

REUSABLE FRAMEWORK FOR WEB APPLICATION DEVELOPMENT

MOHD RAZAK BIN SAMINGAN

UNIVERSITI TEKNOLOGI MALAYSIA

REUSABLE FRAMEWORK FOR WEB APPLICATION DEVELOPMENT

MOHD RAZAK BIN SAMINGAN

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

Faculty of Computing
Universiti Teknologi Malaysia

JUNE 2017

ABSTRACT

Web application (WA) is among the mainstream enterprise-level software solutions. One of the reasons for this trend was due to the presence of Web application framework (WAF) that in many ways has helped web developer to implement WA as an enterprise system. However, there are complexity issues faced by the developers when using existing WAFs as reported by the developers themselves. This study is proposed to find a solution to this particular issue by investigating generic issues that arise when developers utilize Web as a platform to deliver enterprise-level application. The investigation involves the identification of problems and challenges imposed by the architecture and technology of the Web itself, study of software engineering (SE) knowledge adaptation for WA development, determination of factors that contribute to the complexity of WAF implementation, and study of existing solutions for WA development proposed by previous works. To better understand the real issues faced by the developers, hands-on experiment was conducted through development testing performed on selected WAFs. A new highly reusable WAF is proposed, which is derived from the experience of developing several WAs case studies guided by the theoretical and technical knowledge previously established in the study. The proposed WAF was quantitatively and statistically evaluated in terms of its reusability and usability to gain insight into the complexity of the development approach proposed by the WAF. Reuse analysis results demonstrated that the proposed WAF has exceeded the minimum target of 75% reuse at both the component and system levels while the usability study results showed that almost all (15 out of 16) of the questionnaire items used to measure users' attitudes towards the WAF were rated at least moderately by the respondents.

ABSTRAK

Aplikasi Web (WA) telah menjadi salah satu pilihan utama bagi penyelesaian masalah dalam industri pada masa kini. Kewujudan rangka-kerja aplikasi Web (WAF) yang dapat memudahkan pembangunan WA telah menjadi pemangkin utama kepada perkara ini. Namun, seperti yang telah dilaporkan oleh pembangun, terdapat masalah bagi penggunaan WAF sedia ada iaitu tahap kerumitannya yang tinggi. Kajian ini dilaksanakan bagi menyelesaikan masalah tersebut melalui penyiasatan terhadap isu-isu generik yang telah dikenal pasti sebelum ini apabila pembangun cuba membangunkan aplikasi pada tahap industri menggunakan teknologi Web. Penyiasatan melibatkan pengenalanpastian masalah dari segi cabaran dan kekangan oleh seni bina dan teknologi Web itu sendiri, kajian terhadap kesesuaian disiplin kejuruteraan perisian (SE) sedia ada bagi pembangunan WA, penentuan faktor utama yang menyebabkan kerumitan dalam pelaksanaan sebenar WAF, dan kajian terhadap penyelesaian dalam pembangunan WA yang telah dicadangkan oleh penyelidik terdahulu. Untuk memahami isu sebenar dialami pembangun, eksperimen praktikal telah dibuat melalui kajian pembangunan terhadap beberapa WAF terpilih. Pelaksanaan WAF dengan kadar boleh guna semula komponen yang tinggi telah dicadangkan berdasarkan pengalaman pembangunan beberapa siri aplikasi kajian kes yang dipandu teori dan pengetahuan teknikal yang diperolehi sebelum ini. Pelaksanaan WAF tersebut telah diuji dan dinilai secara kuantitatif dan statistik untuk menentukan tahap kerumitannya. Ia melibatkan analisis kadar boleh guna semula komponen WAF dan kebolehgunaan WAF oleh pihak pembangun. Hasil analisis menunjukkan kadar boleh guna semula komponen WAF melebihi tahap minimum yang disasarkan iaitu 75%. Analisis soal selidik kebolehgunaan WAF pula menunjukkan hampir semua (15 daripada 16) item soal selidik telah mendapat maklum balas yang sekurang-kurangnya sederhana daripada pihak responden.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	DECLARATION	ii
	ABSTRACT	iii
	ABSTRAK	iv
	TABLE OF CONTENTS	v
	LIST OF TABLES	xi
	LIST OF FIGURES	xiii
	LIST OF ABBREVIATIONS	xvii
	LIST OF APPENDICES	xx
1	INTRODUCTION	1
	1.1 Background of Problem	1
	1.2 Statement of Problem	3
	1.3 Objectives	4
	1.4 Scope of Study	4
	1.5 Significance of Study	6
	1.6 Organization of Thesis	7
2	LITERATURE REVIEW	8
	2.1 Introduction	8
	2.2 Introduction to Web Application	9
	2.2.1 Web Application Types and Features	9

2.2.2	Issues in Web Application Development	10
2.3	Web Application Architecture	11
2.3.1	Conventional Web Application	12
2.3.2	Java-based Web Application	13
2.4	Software Engineering Disciplines for Web Application Development	15
2.4.1	Object-Oriented Paradigm	15
2.4.2	Software Design Pattern	16
2.4.3	Architecture Design Pattern	17
2.4.4	Class Design Pattern	18
2.4.5	Component-Based Design	19
	2.4.5.1 Software Component Motivation	20
	2.4.5.2 Component Specification	21
	2.4.5.3 Development Strategies	22
2.4.6	MVC Architecture	23
	2.4.6.1 Separation of Concern Design Principle	23
	2.4.6.2 MVC and Design Pattern	24
2.5	Application Framework	26
2.5.1	Software Engineering Aspects in Application Framework	26
	2.5.1.1 Framework Architecture Design	27
	2.5.1.2 Object-oriented Paradigm Implementations	27
2.5.2	White-Box versus Black-Box	28
2.5.3	Web Application Framework	29
	2.5.3.1 Types of Web Application Framework	30
	2.5.3.2 Comparison of Web Application Framework by Other Researchers	31
	2.5.3.3 Best of Breed	32
	2.5.3.4 Monolithic	33

	2.5.3.5 Best of Breeds versus Monolithic Approaches	34
2.6	Development Testing of Existing Web Application Framework	35
2.6.1	Case Study Application Used in Performing WAF Development Testing	36
2.6.2	Development Testing Experience	36
2.7	Other Related Works on Web Application Development Solutions	38
2.7.1	Architecture and Design	39
2.7.2	View Template System	42
2.7.3	Security	43
2.7.4	Tools	45
2.7.5	State-of-the-art Web Application Development Solutions	47
2.8	Software Reuse Metrics	49
2.8.1	Reuse Concepts and Measurement Techniques	49
2.8.2	Implementation Issues and Recommended Practices	50
2.8.3	Web Application Framework Reusability Measurement	52
2.8.4	Reuse Performance Classifications	55
2.9	Software Usability	56
2.9.1	Usability Testing Elements	56
2.9.2	Test Respondents	57
2.9.3	Testing Methods and Instruments	58
2.9.4	Web Application Framework Usability Testing	60
2.10	Summary	60

3	RESEARCH METHODOLOGY	62
3.1	Introduction	62
3.2	Research Design and Procedures	63
3.2.1	Study on Issues Related to Web Application Implementation	64
3.2.2	Development Testing of Existing Web Application Framework	65
3.2.3	Reverse Engineering Task	66
3.2.4	Development of the Proposed Web Application Framework	67
3.2.5	Development Testing and Evaluation of the Proposed Web Application Framework	67
3.3	Theoretical Framework	68
3.4	Summary	71
 4	 THE DESIGN OF REUSABLE WEB APPLICATION FRAMEWORK (ReWAF)	 72
4.1	Introduction	72
4.2	Web Application Framework Architecture and Design Extraction	72
4.2.1	Application Class Design	74
4.2.2	Link Structure, Path, and References	79
4.2.3	Dynamic and Static Module Instantiation	82
4.2.4	Application Controller and View- template Interaction	85
4.2.5	Overall Architecture and Design Implementation	92
4.3	Web Application Framework Design and Architecture Generalization	95
4.3.1	Class Design	96

4.3.2	Reusable Web Application Framework (ReWAF) Architecture	99
4.4	Controller-View-Controller Architecture Classification	105
4.5	Comparison with Similar Architectures	106
4.6	Summary	108
5	EVALUATION	109
5.1	Introduction	109
5.2	Web Application Framework Reusability Analysis	110
5.2.1	Web Applications Case Studies	111
	5.2.1.1 Timetable Management System	112
	5.2.1.2 Question Bank System	112
	5.2.1.3 Johor State Investment Centre Web	114
	5.2.1.4 Collaborative Water Treatment Plant Monitoring System	115
5.2.2	Reuse Measurement and Analysis	116
	5.2.2.1 Reused Module Structure	117
	5.2.2.2 Physical and Logical Source Lines of Code Functions	119
	5.2.2.3 Measurement and Analysis Implementation	123
5.2.3	Reuse Analysis Results	130
	5.2.3.1 System Level Reuse Analysis	130
	5.2.3.2 Component Level Reuse Analysis	137
5.2.4	Reuse Analysis Results Summary	141
5.3	Web Application Framework Usability Testing	142
5.3.1	Respondents and Questionnaire	143
5.3.2	Usability Analysis Results	147
5.3.3	Usability Analysis Results Summary	156

5.4	Comparison with Existing Web Application Frameworks	156
5.4.1	Design Pattern Integration	158
5.4.2	Component-based Specifications	158
5.4.3	Database-oriented Approach	159
5.5	Summary	161
6	CONCLUSION AND FUTURE WORKS	162
6.1	Introduction	162
6.2	Research Objective Achievement	162
6.3	Major Contributions	163
6.4	Future Works	164
6.5	Summary	166
	REFERENCES	168
	Appendices A - C	190-195

LIST OF TABLES

TABLE NO.	TITLE	PAGE
2.1	Comparison of development features of selected WAFs	35
2.2	WA development solution aspects to be incorporated inside the WAF to be proposed in the research	48
2.3	WAF's reuse concepts and measurement contexts	53
3.1	Initial comparison of the new WAF with existing WAFs	71
4.1	ReWAF component's main functions and their task specifications	76
5.1	Reuse types of top, middle, and bottom layer module associations	119
5.2	Different SLOCs functions implementations on Use Case # 1 and Use Case #2	122
5.3	Measurement parameters and equations for the ReWAF's reuse performance analysis	124
5.4	SQL statements to measure the SLOC value of $PS_{CLT}(S)$ and $LSRD_{SVR}(S)$ parameters	127
5.5	Type, number, and size (SLOC) of case study applications modules	132
5.6	Physical, logical, and direct logical reuse size of the case study applications	132
5.7	Physical, logical, and direct logical reuse percent of the case study applications	132
5.8	Descriptive statistics of $RPD_L(m_c)$ distribution across the four case-study applications	138

5.9	Items in the first section of the questionnaire	144
5.10	Items in the second section of the questionnaire and their corresponding usability characteristics	145
5.11	Analysis status of the selected questionnaire items	155
5.12	Comparison of ReWAF with other existing WAFs	157

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
2.1	Performance and complexity comparisons matrix of different WA types	10
2.2	Conventional WA architecture runs on 1st, 2nd, and 3rd models described in Cho et al. (1997)	12
2.3	A general view of J2EE architecture in 3-tier environment (Oracle9i, 2002)	14
2.4	Component interaction in MVC architecture	23
2.5	Realization of MVC architecture	25
2.6	Software engineering elements for WAF implementation	28
3.1	Overall research framework and methodology	63
4.1	Link structure, link node reference, and controller's parameter settings of WA through database-oriented approach of ReWAF	73
4.2	The class design of guestbook-entry function inside "mygb" application	75
4.3	A part of the link tree structure of "mygb" application	80
4.4	Constructed link path and corresponding controllers on each link-node when "Sign Guestbook" end-child link node is selected	80
4.5	Example of parameter settings of link-node references controlled and viewed under ADT's "Link Structure/Reference" main link	81

4.6	The generic component-type modules implementations inside the main-controller	84
4.7	Database table relationships to support ReWAF's database-oriented implementations on link structure, reference, and controller's parameter-setting	85
4.8	The content of main view-template file "template_main.html" assigned to the main-controller	87
4.9	Link-node reference to "webman_component_selector" is made as DYNAMIC_MODULE type of reference assigned to DYNAMIC template-element named "content_main"	88
4.10	The implementation of process_DYNAMIC() hook function inside the main-controller	89
4.11	Sequence diagram describes the flow of controllers' function calls for displaying guestbook entry form page in "mygb" application	91
4.12	Overall content view generated by "Sign Guestbook" main link	92
4.13	Overall architecture and design implementation of the ReWAF	94
4.14	Pseudo-code of the main algorithm implemented in the main-controller of the ReWAF	95
4.15	Logical view of TM pattern applied in the class diagram shown in Figure 4.2	97
4.16	Logical view of Composite pattern applied in the class diagram shown in Figure 4.2	98
4.17	Physical existence and interconnections of "Controller" and "View" elements for the sequence diagram shown in Figure 4.11	100
4.18	General interconnection sequences of "Controller" and "View" elements inside the WAF	101
4.19	General structure of CVC architecture	102
4.20	Hierarchical structure of "Controller" and "View" triads of class diagram structure shown in Figure 4.2	107

5.1	Associations among modules at top, middle, and bottom layers of application module structure	118
5.2	Module T associated with other modules E , G , and P	119
5.3	Two different possible use cases of module T implementations	121
5.4	Relational database tables to model the module's information structure for WAF's reuse performance analysis	125
5.5	Relationship between physical reuse percent ($RP_P(S)$) and size of the case study application ($PS_{CLT}(S)$)	133
5.6	Relationship between direct logical reuse percent ($RPD_L(S)$) and size of the case study application ($PS_{CLT}(S)$)	134
5.7	Relationship between $RPD_L(S)$ and $RP_P(S)$ of the case study applications	135
5.8	Proportions of direct logical reuse percent by association types of modules (In: inheritance, Co: composition, and Ag: aggregation)	136
5.9	Distribution frequency of $RPD_L(m_c)$ of each case study application	139
5.10	Relationship between $RPD_L(m_c)$ and $SLOC(m_c)$ of the case study applications	140
5.11	Developers responses to WAD Experience	148
5.12	Developers responses to SE knowledge level	148
5.13	Developers responses to their roles in WA development	149
5.14	Distribution of responses for WAF's attributes	150
5.15	Distribution of responses for WAF's support tools	150
5.16	Distribution of responses for scripting language used (Perl)	151
5.17	Distribution of responses for respondents' specific opinions	151

5.18	Percentage of responses for WAF's attributes	152
5.19	Percentage of responses for WAF's support tools	152
5.20	Percentage of responses for scripting language used (Perl)	152
5.21	Percentage of responses for user's specific opinions	153
5.22	Database table design of the database-oriented approach implemented in IWTP case study application	160

LIST OF ABBREVIATIONS

AAT	-	Application Administration Tool
ADT	-	Application Development Tool
AJAX	-	Asynchronous JavaScript and XML
ALM	-	Access Log Management
API	-	Application Programming Interface
ASP	-	Active Server Pages
CBD	-	Component-based Development
CBS	-	Component-based System
CGI	-	Common Gateway Interface
CMS	-	Content Management System
CoC	-	Convention over Configuration
CoR	-	Chain of Responsibility
CORBA	-	Common Object Request Broker Architecture
CRP	-	Component's Runtime Parameter
CVC	-	Controller View Controller
DAC	-	Discretionary Access Control
DCOM	-	Distributed Component Object Model
DFD	-	Data-Flow Diagram
DI	-	Dependency Injection
EJB	-	Enterprise JavaBeans
ERD	-	Entity-Relationship Diagram

GUI	-	Graphical User Interface
HDM	-	Hypermedia Design Method
HLD	-	Hyperlink Diagram
HTML	-	Hypertext Markup Language
IoC	-	Inversion of Control
J2EE	-	Java 2 Platform, Enterprise Edition
JDBC	-	Java Database Connectivity
JSON	-	JavaScript Simple Object Notation
JSP	-	Java Server Pages
JVM	-	Java Virtual Machine
LDAP	-	Lightweight Directory Access Protocol
LLOC	-	Logical Lines of Code
LOC	-	Lines of Code
LSL	-	Link Structure Logic
MAC	-	Mandatory Access Control
MIS	-	Management Information System
MVC	-	Model View Controller
OO	-	Object-oriented
OOD	-	Object-oriented Design
OOHDM	-	Object Oriented Hypermedia Design Method
OOP	-	Object-oriented Programming
ORM	-	Object Relational Mapping
PAC	-	Presentation Abstraction Control
PHP	-	PHP: Hypertext Preprocessor
RBAC	-	Role-based Access Control
RDBMS		Relational Database Management System

REST	-	Representational State Transfer
RMM	-	Relationship Management Methodology
SAC	-	Security and Access Control
SE	-	Software Engineering
SLOC	-	Source Lines of Code
SoC	-	Separation of Concern
SQL	-	Structured Query Language
TM	-	Template Method
URL	-	Uniform Resource Locator
WA	-	Web Application
WAD	-	Web Application Development
WAF	-	Web Application Framework
XML	-	Extensible Markup Language
XSL	-	Extensible Stylesheet Language
XSLT	-	Extensible Stylesheet Language Transformations

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	ReWAF's standard module functions and types	190
B	Example of responses to the questionnaire	192
C	Publications	195

CHAPTER 1

INTRODUCTION

1.1 Background of Problem

As the Web evolves from a simple content delivery system to a platform for complex online applications, its trend in terms of research also shifted from concentrating on technology, service, and performance to application development activities requiring intensive use of software engineering (SE) disciplines. According to the basic steps of software development in SE, the general software development process normally involves analysis, design, implementation, testing, and maintenance. Directly adapting the concepts, principles, and techniques of conventional SE disciplines to be used in Web application (WA) especially for WA architecture tend to lead to design flaws. This is based on the fact that WA is different from conventional SE disciplines and can be far more complex from conventional client-server application since it heavily relies on hyperlinks and runs in stateless environment. Due to the stateless nature of Web, keeping track of user session in Web application is more challenging than previously experienced in traditional client/server application and is fully under developers' responsibility (Du et al., 2011). Thus, there is an interest in this type of research as demonstrated by Schwabe et al. (1999) and Isakowitz et al. (1995) that emphasizes on Web development methodologies. However, these studies only provide solutions at the highest level of design and conceptual view of Web application, and are only suitable to be used to model information structure that embodied Web application.

Through a quick review of previous related works, there are various solutions that have been proposed with regard to unique problems that need to be solved in WA. However, majority of the studies solved the problems separately, not as an integrated system that covers all aspect of solutions required in WA development. At the implementation stage of a large-scale problem, most developers normally turn to Web application framework (WAF) that can cover all possible solution aspects including architecture and design, security control, view-template system facility, and tools. WAF has the obligation to incorporate all aspects of WA solutions in its development environment.

There are many new WAFs that have been proposed by Web developer community as a solution for WA development. Most developers use these WAFs with the intention to speed up the development process and to simplify the coding task. However, through a quick survey from several Web developer forums and blogs (Eckel, 2006; O'Brien, 2006; Regebro, 2009; Abid, 2011), it is often stated that despite offering full-stack solutions, many of these WAFs are considered too complex and have high learning curve from novice developers' or even more experienced developers' points of view. Developers might face complicated WAF tasks starting from the initial set-up and installation process up to the development and maintenance phases. In Java world, the Struts framework (Struts, 2000), a decent and among the oldest WAF, is a good example where its complexity is well known and agreed by its own expert community (Grobmeier, 2011; Grashel, 2014).

There are also arguments that some of the WAFs have been over engineered (Lapide et al., 2010), for example by introducing complex XML-based set-up files and new coding syntax that slightly or even totally different from the programming language that the WAF itself is based on. In general, from the developers' perspective, the main issue to be solved by WAF is it must be specifically oriented to be used by developers, so that they are able to implement the best development practices through the use of view-template system, object orientation paradigm, and database-centric approach in a simpler way. Complexity issues that can be either directly or indirectly related with WAF implementation have also been discussed by the academia (Schmidt and Fayad, 1997; Uhler, 2001; Zhang et al., 2004; Silva and

Moreira, 2005; Vuksanovic and Sudarevic, 2011). Majority of the problems associated with WAF's complexities stated by the academia were also in-line with those mentioned by the practitioners or developers. Therefore, it is crucial to find the solutions that solve both the development aspects of WA as well as the complexity of WAF itself.

1.2 Statement of Problem

As briefly discussed in previous section, existing WAFs were in some way able to provide solutions in many WA development aspects. However, WAF itself may become an issue due to its complex nature. Thus, it is crucial to address the research problem as stated below:

“Why is the WA development through the use of WAF too challenging (time-consuming, complex, and not cost-effective) although there are a lot of approaches, techniques, and design patterns being introduced to the WAF implementation?”

Several research questions need to be answered by WAF designers and developers before they try to solve the research problem stated above. The list of the research questions is as follows:

- (i) What are the challenges faced by developers in using existing WAFs for WA development?
- (ii) Why is the process of developing WA using existing WAFs challenging?
Are the challenges due to the following reasons?
 - approaches, techniques, and architecture and design embodied inside WAF,
 - the implementation of WAF and the way it was presented to the developers,

- the complexity of WAF that is used to build a complex system.
- (iii) What are the important aspects that should be emphasized in a WAF to make it easier to be used by developers?
- (iv) How should these important aspects be incorporated in a WAF and presented to the developers?
- (v) How can these important WAF aspects be exploited to reduce the complexity of WAF?

1.3 Objectives

The objectives of this study are as the following:

- (i) To propose a reusable software framework as a solution to the problems and challenges encountered in WA development.
- (ii) To implement the proposed reusable software framework in the form of WAF.
- (iii) To evaluate the reusability and usability aspects of the WAF through a series of development tests of selected case study applications.

1.4 Scope of Study

The scope of the study will be mainly focused on architecture and design of WA. As described in Bourque and Fairley (2014), software architecture and design has become one of the sub elements of body of knowledge (BoK) in SE discipline categorized under “Software Design” knowledge area (KA). Specific to this

particular sub element of SE's BoK, the concentration will be more on object-oriented design (OOD), component-based design (CBD), architecture styles, and design patterns. All these will become the major focus of the study.

The study is also scoped by the type of WAF to be proposed that is mainly targeted to be used to develop WA that heavily relies on database. Thus, the study tries to explore on how to extent the role of the database to not only act as a content feeder to the application but also to provide control on some logical parts of the application. Other possible aspects of WA such as application links management, users' authentication, session handling, and access control on application's resources will also be considered to be controlled through the use of database technology.

Within the WA domain itself, the study will not take these issues into consideration:

- (i) Support for client-side business solutions that requires integration with client script technology (JavaScript or AJAX). The new WAF will apply thin-client architecture with all core business solutions to be allocated at the server side. This is one of the strategies to make the WAF less complicated by avoiding application logic from being scattered to both the client and the server sides.
- (ii) WA performance (speed and reliability) as the research only emphasizes on architecture to ease WA development process (implementation). This is based on the fact that not all WA require high computing resources such as amazon.com. There are many requests from small organizations to develop WAs that are able to solve complex business tasks but only cater small number of users.
- (iii) Persistency and transactional operations on data as it can usually be handled by add-in module inside the Web server (mod_perl or mod_python) and standard functions that can be provided by the database

application server through the implementation of Relational Database Management System (RDBMS).

- (iv) Web development methodologies as the WAF will be used as a collection of tools, regardless of any methodology used prior to the implementation phase. However, a quick review of existing Web development methodologies such as Hypertext Design Model (HDM), Object-Oriented Hypertext Design Model (OOHDM), and Relationship Management Methodology (RMM) is performed in order to understand the common issues arising in WA development.

1.5 Significance of Study

The study is intended to benefit both the industry and academia. In WA industry, Web developers benefited from the productivity and ease of development through the use of WAFs that are able to realize the implementation of reusable component-based architecture. Component-based software has a significant positive impact in software industry in terms of quality, productivity, and cost improvement (Bose, 2010). This is based on the fact that component-based architecture facilitated by productive development tools help to reduce development effort and simplify the overall application development process (Jha et al., 2014; Vale et al., 2016). For the academic research community, the study can lead to a better understanding of how software architecture and design should be incorporated in WA through the implementation of WAF. This is important as the search for better WA architecture and design is still being investigated by other researchers until these recent years (Huiyao et al., 2014; Villamizar et al., 2015; Cheng et al., 2016).

1.6 Organization of Thesis

This chapter provides background of the problems, research questions to be addressed, and objectives of the research. The rest of the chapters in this thesis are organized as follows. Chapter 2 reviews the theoretical foundation of SE to be applied in the research. Generally, the review provides the basis for the implementation of Object-Oriented Design (OOD), Component-Based Development (CBD), and application framework concepts in WA development. Previous works related to WA development solutions, software metric models, and software usability testing techniques are also discussed in Chapter 2. The theories and methods of software metric and software usability will be used to empirically and statistically evaluate the WAF proposed in this research.

The overall research framework is presented in Chapter 3. The major phases and iterative process of research designs and procedures are outlined in this particular chapter. Chapter 4 describes the proposed reusable component-based architecture that has been realized in the form of a WAF. It also explains on how the architecture is extracted and established based on the SE knowledge gained from the review presented in Chapter 2.

The reusability and usability analyses of the proposed WAF are discussed in detail in Chapter 5. The reusability analysis technique proposed will be mainly derived from the software metric models described in Chapter 2. The usability analysis will be performed by conducting WAF development test to selected developers. The results for both reusability and usability analyses will be discussed in the same chapter. Finally, Chapter 6 summarizes the overall conclusions of the research.

proposed WAF (ReWAF) does not require it since they have been defined as a built-in feature of the ReWAF through the component-based specifications derived from the implementation of CBD approach.

REFERENCES

- Abid, U. (2011). *Why is Zend Framework so complicated?* (Stack Exchange Inc.) Retrieved 6 March, 2015, from Programmers Stack Exchange: <http://programmers.stackexchange.com/questions/123495/why-is-zend-framework-so-complicated>
- ActiveRecord. (2014). *RubyGems.org*. Retrieved 9 July, 2014, from ActiveRecord: <https://rubygems.org/gems/activerecord>
- Appleton, B. (1997). Patterns and Software: Essential Concepts and Terminology. *Object Magazine Online*, 3(5), pp. 20-25.
- Arndt, C. (2009). *Best of Breed: TG's job is hard. Here's why*. Retrieved 13 May, 2015, from Percious.com: <http://web.archive.org/web/20140118052005/http://percious.com/blog/archives/31>
- Arthur, J., & Azadegan, S. (2005). Spring Framework for rapid open source J2EE Web Application Development - A case study. *6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2005)*, pp. 90-95. IEEE.
- ASF. (2010). *Common Chain*. Retrieved from Apache Commons: <http://commons.apache.org/proper/commons-chain/>
- Ates, M., Schneider, J., & Dauvergne, B. (2012). An architecture for Web application session switching in virtual organizations. *7th International Conference on Digital Information Management (ICDIM)*, pp. 232-238. IEEE.
- Atwood, J. (2005). *Are Design Patterns How Languages Evolve?* Retrieved 9 July, 2014, from Coding Horror: <http://blog.codinghorror.com/are-design-patterns-how-languages-evolve/>

- Avraam N., C., & George A., P. (2007). Implementing a generic component-based framework for telecontrol applications. *Software: Practice and Experience*, 37(10), pp. 1087–1132.
- Barb, A. S., Neill, C. J., Sangwan, R. S., & Piovoso, M. J. (2014). A Statistical Study of the Relevance of Lines of Code Measures in Software Projects. *Innovations in Systems and Software Engineering*, 10(4), pp. 243-260.
- Barkley, J. F., Cincotta, A. V., Ferraiolo, D. F., Gavrilla, S., & Kuhn, D. R. (1997). Role Based Access Control for the World Wide Web. *20th National Information System Security Conference*. NIST/NSA.
- Basili, V. R., Briand, L. C., & Melo, W. L. (1996). How Reuse Influences Productivity in Object-Oriented Systems. *Communications of the ACM*, 39(10), pp. 104-116.
- Bevan, N., Carter, J., & Harker, S. (2015). ISO 9241-11 revised: What have we learnt about usability since 1998? *International Conference on Human-Computer Interaction*, pp. 143-151. Springer.
- Bieman, J. M. (1992). Deriving Measures of Software Reuse in Object Oriented Systems. *Formal Aspects of Measurement*, pp. 79-82.
- Bose, D. (2010). *Component Based Development*. Indian Statistical Institute.
- Bourque, P., & Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge, Version 3.0*. IEEE Computer Society.
- Bozzon, A., Fraternali, P., Comai, S., & Carughi, G. T. (2006). Conceptual Modeling and Code Generation for Rich Internet Applications. *6th International Conference on Web Engineering. ICWE'06*, pp. 353-360. ACM.
- Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*, 189(194), pp. 4-7.
- Bytes. (2008). *Bytes: A community for Developers and IT Professionals*. Retrieved 27 Jun, 2016, from <https://bytes.com>
- Cai, J., Kapila, R., & Pal, G. (2000). *HMVC: The layered pattern for developing strong client tiers*. Retrieved from <http://www.javaworld.com/javaworld/jw-07-2000/jw-0721-hmvc.html>
- CakePHP. (2005). *Cake Software Foundation*. (Cake Software Foundation, Inc.) Retrieved 1 July, 2016, from <http://cakephp.org/>

- Capretz, L. F. (2003). A Brief History of the Object-Oriented Approach. *ACM SIGSOFT Software Engineering Notes*, 28(2).
- Carromeu, C., Paiva, D. M., Cagnin, M. I., Rubinsztein, H. K., & Turine, M. A. (2010). Component-based Architecture for e-Gov Web Systems Development. *17th IEEE International Conference and Workshops on Engineering of Computer Based System. ECBS*, pp. 379-385. IEEE.
- Cassim, H. (2013). *Using Frameworks to Build Websites and Web Applications*. Retrieved 27 Jun, 2016, from OStraining: <https://www.ostraining.com/blog/webdesign/frameworks/>
- Catalyst. (2012). *Perl MVC framework*. (Catalyst Foundation) Retrieved 1 July, 2016, from <http://www.catalystframework.org/>
- Ceri, S., Fraternali, P., & Paraboschi, S. (1999). Data-Driven One-to-One Web Site Generation for Data-Intensive Applications. *VLDB*, 99(200), pp. 7-10.
- Chae, J.-H., Yoo, C.-J., Kim, Y.-S., & Chang, O.-B. (2003). XSLT Template Design for Generating the Web Presentation Layer. *Tenth Asia-Pacific Software Engineering Conference (ASPEC 2003)*, pp. 396-404. IEEE.
- Cheng, R., William, S., Ellenbogen, P., Howell, J., Roesner, F., Krishnamurthy, A., & Anderson, T. (2016). Radiatus: a Shared-Nothing Server-Side Web Architecture. *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pp. 237-250. ACM.
- Chidamber, S. R., & Kemerer, C. F. (1994). A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6), pp. 476-493.
- Chimaris, A. N., & Papadopoulos, G. A. (2007). Implementing a generic component-based framework for telecontrol applications. *Software: Practice and Experience*, 37(10), pp. 1087-1132.
- Cho, E. S., Kim, M. S., & Kim, S. D. (2001). Component Metrics to Measure Component Quality. *Eighth Asia-Pacific Software Engineering Conference (APSEC 2001)*, pp. 419-426. IEEE.
- Cho, E. S., Kim, S. D., Rhew, S. Y., Lee, S. D., & Kim, C. G. (1997). Object-Oriented Web Application Architectures and Development Strategies. *Asia Pacific Software Engineering Conference and International Computer Science Conference (APSEC/ICSC '97)*, pp. 322-331. IEE.

- Christensen, E., Curbera, F., Meredith, G., & Weerawarana, S. (2001). *Web Services Description Language (WSDL) 1.1*. (W3C) Retrieved 30 August, 2016, from W3C Note: <http://www.w3.org/TR/wsdl>
- Christiansson, B., & Christiansson, M.-T. (2003). The Missing Approach for Component Specification. In S. Assar, F. Semmak, & R. Barkhi (Ed.), *Proceedings of 1st International Workshop on Component-Based Business Information Systems Engineering. CBBISE'03*. Geneva.
- Codehaus. (2003). *PicoContainer*. (PicoContainer Committers) Retrieved 9 July, 2014, from <http://picocontainer.codehaus.org/>
- CodeIgniter. (2006). *CodeIgniter*. (British Columbia Institute of Technology) Retrieved 1 July, 2016, from <https://www.codeigniter.com/>
- Coutaz, J. (1987). PAC, an Object Oriented Model for Dialog Design. In H.-J. Bullinger, & B. Shackel (Ed.), *INTERACT 87 - 2nd IFIP International Conference on Human-Computer Interaction*, pp. 431-436. Stuttgart Germany.
- Crnkovic, I. (2001). Component-based Software Engineering - New Challenges in Software Development. *Software Focus*, 2(4), pp. 127-133.
- Crnkovic, I., Vulgarakis, A., & Chaudron, M. R. (2011). A Classification Framework for Software Component Model. *IEEE Transactions on Software Engineering*, 37(5), pp. 593-615.
- Dallal, J. A., & Morasca, S. (2014). Predicting Object-oriented Class Reuse-proneness Using Internal Quality Attributes. *Empirical Software Engineering*, 19(4), pp. 775-821.
- Dancer. (2009). *Dancer Perl web framework*. Retrieved 1 July, 2016, from <http://www.perldancer.org/>
- Davis, F. D. (1993). User Acceptance of Information Technology: System Characteristics, User Perceptions and Behavioral Impacts. *International Journal of Man-Machine Studies*, 38, pp. 475-487.
- DBI. (2002). *Perl DBI*. (Perl.org) Retrieved from Perl's Database Interface: <http://dbi.perl.org/>
- DBIx::Class. (2005). *CPAN*. Retrieved from DBIx::Class - Extensible and flexible object <-> relational mapper: <http://search.cpan.org/~ribasushi/DBIx-Class-0.08270/lib/DBIx/Class.pm>

- Devanbu, P., Karstu, S., Melo, W., & Thomas, W. (1996). Analytical and Empirical Evaluation of Software Reuse Metrics. *18th International Conference on Software Engineering*, pp. 189-199. Berlin, Germany: IEEE Computer Society.
- Distante, D., Pedone, P., Rossi, G., & Canfora, G. (2007). Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces. *Web Engineering: 7th International Conference (ICWE 2007)*, pp. 457-472. Springer.
- Django. (2005). *Django Software Foundation*. (Django Software Foundation) Retrieved 1 July, 2016, from <https://www.djangoproject.com/>
- Dolbec, J., & Shepard, T. (1995). A component based software reliability model. *1995 conference of the Centre for Advanced Studies on Collaborative research*, p. 19. IBM Press.
- Driscoll, J. (2005). *Servlet History*. (Oracle) Retrieved 25 March, 2015, from java.net:
https://weblogs.java.net/blog/driscoll/archive/2005/12/servlet_history_1.html
- Du, W., Jayaraman, K., Tan, X., Luo, T., & Chapin, S. (2011). Position Paper: Why Are There So Many Vulnerabilities in Web Applications? *2011 workshop on New security paradigms workshop*, pp. 83-94. ACM.
- Durham, J. (2001). *History-making components: Tracing the roots of components from OOP through WS*. Retrieved 8 July, 2014, from <http://archive.today/hD88t>
- Dworak, H. (2009). A Concept of a Web Application Blending Thin and Fat Client Architectures. *Fourth International Conference on Dependability of Computer Systems (DepCos-RELCOMEX)*, pp. 84-90. IEEE.
- Eckel, B. (2006). *Python Directions and the Web Framework Problem*. Retrieved 10 September, 2014, from artima developer:
<http://www.artima.com/weblogs/viewpost.jsp?thread=150834>
- Evans, B. (2013). *The small, medium, and large of Ruby Frameworks*. (CBS Interactive) Retrieved 29 April, 2015, from TechRepublic:
<http://www.techrepublic.com/blog/australian-technology/the-small-medium-and-large-of-ruby-frameworks/>

- Faulkner, L. (2003). Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, and Computers*, 35(3), pp. 379-383.
- Favaro, J. A. (1996). Comparison of Approaches to Reuse Investment Analysis. *Fourth International Conference on Software Reuse*, pp. 136-145. Orlando, FL, USA: IEEE.
- Fernandez-Villamor, J. I., Diaz-Casillas, L., & Iglesias, C. A. (2008). A Comparison Model for Agile Web Frameworks. *2008 Euro American Conference on Telematics and Information Systems (EATIS '08)*, p. 14. New York, NY, USA: ACM.
- Flask. (2010). *Flask web development, one drop at a time*. Retrieved 1 July, 2016, from <http://flask.pocoo.org/>
- Florenzano, E. (2007). *Cheetah and Django*. (eflorenzano.com) Retrieved January, 2016, from Eric Florenzano's Blog: <http://eflorenzano.com/blog/2007/08/04/cheetah-and-django/>
- Fowler, M. (2004). *Inversion of Control Containers and the Dependency Injection pattern*. Retrieved from <http://martinfowler.com/articles/injection.html>
- Fowler, M. (2012). *Martin Fowler on ORM Hate*. (DZone, Inc.) Retrieved 23 September, 2014, from <http://java.dzone.com/articles/martin-fowler-orm-hate>
- Frakes, W. B., & Succi, G. (2001). An Industrial Study of Reuse, Quality, and Productivity. *Journal of Systems and Software*, 57(2), pp. 99-106.
- Frakes, W., & Terry, C. (1994). Reuse Level Metrics. *Third International Conference on Software Reuse: Advances in Software Reusability*, pp. 139-148. IEEE.
- Frakes, W., & Terry, C. (1996). Software Reuse: Metrics and Models. *ACM Computing Surveys*, 8(2), pp. 415-435.
- Frankel, D. (2004). Model-Driven Software Development. *MDA Journal*.
- Fraternali, P. (1999). Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. 31(3), pp. 227-263.
- Gabriel, R. P. (1996). *Patterns of Software: Tales From The Software Community*. (Oxford University Press) Retrieved from <http://www.dreamsongs.com/NewFiles/PatternsOfSoftware.pdf>

- Gallidabino, A., & Pautasso, C. (2016). The Liquid.js Framework for Migrating and Cloning Stateful Web Components across Multiple Devices. *25th International Conference Companion on World Wide Web*, pp. 183-186. ACM.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA: Addison-Wesley Longman.
- Garcia, F. J., Castanedo, R. I., & Fuente, A. A. (2007). A Double-Model Approach to Achieve Effective Model-View Separation in Template Based Web Applications. *International Conference on Web Engineering. ICWE 2007*, pp. 442-456. Springer.
- Garn, B., Kapsalis, I., & Simos, D. E. (2014). On the Applicability of Combinatorial Testing to Web Application Security Testing: A Case Study. *Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing*, pp. 16-21. ACM.
- Gill, N. S. (2003). Reusability Issues in Component-Based Development. *ACM SIGSOFT Software Engineering Notes*, 28(4), pp. 4-4.
- Gitzel, R., & Aleksy, M. (2004). Implementation of a Model-Centric Web Application Framework with J2EE. *3rd International Symposium on Principles and Practice of Programming in Java*, pp. 148-153. Trinity College Dublin.
- GlassFish. (2014). *GlassFish - World's first Java EE 7 Application Server*. Retrieved 25 March, 2015, from <https://glassfish.java.net/>
- Gonzalez, R., & Torres, M. (2006). Issues in Component-Based Development: Towards Specification with ADLs. *Journal of Systemics, Cybernetics, and Informatics*, 4(5), pp. 49-54.
- Goschka, K. M., & Riedling, E. (1997). Development of an Object Oriented Framework for Design and Implementation of Database Powered Distributed Web Applications with the DEMETER Project as a Real-Life Example. *23rd Euromicro Conference. 'New Frontiers of Information Technology'. EUROMICRO 97*, pp. 132-137. IEEE.
- Grails. (2005). *A powerful Groovy-based web application framework for the JVM*. Retrieved 1 July, 2016, from <https://grails.org/>

- Grashel, R. (2014). *Stripes Wiki*. Retrieved 29 March, 2016, from <https://stripesframework.atlassian.net/wiki/display/STRIPES/Stripes+vs.+Struts>
- Gravelle, R. (2009). *Introduction to Server-side JavaScript*. (QuinStreet, Inc.) Retrieved 29 June, 2015, from WebReference: <http://www.webreference.com/programming/javascript/rg37/index.html>
- Grobmeier, C. (2011). *APACHE WICKET VERSUS APACHE STRUTS 2*. Retrieved 29 March, 2016, from <https://www.grobmeier.de/apache-wicket-versus-apache-struts-2-04052011.html>
- GuangChun, L., Lu, W., & Hanhong, X. (2003). A Novel Web Application Frame Developed by MVC. *ACM SIGSOFT Software Engineering Notes*, 28(2), p. 7.
- Gui, G., & Scott, P. D. (2009). Measuring Software Component Reusability by Coupling and Cohesion Metrics. *Journal of Computers*, 4(9), pp. 797-805.
- Halstead, M. H. (1977). *Elements of Software Science (Operating and programming systems series)*. New York, NY, USA: Elsevier Science Inc.
- Hamlet, D., Mason, D., & Woit, D. (2001). Theory of software reliability based on components. *23rd international conference on Software engineering*, pp. 361-370. IEEE Computer Society.
- Harper, B. D., & Norman, K. L. (1993). Improving User Satisfaction: The Questionnaire for User Interaction Satisfaction Version 5.5. *1st Annual Mid-Atlantic Human Factors Conference*, pp. 224-228.
- Heer, J., & Agrawala, M. (2006). Software Design Patterns for Information Visualization. *IEEE Transaction on Visualization and Computer Graphics*, 12(5), pp. 853-860.
- Henry, E., & Faller, B. (1995). Large-Scale Industrial Reuse to Reduce Cost and Cycle Time. *IEEE Software*, 12(5), pp. 47-53.
- Hibernate. (2014). *Hibernate ORM*. (Red Hat) Retrieved 9 July, 2014, from Red Hat JBoss Middleware: <http://hibernate.org/orm/>
- Holzinger, A. (2005). Usability Engineering Methods for Software Developers. *Communication of the ACM*, 48(1), pp. 71-74.
- Hsu, C.-L., Liao, H.-C., Chen, J.-L., & Wang, F.-J. (1999). A Web Database Application Model for Software Maintenance. *The Fourth International*

Symposium on Autonomous Decentralized Systems. ISADS 1999, pp. 338-344. IEEE.

- Huiyao, A., Yang, S., Tao, Y., Hui, L., Peng, Z., & Jun, Z. (2014). A New Architecture of Ajax Web Application Security Crawler with Finite-State Machine. *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 112-117. IEEE.
- Hunter, J. (1998). *Introducing the new Servlet API 2.1*. (JavaWorld, Inc.) Retrieved 25 March, 2015, from JavaWorld:
<http://www.javaworld.com/article/2076838/java-web-development/introducing-the-new-servlet-api-2-1.html>
- Huston, V. (2006). *GoF Structure Similarities*. Retrieved 9 July, 2014, from
<http://www.vincehuston.org/dp/index.html#similarities>
- IEEE. (1990). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*, 610. IEEE.
- Isakowitz, T., Stohr, E., & Balasubramanian, P. (1995). RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38(8), pp. 34-44.
- Izquierdo, R., Juan, A., Lopez, B., Devis, R., Cueva, J. M., & Acebal, C. F. (2003). Experiences in Web Site Development with Multidisciplinary Teams. From XML to JST. *International Conference on Web Engineering (ICWE 2003)*, pp. 459-462. Springer.
- Jabangwe, R., Borstler, J., Smite, D., & Wohlin, C. (2015). Empirical evidence on the link between object-oriented measures and external quality attributes: a systematic literature review. *Empirical Software Engineering*, 20(3), pp. 640-693.
- Jang, Y.-S., & Choi, J.-Y. (2014). Detecting SQL injection attacks using query result size. *Computer & Security*, 14, pp. 104-118.
- Java. (1995). *Java Software*. Retrieved 1 July, 2016, from
<https://www.oracle.com/java/index.html>
- Jazayeri, M. (2007). Some Trends in Web Application Development. *Future of Software Engineering. FOSE '07*, pp. 199-213. IEEE.
- JDX. (1997). *J-Database Exchange*. (Software Tree, LLC) Retrieved 22 July, 2015, from <http://www.softwaretree.com/products/jdx/Jdx1.htm>

- Jha, P. C., Bali, V., Narula, S., & Kalra, M. (2014). Optimal component selection based on cohesion & coupling for component based software system under build-or-buy scheme. *Journal of Computational Science*, 5(2), pp. 233-242.
- Jin, Y. (2005). International Conference on Information Reuse and Integration. *International Conference on Information Reuse and Integration*, pp. 536-541. IEEE.
- Johnson, R. E. (1997). Frameworks=(Components+Patterns). *Communications of the ACM*, 40(10), pp. 39-42.
- Jones, C. (2013). *A Short History of the Lines of Code (LOC) Metric*. Retrieved 11 October, 2015, from NAMCOOK ANALYTICS:
<http://namcookanalytics.com/wp-content/uploads/2013/07/LinesofCode2013.pdf>
- Joshi, J. B., Aref, W. G., Ghafoor, A., & Spafford, E. H. (2001). SECURITY MODELS FOR WEB-BASED APPLICATIONS. *Communications of the ACM*, 44(2), pp. 38-44.
- Kirakowski, J., & Corbett, M. (1993). SUMI: The Software Usability Measurement Inventory. *British Journal of Educational Technology*, 24(3), pp. 210-212.
- Koirala, S. (2008). *Design pattern – Inversion of control and Dependency injection*. (CodeProject) Retrieved 2 July, 2015, from CODE PROJECT:
<http://www.codeproject.com/Articles/29271/Design-pattern-Inversion-of-control-and-Dependency>
- Krishnamurthy, S., & Aditya, P. M. (1997). On the estimation of reliability of a software system using reliabilities of its components. *The Eighth International Symposium on Software Reliability Engineering*, pp. 146-155. IEEE.
- Krug, S. (2006). *Don't Make Me Think: A Common Sense Approach to Web Usability* (2nd ed.). Berkeley, CA: New Riders.
- Kumar, V., Sharma, A., Kumar, R., & Grover, P. S. (2012). Quality aspects for component-based systems: A metrics based approach. *Software: Practice and Experience*, 42(12), pp. 1531-1548.
- Laravel. (2011). (TAYLOR OTWELL) Retrieved 2017, from Laravel:
<https://laravel.com/>

- Laskowski, J. (2003). *OpenEJB: EJB for Tomcat*. (O'Reilly Media, Inc.) Retrieved 25 March, 2015, from ONJava:
http://www.onjava.com/pub/a/onjava/2003/02/12/ejb_tomcat.html
- Lea, D. (1994). Christopher Alexander: An Introduction for Object-Oriented Designers. *SIGSOFT Software Engineering Notes*, 19(1), pp. 39-46.
- Leach, R. J. (2012). *Software Reuse: Methods, Models, Costs* (2nd ed.). Aftermath.
- Lee, W.-M. (2005). *What is ASP.NET*. Retrieved 19 March, 2015, from
<http://www.windowsdevcenter.com/pub/a/dotnet/2005/09/19/what-is-asp-net.html>
- Lewis, J. R. (1995). IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction*, 7(1), pp. 57-78.
- Li, J., & Chusho, T. (2012). A Web Application Framework for End-User-Initiative Development with a Visual Tool. *Proceedings of the International MultiConference of Engineers and Computer Scientists. 1*, pp. 816-822. Hong Kong: IAENG.
- Li, J., Han, J., Li, Z., & Zhao, Y. (2011). SCENE Admin: A Component-based Integrated Management Framework for Web Service Platforms. *International Symposium on IT in Medicine and Education. ITME*, pp. 77-81. IEEE.
- Lim, W. C. (1994). Effects of Reuse on Quality, Productivity, and Economics. *IEEE Software*, 11(5), pp. 23-30.
- Lindquist, T. E., Gary, K. A., Koehnemann, H. E., & Naccache, H. (1999). Component Framework for Web-Based Learning Environments. *29th Annual Frontiers in Education Conference. FIE '99*, pp. 23-28. IEEE.
- Luders, F., & Lau, K. K. (2002). Specification of Software Components. In I. Crnkovic, & M. Larsson (Eds.), *Building Reliable Component-Based Software Systems*. Artech House.
- Mahoney, M. S. (2004). Finding a History for Software Engineering. *IEEE Annals of the History of Computing*, 26(1), pp. 8-19.
- Mak, G., & Guruzu, S. (2010). *Hibernate Recipes: A Problem-Solution Approach*. Apress.
- Mao-Shan, S., Yi-Hai, C., Sheng-Bo, C., & Jia, M. (2010). A model checking approach to Web application navigation model with session mechanism.

International Conference on Computer Application and System Modeling (ICCASM), pp. V5-398. IEEE.

Marston, T. (2004). *A Role-Based Access Control (RBAC) system for PHP*. Retrieved 15 July, 2015, from <http://www.tonymarston.net/php-mysql/role-based-access-control.html>

Martin, R. C. (2000). *Design Principles and Design Patterns*. Retrieved 9 July, 2014, from Object Mentor:
http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf

Mascena, J. C., Almeida, E. S., & Meira, S. R. (2005). A Comparative Study on Software Reuse Metrics and Economic Models from a Traceability Perspective. *IEEE International Conference on Information Reuse and Integration. IRI-2005*, pp. 72-77. IEEE.

May, J. (2002). *Component-Based software reliability analysis*. Department of Computer Science, University of Bristol. CSTR.

McKendrick, J. (2006). *Another view: XML not meant to be 'human readable'*. (CBS Interactive) Retrieved 11 July, 2015, from ZDNet:
<http://www.zdnet.com/blog/service-oriented/another-view-xml-not-meant-to-be-human-readable/758>

Milosavljevid, B., Vidakovid, M., & Konjovid, Z. (2002). Automatic Code Generation for Database-Oriented Web Applications. *Inaugural Conference on the Principles and Practice of Programming. PPPJ '02*, pp. 59-64. National University of Ireland.

Mockus, A., Zhang, P., & Li, P. L. (2005). Predictors of Customer Perceived Software Quality. *27th International Conference on Software Engineering*, pp. 225-233. St. Louis, Missouri, USA: ACM.

mod_perl. (2014). *mod_perl*. Retrieved 18 March, 2015, from <http://perl.apache.org/>

Mohagheghi, P., & Conradi, R. (2007). Quality, productivity and economic benefits of software reuse: a review of industrial studies. *Empirical Software Engineering*, 12(5), pp. 471-516.

Mohagheghi, P., & Conradi, R. (2008). An Empirical Investigation of Software Reuse Benefits in a Large Telecom Product. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 17(3), p. 13.

- Mohagheghi, P., Conradi, R., Killi, O. M., & Schwarz, H. (2004). An Empirical Study of Software Reuse vs. Defect-Density and Stability. *26th International Conference on Software Engineering (ICSE 2004)*, pp. 282-291. IEEE.
- Mojolicious. (2008). *Mojolicious*. Retrieved 1 July, 2016, from <http://mojolicious.org/>
- moodledev. (2016). *Data Manipulation API*. Retrieved 25 June, 2016, from Moodle Doc: https://docs.moodle.org/dev/Data_manipulation_API
- Moreno-Ger, P., Torrente, J., Hsieh, Y. G., & Lester, W. T. (2012). Usability Testing for Serious Games: Making Informed Design Decisions with User Data. *Advances in Human-Computer Interaction, 2012*, pp. 1-13.
- Mosher, B. (2007). *Specs and Version History of Microsoft Active Server Pages*. Retrieved 19 March, 2015, from suite101: http://web.archive.org/web/20071107074935/http://web-programming.suite101.com/article.cfm/from_classic_asp_to_aspnet_20
- Mozilla. (2014). *Mozilla Developer Network*. Retrieved 18 March, 2015, from <https://developer.mozilla.org/en-US/docs/AJAX>
- Murk, O., & Kabanov, J. (2006). Aranea - Web Framework Construction and Integration Kit. *4th International Symposium on Principles and Practice of Programming in Java (PPPJ '06)*, pp. 163-172. ACM Press.
- Nelson, M. L. (1999). A Design Pattern for Autonomous Vehicle Software Control Architectures. *23rd Annual International Computer Software and Applications Conference. COMPSAC '99*, pp. 172-177. IEEE.
- Nguyen, V., Deeds-Rubin, S., Tan, T., & Boehm, B. (2007). *A SLOC Counting Standard*. University of Southern California. California, USA: Center for Systems and Software Engineering.
- Nidiffer, K. E. (2007). Addressing the Software Engineering Challenges over the Years and into the Future. *Journal of Software Technologies: Future Directions in Software Engineering, 10*(3).
- Nielsen, J. (1996). Usability Metrics: Tracking Interface Improvements. *IEEE Software, 13*(6), pp. 12-13.
- Nielsen, J. (2000). *Why You Only Need to Test with 5 Users*. Retrieved 27 July, 2016, from <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>

- Nielsen, J. (2010). *Testing Expert Users*. Retrieved 17 May, 2016, from <https://www.nngroup.com/articles/testing-expert-users/>
- Nielsen, J. (2012). *Usability 101: Introduction to Usability*. Retrieved 26 July, 2016, from <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>
- Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. *Proceedings of the INTERACT'93 and CHI'93 conference on Human Factors in Computing Systems*, pp. 206-213. ACM.
- O'Brien, T. (2006). *What Web Application framework should you use?* (O'Reilly Media Inc.) Retrieved 6 March, 2015, from O'REILLY: http://archive.oreilly.com/pub/post/isnt_rails_supposed_to_change.html
- Offutt, J. (2002). Quality Attributes of Web Software Applications. *IEEE Software*, 9(2), pp. 25-32.
- OJB. (2012). Retrieved from Apache ObjectRelationalBridge: <http://db.apache.org/ojb/index.html>
- Okanović, V., & Mateljan, T. (2011). Designing a new web application framework. *MIPRO, 2011 proceedings of the 34th international convention*, pp. 1315-1318. IEEE.
- Oracle9i. (2002). *Oracle9i Application Server Migrating From WebSphere Release 2 (9.0.2)*. Retrieved from Oracle9i Application Server: https://docs.oracle.com/cd/B10570_07/migrate.902/a95110/overview.htm
- Padrino. (2010). *Padrino Ruby web framework*. Retrieved 1 July, 2016, from <http://padrinorb.com/>
- Palviainen, M., Evesti, A., & Ovaska, E. (2011). The reliability estimation, prediction and measuring of component-based software. *Journal of Systems and Software*, 84(6), pp. 1054-1070.
- Park, R. E. (1992). *Software Size Measurement: A Framework for Counting Source Statements*. Software Engineering Institute, Carnegie Mellon University.
- Parmanto, B., Lewis, A. N., Graham, K. M., & Bertolet, M. H. (2016). Development of the Telehealth Usability Questionnaire (TUQ). *International Journal of Telerehabilitation*, 8(1), pp. 3-10.
- Parr, T. (2004). Enforcing Strict ModelView Separation in Template Engines. *13th International Conference on World Wide Web*, pp. 224-233. ACM.

- Parr, T. (2013). *String Template*. Retrieved 14 July, 2015, from <http://www.stringtemplate.org/>
- Perl. (1987). *The Perl Programming Language*. Retrieved 1 July, 2016, from <https://www.perl.org/>
- Pesot, J., Hancock, S., & Meyers, C. (2002). Simplifying the development of enterprise-scale e-business applications. *A WebSphere Studio Enterprise Developer solution*. New York: IBM Software Group.
- PHP. (1995). *The PHP Group*. Retrieved 1 July, 2016, from <http://php.net/>
- Poulin, J. (2002). An agenda for software Reuse Economics. *International Conference on Software Reuse*, 15.
- Poulin, J. S. (1994). Measuring Software Reuse. *Third International Conference on Software Reuse: Advances in Software Reusability*, pp. 126-138. Rio de Janeiro: IEEE.
- Poulin, J. S. (1996). The Search for a General Reusability Metric. *Proceeding of the Workshop on Reuse and the NASA Software Strategic Plan*. Fairfax, VA.
- Poulin, J. s., & Caruso, J. M. (1993). A Reuse Metrics and Return on Investment Model. *2nd Workshop on Software Reuse: Advances in Software Reusability*, pp. 152-166. IEEE.
- Prieto-Diaz, R. (1993). STATUS REPORT: SOFTWARE REUSABILITY. *IEEE Software*, 10(3), pp. 61-66.
- PSE. (2016). *Programmers Stack Exchange*. (Stack Exchange Inc.) Retrieved 27 Jun, 2016, from <http://programmers.stackexchange.com/>
- Python. (1994). *Python Software Foundation*. Retrieved 1 July, 2016, from <https://www.python.org/>
- Qiao-ming, Z., Lei, Z., & Pei-de, Q. (2004). A Simplified Database Oriented Web Framework. *Wuhan University Journal of Natural Sciences*, 9(5), 706-710.
- Radosevic, D., & Magdalenic, a. I. (2011). Python Implementation of Source Code Generator Based on Dynamic Frames. *Proceedings of the 34th International Convention. MIPRO*, pp. 969-974. IEEE.
- Radosevic, D., Orehovacki, T., & Magdalenic, I. (2012). Towards Software Autogeneration. *Proceedings of the 35th International Convention. MIPRO*, pp. 1076-1081. IEEE.

- Rajapakse, D. C., & Jarzabek, S. (2009). Towards generic representation of web applications: solutions and trade-off. *Software: Practice and Experience*, 39(5), pp. 501-530.
- Ramm, M. (2008). "Site Components" in *Django and TG2*. (Compound Thinking) Retrieved 24 May, 2015, from <http://compoundthinking.com/blog/index.php/2008/02/13/site-components-in-django-and-tg2/>
- Ramm, M. (2009). *Coupling Django Style*. (Compound Thinking) Retrieved 24 May, 2015, from Compound Thinking: <http://web.archive.org/web/20140118153633/http://compoundthinking.com/blog/index.php/2008/02/13/site-components-in-django-and-tg2/>
- Regebro, L. (2009). *Comments on Django's design decisions*. (WordPress.com) Retrieved 10 September, 2014, from Lennart Regebro: Python, Plone, Web All your Python needs: <http://regebro.wordpress.com/2009/04/11/comments-on-djangos-design-decisions/>
- Regebro, L. (2009). *Comments on Django's design decisions*. (Wordpress) Retrieved 6 March, 2015, from Lennart Regebro: Python, Plone, Web: <https://regebro.wordpress.com/2009/04/11/comments-on-djangos-design-decisions/>
- Richardson, M. (1999). *Larry Wall, the Guru of Perl*. Retrieved 26 May, 2015, from Linux Journal: <http://www.linuxjournal.com/article/3394>
- Ridjanovic, D., & Okanovic, V. (2004). Using Database Framework in Web Applications. *12th IEEE Mediterranean Electrotechnical Conference. MELECON 2004*, pp. 697-700. IEEE.
- Riehle, D., & Zullighoven, H. (1996). Understanding and Using Patterns in Software Development. 2(1), pp. 3-13.
- Rine, D. C., & Nada, N. (2000). Three Empirical Studies of a Software Reuse Reference Model. *Software: Practice and Experience*, 30(6), 685-722.
- Roberts, D., & Johnson, R. (1996). Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks. *Pattern Languages of Programs. PLoP '96*.
- Rode, J. (2004). Nonprogrammer Web Application Development. *Human Factors in Computing Systems 2004 (CHI '04)*, pp. 1055-1056. ACM.

- Roichman, A., & Gudes, E. (2007). Fine-grained Access Control to Web Databases. *ACM Symposium on Access Control Models and Technologies. SACMAT '07*, pp. 31-40. ACM.
- Roma. (2006). *Roma Framework*. Retrieved 1 July, 2016, from <https://sourceforge.net/projects/romaframework/>
- RoR. (2003). *Ruby on Rails*. (Basecamp) Retrieved 11 July, 2014, from <http://rubyonrails.org/>
- Roshandel, R., Banerjee, S., Cheung, L., Medvidovic, N., & Golubchik, L. (2006). Estimating software component reliability by leveraging architectural models. *28th international conference on Software engineering*, pp. 853-856. ACM.
- Rubin, J., & Chisnel, D. (2008). In *Handbook of Usability Testing: Howto Plan, Design and Conduct Effective Tests* (2nd ed.), pp. 22-25. Wiley Publishing, Inc.
- Ruby. (1995). *Ruby A PROGRAMMER'S BEST FRIEND*. Retrieved 1 July, 2016, from <https://www.ruby-lang.org>
- Ryck, P. D., Nikiforakis, N., Desmet, L., Piessens, F., & Joosen, W. (2012). Serene: Self-Reliant Client-Side Protection against Session Fixation. *12th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2012)*, pp. 59-72. Springer.
- Saccoccio, R. (1996). *FastCGI*. (Open Market) Retrieved 19 March, 2015, from <http://www.fastcgi.com/>
- Santos, M. E., Polvi, J., Taketomi, T., Yamamoto, G., Sandor, C., & Kato, H. (2015). Toward Standard Usability Questionnaires for Handheld Augmented Reality. *IEEE Computer Graphics and Applications*, 35(5), pp. 66-75.
- Sanz, D., Diaz, P., & Aedo, I. (2002). Implementing RBAC Policies in a Web Server. *6th International ICC/IFIP Conference on Electronic Publishing*. Karlovy Vary, Czech Republic: VWF Berlin via ELPUB.
- Schmidt, D. C. (1995). Using Design Patterns to Develop Reusable Object-Oriented Communication Software. *Communications of the ACM*, 38(10), pp. 65-74.
- Schmidt, D. C. (2000). Developing Flexible and High-performance Web Servers with Frameworks and Patterns. *ACM Computing Surveys (CSUR)*, 32(1), p. 39.

- Schmidt, D. C. (2012). *Why Software Reuse has Failed and How to Make It Work for You*. Retrieved from Distributed Object Computing (DOC) Group:
<http://www.dre.vanderbilt.edu/~schmidt/reuse-lessons.html>
- Schmidt, D. C., & Fayad, M. (1997). Object-Oriented Application Frameworks. *Communications of the ACM, Special Issue on Object-Oriented Application Frameworks*, 40(10), pp. 32-38.
- Schmientendorf, A., Dimitrov, E., Dumke, R., Foltin, E., & Wipprecht, M. (1999). Conception and Experience of Metrics-Based Software Reuse in Practice. *International Workshop on Software Measurement (IWSM'99)*, pp. 178-189. Lac Superieur, Canada.
- Schwabe, D., Pontes, R. d., & Moura, I. (1999). OOHDM-Web: An Environment for Implementation of Hypermedia Applications in the WWW. *SIGWEB Newsletter*, 8(2), pp. 18-34.
- Schwartz, B. (2004). *The Paradox of Choice: Why More Is Less* (1st ed.). New York, USA: HarperCollins.
- Selby, R. W. (2005). Enabling Reuse-based Software Development of Large-scale Systems. *Transactions on Software Engineering*, 31(6), pp. 495-510.
- Selfa, D. M., Carrillo, M., & Boone, M. d. (2006). A Database and Web Application Based on MVC Architecture. *Electronics, Communications and Computers. CONIELECOMP 2006*, pp. 48-53. IEEE.
- Seshadri, G. (1999). *Understanding JavaServer Pages Model 2 architecture*. (JavaWorld, Inc.) Retrieved 25 March, 2015, from JavaWorld:
<http://www.javaworld.com/article/2076557/java-web-development/understanding-javascript-model-2-architecture.html>
- Shaik, A. S., Hossain, G., & Yeasin, M. (2010). Design, Development and Performance Evaluation of Reconfigured Mobile Android Phone for People Who are Blind or Visually Impaired. *28th ACM International Conference on Design of Communication*, pp. 159-166. ACM.
- Siggelkow, B. (2005). *A Look at Commons Chain, Part 1*. Retrieved 22 September, 2016, from ONJava:
<http://www.onjava.com/pub/a/onjava/2005/03/02/commonchains.html>

- Silber, D. H. (1998). *Java CGI HOWTO*. (Advameg, Inc.) Retrieved 19 March, 2015, from Internet FAQ Archives: <http://www.faqs.org/docs/Linux-HOWTO/Java-CGI-HOWTO.html>
- Silva, E. Q., & Moreira, D. d. (2005). Developing Customizable Web-based Educational Applications through a Component-based Framework. *International Conference on Next Generation Web Services Practices. NWeSP 2005*. pp. 6-pp. IEEE.
- Simons, P. (28 October, 2002). *FastCGI Application Framework*. Retrieved 18 March, 2015, from Savannah: <http://www.nongnu.org/fastcgi/#framework>
- Sinatra. (2007). *Sinatra* . Retrieved 1 July, 2016, from <http://www.sinatrarb.com/>
- Smarty. (2002). *Smarty Template Engine*. (New Digital Group, Inc.) Retrieved 14 July, 2015, from <http://www.smarty.net/>
- Souer, J., & Mierloo, M. v. (2008). A Component Based Architecture for Web Content Management: Runtime Deployable WebManager Component Bundles. *Eighth International Conference on Web Engineering. ICWE '08*, pp. 366-369. IEEE.
- Sova, D. H., & Nielsen, J. (2003). *How to Recruit Participants for Usability Studies*. Retrieved 31 July, 2016, from <https://www.nngroup.com/reports/how-to-recruit-participants-usability-studies/>
- Spring. (2014). *Spring Framework*. (GoPivotal Inc.) Retrieved 10 July, 2014, from Spring: <http://projects.spring.io/spring-framework>
- SQLAlchemy. (2014). *The Python SQL Toolkit and Object Relational Mapper*. Retrieved 9 July, 2014, from SQLAlchemy: <http://www.sqlalchemy.org/>
- stackoverflow. (2016). *Stack Overflow*. (Stack Exchange Inc.) Retrieved 27 Jun, 2016, from <http://stackoverflow.com/>
- Stella, L. F., Jarzabek, S., & Wadhwa, B. (2008). A Comparative Study of Maintainability of Web Applications on J2EE, .NET and Ruby on Rails. *10th International Symposium on Web Site Evolution (WSE2008)*, pp. 93-99. IEEE.
- Struts, A. (2000). (Apache Software Foundation) Retrieved 9 July, 2014, from Apache Struts: <http://struts.apache.org/>

- Succi, G., Benedicenti, L., & Vernazza, T. (2001). Analysis of the Effects of Software Reuse on Customer Satisfaction in an RPG Environment. *Transactions on Software Engineering*, 27(5), pp. 473-479.
- Sun. (2004). Web Application Framework Overview. *Sun Java (TM) Studio Enterprise 7 2004Q4*. Sun Microsystems, Inc.
- Sun, D., Wong, K., & Moise, D. (2003). Lessons Learned in Web Site Architectures for Public Utilities. *Fifth IEEE International Workshop on Web Site Evolution (WSE 2003)*, pp. 93-100. IEEE.
- Sutter, H. (2000). *GotW Archive, Issue #70*. Retrieved from Guru of the Week: <http://www.gotw.ca/gotw/070.htm>
- Symfony. (2005). *Symfony*. (SensioLabs) Retrieved 1 July, 2016, from <https://symfony.com/>
- Szyperski, C., Gruntz, D., & Murer, S. (1998). *Component Software - Beyond Object-Oriented Programming*. New York: Addison-Wesley.
- Taguchi, M., Suzuki, T., & Tokuda, T. (2003). A Visual Approach for Generating Server Page Type Web Applications Based on Template Method. *Human Centric Computing Languages and Environments (HCC 2003)*, pp. 248-250. IEEE.
- Tate, B. (2006). Pain. In *From Java to Ruby* (pp. 14-33). Pragmatic Bookshelf.
- Thomas, W. M., Delis, A., & Basili, V. R. (1997). An Analysis of Errors in a Reuse-oriented Development Environment. *Journal of Systems and Software*, 38(3), pp. 211-224.
- Tilley, S., & Huang, S. (2001). Evaluating the Reverse Engineering Capabilities of Web Tools for Understanding Site Content and Structure: A Case Study. *23rd International Conference on Software Engineering (ICSE 2001)*, pp. 514-523. IEEE.
- Trails. (2006). *Trails Framework*. Retrieved 1 July, 2016, from <https://www.openhub.net/p/trails>
- Trung, Thanh, P., & Thang, H. Q. (2009). Building the reliability prediction model of component-based software architectures. *Int'l Journal of Information Technology*, 5(1), pp. 18-25.
- TurboGears. (2005). *TurboGears: The Web Framework that scales with you*. Retrieved 10 July, 2014, from <http://turbogears.org/>

- u1db. (2011). *u1db 0.1.4 Documentation*. Retrieved 25 June, 2016, from <https://pythonhosted.org/u1db/high-level-api.html>
- Uhler, S. A. (2001). *Design and Architecture of the Brazil Web Application Framework*. Sun Microsystems, Inc.
- USDA. (2014). *USDA Enterprise Architecture Program*. Retrieved 9 July, 2014, from Office of the Chief Information Officer, USDA: <http://www.ocio.usda.gov/about-ocio/governance-and-strategic-investment-gsi/enterprise-architecture>
- Vale, T., Crnkovic, I., de Almeida, E. S., Neto, P. A., Cavalcanti, Y. C., & Romero, S. (2016). Twenty-eight years of component-based software engineering. *Journal of Systems and Software*, 111, pp. 128-148.
- Ventuneac, M., Coffey, T., & Salomie, I. (2003). A POLICY-BASED SECURITY FRAMEWORK FOR WEB-ENABLED APPLICATION. *1st International Symposium on Information and Communication Technologies (ISICT '03)*, pp. 487-492. Trinity College Dublin.
- Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. *Computing Colombian Conference (10CCC)*, pp. 583-590. IEEE.
- Vuksanovic, I. P., & Sudarevic, B. (2011). Use of Web Application Framework in the Development of Small Applications. *34th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2011*, pp. 458-462. Opatija, Croatia: IEEE.
- Washizaki, H., Yamamoto, H., & Fukazawa, Y. (2003). A Metrics Suite for Measuring Reusability of Software Components. *9th International Symposium on Software Metric*, pp. 211-223. Sydney, NSW, Australia.
- Weisfeld, M. (2005). *The Evolution of Object-Oriented Languages*. Retrieved from Developer.com: <http://www.developer.com/java/other/article.php/3493761/The-Evolution-of-Object-Oriented-Languages.htm>
- West, A. (2009). *NASA study on flight software complexity*. NASA. Retrieved 11 August, 2015, from http://www.nasa.gov/pdf/418878main_FSWC_Final_Report.pdf

- Yii. (2006). (Yii Software LLC) Retrieved 2017, from Yii Framework:
<http://www.yiiframework.com/>
- Zaharias, P., & Angeliki, P. (2009). Developing a usability evaluation method for e-learning applications: Beyond functional usability. *International Journal of Human-Computer Interaction*, 25(1), pp. 75-98.
- Zeiger, S. (1999). *Servlet Essential*. Retrieved 25 March, 2015, from Novocode.com:
<http://www.novocode.com/doc/servlet-essentials/>
- Zhang, F., Zhou, X., Chen, J., & Dong, Y. (2008). A novel model for component-based software reliability analysis. *11th IEEE High Assurance Systems Engineering Symposium*, pp. 303-309. IEEE.
- Zhang, J., & Buy, U. (2003). A Framework for the Efficient Production of Web Applications. *Eighth IEEE International Symposium on Computers and Communications. ISCC '03*, pp. 419-424. IEEE.
- Zhang, J., Chung, J.-Y., & Chang, C. K. (2004). Towards Increasing Web Application Productivity. *ACM Symposium on Applied Computing (SAC '04)*, pp. 1677-1681. ACM Press.
- Zhang, W., & Jarzabek, S. (2005). Reuse without Compromising Performance: Industrial Experience from RPG Software Product Line for Mobile Devices. In H. Obbink, & K. Pohl (Ed.), *9th International Software Product Lines Conference (SPLC 2005)*, pp. 57-69. Rennes, France: Springer.
- Zhang, W., & Kim, M. (2005). Application Frameworks Technology in Theory and Practice. *The Fifth International Conference on Electronic Business*, pp. 769-776. Hong Kong.
- Zope. (2009). *The Zope Framework*. Retrieved 25 May, 2015, from
<http://zope2.zope.org/>