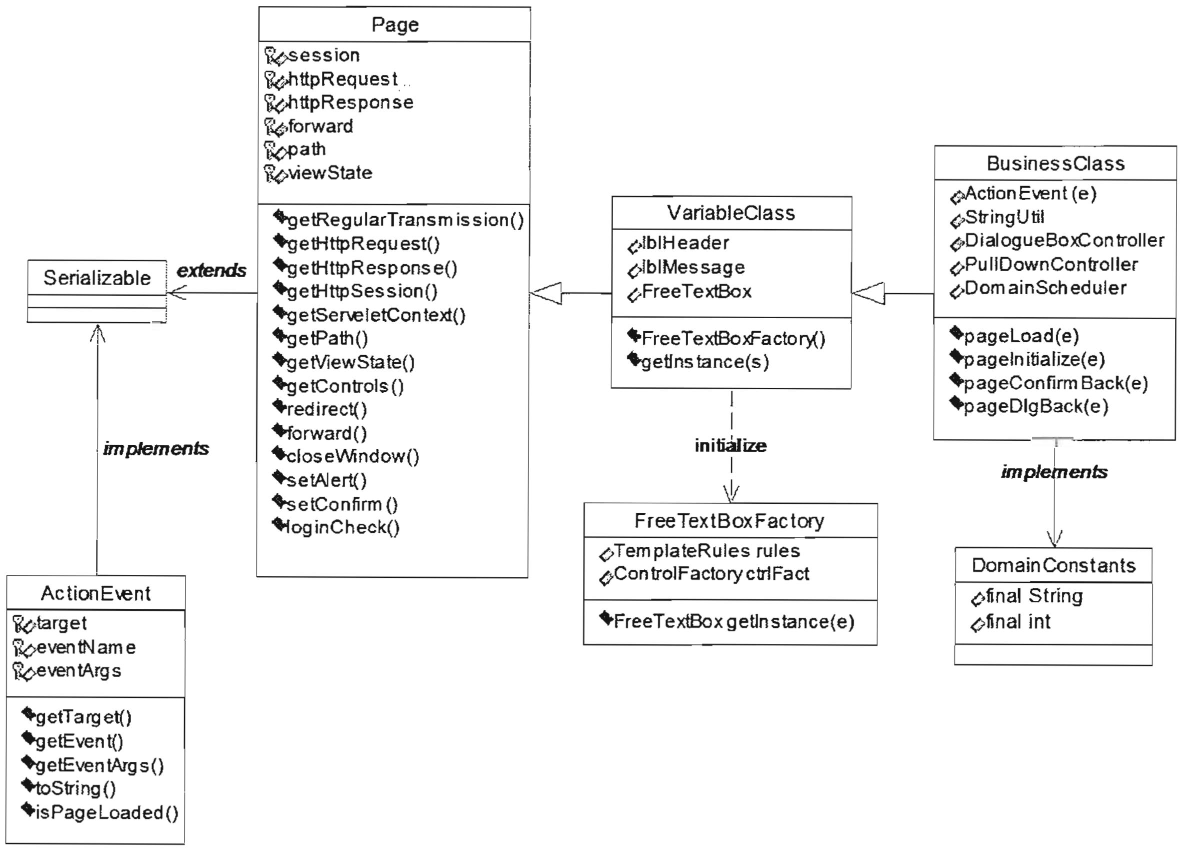
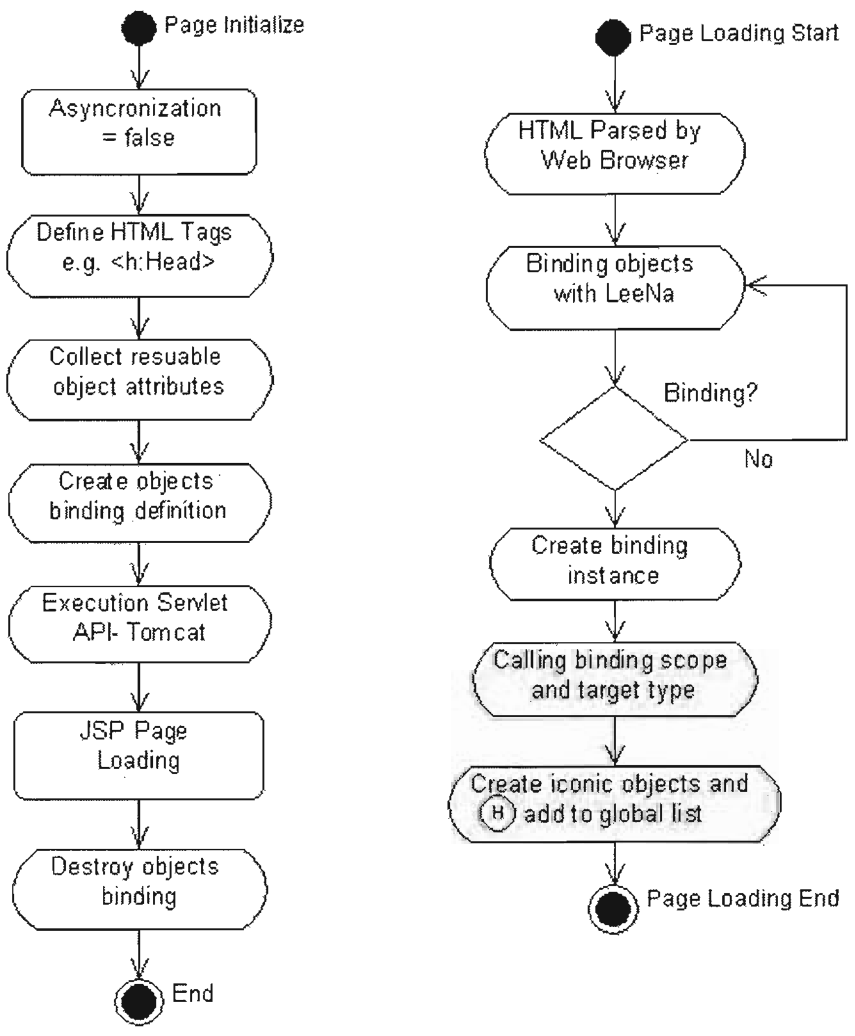


Figure 4.9: Class Diagram of a Visual Object

As we can see that "Page" class is superclass which extends Serializable class and implements ActionEvent class. Page class is parent and controls HttpRequest, HttpResponse, HttpSession, viewstate, forward, redirect, alert and many more. Visual object initialize "Factory" class to display the object into JSP through XML.

4.4.4 Presentation Architecture Design

The presentation architecture is responsible for binding object behaviors to page elements, display visual objects into JSP and destroys object binding. This allows a dynamic and loose association of bindings on a particular page. This approach enables resource reusability.



Page Initialize
Asyncronization = false
Define HTML Tags e.g. <h:Head>
Collect resuable object attributes
Create objects binding definition
Execution Servlet API- Tomcat
JSP Page Loading
Destroy objects binding
End
Page Loading Start
HTML Parsed by Web Browser
Binding objects with LeeNa
Binding?
No
Create binding instance
Calling binding scope and target type
Create iconic objects and add to global list
H
Page Loading End

Figure 4.10: Visual Object Presentation Architecture

Figure 4.10 shows steps which occur when a JSP page request and loaded. Initially OOAF will define page header <h:Head>, body of the page to create object binding definition and servlet execution to load the JSP page. When page loaded into browser, object binds with OOAF where binding declaration used XML and create binding instance. Bindings can be declared as a group or individually. Initial bindings on a page typically established using binding element. In this instance, the CSS, JavaScript class, events bound to the HTML element on the web page and display visual objects. This is application level bindings that are managing the relationship and communication for other sub-bindings on the JSP page. A binding is implemented by a JavaScript class. Standard OO design concepts of inheritance, public, private, function and constructor are supported by binding definition. Figure 4.11 shows how object binds into JSP page.

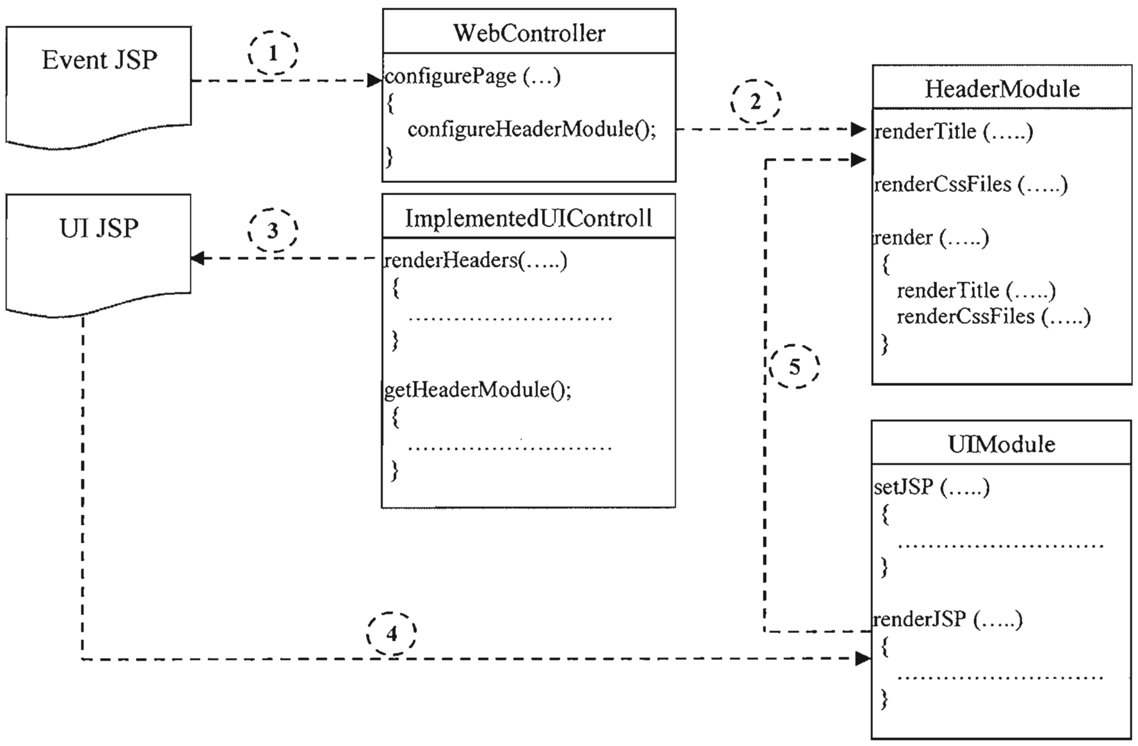
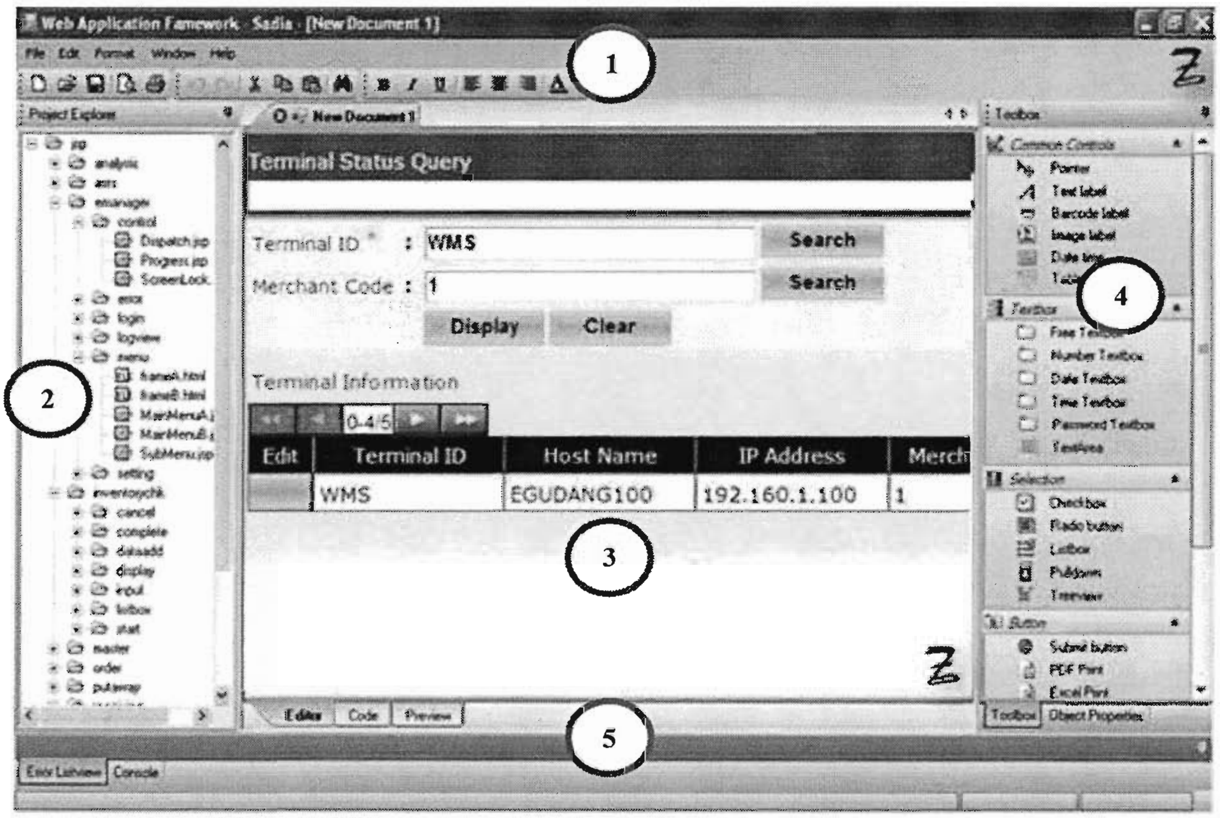


Figure 4.11: Binding XML-based JSP Page Rendering

4.5 Interface Design

Figure 4.12 shows the proposed OOAF toolkit prototype. The prototype divided into five parts-(1) Menu and Toolbar- where contains execution command and other features, (2) Project explorer- where shows project's JSP files, (3) Working area- where visual object can be drug to design user interface/ create JSP, (4) Toolbox- visual objects or user interface elements and view object properties (5) Console area- where shows relevant error and guided message.



Terminal Status Query

Terminal ID * : WMS [Search]

Merchant Code : 1 [Search]

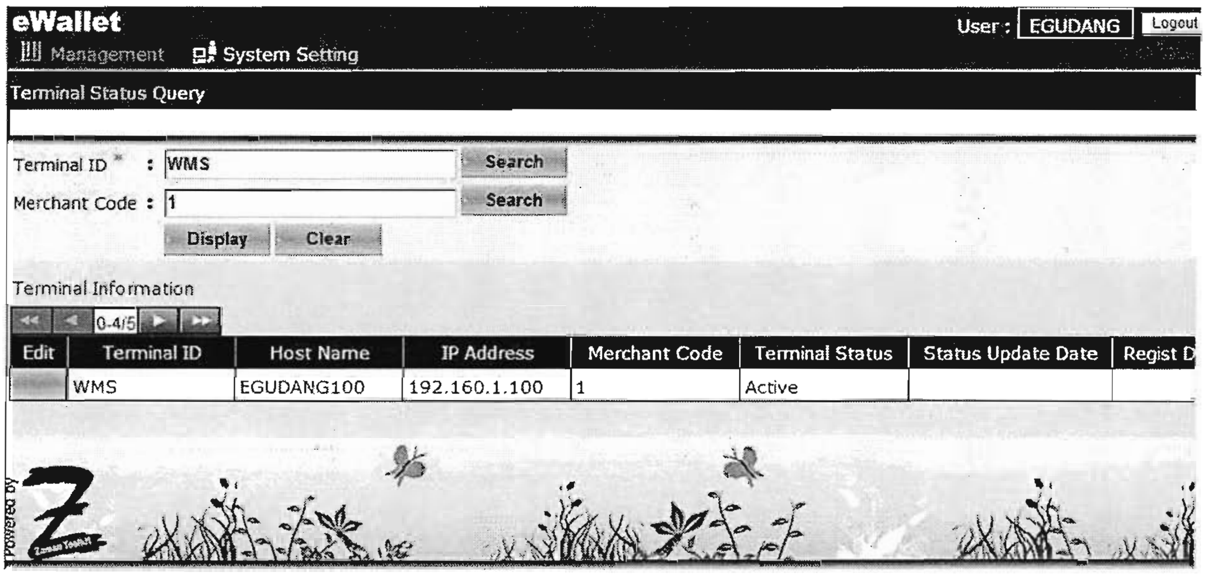
[Display] [Clear]

Terminal Information

Edit	Terminal ID	Host Name	IP Address	Merch
	WMS	EGUDANG100	192.160.1.100	1

Figure 4.12: Interface Design of OOAF

Web developers require to drag visual elements from no. (4) to working area at no. (3). OOAF will visualize the drag object and developer need to named it. Upon completion UI design, developers require to click "Generate" button from toolbar which will produce all related codes and JSP file. Figure 4.13 shows complete and final user interface of a module generated from proposed OOAF.



Terminal ID * : WMS
Merchant Code : 1

Edit	Terminal ID	Host Name	IP Address	Merchant Code	Terminal Status	Status Update Date	Regist D
	WMS	EGUDANG100	192.160.1.100	1	Active		

Figure 4.13: A Complete UI Design form

4.6 Summary

Object-oriented approach is core function to increase web application development productivity. In this chapter, we have presented layer-based architecture of the proposed OOAF, high-level and low-level design. We used visual object sharing technique to design UI, customize and reuse it. This approach reduces web application design complexity to develop domain-oriented OO system and improve developer productivity.

CHAPTER FIVE
DATA ANALYSIS & RESULT

Measuring the software development productivity is a complex process. Different organization or researcher has different opinion on software productivity about the concept of what is produced. Thus, proper method or process is required for measuring productivity for each of the organization. This research used software metrics to measure productivity from two projects. This chapter described data collection, software size, productivity calculation and result of the statistical analysis. It is state that the result in this case study cannot be generalized as it is only be implemented at two projects, but the result are promising.

5.1 Factors Effecting Case Study

There are several factors effect case study. Most of them are—management issues, personal capabilities, experiences, level of skills and communication. Research shows that these variables take affect about 80% of total variance in productivity improvement. In order to estimate accurate software productivity we considered that both of the projects were supervised by similar set of skills, experiences and personnel capability.

Both of the projects were also non-embedded, similar domain, requirements, level of complexity, design, functionality, development environment, programming language, team size and similar capabilities in team members.

5.2 Data Collection

The data presented in this research were collected from Opensources Technology Sdn Bhd, Selangor, Malaysia. The company is "*MSC Status*" certified and hires about 50 staffs, where nearly 85% directly involved in software development.

Data were collected from company's metrics database on two software projects. The data was documented by member of each development team leader and used to controlling and managing projects. The type of application has developed is "Automatic Jobsheet Processing and Invoice Management System" using 3GL. The core programming language was Java. Project 1 (SWT-P1) was developed by proposed OOAF, while project 2 (ENT-P2) was developed in traditional method.

Measuring productivity in OOAF is not as simple as it sounds due to OOAF use code reuse techniques. Sometimes developers reuse whole program without modification and often modify a module to some extent. However, we accounted reused code, distinguished a module with one modified line from a module with 100 modified lines. Thus, this research considered the notion of reuse on an ordinal scale as shown Table 5.1.

Table 5.1: *Code Reuse Classification*

Type	**Description**
New Code	None of the code comes from previously constructed class.
Reuse	Code reused without any changes or less than 25% of lines of code in a class were modified.
Modified	More than 30% of lines of code in a class were modified.

The total software size comprised of all new, reuse and modified codes added together as shown in Table 5.2. It is required to measure the modification on existing classes. The change size metrics used to count effective SLOC modified or adapted from existing class.

Table 5.2: *Actual Project Size Data*

Project	**Method**	**Code Size [KSLOC]**			
		New	**Reuse**	**Modified**	**Total KSLOC**
P1	OOAF	75	275	15	365
P2	Traditional	107	202	20	329

Table 5.3: *Actual Efforts Data*

Project	**Effort [MD]**					
	Design	**Develop**	**Test**	**Rework**	**All Others**	**Total Effort**
P1	15	45	7	8	10	85
P2	20	55	10	12	10	107

The duration of the development was documented in man-day (MD) from project started date to deployment date. The total efforts are consist of the sum of the man days comprising analysis, design, development, test and all others days as shown in Table 5.3. All others days comprises time spent in discover defects, configuration, learning new tools and supervision of the project.

Both of the projects SWT-P1 and ENT-P2 were both non-embedded, partial real-time with same functionalities. They were implemented on commercial servers. The complexity was medium on their formal QA level. The teams were formed based on experiences and level of skills. The team that developed using OOAF was well-trained in this discipline and tools used.

Data was collected through questionnaire and it was distributed among the participants and used software metrics to measure development productivity. The participants were top level executive management, project managers and system analysts.

The following section provides the results of the data analysis conducted on two software projects namely—SWT-P1 and ENT-P2. Firstly, data analysis was performed to establish the software productivity metrics. Secondly, a descriptive analysis of the dataset was reported. Thirdly, correlation coefficient was tested to validate the result.

5.3 Measuring Productivity through Software Metrics

Basically, Productivity is calculated as size of the software product divided by cost spent to develop it. As we classified codes into reuse, new and modified so our productivity formula as shown Eq. (5).

$$Total\ Productivity = \sum_{i=1}^{n} \frac{r_i + n_i + m_i + c_i}{\sum e} \tag{5}$$

Where, r_i is reuse codes, n_i is new codes, m_i is modified codes, c_i is complexity and $\sum$e is total efforts required. According to the "*IEEE Standard for Software Productivity Metrics*", defined steps of productivity improvement consist—adjusting software, backfiring and measuring. The following section will elaborate it.

5.3.1 Adjusting Software Size

Total size of software consists of new, reuse and modified SLOC. Reuse codes (RSLOC) are adapted or pre-existing code without any changes, modified codes (MSLOC) are 30% of lines of code in a class were changed and new codes (NSLOC) are code newly developed. MSLOC were transformed to equivalent source lines of code (ESLOC) using "*Adaption Adjustment Factor (AAF)*". The factor captures other effort required to design, develop and test the previous version of code. The ESLOC measurement equation is shown in Eq. (6).

$$\begin{aligned} ESLOC &= NSLOC + RSLOC + MSLOC\ x\ AAF \\ AAF &= (0.4xDM) + (0.3xCM) + (0.3xWM) \end{aligned} \tag{6}$$

Where, *DM* is design modified, *CM* is source code modified and *WM* is work modified (after discover defects and test).

5.3.2 Backfiring

According to "*IEEE Standard for Software Productivity Metrics*" defined- "*the productivity computed for a structured design project developed in a third-generation language (3GL) shall not be directly compared with the productivity of an OOSAD project developed in fourth-generation language (4GL)*". Therefore, ESLOC transformed into "Unadjusted Function Point (UFP)" using Backfiring approach. Table 5.4 shows the conversion ratios used in this research.

Table 5.4: *Language Conversion Ratio*

Language	**ESLOC per UFP**
Assembly	320
C++	55
Java	53
Visual Basic	29
HTML	15

Unadjusted function point (UFP) equation as—

$$UFP = \sum_{i=1}^{n} (No. of\ items\ of\ variety\ x\ weight) \qquad (7)$$

5.3.3 Calculate Productivity

After ESLOC transformed into UFP, the final step is to calculate software productivity based on Eq. (5). Table 5.5 is shown productivity achieved by projects.

Table 5.5: *Total Project's Productivity*

Project	**Method**	**Productivity**		
		ESLOC/MD	**Backfiring**	**UFP/MD**
P1	OOAF	768	55	13.96
P2	Traditional	428	71	6.03

5.4 Descriptive Analysis

Descriptive analysis compared and differentiated the metrics of the two developed software products to answer specific questions related to software productivity. JMP Statistical Discovery Software, version 10.0.2 was used to assist in performing the analysis. The result showed that OOAF productivity is higher than traditional productivity as shown in Figure 5.1.

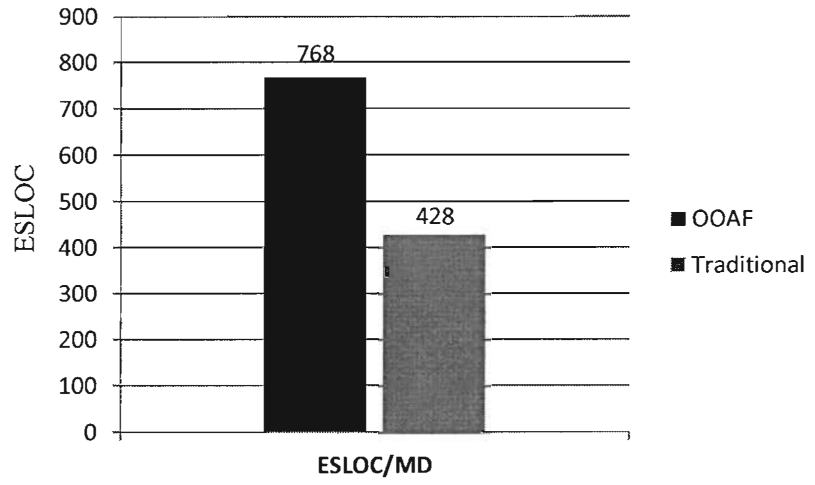


Figure 5.1: Productivity Comparison by ESLOC/ MD

Figure 5.2 shows the differences of adjusted productivity by OOAF versus traditional method, where adjusted productivity means ESLOC has converted into FP. Again, from the result we can say that the OOAF productivity is higher than traditional productivity.

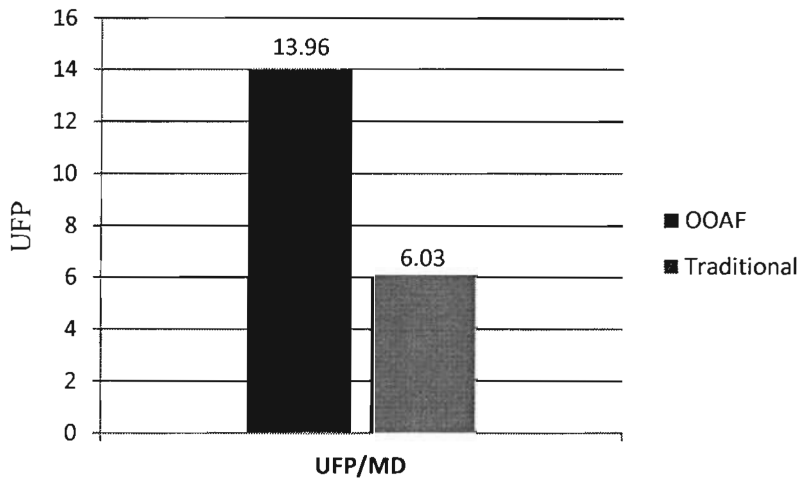


Figure 5.2: Productivity Comparison by Unadjusted Function Points/ Man-days

Figure 5.3 compares that total efforts spend to produce total source codes. The result shows that OOAF used less effort to produce more SLOC compare to traditional method. Again, we can say that OOAF productivity is higher than traditional productivity.

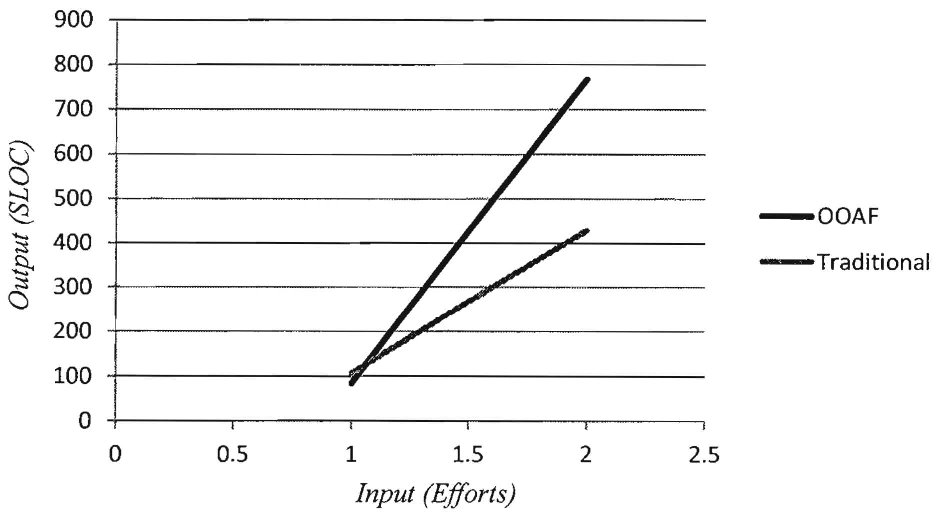


Figure 5.3: Productivity Comparison by Output versus Input

5.5 Correlation Test

Correlation test determined, whether there is validity issue with the study given that there may exist relationship between independent and control variables. This step divided into two sub-sections. Firstly, sub-section used correlation coefficient to measure the extent where OOAF method and control variables are related. Secondly, sub-section used P value to test whether there are relationships between OOAF and control variables.

5.5.1 Correlation Coefficient

The correlation coefficient ranges from negative (≥ -1) to positive ($\leq +1$) values. The larger absolute value is the stronger correlation. Table 5.6 shows the correlation coefficient between OOAF, control variables and productivity.

Table 5.6: Correlation Coefficient

	PP	*SIZE*	*AD*	*OOM*	*CPLX*	*PERC*	*RELY*	*PVOL*
PP	1.00							
SIZE	0.00	1.00						
AD	-0.62	-0.19	1.00					
OOM	0.50	0.22	0.01	1.00				
CPLX	-0.69	-0.01	0.82	0.06	1.00			
PREC	0.00	0.10	-0.01	0.06	-0.06	1.00		
RELY	0.00	0.47	-0.25	0.01	0.05	0.43	1.00	
PVOL	-0.10	0.01	0.23	0.08	0.03	0.75	0.36	1.00

"PP= Product Productivity, SIZE= Product Size, AD= Application Domain, OOM= Object-oriented Method, CPLX= Complexity, PERC= Personal Capability, RELY= Reliability, PVOL= Platform Volatility".

The criterion for acceptance is ≥ 0.45. The results shown in Table 5.6 indicated that correlation coefficient for the relationship between OOM and PP is strong (0.50).

That's means productivity gradually increases when OOAF is chosen over traditional method. In contrast, the correlation between AD is -0.62 and CPLX is -0.67. Both of them are strong and negative. This means that when application complexity increase, productivity tends to decrease.

5.5.2 P Values

The *P* values showed whether the correlation coefficient is different from 0. Coefficient of 0 indicates no linear relationship between independent and control variables. If the P value is ≤ α level, then a strong correlation exist between OOAF and any of the control variables. Table 5.7 shows the linear relationship between OOM, control variables and productivity.

Table 5.7: *Correlation of P Values*

	PP	*SIZE*	*AD*	*OOM*	*CPLX*	*PERC*	*RELY*	*PVOL*
PP								
SIZE	0.994							
AD	0.012	0.372						
OOM	0.020	0.445	1.000					
CPLX	0.001	0.961	0.000	1.000				
PREC	0.986	0.663	0.131	1.000	0.814	1.000		
RELY	0.986	0.035	0.475	1.000	0.843	0.061		
PVOL	0.689	0.965	0.057	1.000	0.914	0.000	0.117	

"PP= Product Productivity, SIZE= Product Size, AD= Application Domain, OOM= Object-oriented Method, CPLX= Complexity, PERC= Personal Capability, RELY= Reliability, PVOL= Platform Volatility".

Table 5.7 used α level of 0.05. The result showed that linear correlation between OOM and PP is 0.020. The value is ≥ 0.05, thus, there is a relation between object-oriented method and productivity. In contrast, correlation between reliability (RELY) and PP is 0.986 and ≥ 0.05. This means there is a relation between reliability and productivity. Result also showed that linear correlation between personnel capability (PERC) and productivity (PP) is 0.986. Meaning that there is also a relation between personnel capability and productivity.

5.6 Summary

This chapter presented software productivity measure and analysis that can be used when there are several size measures related to different aspects of a software product that are significantly related to effort. This research used JMP Statistical tool to analyze and validate whether OOAF can be productive. The result showed that OOAF method has a significant factor affecting productivity. OOAF can dramatically improve higher productivity over traditional methods. It is state that the result in this case study cannot be generalized as it is only be implemented at two projects, but the result are promising.

CHAPTER SIX
CONCLUSION

An object-oriented application framework (OOAF) was successfully designed and developed. The purpose of this research was to reduce development effort and increase productivity. As markets more competitive, continues productivity improvement become major concern in software industry. OOAF and reuse is an important aspect of software productivity when size is provided as input to effort and productivity model. This research demonstrated that it provides a breakthrough solution to software customization, design reuse and productivity improvement. It is fair to claim that this goal is achieved. This chapter summarized the key contributions, recommended and suggested some research directions to be pursued in the future.

6.1 Discussion

In this section we have answered the research questions. The research question I was—"What are the complexities and productivity factors to achieve reuse design and easier customization in OO-based web application development?". To answer this question, sources of software complexities have been identified in the section 3.1.1.1. Research showed that poor design, static interface, lack of clarification in requirements, complexity of the domain, complex relationship among objects, size of classes and proper documentation are root cause of software complexity.

Software productivity factors discussed in section 2.2 and section 3.1.1.2. It said that productivity varies on individual developer's efficiency, understanding requirements, complexity, communication between team members, team size, working environment, process, development tools, and methods. This research overcomes mentioned problems using proposed OOAF.

The research question II was—"How to design and develop OOAF that could support design reuse and existing web applications customization??". The answer provided in Chapter 4. To validate the research question data collection has been discussed in section 5.2 and descriptive analysis in section 5.4. This research showed that it is easier to customize, reuse design and develop web application from OOAF than traditional tools.

The research question III was—"How to evaluate the proposed OOAF, can it increase development productivity for web engineers and developers?". In Chapter 3 section 3.2.1 elaborated how software productivity assessment gone through this study. In Chapter 5 section 5.3 showed measuring software productivity through metrics. The results indicated that the project used OOAF is higher than the project used traditional tools. At the end, the result also validated using correlation test as shown in section 5.5.1.

6.2 Summary of Key Contributions

In this research, we have presented a novel OOAF which generally increased web development productivity and decrease development efforts for web engineers. It supports developer easier customize, reuse and flexible module integration without any code modification. There are number of contributions in this research. The key contributions of this research summarized as:-

Firstly, identify the main factors to improve development productivity. This research identified software complexity, size, reuse (code and design) and process as biggest obstacles in software productivity development.

Secondly, design and develop OOAF. This research successfully developed ample and comprehensive OOAF. Its simplicity enables to create new modules, easier customization and reuse design without source code modification.

Thirdly, OOAF provided "*separation of concerns*" among software architecture layers, extends reusability and customization. It supports underlying code generation, interaction between objects provided by OOAF. Our successful experience for "Automatic Jobsheet Processing and Invoice Management System" showed that OOAF provides a feasible solution to software industry.

Finally, there are difficulties to apply new tools in an organization where traditional tools, methods and processes are dominated. This research proved that OOAF and agile approaches are inherently better than traditional tools, methods and processes. Evidence about OOAF over traditional method in terms of productivity development was presented. The result indicated that software industry should consider OOAF method than traditional method if software productivity is a concern.

6.3 Future Research

There are number of ways for future research that are worth exploring. In the future, a similar but more focused study could be done. Several suggested areas of future work could be included as:-

This research does not cover more detail design of OOAF. A more detail design is necessary to make the concept of framework widely accepted. Also, more detail analysis is required in different domain to explore for better requirements, reusable and customization technique.

Tool support availability is an important part of any application framework. Effective tool integration with OOAF can increase more productivity and high quality products. In order to add existing plug-ins and tools further research works required. Some of them may include—mapping tools, better component selection tools or tools to evaluate models.

This research used visual object sharing technique to visualize an object. It could be great through if model done from Unified Modeling Language (UML) diagrams. Determining the feasibility of using requirements to generate business logic code remains a future research topic.

Economic view point is not directed to this research. It is suggested that future researchers should investigate the impact of IT from other perspectives such as intangible outputs, financial perspective, system performance, reliability and quality.

OOAF API is complex. Additional time is required to learn framework API which decrease overall development productivity or encountered many obstacles during development process. Once framework API is learned, following projects can be easier and faster to complete.

Finally, the quality improving technology such as design reuse, code generation and input validation check has been provided with OOAF. However, defect closure metrics, inspection checklists and module test plan need to be added

6.4 Conclusion

We believe that this research on OOAF begins to fulfill an important gap in web application development. Best practices for designing, developing, deploying and maintaining highly customizable systems need to be researched, discovered and documented. Our proposed OOAF offered the research community a vocabulary to discuss and describe software customization and reuse design to improve software development productivity.

We do believe that the need for customization that is feasible to build, possible to deploy and successful in meeting the needs of software industry to grow as time progresses. Finally, we hope that we have inspired others to research this topic and contribute to the knowledge of how to cope, manage and tackle these difficult challenges.

REFERENCES

Alvaro, A., Santana de Almeida, & Romero de Lemos (2010). “A Software Component Quality Framework”, ACM SIGSOFT, Software Engineering Notes, vol. 35, no. 1, pp. 1-17.

Andreou, A. S. & Tziakouris, M., (2007).A quality framework for developing and evaluating original software components In the Information & Software Technology. vol. 49, no. 2, pp. 122-141.

Bandi, R., Vaishnavi, V. & Turk, D. (2003). *“Predicting maintenance Performance using Object-Oriented Design Complexity Metrics*”, IEEE Transactions on Software Engineering, 29(1), pp. 77-78.

Baker, A., & van der Hoek, A. (2009). An Experience Repot on the Design and Delivery of Two New Software Design Courses, Fortieth ACM Technical Symposium on Computer Science Education, pp. 319-323.

Boehm, B.W, Supannika Koolmanojwong, Jo Ann Lane, Richard Turner: Principles for Successful Systems Engineering Procedia CS 8: 297-302 (2012)

Boehm, B.W, Sunita Chulani, June M. Verner, Bernard Wong: Seventh workshop on Software Quality ICSE Companion 2009: 449-450

Boehm, B.W, Ricardo Valerdi: Achievements and Challenges in Cocomo-Based Software Resource Estimation IEEE Software 25(5): 74-83 (2008).

Boehm, B.W. (1999). "Managing Software Productivity and Reuse," IEEE Computer, Vol. 32, No. 9, pp. 111-113, Sept. 1999.

Blackburn, J. D., Lapre, M. A., & Van Wassenhove, L. N. (2002) Brooks’ Law Revisited: Improving Software Productivity by Managing Complexity. Vanderbilt University Working paper 2002-85.

Cross, N., Christiaans, H., & Dorst, K. (2007). Design expertise amongst student designers, Journal of Art & Design Education Vol. 13, No. 1, pp. 39-56.

Chiang, R.I. and Mookerjee, V.S. (2004). "Improving software team productivity," Commun. ACM, pp. 89-93.

Cusumano, M., MacCormack, M.A., Kemerer, C.F. and Crandall, B. (2003). "Software development worldwide: The state of the practice," IEEE Software, vol. 20, pp. 28-34.

Carlos, J. & A. Pedro (2002). "*Domain Analysis of Object-oriented Frameworks in FrameDoc*". SEKE'02, Ischia, Italy. Journal of ACM, 1-58-113-556-4/02/0700, vol. 4, no. 2, pp. 27-33.

Clarke, S. & Walker, R.J. (2001). "Composition patterns: An approach to designing reusable aspects in ICSE 2001", International Conference on Software Engineering, IEEE Computer Society Press, pp. 5-14.

Christiaans, H., & Almendra, R. A. (2010). Accessing decision-making in software design. Design Studies, Vol. 31(6), pp. 641-662.

David, O., Ascough, J. C., Lloyda, W., Green, T. R., Rojas, K. W., Leavesleya, G. H., & Ahujac, L. R. (2012). A software engineering perspective on environmental modeling framework design: The Object Modeling System.ScienceDirect, Environmental Modelling & Software (2012), pp. 1-13.

Erne, R. (2011). "What is Productivity in Knowledge Work? - A Cross-industrial View", Journal of Universal Computer Science, vol. 17, no. 10, pp. 1367-1389.

Fayad, M.E., Hamza, S.H. & Yi Chen, (2005). "*A Framework for Developing Design Models with Analysis and Design Patterns*", Communication of IEEE, 0-7803-9093-8/05.

Frakes, W.B. & Kyo Kang (2005). "*Software Reuse Research: Status and Future*", IEEE Transactions on Software Engineering, vol. 31, no. 7, pp. 529-536.

Furtado, F., Aquino, G. and Meira, S. (2009). "Incentive Systems in Software Organizations," Software Engineering Advances International Conference, pp. 93-99.

Gummesson, E. (1992). "Quality dimensions: what to measure in service organizations", Advances in services marketing and management, T. A. Swartz, et al., eds., JAI Press, 1992, pp. 64-78.

Gill, G. and Kemerer, C. (2001). "Cyclomatic complexity density and software maintenance productivity", IEEE Trans. Software Engineering, vol. 17(12), pp. 1284-1288.

Hazem, M.H., Wassim, E.H., Dana, M., Marwan, D. & Faysal, F. (2010). "*An Extensible Software Framework for Building Vehicle to Vehicle Applications*", IWCMC'2010, ACM Press, 978-1-4503-0062-9/10/06.

Hernández-López, A., Colomo-Palacios, R., García-Crespo, Á. (2012). "Software Engineering Job Productivity: a systematic review", International Journal of Software Engineering and Knowledge Engineering.

Hevner, A.R., Linger, R.C., Collins, R.W. & Prowell, S.T. (2005). *"Next Generation Software Engineering",* Software Engineering Institute (SEI), Carnegie Mellon, Pittsurgh, PA, CMU/SEI-2005-TR-015.

Hneif, M. and Sai Peck, Lee (2011). "Using Guidelines to Improve Quality in Software Nonfunctional Attributes," *Software, IEEE* , vol.28, no.6, pp.72-77.

Hernández-López, A., Colomo-Palacios, R., García-Crespo, Á. and Cabezas-Isla, F. (2011). "*Software Engineering Productivity: Concepts, Issues and Challenges*", International Journal of Information Technology Project Management, vol. 2, no. 1, pp. 37-47.

Jones, C. (1996). *Applied Software Measurement: Assuring Productivity and Quality.* 2 ed. McGraw-Hill.

Müller, J., Krüger, J., Enderlein, S., Helmich, M. Zeier, A. (2009). Customizing Enterprise Software as a Service Applications: Back-End Extension in a Multi-tenancy Environment, 11th International Conference, ICEIS Proceedings, Italy, Vol. 24, pp. 66-77.

Jørgensen, M., Indahl, U. and Sjøberg, D.I.K. (2003). "Software effort estimation by analogy and 'regression toward the mean'," J. of Systems & Software, vol. 68, pp. 253-262, 2003.

Jovan Popovič1 and Dragan Bojič1. 2012. A Comparative Evaluation of Effort Estimation Methods in the Software Life Cycle. ComSIS Vol. 9, No. 1.

Kaur, P. & Singh, H. (2008). "*Certification of Software Components*", ACM SIGSOFT Software Engineering Notes, DOI: 10.1145/1384 139.1384142, vol. 33, no. 4, pp. 1-6.

Krishnan, M.S. et al., (2000). "An Empirical Analysis of Productivity and Quality in Software Products", Management Science, vol. 46, no. 6, pp. 745-759.

Kung-Kui Lau & Zheng Wang (2007). "*Software Component Models*", IEEE Transactions on Software Engineering, IEEE Computer Society, vol. 33, no. 10.

Kitchenham, B. and Mendes, E. (2004). "Software Productivity Measurement Using Multiple Size Measures", IEEE Transactions on Software Engineering, vol. 30, no. 12, pp. 1023-1035.

Laakso, T. & Niemi, J. (2008). "*An Evaluation of AJAX-enable Java-based Web Application Frameworks*". Preceedings of MoMM 2008, Linz, Austria, ACM 978-1-60558-269-6/08/0011, pp. 431-437.

Lee, S.P., Thin, S.K. & Liu, H.S. (2000). "*Object-Oriented Application Framework on Manufacturing Domain*". Malaysian Journal of Computer Science, vol. 13, no. 1, pp. 56-64.

Lapouchnian., A. (2011), "Exploiting Requirements Variability for Software Customization and Adaption", Department of Computer Science, University of Toronto.

Malavolta, I. (2010). "*Providing support for creating next generation software architecture languages*", International Conference of Software Engineering

(ICSE' 10), Cape Town, South Africa, Journal of ACM, 2010, 978-1-60558-7196/10/05, pp. 517-518.

Michaela Weiss and Norbert Heidenbluth (2012). "Future Chances of Software Customization: An Empirical Evaluation", 7th International Conference on Software Engineering Advances (ICSEA), IARIA 2012, Vol. 2, pp. 479-485.

Maxwell, K. D. (2001). Collecting Datafor Comparability: Benchmarking software Development Productivity. IEEE Software, September/October 2001.

Maxwell, K.D. and Forselius, P. (2000), "Benchmarking Software Development Productivity", IEEE Software, Vol. 17, pp. 80-88.

Nwelih, E., & Amadin, I.F. (2008) Modeling Software Reuse in Traditional Productivity Model. Asian Journal of Information Technology, 7(11):484-488

Nunamaker, F.J. & Minder Chen, Purdin, T. (2001).*"Systems Development in Information Systems Research"*, Journal of Management Information Systems, Proceedings of the Twenty-Third Haiwaii International Conference on System Sciences (IEEE Computer Society Press), vol. 7, no.3, pp. 89-106.

Oscar, C. & Angel, L. (2006). "The ODESew 2.0 Semantic Web Application Framework". Journal of ACM 1-59593-323-9/06/005, WWW 2006, Edingurgh, Scotland, pp. 1049-1050.

PardoLeite, J., Yu, Y., & Liu L. (2005).*"Quality-Based Software Reuse"*, Department of Computer Science, University of Toronto, Canada.

Premraj, R., Kitchenham, B., Shepperd, M. & Forselius, P. (2005). An Empirical Analysis of Software Productivity over Time. 11th IEEE International Symposium on Software Metrics, 2005.

Paiva, E., Barbosa, D., Lima, R. and Albuquerque, A. (2010). "Factors that Influence the Productivity of Software Developers in a Developer View", Innovations in Computing Sciences and Software Engineering, T. Sobh and K. Elleithy, eds., Springer Netherlands, pp. 99-104.

Premraj, R., Twala, B., Forselius, P. and Mair, C. (2004). "Productiv-ity of Software Projects by Business Sector: An Empirical Analysis of Trends," Late Breaking Paper presented at 10th IEEE Intl. Softw. Metrics Symp., Chicago, USA.

Riehle, D. & Thomas, G. (2005). *"Role Model based Framework Design and Integration"*, In proceedings of the 2005 Conference on object-oriented programming systems, languages and applications (OOPSLA), ACM Press, pp.117-133.

Rajesh Bhatia, M. Dave, and Joshi, R. C. (2010), Ant Colony Based Rule Generation for Reusable Software Component Retrieval, ACM SIGSoft Software Engineering Notes, vol. 35, no. 2, pp.1-4.

Ramirez, Y. W. and Nembhard, D. A. (2004). "Measuring knowledge worker productivity: A taxonomy", Journal of Intellectual Capital, vol. 5, no. 4, pp. 602-628.

Schmidt, D.C. & Hu (2010). *"Applying design patterns and frameworks to develop object-oriented communications software"*. In Handbook of Programming Languages, vol. 10, P. Salus, Ed. Macmillan Publishing Co., Inc., Indianapolis, IN.

Schwabe, D., Esmeraldo, L., Rossi, G. & Lyardet, F. (2001). *"Engineering Web applications for reuse"* IEEE Multimedia, vol. 8, no. 1, pp. 20-31.

Stein Grimstad*, Magne Jørgensen, Kjetil Moløkken-Østvold. 13 June 2005. Software effort estimation terminology: The tower of Babel. Information and Software Technology 48 (2006) 302–310

Stoev, A. & Dimov, A. (2008). *"Architectural framework for Dynamic web-applications"*, International Conference on Computer Systems and Technologies- CompSysTech'08, vol. 2, no. 10, pp. 1-6.

Stoev, A. & Dimov, A. (2008). "Architectural framework for Dynamic web-applications", International Conference on Computer Systems and Technologies- CompSysTech'08, vol. 2, no. 10, pp.1-6.

Sharp, H., Baddoo, N., Sarah, B., Hall, T. and Robinson, H. (2008). "Models of motivation in software engineering," Inf. Software Technology.

Sentas, P., Angelis, L., Stamelos, I. and Bleris, G. (2005). "Software Productivity and Effort Prediction with Ordinal Regression", Information & software Technology, Vol. 47, pp. 17-29.

Tangen, S. (2005). "Demystifying productivity and performance", International Journal of Productivity and Performance Management, vol. 54, no. 1, pp. 34-46.

Yamamoto, H., Washizaki, H., & Fukazawa, Y. (2004). *"A Metrics Suite for Measuring Resuabiluty of Software Components"*, Matsushita Electric Industrial Co., Ltd., Waseda University, Osaka, Japan.

Williams, A.S, Szyperski, C.A., and Wittenberg, C. (2012), "XML Application Framework", Patent No. US 8132148B2, pp. 37, Date of Patent- Mar 6, 2012.

Wagner, S. and Ruhe, M. (2008). "A Systematic Review of Productivity Factors in Software Development", Proc. 2nd International Workshop on Software Productivity Analysis and Cost Estimation (SPACE 2008), IEEE Computer Society.

Wallace & Bruce (2011). *"A Hole for every Component and every Component in its Hole", Existential Programming-2011, Retrieved on 9th April, 2011 from* http://existentialprogramming.blogspot.com/2010/05/hole-for-every-component-and-every.html

Zhuge, J. (2008). "Reward Systems for Implementing Knowledge Sharing in Knowledge - Intensive Corporation," Proc. of the 2008 ISECS International Colloquium on Computing, Com., Control, and Management, pp. 514-518.