

The copyright © of this thesis belongs to its rightful author and/or other copyright owner. Copies can be accessed and downloaded for non-commercial or learning purposes without any charge and permission. The thesis cannot be reproduced or quoted as a whole without the permission from its rightful owner. No alteration or changes in format is allowed without permission from its rightful owner.



**MONITORING ORIENTED AGILE BASED WEB APPLICATIONS
DEVELOPMENT METHODOLOGY FOR SMALL SOFTWARE FIRMS
IN JORDAN**



MOATH HUSNI AHMAD ALTARAWNEH

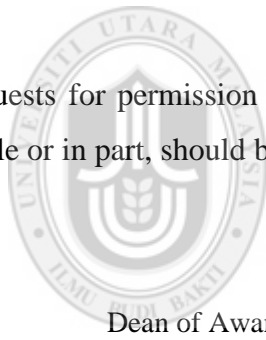
UUM
Universiti Utara Malaysia

**DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2016**

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:



UUM

Universiti Utara Malaysia

Dean of Awang Had Salleh Graduate School of Arts and Sciences

UUM College of Arts and Sciences

Universiti Utara Malaysia

06010 UUM Sintok

Abstrak

Firma perisian bersaiz kecil (SSF) penting kepada industri perisian di kebanyakan negara kerana ia memberi sumbangan besar kepada pertumbuhan ekonomi. Di Jordan, kebanyakan syarikat perisian yang terlibat dengan pembangunan aplikasi Web adalah firma bersaiz kecil. Walau bagaimanapun, tahap penggunaan amalan terbaik bagi pembangunan dan pengurusan aplikasi Web dalam firma-firma ini adalah terhad. Selain itu, kaedah pembangunan perisian yang sedia ada masih kurang pemantauan terhadap proses dan produk. Hasilnya, aplikasi web yang dibangunkan gagal disiapkan dalam tempoh dan bajet yang ditetapkan serta tidak memenuhi keperluan pengguna. Oleh itu, kajian ini bertujuan untuk membina satu metodologi baru dikenali sebagai Metodologi Pembangunan Aplikasi Web Berasaskan Agil yang Berorientasikan Pemantauan (MOGWD) bagi SSF. Kajian ini telah memperkenalkan satu Kaedah Agil Lanjutan melalui penambahbaikan elemen bagi kaedah Scrum dengan Extreme Programming (XP). Seterusnya, kaedah Agil Lanjutan tersebut telah ditambah baik dengan menggabungkan langkah-langkah utama kaedah reka bentuk Web dan Kaedah Pemantauan Berorientasikan Matlamat (GOMM). GOMM telah mendefinisikan dua puluh matlamat. Setiap matlamat mempunyai satu atau lebih soalan. Setiap soalan dijawab melalui metrik yang telah ditakrifkan. Terdapat 101 metrik kualitatif untuk memantau kualiti proses, dan 37 metrik kuantitatif untuk memantau kualiti proses dan produk. Selain itu, metodologi MOGWD yang dicadangkan mentakrifkan empat fasa: Perancangan, Pelaksanaan, Penyemakan dan Tindakan. Metodologi MOGWD telah dinilai menggunakan semakan pakar dan kajian kes. Hasil penilaian menunjukkan bahawa metodologi MOGWD telah mencapai kepuasan pengamal SSF dan didapati boleh dipraktikkan dalam persekitaran yang sebenar. Kajian ini memberi sumbangan kepada bidang pembangunan berasaskan Agil dan pengukuran aplikasi Web. Ia juga menyediakan kepada pengamal SSF satu metodologi pembangunan yang dapat memantau kualiti proses dan produk bagi pembangunan Web.

Kata kunci: Syarikat perisian kecil, kaedah Plan-Do-Check-Act, kaedah Agil, Pemantauan berorientasikan matlamat, Pembangunan Web.

Abstract

Small software firms (SSF) is vital to the software industry in many countries as they provide substantial growth to their economy. In Jordan, most software companies that are involved with developing Web applications are small firms. However, the extent of applying best Web applications development and management practices in these firms is limited. Besides, the existing software development methods are still lack of monitoring the quality of process and product. As a result, the Web application being developed exceeds deadlines and budget, and not meeting user requirements. Therefore, this research aims to construct a new methodology referred as Monitoring Oriented Agile Based Web Applications Development (MOGWD) Methodology for SSF. This study introduced an Extended Agile Method by extending the Scrum method with Extreme Programming (XP) elements. The Extended Agile Method was improved by combining common steps of Web design method and incorporating the Goal Oriented Monitoring Method (GOMM). The GOMM has defined twenty goals. Each goal has one or more questions. The questions are answered through the defined metrics. There are 101 qualitative metrics for monitoring the process quality, and 37 quantitative metrics for monitoring the process and product quality. Moreover, the proposed MOGWD methodology defines four phases: Plan, Do, Check and Act. The MOGWD methodology was evaluated using expert review and case study. The evaluation results show that the MOGWD methodology has gained SSF practitioners' satisfaction and found to be practical for the real environment. This study contributes to the field of Agile based development and Web applications measurement. It also provides SSF practitioners a development methodology that monitors the quality of the process and product for Web development.

Keywords: Small software firms, Plan-Do-Check-Act method, Agile method, Goal oriented monitoring, Web development.

Acknowledgement

By the name of ALLAH, I would like to convey my appreciation to everyone who provided me with exceptional support, encouragement and wisdom in completing this thesis. First and foremost, I am deeply grateful to my two supervisors Assoc. Prof. Dr. Fauziah Baharom and Assoc. Prof. Dr. Fudziah Ahmad for all their efforts in providing me fervent support, intelligent guidance and invaluable suggestions during this work.

On a more personal level, I want to thank my family for their unconditional love, understanding and support. My late Father and Mother raised me to believe that I could achieve anything I set my mind to. My brothers, Hatem, Dirar and Dergam and My sisters Khawla and Tamador, have been an endless source of great joy, love and assistance. I want to thank them all for their interest and assurance that the journey does have an end at times when it seems like no end was insight.

Many thanks go to my wonderful friends for their consistent support, encouragement, and real friendship that I needed in UUM and who were there for me all the time. Special thanks to Ahmad, Ibrahim, Marwan, Mejhem and Omar Tarawneh, Ali and Wa'el Naimat, Hamza Alba'ol, Ibraheem Al shamayleh, and many others. I will never forget the great times I spent with.

Thank You All Very Much

Table of Contents

Permission to Use	ii
Abstrak.....	iii
Abstract	iv
Acknowledgement	v
Table of Contents	vi
List of Tables	xi
List of Figures	xiv
List of Appendices	xv
List of Abbreviations	xvi
CHAPTER ONE INTRODUCTION	1
1.1 Overview.....	1
1.2 Background study.....	1
1.3 Problem Statement.....	4
1.4 Research Questions.....	8
1.5 Research Objectives.....	9
1.6 Research Scope.....	9
1.7 Research Contribution	10
1.8 Significance of this Research.....	12
1.9 Organization of Chapters	13
CHAPTER TWO LITERATURE REVIEW	15
2.1 Introduction.....	15
2.2 Related studies on SSF	15
2.2.1 SSF characteristics and problems	16
2.2.2 Studies on the practices in SSF.....	18
2.3 Existing development methods	24
2.3.1 XP and Scrum analysis	28
2.3.1.1 The Development Process	29
2.3.1.2 Project Management.....	31
2.3.1.3 Requirements.....	31

2.3.1.4 Testing	32
2.3.1.5 Design.....	32
2.3.1.6 Team Structure	33
2.3.1.7 Comparison Results.....	33
2.3.2 Web Design Methods.....	45
2.3.2.1 Web application Common Design Steps.....	49
2.4 Software Measurements	51
2.4.1 Measurement Methods.....	52
2.4.1.1 Measurement Methods Evaluation.....	56
2.4.2 Measurement Mechanism Purposes.....	58
2.4.2.1 Benefits of Using Monitoring	58
2.4.2.2 Measurement mechanism critical success factors	59
2.4.2.3 Development process quality factors	61
2.5 Criteria of a good methodology for Web applications in SSF.....	66
2.6 Validation methods	69
2.6.1 Validation factors.....	72
2.7 Summary	74
CHAPTER THREE RESEARCH METHODOLOGY	77
3.1 Introduction.....	77
3.2 Research Design Approach.....	77
3.3 Research Methodology	79
3.3.1 Theoretical study.....	79
3.3.2 Survey	83
3.3.3 Methodology Construction	86
3.3.4 Methodology Evaluation.....	93
3.4 Summary	102
CHAPTER FOUR SURVEY	103
4.1 Introduction.....	103
4.2 Questionnaire Structure	103
4.3 Respondent's Background	103
4.3.1 Organization Background.....	104

4.3.2 Software Development and Measurement Practices	104
4.3.3 Web application development and measurement practices	104
4.4 Questionnaire Validation	105
4.4.1 Construct Validity.....	105
4.4.2 Content Validity.....	106
4.5 Identify Respondents and Sampling Type	106
4.6 Questionnaire Distribution and Data Collection.....	107
4.7 Analysis and Results	108
4.7.1 Demographic Data	108
4.7.1.1 Respondents Background.....	108
4.7.1.2 Organization background	110
4.7.2 Current Software Development and Measurement Practices	111
4.7.2.1 Software Development Practices.....	111
4.7.2.2 Software Measurement Practices	119
4.7.3 Web Application Development and Measurement Practices	126
4.7.4 Discussion of Findings.....	132
4.7.5 Summary.....	135
CHAPTER FIVE METHODOLOGY CONSTRUCTION.....	136
5.1 Introduction.....	136
5.2 The Extended Agile Method.....	136
5.3 The overview of MOGWD methodology	140
5.4 MOGWD Methodology	144
5.4.1 Plan Phase	144
5.4.1.1 Management Planning.....	144
5.4.1.2 Development planning	147
5.4.1.3 Monitoring planning.....	153
5.4.2 Do (iteration).....	175
5.4.3 Check	179
5.4.4 Act.....	185
5.5 Summary	187
CHAPTER SIX METHODOLOGY EVALUATION.....	188

6.1 Introduction.....	188
6.2 Verification based on the experts review	188
6.2.1 Results of Round one	188
6.2.1.1 Answers and suggestions related comprehensive criterion	189
6.2.1.2 Answers and suggestions related understandability criterion	190
6.2.1.3 Answers and suggestions related feasibility criterion	193
6.2.1.4 Answers and suggestions related to the general overview part... ..	194
6.2.2 Results of round two	195
6.2.2.1 Process and Methods Modifications.....	196
6.2.2.2 Tools Modification	204
6.2.2.3 Team structure Modifications	204
6.2.2.4 General overview modifications	204
6.2.3 Results of Round Three	205
6.3 Validation based on case study and yardstick method.....	205
6.3.1 Validation based on the case study method	205
6.3.1.1 Case Study: Developing the CMS web application by Firm “A”.....	207
6.3.1.1.1 Plan	209
6.3.1.1.2 Do phase	213
6.3.1.1.3 Check phase.....	215
6.3.1.1.4 Act phase	215
6.3.1.2 Validation results.....	224
6.3.2 Validation based on the yardstick method	227
6.4 Summary	233
CHAPTER SEVEN CONCLUSION.....	234
7.1 Introduction.....	234
7.2 Overview of Results.....	234
7.2.1 Theoretical study.....	234
7.2.2 Survey	235
7.2.2.1 SSF Characteristics.....	235
7.2.2.2 Development issues	236
7.2.2.3 Measurement issues.....	236

7.2.2.4 The current Web applications development and measurement practices.....	237
7.2.3 Methodology Construction	237
7.2.4 Methodology Evaluation.....	238
7.2.4.1 Verification.....	238
7.2.4.2 Validation	238
7.3 Research contributions.....	239
7.3.1 MOGWD methodology	239
7.3.2 Extended Agile method with Web design method	240
7.3.3 GOMM.....	241
7.3.4 Survey results.....	242
7.4 Limitations of the Research	242
7.4.1 Lack of the Related Researches	242
7.4.2 Limited Scope in the Evaluation Processes	243
7.5 Future Work.....	243
7.5.1 Add more quality factors for the process.....	243
7.5.2 Using other Agile Practices or methods for the Extended Agile method	244
7.5.3 Extend the MOGWD to include other Key process areas	244
7.6 Summary	245
REFERENCES.....	246

List of Tables

Table 2.1 Problems Faced by SSF	16
Table 2.2 Previous Studies in Web and Software Development Practices	20
Table 2.3 Web Applications Development Best Practices for SSF	21
Table 2.4 Agile Development Methods	25
Table 2.5 Software Development Methods Comparison	27
Table 2.6 Development Process Criteria.....	29
Table 2.7 XP and Scrum Comparison Table.....	30
Table 2.8 Agile Principles Symbols.....	34
Table 2.9 XP, Scrum practices and Agile principles mapping	35
Table 2.10 XP and Scrum Combination from Previous Studies comparison.	41
Table 2.11 Type of Practices and Reason of Adoption.....	44
Table 2.12 Advantages and disadvantages of the Web design methods.....	47
Table 2.13 Common Activities of Web Design Methods.....	50
Table 2.14 Goal Format	54
Table 2.15 Light weight GQM against original GQM adopted from Wangenheim et al. (2003).	55
Table 2.16 Measurement Methods Evaluation.....	57
Table 2.17 Measurement Mechanism Success Factors.....	60
Table 2.18 Process Quality Factors.....	63
Table 2.19 Criteria of Good Development Methodology	67
Table 2.20 Validation method comparison.....	71
Table 2.21 Kunda's validation factors adopted from Kunda (2001)	73
Table 2.22 Literature Review Analysis.....	75
Table 3.1 The Activities of the Web Design Methods.....	82
Table 3.2 Experts Profile.....	96
Table 3.3 Factors for validating the Effectiveness of the Proposed MOGWD Methodology	100
Table 4.1 Questionnaire Response Rate	107
Table 4.2 Current Position Activities.....	110
Table 4.3 Numbers of Employees	110
Table 4.4 Methods that Respondents are Familiar with.....	113
Table 4.5 Requirements Collection Method	113
Table 4.6 Requirements Specification Notation	114
Table 4.7 Programming Languages	115
Table 4.8 Test Types	115
Table 4.9 Testing Process Stages	116
Table 4.10 Reasons of Not Using the Current Methods	117
Table 4.11 Reuse Types	117
Table 4.12 Quality Assurance Activities	118
Table 4.13 Performing QA Activities	119
Table 4.14 Measurements stages and size of company.....	121
Table 4.15 Metrics Type and Development Methods Type.....	124
Table 4.16 Development method and Measurement Methods.....	125
Table 4.17 Why Organization Does Not Use Measurements	126

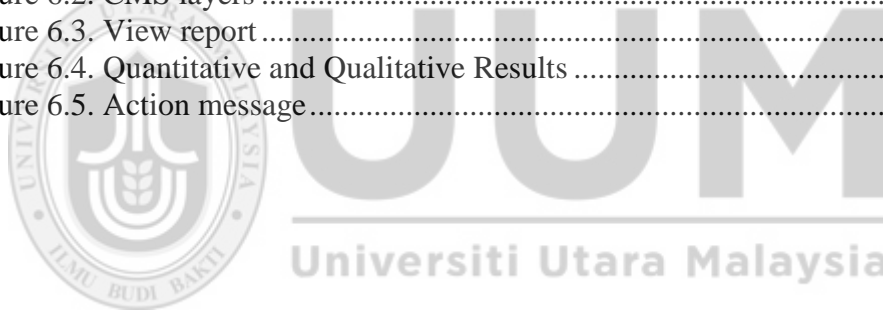
Table 4.18 Practices and SPSS Variable Name	126
Table 4.19 Internal Representations for the Degree of Acceptance.....	130
Table 4.20 Current Web Applications Development Practices	130
Table 5.1 The elements of the Extended Agile method	137
Table 5.2 Management Planning Sub Activities.....	145
Table 5.3 The MOGWD Methodology Team Structure	145
Table 5.4 Development Planning.....	148
Table 5.5 Monitoring Planning	153
Table 5.6 Goal Definition	154
Table 5.7 Requirements Questions and Metrics	158
Table 5.8 Designed Questions and Metrics.....	158
Table 5.9 Testing Questions and Metrics.....	159
Table 5.10 Practices Questions and Metrics	160
Table 5.11 Productivity Questions and Metrics.....	161
Table 5.12 Completeness Questions and Metrics	162
Table 5.13 Consistency Questions and Metrics	165
Table 5.14 Accuracy Questions and Metrics	167
Table 5.15 Tailorability Questions and Metrics.....	168
Table 5.16 Flexibility Questions and Metrics	168
Table 5.17 Compatibility Goal, Questions and Metrics.....	169
Table 5.18 Accessibility Questions and Metrics.....	169
Table 5.19 Applicability Question and Metrics	170
Table 5.20 Changeability Questions and Metrics	171
Table 5.21 Supportability Questions and Metrics.....	171
Table 5.22 Cost Questions and Metrics	172
Table 5.23 Quality Questions and Metrics.....	173
Table 5.24 Time Questions and Metrics	174
Table 5.25 Do phase.....	175
Table 5.26 Check Phase	179
Table 5.27 Development Process and Quantitative Metrics	181
Table 5.28 Data Collection for the Process Factors.....	184
Table 5.29 Act Phase.....	185
Table 6.1 Experts Answers related to comprehensiveness	189
Table 6.2 Experts Answers related to understandability.....	191
Table 6.3 Experts Answers related to feasibility	193
Table 6.4 Experts Answers related to the general overview Part.	194
Table 6.5 Required Modifications.....	195
Table 6.6 New list for quantitative metrics	199
Table 6.7 List of product backlog items.....	210
Table 6.8 The Main Outputs of Web Design Method.....	211
Table 6.9 The Outputs of the Do Planning Meeting	212
Table 6.10 Construction Action, Results	212
Table 6.11 Quantitative results	220
Table 6.12 Qaulitative results	223
Table 6.13 Representations of the achievement levels	225
Table 6.14 Validation results	225
Table 6.15 Baseline Models and Comparison Criteria.	228

Table 6.16 Yardstick validation230



List of Figures

Figure 2.1. Lean, Scrum and XP Combination Levels adopted from Temprado and Bendito (2010).	40
Figure 2.2. The evolution of the Web application methods adopted from Lang (2002)	45
Figure 2.3. GQM Method.	54
Figure 3.1. The extension process.....	89
Figure 3.2. The Delphi technique steps.....	95
Figure 4.1. Respondents Position and Experience	109
Figure 4.2. Software Philosophy	112
Figure 4.3. Measurements Stage and Application Domain.....	120
Figure 4.4. Measurement Stage and Development Method Type.....	123
Figure 4.5. Dendrogram	129
Figure 5.1. The improvements of Extended Agile Method	140
Figure 5.2. The MOGWD methodology	143
Figure 5.3. Web Design method	150
Figure 6.1. MO-PT structure	206
Figure 6.2. CMS layers	208
Figure 6.3. View report	217
Figure 6.4. Quantitative and Qualitative Results	218
Figure 6.5. Action message.....	219



List of Appendices

Appendix A Questionnaire.....	271
Appendix B Questionnaire Face Validity Cover Letter.....	280
Appendix C Expert cover letter.....	281
Appendix D Knowledge expert questionnaire.....	282
Appendix E Domain expert questionnaire.....	301
Appendix F Questions Objective, Content and Source.....	316
Appendix G Practices.....	319
Appendix H Tools.....	322
Appendix I Quantitative metric check list.....	323
Appendix J Qualitative metrics.....	328
Appendix K Validation form.....	333
Appendix L MO-PT.....	335



UUM
Universiti Utara Malaysia

List of Abbreviations

AM	Agile Modeling
ASD	Adaptive Software Development
AUP	Agile Unified Process
CFM	Crystal Family Methodologies
DSDM	Dynamic Systems Development Method
DT	Development Team
FDD	Feature-Driven Development
GOMM	Goal Oriented Monitoring Mechanism
GQM	Goal Question Metric Method
HDM	Hypermedia Design Model
LSD	Lean Software Development
MOGWD	Monitoring Oriented Agile Based Web Applications development Methodology
MT	Monitoring Team
MO-PT	Monitoring prototype tool
OOA	Object Oriented Analysis
OOHDM	Object-Oriented Hypermedia Design Methodology
PDCA	Plan, DO, Check and Act
PSM	Practical Software and Systems Measurement
QA	Quality Assurance
QAA	Quality Assurance Activities
RMM	Relationship Management Methodology
RUP	Rational Unified Process
SAD	Structure Analysis and Design
SOHDM	Scenario-based Object-Oriented Hypermedia Design Methodology
SPSS	Statistical Package for Social Science
SSADM	Structured System Analysis and Design Method
SSF	Small Software Firms
TDD	Test Driven Development

UWE	UML-based Web Engineering
WebML	Web Modeling Language
WSDM	Web Site Design Method
XP	Extreme Programming (XP)



UUM
Universiti Utara Malaysia

CHAPTER ONE

INTRODUCTION

1.1 Overview

This chapter provides an overview about this research. It describes the background and the problem statement of this research. Research questions, research objectives, and research scope are also presented in this chapter. Chapter One also presents the significance of the study, followed by the expected contributions of the research. This chapter ends with the chapters' organization of the thesis.

1.2 Background study

Over years, Web applications have been used by millions of organizations and companies to facilitate communication and information exchange with their customer in an economical manner. Hence, the development of such applications should be guaranteed in terms of the quality of the final product to prevent from failures. Based on the initial study, 80% of the small software firms are involved in the Web application development. A small software firm (SSF) is defined as an organization or company that has approximately 10 to 50 employees (Al-Tarawneh, 2013; Fayad et al., 2000; Hofer, 2002; Laporte et al., 2005; Richardson & Wangenheim, 2007).

Web applications in any SSF should be developed by following a systematic approach, taking into account Web application characteristics and the SSF limitations (Haung et al., 2008; Howcroft & Carroll, 2000). The systematic development can be achieved through the use of appropriate methodology.

Currently, there are several conventional development methods being introduced for developing Web applications in SSF such as waterfall, spiral and incremental. However, these development methods were found to be inadequate for developing the Web application in SSF as the unique characteristics of Web applications are poorly addressed (Altarawneh & El Shiekh, 2008; Haung et al., 2008; Okoli & Carillo, 2012). Moreover, these methods are too complex for small organizations (Altarawneh, 2013); cannot deal with high requirements changes (Moniruzzaman & Hossain, 2013; Okoli & Carillo, 2012); involve less customer collaboration (Moniruzzaman & Hossain, 2013; Okoli and Carillo, 2012); and not meant for building Web applications as they require a large number of resources such as skills and staff (Altarawneh & Shiekh, 2008; Okoli and Carillo, 2012).

To overcome the problems and limitations of the conventional development methods, the Agile methods have been introduced. The new methods concentrate on faster development life cycle, involve limited resources and engage more customer collaboration (Marinelarena, 2014). The most popular Agile methods that are suitable for SSF are Extreme Programming (XP) and Scrum (Spasibenko & Alite, 2009). However, these two methods are not mainly built for developing Web applications as these applications development required more emphasis on design and quality. Besides, the XP and Scrum are lacking in terms management and development practices respectively (Jyothi and Rao, 2011; Qureshi, 2011). Therefore, these methods need to be analyzed and enhanced for further improvements of their practices.

High quality Web application is a reliable, usable and well-designed product that delivered to the market within time, budget and shorter life cycle (Eldai et al., 2008; Haung et al., 2008). Furthermore, these characteristics are influenced by the quality of the process (Guceglioglu & Demirors, 2011; Kroeger et al., 2014; and Tyrrell, 2000). In order to fulfill these characteristics, the Web application development needs a systematic, well-managed, incremental and measurable development process (Eldai et al., 2008; Deshpande et al., 2002).

Therefore, to ensure that the Web applications have been developed using a systematic methodology, the management activities should be performed in parallel with the development activities (Abran et al, 2004). Greenfield and Short (2003) highlighted that one of the important management activities besides planning and controlling activities is the measurement. Performing a measurement mechanism during the development process has many purposes and one of them is monitoring (Abran et al., 2004).

Monitoring helps organizations to gain many benefits such as tracking the progress of the development process by giving feedback to the management and development team; finding and correcting errors during the development process and reducing risks of project failure (Ardimento et al., 2004; Briand et al., 1996; Esaki et al., 2012; Morasca, 1999; Solingen & Berghout, 2001; Tsai & Cheung, 1999; Tu et al., 2009). Therefore, the SSF that are involved in Web application development need to apply a new methodology to guide them during the development process, monitor the quality of the process and the final product, save resources and reduce the development time

and cost (Alesky et al., 2004; Baskerville & Pries-Heje, 2002; Costagliola et al., 2002; Murugesan et al., 2001; El-Sheikh & Tarawneh, 2007; Haung et al., 2008). Thus, this study aims to construct a monitoring oriented Agile based Web application development methodology for SSF.

1.3 Problem Statement

Web application development in SSF is currently facing various problems such as failure to deploy a proper development and measurement practices, complex design and unable to monitor the quality of the process and the product. These problems are discussed as follows:

- i. **Need to investigate the current Web application development and measurement practices in SSF.** Many researchers such as McDonald and Welland (2001), Cater-Steel (2004), Kirk and Tempero (2012) and El-Sheikh and Tarawneh (2007) highlighted the need of following best practices in developing Web applications in SSF to improve the productivity, reduce cost, minimize time and increase quality. The results of a survey conducted in UK by McDonald and Welland (2001) conveyed that there is a need to deploy the best practices in Web application development as the majority of the targeted organizations still used ad-hoc development process. In addition, the survey also pointed out there is a little attention paid to the measurement of the product quality. Similar work also done by Cater-Steel (2004) where the study found that most of the SSF in Australia failed to apply any standard software

development practices. However, the study only investigated the development and management practices instead of the measurement practices. Another study conducted by Kirk and Tempero (2012) also indicated that the majority of the SSF in New Zealand did not follow any standard development methods or development practices. Unfortunately, the study only focused on the organization and participant practices rather than the measurement practices. Another related study on the development and management practices conducted by El-Sheikh and Tarawneh (2007) indicated that the degree of applying these practices in the Jordanian SSF was very low. Again, this study only discussed on the development and management practices. Therefore, findings from these studies clearly shown that there is a need to investigate the current Web applications development and measurement practices in SSF.

- ii. **The importance of improving the design phase in Web application development.** The quality of Web application development depends on the deployment of the management and development practices. In addition, design is a very important phase in Web application development as it describes the content, navigation and interface and usability of the Web applications (Ginige & Murugesan, 2001; Tarafdar & Zhang, 2008). The key issue that may cause failure in the Web application development is poor design (Ginige & Murugesan, 2001; Haung et al., 2008; Mccarthy & Aronson, 2001; Tarafdar & Zhang, 2008). Unfortunately, XP and Scrum cannot meet the Web application design complexity as Scrum does not define any specific software

development techniques for design (Clutterbuck et al., 2009; Fernandes & Almeida, 2010; Jyothi and Rao, 2011; Moniruzzaman & Hossain, 2013; Qureshi, 2011) and XP only provides a simple design phase (Kumar & Bhatia, 2012; Pressman, 2009; Whiston, 2006). In order to overcome the drawbacks of the development practices in Scrum and the drawbacks of management practices in XP, many studies had combined XP and Scrum methods such as Mar and Schwaber (2002), Fitzgerald et al., (2006), Clutterbuck et al., (2009), Jyothi and Rao (2011) and Qureshi, (2011). These studies combined the Scrum with specific XP development practices. However, the combination process was not mainly built for developing Web application in SSF. Additionally, all of these studies still using XP design practices which were considered very simple in fulfilling the Web application design complexity. Therefore, there is a need for analyzing the development and management practices of XP and Scrum and enhancing the design phase with the Web design practices.

iii. **Need of monitoring the quality of the process and the product during the development.**

In order to get high quality Web applications in SSF, the process and product quality should be monitored. This can be accomplished by adopting a monitoring mechanism that incorporates the quantitative and qualitative measures. Unfortunately, XP and Scrum failed to provide any measurement mechanism that able to monitor the process and product quality (Berardi &

Santillo, 2010; Fritzsche & Keil, 2007; Javdani et al., 2012; Jiang & Eberlein, 2008; McCurley et al. 2008; Turk et al., 2002; Turk et al., 2005; Qumer & Henderson-Sellers, 2008). Furthermore, very limited studies were conducted to measure the Agile development process. For example, Kroeger et al., (2014) only determined the qualitative measurement of process characteristics without using a specific measurement method. Another study conducted by Kettelerij (2006) focused on the designing of a quantitative measurement program for software development, while Kulas (2012) used some metrics such as size, defects, requirement and design for measuring the quality of product in XP. Kunwar (2013) conducted a study that uses quantitative metrics to evaluate three XP practices which are Test Driven Development (TDD), pair programming and on-site customer. Most of the metrics mentioned in the previous studies were performed at the end of the iteration instead of covering the whole process. In order to have a successful measurement mechanism, the metrics should monitor the process and product quality. Apart from that, a successful measurement should use both the quantitative and qualitative goal oriented mechanism (Calero et al., 2005; Wangenheim et al., 2003; Murphy and Cormican, 2012). Consequently, none of the above studies achieved all these factors. Therefore, there is a need to enhance the XP and Scrum methods by integrating a measurement mechanism for monitoring the quality of the process and the final product.

SSF projects may fail by using poor methodologies (Alite & Spasibenko, 2008; Fritzsche & Keil, 2007; Esaki et al., 2012; Jyothi & Rao, 2011; Moniruzzaman &

Hossain; 2013; Whitson, 2006) because they cannot deal with the existing issues that faced by SSF such as limited resources (human, financial and experience), high changing requirements, tight deadlines, lack of customer collaboration, ineffective project management, lack of unique processes and methods, and lack of specific software process and quality monitoring measurement mechanism (Al-Tarawneh, 2013; Huang et al., 2008; Kirk & Tempero 2012; Pusatli & Misra, 2011; Tarawneh & Allahawiah, 2009).

Based on the above discussion, it is clear that the SSF need a new methodology for developing Web applications. The new methodology should consider these three important elements: the characteristics of the Web applications, the SSF and the limitations of the existing methods. This methodology will also consider the Scrum management practices, XP development practices, and improves the design phase. Finally, the methodology will provide a suitable measurement mechanism for monitoring the quality of the development process and the final product.

1.4 Research Questions

- What are the current Web applications development and measurement practices in SSF?
- How to enhance the design phase of the Extended Agile method?
- How to perform monitoring during the Web application development process?

- How to ensure that the proposed methodology can be effectively implemented in SSF?

1.5 Research Objectives

The main objective of this study is to develop a Monitoring Oriented Agile Based Web Applications Development Methodology for SSF.

In order to achieve the main objective, the following specific objectives are proposed:

- To investigate the current Web applications development and measurement practices in SSF.
- To enhance the design phase of the Extend Agile method.
- To construct quantitative and qualitative measurement metrics for monitoring the quality of the process and product.
- To evaluate the proposed methodology.

1.6 Research Scope

SSF play an important role that provides substantial growth to their countries economy. In Jordan, most of the software firms are small (El Sheikh & Tarawneh, 2007). The study was conducted in Jordan because of the following reasons; (i) most of SSF in Jordan involved in Web application development; (ii) high percentage of the SSF in Jordan are still using ad hoc development process; and (iii) high percentage

of the SSF in Jordan are not aware of applying the best Web application development practices (El Sheikh & Tarawneh, 2007).

A survey conducted in Jordanian small software firms to investigate the current practices items of using methods, practices and the issues that faced by these firms. The findings of the survey were used to construct a new monitoring oriented Agile based Web application development methodology for SSF. This methodology covers the development, management and monitoring process. The development process was performed by referring to the extended Agile method that was enhanced by incorporating a Web design method. The monitoring process was constructed by performing qualitative and quantitative metrics during the development process to ensure the quality of the final product. This study adapts Plan, Do, Check and Act method (PDCA) to perform the development and monitoring process together.

The new methodology verified was based on the comprehensiveness, understandability, and feasibility by the knowledge and domain experts. In the validation, one case study has been performed in Jordan validate the effectiveness of the new methodology in SSF.

1.7 Research Contribution

The contributions of this study are: Monitoring Oriented Agile Based Web Applications Development Methodology for SSF, Extended Agile method with Web design method, GOMM, and the survey results.

- Monitoring Oriented Agile Based Web Applications Development Methodology for SSF: The methodology provides descriptions of the activities, methods, practices, tools and team structure that should be considered when developing Web applications in SSF. In addition, the methodology adapts the phase of PDCA method to construct the development and monitoring processes and organize the methodology components.
- Extended Agile method with Web design method: The Extended Agile method was constructed by extending the Scrum methods with important XP elements. The design phase in this method is enhanced by adding the Web design method that uses design steps generated from the existing Web design methods. The resultant method will be used to guide the development process by the development team.
- The measurement mechanism uses the light weight goal question method to achieve the quality characteristics, and ensure the quality of the product and process. This method will be used to guide the monitoring process by the monitoring team.
- In addition, the study had conducted a survey on the current practices of Web application development and management in SSF. The survey contributes a collection of development and management practices that are recommended to be used by SSF practitioners. Moreover, the instrument used this survey can be adopted or adapted by the researcher in the field.

1.8 Significance of this Research

This research contributes towards the field of software engineering, Web application development, Agile based development and Web application measurement by providing a set of components: activity, methods, practices, tools, and team structure to deal with Web application characteristics and SSF limitations.

This study aims to benefit four stakeholders:

- **Small software firm's developers and managers**

This study will help this branch of companies to develop high quality Web applications using a systematic way, considering the time and budget limitations. This methodology will encourage SSF's Developers to gain high quality process and product. On the other hand, the new methodology gives an opportunity for managers to monitor the quality of the product in terms of its progress, cost, and quality characteristics. In addition, the process activities, practices, productivity and process quality factors can also be monitored. The monitoring process will be performed using the GOMM. Finally, the GOMM member would be able to produce a final report that describes all of the above goals.

- **Software engineer**

This study will provide the software engineers with a new methodology to develop Web applications in SSF using the GOMM method for monitoring the process and product quality. Furthermore, it gives a full set of current practices of

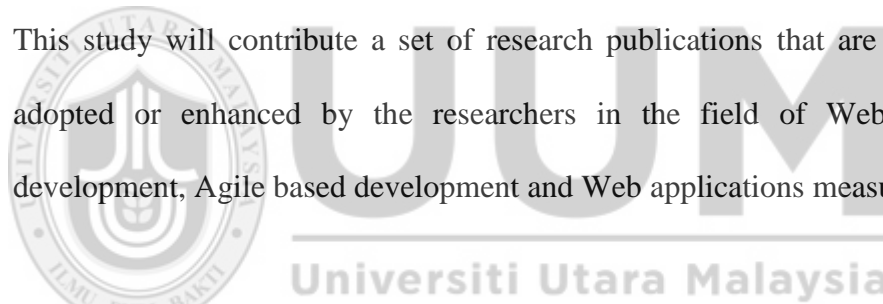
Web application development in SSF which are useful for any software engineer to adopt or adapt in different countries or study areas.

- **Evaluators**

This study will construct a quality metrics to verify and validate the new methodology. These quality metrics will be very useful for the evaluators of the same area or other software areas to ensure the acceptance of their new methodologies or frameworks.

- **Researchers**

This study will contribute a set of research publications that are useful to be adopted or enhanced by the researchers in the field of Web application development, Agile based development and Web applications measurement.



1.9 Organization of Chapters

Chapter Two: This chapter provides a description of the SSF characteristics and problems faced by the SSF as well as the development practices that must be followed by the SSF. In addition, the chapter gives an overview of the methods used for developing the Web applications such as conventional, Agile and Web design methods. It also includes the advantages and disadvantages of each method. Measurement methods were also discussed and analyzed in this chapter. The last section of Chapter Two presents the process quality factors that should be monitored during the development process.

Chapter Three: This chapter describes the research methodology used to achieve the research objectives. It provides explanations of the four stages that are used to construct a new methodology for developing the Web applications in SSF.

Chapter Four: This chapter firstly illustrates on how the survey approach was conducted. Secondly, it defines the data collection method (questionnaire) and describes its construction. Thirdly, this chapter presents the results of the survey conducted on the Jordanian SSF.

Chapter Five: This chapter presents the new methodology. In addition, this chapter defines the components of the new methodology in term of activities, methods, practices, tools and team structure.

Chapter Six: This chapter shows the evaluation process results using the expert review method and yardstick validation. The expert review was performed by verifying the completeness, understandability and feasibility of the proposed methodology using the Delphi technique rounds. The validation process was performed by two approaches case study and yardstick. The aim of conducting the case study is validate the effectiveness of the MOGWD methodology. The yardsticks validation was conducted to determine the strengths and weaknesses of the proposed methodology by comparing it with the baseline methods in the field.

Chapter Seven: This chapter concludes the finding of the research. It also highlights the research contributions and future work.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The aim of this chapter is to identify the best Web applications development and measurement practices for SSF. The chapter continues with the descriptions of the existing methods that are currently used by SSF for Web application development. A discussion of the investigation relating to the problem associated with each method is also included. In addition, the Web design and software measurement methods are also analyzed. This chapter ends by providing good methodology criteria of for SSF followed by a summary of the chapter.

2.2 Related studies on SSF

SSF represent a high percentage of firms in most countries around the world (Richardson & Wangenheim, 2007). The definition of these firms depends on the size of the firm. Thus, there is no one fixed size to distinguish the SSF among different countries. Nevertheless, there are some studies referred to the number of employees in SSF as less than 50 (Carter-Steel, 2001; Fayad et al., 2000; Rocha et al., 2007; Sulayman & Mendez, 2010; Wangenheim et al., 2003). On the other hand, Laporte et al. (2005) determined the size of SSF to be less than 60 employees. Furthermore, an empirical study conducted in Australia by Hofer (2002) concluded that the size of SSF is between 10 to 50 employees. Therefore, it can be concluded that the size of

SSF ranges from 10 to 50 employees. This range is used in this study to refer a small firm.

2.2.1 SSF characteristics and problems

SSF have many characteristics. These characteristics differ from a country to another. Even though there are no specific criteria in the world to categorize SSF, a number of literatures had pointed out some common characteristics of these firms. Alexandre et al. (2006) and Hofers (2002) stated that SSF comprised of the following important characteristics: frequent project meetings, quality management and teamwork, direct interaction with customers in the development process work, newest technology utilization, dynamic and flexible company and customer feedback deliberation.

On the other hand, SSF faced many problems in the software development and management. Table 2.1 described several studies that conducted in SSF and highlighted the problem that they faced.

Table 2.1

Problems Faced by SSF

Study	Description	Problems	Country
Fayad et al. (2000)	The paper discusses the major software engineering issues that face SSF.	<ul style="list-style-type: none"> - Limited staff. - High changing requirements. - No effective measurement of time and cost. 	Not determined
Hofer (2002)	The primary goals of this paper is to find out the characteristics of SSF, and identify the Software development and management issues	<ul style="list-style-type: none"> - Customer collaboration. - Project management. - Changing requirements. - Limited resources. - Lack of unique processes and methods. 	Australia
Wangenheim et al. (2003)	This paper discussed SSF issues that related to software measurement.	<ul style="list-style-type: none"> - Lack of experience. - Lack of employees. - No specific measurement 	Brazil

		mechanism for monitoring the process quality.	
Dangle et al. (2005)	The paper identified the role of process improvement in the context of a small organization	- Limited resources (human and financial).	USA
El Sheikh and Tarawneh (2007)	This study shows the level of Web engineering best practices adoption in the Jordanian small firms	- Lack skills and experience. - Lack of staff. - Limited budget. - Project management issues.	Jordan
Huang et al. (2008)	This paper discussed the issues of Web application development in SMEs.	- Lack of software processes. - Limited resources. - High changing requirements. - Small budget. - Tight deadlines.	UK
Di'az-Ley et al. (2008)	This discussed the SMEs limitations in performing measurement program.	- Limited human resources. - No specific software measurement mechanism.	Not determined
Tarawneh and Allahawiah (2009)	This study determined the current practices of industry participants, and in devising improvement initiatives which are feasible for small firms.	- Use an ad hoc process. - Use an ad hoc metrics.	Jordan
Ribeiro and Fernandes (2010).	This paper presented a model to prioritize available management systems to help SMEs address the challenge of today's market competition.	- Lack of technical skills. - Shortage of finance.	Not determined
Pusatli and Misra (2011)	This study aims to evaluate the appropriate implementation of measurement programs in SMEs.	- Use ad hoc measurement program. - Limited number of employees.	Turkey
Kirk and Tempero (2012)	This paper aims to Understand the practices used by SSF.	- Ineffectiveness in requirements practices. - Lack of applying development practices. - No standard development method.	New Zealand
Al-Tarawneh (2013)	This study presents the problems faced by the SSF through constructing a software development process improvement framework for SSF.	- No specific software process model. - Changing project requirements. - Limited resources. - Customer collaboration problem. - Project management problems.	Jordan

As concluded from Table 2.1, most of the problems that face SSF include: limited resources (human, financial and experience), high changing requirements, tight deadlines, lack of customer collaboration, ineffective project management, lack of unique processes and methods, and lack of specific software process and quality monitoring measurement mechanism. Therefore, for any firm that has limited resources, it is recommended to use an easier development process that is suitable for smaller teams. For those facing high changing requirements issue can opt an iterative style process. To tackle the customer collaboration and project management issues, SSF may use a process that emphasizes on customer involvement and management respectively. As for the tight deadlines and no specific measurement mechanism can be managed by applying certain measurement mechanisms to monitor the product and process quality.

2.2.2 Studies on the practices in SSF

A best practice is defined as “a management or technical practices that has consistently demonstrated and should be deployed to improve one or more of productivity, cost, schedule, quality user satisfaction and predictability of cost and schedule” (Withers, 2000).

Software engineering best practices were produced by the software council sponsored by the Department of Defense (DOD) to improve the success of large software projects (Brown, 1999). These practices had been categorized by Software Program Managers Network (SPMN, 1999) into three categories, namely project management, design and development, and quality of product.

SSF is considered as a special case representing the software industry, which differs from other large companies, especially in the application of development approaches. El Sheikh & Tarawneh (2007) and Bucci et al. (2001) stated that a high percentage of SSF are not aware of performing the software best practices. Furthermore, there is a lack of well-defined development process for SSF (Alexandre et al., 2006; Hofers, 2002; Knauber et al., 2000; Tarawneh & Allahawiah, 2009).

There are a number of recent empirical studies that has recommended a set of practices to be performed by the SSF while developing their software towards achieving a high quality product within the time and budget constraints. A survey conducted by Azuma and Mole (1994) highlighted the differences between the development practices used by the European firms compared to the Japanese companies, whilst Blackburn et al. (1996) concluded that the companies in the United State, Japan and Western Europe are using the same practices. Other researchers focused on a specific location, for example, a survey on software process adoption in Singapore (Tan & Yap 1995).

The most widely reported and well known survey of best practice in Europe was that conducted by the European Software Institute (ESI) (Dutta et al., 1998). This survey (ESI, 1997) defined best practice as “a management practice that is commonly recommended as excellent and suggested by most practitioners and experts in the field”. In addition, the survey categorized the development best practices into five categories: organizational issues, standards and processes, metrics, development process control as well as tools and technology.

Nevertheless, the most recent studies that focused on the development of Web applications and software are shown in Table 2.2

Table 2.2

Previous Studies in Web and Software Development Practices

Study Name	Respondent/data collection method	Goal	Outcomes
A survey of Web engineering in practice (McDonald & Welland, 2001b)	Web developers/interview.	To identify the major issues relating to the development of Web based systems.	There is an indication of the importance of deploying certain development practices as the majority of the targeted organization still used ad-hoc development process.
An evaluation of software development practice and assessment-based process improvement in small software development firms (Cater-Steel, 2004).	Software developer and managers/questionnaires.	To provide a much better understanding of practices used by small software development firms.	Most of the practices have not been applied by the targeted organizations.
Northern Ireland (NI) software industry survey (McCaffery et al., 2004)	Managers, technical directors and quality managers/interview.	To achieve an accurate understanding of the quality of the software development organizations in NI.	General awareness of various standards that can be applied to software process is still limited.
A survey of Web engineering practice in small Jordanian Web development firms (El Sheikh & Tarawneh, 2007).	Web developer/questionnaires.	To show the level of Web engineering best practices adoption in the Jordanian small firms.	The degree of applying these practices was very low.
Software development practices in New Zealand (Kirk & Tempero, 2012)	Developers/online questionnaire.	Understanding the practices used by the New Zealand software organizations in developing software.	The majority of the organizations did not follow any standard development methods or set of development practices.
A survey on the current practices of software development process in Malaysia (Baharom et al., 2006)	Managers, technical directors and developers/questionnaire.	To determine the current practices of software development process in Malaysia.	Most of the targeted organizations were not aware of deploying good software development practices and standard that cause low quality product.

Referring to Table 2.2, it is obvious that three out of the six studies were conducted on small sized organizations. These studies are those of Cater-Steel (2004), Kirk and Tempero (2012) and El Sheikh & Tarawneh (2007). The common result from all these studied showed that SSF failed to apply specific best or recommended practices. Therefore, there is a need to conduct a study to investigate the current Web application development and measurement practices in SSF. Before conducting this study, the best practices for developing Web application in SSF should be identified. The Web application development practices were identified by referring to: El Sheikh and Tarawneh (2007) and McDonald and Welland (2001a). The rest of the practices related to the project and quality management were determined from various authors such as Deshpande et al. (2002), Haung et al. (2008), Redouane (2002) and Wu and Offutt (2002). These practices are shown in Table 2.3.

Table 2.3

Web Applications Development Best Practices for SSF

Practice	Description
1- Short development life-cycle times	The Web application development process should deal with time pressure because the average time of Web application development life cycle is less than three months (McDonald & Welland, 2001a).
2- Delivery of bespoke solutions	The development process of Web applications should not only deal with software components, however, it should cope with data interdependencies (McDonald & Welland, 2001a).
3- Multidisciplinary development teams	The web application development process should clarify team member's roles and responsibilities. This team should know that the process is semi-formal with little documentation (McDonald & Welland, 2001a; Haung et al., 2008; El Sheikh & Tarawneh , 2007).
4- Analysis and Evaluation	Web application development process answers these questions: a.) Why are we going to develop a Web application? b.) What problems or goals will the Web application address? c.) How will we know if the solution addresses these problems or goals? d.) How will we measure and validate the deliverables?

	(McDonald & Welland, 2001a).
5- Requirements management	<p>1- Requirements should be collected from the user or/and manager. There will be no intermediate person between the source of requirement and the developer (Haung et al., 2008; Redouane, 2002).</p> <p>2- Translate informal requirements to a formal or semi-formal specifications using any formal notation which both of the teams are familiar with (Redouane, 2002; Withers, 2000).</p> <p>3- Web application development requires an iterative process to cope with change requirement (Haung et al., 2008).</p>
6- Testing	<p>1- Generate test cases based on the requirement specification performed for every component (Redouane, 2002).</p> <p>2- All components of the Web applications such as page, code, site, navigation, must be tested by testing cases generated according to the requirements specifications. Services such as HTML and XHTML must be tested to ensure all of the Web application components are tested (Redouane, 2002; Deshpande et al., 2002).</p> <p>3- In order to avoid biases; testing process must be carried out by people who are not involved in the development process (Redouane, 2002).</p>
7- Maintenance	To ensure proper maintenance and deliverables update, it is very necessary for a developer to build a well-documented system that able to determine how content maintenance should be carried out and which policies will be used for that (McDonald & Welland, 2001a; Deshpande et al., 2002).
8- Project management	<p>1- Project management practices such as risk management and software measurements is important to avoid failure and any negative impact on the final product (El Sheikh & Tarawneh, 2007; Brown, 1999).</p> <p>2-A proper project plan that include budget and time estimation (McDonald & Welland, 2001a).</p>
9-Quality management	1-Developers should pay more attention to the quality management and standards such as usability and user interface design as it is important to influence the maintenance and improve final product and organizational issues (Haung et al., 2008; El Sheikh & Tarawneh, 2007; Wu & Offutt, 2002).

Based on Table 2.3, SSF should pay more attention to certain practices during Web applications development in order to gain high quality product within the available resources. The practices should relate to the development process as well as the team, and quality management. Having a systematic and disciplined development process,

helped in clarifying the practices related to requirement, test, design and maintenance that will ensure the quality of the final product (McDonald & Welland, 2001b). Therefore, these relevant practices should be deployed to ensure the quality of the Web application product.

Requirement practices are about gathering requirement from those who requested for software development and it's related to the way of collecting them. Testing practices relate to individuals responsible for carrying out the testing process, identifying the components that have been tested and designing a simple approach during the iteration without ignoring the complexity design of Web applications.

Regarding the team structure, all the development team members should have a clear understanding of their roles and responsibilities.

The project management activities are very important to be used in parallel with the software development. Using such activities as planning, coordinating, measuring, monitoring, controlling, and reporting will not only activate the development process, but also make everything inside the process activities, roles and deliverables clear and manageable (Abran et al., 2004).

The quality management involves many factors depending on the development environment. For example, as the environment of a Web application is different than that of the traditional software development, the factors that influence the quality of Web application varies. Therefore, the main factors that influence the quality of Web

applications are security, product reliability, usability and maintainability (Wu & Offutt, 2002; Lilburne et al., 2004).

The above mentioned practices will be taken as a baseline to construct a questionnaire to investigate the current development and measurement practices in SSF. Beside the practices, the SSF should also follow a particular discipline method to guide their development process. These methods are discussed in the next section.

2.3 Existing development methods

Many methods for developing Web application in SSF have been proposed such as the conventional, Agile and Web design methods.

The conventional methods are popular software development methods that have been developed long before the Agile method. Among the examples of the conventional methods are waterfall (Royce, 1970), incremental (Basili & Turner, 1975), v-model (appeared in the Hughes Aircraft circa 1982), prototyping (Floyd, 1984) and spiral (Boehm, 1988). These conventional methods are considered as a heavy weight, planned, driven, heavy testing and strict control methods (Imreh & Raisinghani, 2011).

The Agile development methods are proposed to overcome the limitations of the conventional methods. Pressman (2009) concluded that Agile methods include: XP, Scrum, Crystal Family Methodologies (CFM), Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Feature-Driven

Development (FDD), Lean Software Development (LSD), Agile Modeling (AM) and Rational Unified Process (RUP) or Agile Unified Process (AUP). Table 2.4 describes the Agile development methods advantages and disadvantages.

Table 2.4

Agile Development Methods

Method	Advantages	Disadvantages	Source
XP	<ul style="list-style-type: none"> - The most common method in Agile software development. - It is considered as a collection of development practices. 	<ul style="list-style-type: none"> - Lack of management practices. - Simple design phase. 	(Beck, 1999; Boehm, 2006; Stojanovic et al., 2003; Qureshi, 2011; Väänänen, 2008)
Scrum	<ul style="list-style-type: none"> - One of the popular Agile software development methods that focus on management 	<ul style="list-style-type: none"> - Lack of development practices. 	(Boehm, 2006; Larman, 2003; Schwaber & Beedle, 2001; Qureshi, 2011; Väänänen, 2008).
CFM	<ul style="list-style-type: none"> - Allows adoption of other Agile methods. - No specific process activities, but selected methodologies based on the project demands. 	<ul style="list-style-type: none"> - Suitable for one team in one room. - Lack of validation. - Not suitable for life critical system. 	(Abrahamsson et al., 2002; Stojanovic et al., 2003; Väänänen, 2008).
AM	<ul style="list-style-type: none"> - Used for modeling and documentation of software-based systems. - AM has no specific process activities because it focuses on practices and cultural principles. 	<ul style="list-style-type: none"> - It is not sufficient by itself, as it should be supported by other models such as UML models. 	(Abrahamsson et al., 2002; Stojanovic et al., 2003).
ASD	<ul style="list-style-type: none"> - Provides a framework as guidance for projects in preventing from falling into chaos. 	<ul style="list-style-type: none"> - ASD is more about concepts and culture than the software practice. 	(Abrahamsson et al., 2002; Awad, 2005; Stojanovic et al., 2003)
DSDM	<ul style="list-style-type: none"> - Updates the product functionality rapidly. - Conforms product to meet the time and resource constraints. 	<ul style="list-style-type: none"> - More easily applied to business system than engineering or scientific applications. 	(Abrahamsson et al., 2002; Stojanovic et al., 2003; Väänänen, 2008)
FDD	<ul style="list-style-type: none"> - Focuses on the designing and building phases iteratively. 	<ul style="list-style-type: none"> - It did not cover the other software development phases. 	(Abrahamsson et al., 2002).
RUP	<ul style="list-style-type: none"> - An iterative approach to object oriented. - Use cases for modeling the requirements. 	<ul style="list-style-type: none"> - Fail to provide any clear implementation guidelines. 	(Abrahamsson et al., 2002; Väänänen, 2008).

LSD	- LSD has adapted the principles of lean manufacturing to the world of software engineering	- More Flexibility will quickly lead to a development that loses sight of its objectives and which never finishes.	(Pressman, 2009).
------------	---	--	-------------------

The development of Web applications in SSF is not an easy task because of the characteristics of Web applications and the limitations of SSF. Therefore, conventional and Agile development methods were compared in this study based on specific criteria collected from the SSF' limitations and Web application characteristics. Five criteria were selected for the comparison: fit to 10-50 size, complexity, flexible to change, customer collaboration and quality assurance measurement mechanism (QAMM). The results of comparing the conventional methods were extracted from Awad (2005), Imreh and Raisinghani (2011), Koblenz (2003), Naqvi (2007), Munassar and Govardhan (2010), Okoli and Carillo (2012), Pressman (2009) and Spasibenko and Alite (2009). In addition, the results of comparing the agile development methods were extracted from Abrahamsson et al.(2002), Beck (1999), Larman (2003), Lindstrom and Jeffries (2004), Pressman (2009), Salo (2006), Schwaber and Beedle (2001), Stojanovic et al. (2003) and Väänänen (2008). Table 2.5 shows the comparison between the conventional and Agile development methods.

Table 2.5

Software Development Methods Comparison

Criteria		Fit to 10-50 size	Complexity	Flexible to change	Customer collaboration	QAMM
Method						
Conventional methods	Waterfall	×	×	×	×	×
	Incremental	×	√	<	<	×
	V- model	×	√	×	×	×
	Prototyping	×	√	<	<	×
	Spiral	×	√	<	<	×
Agile Development Methods	XP	√	<	√	√	×
	Scrum	√	<	√	√	×
	CFM	×	<	√	√	×
	AM	×	√	√	√	×
	ASD	×	√	√	√	×
	DSDM	×	<	√	√	×
	FDD	×	√	√	√	×
	RUP	×	√	√	√	×
	LSD	×	<	√	√	×

(√) means satisfy the criterion, (×) means not satisfied the criterion and (<) means partially satisfy the criterion

Results in Table 2.5 convey that all conventional methods are considered not suitable for the SSF' size. The waterfall and V model cannot meet requirement changes as the methods used linear development style and lack of customer collaboration. The V model and incremental method are considered as complex development methods.

The prototyping and spiral methods are not suitable for the size of SSF as they are complex, less flexible to change, less customer collaboration and do not use any measurement mechanism to ensure the quality of the process and product. This clarifies that conventional method is not suitable for developing Web applications in SSF.

Table 2.5 illustrates that all Agile development methods concentrate on customer collaboration and requirement changes. However, these methods have not used any measurement mechanism to ensure the quality of the process and product. Consequently, regarding the size criterion, XP and Scrum are the only methods that satisfy and fit the size of SSF. In addition, four out of nine Agile development methods are found to be too complex to be used in real life. These methods are the AM, ASD, RUP and FDD. The XP, Scrum, CFM, LSD and DSDM methods are found to be less complex. These results confirm that the most Agile development methods that can be used for developing Web application in SSF are XP and Scrum. This is relevant to the findings of Spasibenko and Alite (2009); Theunissen et al. (2005) and Väänänen (2008). Thus, this study will focus on these two development methods.

2.3.1 XP and Scrum analysis

This section aims to analyze the XP and Scrum based on on their similarities and differences. The XP and Scrum are similar in using iterative development style which is recommended for small teams. Both methods also do not have any design method and measurement mechanism. However, XP and Scrum have certain differences in terms of the development process, project management, requirements, testing, design and team structure (McDonald and Welland, 2001b; Deshpande et al., 2002; Redouane, 2004; Abran et al., 2004; Haung et al., 2008; Qumer & Henderson-Sellers, 2008). These differences are used as criteria for comparing the XP and Scrum in the next sections.

2.3.1.1 The Development Process

In order to ensure the quality of the Web application, the development process should be performed by using a systematic and disciplined methodology which clarifies the roles and responsibilities for each team member (McDonald & Welland, 2001b). The comparison between the Scrum and XP in terms of the development process is done using sub-criteria which are considered as the common Agile development practices (Abrahamsson et al., 2002; Abrantes & Travassos, 2011; Fernandes & Almeida, 2010; Qumer and Henderson-Sellers, 2008). This study uses these sub-criteria to confirm whether both methods performed the common Agile development practices. The sub-criteria are shown in Table 2.6.

Table 2.6

Development Process Criteria

Sub criteria	XP	Scrum
Iterative and rapid development style.	Yes	Yes
Short releases (after the first iteration, new versions are released even daily, and at least monthly).	Yes	No
Metaphor (guides all the development by describing how the system works).	Yes	No
Simple design (unnecessarily complexity and extra code are removed immediately).	Yes	No
Refactoring (removing duplication and adding flexibility).	Yes	No
Pair programming (two programmers + one monitor)	Yes	No
Collective ownership (anyone can change the code at any time)	Yes	No
On-site customer (customer has to be available full time for the team).	Yes	No
Coding standard (coding rules must be followed by the programmers).	Yes	No
Every day meeting.	No	Yes

Every iteration meeting	No	Yes
-------------------------	----	-----

Table 2.6 illustrates that XP satisfies the development process sub-criteria compared to Scrum. However, both are recommended to be used by SSF. The development style for both methods is iterative and rapid. Table 2.7 shows the results of the other comparison criteria such as project management, requirement, testing, design and team structure. Table 2.7 is extracted from Abrahamsson et al. (2002), Berardi & Santillo (2010), Fernandes & Almeida (2010), Fritzsche & Keil (2007), Jiang & Eberlein (2008) and Qumer and Henderson-Sellers (2008).

Table 2.7

XP and Scrum Comparison Table

Criteria	Sub-Criteria	Methods	
		XP	Scrum
Development Process	From Table 5.1	More satisfying	Less satisfying
Project Management	Management Practices	Planning Game	Scrum Master Sprint meeting Daily meeting
	Measurement Mechanism	No	No
Requirement	Requirement gathering practices	User stories	Product backlog
	Requirement repository for trace and reuse	No	No
	Customer involvement	On-site customer practices	By the product owner
Testing	Testing technique	TDD	No
Design	Design approach	Code centered	No
	Code style	Clean and simple	No
	Design prototype	No	No
Team structure	Team Size	3- 20	5- 9
	No. of teams	1 team	Multiple teams

2.3.1.2 Project Management

Project management is defined as the “activities that must be performed during the development process are planning, coordinating, measuring, monitoring, controlling, and reporting which activates the development process on one hand and clarify the need of measurement on the other hand“ (Abran et al., 2004).

Based on the project management criteria, it is obvious that Scrum satisfies the project management criteria better than XP due to the use of the management practices namely, scrum master, sprint meeting and daily meeting during the development process as shown in Table 2.7. However, both methods do not have a specific measurement mechanism to ensure the quality of the product and process. The use of measurement during the development process is important to reduce defects, minimize time and rework of the development life cycle (Kettelerij, 2006; McCurley et al., 2008).

2.3.1.3 Requirements

The way of collecting requirements and from whom the developers collect it affects the speed of the development. In addition, the collection method should deal with continuous changed Web application requirements (Haung et al., 2008; Redouane, 2002).

Both the XP and Scrum are good on the requirement gathering techniques as they use user stories and product backlog that can ensure faster development process. In

addition, XP and Scrum differ in terms of the customer involvement practice where XP insists to have the customer onsite and Scrum uses a product owner who acts as the customer since it is difficult to have that customer onsite all the time. However, both methods do not have the requirements reuse and traceability.

2.3.1.4 Testing

The testing process is very important to ensure the product quality. Therefore, it is necessary to deploy testing practices during the development by a separated testing team (Redouane, 2002; Deshpande et al., 2002).

Referring to Table 2.8, XP is better than Scrum on performing the testing practices by using the TDD technique which ensures that all implemented features must be covered by unit tests. However, nothing is mentioned about the testing practices in Scrum.

2.3.1.5 Design

The design phase of the Web application development should be simple in terms of its iteration process and use a quick prototype to cope with time pressure and high maintenance (McDonald & Welland, 2001b; Qumer & Henderson-Sellers, 2008).

Table 2.7 illustrates that the design approach used in the XP and Scrum is code and design centric respectively. The XP coding style is cleaner and simpler because it is using pair programming and simple design practices. Unfortunately, there is nothing mentioned about the Scrum coding style because it is considered as management

framework. However, both methods do not have a design prototype to meet the complexity of the Web application design.

2.3.1.6 Team Structure

The results attained from Table 2.7 show that the XP is created to serve one team on a project ranging from 3 to 20 team members, whereas Scrum can be used by multiple teams ranging from 5 to 9 team members.

2.3.1.7 Comparison Results

The results attained from Table 2.6 and Table 2.7 illustrate that the XP only concentrates on the development and fail to apply any management practices. Scrum, on the other hand, concentrates on the management practices and fail to include any development practices such as testing, designing and coding.

Both the XP and Scrum do not have a measurable mechanism to ensure the quality of product and process. At the same time, both methods do not use the requirements reuse and traceability and do not have any design method in dealing with the design complexity of Web applications. However, the XP is still performing the testing using the TDD whilst the Scrum has not applied any testing method.

Many authors suggested and recommended to combine the XP and Scrum in order to fulfill the management and development practices. The next section clarifies the number of these studies and implies the limitation and differences.

- Previous Studies on Combining XP and Scrum

Many authors recommended combining the XP and Scrum to cover the development and management sides (Abrahamsson et al., 2002; Beck, 1999). Furthermore, XP and Scrum focus on the same organization brands (small and medium enterprise). This makes the combination fruitful for developers and manager in the industry. These previous studies integrated between the two methods in terms of practices. However, these studies did not highlight the relationship between the practices and Agile principles.

Agile principle is defined as “Basic truths and laws that are derived from assumptions (values) and provide a foundation upon which assumptions (values) are based” (Turk et al., 2005). Twelve principles were generated based on these values. Table 2.8 shows each principle and its symbols used in this study.

Table 2.8

Agile Principles Symbols

Agile principles	
Principle	Principle symbol
“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software”	P1
“Welcome changing requirements, even at the later stage of the development. Agile processes harness change for the customer's competitive advantage”.	P2
“Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale”.	P3
“Business people and developers must work together daily throughout the project”.	P4
“Build projects around motivated individuals. Give them the environment and support their needs, and trust them to get the job done”.	P5

“The most efficient and effective method of conveying information to and within a development team is face-to-face conversation”.	P6
“Working software is a primary measure of progress”.	P7
“Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely”.	P8
“Continuous attention to technical excellence and good design enhances agility”.	P9
“Simplicity--the art of maximizing the amount of work not done--is essential”.	P10
“The best architecture requirements and design emerge from self-organizing team“.	P11
“Team reflects how to become more effective, then tunes and adjust its behavior accordingly”.	P12

The twelve principles are considered as a set of policies and rules that should be supported by processes that claimed to be “Agile” (Turk et al., 2005). Therefore, any Agile development method should support these principles.

Turk et al. (2005) describe the relation between the twelve principles by mentioning that principles should be supported by practices and both practices and principles were built based on the Agile values. Table 2.9, which were extracted from Visconti and Cook (2004), relates and maps the important practices of the XP and Scrum and Agile principles.

Table 2.9

XP, Scrum practices and Agile principles mapping

Practices		Principles related
Scrum practices	Iteration planning meeting	P6, P10
	Daily meeting	P6, P10
	Iteration review meeting	P1, P2,P3,P5, P7,P8, P12
XP practices	Pair programming.	P5 , P6, P9 , P10
	TDD.	P1 , P9, P12
	Simple design.	P9
	Refactoring.	P9

Collective ownership.	P5
Coding standard	P9
Continuous integration.	P1, P2,P3,P7,P8,P12
Metaphor	P4
Small release	P1, P2,P3,P7,P8,P12

Table 2.9 points out that the most important Scrum practices is the iteration review meeting, which influences the application of seven principles (P1, P2, P3, P5, P7, P8 and P12) followed by the iteration planning meeting and daily meeting which influence the application of two principles (P6 and P10). The most important XP practices are small release and continuous integration, which influence six principles (P1, P2, P3, P7, P8 and P12), followed by pair programming which influences four principles (P5, P6, P9 and P10). The test driven development (TDD) practices affect three principles (P1, P9 and P12), whilst simple design, refactoring, and coding standard influences the same principles (P9). However, metaphor influences (P4) and collective ownership influences the application of (P5).

The importance of the XP and Scrum practices will be determined based on Table 2.9 and from the previous studies conducted to discuss the combination between XP and Scrum (Mar & Schwaber, 2002; Fitzgerald et al., 2006; Clutterbuck et al., 2009; Qureshi, 2011; Jyothi & Rao, 2011; Temprado & Bendito, 2010).

Mar and Schwaber (2002) merged the XP practices with the Scrum process. The combination process is performed following these steps:

- Determine the similarities of both methods.

- Use the Scrum process activities (planning, development and post-game), roles (Scrum master, product owner and development team) and meeting (daily Scrum, Sprint planning meetings and others).
- Use product and sprint backlogs to represent whole system requirements and the iteration requirements respectively.
- Use seven XP practices, namely: simple design, TDD, continuous integration, refactoring, pair programming, collective ownership and coding standards.

Fitzgerald et al., (2006): this study was conducted to investigate the tailoring of the XP and Scrum according to more than three years of developers experience inside the Intel Shannon company. The process of tailoring the two methods was done through the following:

- Use the Scrum process as a baseline and give justifications.
- Divide the project into two levels; organizational level (Scrum) and project level (XP).
- Use the same Scrum activities (planning, development and post-game phase).
- Map between the used and unused activities or practices of both Scrum and XP development methods.
- Based on the usage of the XP practices in the company, six XP practices were selected.
- The six selected practices were pair programming, TDD, collective ownership, refactoring, coding standard and simple design.

The tailoring process was conducted according to the method engineering theory that discussed in Brinkkemper (1996).

Clutterbuck et al. (2009): this study examined the application of the Agile development methods by small and medium enterprise developers. The tailoring process was done according to the Agile development practices that have been assessed and evaluated by the developers. The assessment process was conducted by:

- Interviewing the project members (managers and developers).
- Assessing the activities and practices of the two methods based on five ordinal levels (strongly helpful, helpful, improvable, difficult and not workable).
- Indicating that the results shown by all the Scrum practices are strongly helpful. Therefore, the Scrum method will be used as a baseline for the combination.
- Adding the seven XP practices to the combination method, namely: simple design, TDD, refactoring, pair programming, collective ownership, continuous integration and coding standard.

Qureshi (2011): this study indicated that the XP and Scrum have various drawbacks and limitations. Therefore, the integration of the two methods is very useful for the developers and managers to get high quality product and use mature process that covers both sides of management and development.

The integration was done in this study through the following steps:

- Compare XP and Scrum to determine their shortcomings.

- Take the Scrum process as a baseline.
- Select the suitable XP practices such as the simple design, collective ownership, pair programming, coding standards, TDD, continuous integration and refactoring.
- Integrate the Scrum process with the selected XP development practices.

Jyothi and Rao (2011). The study concluded that the most common Agile development methods to be combined together are the XP and Scrum. The process of combining the two methods was performed in the following steps:

- The process created based on the Scrum process.
- The Scrum sprint used iteration to produce a new increment.
- The Sprint included a traditional phase of software development such as requirement, analysis, design and evolution.
- The XP practices that had been added to the Scrum sprint are refactoring, pair programming, collective ownership and continuous integration.
- The functional testing was performed by the customer at the end of each iteration.

Temprado and Bendito (2010), The aim of this study is to construct a new framework that implements Lean practices in combination with Scrum and XP. The process of combining the three methods was performed by the following steps:

- Use the XP on the developmental level for fast development and high quality coding.

- Scrum on the project level is to increase the communication and reduce risk by dealing with requirements changes, splitting system into tasks and conducting Scrum meetings.
- Lean is a concept that must be applied to a higher level of the organizational level. The lean is used for removing waste, adding value and learning continuously in every process. By applying the lean principles the organization is able to change its thinking and development practice. The three levels of the combination are shown in Figure 2.1.

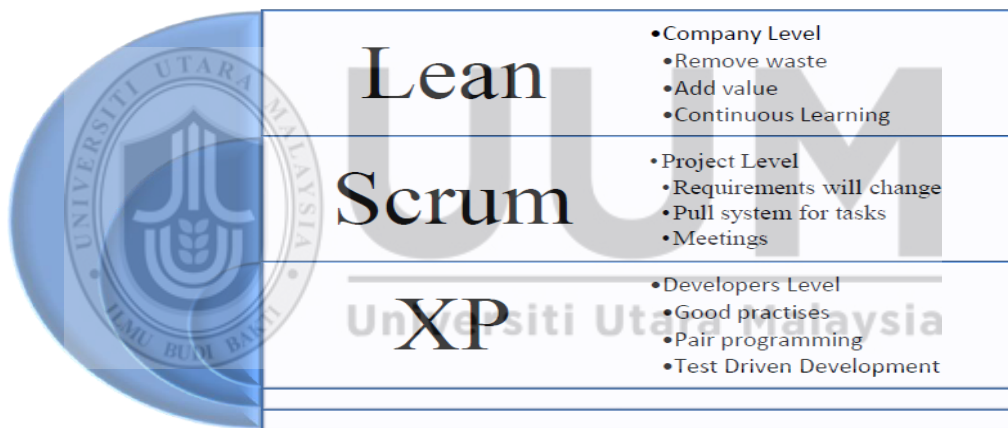


Figure 2.1. Lean, Scrum and XP Combination Levels adopted from Temprado and Bendito (2010).

Table 2.10 shows the integration of the XP and Scrum methods that had been done in previous.

Table 2.10

XP and Scrum Combination from Previous Studies comparison.

Study name		Study one Mar and Schwaber (2002)	Study two Fitzgerald et al. (2006)	Study three Clutterbuck et al. (2009)	Study four Qureshi (2011)	Study five Jyothi and Rao (2011)	Study six Temprado and Bendito (2010)
Criteria							
Methods		Scrum and XP	Scrum and XP	Scrum and XP	Scrum and XP	Scrum and XP	Lean + Scrum +XP
Practices	Scrum	Product backlog, sprint backlog, sprint, effort estimation, sprint planning meeting, daily Scrum meeting and sprint review meeting.	Product backlog, sprint backlog, sprint, effort estimation, sprint planning meeting, daily Scrum meeting and sprint review meeting.	Product backlog, sprint backlog, sprint, effort estimation, sprint planning meeting, daily Scrum meeting and sprint review meeting.	Product backlog, sprint backlog, sprint, effort estimation, sprint planning meeting, daily Scrum meeting and sprint review meeting.	Product backlog, sprint backlog, sprint, effort estimation, sprint planning meeting, daily Scrum meeting and sprint review meeting.	Product backlog, sprint backlog, sprint, effort estimation, sprint planning meeting, daily Scrum meeting and sprint review meeting.
	XP	Simple design, collective ownership, pair programming, coding standards, TDD, continuous integration and refactoring.	Pair programming, TDD, collective ownership, refactoring, coding standards and simple design.	Simple design, collective ownership, pair programming, coding standards, TDD, continuous integration and refactoring.	Simple design, collective ownership, pair programming, coding standards, TDD, continuous integration and refactoring.	Refactoring, pair programming, collective ownership and continuous integration.	Pair Programming, TDD, onsite customer, coding standards and refactoring.
Team structure		Scrum team (Master, product owner, customer and development team).	Scrum team (Master, product owner, customer and development team).	Scrum team (Master, product owner, customer and development team).	Scrum team (Master, product owner, customer and development team).	Scrum team (Master, product owner, customer and development team).	Scrum team (Master, product owner, customer and development team).

Process	Similar Scrum activities process integrated with the selected XP practices inside the sprint.	Similar Scrum activities process integrated with the selected XP practices inside the sprint.	Similar Scrum activities process integrated with the selected XP practices inside the sprint.	Similar Scrum activities process integrated with the selected XP practices inside the sprint.	Similar Scrum activities process integrated with the selected XP practices inside the sprint.	Similar Scrum activities process integrated with the selected XP practices inside the sprint to fulfill the Lean principles.
Theory	No	method Engineering	No	No	No	No
Principles that are not achieved	P4 (no metaphor), P1,P2, P3, P7, P8, p12 will not be achieved unless the small release is applied.	P1,P2, P3, P7,P8, P12 (no continuous integration and small release) P4 (no metaphor), P1 will not be achieved unless the TDD practice is applied.	P4 (no metaphor), P1, P2, P3, P7, P8, P12 will not be achieved unless the small release is applied.	P4 (no metaphor), P1,P2, P3, P7, P8, P12 will not be achieved unless the small release is applied.	P4 (no metaphor), P1,P2, P3, P7, P8,P9, P12 will not be achieved unless the small release, simple design, coding standards, and TDD are applied.	P4 (no metaphor), P1,P2, P3, P7, P8,P9, P12 will not be achieved unless the small release, simple design, coding standards, and refactoring are applied.

Based on Table 2.10, all of the studies took the Scrum as the baseline and integrate the XP practices into the Scrum sprint. The studies also used all the Scrum practices.

Three studies (Study one, Study three and Study four) added seven XP practices to the Scrum sprint which are simple design, collective ownership, pair programming, coding standards, TDD, continuous integration and refactoring.

Study two added six practices to the Scrum sprint namely the pair programming, TDD, collective ownership, refactoring, coding standards and simple design without applying any of the continuous integration practices.

Study six added five practices to the Scrum sprint namely the pair programming, TDD, on-site customer, coding standards and refactoring. On the other hand, only four of the XP practices were added to the Scrum sprint by Study five. Those were refactoring, pair programming, collective ownership and continuous integration.

As for the theory, only study two used engineering method as a basis theory for conducting the integration.

Based on the results obtained from Table 2.9 and Table 2.10 respectively, it can be concluded that the most common (core) practices used by all of the previous studies are the Scrum iteration review, daily and iteration planning meetings. In addition, these studies also used the pair programming, TDD, refactoring, and coding standards. However, there are other important practices need to be added to the integration between the XP and Scrum as they are very important to fulfill the application of Agile principles as shown in Table 2.10. These XP practices include small release, continuous integration, metaphor, simple design and collective ownership. Small release and continuous integration influence the applications of six

principles (P1, P2, P3, P7, P8, P12) whereas the application of metaphor, simple design and collective ownership practice influences the application of principles (P4), (P9) and (P5) respectively. Consequently, this study will use the two categories of practices. Table 2.11 illustrates the types of practices and the reason for adopting those practices.

Table 2.11

Type of Practices and Reason of Adoption

Practices category	Practices name	Reason of adoption
Common Scrum practices (core)	Iteration planning meeting	Used by all previous studies
	Daily meeting	
	Iteration review meeting	
Common XP practices (core)	Pair programming	Used by all previous studies
	TDD	
	Refactoring	
	Coding standards	
Supported XP practices	Small release	Influence the application of (P1, P2, P3, P7, P8, P12).
	Continuous integration	Influence the application of (P1, P2, P3, P7, P8, P12).
	Metaphor	Influence the application of (P4).
	Simple design	Influence the application of (P9).
	Collective ownership	Influence the application of (P5).

All of the combination methods between the XP and Scrum are still having a lack of quantitative and qualitative measurements to monitor the development process and the product. Furthermore, requirements traceability issue with the XP and Scrum is still need to be solved. Even though the combination methods still using the XP design practice which is very simple, it can still fulfill the Web application complexity. Therefore, the design phase needs to be improved as well. As a result,

four enhancements are needed to overcome the limitations of the XP and Scrum. This will be discussed thoroughly in chapter five.

2.3.2 Web Design Methods

There are many design methods that were proposed for designing Web applications (Wills et al., 2007). These methods are: Hypermedia Design Model (HDM) (Garzotto, Paolini & Schwabe, 1991), Relationship Management Methodology (RMM) (Isakowitz et al., 1995), and Object-Oriented Hypermedia Design Methodology (OOHDM) (Schwabe & Rossi, 1995). The methods were derived from the E-R modeling or Object Modeling Techniques (OMT) or UML extensions.

Figure 2.2 shows the evolution of the Web application design methods.

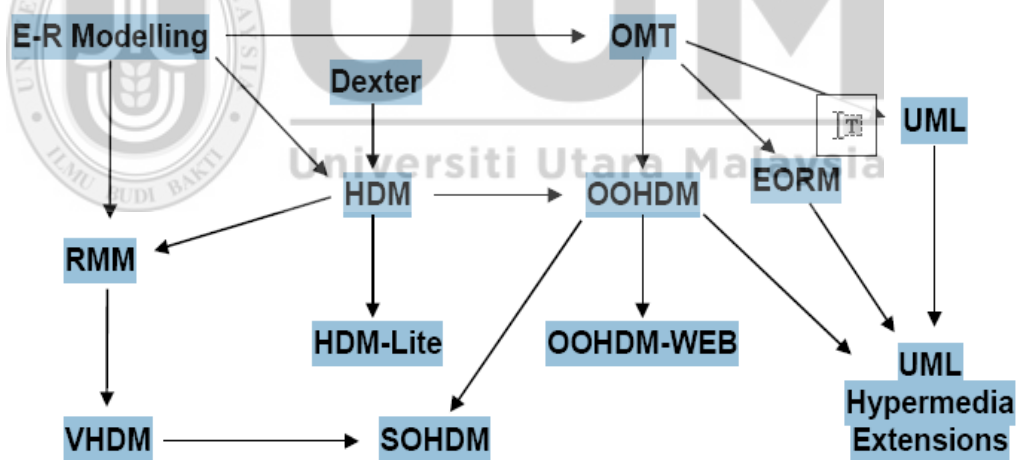


Figure 2.2. The evolution of the Web application methods adopted from Lang (2002)

Based on the evolution in Figure 2.2, the following sections will discuss these methods: HDM, RMM, OOHDM, WSDM, SODHM and WebML.

HDM is the first model that was developed to define the structure of hypermedia application based on the Entity Relationship model by Garzotto, Paolini and Schwabe (1991). These designers stated that the basic features of HDM are the representation of hypermedia application through several or different design primitives. Table 2.12 describes the advantages and disadvantages of this model.

The OOHDM is a model based approach for designing large hypermedia applications; the model was constructed by Schwabe and Rossi (1995). This model consists of four activities, namely, conceptual, navigational, and abstract interface designs as well as implementation. This model is performed with the mixture of incremental, iterative and prototyping based styles. Table 2.12 describes the advantages and disadvantages of this model.

The RMM, created by Isakowitz et al. (1995), consists of seven steps that include E-R, slice, navigation, user interface, and protocol conversion designs as well as run-time behavior, construction and testing. Table 2.12 describes the advantages and disadvantages of this methodology.

The WSDM, proposed by Troyer and Leune (1998), is about “user centered” rather than “data driven”. In a data driven method, the starting point refers to the available data. In the WSDM approach, the starting point is the set of the Web site users. Table 2.12 describes the advantages and disadvantages of this method.

The Scenario-Based Object-Oriented Methodology is an object-oriented methodology for developing hypermedia information systems, it was proposed by Lee et al. (1998). This methodology consists of six phases: domain analysis, object modeling as well as view, navigation, construction and implementation designs. Table 2.12 describes the advantages and disadvantages of this methodology.

The WebML, a modeling language for designing Web sites, was developed by Ceri et al. (2000). This language is considered as annotation for specifying complex Website at the conceptual level. Table 2.12 describes the advantages and disadvantages of this method.

The UWE approach was presented by Koch (2001). It is an object-oriented, iterative and incremental approach based on the Unified Modeling Language. This approach consists of several activities for designing Web applications. The activities are requirements analysis, conceptual, navigation and presentation design supplemented with task and deployment modeling and visualization of Web scenarios. Table 2.12 describes the advantages and disadvantages of this method.

Table 2.12

Advantages and disadvantages of the Web design methods

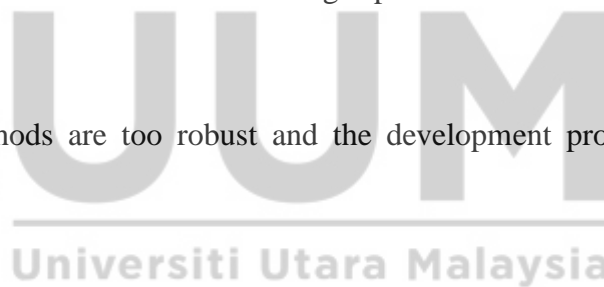
Method	Advantages	Disadvantages
HDM	<ul style="list-style-type: none"> - HDM is a top-down hypertext structured design model for designing hypertext applications (Garzotto, Paolini & Schwabe, 1991). 	<ul style="list-style-type: none"> - Small applications are not covered in the HDM scope (Garzotto, Paolini & Schwabe, 1991). - HDM is a design model rather than a process model (Gaedke & Graf, 2001). - This model is not suitable for developing

		Web applications (Redouane, 2004).
OOHDM	<ul style="list-style-type: none"> - It treats the conceptual, navigational and interface designs as separate activities (Schwabe & Rossi, 1998). - - By using the OOHDM, more modular and reusable designs can be obtained (Schwabe & Rossi, 1998). 	<ul style="list-style-type: none"> - This method is used for large hypermedia designs such as Web sites (Schwabe & Rossi, 1998). - - A very complex and difficult method to understand which requires a lot of training (Lang, 2002; Eldai et al., 2008).
RMM	<ul style="list-style-type: none"> - RMM is applicable for the highly structured and high information volatility such as hypermedia front-ends of databases or legacy applications (Isakowitz et al., 1995). 	<ul style="list-style-type: none"> - This methodology is not suitable for low structure and low volatility applications such as a literary work (Isakowitz et al., 1995). - This methodology is complex and hard to understand (Russo & Graham, 1998; Eldai et al., 2008). - - This methodology needs specialized training (Eldai et al., 2008)
WSDM	<ul style="list-style-type: none"> - This method is user centric, which means that the developer Web site has high usability as it is built by considering users' viewpoints (Troyer & Leune, 1998). 	<ul style="list-style-type: none"> - WSDM is a method for designing a complex website structure such as kiosk Web sites. It is not suitable for Web-based applications (Troyer & Leune, 1998).
SOHDM	<ul style="list-style-type: none"> - Use scenarios to capture the user requirements starting from the earliest opportunity to ensure flexibility and improve the quality of the delivered application (Lee et al., 1998). - The methodology is effective for integrating WWW hypermedia system with enterprise databases (Lee et al., 1998). 	<ul style="list-style-type: none"> - This methodology does not offer any tool in the development process (Koch, 1999). - It does not cover all the development process phases (Koch, 1999).
WebML	<ul style="list-style-type: none"> - It is an annotation for modeling complex Web sites and gives a high-level description for designing the data intensive Website (Ceri et al., 2000). - WebML process primitives are expressive and rich (Distante et al., 2007) 	<ul style="list-style-type: none"> - WebML lacks all of the multimedia, synchronization and interaction aspects (Preciado et al., 2005).
UWE	<ul style="list-style-type: none"> - Modeling elements are fully and widely described in the UML documentation (Koch & Kraus, 2002). 	<ul style="list-style-type: none"> - This approach does not have user modeling and does not support the bottom up design (Montero et al., 2003). - Lack of a complete integration of the different modeling techniques (Koch & Kraus, 2002).

Based on Table 2.12, most of the Web design methods are too complex, which require specialized training. Moreover, most of these methods concentrated on the design part of the Web application development.

Many researchers such as Eldai et al. (2008), Lang (2002), Wills et al. (2007) and Zelenka (2006) pointed out that these methods are not suitable for building Web application in SSF because of the following reasons:

- These methods are too complex and need specialized training
- Quite a few of these methods have been applied outside of academic contexts, or adequately tested in real life situations.
- These methods concentrate on the design part of the Web application development.
- Web design methods are too robust and the development process is time consuming.



2.3.2.1 Web application Common Design Steps

After discussing the advantages and disadvantages of these design methods, it can be concluded that those methods cannot be used as a full methodology for building Web applications. The main reason is that the methods concentrate more on the design phase. Thus, these methods should be taken into account to improve the design phase of the new methodology in this study. The activities of these design methods should be analyzed to come up with common activity that covers the whole design methods. Table 2.13 illustrates each method activity and the common activities for all methods.

Table 2.13

Common Activities of Web Design Methods.

Design method	Activities	Common activities
HDM	<ol style="list-style-type: none"> 1. Entity definition. 2. Object design. 3. Link design. 4. Interface design. 	<ol style="list-style-type: none"> 1. Requirements analysis. 2. Conceptual design (object design). 3. Navigational design. 4. Implementation design (interface). 5. Construction.
OOHDM	<ol style="list-style-type: none"> 1. Conceptual design. 2. Navigational design. 3. Interface design. 4. Implementation. 	
RMM	<ol style="list-style-type: none"> 1. Requirement analysis. 2. Entity and navigational design. 3. Interface design. 4. Construction. 	
WSDM	<ol style="list-style-type: none"> 1. User modeling (requirements). 2. Object design. 3. Navigational design. 4. Implementation look and feel (interface). 5. Actual implementation. 	
SOHDM	<ol style="list-style-type: none"> 1. Domain analysis (requirements). 2. Navigational design. 3. Implementation (interface). 4. Construction. 	
Web ML	<ol style="list-style-type: none"> 1. Requirement collection. 2. Data design. 3. Hypertext in large (object design for the whole system). 4. Hypertext in small (navigational+interface) for each page. 5. Presentation design (implementation until requirement stable). 	
UWE	<ol style="list-style-type: none"> 1. Requirement analysis. 2. Conceptual design. 3. Navigational design. 4. Presentation design (interface). 5. Deployment (construction). 	

Table 2.13 indicates that any prototype design should follow the common activities that have been extracted from the design development methods analysis. These activities are:

- Requirements analysis: collect the whole system requirements directly from the user. These requirements include Web application objectives, targeted audiences, content, style guidelines, and constraint development.
- Conceptual design (object design): determine the objects, classes, subclasses, relationships, attributes and perspectives on the Web application using any object oriented constructs (classes, relationships or use cases).
- Navigational design: this phase describes how users can navigate through a Web application as well as specify the link of pages and content units to the whole application. This will be done by determining the nodes, links, as well as the access and navigational structures.
- Implementation design (interface): the aim of this phase is to design the look and feel of the Web application by generating the required page structure, page flow, user interface and logical database schema.
- Construction: developers run the Web application output in the target or real environment.

This study uses these activities as a baseline to build a simple design method to improve the design of the Web application in SSF.

2.4 Software Measurements

According to Kettelerij (2006), software measurement is defined as “an effective means to understand, control, predict and improve software development projects”. Measurement of both the product and development processes has been considered a long time ago as an important activity for successful software development. The

analysis of the appropriate measures of software artifacts such as requirements, designs, and source code, problems can be recognized and solutions can be determined during the project execution. This may reduce defects, rework (effort, resources, etc.), and cycle time (Graf, 2005; McCurley et al., 2008).

Kettelerij (2006) Morasca (1999), Solingen and Berghout (2001) and Wangenheim et al. (2003) pointed out that the implementation of software measurement during the development process provides many benefits such as:

- Increase understanding and controlling of software development processes;
- Increase capacity to improve the software development process;
- More accurate estimates of software project costs and schedule;
- More objective evaluations of changes in technique, tool, or methods;
- More accurate estimates of the changes effects on project cost and schedule;
- Decreased development project cycle time and costs due to increased productivity and efficiency;
- Improve customer satisfaction and confidence due to higher product quality.

2.4.1 Measurement Methods

Measurement method is “a systematic way or procedure to implement a software measurement mechanism in development organization that can give a general guidance about measuring, analyzing, and recording information that can be used for monitoring performance of the process” (Kettelerij, 2006). A set of measurement methods is discussed in this section, which include Practical Software and Systems

Measurement (PSM), Quality Function Deployment (QFD) and Goal Question Method (GQM).

The PSM is “information driven” measurement mechanism that includes select, collect, define, analyze and report specific software issues. These issues are risks, problems, progress, cost, product size and stability, product quality, process performance, technology effectiveness and customer satisfaction (Jones, 2003).

The QFD is “a top down customer oriented approach to product innovation that guides the product managers and design teams through conceptualization, creation and realization process of new products” (Govers, 1996). The QFP is also considered as a product quality measurement method that consists of four phases: product concept, product design, process design and manufacturing operations (Govers, 1996).

The GQM is “a method to collect software engineering data, whereby measurement goals are established, questions are linked to the goals and metrics are derived to satisfy the questions” (Kettelerij, 2006; Morasca, 1999; Solingen, 1999; Solingen, 2002). This method requires organizations to define their measurement mechanism based on specific goals. As shown in Figure 2.3, goals are transformed into questions that consecutively converted to metrics.

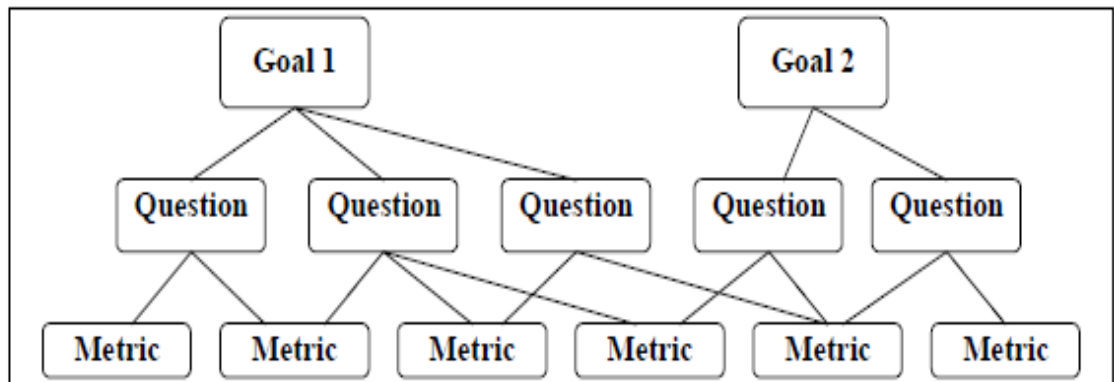


Figure 2.3. GQM Method.

The result of applying the GQM approach is the specification of a measurement mechanism or system which targeting a set of issues and rules. The resulting measurement mechanism using the GQM methods has three levels to be performed:

- 1- Conceptual level (GOAL): The goal specifies the purpose of measurement, object to be measured, issue to be measured, and the viewpoints from which the measure is taken. Goals identification format in Table 2.14 was extracted from Basili et al. (1994), who mentioned that the measurement goal should be built in an understandable and a clear structure. This structure should clearly define the purpose (what object and why), perspective (what aspect and who) and context characteristics.

Table 2.14

Goal Format

Analyze	The object to be measured
For the purpose of	Understanding, monitoring or improving.
With respect to	Quality focus of the object that the measurement focuses on.
From the viewpoint of	People that measure the object.

- 2- Operational level (QUESTION): A set of questions is used to describe the way to achieve a specific goal. Questions try to demonstrate the object of measurement (product, process, resource) with respect to a selected quality issue and to determine its quality from a specified viewpoint.
- 3- Quantitative level (METRIC): A set of data collected using several quantitative and qualitative metrics in order to answer each question.

The GQM is considered as the most popular measurement method that represents top down goal oriented method (Ardimento et al., 2004; Caldiera & Rombach, 1996; Kettelerij, 2006; Solingen, 2002; Weiss, 1994). However, the GQM cannot deal with SSF because it performs the measurement mechanism by separating the GQM team. This separated team cannot be established due to the small number of the SSF and organization structure. The light weight GQM approach proposed by Wangenheim et al., (2003) uses the same phases of the GQM. However, the measurement mechanism is performed by one member and some activities of the GQM are excluded to fit the small software firm's employee size and minimize efforts by reducing measurement activities. Table 2.15 describes the activities of the light weight GQM compared to the original GQM.

Table 2.15

Light weight GQM against original GQM adopted from Wangenheim et al. (2003).

Phases	GQM method	Light weight GQM
Planning	Establish team, project plan, and training.	Introduce measurement mechanism.
	Select improvement area and application project.	
Definition	Define a measurement goal, conduct interview, and review	Define measurement goals and format the goals.

	software process models.	
	Define question, hypothesis and review.	Define questions.
	Produce an analysis plan.	
	Define metrics and review	Define metrics
	Produce the GQM and measurement plans.	Produce a measurement plan, data collection procedures, and data collection instruments.
	Trial period, hold a kick off the session.	Produce a data collection plan and create a metric base.
Data collection	Create a metric base.	Collect and validate data.
	Collect and check data collection form, store measurement data in metric base.	Store the collected data.
Interpretation	Define analysis sheet and presentation slide Prepare, organize and hold feedback session, report metrics result.	Data analysis. Feedback session.
Packaging		Packaging the results.

As on the above Table, it's clearly shown that lightweight GQM required minimum number of team members for conducting the measurement mechanism, less process steps and fewer efforts.

2.4.1.1 Measurement Methods Evaluation

To perform a successful measurement mechanism in SSF, many aspects should be considered. For example, Ardimento et al. (2004), Caldiera & Rombach (1996), Kettelerij (2006), Rombach and Basil (1991), Solingen (2002) and Weiss (1994) insisted that the measurement applied should be top down goal oriented methods. In addition, Scholtz and Steves (2004) stated that the measurement methods should be used for process and product quality. As for Wangenheim et al. (2003), the

measurement mechanism should take into account the limitations of SSF such as the experience and lack of staff.

Based on the above discussion, it can be concluded that many criteria were used to evaluate the measurement methods; PSM, QFD, GQM and light weight GQM. These criteria are top down, goal oriented, process and product oriented, simplicity and small team size. The results of the measurement methods, evaluation were extracted from Kettelerij (2006), Rombach and Basili (1991), Scholtz and Steves (2004), Solingen (2002) and Wangenheim et al. (2003). Table 2.16 shows the results.

Table 2.16
Measurement Methods Evaluation

Criteria Method	Top down	Goal Oriented	Process and Product Oriented	Simplicity	Fit to 10- 50 size
PSM	×	×	×	×	×
QFD	√	×	×	×	×
GQM	√	√	√	×	×
Light weight GQM	√	√	√	√	√

(√) means satisfy the criterion, (×) means not satisfied the criterion

As conclude from Table 2.16, the most suitable measurement method to be used by the SSF is the light weight GQM because it satisfies all the evaluation criteria compared to the other methods. Therefore, this method is used as the baseline to create a measurement mechanism for developing Web application in SSF.

2.4.2 Measurement Mechanism Purposes

Measurement mechanism is the process of achieving the measurable goals by clearly defining the questions, measures (metrics), stakeholders and the information required by the stakeholders (Kettelerij, 2006).

The application of measurement mechanism during the development process aims to meet the following purposes: planning, coordinating, monitoring, controlling, and reporting to ensure that the development and maintenance of the software are systematic, disciplined and quantified (Abran et al., 2004; Braind et al., 2002; Solingen & Berghout, 1999). This study focuses on the monitoring purpose.

2.4.2.1 Benefits of Using Monitoring

Monitoring refers to the review of the development process in order to follow the activities and the product performance evolution starting from the early-stage of the project. The aim is to find the latent project risks and other related problems (Briand et al., 1996; Esaki et al., 2012). Monitoring provides many benefits such as the ability of tracking the progress of the development process, giving feedbacks to the team members for improvement, continuously fulfill the quality goals, and accelerating the development process of finding and correcting errors. In addition, performing monitoring helps to reduce risks, ease maintenance, and pursue the project management efficiently, which will then lead a project to success (Ardimento et al., 2004; Briand et al., 1996; Esaki et al., 2012; Morasca, 1999; Solingen & Berghout, 2001; Tsai & Cheung, 1999; Tu et al., 2009).

2.4.2.2 Measurement mechanism critical success factors

It is evidenced in many studies that measurement mechanism has its critical success factors. Therefore, performing a measurement mechanism needs to consider various related factors. Among the critical success factors as compiled by Kettelerij (2006) are to begin small with goals and extend the mechanism as you go, provide training to people affected by the mechanism, involve developers, test and manage the mechanism implementation, provide regular feedback to those involved in using the mechanism, and automate the measurement if possible. Murphy and Cormican (2012) categorized the best practices of the measurement mechanism into five categories: organization, management practices, people, information communication and technology. Unfortunately, the technology had the lowest scoring category in this study. Other issues that should be taken into consideration are SSF limitations and Web application characteristics. Wangenheim et al. (2003) pointed out that the measurement mechanism used by the SSF should be simple to match the small size of the firms and less effort to deal with less experience team. On the other hand, Calero et al., (2005) stated that the Web application measurement metrics mechanism should concentrate on the quality characteristics and life cycle process. Based on the discussion, this study focuses on the first four categories to form as the baseline in determining the success factors of performing a measurement mechanism and to harmonize them in the Web application development in SSF. Table 2.17 shows the critical success factors of performing measurement mechanism in SSF. This table was extracted from Calero et al. (2005), Wangenheim et al. (2003), Murphy and Cormican (2012) and Kettelerij (2006).

Table 2.17

Measurement Mechanism Success Factors

Category	Critical factor	Specification
Organization	Simple goal oriented approach.	The measurement mechanism takes the improvement goals and converts them into metrics using the light weight GQM method.
	A measurement mechanism should be qualitative and quantitative.	The mechanism should use direct (quantitative) metrics such as the LOC and indirect (qualitative) metrics such as the quality characteristics.
	The measurement mechanism should concentrate on the Web application and process quality.	Web application quality measured by the Web application characteristics (functionality, usability... etc.) and quality of the process measured by development activities, maintenance, effort, and reuse.
Management	Incremental approach.	The measurement mechanism is tied to the development process which used the combined XP and Scrum so that the collection of the data will be incrementally over time.
	Use standard method.	Using a well-known method to perform the measurement mechanism will reduce the effort and ensure clarity.
People	Stakeholder participation.	Developer, tester and manger will be the data owner for the measurement mechanism.
	Measurement member training.	One developer will act as a measurement member to collect data and the other for data analysis. The monitoring team should attend training session to know how to perform the measurement activities.
	Monitoring stakeholders.	The people were monitored during the development process by monitoring their development practices and measuring their productivity.
Information communication	Transparency.	The nature of data collection and data collection purpose should be clear in the planning phase of the development.
	Usefulness.	The stakeholder (data owner) should understand the reason of collecting data.
	Feedback.	Feedback assured that the data being analyzed, processed and put to use.

The measurement mechanism should be done quantitatively and qualitatively because it involves measuring the product and process quality. The product should be measured by several metrics in terms of time, cost and other related features (Basili, 1992; Daskalantonakis, 1992; Dumke et al., 1998; Kettelerij, 2006). On the other hand, the process can be measured using a set of metrics related to the process activities, practices, productivity, process quality characteristics (factors) (Basili, 1992; Dumke et al., 1998; Kroeger et al., 2014). In this study, the monitoring mechanism will be involved with measuring the quality of the product and the process. The product quality will be measured quantitatively using time, cost and Web application quality attributes. The process quality will be measured using the quantitative and qualitative metrics. The quantitative metrics involved with process activities, development and management practices and process productivity as discussed in section 2.4.2.3. Whereas, the process can be measured qualitatively by monitoring the process quality factors that are discussed section 2.4.2.4.

2.4.2.3 Development process quality factors

There are many quality factors were proposed for measuring the quality of a product. However, there is no much study of the measurement of the quality of a development process (Kroeger et al., 2014). Only a few studies mentioned or defined some of the factors that should be considered in measuring the development process quality. For instance, Sørungård and Sindre (1995) proposed an approach containing product quality factors that can be applied to measure the development process. These factors include correctness, efficiency, expandability, flexibility, integrity, interoperability,

maintainability, manageability, portability, reliability, reusability safety, survivability, verifiability and usability.

Feiler and Humphrey (1993) divided the process quality factors into two categories; static and dynamic. The static factors are accuracy, fidelity, fitness, precision, redundancy, scalability and maintainability, whilst the dynamic factors include lifeness, robustness, fault tolerance, autonomy and responsiveness.

Guceglioglu and Demirors (2011) created a measurement model for software process improvement that consists of various quality factors such as suitability, IT-based functionality, accuracy, interoperability, security, maturity, recoverability, understandability, operability, attractiveness and analyzability.

In their study, Kroeger et al. (2014) and Kroeger (2011) were able to identify four most important process quality factors based on the interview conducted with 17 software developers. The factors are effectiveness adaptability, compatibility and applicability. When the same model was applied for the Agile environment (Scrum), the following five factors were determined; effectiveness, accessibility, adaptability, changeability and supportability. Therefore, in this study, all the seven factors were considered. This is depicted in Table 2.18.

Table 2.18

Process Quality Factors

Factor	Definition	Measured by
Effectiveness	An effective process must help us produce the right product. This shows the capability of a software engineering process to transform a set of inputs into a desired set of out-puts (Kroeger et al., 2014).	<p>Consistency: the use of procedure and standard.</p> <p>Accuracy: the use of tools, methods and procedure.</p> <p>Completeness: the correctness in performing process and the production of appropriate outcome (Baharom et al., 2011).</p>
Adaptability	The ability of process users to adapt to a software engineering process applied in different situations (Kroeger et al., 2014; Sorumgard and Sindre, 1995).	<p>Tailorability: The ability of a standard process to be adapted to form a more specific process (Kroeger et al., 2014).</p> <p>Flexibility of a process refers to the ability of a practitioner to adapt to the performance of process activities to meet a specific need, without requiring a change to the process itself (Kroeger et al., 2014).</p>
Compatibility	The capability of a software engineering process to interact with one or more specified process (Kroeger et al., 2014; Guceglioglu & Demirors, 2005),	This factor is required, especially when the organization used multiple processes. Therefore, it is important that the interfaces between these processes are considered (Kroeger et al., 2014).
Accessibility	The ability of a process user to find information about a software engineering process (Kroeger et al., 2014).	The medium of a process is widely considered by practitioners to have a significant influence on the perceived accessibility of the process. The electronic process descriptions are highly favored compared to the hard-copy documentation. The extent to which the process is described using graphical, rather than textual, notations were found to positively influence stakeholders' perceptions of process accessibility (Kroeger et al., 2014). (Organization training in CMMI)
Applicability	Applicability is defined as the extent to which a software engineering process describes activities that are required to be performed to complete a piece of work in a specified context (Guceglioglu & Demirors, 2005; Kroeger et al., 2014).	Process applicability is often an issue where highly standardized processes are used across a wide range of problem situations. If such processes are not tailored correctly to the specific context, then practitioners may be required to perform activities that do not directly relate to the task at hand and as a result the effort may be wasted (Integrated project management practices CMMI).

Changeability	The ability of a process to meet requirement changes (Kroeger, 2011).	<ul style="list-style-type: none"> • Is there a way to determine risk sources and categories? • Is there a strategy established for risk management? • Is there a way to evaluate, categorize, and prioritize risks?
Supportability	This is defined as the ability of a software engineering process to be supported within a specified context. It is important that the necessary resources, expertise and technology for performing a successful process are available prior to that process being deployed (Kroeger et al., 2014).	High-quality project management methodology that has a strong focus on the metrics collection and analysis may be introduced to a project. However, if the project team does not have the necessary data analysis skills or if the data takes a significant amount of effort to collect due to a lack of supporting technology, then the process is unlikely to achieve the desired outcomes. (Supplier agreement management practices in CMMI).

Effectiveness: Based on the definition and what mentioned by Baharom et al. (2011) in terms of effectiveness, effective process is represented by a consistent, accurate and complete process. Therefore, a set of practices related to these three sub factors should be performed during the development process activities. These practices are described in Chapter Five.

Adaptability: in order to ensure the adaptability process, two sub-factors, tailorability and flexibility were identified by Kroeger et al. (2014). Tailorability is related to the type of integrated process performed by an organization, the theory used for integration, the process performance and the ease to use. Flexibility is related to the ability of the team members to adapt to the performance of the process without affecting the process itself. Therefore, there is a need to have a set of practices

during the development process to ensure the tailorability and flexibility of the process. These practices are shown in Chapter Five.

Compatibility: the compatibility process can be ensured by determining whether the interaction between more than one process is easy and clear during the development of the product (Kroeger et al., 2014). As a result, in this study, two practices were identified to ensure the compatibility of the process. These practices are shown in Chapter Five.

Accessibility: based on the definition, it is clear that accessibility relates to the training practices as introduced in the CMMI. These training practices are important to help any team member to access any process activity easily. In addition, electronic access and graphical process representation support the accessibility factor (Kroeger et al., 2014).

Applicability: a process is applicable if it is tailored correctly to specific context. This means that for each piece of work there is a clear activity to be performed and applied throughout the whole project. In other words, the process used should have defined activities from the beginning to the end, should be measured by measurement mechanism, should be managed by specific plan and contribute product measures and experience to the future product. Therefore, a set of practices introduced by the CMMI is used to cover the process applicability. This is shown in Chapter Five.

Changeability: is the ability of a process to meet requirement change. This is important because the requirement change is one of the risks that any organization may encounter. As a result, the CMMI risk management practices should be performed to manage the potential risk including changing requirement. These practices are shown in Chapter Five.

Supportability: is defined as the extent that process has been supported from resources, expertise and technology. Therefore a set of practice introduced from supplier agreement management practices in CMMI proposed to ensure the process supportability. These practices are shown in Chapter Five.

2.5 Criteria of a good methodology for Web applications in SSF

Costagliola et al. (2002) defined methodology as “a comprehensive, multiple-step approach to system development that guides the development process and influences the quality of the final product. It describes both the activities to be carried out and the deliverables that should be produced at the end of each activity. Furthermore, it gives a full set of concepts and models which are internally self-consistent and a collection of rules and guidelines”. Table 2.19 describes the required features that must be taken into account when proposing a new Web application development methodology for SSF.

Table 2.19

Criteria of Good Development Methodology

Feature	Resource
Iterative process, deal with changing requirements and small teams.	Costagliola et al. (2002), Eldai et al. (2008), Fayad et al. (2000) and Rumbaugh (1995), Henderson-Sellers (1995).
Full life cycle (model, process, rules, guidelines, practices and activities).	Costagliola et al. (2002), Henderson-Sellers (1995) and Rumbaugh (1995),
Must be incremental, flexible and generic enough to meet the uniqueness and individuality that are specific to Web applications. Therefore, several methodologies may need to be combined and merged to cover and cope with the above features.	Costagliola et al. (2002) and Howcroft & Carroll (2000).
Quality attributes and assurance for the Web applications.	Fritzsche & Keil (2007), Nawaz & Malik (2008) and Wu & Offutt (2002).
Should have a suitable measurement mechanism for monitoring the quality of the development and final process.	Kettelerij (2006), Solingen and Berghout (2001), Wangenheim et al. (2003).
Should be built based on a specific theory.	Fitzgerald et al. (2006), Ralyte et al. (2003) and Brinkkemper (1996).

Table 2.19 indicates that the new methodology should be iterative and flexible to meet the unique characteristics that are specific to Web applications. In addition, these features can also deal with the limited number of staff in the SSF. However, the new methodology must also include a full set of activities, models, rules, practices and guidelines that describe the whole development process. Therefore, several methodologies may need to be combined and merged to cover and cope with the above features.

The quality attributes of the Web application product are another important aspect that need to be considered while constructing or proposing the new methodology. Besides these features and attributes, the measurement mechanism should also be integrated into the new development methodology. The function of the measurement mechanism is to analyze the collected data from a specific metrics for monitoring the

quality of the development process, the final product and also for reducing the defect and accelerate the development cycle.

The aim of this study is to construct a new methodology for Web application development and measurement. This methodology combined the XP and Scrum. The reasons for performing this combination are (i) to overcome the XP and Scrum limitations, (ii) to build one specific development method that suits all projects requirements circumstances, and (iii) to increase development method efficiency and applicability (de Cesare et al., 2004). Based on the literature, there are several theories that can be used to perform the combining of the two development methods, which include the contingency-based selection, engineering and tailoring methods.

The contingency-based selection method (Iivari, 1989) is based on the principle that, rather than using a specific method for being commonly applied, the team should choose a method from a broad portfolio of development methods to suit each different project context. One of the fundamental problems of using the contingency-based selection method is that the developers should be familiar with many methods so that they can switch to other methods if a problem occurs while using the current (Fitzgerald et al., 2006).

The engineering method is a meta-method process, whereby a new method is constructed or “engineered” from the ground up using the existing “method fragments” instead of selecting a method from any available method base (Brinkkemper, 1996). In addition, the new method should be constructed from the existing methods (Fitzgerald et al., 2006). The engineering method theory has three

types of strategies, which include assembly-based, extension-based and paradigm-based strategies (Ralyte et al., 2003). The assembly-based strategy is used to construct a new method by assembling many methods. The extension-based strategy is to extend an existing method, while the paradigm-based is to construct a new method from scratch. The theory that will be adapted in this study is the extension-based strategy which comprises of the two stages:

- Specify and analyzes the baseline method, by determining the limitations and strengths of the method.
- Determine the parts that should be extended to the baseline method. These parts are included from other methods based on the limitations of the baseline method,

2.6 Validation methods

Empirical methods are commonly used for validation in the software engineering field; examples of the empirical methods are experimentation, surveys, action research and case studies (Sjoberg et al., 2007; Tofan et al., 2011). An experiment is “an empirical inquiry that investigates causal relations and processes. The identification of causal relations provides an explanation of *why* a phenomenon occurred, while the identification of causal processes yields an account of *how* a phenomenon occurred” (Sjoberg et al., 2007). Experiments are used when the researcher controls the situation with immediate, exact, and efficient control of the behavior of the phenomenon to be examined (Yin, 2003).

Survey is “a retrospective study of a situation that investigates relationships and outcomes” (Sjoberg et al., 2007). It is useful for studying a large number of variables using a large sample size and accurate statistical analysis. Surveys, particularly well-suits studies that conducted in order to answer what, how much, and how many questions (Pinsonneault & Kraemer, 1993).

Action research is an “an iterative process involving researchers and practitioners acting together on a particular cycle of activities, including problem diagnosis, action intervention, and reflective learning” (Avison et al., 1999).

Case study is “an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident” (Yin, 2003).

The four methods were compared based on specific criteria. These criteria are:

Researchers control, Cost, Focus and Sample size.

Researchers control means that the researcher control over the situation, with direct, precise, and systematic manipulation of the behavior of the phenomenon to be studied.

Cost: the cost of performing the method.

Focus: focus of the investigation in terms of how the researcher will investigate the phenomenon.

Sample size: the number of targeted respondents that identified to perform the method.

Table 2.20 shows the comparison between validation methods based on the previous criteria. The criteria used and the information of Table 2.20 was extracted from Easterbrook (2008), Sjoberg et al. (2007), Wohlin et al. (2006), Tofan et al. (2011) and Yin (2003).

Table 2.20

Validation method comparison

Method	Experiment	Survey	Action method	Case study
Criteria				
Researcher control	High	Low	High	Low
Cost	High	Low	High	Medium
Focus	Why and How	How many and how much	How	How and why
Sample size	Small	Large	Small	Small

Based on the comparison, it's clearly shown that experiment supports the researcher control, consume more budget, concentrates on the how and why and suitable for small sized sample. Survey method does not support the researcher control, has lower cost, concentrates on how many and how much questions, adequate to the large size sample. In addition, action method supports researcher control, consumes more budget, concentrate on *how* question, suitable for small sized sample. Lastly, case study provides less control of the researcher, consumes fair cost, focuses on how and why questions, suitable for small sized sample.

Therefore, this study will use the case study method to validate the proposed methodology as it has a distinct advantage to be used in the study that considered

the researchers an observer with little or no control on the process. In addition, using case study not consume much budget like experiment and action method. Furthermore, It is useful to use case studies to answer a ‘how’ or ‘why’ questions. Moreover, the case study is often used as a plain working example of a newly proposed method that applied to a limited number of respondents.

2.6.1 Validation factors

Several studies discuss the factors that are needed to evaluate the effectiveness of implementing software methods, models, and frameworks, such as Kitchenham and Pickard (1998), and Kunda (2001). Kitchenham and Pickard (1998) used three major factors in evaluating his method of success: basic, use and gain evaluation. The basic evaluation is concerned with the quality of the component documentation, for example, completeness, readability and understandability of the component description. Use validation is concerned with the quality of the component, for example, whether the component is easy to implement and “helpful”. Gain validation is concerned with the benefits delivered by the component, for example, whether the component is cost-effective and supports decision making. These factors were also adapted by Kunda (2001) to validate his framework. The factors that were used by Kunda (2001) are shown in Table 2.21

Table 2.21

Kunda's validation factors adopted from Kunda (2001)

Validation Factors	Variables
Gain satisfaction	<ul style="list-style-type: none"> - Perceived usefulness. - Decision support satisfaction. - Comparing with current method. - Clarity. - Appropriateness for task.
Interface satisfaction	<ul style="list-style-type: none"> - Perceived ease of use. - Internally consistent. - Organization (Well organized). - Appropriate for audience. - Presentation (readable and useful format).
Task support satisfaction	<ul style="list-style-type: none"> - Ability to produce expected results. - Completeness. - Ease of implementation. - Understandability (easy to understand).

Recently, Al-Tarawneh (2013) adopted the same factor in validating his framework. Referring to Table 2.21, it's shown that perceived usefulness and perceived ease of use were used as variables to measure the gain satisfaction and interface satisfaction respectively. However, the perceived usefulness and perceived ease of use were used in the technology acceptance model (TAM) that developed by Davis, (1989) as a certain factors. TAM is recognized as the theory that helps users how to accept and use a new technology.

Perceived usefulness is "the degree to which a person believes that using a particular system or method would enhance his or her job performance". Whereas, the ease of use defined as "the degree to which a person believes that using a particular system or would reduce from his effort". Therefore, this study will merge the three factors that were identified by Kunda (2001) and two factors that were identified by Davis,

(1989). The factors are: gain satisfaction, interface satisfaction, task support satisfaction, perceived usefulness and ease of use.

2.7 Summary

This chapter clarifies the need of developing a new Web application development methodology for SSF. The following are several areas that have been discussed while reviewing the related resources:

- The current methods used for developing Web applications.
- Web design methods.
- Measurement methods.
- SSF problems.

The important findings in Chapter Two are used as inputs for the next chapter. These outputs are:

- The most suitable development method to be used for SSF.
- The core and supported Agile practices that should be integrated to the new methodology.
- The best measurement method to be used for SSF.
- The common steps for Web application design.
- Best Web application development and measurement practices for SSF.
- Criteria of a good Web application development methodology.
- Questionnaire and pilot study

The most popular Agile development methods that are recommended to be used in developing Web application in SSF, are XP and Scrum. XP concentrates on the development part and Scrum focuses on the management. Therefore, the combination of both methods will definitely cover the development and management issues. Nevertheless, both methods do have various limitations as described in Table 2.22.

Table 2.22

Literature Review Analysis

Issues and problems of Web applications in SSF	XP and Scrum
Staffing problems. (Small teams).	Covered
Project management problems.	Covered
High changing requirements	Covered
Lower communication.	Covered
Risk management.	Covered
Shorter time to market and product life cycles.	Covered
QA aspects.	Not Covered
Measurement mechanism.	Not Covered
Requirement traceability and reuse.	Not Covered
Simple Design method.	Not exist
Required best development and measurement practices	Not fully covered

Table 2.22 shows that most of the problems faced by the Web application developers in the SSF are fully addressed by both XP and Scrum methods. However, there are problems regarding the integration between the XP practices and the Scrum development method. These problems are related to the quality assurance, measurement mechanism, requirement traceability and prototype design. As a result, there is a need of developing a new methodology based on XP and Scrum.

In this chapter, the core Scrum practices of the management, core and supported XP practices have been identified.

In order to monitor the process and product quality, a measurement mechanism should be integrated during the development process. The best way to perform this mechanism is using a specific measurement method. The most recommended measurement method for SSF is light weight GQM.

Web design methods were compared based on the process activities of each method. This helps to generate a common guideline by extracting the steps from all methods in order to build a Web design method.

Small software firm's developers should follow several practices during the development process of Web applications. The application of these practices helps to get high quality product. These practices are: short development life-cycle times, delivery of bespoke solutions, multidisciplinary development teams, analysis and evaluation, requirements management, testing, maintenance, project management and quality management.

There are many criteria or features for any successful Web application developing methodology in SSF. These criteria are concerned with the development process type, components of the methodology and the monitoring of the product quality. Lastly, the methodology construction should follow a specific theory.

The last output of this chapter is the questionnaire which was designed and formulated to be used as a data collection instrument in the survey. The questionnaire was designed based on various previous studies in the related field.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

This chapter outlines the research methodology of the study. As mentioned in chapter one, the main objective of the study is to construct an Agile Web application development methodology for small software firm (SSF) that emphasized on monitoring the process and product quality during the development. The research was conducted in four main phases and followed a deductive approach. The methodology used in the study will thus be discussed under the four phases. Phase one is mainly focused on conducting the theoretical study and defining the research problem. In Phase two, a quantitative approach was followed to investigate the current practices in Web application development and measurements at SSF. In Phase three, the Plan-Do-Check-Act model was adapted to construct the proposed methodology. Finally, in Phase four, expert review and case study methods were used to evaluate the proposed methodology.

3.2 Research Design Approach

The research design of this study used a deductive approach (Trochim, 2006). This approach begins with general idea (such as theory, principles, and concepts) and ends with specific conclusions. It is appropriate to be used for constructing a model based on theories and concepts that are derived from the literature and empirical

findings. Then the proposed model will be applied and evaluated in real environment (Bryman & Bell, 2007).

Four phases are used to develop a new methodology as shown in Figure 3.1. Each phase consists of goal achievement, set of inputs, activities, and outputs. The following sections explain in detail these four phases.

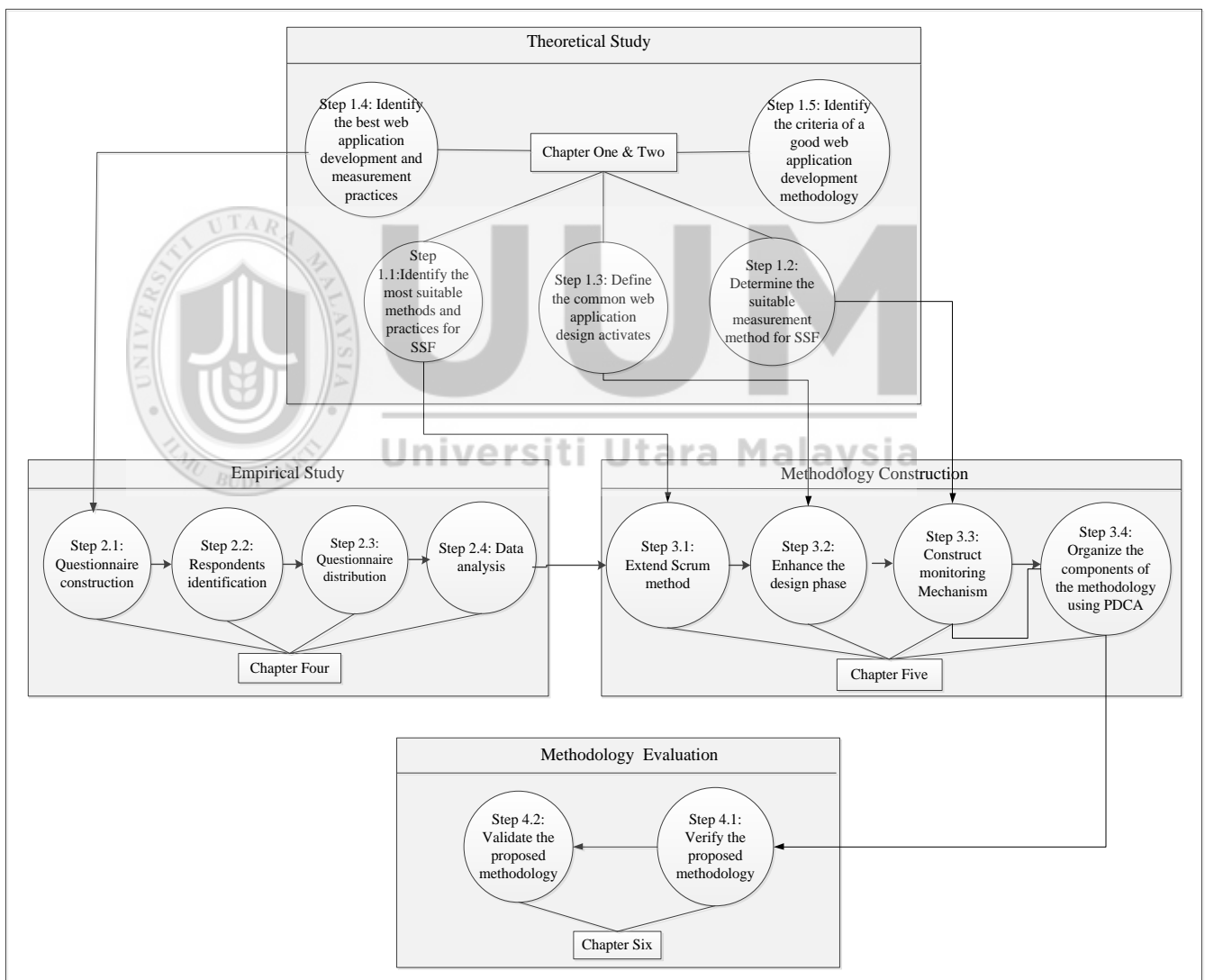


Figure 3.1. Research Methodology Phases.

3.3 Research Methodology

The research methodology is divided into four phases and each phase consists of several steps. The following subsections discuss in detail about the phases.

3.3.1 Theoretical study

The theoretical study focuses on exploring the research directions related to Agile software development practices and methods, particularly those implemented by SSF, Web application development and software measurement and metric for assuring process and product quality. This phase explored the software development methods and identified the best practices that need to be implemented by SSF. In addition, this phase also investigated the suitable measurement method, and finally identified the common activities for Web application design. The information of the related theories was obtained from journals, papers, books, documents and proceedings.

The problems currently faced by the SSF in developing Web application were highlighted. The findings of this phase were used to formalize the research problem and research objectives as well as gain knowledge on the state of art related to the Web application development in the SSF. The key findings of this phase were presented in Chapter Two. Consequently, this phase involved five steps.

Step 1.1: Identify the most suitable methods and practices for the SSF.

The process of this step involved with the comparison of the conventional and Agile development methods based on specific criteria related to Web application development in the SSF. These criteria are fit to the size of 10-50, complexity, flexibility for requirement changing, customer collaboration and the use of the quality assurance measurement mechanism (QAMM). These criteria were extracted from several studies such as those of Haung et al. (2008), Tarawneh and Allahawiah (2009), Pusatli and Misra (2011) and Rodriguez et al. (2002).

Based on the comparison results, Scrum and XP were identified as the most suitable Agile software development methods for SSF. However, the Scrum concentrates on the management part, whilst XP concentrates on the development part. There are a number of studies that have attempted to combine XP and Scrum to fulfill the development and management sides. Among these are Mar and Schwaber (2002), Fitzgerald et al. (2006), Clutterbuck et al. (2009), Qureshi (2011), Jyothi and Rao (2011).

A comparison between these studies was performed to understand how these methods were combined, the practices that have been used in the combinations, and what are the Agile principles that have been achieved from each study. The result obtained from this comparison is used to identify the core and supported practices that need to be considered in this study. The core practices were selected based on

the practices recommended by all the studies. The supported practices are the important practices to fulfill the Agile principles.

Step 1.2: Determine a suitable measurement method for SSF.

This step was carried out by comparing the existing measurement methods using comparison criteria. These criteria are top down, goal oriented, product and process oriented, simplicity and fit to the SSF size. The criteria were extracted from Ardimento et al. (2004), Kettelerij (2006), Scholtz and Steves (2004), Solingen (2002) and Wangenheim et al. (2003). The methods used for the comparison in this step were the PSM, QFD, GQM and light weight GQM. The results of the comparison show that the only method that satisfies all the criteria is the lightweight method. Therefore, the lightweight GQM is used to perform the monitoring process in SSF.

Step 1.3: Define the common activities for designing Web applications

The standard Web design activities consist of requirements analysis, conceptual design (object design), navigational design, presentation design and adaptation design (Barna et al., 2003). This aim of this step is to identify which of the existing method that fully satisfied the Web design standard activities. The comparison of the Web design standard activities and existing methods was done as shown in Table 3.1.

Table 3.1

The Activities of the Web Design Methods

Standard design activities	Web Design Methods						
	HDM	OOHDM	RMM	WSDM	SOHDM	Web ML	UWE
Requirements analysis	√	×	√	√	√	√	√
Conceptual design (object design).	√	√	√	√	×	√	√
Navigational design.	√	√	√	√	√	√	√
Adaptation Design	√	√	√	√	√	√	√
Presentation Design	×	√	√	√	√	×	√

(√) means covered and (×) means not covered

Table 3.1 shows that only three of the Web design methods fully satisfied all the standard design activities. These methods are the RMM, WSDM and UWE. Therefore, the common activities in this study were defined according to these methods.

Step 1.4: Identify the best Web application development and measurement practices for SSF

This step aims to determine the best development and measurement practices for developing Web application in SSF. For the development practices, various empirical studies related to the field of software and Web application development in SSF conducted previously were reviewed. Based on this review, a list of development practices was presented. These practices related to the development life-cycle time, development teams analysis and evaluation, requirements management, testing and maintenance.

For the measurement practices, several studies related to software measurement, software quality and software management were reviewed. Two categories of practices related to measurement were identified, which are quality management and project management.

A list of the best development and measurement practices is shown in Chapter Two based on the results of this step.

Step 1.5: Identify the criteria of good methodology.

In this step, the criteria were identified by studying several previous studies, including those conducted by Costagliola et al. (2002), Fitzgerald et al. (2006), Fritzsche and Keil (2007), Eldai et al. (2008) Ralyte et al. (2003) and Wangenheim et al. (2003). The recommendations of these studies were taken as criteria on how a good methodology looks like. The criteria are the process type development, methodology components, quality and progress monitoring and constructing a new methodology based on specific theory.

3.3.2 Survey

This phase aims to determine the real characteristics of the SSF in Jordan, examine the need of new methodology for developing Web applications in SSF, investigate the current development and measurement practices of Web application development in SSF and classify the development and measurement practices into specific groups using variable clustering. The survey was conducted based specifically on the following steps:

Step 2.1 Questionnaire construction

In this study, a questionnaire was used for data collection because it covers wide access samples with minimum cost (Nachmias and Nachmias, 1996). In addition, the use of questionnaire facilitates data analysis as well as sustaining a high degree of privacy (Kirakowski, 2000; Robson, 1993).

This step was conducted in two parts: questionnaire design and validation. In the first part, the questionnaire was designed based on El Sheikh and Tarawneh (2007), Baharom et al. (2006) and McDonald (2001). The questionnaire consists of four main sections, namely the respondent's background, organizational background, development and measurement issues as well as Web application development and measurement practices. Each section has several set of questions in order to achieve the related survey objective. Details about the objective, content and the source of each question are shown in Appendix F.

The second part is the questionnaire validation that involved two activities, construct and content validity. The construct validity was performed by three developers (experts). The selected developers have at least five years of experience in Web application development in UUM Computer Center. The content validation process was conducted via an interview based on a list of questions that related to the correctness, completeness and readability. The questionnaire was given to the expert with the cover letter (refer to Appendix B).

For the content validity, a pilot study was conducted to find out whether the respondents understand the questions and the time taken to answer the questionnaire is sufficient.

Step 2.2 Respondents identification

This step aims to determine the list of respondents and sampling type. The population of this study is the SSF in Jordan. A list of this organization was obtained from the Ministry of Industry and Trade as well as the Jordan business directory Web site. The sample type that was use in this study is the systematic sampling technique. Details on this step are further explained in Chapter Four.

Step 2.3 Questionnaire Distribution

The aim of this step is to identify the data distribution methods. The common methods used for distributing the questionnaires are: postal, email and face-to-face interview, which were conducted to increase the response rate. For the postal and e-mail, respondents were given one or two weeks to fill up the questionnaires. Through the face-to-face interview, respondents can answer the questions with the researcher guidance. The actual number of respondents' rate was calculated after ignoring the incomplete, none answered and lost questionnaires. As a result, only seventy five fully answered questionnaires were collected and ready to be analyzed in the actual survey.

Step 2.4 Data Analysis

This step describes the survey results analysis. The collected data were analyzed using the SPSS package. The statistical methods used in this study are frequencies, mean, cross tabulation, multiple responses and hierarchical clustering. Details on the results of the analysis are shown in Chapter Four.

3.3.3 Methodology Construction

This phase aims to propose a new monitoring Agile based Web application methodology for SSF. The methodology focused on the quality assuring and monitoring during the development. The study adapted the Plan-Do-Check-Act (PDCA) method to construct the methodology. The PDCA, also known as the Deming cycle or Shewhart cycle, is an iterative four steps management method used for controlling and continuously improve the processes and product quality (Kao et al., 2010). This phase involved four steps:

- 1- Extend the Scrum method by adding the important XP elements.
- 2- Enhance the design phase of the Extend Agile method.
- 3- Construct a monitoring mechanism.
- 4- Organize the components of the methodology using PDCA.

Step 3.1: Extend the Scrum method by adding the important XP elements

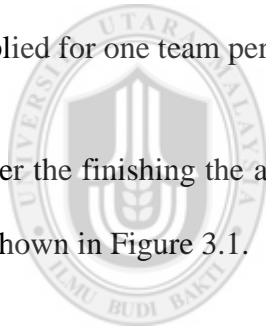
This step was performed based on the extension-based strategy. The step consists of two sub steps: XP and Scrum analysis. The analysis process was performed by comparing XP and Scrum based on specific criteria such as the development process, project management, requirements, testing, design and team structure. The criteria are considered as the differences between XP and Scrum (McDonald and Welland, 2001a; Deshpande et al., 2002; Redouane, 2004; Abran et al., 2004; Haung et al., 2008; Qumer & Henderson-Sellers, 2008). The development process criteria include the common Agile development practices. The project management criterion relates to the planning, staffing, monitoring and controlling activities that should be performed in parallel with the development process. The requirement criterion relates to the way of collecting requirements and from whom. The testing criterion relates to the deployment of the necessary testing practices during the process by separated team. The design criterion relates to the design approach that each method supported. The team structure criterion relates to the team size and number of teams that each method supported.

On the other hand, Scrum only satisfies the project management activities, but not the development criterion. Scrum did not have a measurement mechanism that monitors the quality of process and product. Scrum is good in requirement gathering as it uses product backlog collected by the product owner. However, nothing

mentioned about the design and testing in Scrum. Lastly, Scrum is good for multiple small development teams.

The analysis results show that even though it satisfies the development process criterion, XP is lacking in applying the project management criterion. XP also did not have a measurement mechanism that can monitor the process and product quality. In addition, XP is good in collecting requirements and testing because it uses user stories and TDD respectively. Moreover, XP depends on simple design practice and does not support any design method to deal with the design complexity of Web applications. Lastly, XP concentrates on small development teams that can be applied for one team per project.

After the finishing the analysis, the extension process begins. The extension process is shown in Figure 3.1.



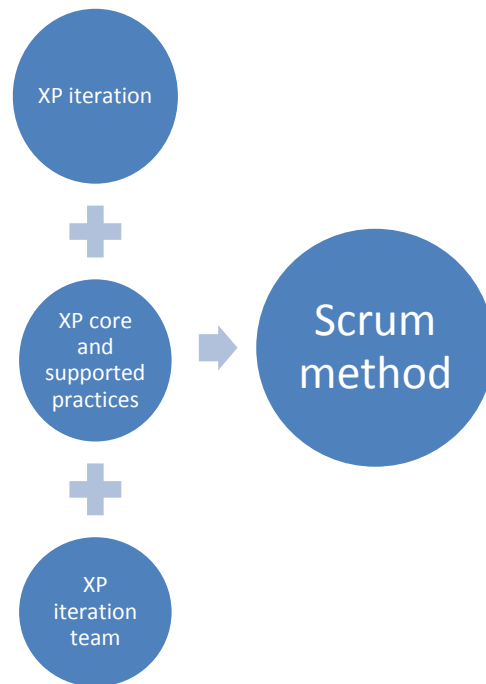


Figure 3.1. The extension process

The main objective of extending the Scrum by adding the important XP elements is to cover the development and management sides for each method.

The extension process started by adapting the Scrum as a base for the extended Agile method. Consequently, three process phases will be used that include planning, development as well as integration and maintenance phases.

The second activity of the extension process is to alternate the sprint from the Scrum with the XP iteration activities. The iteration activities are analysis, code, design and test. The duration of the iteration will be two weeks.

The third activity is integrating the core and supported XP practices. This was done by integrating the core and supported XP development practices to the combination method. The core XP practices are pair programming, TDD, coding standards and refactoring. The supported XP practices are continuous integration, metaphor, small release, collective ownership and simple design.

The fourth activity of extending the Scrum is the structuring of the development team. This activity was done by merging the master and product owner roles to the development team to the XP iteration team which consist of two programmers and one tester. The output of this step is the Extended Agile method that clearly described in Chapter Five.

Step 3.2: Enhance the design phase of the Extended Agile method by adding a simple design method to the combined method.

The steps of the design method were identified in step 1.3. The design method will be performed in the first planning meeting, which is held once per a project. The design features of the whole product will be described in this method in terms of the conceptual, navigational and implementation designs. After implementing this method, the PO will select the items needed to be entered in the first iteration. This method will be performed by the development team and customer (PO).

Step 3.3: Construct a measurement mechanism for monitoring the quality of the process and product.

This mechanism used goal oriented monitoring method (GOMM) for performing the measurement. The GOMM works based on the Lightweight GQM that was selected

from step 1.2. Two developers will be involved in performing the monitoring process. One developer is responsible for collecting data and the other for data analysis and presenting the feedback report. The data for measurement should be prepared by the data owner to decrease the data collection time. The feedback report will be presented to the management once the data are analyzed and processed. The whole team should attend a training session to learn how to perform the measurement during the process. In this session, the roles of the measurement team will be clearly defined, the goals of the measurement will be identified and prioritized and the role of the data owner in the measurement will be clarified.

The mechanism consists of two important parts: the development of guidelines and metrics. The guidelines are activities that should be performed during the development process such as planning, definition, data collection and data analysis.

The metrics used in the GOMM are quantitative and qualitative. These metrics should be performed to measure the quality of the product and process respectively.

For the product quality, the GOMM takes the organization improvement goals such as the quality improvement, budget reduction, shorter development cycle as well as the productivity increment, and formulate them into questions. Next is to define the suitable metrics to answer these questions. Finally, these goals together with the practice monitoring goals will be measured quantitatively. For monitoring the process quality, a set of factors was identified based on the literature review. These factors are effectiveness, adaptability, compatibility, accessibility, applicability, changeability and supportability. Each factor will be measured by a set of practices (metrics). Each practice or metric will have a score. This score ranged from 0 to 4

Lickert scale where 0 = never, 1 = rarely, 2 = sometimes, 3 = often and 4 = always. Then, the mean will be calculated for each factor practices to come up with a final score for each quality factor. Then the average will be divided by (4) the highest value of the score. The result will be multiplied by 100%. Based on this percentage, each factor will be assessed based on the (NPLF) rating scale that adapted from ISO/IEC 15504, where N = not achieved (0 – 15%), P = partially achieved (>15-50%), L = largely achieved (> 50 -85%) and F = fully achieved (> 85- 100%) which demonstrate fulfillment of the process factors. The data obtained from applying these metrics will be formulated as a feedback report to the management to facilitate them in making decisions.

Step 3.4 Organize the components of the methodology using PDCA

After completing the combination and enhancing steps, the components of the new methodology should be identified clearly. The core component of the new methodology is the process. The phases of the process depend on the PDCA method as this method is used for controlling and continuously improve the quality of the processes and product (Kao et al., 2010; Jani, 2011). In addition, the process of the PDCA can be performed under the Agile perspective, particularly in the Scrum (Quaglia, & Tocantins, 2011). Consequently, the process will start with the (Plan) phase that clarifies the planning process for the two sides of the development and measurement. The (Do) phase will describe the development side which relates to the iteration activities. The (Check) phase will be performed in the measurement side for monitoring the quality of the process and product. The (Act) phase will be performed on the development and measurement (monitoring). For the development

side, the next task is to determine the action that will be done after the iteration, whether the development team integrate the increment to the product and go to the other iteration or issue the final version of the product. For the measurement side, the next action is to provide the feedback for the management and the development team. If the iteration is final, a feedback report will be submitted to the management. The use of the PDCA phases is to guide the process of organizing the components of the new methodology as the process should be performed based on methods. These methods are the combined XP and Scrum, Web design method and GOMM. The methods consist of activities and practices. The activities of the method will be supported by specific tools. Finally, the process of the new methodology will be performed by a team. Therefore, the components of the new methodology include the process, methods, practices, tools and team structure. The main output of this phase is the new monitoring oriented Agile based Web application development methodology for SSF.

3.3.4 Methodology Evaluation

The aim of this phase is to evaluate the proposed methodology using expert review and case study respectively. Apart from that, this study also performed a yard stick validation to ensure strength and weakness of the proposed methodology.

Step 4.1 Verification

The aim of this step is to verify the comprehensiveness, understandability and feasibility of the new methodology components. This will be achieved using the

expert review method. The reasons of using the expert review are because it is useful for studying a limited number of cases; and it is very helpful to take the academic and practitioners point of views about the proposed theory (Blaxter et al., 2010; Yin, 2003). In addition, the contributions and opinions can be received from a group of experts who are not in the same place (Murry & Hammons, 1995). Furthermore, the expert review method allows the participant to think deeply and gather further information about the theory between the rounds (Grobbelaar, 2007).

This method was carried out by performing the Delphi technique activities as shown in Figure 3.1. The Delphi technique was selected because it is considered as the best technique to achieve consensus among the experts, is widely accepted method to achieve convergence of perspectives regarding knowledge request from experts within specific domains, and allows the researcher to gain high reliability data from the specified experts. It is performed by several rounds or iterations (feedback) designed to harmonize the experts' opinion (Hallowell & Gambatese, 2010; Rowe & Wright, 1999).

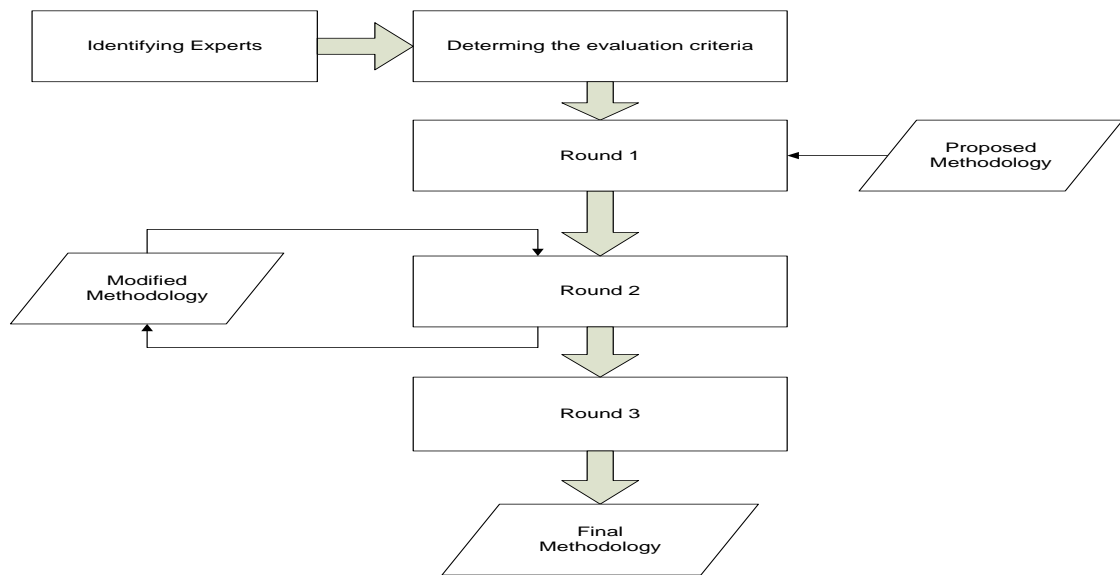


Figure 3.2. The Delphi technique steps

In this study, the Delphi technique was performed based on the following activities:

- i. Identifying the experts: Hallowell and Gambatese (2010) suggested that the experts should hold at least one of these characteristics: i) have a PhD. or any advanced degree, ii) faculty members at an accredited university, iii) authorship, and iv) have at least 5 years of experience. In this study, 30 experts were identified from different countries and were contacted through e-mail and only 12 of them accepted to review the proposed methodology. Unfortunately, after the first round, four experts withdrew from continuing the verification process. Therefore, only 8 experts have completely participated in reviewing and evaluating the proposed methodology. There are four domain experts and four knowledge experts were involved in the

process. As recommended by Hallowell and Gambatese (2010) and Rowe and Wright (1999), these numbers of experts are considered adequately enough to proceed the process. The identified experts represent different environments from the two countries: Malaysia (3) and Jordan (5) (Table 3.2). The domain experts should have at least 3 to 5 years' working experience in developing Web application. While the knowledge experts were identified from PhD holders who have at least 5 years of teaching experience in Software Engineering courses and also have many publications related to the field of Web development, Agile development and software measurement.

Table 3.2
Experts Profile

	ID	Qualifications	Expertise	Years of Experience	Institution
Knowledge	Exp 1	Ph. D	Requirements Engineering, Organizational Analysis, Agile development and software measurement	9 years	University of Malaya
	Exp 2	Ph. D	Empirical Software Engineering, Agile Software Methodology and Software Engineering Software Quality	5 years	Universiti Utara Malaysia
	Exp 3	Ph. D	Software engineering, Web development, Agile development and Software process improvement.	28 years	The Arab Academy for Banking and Financial Sciences
	Exp 4	Ph. D	Web development, Agile development and SSF.	8 years	Al-balqa Applied University (Jordan)
Domain	Exp 5	BS.c.	Web developer for small teams	12 years	University Teknologi Mara (UiTM)
	Exp 6	BS.c.	Web developer	12 years	AL al-Bayt University

Exp 7	High diploma of computer information system	Programming, testing and quality assurance	15 years	University of Jordan
Exp 8	BS.c.	Web developer	14 years	New York Institute of Technology- NYIT Amman, Jordan

ii. Determining the evaluation criteria: Three criteria were used to verify the proposed methodology. These criteria include comprehensiveness, understandability, and feasibility as suggested by Behkamal et al., (2009), Genero et al., (2008), Kunda, (2002) and Kitchenham et al, (1997). The following are the descriptions of the criteria.

a. Comprehensiveness: is the inclusion of the important parts or factors to achieve the desired results (Behkamal et al., 2009). This criterion determines if the methodology components such as activities, methods, practices, tools and team structure are covered Web applications development and measurement process.

b. Understandability: is “the capability of the component to enable the user to understand whether the component is suitable, and how it can be used for particular tasks and conditions of use” (Bertoa et al., 2006). In addition, this criterion is to evaluate the models from the standpoint of software engineering to be clear and unambiguous (Behkamal et al., 2009; Genero et al., 2008). Based on this criterion,

the methodology the structural components should be correct, clear and well organized (genera et al., 2008).

- c. Feasibility: this criterion measures the appropriateness of the methodology for the audience (Kunda, 2002).

These criteria were used to construct a questionnaire that consists of several questions to verify the components of the proposed methodology.

The questions were adapted from the previous studies such as Kunda (2002), Behkamal et al (2009), and Kitchenham and Pickard (1998).

- iii. Conduct Round one (send methodology to the experts): the proposed methodology was sent to the expert via email with agreement paper to review and answer the questionnaire given. The responses were analyzed to come up with the experts' suggestions and comments on each methodology component.

- iv. Conduct Round two (refine the methodology): this step was performed by taking the experts' comments and verifies the new methodology accordingly. This step may take one to three rounds until the expert satisfied.

- v. Conduct Round three: send the methodology back to the experts to receive the final agreement about the modifications.

The result of the verification step is the verified monitoring oriented Agile based Web application development methodology for SSF.

Step 4.2: Validate the proposed methodology using the case study and yardstick method.

After the new methodology has been verified by the experts, it needs to be validated. Validation “is the process of determining whether a model or framework is an accurate representation of the real world from the perspective of the intended usage” (Thacker et al., 2006). Two approaches were used to perform the validation step: i) case study, and ii) yardstick validation.

For the case study, the validation process was performed through the following activities:

1. Selection of the organization that can participate in validating the methodology. The organization was selected based on the following criteria which are: size of organization, has experience in using agile development methods, the current projects involved with developing Web application, and its willingness to cooperate in the validation process.
2. Identifying the factors for validation the proposed methodology. The aim of the case study is to validate the effectiveness of the MOGWD methodology that includes set of factors. The factors were identified by referring to Davis (1989), Kunda (2002) and Kitchenham and Pickard (1998) as discussed in the literature review. These factors are: Gain satisfaction, Interface satisfaction, Task support satisfaction, Perceived usefulness and Perceived ease of use. Each factor will be measured by certain variables (items) as

shown in Table 3.3. These variables were used to construct the validation form.

Table 3.3
Factors for validating the Effectiveness of the Proposed MOGWD Methodology

Validation Factors	Variables	Source
Gain satisfaction	<ul style="list-style-type: none"> - Decision support satisfaction - Comparing with current method - Clarity - Appropriateness for task 	Kunda (2002), Kitchenham and Pickard (1998)
Interface satisfaction	<ul style="list-style-type: none"> - Internally consistent - Organization (Well organized) - Appropriate for audience - Presentation (readable and useful format) 	Kunda (2002), Kitchenham and Pickard (1998)
Task support satisfaction	<ul style="list-style-type: none"> - Ability to produce expected results - Completeness - Ease to implementation - Understandability (easy to understand) 	Kunda (2002), Kitchenham and Pickard (1998)
Perceived usefulness	<ul style="list-style-type: none"> - Using MOGWD methodology enables you to accomplish your tasks more quickly. - Using MOGWD methodology improve the performance of your work - Using MOGWD methodology makes performing your tasks easier - MOGWD methodology is useful to your work - Using MOGWD methodology increases your productivity 	Davis (1989)
Ease of use	<ul style="list-style-type: none"> - Learning the MOGWD methodology is easy for you - Do you find it easy to use MOGWD methodology to do what want to do - The MOGWD methodology is flexible to interact with - Your interactions with the MOGWD methodology clear and understandable - It is easy for you to become skillful in using MOGWD methodology - The MOGWD methodology is easy to use 	Davis (1989)

3. Prepare the case study toolkit. The toolkit includes specifications on how the team will perform the new methodology. In addition, it consists of overview about MOGWD methodology, quantitative metrics checklist, quantitative metrics indicators, qualitative metrics and validation form. The quantitative metrics checklist, qualitative metrics and validation form are shown in appendix I, appendix J and appendix K. However, the quantitative metrics indicators list is shown in section 6.2.4.1, Table 6.9.
4. Data collected through interviews and document analysis. The interview method was selected as it is flexible and adaptable in order to provide deeper understanding and useful information that helps the practitioners to explore complex issues (Sekaran & Bougie, 2010). The interview was supplemented with a toolkit that provides information on how the data will be collected and analyzed.

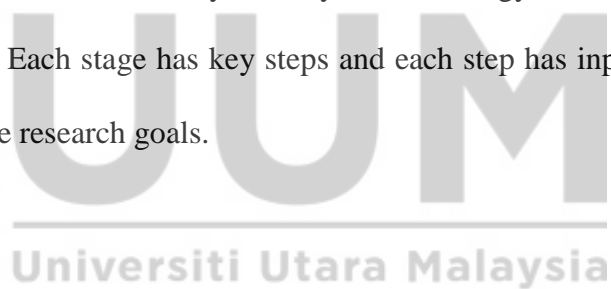
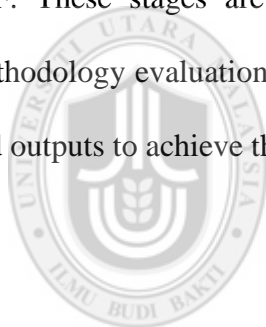
The MOGWD methodology is supported by the use of a prototype tool (MO-PT) to perform the monitoring process in a more systematic way. The prototype tool was developed using PHP language as the programming language. The MO-PT consists of two parts: i) front end that includes the interface for the users and ii) back end that includes the database and server (See section 6.3.1.1).

The yardstick validation was performed by comparing the proposed methodology with ideals or baseline methods in the same field. Using this type of validation will increase the reliability of the proposed methodology. In particular, if the

methodology components match the baseline methods in the same field, it can be taken as a proof that the model performs acceptably (Carson, 2002; Sargent, 2012). The yardstick validation starts by determining the ideal or the baseline methods in the field of study. Then, define the comparison criteria. Finally, the strengths and weaknesses of the proposed method are determined, and the results are discussed.

3.4 Summary

This research methodology is described in four phases, which are used to construct the monitoring oriented Agile based Web application development methodology for SSF. These stages are theoretical study, survey, methodology construction and methodology evaluation. Each stage has key steps and each step has inputs, process and outputs to achieve the research goals.



CHAPTER FOUR

SURVEY

4.1 Introduction

This chapter aims to present the results of a survey that was conducted in Jordan to investigate current development and measurement practices in SSF. Questionnaire was used as an instrument for collecting data. The gathered data were analyzed using multiple statistical methods such as: frequencies mean, cross tabulation, multiple responses and hierarchical clustering. The actual findings of this survey were used to build a new methodology for developing Web application.

4.2 Questionnaire Structure

The questionnaire consists of four main sections: respondent's background, organization background, software development and measurement practices and Web application development and measurement practices. The questionnaire included forty three (43) questions that included multiple choice and five likert scale questions as shown in Appendix A. This part will provide a summary of each section as followed.

4.3 Respondent's Background

This section aims to determine the respondent's qualification. It includes three multiple choice questions that are related to the position, the activity involved and the years of experience of the respondent.

4.3.1 Organization Background

This section aims to study the background of the Jordanian SSF. It includes three multiple choice questions on type, sector and size of the companies.

4.3.2 Software Development and Measurement Practices

The aim of this section is to investigate software development and measurement practices that are currently used by SSF. This section includes twenty two multiple choice questions. The questions are related to the development, reuse, QA and measurement practices. The results of this section were used to identify the development and measurement issues that currently faced by SSF

4.3.3 Web application development and measurement practices

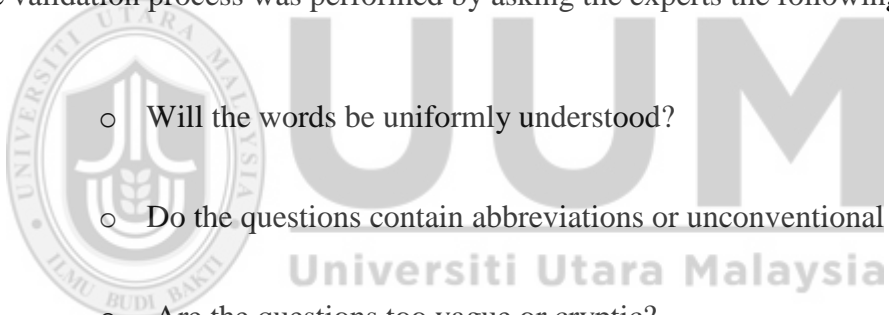
This section aims to investigate the current Web application development and measurement practices in SSF. It consists of seventeen five likert scale questions. The answers of the questions ranged from strongly disagree with the value (1) to strongly agree with the value (5). The practices included are the best Web application development and measurement practices such as a short life development cycle, multidisciplinary development team, requirement management, testing, maintenance, project management and quality management.

4.4 Questionnaire Validation

The questionnaire was validated through construct validity and content validity. These validation methods have been popularly used and described in Sekaran and Bougie (2010). A brief description of both methods is given in the following section.

4.4.1 Construct Validity

The questionnaire was validated by experts using face to face interview. The three experts were software developers from the UUM Computer Center. The questionnaire was validated in term of correctness; completeness and readability. The validation process was performed by asking the experts the following questions:

- 
- Will the words be uniformly understood?
 - Do the questions contain abbreviations or unconventional phrases?
 - Are the questions too vague or cryptic?
 - Are the questions too precise, biased or objectionable?
 - Are the answer choices mutually exclusive?
 - Has too much knowledge been assumed?
 - Are the questions technically accurate?
 - Is each question complete with enough details?

Feedback obtained from this step was used to improve the questionnaire.

4.4.2 Content Validity

The content validity of the questionnaire was done by conducting a pilot study. The aims were to ensure that respondents understand the questions, check the grammar, sentence structure and estimate the required time to answer the questionnaire.

In the pilot study, twenty three SSF were identified randomly. From each company, one respondent either a developer or project manager was selected. The pilot survey has determined that respondents were able to answer the questionnaire. Pilot study respondents advised for minor modifications on some items in the questionnaire. The feedbacks were used to refine the questionnaire.

4.5 Identify Respondents and Sampling Type

At this stage the questionnaire was refined and ready to be answered by the respondents. Regarding to the respondent identification, the list of SSF was determined by the Jordanian Ministry of Trade and Industry, and the Jordan Business Directory Website. The total number of SSF in Jordan is 769 companies. The systematic sampling technique was adopted because it is more convenient compared to other probability sampling techniques and it was calculated according to that population size equal to 256. Three hundred (300) questionnaires printed and ready to be distributed. The target respondents were identified by selecting the first respondent number, then select respondent number +3. For example, choosing respondent 1, 4, 7 and so on.

4.6 Questionnaire Distribution and Data Collection

The questionnaire with covering letter was sent to 300 Jordanian SSF. The targeted respondents were given one month period to answer the questionnaire. Mail, email, and face to face (interview) were used as an instrument for distributing and gathering the data.

After two weeks, a reminder letters were sent by mail and email, and sometime the telephone calling was used in order to improve the response rate.

Only 75 respondents were completely answered the questionnaire and able to analyze. However, out of 300 questionnaires, 188 questionnaires were considered "lost questionnaire" as they were not returned back due to that the respondents do not have time to answer or they travelled outside the country. Apart from that, 11 questionnaires were rejected as they were not completely answered by the respondents. These questionnaires and 24 questionnaires that are out of scope were excluded from the data analysis. Table 4.1 shows the whole number of questionnaires that were sent and the response rate.

Table 4.1

Questionnaire Response Rate

Description	Organizations	Rate (%)
Questionnaires sent	300	100%
Lost	188	63%
Received	112	37%
Usable responses	75	25%
Rejected	11	4%
Not small software firm or the size of company over 50	24	8%

4.7 Analysis and Results

The collected data were entered in Statistical Package for Social Science (SPSS) ver. 14.0 for analysis. This section describes the analysis results which conducted on seventy five SSF in Jordan. The results were presented in three sections, namely: demographic data, current development and measurement practices and the last section Web application development practices.

4.7.1 Demographic Data

The demographic data are presented in terms, respondents and organization background. The analysis results of this section will help to determine the characteristic of SSF in Jordan.

4.7.1.1 Respondents Background

In this section respondents were asked about their position, experience and the activity that they currently occupied.

- **Position and Experience**

Figure 4.1 demonstrates the distribution of respondent's position and the experience of years working in their companies. The data were analyzed using cross tabulation analysis. The results showed that out of 75 respondents, 55% have 3-10 years of experience and 21% are team leaders followed by software engineering process group member (15%), technical members (11%) and managers (6%). On the other hand, 41% of respondents have less than three years of experience and the majority

of them are software engineering process group member (20%), technical members (16%) and team leaders (5%). Lastly 4% of respondents have 10-20 years of experience 3% are manger and 1% are team leaders.

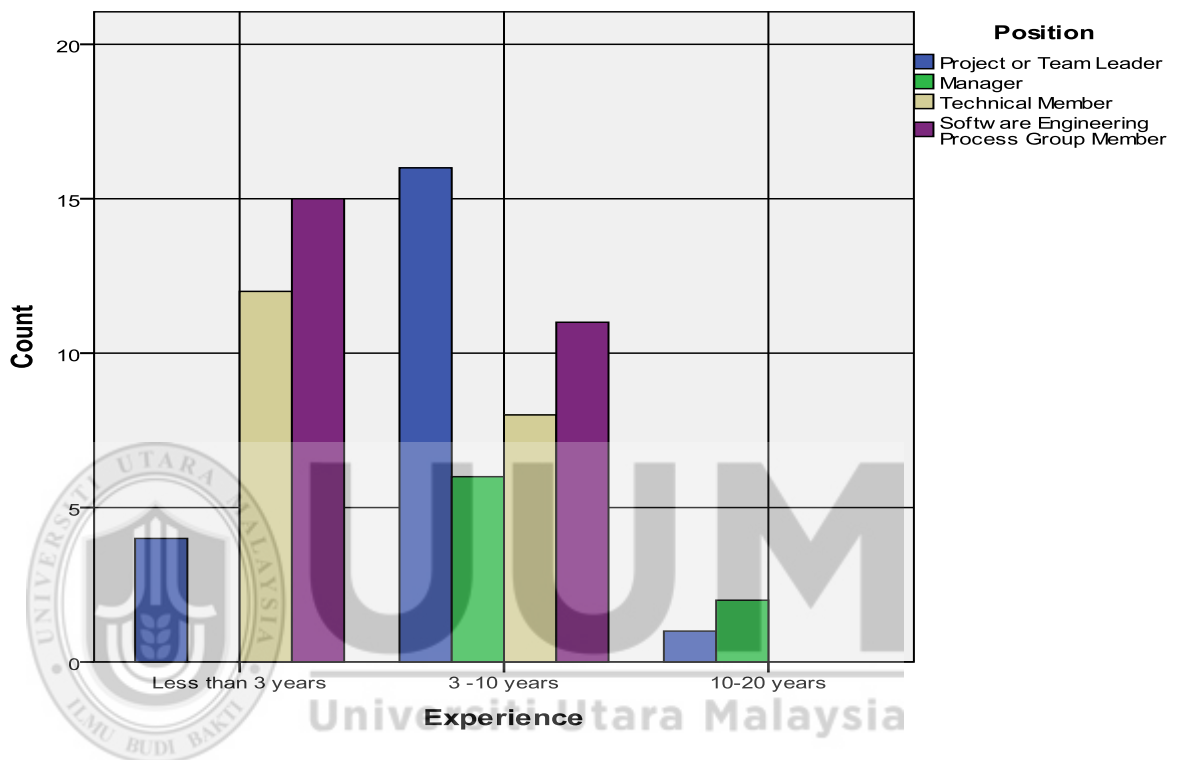


Figure 4.1. Respondents Position and Experience

- **Current activity**

In this section, respondents were asked about the activity they are involved in the development process. The results shows that 29% of the respondents are involved with code and unit test, followed by software design (28%), software requirements (20%), test and integration activities (9%), configuration management (8%). Software QA and software process improvement both occupied with the same percentage 3% of respondents. As shown in Table 4.2.

Table 4.2

Current Position Activities

Current Position Activities	Frequency	Percent
Software Requirements	15	20
Software Quality Assurance	2	2.7
Software Design	21	28
Configuration Management	6	8
Code and Unit Test	22	29.3
Software Process Improvement	2	2.7
Test and Integration	7	9.3
Total	75	100

4.7.1.2 Organization background

In this section, the respondents were asked about the type and the size of their companies. All respondents in this survey are from the private sector companies.

- **Organization size**

Respondents were asked to indicate the number of employees inside their companies. Table 4.3 describes that majority of companies in this study have 10 to 30 employees (48%), where 47% of the companies have 31 to 50 employees. However, only 5% of the companies have less than 10 employees.

Table 4.3

Numbers of Employees

Numbers of Employees	Frequency	Percent
Less than 10 people	4	5.3
10 - 30 people	36	48.0
31-50 people	35	46.7
Total	75	100.0

4.7.2 Current Software Development and Measurement Practices

This section aims to investigate the software development and measurement practices. The practices are related to the development, reuse, QA and measurement methods that currently used by SSF in Jordan. The results of this section identify the development and measurement issues that faced by SSF. Consequently, this section is categorized into two parts software development practices and software measurement practices.

4.7.2.1 Software Development Practices

This section aims to determine the development practices performed in SSF. Findings from the following practices are: philosophy used, development methods used, development method that developers familiar with, requirements method, programming language, testing, reuse and quality assurance.

- **Software philosophy**

Software philosophy: is “the style of a development process that the company refers to and it may cause the success and failure of any software company” (Wikipedia, 2011). Regarding to the software philosophy type, findings showed that the respondents followed their own philosophy (47%), code and fix (33%), Agile software development (13%) and waterfall (7%). As shown in Figure 4.2.

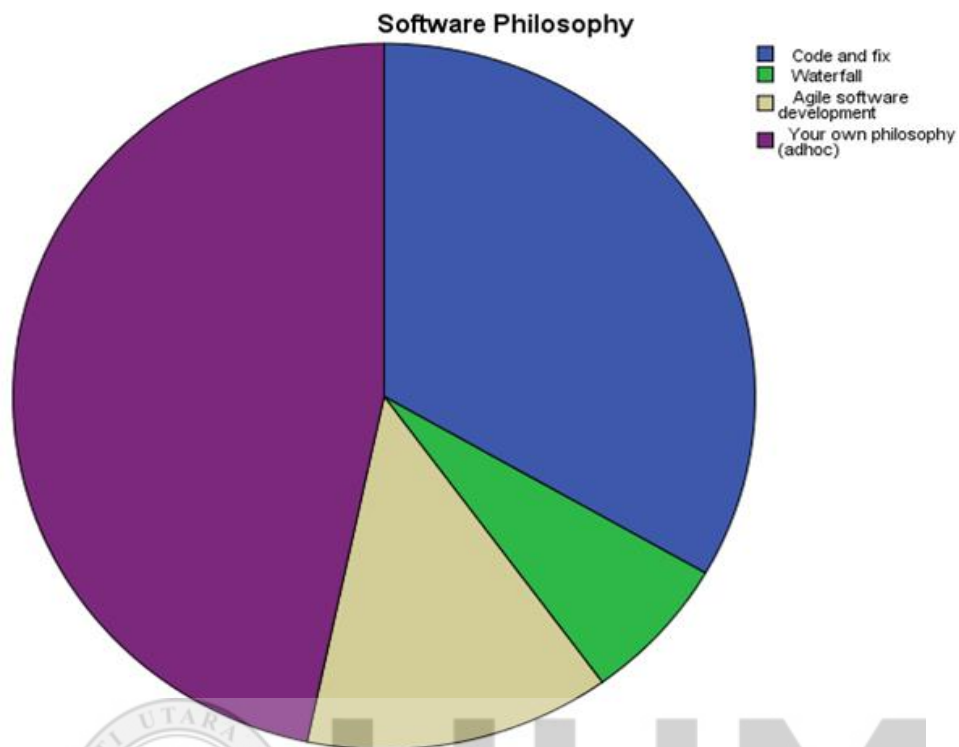


Figure 4.2. Software Philosophy

- **Methods that respondents are familiar with**

Software development method is “a framework that is used to structure, plan, and control the process of developing any software” (Pressman, 2009). Table 4.4 describes that the majority of respondents are familiar with Waterfall (71%) followed by XP (60%), Spiral model (29%), Scrum (27%), Prototyping (17%), DSDM (11%), Incremental (11%), AUP (9%), V-model (7%), FDD (4%), RUP (3%) and EUP (3%).

Table 4.4

Methods that Respondents are Familiar with

Development Methods that Respondents are Familiar with	Frequency	Percent
Waterfall	53	70.7
V- Model	5	6.7
Spiral model	22	29.3
RUP	2	2.7
AUP	7	9.3
DSDM	8	10.7
FDD	3	4
Incremental	8	10.7
Prototyping	13	17.3
Enterprise Unified Process (EUP)	2	4.3
XP	45	65.2
Scrum	20	26.6

- **Requirements Collection Method**

Respondents were asked about the methods or techniques that they use for collecting the requirements. The results indicated that the majority of respondents used interview methods (48%) followed by use case scenarios (33%), document reviews (12%), observation (5%) and questionnaires (1%) (Table 4.5).

Table 4.5

Requirements Collection Method

Requirements Collection Method	Frequency	Percent
Questionnaires	1	1.3
Interviews	36	48.0
Observations	4	5.3
Document reviews	9	12.0

Use case scenarios	25	33.3
Total	75	100.0

- **Requirements Specification Notation**

Requirements Specification Notation is “the way that the development team describes the software system that will be developed” (Leveson, 1994). SSF’s developers were asked about the notation that they use for presenting requirements specification. Table 4.6 reveals that the majority (47%) of respondents do not use any notation to present the requirement specification, 36% use semi-formal notation, 9% use informal notation and 8% use formal notation.

Table 4.6

Requirements Specification Notation

Requirements Specification Notation	Frequency	Percent
Formal	6	8.0
Semi-formal	27	36.0
Informal	7	9.3
No specific notation	35	46.7
Total	75	100.0

- **Programming Languages**

Findings showed that 73% of respondents use the object oriented languages, 23% of them use visual languages and 4% of them use the 4 GL programming languages.

Table 4.7 shows the results.

Table 4.7

Programming Languages

Programming Languages	Frequency	Percent
4GL	3	4
Visual languages	17	22.7
Object oriented	55	73.3
Total	75	100

- **Testing Type**

Software testing is “a method of assessing the functionality of any software” (Basili & Selby, 1987). Table 4.8 shows the results. It can be seen that majority of respondents use unit test (81%) followed by the acceptance test (52%), whole system tests (41%), code coverage test (35%), no test required (13%), alpha test (9%), regression test (7%), beta test (4%) and usability test (1%).

Table 4.8

Test Types

Testing Types	Frequency	Percent
Unit Tests	61	81.3
System Testing	31	41.3
Acceptance Tests	39	52
Usability Testing	1	1.3
Beta Testing	5	6.7
Code Coverage Tests	26	34.7
Regression Testing	5	6.7
Alpha Testing	7	9.3
No tests are required	10	13.3

- **Testing Process Stage**

The results of Table 4.9 demonstrate that most (56%) of respondents perform the testing process at the end of the coding phase, 24% performed the testing as soon as possible software project were acquired, others (12%) performed the testing for documentation or other related tests. Only 4% of the respondents used testing while integrating major software modules or when implementing the final acceptance test.

Table 4.9

Testing Process Stages

Testing Process Stages	Frequency	Percent
The end of the coding phase	42	56
Early as soon as possible software projects were acquiring	18	24
Documentation or element that can be tested	9	12
While integrating major software modules	3	4
When implementing the final acceptance testing	3	4
Total	75	100

- **Reasons for Not Using Any Development Method**

Respondents were asked why they are not using any method for developing Web applications. Most of the respondents mentioned that the current methods need specific training (75%). 71% claimed that the current methods need to form a specific team, 29% stated that the current methods consume more time, and 11% stated that no one in the company is familiar with any type of methods. However, 10% of the respondents mentioned that the current methodologies costly. Table 4.10 shows the results.

Table 4.10

Reasons of Not Using the Current Methods

Reasons of Not Using the Current Methods	Frequency	Percent
Nobody inside the organization familiar with any type of methods	7	11.1
Using any development method takes a lot of time	18	28.6
Consume a lot of money	6	9.5
Need a specific team to be performed	45	71.4
Need specific training to be performed	47	74.6

- **Reuse Types**

Software reuse can be defined as “the process of creating software systems from predefined software components” (Krueger, 1992). Table 4.11 reveals that 84% of the respondents reused the source code, 38% reused templates, 29% reused modules, 18% reused the design of document, 18% reused the documentation or specification, 16% reused media, 12% reused data, 10% reused Web pages, 3% reused feasibility studies and 1% reused Cost benefits calculators and estimation.

Table 4.11

Reuse Types

Reuse Types	Frequency	Percent
Source Code	61	83.6
Media	12	16.4
Templates	28	38.4
User Documentation/Specification	13	17.8
Modules	21	28.8
Cost benefits calculators and estimation	1	1.4
Feasibility Studies	2	2.7
Web Pages	7	9.6
Design Document	13	17.8
Data	9	12.3

- **QA Activities**

Quality assurance (QA) is any systematic process of checking whether a developed product is meeting specified quality requirements (Owens & Khazanchi, 2009). The respondents were asked about what kind of QA activities that they used. The results show that the majority of respondents performed the testing of Web application as QA activity (83%), code review (59%), development process audit (23%), configuration management audit (5%), functional configuration audit (5%), version description document (5%) and physical configuration audit (3%). Table 4.12 shows the results.

Table 4.12

Quality Assurance Activities

QA Activities	Frequency	Percent
Testing of Web-based Applications	62	82.7
Code review	44	58.7
Development Process Audit	17	22.7
Configuration Management Audit	4	5.3
Functional Configuration Audit	4	5.3
Physical Configuration Audit	2	2.7
Version Description Document	4	5.3

- **Performing QA Activities**

The respondents were asked about who is responsible for performing the quality assurance activities inside the company. The majority of respondents indicated that QA activities had been performed by the project team (80%), software assurance group (17%), and only 3% of them are performing QA Activities by other assurance group (Table 4.13).

Table 4.13

Performing QA Activities

Performing QA Activities	Frequency	Percent
Project team	60	80
Software Assurance Group	13	17.3
Other Assurance Group	2	2.7
Total	75	100.0

4.7.2.2 Software Measurement Practices

Software measurement: is “the process of using the appropriate measures of software artifacts such as requirements, designs, and source code that can be analyzed during project execution to reduce defects, rework and life cycle time” (Kettelerij, 2006).

This section aims to determine the following measurement practices: in which stage does the respondents performed there measurement process, what the domain of applications they usually use this measurement inside, as well as what type of development methods did they use and which method they use for performing these measurements.

- **Measurement Stage and Application Domain**

In this part respondents were asked about the stage of measurement that they perform with the development process and the type of Web application domain that are they currently use. Data was analyzed using the cross tabulation. Figure 4.3 shows that 65% of respondents were not using measurement during the development distributed in using the application domain of business information systems (31%), e-business in general (25%), personal Web pages (5%) and learning applications

(4%). On the other hand, 25% of respondents performed the measurement at the end of the coding phase, 17% of them are developing business information systems, 4% of them are developing personal Web pages and 4% of them are developing e-business in general. Furthermore, 9% of companies performed the measurement early, as soon as possible software projects were acquired, distributing in developing e-business in general (3%), business information systems (3%), and personal Web pages (4%).

This means that the most of respondents are not using the measurements at all, where the most application domains that had been developed inside their companies are business information system and e-business applications.

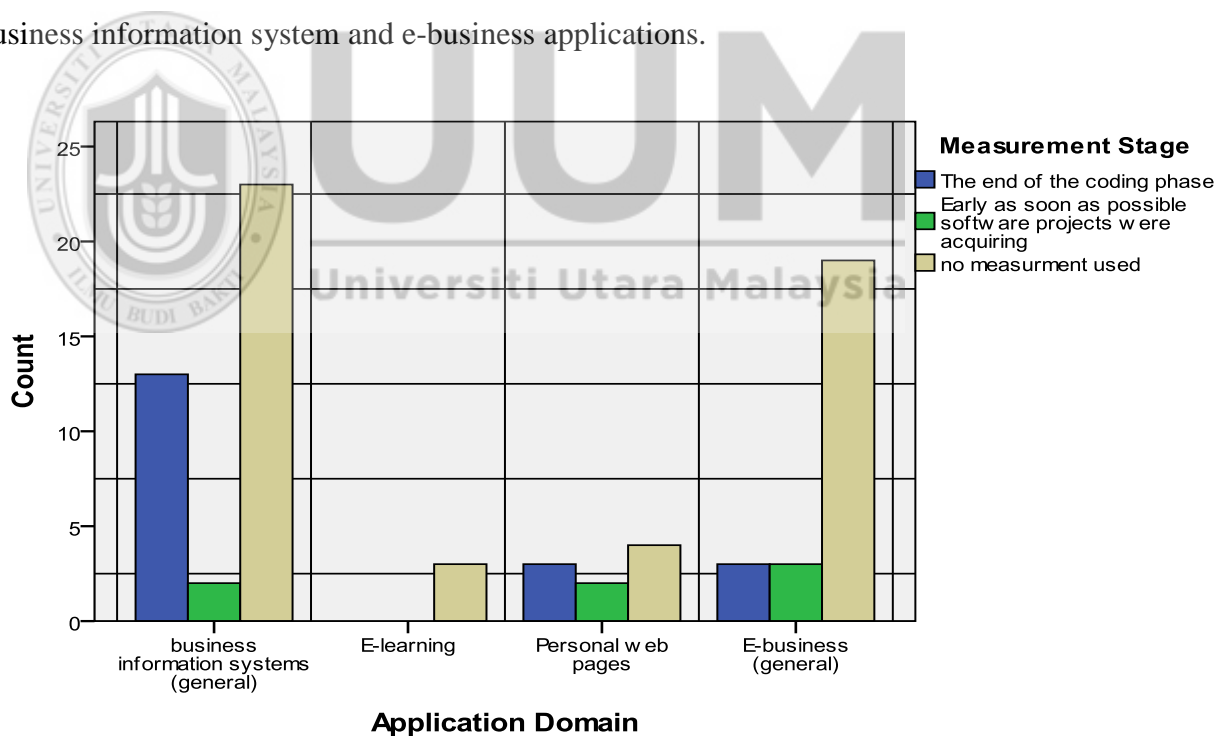


Figure 4.3. Measurements Stage and Application Domain

- **Measurements Stages and Size of Company**

Respondents were asked about measurement stage and the number of employees of each company. Cross tabulation analysis was used to gain the data. Results reveals that the majority of the respondents were not using the measurement (65%) distributed according to the number of employees as, 36% of them have 31- 50 employees size, 24% of them have 10-30 employee size and only 5% less than ten employees. Furthermore, 25% of the companies performed measurement at the end of the coding phase, which distributed as followed: 17% of them have 10-30 employees and 8% of them have 31-50 employees. Moreover, 9% of the companies used measurement early as soon as possible software projects were acquired. These companies are distributed based on the number of employee as, 7% of them have 10-30 employees and 3% have 31-50 employee size. Table 4.14 shows the results.

Table 4.14

Measurements stages and size of company

Measurement stage	No. of employees			Total
	Less than 10 people	10 - 30 people	31-50 people	
The end of the coding phase	0%	17.3%	8%	25.3%
Early as soon as possible software projects were acquiring	0%	6.7%	2.7%	9.3%
No measurement used	5.3%	24%	36%	65.3%
Total	5.3%	48%	46.7%	100%

- **Measurement Stage and Development Method Type**

Respondents were asked about the stage of performing measurement within the development process and the type development method that were currently used.

The data were analysed using cross tabulation analysis. The results illustrate that the majority of respondents did not use any measurements during the development (65%) distributed according to the development method used as, no development method used (47%), using Waterfall (8%), XP (4%), Scrum (4%) and Spiral (3%). Furthermore, 25% of the companies performed the measurement at the end of the coding phase. These companies distributed according to the development method that they used as no method used (7%), XP (9%), Waterfall (4%), Scrum (4%) and DSDM (1%). Moreover, 9% of respondents used measurement early, as soon as possible software projects were acquired. These companies are distributed according to the development method used as using XP (3%), Waterfall (1%), Scrum (1%) and DSDM (1%). See Figure 4.4. This means the majority of respondents are not use measurements and the majority of them also still not use any specific development method. This means the majority of respondents not use the measurement and the majority of them also still not a specific development method.

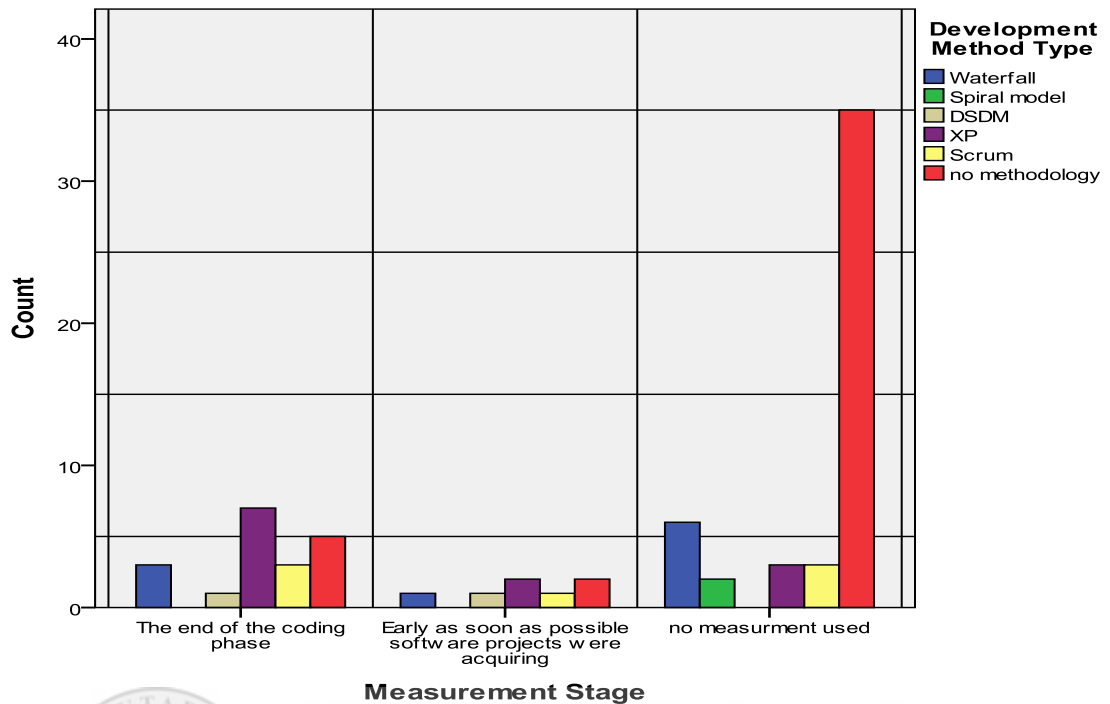


Figure 4.4. Measurement Stage and Development Method Type

- **Metric Type and Development Method Type**

Software metric: is a “quantitative or qualitative measure of some property of a piece of software or its specifications” (Kettelerij, 2006). In this section, respondents were asked about the metric type and the development methods that they currently used. The data were analyzed using cross tabulation. Table 4.15 demonstrates that the majority of the respondents are not using any specific type of metrics (67%). These companies are distributed according to the type of development methods that they used as the follows: no development method used (47%), Waterfall (8%), XP (5%), Scrum (4%) and Spiral (3%). Furthermore, 19% of the companies are using a line of code that's distributed as follows: no development method used (7%), using XP

(4%), Waterfall (3%), Scrum (3%) and DSDM (3%). This means that the majority of SSF still doesn't use any type of metrics while the majority of them still not use any systematic development method as well.

Table 4.15

Metrics Type and Development Methods Type

Metric type	Development method types						Total
	Waterfall	Spiral	DSDM	XP	Scrum	No methodology	
Use Case Points	0%	0%	0%	0%	0%	2.7%	2.7%
Constructive Cost Model	0%	0%	0%	1.3%	2.7%	1.3%	5.3%
Function Points	2.7%	0%	2.7%	6.7%	0%	0%	12%
Line of Code (LOC)	2.7%	0%	2.7%	4%	2.7%	6.7%	18.7%
Links Count	0%	0%	0%	1.3%	0%	1.3%	2.7%
No specific type of metrics	8%	2.7%	0%	5.3%	4%	47%	66.7%
Total	13.3%	2.7%	2.7%	16%	9.3%	56%	100%

- **Development methods and Measurement Methods**

Measurement method: “is the way that the company used for performing the measurement process” (Kettelerij, 2006). Respondents were asked to indicate the type of metric and what measurement method they perform. The data were analyzed using Cross tabulation. Based on Table 4.16, 56% of respondents do not use any development method. Whereas, 16% of the respondents using XP distributed based on the measurement methods that they use as the follows: not using any measurement method (8%), using GQM (4%), using PSM (3%) and only 1% preferred to use QFD. Furthermore, 13% of respondents using Waterfall distributed according to the measurement method they use as, not using any measurement

method (11%) and using PSM (3%). Moreover, 9% of the respondent using Scrum, distributed according to the measurement method that they use as, 7% of them aren't using any measurement method, GQM (1%) and PSM (1%). Based on these results, it obvious that the majority of the respondents did not use any development methods or measurement methods. However, the respondent that used development methods they concentrate on XP, Waterfall and Scrum. And the majority of the respondents whom apply measurement methods during the development used GQM.

Table 4.16

Development method and Measurement Methods

Development Method	Measurement Methods				Total
	GQM	PSM	QFD	No method	
Waterfall	0%	2.7%	0%	10.7%	13.3%
Spiral model	1.3%	0%	0%	1.3%	2.7%
DSDM	1.3%	0%	1.3%	0%	2.7%
XP	4.0%	2.7%	1.3%	8.0%	16.0%
Scrum	1.3%	1.3%	0%	6.7%	9.3%
no method	12.0%	1.3%	4.0%	38.7%	56.0%
Total	20.0%	8.0%	6.7%	65.3%	100.0%

- **Why Organization Does Not Use Measurements**

In this part, respondents were asked to address the reasons why they did not use any measurement. Respondents indicate that the majority of companies were not aware of performing software measurements (68%), software measurements need a specific team (57%), no one in the company familiar with software measurements (47 %), using measurement consumed time (19%) and only 13% of respondents said that using software measurement is costly. See Table 4.17.

Table 4.17

Why Organization Does Not Use Measurements

Reasons of not using specific measurement	Frequency	Percent
Nobody inside the company familiar with software measurement	29	46.8
Take a lot of time to employ software measurement	12	19.4
Consume a lot of money	8	12.9
Need a specific team to perform	35	56.5
Your organization is not aware to perform software measurement	42	67.7

4.7.3 Web Application Development and Measurement Practices

SSF should pay attention to several practices during the development process. The practices related to the development process, team, project management and quality management. This part aims to identify the current Web application development and measurement practices in SSF. Table 4.18 describes the practices and the variable name of each practice that used in SPSS.

Table 4.18

Practices and SPSS Variable Name

No	Practices	Variable Name
1	Does your development process of Web application copes with time pressure?	D1
2	Does your development process of Web applications clarify that all involved in this process understand their roles and responsibilities?	D2
3	Does the development team ensure that the development process must be performed with minimum design and quick prototype?	D3
4	Does each Web project have a nominated Web project manager?	D4
5	Does your Web project plan perform the budget estimation?	D5
6	Are the requirements collected directly from the user or and the manager?	D6
7	Are design notations used in Web design?	D7
8	Does the development process ensure that all components of the Web application such as page, code, site, navigation and services are being tested by test cases	D8

	generated according to requirement specifications?	
9	Is the testing process carried out or performed by the development process team?	D9
10	Do the developers pay attention to the quality management and standards such as usability and user interface design when developing Web applications in your company?	D10
11	Is an independent testing conducted by users (or appropriate representatives) under the guidance of Software QAA before any system or enhancement goes live?	D11
12	Is there a procedure for controlling changes to the Web application requirements, designs and accompanying documentation?	D12
13	Is a change control function established for each Web project?	D13
14	Is there a documented procedure for estimating the Web application's size (such as "Lines of Source Code") and thus for using productivity measures?	D14
15	Is a formal procedure used to produce the Web development effort, schedule, and cost estimates?	D15
16	Is there a required training program for all newly-appointed Web managers which is designed to familiarize them with in-house Web project management procedures?	D16
17	Is there a procedure for maintaining awareness of the state-of-the-art in case of Web engineering technology?	D17

The practices were listed and enter to the SPSS to perform the factor analysis to group it into specific and related groups. However, according to Palant (2007) and Tabachnick, & Fidell (2007) indicated that the sample size, which sufficient for performing factor analysis should be over 150, which means this study not adequate to apply factor analysis as the sample size is 75 cases. Therefore, other technique should be used for group this set of practices, cluster analysis was chosen for this purpose.

Cluster analysis is a technique used for combining variables into groups. These groups are: firstly, homogeneous i.e., variable in the group are similar to each other. Secondly, variables in each group should be different from the other groups (Chatfield & Collins, 1990; Johnson, & Wichern, 1992). One of the common

techniques that's used for grouping variables which exist in SPSS is hierarchal clustering. This technique used different methods. One of the most known and commonly used methods is Ward's method. Using this method, all possible pairs of clusters are combined and the sum of the squared distances within each cluster is calculated. This is then summed over all clusters. The combination that gives the lowest sum of squares is chosen (Chatfield& Collins, 1990; Dingsøyr et al., 2012). In addition, the distance between shorter distances implying greater closeness correlation between the variables (Dingsøyr et al., 2012). In this study the distance means the number of cases (respondents) that have been analyzed.

Consequently, the results were obtained from the hierarchal clustering and Wards method shows that these practices are categorized in seven groups or clusters as shown in Figure 4.5. This figure represents the process of performing the hierarchal clustering and the output clusters. It's called dendrogram. cluster 1 contains the practices (D6, D12 and D13) which are related to requirements phase, cluster 2 contains the practices (D10 and D11) which are related to the quality issues, cluster 3 contains the practices (D5, D14and D15) which are related to the measurement practice, cluster 4 contains (D3, D7 and D17) which are related to the design phase, cluster 5 contains the practices (D4 and D16) which are related to the management, cluster 6 contains the practices (D1 and D2) which are related to the development process and finally cluster 7 contains (D8 and D9) and these practices related to the testing process.

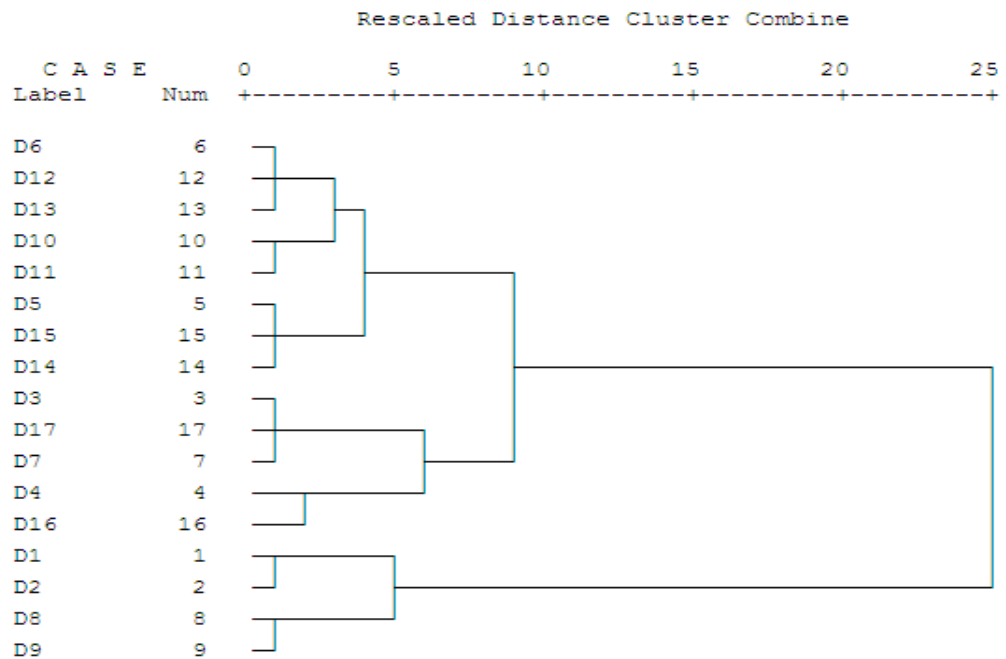


Figure 4.5. Dendrogram

Respondents were asked to rank the degree of performing these practices inside their companies. Therefore, Five Likert scales ranging from strongly disagree (value 1) to strongly agree (value 5) were used to describe the degree of acceptance for applying these practices.

Results were obtained by calculating the mean score and selecting the appropriate interval that represent the actual mean. An appropriate interval scale was required to represent all levels of acceptance. The interval was calculated by the following equation:

$$\text{Appropriate interval} = \text{number of interval between values} / \text{number of variable.}$$

$$\text{Appropriate interval for the study} = (4/5) = 0.8$$

Scales representation for the degree of acceptance for each practice is shown in Table 4.19. This internal was used and recommended by many researchers as such Ali et al. (2011), Bidad and Campiseño (2010) and Ahmad (2008).

Table 4.19

Internal Representations for the Degree of Acceptance

Mean interval presentation	Degree of acceptance
From 1 to 1.80	Strongly Disagree
From 1.81 to 2.60	Disagree
From 2.61 to 3.40	Neutral (Don't Know)
From 3.41 to 4.20	Agree
From 4.21 to 5	Strongly Agree

Table 4.20 illustrates the mean values of each group of practices and the degree its acceptance.

Table 4.20

Current Web Applications Development Practices

Requirements practices	Mean value	Degree of acceptance
1. Are the requirements collected directly from the user or and the manager? (D6)	2.16	Disagree
2. Is there a procedure for controlling changes to the Web application requirements, designs and accompanying documentation? (D12)	2.16	Disagree
3. Is a change control function established for each Web project? (D13)	2.17	Disagree
Quality practices	Mean value	Degree of acceptance
4. Do the developers pay attention to the quality management and standards such as usability and user interface design when developing Web applications in your company? (D10)	1.99	Disagree
5. Is an independent testing conducted by users (or appropriate representatives) under the guidance of Software QAA before any system or enhancement goes live? (D11)	1.97	Disagree

Measurements practices	Mean value	Degree of acceptance
6. Does your Web project plan, perform the budget estimation? (D5)	1.85	Disagree
7. Is there a documented procedure for estimating the Web application's size (such as "Lines of Source Code") and thus for using productivity measures? (D14)	1.85	Disagree
8. Is a formal procedure used to produce the Web development effort, schedule, and cost estimates? (D15)	1.83	Disagree
Design practices	Mean value	Degree of acceptance
9. Does the development team ensure that the development process must be performed with minimum design and quick prototype? (D3)	2.44	Disagree
10. Are design notations used in Web design? (D7)	2.47	Disagree
11. Is there a procedure for maintaining awareness of the state-of-the-art in case of Web engineering technology? (D17)	2.40	Disagree
Management practices	Mean value	Degree of acceptance
12. Does each Web project have a nominated Web project manager? (D4)	2.64	Neutral
13. Is there a required training program for all newly-appointed Web managers which is designed to familiarize them with in-house Web project management procedures? (D16)	2.31	Disagree
Process practices	Mean value	Degree of acceptance
14. Does your development process of Web application copes with time pressure? (D1)	3.53	Agree
15. Does your development process of Web applications clarify that all involved in this process understand their roles and responsibilities? (D2)	3.52	Agree
Test practices	Mean value	Degree of acceptance
16. Does the development process ensure that all components of the Web application such as page, code, site, navigation and services are being tested by test cases generated according to requirement specifications? (D8)	3.35	Neutral
17. Is the testing process carried out or performed by the development process team? (D9)	3.53	Agree

The results reveal that the majority of the important practices (12 practices) have “disagree” acceptance, two practices have neutral acceptance and the last three practices have the value agree acceptance.

4.7.4 Discussion of Findings

The results of the survey can be summarized as point:

- **Determine the SSF characteristics:** The majority of SSF in Jordan are private sector, and they have 10 to 30 employees followed by 31 to 50 employees, which consistent with the finding of (Fayad et al., 2000; Hofer, 2002; Laporte et al., 2005). Developers inside these firms are working with requirements, design, coding and testing activities. In addition, all developers have ten or less than ten years of experience and few managers and team leaders have more than ten years' experience. Moreover, it clearly obvious that the greater part of the respondents working in developing business information systems and e-business in general as an application domain. Therefore, the development method will be proposed for the SSF should be performed by a small number of developers and provide a training session to meet their lack of experience.
- **Determine the development issues:** a greater part of respondents still did not use any method for developing Web applications. Therefore, there a need of new methodology for developing Web application in SSF which is consistent with several studies such as Ahmad et al. (2005), Baskerville and Pries-Heje (2002), Costagliola et al. (2002) and Murugesan et al. (2001). Furthermore, the reasons for not using a specific method, a high percentage of respondents answered that using particular method need a specific team to be performed and assume that using specific method need training.

Regarding to the development method that the respondents are familiar with, most of them are familiar with waterfall followed by XP and Scrum. The most of developers perform the testing process at the end of the coding phase of the development. The most component that reused often by the developers of SSF are: source code, templates and modules during the development process. The most common QA activities that had been performed by the SSF are: testing Web applications and code review and these activities currently performed by the project team. Therefore, there is a need for a development process that constructed based on XP and Scrum. This process should cover all the development stages and able to reuse the existing components. Furthermore, the role and responsibilities of the development team members should be clearly defined.

- **Determine the measurement issues:** The majority of respondents still don't use any measurements during the development process. Whereas, there is minimal percentage of respondents used line of code and use GQM as a measurement method after the coding phase. These results are consistent with the findings Kettelerij (2006) and McCurley et al. (2008). This means there is a lack of performing measurements types and methods during development process in SSF. The reasons for not using a specific measurement methods and metrics were because there is nobody in the company familiar with measurement process and using measurements need a trained team to be performed. In addition, respondents who like to perform measurement after

the coding phase, they often use XP then Scrum followed by Waterfall as development method and the most used metric in these three development methods is a line of code. Respondents who are using XP, Scrum, DSDM and Spiral respectively, they prefer to apply measurement by using the GQM method. Therefore, there is need of a goal oriented measurement mechanism based on the GQM method that covers the whole development process stages. This mechanism should use a quantitative and qualitative metrics in order to monitor the process and product. In addition, this mechanism should take into account the small software firm staff limitation.

- **Investigate the current Web application development and measurement practices:** The degree of applying the important Web applications development practices was low since three out of seventeen practice were applied in the SSF in Jordan as well as three are partially applied. The practices that are not performed in SSF are requirement, test, quality management and measurement practices. This means there is a lack of applying the development and measurement practices inside these companies which consistent with the findings of Bucci et al. (2001) and El Sheikh & Tarawneh (2007). Therefore, there is a need for development methodology that performs these important practices.

4.7.5 Summary

The findings of the survey demonstrate the current practices of Web application development and measurement in Jordanian SSF. A survey approach was adopted for this study using questionnaires as an instrument for collecting data. The sample comprised of seventy-five from Jordanian SSF. The respondents were mainly managers and developers.

This survey gives a better understanding of the current development and measurement practices that were performed by the Jordanian SSF. The issues of using the current development and measurement methods were also highlighted. The findings of the survey will be used for constructing a new Monitoring Oriented Agile Based Web Applications Development Methodology for SSF



UUM
Universiti Utara Malaysia

CHAPTER FIVE

METHODOLOGY CONSTRUCTION

5.1 Introduction

The main outcome of this chapter is a new Monitoring Oriented Agile Based Web Applications Development Methodology (MOGWD) for Small Software Firms. As mentioned in Chapter Three, this methodology was constructed based on four steps; extending the Scrum method by adding the important XP elements, enhancing the design phase by incorporating a Web design method, constructing a monitoring mechanism and organizing the MOGWD methodology components by adopting the PDCA method. The chapter starts by describing the Extended Agile method, the required improvements for the Extended Agile method and presenting the details of the methodology phases and components.

5.2 The Extended Agile Method

The methodology construction begins by analyzing the XP and Scrum methods before extending the Scrum method. This analysis was conducted in Chapter Two based on the specified criteria, namely the development process, project management, requirements, testing, design and the team structure.

The results from the analysis found that the Scrum method is suitable to be used as the basis for proposing the Agile Extended method because the Scrum is an iterative development method that performed management practices, which are strongly

recommended to manage the development processes. Furthermore, it concentrates on smaller development team. Therefore, this study adapted three phases of the Scrum which are planning, development and integration. Each phase has a set of activities and practices to be performed. However, the Scrum method is still lacking on the development practices. Therefore, the study had improved the development phase of the Scrum by analyzing the XP method.

The results from the XP method analysis found that some activities and practices in the XP should be integrated to improve the Scrum development phase. The elements that have been taken from the XP are the XP iteration activities, XP core and supported practices and XP iteration team (programmer and tester). The result of this combination is the Extended Agile method for SSF.

Table 5.1 shows the elements of the extended Agile method which include process phases, activities and practices.

Table 5.1

The elements of the Extended Agile method

Process phase	Activities	Practice	Taken from	
			XP	Scrum
Planning	- Identify the product backlog items	First planning meeting		√
	- Prioritize the items - Split the large items if any, to smaller items. - Estimate the items	Iteration planning meeting		√
	Development	Analysis	√	
	Design	Simple design	√	

	Code	Coding standards, pair programming, Refactoring, metaphor and collective ownership	√	
	Test	TDD	√	
	Daily reviewing	Daily meeting		√
	Iteration reviewing	Iteration review meeting		√
Integration	Integrate increment with the system	Continuous integration	√	
	Final release	Small release	√	

As shown in Table 5.1, the planning phase has four activities which are identifying the product backlog items, prioritizing the items, splitting the large items (if any) to smaller items and estimating the items. The activity for identifying the product backlog items should be performed by deploying the first planning meeting practice. However, the last three activities of planning should be performed in the iteration planning meeting practice. These activities and practices were taken from the Scrum.

The development phase will be performed through several activities. The first four activities and their practices were taken from the XP as it concentrates on the development more than that of the Scrum. These activities are analysis, design, code and test. The last two development activities; the daily reviewing and iteration reviewing were adopted from the Scrum.

The integration phase involved two activities which are integrating the new increment with the system and final release. These activities were adopted from the Scrum. However, the increment in the Scrum required at least one month to be

integrated into the system and at the moment there are no specific practices in the Scrum to perform such integration. Therefore, this study improved the practices by adopting a continuous integration and small release practices from the XP which are more suitable for reducing the cycle time and risk of failure.

However, there are still some issues that failed to be covered in the proposed Extended Agile method. The two issues are (1) the existing design phase in the iteration is simply performed and focus more on coding and (2) the method does not have any measurement mechanism that can monitor the quality of the process and product. To counter these issues, two solutions have been proposed in this study. The solutions that can be proposed are to enhance the design phase by adding a Web design method and to construct a measurement mechanism by using the Goal Oriented Monitoring Mechanism (GOMM) that emphasize on monitoring the quality of the process and product. Figure 5.1 shows the improved Extended Agile Method.

Figure 5.1 indicates that the design activity in the Extended Agile method has been improved by adopting the activities and practices from the existing Web design method. In addition, the method was also referred to improve the first planning meeting practice. Moreover, the study has proposed a set of qualitative and quantitative metrics as a mechanism for monitoring the process and product quality. The metrics were derived by using the GOMM that refers to the lightweight GQM.

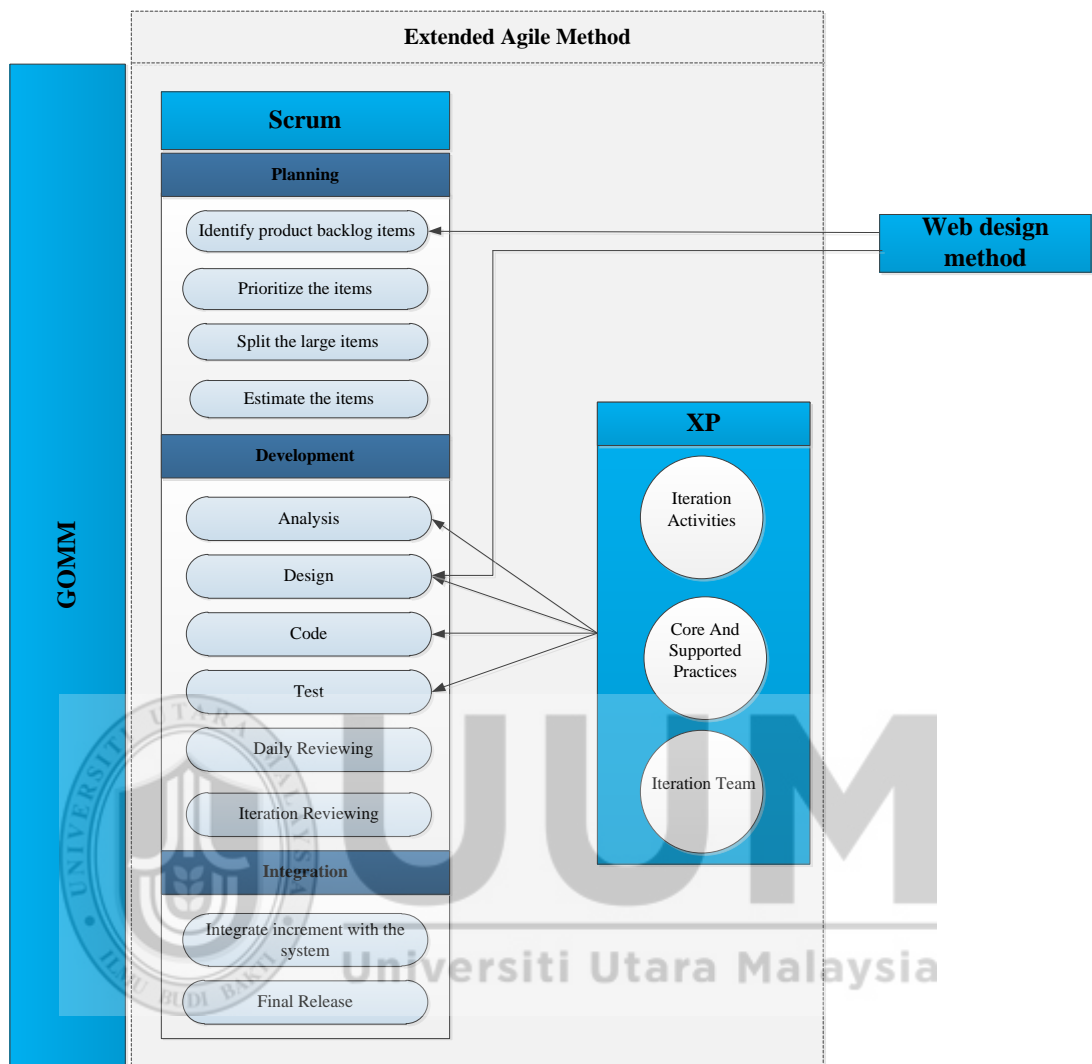


Figure 5.1. The improvements of Extended Agile Method

All improvements made for the proposed Extended Agile Method were meant to construct the MOGWD methodology.

5.3 The overview of MOGWD methodology

Findings from the literature review (as discussed in Chapter Two) and the survey (as discussed in Chapter Four) have contributed to the construction the MOGWD

methodology. The main findings from the literature review are: the XP and Scrum are the suitable development methods to be used for SSF, the light weight GQM is the appropriate measurement method to be used for SSF, the common steps of Web application design, the criteria of good methodology and the list of best practices that should be performed by SSF.

Meanwhile, the findings from the survey indicate that even though they are familiar with the XP and Scrum methods, the majority of the practitioners in SSF are still using ad-hoc approach for developing the Web application. The findings also show that the practitioners in SSF are still lacking in the awareness on monitoring the quality of the process and product. Therefore, these outputs clarify the need of a new methodology that emphasizes on monitoring the quality of the Web applications product and development process. Hence, this study proposed the MOGWD methodology that focuses on producing a high quality Web application for SSF. The main characteristic of the MOGWD methodology is an iterative Agile development methodology that emphasizes on continuous quality monitoring for the process and product.

This methodology concentrates on the management, development and monitoring processes. The management and development processes were taken from the Extended Agile method that has been improved with the Web design method, while the monitoring process was constructed by performing the GOMM.

The Plan, Do, Check and Act (PDCA) method was adapted in this study to organize the components of the MOGWD methodology. According to Quaglia and Tocantins (2011), the process of the PDCA can be performed under the Agile perspective particularly in the Scrum. In addition, the development and the measurement processes can be applied together based on the PDCA phases. The MOGWD methodology has defined four phases adapted from the PDCA, namely the Plan, Do, Check and Act. Figure 5.2 shows the four phases of the MOGWD methodology. Each phase has clearly defined the aim and activities.



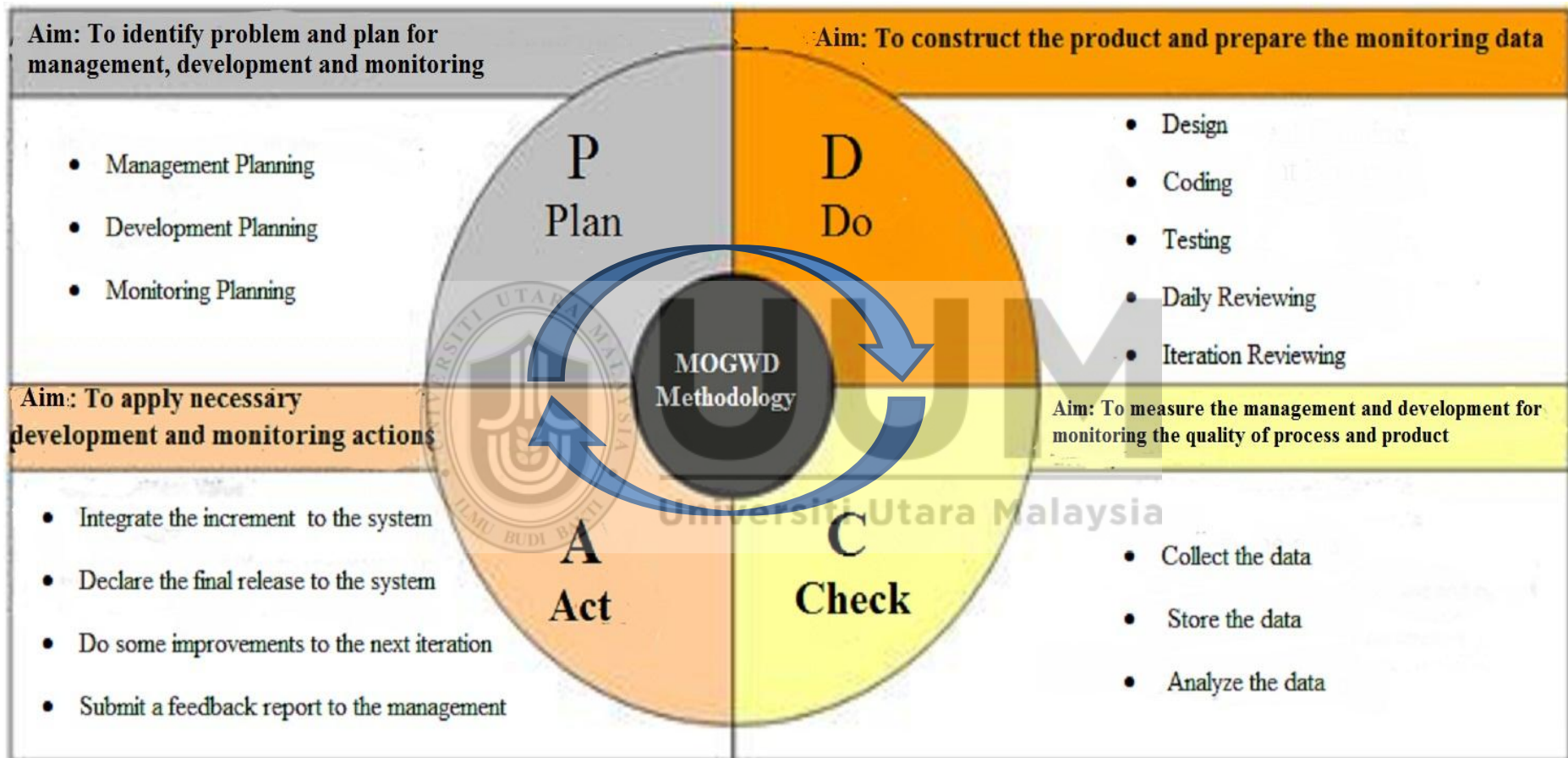


Figure 5.2. The MOGWD methodology

5.4 MOGWD Methodology

As mentioned earlier, the methodology has four phases, Plan, Do, Check and Act. Each phase provides well-defined components. These components are activities, methods, practices, tools and team structure. Additionally, the activities performed are based on particular methods, specific practices, and set of tools. The activities should be carried out by team member(s). The next sections discuss in details about the components of each phase.

5.4.1 Plan Phase

This phase aims to identify the problem and plan for the management, development and monitoring activities. Each activity has a set of sub activities. The next section provides a detailed explanation of these activities.

5.4.1.1 Management Planning

The management planning involved several sub activities such as staffing, training and controlling. The first two sub activities of the management planning, staffing and training, will be performed in the plan phase. However, the controlling sub activity will be performed during the whole process. In this activity, the top management will identify the master and product owner (PO). The master takes all the responsibilities of managing the project. The master will discuss with the PO in order to understand the problem to be solved. He also has to produce plan for performing the management activities. This plan includes time frame, budget and brief explanation of the management activities. Table 5.2 described the management planning sub activities. Each sub activity includes several actions.

Table 5.2

Management Planning Sub Activities

Sub activity	Actions	Team member involved
Staffing	- Assigning roles and responsibilities for each team member	Master
Training	- Identifying phases, activities, methods, practices, and tools of the MOGWD methodology - Identifying the roles and responsibilities of each team member that involved in certain activities.	The whole development team
Controlling	- Keeping the process Agile and accelerating the process.	Master

Each sub activity is described as follows:

Staffing: is an activity for identifying the team members to be involved in the project and defining the roles and responsibilities for each member. The team structure of the MOGWD is described in Table 5.3.

Table 5.3

The MOGWD Methodology Team Structure

Role	Responsibility	Stakeholder
Master	Acts as the leadership role to ensure that the practices, rules and values process are followed according to the planned project execution. In addition, the Master should be aware with the XP practices and software measurement.	Development team (DT)
Product owner (PO)	One of the team selected by the management and master. He is responsible for managing, controlling and making the product backlog visible. He is also responsible for writing the stories and functional tests, setting requirement priority and deciding when each requirement satisfied. He should be aware with the XP practices and software measurements.	

Customer	Tasks related to determine the product backlog.	
Programmer	Writing tests and keeping the code simple. In addition, he must be aware with the XP practices such as pair programming, coding standard, and software measurement.	
Tester	Help the customer to write functional test, run functional test, broadcast test results and maintain testing tools. He must also be aware with the XP practices such as TDD and software measurement.	
GOMM team member	One member is responsible for assuring the product quality and conducting the required measurements. Gives feedback on how accurate the effort estimations which made by the team are, progress tracking, evaluate whether the goal achieved within time and budget and determine if any changes needed in the process. In addition, they should be aware with the QA practices and software measurement. The other GOMM members are responsible for analyzing data and preparing the feedback report to the management	Monitoring team (MT)
Management	Decision making, communicate with the team, setting goals and requirements and select the master and product owner.	Top management

Based on Table 5.3, the minimum number of members who should be involved in performing the MOGWD methodology is seven. These members will play the roles as master, PO, two programmers, tester, and two GOMM members.

The roles and responsibilities of the MOGWD methodology are classified into three categories of stakeholders: development team (DT), monitoring team (MT) and top management.

Training: the whole team members should attend a simple training session that takes around two to seven days to understand the MOGWD methodology as well as its functions. The training session will be conducted by the master with all the team

members. In this session, all roles and responsibilities will be explained, process activities will be clarified and practices will be clearly discussed. After completing this training, each team member should know his/her roles and responsibilities during the process. Furthermore, each team member should know the activity that he/she will play. Lastly, each team member should know how to perform the assigned practice.

Controlling: is one of the master responsibilities to ensure that the process remains agile, deploys Agile practices and can be accelerated by removing impediments that makes the process slow. A plan produced by the master clearly defines the Agile practices, the activity to be performed and the person who will be performing.

5.4.1.2 Development planning

The development planning includes five sub activities, namely creating the product backlog, performing the Web design method, selecting the items that will be entered to the next Do (iteration), splitting the large items (if any) to smaller and estimating the items. The first two sub activities are performed in order to plan for the whole product, whereas the last three sub activities are performed to plan for the next Do (iteration). Table 5.4 shows the sub activities, methods used, practices, tools, team members and the outcome of each action.

Table 5.4

Development Planning

Sub activities	Methods	Practices	Tools	The team member involved	Outcomes
<ul style="list-style-type: none"> - Create the product backlog - Perform the Web design method 	<ul style="list-style-type: none"> - Extended Agile method. - Web design method. 	First planning meeting	User stories, requirement repository and ArgoUWE	Master, PO, development team (DT) and monitoring team (MT).	<ul style="list-style-type: none"> - Product backlog - Web design prototype.
<ul style="list-style-type: none"> - Select the items that will be entered for the next Do (iteration). - Split the large items (if any) to smaller items. - Estimate the items. 	<ul style="list-style-type: none"> - Extended Agile method. 	Do (Iteration) planning meeting	Previous report for estimating and prioritizing the product backlog items	PO, DT and MT	<ul style="list-style-type: none"> - Do backlog - The estimated time, cost, line of code and others.

Each sub activity of the development planning is discussed in details as follows:

Create the Product backlog. The product backlog is an ordered list of requirements that is maintained for a product. It consists of features, bug fixes, non-functional requirements and whatever needs to be done in order to successfully deliver a viable product. This sub activity was performed using the Extend Agile Method that emphasizes on deploying the specific practice known as the first planning meeting. In this meeting, the PO will order the product backlog items for the development team (DT) to choose based on risks, business values, dependencies, date needed, and others. The meeting will be held by all the team members' master, product owner (PO), DT and monitoring team (MT). The Product

backlog items should be collected directly from the customer or the product owner using the user stories tool.

The user requirement specifications should to be saved in a simple data repository at the first planning meeting to help customers and developers to trace the customer's requirements status and reuse the old requirements. The output of this sub activity is the order list of product backlog items.

Performing the Web Design Method: This sub activity will be performed using the Web design method. This method will be performed during the first planning meeting to create a simple design prototype that may require the ArgoUWE tool to support the action method. The Web design method is performed by the development team and the PO (customer). Five actions are required to perform the Web design method which is requirements analysis, conceptual design, navigational design, implementation design (interface) and construction as shown in Figure 5.3.

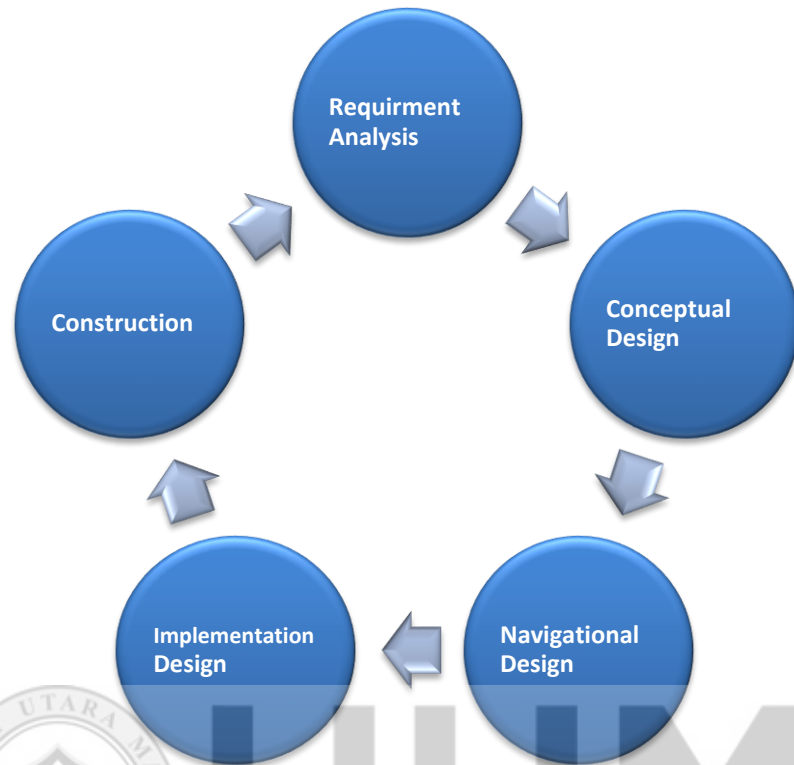


Figure 5.3. Web Design method

- Requirements analysis: this action aims to take the requirements that have been identified in the product backlog. The backlog includes items such as Web application objectives, targeted audiences, content, style guidelines, and development constraints. Others include requirements analysis, requirements checking for necessity (the need for the requirement), consistency (requirements should not be conflicting) and completeness (no service or constraint is missing). Requirement necessity and completeness will be ensured in this action. However, requirement consistency is determined

through requirement prioritizing which will be carried out in the Do iteration planning.

- Conceptual design (object design): determine the objects, classes, subclasses, relationships, attributes and perspectives of the Web application using any object oriented constructs (classes, relationships or use cases). During the object modeling sub phase, the requirements of the different user classes and their perspectives are formally described. The object models do not only describe the object types and relationships, but also the rules or constraints on the object types and relationships. The object oriented models also describe the behavior of the objects.
- Navigational design: this action describes how the user can navigate through the Web application as well as specifies how pages and content units linked to the whole application. This will be done by determining the nodes, links, access structure and navigational structure. In addition, the navigation design describes how the different users can navigate through the Web application. The navigation, design consists of a number of navigation tracks. A navigation track expresses how users can navigate through the available information. This is described in terms of components, links and flow charts.
- Implementation design (interface): the aim of this action is to design the look and feel as described in the conceptual design phase by generating page structure, page flow, user interface and logical database schema required by the design Web application.

- **Construction:** The product owner prioritizes requirements related to the Web design prototype, and sends it to the programmer to start developing it into a real system. The ArgoUWE will be an effective tool to support the creation of the design method.

After completing the planning for the whole product in the first planning meeting, three sub activities will be performed for planning the next Do (iteration): prioritize the backlog items to be entered for the next Do, split the large item into smaller item and estimates the items. These actions will be performed by using the extended Agile method during the Do (iteration) meeting.

Select the items that will be entered for the next DO (iteration). This sub activity aims to specify the selected product backlog items for the next Do (iteration). The selection will be performed by using Do (iteration) meeting practice. The product owner is responsible for prioritizing and ordering the items. The prioritizing sub activity is carried out based on the previous reports for estimating and prioritizing items. The outcome of this sub activity is the Do backlog.

Split the large items if any, to smaller items and estimates the items. The two sub activities will be performed during the Do (iteration) planning meeting. In this two sub activities, the DT used their experience and reports from previous project to select the large item and split it into smaller task to perform it in the next Do (iteration). By the end of this meeting, the DT is ready to do his/her job for the Do

(iteration) and move to the Do (development phase). Among the outcomes of the two sub activities are the split tasks, estimated time, cost, and line of code.

5.4.1.3 Monitoring planning

For the monitoring planning activity, three sub activities are involved, which include defining the monitoring goals, determining questions and metrics, producing monitoring plan and prioritizing monitoring goals as shown in Table 5.5. The first two sub activities will be performed by the extended Agile using the first planning meeting practice method and GOMM. The last sub activity will be performed in the Do (iteration) meeting.

Table 5.5

Monitoring Planning

Sub activities	Methods	Practices	Tools	The team member involved	Outcomes
<ul style="list-style-type: none"> - Define the monitoring goals, questions and metrics. - Produce the monitoring plan that includes data collection procedure and data collection instrument. 	Extended Agile method and GOMM	First planning meeting	-	Master, PO, DT and Monitoring team (MT).	<ul style="list-style-type: none"> - Monitoring goals, questions and metrics - Monitoring plan.
<ul style="list-style-type: none"> - Prioritize the monitoring goals. 	Extended Agile method and GOMM	Do (Iteration) planning meeting and prioritize goals practice	Previous report for estimating and prioritizing the product backlog items	Master and MT	<ul style="list-style-type: none"> - The prioritized monitoring goals.

Each sub activity of the monitoring planning is discussed in details as follows:


Define the goals, questions and metrics: this sub activity starts by defining the goals followed by deriving questions and metrics.

- **Define the goals:**

Two types of goals were defined based on the goal template mentioned in Chapter Two. The types of goals are quantitative and qualitative as shown in Table 5.6. The goal definition should describe the purpose of measurement, the object that the measurement is focusing on and the people that measure the object.

Table 5.6

Goal Definition



	Goal	Definition	Type	Object
Development Process Quality	G1.1	To analyze the requirement activity for the purpose of monitoring with respect to a number of requirements completed from the viewpoints of the GOMM members	Quantitative	Requirement
	G 1.2	To analyze the design activity for the purpose of monitoring with respect to the number of SLOC, a number of Web pages and total number of links from the viewpoints of the GOMM members	Quantitative	Design
	G 1.3	To analyze the testing for the purpose of monitoring with respect to the current size of the test status from the viewpoints of the GOMM members	Quantitative	Testing
	PG1	To analyze the common Scrum practices (core) for the purpose of monitoring with respect to the Scrum meetings from the viewpoints of the GOMM members	Quantitative	Management practices

PG2	To analyze the common XP practices (core) for the purpose of monitoring with respect to the pair programming, TDD, refactoring, coding standards, from the viewpoints of the GOMM members	Quantitative	Development practices (core)
PG3	To analyze the supported XP practices for the purpose of monitoring with respect to small release, continuous integration, simple design, metaphor and collective ownership from the viewpoints of the GOMM members	Quantitative	Development practices (supported)
G2	To analyze the productivity tracking for the purpose of monitoring with respect to the value of staff productivity from the viewpoints of the GOMM members	Quantitative	Staff Productivity
QG1	To analyze the process completeness for the purpose of monitoring with respect to the process activities requirements, design, coding, testing and project management from the viewpoints of the GOMM members through a questionnaire	Qualitative	Completeness
QG2	To analyze the process consistency for the purpose of monitoring with respect to the process activities requirements, design, coding, testing and project management from the viewpoints of the GOMM members through questionnaire	Qualitative	Consistency
QG3	To analyze the process accuracy for the purpose of monitoring with respect to the process activities requirements, design, coding, testing and project management from the viewpoints of the GOMM members through a questionnaire	Qualitative	Accuracy
QG4	To analyze the process of tailorability for the purpose of monitoring with respect to the tailorability practices from the viewpoints of the GOMM members through questionnaire	Qualitative	Tailorability
QG5	To analyze the process flexibility for the purpose of monitoring with respect to the flexibility practices from the viewpoints of the GOMM members through a questionnaire	Qualitative	Flexibility
QG6	To analyze the process of compatibility for the purpose of monitoring with respect to the compatibility practices	Qualitative	Compatibility

		from the viewpoints of the GOMM members through a questionnaire		
	QG7	To analyze the process of accessibility for the purpose of monitoring with respect to the accessibility practices from the viewpoints of the GOMM members through a questionnaire	Qualitative	Accessibility
	QG8	To analyze the process of applicability for the purpose of monitoring with respect to the applicability practices from the viewpoints of the GOMM members through a questionnaire	Qualitative	Applicability
	QG9	To analyze the process of changeability for the purpose of monitoring with respect to the changeability practices from the viewpoints of the GOMM members through a questionnaire	Qualitative	Changeability
	QG10	To analyze the process supportability for the purpose of monitoring with respect to the supportability practices from the viewpoints of the GOMM members through a questionnaire	Qualitative	Supportability
Web application Product Quality	G3	To analyze the development process cost for the purpose of monitoring and controlling with respect to the cost of fix, cost of activity and project budget from the viewpoints of the GOMM members	Quantitative	Cost
	G4	To analyze the quality aspects for the purpose of monitoring with respect to the security, product reliability, usability and maintainability from the viewpoints of the GOMM members	Quantitative	Quality
	G5	To analyze the development life cycle time for the purpose of monitoring with respect to the reuse artifacts, time for each iteration, project velocity from the viewpoints of the GOMM members	Quantitative	Time

Based on Table 5.6, the goals were defined to monitor the quality of the development process and the Web application product. The quality of the process will be monitored quantitatively and qualitatively. However, the Web application product quality will be monitored quantitatively. Seven quantitative goals were

defined to monitor the process quality; three goals described the process quality activities such as requirement, design and test. Another three goals described the application of the Agile practices during the development such as the core management practices, core development practices and supported development practices. One goal described the staff productivity. In addition, ten qualitative goals were defined to monitor the process quality factors such as completeness, consistency, accuracy, tailorability, flexibility, supportability, accessibility, applicability, changeability and compatibility. On the other hand, three quantitative goals were defined to monitor the quality of the Web application product namely the cost, quality and time.

- **Derive the questions and metrics:** after defining the goals, the questions and metrics were derived. Each goal may have set of questions which were answered by performing a set of metrics. The multilevel list numbering was used for the question and metrics that started from the goal number. For example, G1.1 has question Q1.1.1 and metric M1.1.1.1. This activity derives the question and metrics for the defined goals.

Requirements questions and metrics: A question, Q1.1.1, related to the requirement status was answered by performing the M 1.1.1 as shown in Table 5.7.

Table 5.7

Requirements Questions and Metrics

G1.1: To analyze the requirement status for the purpose of monitoring with respect to the number of requirements completed from the viewpoints of the GOMM members	
Question	Metrics
Q1.1.1: What is the current size of the requirements status?	M1.1.1.1: Number of product backlog items completed to date/total number of requirements planned.

Design questions and metrics. A question, Q1.2.1, related to the design status that was answered by performing three metrics, M1.2.1.1, M1.2.1.2 and M1.2.1.3 as shown in Table 5.8.

Table 5.8

Designed Questions and Metrics

G1.2: To analyse the requirement status for the purpose of monitoring with respect to the number of requirements completed from the viewpoints of the GOMM members	
Question	Metrics
Q1.2.1: What is the current size of the design status?	M1.2.1.1: Number of LOC completed to date / Total Number of SLOC planned.
	M1.2.1.2: Number of Web Pages to date / Total Number of Web Page planned.
	M1.2.1.3: Total Number of internal links / Number of Web pages.

Testing Question and Metrics: Question Q1.3.1 related to the test status that was answered by performing the M1.3.1.1 and M1.3.2.1. Table 5.9 shows the testing questions and metrics.

Table 5.9

Testing Questions and Metrics

G 1.3: Analyze the test status for the purpose of monitoring with respect to the number of tests completed to date and total number of tests planned from the viewpoints of the GOMM members	
Question	Metrics
Q1.3.1: What is the current size of the test status?	M1.3.1.1: The number of tests completed to date / Total Number of tests planned.
	M1.3.1.2 number of testing line of code vs. total number lines of code

Management practice questions and metrics: Three questions were identified to monitor the application of the management practices adopted from the Scrum; PQ1.1, PQ1.2 and PQ1.3. Question PQ1.1 related to the application of the iteration planning meeting that can be measured by performing the metric PM1.1.1. Question PQ1.2 related to the application of the daily meeting that can be measured by performing the metric PM1.2.1. Question PQ1.3 related to the application of the iteration review meeting that can be measured by performing the metric PM1.3.1 as shown in Table 5.10.

Core development practices questions and metrics. Three questions were defined to monitor the application of the core development practices adopted from the XP. These questions are PQ2.1, PQ2.2 and PQ2.3. The questions can be measured by performing the next four metrics PM2.1.1, PM2.1.2, PM2.2.1 and PM2.3.1 respectively as shown in Table 5.10.

Supported development practices questions and metrics. Five questions were defined to monitor the application of the supported development practices adopted from the XP. These questions are PQ3.1, PQ3.2, PQ3.3, PQ3.4 and PQ3.5. The questions can be measured by performing the next five metrics PM3.1.1, PM 3.2.1, PM3.3.1, PM3.4.1 and PM3.5.1 respectively as shown in Table 5.10.

Table 5.10

Practices Questions and Metrics

PG1: To analyze the common Scrum practices (core) for the purpose of monitoring with respect to the Scrum meetings from the viewpoints of the GOMM members	
Questions	Metrics
PQ1.1: How to measure the iteration planning meeting?	PM1.1.1: Number of iteration planning meetings per one application.
PQ1.2: How to measure the daily meeting?	PM1.2.1: Number of daily meetings per one application?
PQ1.3: How to measure the iteration review meeting?	PM1.3.1: Number of review meetings done per one application?
PG2: To analyze the common XP practices (core) for the purpose of monitoring with respect to the pair programming, TDD, refactoring, coding standards from the viewpoints of the GOMM members	
Questions	Metrics
PQ2.1: How to monitor the TDD practice?	PM2.1.1: Number of testing line of code /total number lines of code.
	PM2.1.2: Number of tests completed to date vs. Total Number of tests planned.
PQ2.2: Does the duplicated code removed to decrease ambiguity and redundancy, and improve communication and adding flexibility?	PM2.2.1: Number of lines of duplicated code removed / total line of code per iteration.
PQ2.3: Does the development team follow a coding standard?	PM2.3.1: Adherence of coding standard (High, Low).
PG3: To analyze the supported XP practices for the purpose of monitoring with respect to the small release, continuous integration, simple design, metaphor and collective ownership from the viewpoints of the GOMM members	
Questions	Metrics
PQ3.1: Is every iteration release with small size of code?	PM3.1.1: (Number of LOC of the first release - the LOC of the next release) / total NLOC

PQ3.2: Does the new created release reflecting all the changes?	PM 3.2.1: (Total number of lines of code added, removed and updated) / total line of code for the previous iteration.
PQ3.3: Is the architecture and the code (including the unit tests) as simple as possible?	PM3.3.1: (Number of LOC of the current release - total LOC) / Total LOC
PQ3.4: Does the system created by setting of the metaphors between the client and programmers?	PM3.4.1: Number of meetings between development team and the client?
PQ3.5: Do all team members are owners of the code (can make changes on the code)?	PM3.5.1 Number of team members who made changes in the code.

Productivity questions and metrics: A question, Q2.1, was identified to monitor the staff productivity using one metric, M2.1.1, as shown in Table 5.11.

Table 5.11

Productivity Questions and Metrics

G2: To analyze the productivity tracking for the purpose of monitoring with respect to the value of staff productivity from the viewpoints of the GOMM members	
Questions	Metrics
Q2.1: What is the value of the productivity of the project staff?	M2.1.1: Number of KLOC for staff in month.

Completeness questions and metrics. Five questions were defined to monitor the process completeness. These questions are QQ1.1, QQ1.2, QQ1.3, QQ1.4 and QQ1.5. Question QQ1.1 related to the requirement completeness that can be measured by performing six metrics QM1.1.1, QM1.1.2, QM1.1.3, QM1.1.4, QM1.1.5 and QM1.1.6. Question QQ1.2 related to the design completeness that can be measured by performing seven metrics QM1.2.1, QM1.2.2, QM1.2.3, QM1.2.4, QM1.2.5, QM1.2.6 and QM1.2.7. Question QQ1.3 related to the code completeness

that can be measured by performing eight metrics QM1.3.1, QM1.3.2, QM1.3.3, QM1.3.4, QM1.3.5, QM1.3.6, QM1.3.7 and QM1.3.8. Question QQ1.4 related to the testing completeness that can be measured by performing ten metrics QM1.4.1, QM1.4.2, QM1.4.3, QM1.4.4, QM1.4.5, QM1.4.6, QM1.4.7, QM1.4.8, QM1.4.9 and QM1.4.10. Question QQ1.5 related to the project management completeness that can be measured by performing five metrics QM1.5.1, QM1.5.2, QM1.5.3, QM1.5.4 and QM1.5.5 as shown in Table 5.12.

Table 5.12

Completeness Questions and Metrics

QG1: To analyze the process completeness for the purpose of monitoring with respect to the process activities requirements, design, coding, testing and project management from the viewpoints of the GOMM members through questionnaire	
Questions	Metrics
QQ1.1: What is the degree of requirement completeness?	Q.M1.1.1: Customers or P.O was available on-site for face-to-face discussions during the requirement elicitation
	Q.M1.1.2: The scope of project was identified at the beginning of a project to create initial prioritized product backlog items
	Q.M1.1.3: The requirements were validated by customers in review meetings by using prototype/release
	Q.M1.1.4: Requirements were prioritized and can be reprioritized by customers throughout the development
	Q.M1.1.5: The development team was enabled to re-estimate the time and velocity of user stories
	Q.M1.1.6: The requirements were written on cards in a short statement
QQ1.2: What is the degree of design completeness?	Q.M1.2.1: Model storming was performed (architecture, interface, data structure and algorithm)
	Q.M1.2.2: The architecture designs were produced
	Q.M1.2.3: The interface designs were produced
	Q.M1.2.4: The data structure was produced
	Q.M1.2.5: The algorithms were produced
	Q.M1.2.6: Iteration modeling was performed at the beginning of each iteration
	Q.M1.2.7: The designs were documented

QQ1.3: What is the degree of coding completeness?	Q.M1.3.1: Reuse of software components was encouraged
	Q.M1.3.2: Detailed explanations of the functions and variables were included in the code
	Q.M1.3.3: The code was produced and integrated to system baseline iteratively and incrementally
	Q.M1.3.4: Web application was delivered frequently with increments of features
	Q.M1.3.5: Customer involved with the team for giving immediate feedbacks
	Q.M1.3.6: The features with high priority were delivered first
	Q.M1.3.7: Web application was deployed gradually in real environment
	Q.M1.3.8: The deliverable documentation was produced late
QQ1.4: What is the degree of testing completeness?	Q.M1.4.1: Tests were automated
	Q.M1.4.2: Tests were performed continuously throughout the development
	Q.M1.4.3: Frequent integration tests were performed
	Q.M1.4.4: Unit tests were performed to ensure that all requirements were fulfilled
	Q.M1.4.5: User interfaces were tested
	Q.M1.4.6: Database regression testing was performed
	Q.M1.4.7: Customer (P.O) wrote the user acceptance tests according to stories/features
	Q.M1.4.8: Acceptance tests were used to validate and verify user's requirements
	Q.M1.4.9: Results of the tests were documented
	Q.M1.4.10: Results from the automated tests were compared to the manual tests
QQ1.5: What is the degree of project management completeness?	Q.M1.5.1: The project was started with a clear scope, goals and objectives
	Q.M1.5.2: Planning for the project was performed collaboratively with team members
	Q.M1.5.3: The current progress of iteration was revealed to everyone on iteration burn down chart
	Q.M1.5.4: Customer and end-user involvement were monitored in project activity
	Q.M1.5.5: The project plan was documented

Consistency question and metrics. The consistency is one of the effectiveness sub factors that need to be monitored and measured for the whole process activities. It is defined as applying the Agile standard and principles during the development

process to ensure the agility of the process. Consequently, as mentioned in Chapter Two, applying the Agile principles required to follow the important Agile development and management practices. Therefore, the consistency factor concentrates on asking about the Agile development and measurement practices. Table 5.13 describes the questions and metrics of process consistency. Five questions were defined to monitor the process consistency. These questions are QQ2.1, QQ2.2, QQ2.3, QQ2.4 and QQ2.5. Question QQ2.1 related to the requirement consistency that can be measured by performing two metrics QM2.1.1 and QM2.1.2. Question QQ2.2 related to the design consistency that can be measured by performing four metrics QM2.2.1, QM2.2.2, QM2.2.3 and QM2.2.4. Question QQ2.3 related to the code consistency that can be measured by performing ten metrics QM2.3.1, QM2.3.2, QM2.3.3, QM2.3.4, QM2.3.5, QM2.3.6, QM2.3.7, QM2.3.8, QM2.3.9, QM2.3.10. Question QQ2.4 related to the testing consistency that can be measured by performing six metrics QM2.4.1, QM2.4.2, QM2.4.3, QM2.4.4, QM2.4.5 and QM2.4.6. Question QQ2.5 related to the project management consistency that can be measured by performing seven metrics QM2.5.1, QM2.5.2, QM2.5.3, QM2.5.4, QM2.5.5, QM2.5.6 and QM2.5.7 as shown in Table 5.13.

Table 5.13

Consistency Questions and Metrics

QG2: To analyze the process consistency for the purpose of monitoring with respect to the process activities requirements, design, coding, testing and project management from the viewpoints of the GOMM members through questionnaire.

Questions	Metrics
QQ2.1: What is the degree of requirement consistent?	Q.M2.1.1: Appropriate procedure is used to handle frequently changing requirements
	Q.M2.1.2: The requirements were documented by following a particular standard
QQ2.2: What is the degree of design consistency?	Q.M2.2.1: Appropriate procedure was used to handle frequently changing designs
	Q.M2.2.2: The design was documented by following a particular standard
	Q.M2.2.3: Web application designs were refactored frequently
	Q.M2.2.4: Metaphor was used for determining the architecture of the system
QQ2.3: What is the degree of coding consistency?	Q.M2.3.1: Appropriate procedure was used to ensure that the code was developed based on the requirements and design
	Q.M2.3.2: Appropriate procedure was used to handle frequently changing code
	Q.M2.3.3: Appropriate procedure was used to deliver the Web application releases to customers
	Q.M2.3.4: Appropriate code integration strategy was followed
	Q.M2.3.5: Appropriate coding/ interface/ database standards were followed
	Q.M2.3.6: Team members had authority to make changes in any part of the code
	Q.M2.3.7: Pair programming was performed
	Q.M2.3.8: Failing unit tests were developed before the code was written (TDD)
	Q.M2.3.9: Rigorous code and database refactoring were implemented
	Q.M2.3.10: Code integration strategy was established and revised
QQ2.4: What is the degree of testing consistency?	Q.M2.4.1: The testing results were documented by following a particular standard
	Q.M2.4.2: Appropriate procedure was followed for implementing automated tests
	Q.M2.4.3: Appropriate procedure was followed for implementing integration tests
	Q.M2.4.4: Appropriate procedure was followed for implementing

	interface tests
	Q.M2.4.5: Appropriate procedure was followed for implementing user acceptance tests
	Q.M2.4.6: Appropriate procedure was followed for implementing database regression tests
QQ2.5: What is the degree of project management consistency?	Q.M 2.5.1: Appropriate procedure was used to plan the project (estimation and work breakdown)
	Q.M 2.5.2: The project plan was documented by following a particular standard
	Q.M 2.5.3: Release meetings were conducted at the beginning of the project and each release to create release plan
	Q.M 2.5.4: Iteration meetings were conducted at the beginning of each iteration to plan the iteration
	Q.M 2.5.5: Daily stand-up meetings were conducted for daily plan
	Q.M 2.5.6: Continuous review meetings were conducted at the end of each iteration to demonstrate the latest version of Web application
	Q.M 2.5.7: Retrospectives were conducted at the end of each iteration

Accuracy questions and metrics. Process accuracy is one of the effectiveness sub factor that need to be measured during the whole development activities (Baharom, 2008). Five questions were defined to monitor the process accuracy. These questions are QQ3.1, QQ3.2, QQ3.3, QQ3.4 and QQ3.5. Question QQ3.1 related to the requirement accuracy that can be measured by performing three metrics QM3.1.1, QM3.1.2 and QM3.1.3. Question QQ3.2 related to the design accuracy that can be measured by performing three metrics QM3.2.1, QM3.2.2 and QM3.2.3. Question QQ3.3 related to the code accuracy that can be measured by performing two metrics QM3.3.1 and QM3.3.2. Question QQ3.4 related to the testing accuracy that can be measured by performing two metrics QM3.4.1 and QM3.4.2. Question QQ3.5

related to the project management accuracy that can be measured by performing the QM3.5.1 metric as shown in Table 5.14.

Table 5.14

Accuracy Questions and Metrics

QG3: To analyze the process accuracy for the purpose of monitoring with respect to the process activities requirements, design, coding, testing and project management from the viewpoints of the GOMM members through a questionnaire

Questions	Metrics
QQ3.1: What is the degree of requirement accuracy?	Q.M3.1.1: Requirements were gathered using customer card
	Q.M3.1.2: Appropriate tools were used to facilitate requirements gathering activities
	Q.M3.1.3: A particular notation was used to represent the requirements
QQ3.2: What is the degree of design accuracy?	Q.M3.2.1: Web application was designed by following Web design method steps
	Q.M3.2.2: Appropriate tools were used to facilitate design activities
	Q.M3.2.3: A particular notation was used to represent the design
QQ3.3: What is the degree of coding accuracy?	Q.M3.3.1: Appropriate tools were used for bug tracking
	Q.M3.3.2: Appropriate programming language was used
QQ3.4: What is the degree of testing accuracy?	Q.M3.4.1: Appropriate tools were used to facilitate testing activities
	Q.M3.4.2: Appropriate techniques or methods were followed for the implemented tests
QQ3.5: What is the degree of project management accuracy?	Q.M3.5.1: Appropriate tools were used to facilitate the planning activities

Tailorability questions and metrics. One question was defined to monitor the tailorability process, QQ4.1 that can be measured by performing three metrics QM4.1.1, QM4.1.2 and QM4.1.3. Table 5.15 describes the questions and metrics that are required to measure the tailorability process.

Table 5.15

Tailorability Questions and Metrics

QG4: To analyze the process tailorability for the purpose of monitoring with respect to the tailorability practices from the viewpoints of the GOMM members through questionnaire

Questions	Metrics
QQ4.1: What is the degree of process tailorability?	Q.M4.1.1: Is the development of the Web application performed using the integration of the XP and Scrum?
	Q.M4.1.2: Is the using of the Web design method and measurement process performed without affecting the process performance?
	Q.M4.1.3: Is the integration of the Scrum, XP and GOMM easy to be performed in the organization?

Flexibility questions and metrics. One question was defined to monitor the process flexibility QQ5.1 that can be measured by performing two metrics QM5.1.1 and QM5.1.2. Table 5.16 describes the questions and metrics that are required to measure the process flexibility.

Table 5.16

Flexibility Questions and Metrics

QG5: To analyse the flexibility process for the purpose of monitoring with respect to the flexibility practices from the viewpoints of the GOMM members through a questionnaire

Questions	Metrics
QQ5.1: What is the degree of process flexibility?	Q.M5.1.1: Is any team member can vary the process performance for a specific need?
	Q.M5.1.2: Is this variation performed without requiring affecting the process itself?

Compatibility question and metrics. This factor is used when the organization used multiple processes to show the extent to which the interface and interaction between these processes is easy and clear. One question was defined to monitor the process compatibility QQ6.1 that can be measured by performing two metrics QM6.1.1 and QM6.1. Table 5.17 describes the questions and metrics that are required to measure the process compatibility.

Table 5.17

Compatibility Goal, Questions and Metrics

QG6: To analyze the process compatibility for the purpose of monitoring with respect to compatibility practices from the viewpoint of GOMM member through a questionnaire	
Questions	Metrics
QQ6.1: what is the degree of process compatibility?	Q.M6.1.1: Is the development of Web application performed by interacting with measurement and development process.
	Q.M6.1.2: Is this interact done easily and clear

Accessibility question and metrics: This factor is used to assess the ease of finding information about the product by the users. One question was defined to monitor the process accessibility QQ7.1 that can be measured by performing seven metrics QM7.1.1, QM7.1.2, QM7.1.3, QM7.1.4, QM7.1.5, QM7.1.6 and QM7.1.7. Table 5.18 describes the questions and metrics that are required to measure the process accessibility.

Table 5.18

Accessibility Questions and Metrics

QG7: To analyze the process accessibility for the purpose of monitoring with respect to the accessibility practices from the viewpoints of the GOMM members through a questionnaire	
Questions	Metrics
QQ7.1: What is the degree of process accessibility?	Q.M7.1.1: Is there a strategic established for training in the organization?
	Q.M7.1.2: Is determining of the training is the responsibility of the organization?
	Q.M7.1.3: Is there any training and tactical plan in the organization?
	Q.M7.1.4: Is there a record of the training organization?
	Q.M7.1.5: Is there any way to assess the training organization?
	Q.M7.1.6: Is the process practitioner able to access the training process electronically, not by hard copy?
	Q.M7.1.7: Is the process described graphically not textually?

Applicability Question and Metrics. Applicability describes the required activities to perform a piece of work. One question was defined to monitor the process applicability QQ8.1 that can be measured by performing four metrics QM8.1.1, QM8.1.2, QM8.1.3 and QM8.1.4. Table 5.19 describes the questions and metrics that are required to measure the process applicability.

Table 5.19

Applicability Question and Metrics

QG8: To analyze the process applicability for the purpose of monitoring with respect to the applicability practices from the viewpoints of the GOMM members through a questionnaire

Questions	Metrics
QQ8.1: What is the degree of process applicability?	Q.M8.1.1: Is there a defined process for each project from start up until the end?
	Q.M8.1.2: Is there a measurement mechanism used to estimate and plan the project activities?
	Q.M8.1.3: Is the project managed based on a specific plan?
	Q.M8.1.4: Is there a contribute product, measures, and experience for the future project

Changeability questions and metrics. This factor measures the extent of the process meeting the requirement change. One question was defined to monitor the process changeability QQ9.1 that can be measured by performing four metrics QM9.1.1, QM9.1.2, QM9.1.3 and QM9.1.4. Table 5.20 describes the questions and metrics that are required to measure the process changeability.

Table 5.20

Changeability Questions and Metrics

QG9: To analyze the process changeability for the purpose of monitoring with respect to the changeability practices from the viewpoints of the GOMM members through a questionnaire

Questions	Metrics
QQ9.1: What is the degree of process changeability?	Q.M9.1.1: is there a way to determine the change requirement sources and categories?
	Q.M9.1.2: Is there a strategy established for change requirement?
	Q.M9.1.3: Is there a way to evaluate, categorize, and prioritize these changes?
	Q.M9.1.4: Is the team going to develop and implement change management plans?

Supportability questions and metrics. This factor measures the extent of the easiness of support process in specific contexts. One question was defined to monitor the process supportability QQ10.1 that can be measured by performing four metrics QM10.1.1, QM10.1.2, QM10.1.3 and QM10.1.4. Table 5.21 describes the questions and metrics that are required to measure the process supportability.

Table 5.21

Supportability Questions and Metrics

QG10: To analyze the process supportability for the purpose of monitoring with respect to the supportability practices from the viewpoints of the GOMM members through questionnaire

Questions	Metrics
QQ10.1: What is the degree of process supportability?	Q.M10.1.1: Is there an agreement established and maintained between the supplier and the organization for supporting any item?
	Q.M10.1.2: Is the selection of the suppliers based on their ability of satisfying a specific requirement?
	Q.M10.1.3: Is the acquired product from the supplier evaluated from the organization before accepting it?
	Q.M10.1.4: Is the organization ensures that the agreement satisfied before accepting the acquired product?

Cost questions and metrics. Three questions were derived related to the monitoring of the cost Q3.1, Q3.2 and Q3.3. Each question has one metric M3.1.1, M3.2.1 and M3.3.1 respectively. The cost questions and metrics are shown in Table 5.22.

Table 5.22

Cost Questions and Metrics

G3: To analyze the development process cost for the purpose of monitoring and controlling with respect to the cost of fix, cost of activity and project budget from the viewpoints of the GOMM members	
Questions	Metrics
Q3.1: What is the cost of fix post to release problem in a month?	M3.1.1: Dollar cost related to fix post to release problems.
Q3.2: What is the current cost by activity for each Web application product?	M3.2.1: Number of dollars spent to date for activity i / Number of dollars estimated for activity.
Q3.3: What is the current budget status of the project?	M3.3.1: Number of total dollars spent to date / Number of total dollars estimated.

Quality questions and metrics. Seven questions were defined for monitoring the quality of the product Q4.1, Q4.2, Q4.3, Q4.4, Q4.5, Q4.6 and Q4.7. Question Q4.1 related to the distribution of the failure that can be measured by metric M4.1.1. Question Q4.2 related to the defect density that can be measured by metric M4.2.1. Question Q4.3 related to the defect detection process that can be measured by performing metric M4.3.1. Question Q4.4 related to the product reliability that can be measured by performing one metrics M4.4.1. Question Q4.5 related to the fault locating efforts and fixing fault effort that can be measured by performing two metrics M4.5.1 and M4.5.2. Question 4.6 related to the product usability that can be measured by performing metric M4.6.1, M4.6.2 and M4.6.3. Question 4.7 related to

the maintainability that can be measured by performing three metrics M4.7.1, M4.7.2, and M4.7.3. Quality questions and metrics are shown in Table 5.23.

Table 5.23

Quality Questions and Metrics

G4: To analyze the quality aspects for the purpose of monitoring with respect to the security, product reliability, usability and maintainability from the viewpoints of the GOMM members	
Questions	Metrics
Q4.1: What is the distribution of failure after delivery?	M4.1.1: Severity classification for each detected failure (fatal, major, minor and other).
Q4.2: What is the defect density?	M4.2.1: Number of Do (iteration) i defects / metric for size in iteration i(LOC).
Q4.3: What is the quality of the defect detection process?	M4.3.1: Number of pre-release defects in Do (iteration) / Number of pre-release + post-release defects.
Q4.4: What is the product reliability?	M4.4.1: Number of defects / execution time.
Q4.5: What is the total effort in hours spent in locating the fault vs. total effort spent for fixing the fault?	M4.5.1: Effort in hours for locating each fault.
	M4.5.2: Efforts in hours for fixing the fault.
Q4.6: How to monitor the usability of Web application?	M4.6.1 No. of page links/ total number of internal links (navigability)
	M4.6.2 Response time
	M4.6.3 Memory space
Q4.7 How to monitor Web application's maintainability?	M4.7.1 Dynamic pages/ total no. of pages (changeability) should be low
	M4.7.2 Dynamic testing LOC/ total LOC testability should be low
	M4.7.3 1/ no of direct links (stability) should be high

The questions and metrics defined in Table 5.23 are related to the security, reliability, usability and maintainability as identified by Wu and Offutt (2002), and Lilburne et al., (2004) as the most important Web application quality factors.

Time questions and metrics. Three questions were defined for monitoring the time Q5.1 and Q5.2. Question Q5.1 related to the reuse artifacts percentage that can be measured by performing two metrics M5.1.1 and M5.1.2. Question 5.2 related to the

development time that can be measured by one metric M5.2.1. The monitoring time questions and metrics are shown in Table 5.24.

Table 5.24

Time Questions and Metrics

G5: To analyze development life cycle time for the purpose of monitoring with respect to the reuse artifacts, time for each iteration, project velocity from the viewpoints of the GOMM members	
Questions	Metrics
Q5.1: What is the percentage of the reuse artifacts?	M5.1.1: Number of SLOC of reusing code / Number of SLOC completed to date.
	M5.1.2: Number of reused Web pages / total Web Pages number.
Q5.2: What is the development time for each Web application product?	M5.2.1: Elapsed time / estimated time.

Produce the monitoring plan. After defining the goals, questions and metrics for the whole measurement mechanism, a measurement plan should be identified. This plan clarifies the data collection procedures and instruments, then move to the data collection step. The outputs of the first planning meeting for monitoring are MT, goals, questions and metrics for the whole process, as well as the monitoring plan.

Prioritizing the goals, this sub activity will be performed in the Do (iteration) meeting by the master and MT in order to specify which goal should the team concentrate on the next iteration. The outputs of this action are the prioritized goals for the next iteration.

After the planning phase the Do will start by taking the outputs of the planning phase.

5.4.2 Do (iteration)

This phase will be performed based on the development activities identified in the Extended Agile method. Some other activities related to the measurement (monitoring) should also be performed. The aim of this phase is to perform and execute all the Do (iteration) backlog items that were specified by the Do iteration planning meeting. In this phase, the Web application product is developed through many Do (iterations). The number of iterations ranges from 3-8 iterations. The development activities are performed by the DT during the Do phase. However, the MT should create the metric base for saving and retrieving metrics. The Do phase activities are shown in Table 5.25.

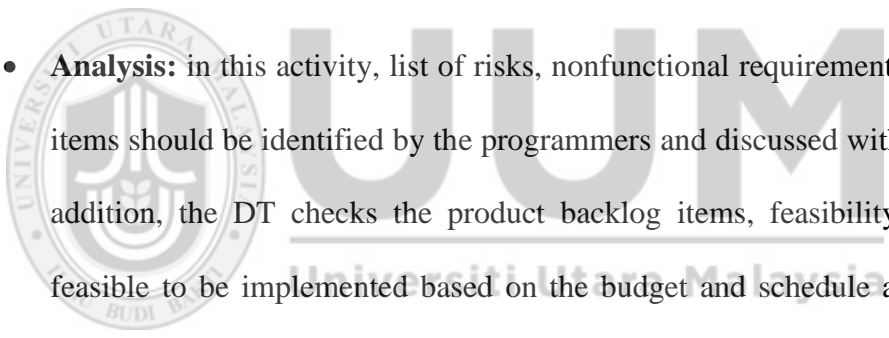
Table 5.25

Do phase

Phase name	Activities	Method	Practice	Tools	Team member(s) involved	Outcomes
Do (iteration)	Analysis	Extended Agile method.	-	Rational rose	DT	Use case diagram, context diagram, ER diagram and monitoring data
	Design	Extended Agile method.	Simple design	Argo UWE	2 programmers	Interface design, navigation design, content design and monitoring data
	Code	Extended Agile method.	Coding standards, pair programming, refactoring and collective ownership	Code base	2 programmers	Code, feedback to the design activity and monitoring data

Test	Extended Agile method.	TDD	Casper JS	Tester	Testing code and monitoring data
Daily reviewing	Extended Agile method.	Daily meeting	Burn down chart and task board	Master and DT	Progress and what to do next day, and monitoring data
Do (iteration) reviewing	Extended Agile method.	Do (iteration) review meeting, continuous integration, small release and metaphor	Task list, physical or electrical task board.	Master, PO, DT and MT	Increment and refined backlog item and the monitoring data

Several activities involved in this phase are discussed as follows:

- 
Analysis: in this activity, list of risks, nonfunctional requirements, and reuse items should be identified by the programmers and discussed with the DT. In addition, the DT checks the product backlog items, feasibility (items are feasible to be implemented based on the budget and schedule available for the system development). The main outcomes of this activity are use cases and ER diagrams that are necessary for executing the whole increment by using specific tools such as rational rose.
- Design:** in this activity, class and object diagrams that describe the GUI and the entire design specification which are determined by the Web design method must be also presented by the two programmers. In addition, the Web pages, conceptual design of the whole Web application and the navigation methods should be identified in this activity. Design activity must follow the

simple design practice. The suggested tool for this activity is the Argo UWE. Two programmers were involved in this activity. The outcomes of this activity are interface design, navigation design, content design and monitoring data.

- **Coding:** This activity should follow coding standards, code ownership, pair programming and continuous integration practices to ensure and confirm the XP practices application during the development process. The coding process is performed by two programmers who are using one monitor. One for writing the code and the other for validating the code. The quality of the code is assured by using TDD and refactoring practices. Suggestion tool for coding activity is the code base to save the produced code. The outcomes of the coding activity are the Web application itself, unit tests and feedback to the design activity.
- **Test:** The code will be tested frequently. Any part of the code that has been tested will be integrated into the system. This activity will be repeated until the whole system is integrated. This step also ensures the use of continuous integration practices, which is useful for reducing the implementation risks. The testing activity emphasized on performing the TDD practice. Suggested tools for performing the testing is Casper JS. One tester will be involved in this activity to produce the tested code and monitor the data.

- **Daily reviewing:** this activity will be performed by deploying practice known as daily meeting organized by the master. This meeting is conducted to keep track of the progress. Each DT member summarizes “what we have done today, what we will do tomorrow and what impediments he faces”. The duration of this meeting is fifteen minutes a day. The daily meeting will be conducted by the DT and master. The DT may find it useful to maintain the current Do (iteration) tasks list using the tools that are likely used in this meeting such as burn down chart and task board.
- **Do (iteration) reviewing:** this activity will be performed by conducting a particular practice called Do (iteration) review meeting. This meeting is held by the master, PO, DT and MT on the last day of the Do (iteration) to assess the iteration and decide on the following activity. The PO decides on the product backlog item which is done item by negotiating with the DT who will ensure the metaphor practice. In the event that the PO announces any item as not done, this item will be returned to the product backlog and prioritized by the PO as a candidate for the future Do (iteration). The Master helps the PO and DT to change over their feedbacks into product backlog items. Based on that, this meeting may refine the product backlog list items by including new items. The outcomes of this meeting are the increment by applying small release duration time (2 weeks) and continuous integration practices that helped the team to add the increment to the system in the next phase. This meeting also provides a time for the GOMM team members to

collect data and monitor the progress. A tasks list described by the task board (electrical or physical) is helpful to be used in order to determine whether the status of the iteration backlog items is completed or not.

The outcomes of this phase are the system increment, refined backlog items and the monitoring data.

5.4.3 Check

This phase depends on the monitoring that includes three activities: collect, store, and analyze the metrics as shown in Table 5.26.

Table 5.26

Check Phase

Phase name	Activities	Methods	Practices	Tools	Team involved	Outcomes
Check	Collect the Metrics	GOMM	Self-prepared data	Questionnaire	GOMM member 1	Collected questionnaires
	Store the metrics	GOMM	-	Metric base	GOMM member 1	Updated metric base
	Data analysis	GOMM	-	SPSS	GOMM member 2	Results

Each activity of the check phase is discussed in details as follows:

Collect the metrics. The data collection for monitoring will be performed for all the process activities. In this activity, the way of collecting metrics during the development process according to the measurement plan ought to be elucidated. Two

types of data used in the methodology are quantitative and qualitative. Quantitative data used metrics that measure the process activities, product cost, product quality, product time, process productivity and practices. Qualitative data measure the process quality factors. For the quantitative metrics, Table 5.27 describes the data owner of each metric and in which stage the data will be collected. In addition, the Table shows each metric (process activities, cost, quality, time, productivity and practices) and in which process activity was used.



Table 5.27

Development Process and Quantitative Metrics

Process phase	Process Activity	Metrics					
		Process activities	Cost	Quality	Time	Productivity	Practice
Plan	Identifying the product backlog items	Product backlog items number, estimated iteration number and estimated LOC					
	<ul style="list-style-type: none"> • Prioritize the items • Split the items • Estimate the items 	M1.1.1.1: Number of product backlog items completed to date vs. total number of product backlog planned.	Estimated cost M3.3.1: Number of total dollars spent to date vs. number of total dollars estimated.		Estimated time		PM1.1.1: Number of Do (iteration) planning meetings per one application
Do	Analysis						
	Design	M1.2.1.3: Total Number of internal links vs. number of Web pages		M4.6.1: No. of page links/total number of internal links (navigability) M3.6.3 Memory space M3.6.2 Response time M4.7.1:dynamic pages/total no. of pages (changeability) should be low M4.7.3:	M5.1.2: Number of reused Web pages vs. total Web Pages number.		PM3.3.1: Number of LOC of the current release - total LOC vs. Total LOC

				1/number of direct links (stability) should be high			
Code	M1.2.1.1: Number of LOC completed to date vs. Total Number of planned SLOC. M1.2.1.2: Number of Web pages to date vs. total number of Web planned page.			M4.5.1: Effort in hours for locating each fault. M4.5.2: Efforts in hours for fixing the fault.	M5.1.1: Number SLOC of reusing code vs. number of SLOC completed to date.	M2.1.1: Number of KLOC vs. staff in month.	PM2.4.1: Adherence of coding standard (High, Low). PM3.5.1: Number of team members who made changes in the code.
Test	M1.3.1.1: Number of test completed to date vs. total number of planned test. M1.3.1.2 number of testing line of code / total number lines of code						PM2.1.1: Number of lines of duplicated code removed vs. total line of code per iteration. PM2.1.2: Number of tests completed to date vs. Total Number of tests planned.
Daily reviewing		M3.2.1: Number of dollars spent to date for activity i vs. number of dollars estimated for activity.					PM1.2.1: Number of daily meetings per one application?

	Iteration reviewing		M3.1.1: Dollar cost related to fix post to release problems.	M4.2.1: Number of iteration i defects vs. metric for size in iteration I (LOC). M4.3.1: Number of pre-release defects of an iteration vs. Number of pre-release + post-release defects.	M5.2.1: Elapsed time vs. estimated time.		PM1.3.1: Number of review meetings done per one application? PM3.4.1: Number of meetings between development team and the client?
Act	Save the increment to the repository						PM3.1.1: Number of LOC of the first release - the LOC of the next release vs. total NLOC
	Integrate with the system						PM 3.2.1: Total number of line of code added, removed and updated) vs. total line of code for the previous iteration.
	Final release			M4.4.1: Number of defects vs. execution time.			

For the process factors Data, Table 5.28 describes the factor data, where to collect and the data owner.

Table 5.28

Data Collection for the Process Factors

Factor	Where to collect	Data owner
Requirement completeness	Plan phase	PO
Design completeness	Design activity	Programmer
Coding completeness	Coding activity	Programmer
Testing completeness	Testing activity	Tester
Project management completeness	Review meeting	Master
Requirement consistency	Plan phase	PO
Design consistency	Design activity	Programmer
Coding consistency	Coding activity	Programmer
Testing consistency	Testing activity	Tester
Project management consistency	Review meeting	Master
Requirement accuracy	Plan phase	PO
Design accuracy	Design activity	Programmer
Coding accuracy	Coding activity	Programmer
Testing accuracy	Testing activity	Tester
Project management accuracy	Review meeting	Master
Tailorability	Review meeting	Master
Flexibility	Review meeting	Master
Compatibility	Review meeting	Master
Accessibility	Review meeting	Master
Applicability	Plan phase	PO
Changeability	Plan phase	PO
Supportability	Plan phase	PO

Store the metrics. After the data collection activity ends, the data of the metrics should be stored in the (metric base).

Data analysis: after completing the metrics collection, the data analysis will be begun. The analysis activity will be performed by one MT member using the SPSS tool. The analysis results will be displayed in a simple report. This report contains feedback for the management in decision making to improve the work if there is an urgent situation in earlier times. The report will be updated consistently and introduced in the daily meeting.

5.4.4 Act

The Act phase depends on the development and monitoring. Three activities are included in the development: save the increment to the repository, integrate with the system and final release. For the monitoring, two activities are included: (1) making some improvements in the process, practices and progress based on the analysis results obtained from the Check phase and (2) preparing the final report. The activities for the development and measurement of the Act phase are shown in Table 5.29.

Table 5.29

Act Phase

Phase name	Development Activities	Monitoring activities	Practices	Tools	Team member	Outcomes
Act	Save the increment to the repository	-	-	Requirement repository	Programmer	Updated requirement repository
	Integrate with the system	Make some improvements in the process, practices and progress.	Continuous integration	System integration	DT, MT	New system version and recommendations to the next DO
	Final release	Prepare a feedback report	-	Burndown chart, task list	Master, PO, DT and MT	Final product and feedback report

Table 5.29 shows the practices, tools and team members involved in the Act phase.

The activities of this phase are described as follows:

Save increment to the repository: Once produced from the development, the increment should be saved in the requirement repository by the programmer.

Integrate with the system: This activity will be performed based on the development. However, there are some monitoring activities that should be performed during this activity. For the development, once the increment is saved to the requirement repository, it should be integrated into the system to make a new version of the product by performing continuous integration practice. The PO is in charge of announcing that all product backlog items have been fulfilled by making an agreement with the DT. This agreement is determined in the last Do (iteration) review meeting. In addition, the agreement represents a declaration that no more items should be added to the product backlog by using the burn down chart and the task list tools. The endorsement of whether the system is completed successfully depends on the PO satisfaction. The system is now ready to be launched by performing the integration sub activities: completing the requirements, saving (requirements repository), integrating the system, testing, and documenting.

For the monitoring activity, the MT should provide some improvements for the next Do iteration that related to the process, practices and progress. The main outcomes of this phase are new version of the system and recommendation to the next Do phase. Furthermore, if the previous Do was the last Do (iteration), then the final product

will be submitted to the customer and final feedback report will be presented to the management. The report describes the monitoring process, results and feedback from planning until receiving the final product. For further discussion about the practices and tools please refer to Appendix G and appendix H respectively.

5.5 Summary

This chapter gives a comprehensive overview on how to build monitoring and Agile based Web application development methodology for small software firms. This methodology consists of a process (Plan, Do, Check and Act), set of methods (combined XP and Scrum method, Web design method and GOMM), practices (development, Management and measurement) and team structure (roles and responsibilities). The process of this methodology is to ensure the quality of the Web application development using the Agile development methods (Scrum combined with XP) and monitor the measurement mechanism using the GOMM method. The measurement mechanism monitors the quality of both process and product.

CHAPTER SIX

METHODOLOGY EVALUATION

6.1 Introduction

This chapter explains the evaluation process of the proposed MOGWD. The evaluation was carried out through verification and validation. Verification was performed using expert review method based on Delphi technique and the validation was performed using case study and yard stick validation. The discussion of this chapter begins with verification, and ends with validation of the proposed methodology.

6.2 Verification based on the experts review

The aim of the verification process is to ensure that the main components in the proposed methodology, such as activities, methods, practices, tools and team structure are comprehensive, understandable and feasible to be used by SSF. The verification process was carried out through expert review method based on the Delphi technique. This technique was performed using sequential rounds, in which, each round has several activities. Three rounds were required to complete the verification process. The following sections explain the results of each round.

6.2.1 Results of Round one

After identifying the experts, and getting their acceptance to participate the verification, the first round of the verification started by sending a copy of the questionnaire, proposed methodology and expert cover letter via email and interview

to the experts. The expert cover letter is shown in Appendix C. This round gave the experts an opportunity to study the proposed methodology carefully and fill up the questionnaire. Two types of questionnaires were used in this round: questionnaire for the knowledge experts who focus on the theoretical part (Appendix D) and questionnaire for a domain expert those who focus on the technical part (Appendix E). However, there are some questions that can be answered by both knowledge and domain experts. The experts took one month to send their feedback. The feedback was analyzed once received.

The following sections illustrate the experts' answers and the suggestions that are related to the verification criteria discussed in Chapter Three.

6.2.1.1 Answers and suggestions related comprehensive criterion

This part illustrates the expert's answers and suggestions that related to the comprehensiveness of the methodology. Table 6.1 describes the expert's answers and suggestions.

Table 6.1

Experts Answers related to comprehensiveness

Component		Percentage	Experts involved	Suggestion
Activities	Development	100%	Knowledge and domain	-
	Measurement	100%	Knowledge and domain	-
Methods	GOMM (metrics)	100%	Knowledge and domain	-
	Web design method	-	-	-
Practices		100 %	Knowledge and domain	-
Tools		-	-	-

Team structure	100%	Knowledge and domain	-
----------------	------	----------------------	---

Table 6.1 shows that all the experts indicated that all the MOGWD methodology components are comprehensive. The development and the measurement activities are comprehensive for Web application development in SSF. Regarding the Web design method, the comprehensiveness criterion was excluded as this method concentrates only on the design activity. In addition, experts were not asked about the tools comprehensiveness as the tools are not meant to be a contribution in this study, and it should not cover all the MOGWD methodology phases. Based on the above Table, it's clearly shown that MOGWD methodology provides a set of comprehensive components such as activities, methods, practices, tools and team structure. Therefore, the achievement of this criterion ensures that the methodology components are well-interactive with each other to produce a high quality Web application.

6.2.1.2 Answers and suggestions related understandability criterion

This part illustrates the expert's answers and suggestions which are related to the understandability of the methodology. Based on this criterion, the methodology components should be correct, clear and well organized. However, the practices have been excluded from this criterion as the practices used in this study are adopted from previous studies. Table 6.2 describes the expert's answers and suggestions.

Table 6.2

Experts Answers related to understandability

Component		Correct	Clear	Well-organized	Experts involved	Suggestion
Activities	Development	100%	100%	100%	Knowledge and domain	- Add training session
	Measurement	100%	100%	100%	Knowledge and domain	
Methods	GOMM (metrics)	98%	96.8%	100%	Knowledge and domain	- Update the metrics
	Web design method	87.5%	87.5%	100%	Knowledge and domain	- Discuss the steps of the Web design method
Practices		-	-	-	-	-
Tools		-	87.5%	-	Knowledge and domain	- Include a table to clarify consist of tool name, purpose and place of using each tool
Team structure		87.5%	87.5%	100%	Knowledge and domain	- Define the role and responsibilities of each team member - Add another team member to perform the measurement process

Table 6.2 shows that the majority of the experts indicated that all the MOGWD methodology components are correct, clear and well-organized. Regarding to the development and measurement activities, all of the experts indicated that this component is correct, clear and well-organized. However, one of the knowledge experts suggested adding a training session to help the team to understand the process activities. Nevertheless, a majority of the experts (87.5 %) indicated that the Web design method is correct and clear. Furthermore, they claimed that it needs to be discussed further in details.

Whereas, the majority of the experts (98 %) found the metrics are used in the GOMM correct. However, two out of eight experts said that four metrics (M4.6.2), (M 4.6.3), (M5.1.1) and (PM2.1.1) are not correct which means these metrics need to be modified or updated. Furthermore, 96.8% of the metrics were found clear. However, two out of eight experts said that six metrics (M4.6.2), (M4.6.3), (M2.1.1), (PM2.1.1), (PM2.3.1), (Q.M6.1.1) and (Q.M8.1.4) are not clear which means these metrics need to be modified or updated.

For the tools part, experts indicated that the tools are 87.5 % clear. However, they suggested creating a table to clarify the tool name, purpose and where to use each tool.

Finally, the experts agreed that team structure 87.5 % correct and 87.5% clear. In addition, all the experts were asked if one GOMM member is enough to perform the measurement process, 50% of them said it is not enough to use one team member to perform the measurement process. Therefore the experts suggest clearly defining the role and responsibility of each team member and adding another team member to the monitoring team.

Based on the Table 6.2, it can be concluded that the MOGWD methodology is understandable to the experts in terms of its clearness, correctness and its well-organized components with some modifications. The achievement of this criterion supports the suitability and usability of the MOGWD methodology for the team.

6.2.1.3 Answers and suggestions related feasibility criterion

This part illustrates the experts' answers and suggestions that are related to the feasibility of the methodology. Table 6.3 describes the expert's answers and suggestions.

Table 6.3

Experts Answers related to feasibility

Component		Percentage	Experts involved	Suggestion
Activities	Development	100%	Knowledge and domain	-
	Measurement	100%	Knowledge and domain	-
Methods	GOMM (metrics)	97.4%	Knowledge and domain	Update the metrics
	Web design method	100%	-	-
Practices		100 %	Knowledge and domain	-
Tools		100%	-	-
Team structure		100%	Knowledge and domain	-

Table 6.3 shows that all the experts indicated that all of the MOGWD methodology components are feasible. However, 97.4% of the metrics were found feasible to be used. However, three out of eight experts said that (M4.6.2), (M4.6.3) are not feasible to be used for SSF, while two experts said that (M2.1.1), (PM2.1.1) and (PM3.2.1) are not feasible which mean these metrics need to be modified or updated.

Based on the results of Table 6.3, it can be concluded that methodology is feasible to be used for the SSF. Therefore, the achievement of this criterion means that the MOGWD methodology meets the SSF characteristics in the lack of resources such as

employee number, budget, and experience. Moreover, it can deal with Web application high changing requirements.

6.2.1.4 Answers and suggestions related to the general overview part.

In this part, the experts were asked some questions related to the general overview of the whole methodology. The questions were asked to determine the correctness and the clearness of the theory used to build the methodology. The questions were answered by knowledge experts. Expert answers related to the general overview part. The results of this part are shown in Table 6.4.

Table 6.4

Experts Answers related to the general overview Part.

Questions	Percentage
The theory used for building the methodology correct	100%
The theory used for building the methodology clear	100%

Table 6.4 shows clearly that all of knowledge experts agreed that the theory which has been used for building the proposed methodology is correct and clear. However, the experts provided some suggestions to improve the MOGWD methodologies such as:

- The MOGWD methodology creation process should be explained clearly.

- Because of the methodology have many metrics to be performed during the process it may take longer time, some quantitative metric should be excluded or prioritized.

6.2.2 Results of round two

In this round, data collection and analysis were completed. Results obtained from round one used as input for round two. This round aimed to modify the proposed methodology based on the required modifications that were suggested by the experts in the first round. Table 6.5 summarizes the required modifications that were needed to modify the proposed methodology.

Table 6.5

Required Modifications

Component name	Expert suggestions	Required modifications
Process and methods	Add simple training session before performing the methodology activities in order to make the developers understand all the components of the methodology before they start using it.	Conduct training session for 3 to 7 days before starting development and measurement process.
	Clarify the Web design step by adding chart.	Chart added.
	Make each team member prepare the data that he owns on small sheet and give to GOMM member in order to reduce the time and efforts for collecting monitoring program data.	Every team member should prepare the data that he owns for the GOMM member.
	Priorities the goal of measurement based on the company aims or demands. Furthermore, reduce from the quantitative metric also to reduce the time consuming.	Priorities the goal in the planning phase of the measurement by development team and customer.
	Add, Exclude and modify some metric in GOMM.	<ul style="list-style-type: none"> • Add metrics M4.4.2, M4.4.3 and M4.4.4 for reliability. • Exclude metrics M4.6.2 and M4.6.3

		<p>from the usability metrics because they are not applicable.</p> <ul style="list-style-type: none"> • Delete the metric M1.3.1.2 because it's covered by the practice monitoring metrics PM2.2.2 and update the metric numbering conversely. • Delete the metric PM2.2.1 because it's covered by the process monitoring metrics M1.3.1.1 and update the metric numbering conversely. • Delete the question PQ2.1 and metric related PM2.1.1 because the metric covered by consistency factor metric QM2.3.7 update the metric numbering conversely. • Update the metrics QM6.1.2 and QM8.1.4 as the expert mentioned are not clear.
Tools	Add table for each tool, purpose and the place of use.	Table is added.
Team structure	Clearly define the role and responsibility of each team member.	Table is added.
	Clearly show in which activity each member plays his role.	
	Add one team member to perform the monitoring process	One team member is added.
General Overview	Explain the creation phase clearly	<ul style="list-style-type: none"> • Explain about the methodology construction in the research methodology part

Table 6.5 illustrates the required modifications which have been done to improve the proposed methodology. The next sections explain in details these modifications.

6.2.2.1 Process and Methods Modifications

The experts suggested a number of modifications related to the process and methods parts such as:

- Adding training session to be conducted by the whole team members before using the MOGWD methodology. The session intends to familiarize the team members with methodology and clarifies their role during the process.
- Including a chart to clarify the Web design steps and structure. The chart should show that steps are performed iteratively. However, it is important to say that the steps should be performed during different phases of the process. For example, the requirement analysis will be performed in Plan phase. Conceptual design, navigational design and implementation design will be performed in the design activity in the Do phase. Construction will be performed in the Act phase. The chart is shown in Figure 5.3 in Chapter five.
- Adding two practices to the GOMM. The first practice is self-preparing data which involves improving the process of monitoring by making each development member prepares the data that he owns such as the tester is responsible for prepare the testing LOC. While the second practice is prioritizing the goals by helping the team to prioritize the monitoring goals based on their importance and the company demands (see appendix G).
- Modifying some of the GOMM quantitative and qualitative metrics. For the quantitative metrics, this action is done by adding some metrics related to the reliability, deleting the inapplicable metrics, removing the duplicated metrics and updating the unclear metrics. Table 6.6 shows the new list of quantitative metrics. Additionally, the table also includes in which activity the metric will

be performed, the calculation of each metric, the indicator of each metric and possible improvement. This table will be used in the validation stage.

For the qualitative metrics, the experts asked to update two metrics and the results of updating the metrics as followed:

Q.M6.1.2: Is this interaction between the team and the process done easily and clearly?

Q.M8.1.4: Is the contributed product, modules, code and measures saved to be used for the future project?



Table 6.6

New list for quantitative metrics

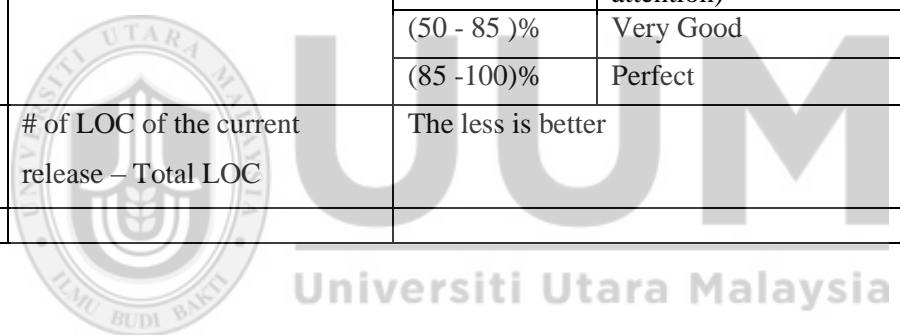
Phase	Activity	Metric #	Calculation	Indicators	Action for improvement	
Plan	Identifying the product backlog items					
	<ul style="list-style-type: none"> • Prioritize the items • Split the items • Estimate the items 	PM1.1.1	# of Do items number			
Do	Design	M1.2.1.3	#of internal links / #of web pages	Acceptable => 1 Need to improve: less than 1 The improvement by increasing the internal links or reducing the web pages	The improvement by increasing the internal links or reducing the web pages	
		M4.6.1	# of page links / #of internal links	(0 – 15) %	Poor (need to improve)	By increasing the page link (internal and external)
				(15 – 50) %	Acceptable (pay more attention)	
				(50 - 85) %	Very Good	
				(85 -100)%	Perfect	
		M4.7.1	# of dynamic pages / Total # of web pages	(0 – 15) %	Perfect	By reducing the dynamic ages
				(15 – 50) %	very good	
				(50 - 85) %	Acceptable (pay more attention)	
				(85 -100)%	Poor (need to improve)	
		M4.7.3	1 / # of direct links	(0 – 15) %	Poor (need to improve)	Reducing the direct inks
(15 – 50) %	Acceptable (pay more					

				attention)	
			(50 - 85)%	Very Good	
			(85 -100)%	Perfect	
	M5.1.2	#of reused web pages / Total # of web pages	Monitoring the reuse it should be acceptable if less than 50%.		Reducing the reused web pages
Code	M1.2.1.1	LOC completed to date / LOC estimated	(0 – 15) %	Poor (need to improve)	The action should be taken to increase the programmer productivity
			(15 – 50) %	Acceptable (pay more attention)	
			(50 - 85)%	Very Good	
			(85 -100)%	Perfect	
	M1.2.1.2	# of web pages to date / Estimated # of Web pages	(0 – 15) %	Poor (need to improve)	The action should be taken to increase the programmer productivity
			(15 – 50) %	Acceptable (pay more attention)	
			(50 - 85)%	Very Good	
			(85 -100)%	Perfect	
	M4.5.1	Effort in hour for locating each fault	Acceptable if the result < 3		Reduced by the application of pair programming
	M4.5.2	Effort in hour for fixing each fault	Acceptable if the result < 3		Reduced by the application of pair programming
M5.1.1	# of reused LOC / Total LOC	Acceptable if not more, than 50%		Reducing the reused code	
M2.1.1	# of KLOC for the programmer in the month	Not less than 3		Increase the programmer productivity	
PM3.5.1	# of team members who made changes on the code	Just the development team ranges from 2-5		Just he DT can change the code and at least the two programmers	
Test	M1.3.1.1	# of test completed to date / Total # of planned test	(0 – 15) %	Poor (need to improve)	Increase the number of tests by applying the TDD practice.
			(15 – 50) %	Acceptable (pay more attention)	

			(50 - 85)%	Very Good	
			(85 -100)%	Perfect	
	PM2.1.1	# of duplicated LOC removed / Total LOC	Not more than 30%		Reduce the duplicate the code by monitoring the code quality and ensuring the application of pair programming practice by the master
Daily reviewing	M1.1.1.1	# of product backlog items completed to date / Total # of product backlog planned	(0 – 15) %	Poor (need to improve)	Adding staff resources and increase the productivity.
			(15 – 50) %	Acceptable (pay more attention)	
			(50 - 85)%	Very Good	
			(85 -100)%	Perfect	
	M3.2.1	# of dollars spent for the Do/ Estimated Do budget	Acceptable if less than 100%		If the ratio high and the product still need time to be achieved the team should take some action to reduce the budget, by reusing code, pages... etc.
M3.3.1	Total # of Dollars spent / Estimated cost in dollars	Acceptable if less than 100%		If the ratio high and the product still need time to be achieved the team should take some action to reduce the budget, by reusing code, pages... etc.	
	PM1.2.1	# of daily meeting per one application	10-12 acceptable		The master should ensure that the meeting conducted daily
Iteration reviewing	M3.1.1	Dollars spent to fix post to release problems	-		-
	M4.2.1	# of Do defects / LOC for the DO	(0 – 15) %	Perfect	Monitor the code quality by insuring pair programming practice
			(15 – 50) %	very good	
			(50 - 85)%	Acceptable (pay more attention)	
(85 -100)%			Poor (need to improve)		

	M4.3.1	# of pre-release defect of the DO / (# of pre-release+ post-release defects of the DO)	(0 – 15) %	Poor (need to improve)	Improve by ensuring the application of the pair programming, continuous integration and refactoring practices.
			(15 – 50) %	Acceptable (pay more attention)	
			(50 - 85)%	Very Good	
			(85 -100)%	Perfect	
	M4.4.2	Mean time to find defects	Acceptable if the result <3.		Improve by ensuring the application of the pair programming, continuous integration and refactoring practices.
	M4.4.3	Mean time between two defects	Acceptable if the result <3.		Improve by ensuring the application of the pair programming, continuous integration and refactoring practices.
	M4.4.4	Mean time to recover	Acceptable if the result <3.		Improve by ensuring the application of the pair programming, continuous integration and refactoring practices.
	M5.2.1	Elapsed time / Estimated time	Acceptable if < 100		Encourage reuse to gain more time.
	M5.2.2	(Current DO time / Estimated DO time) * 100%	Acceptable if < 100		Encourage reuse to gain more time.
	PM1.3.1	# of review meeting per one application	Acceptable if the results = 1.		The master should insure the meeting after completing each iteration.
PM3.4.1	# of meeting between DT and client	Acceptable if the result >2.		The master should monitor the application of metaphor practice.	
M4.4.1	# of defects / Execution time	(0 – 15) %	Perfect	Improve by ensuring the application of the pair programming, continuous integration and refactoring practices	
		(15 – 50) %	very good		
		(50 - 85)%	Acceptable (pay more attention)		
		(85 -100)%	Poor (need to improve)		

Act	Save the increment to the repository	PM3.1.1	(LOC of the first release - LOC of the current release) / Total LOC	(0 – 15) %	Perfect	Reduce the iteration LOC to ensure the small release practices
				(15 – 50) %	very good	
				(50 - 85) %	Acceptable (pay more attention)	
				(85 -100)%	Poor (need to improve)	
	Integrate with the system	PM3.2.1	LOC added, removed and updated / Total LOC of the previous iteration	(0 – 15) %	Poor (need to improve)	By applying the pair programming and continuous integration practices
				(15 – 50) %	Acceptable (pay more attention)	
				(50 - 85) %	Very Good	
				(85 -100)%	Perfect	
		PM3.3.1	# of LOC of the current release – Total LOC	The less is better		Reduce the iteration LOC to ensure the small release practices
	Final release					



6.2.2.2 Tools Modification

The experts suggested a modification related to this part. The modification was to create a table that shows the phase, tool name and the aim of each tool. The table is shown in Appendix H.

6.2.2.3 Team structure Modifications

The experts suggested three modifications regarding to the team structure. The first defines the role and responsibility of each team member, the second modification shows the activities each member plays and the third adding one team member to perform the monitoring process. However, the first two modifications were done by adding Table 5.3 in the Plan phase in Chapter five that shows the role, responsibilities and stakeholder of each team member. The third modification performed by assigning new team member to be another GOMM member. As a result two members were assigned for monitoring, one for collecting the data and the other for analyzing and preparing the management report.

6.2.2.4 General overview modifications

A modification has been suggested by the expert regarding to the general overview part; it was clarifying the methodology construction in the research methodology part. Deep discussion is provided in research methodology section 3.3.3 on how the MOGWD methodology was created.

Nonetheless, the main output of this round was a report included the modifications and the improved methodology.

6.2.3 Results of Round Three

The report mentioned above was sent to the experts as a new round in order to get their approval and acceptance. E-mail was used to communicate with the experts.

The results of this round convey that the experts approved the modifications that have been done to the methodology components.

6.3 Validation based on case study and yardstick method

The aim of the validation process is to evaluate the effectiveness MOGWD methodology. Accordingly, two approaches have been used to perform the validation which are case study and yardstick validation.

6.3.1 Validation based on the case study method

In order to validate the effectiveness of the MOGWD methodology, one Jordanian SSF was identified and agreed to implement the methodology. The case study project aimed to develop a Content Management System (CMS). In order to simplify the validation process, this study had proposed a prototype system support tool that was used to assist the application of the MOGWD methodology. This tool called monitoring prototyping tool (MO-PT). The MO-PT is a prototype tool aims to support the monitoring of the product and the process quality during the

development. In addition, the tool performs the metrics calculation and provides a feedback report to the management at the end of each iteration.

The MO-PT was built using the PHP technology and My SQL. The structure of the tool consists of a front-end and back-end (see Figure 6.1).

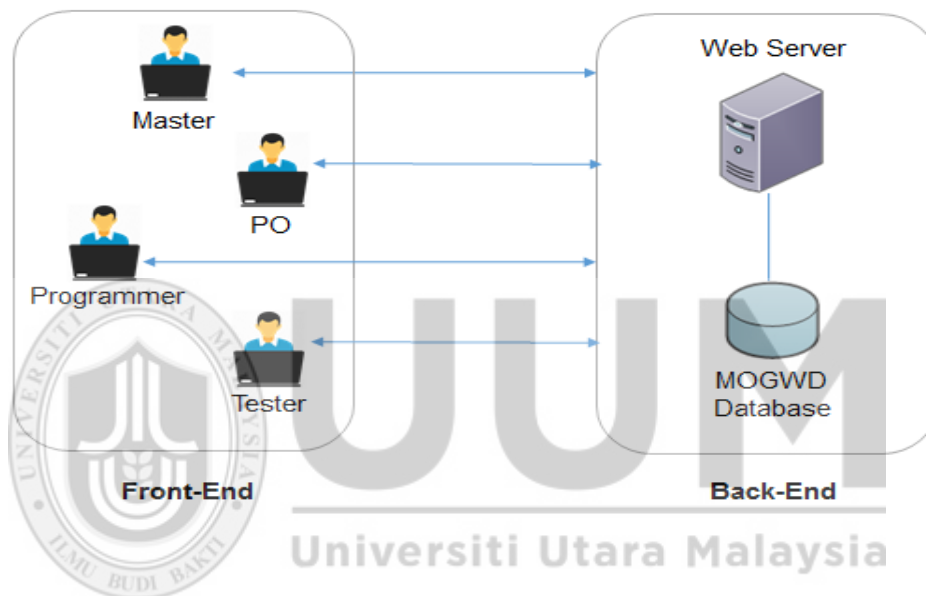


Figure 6.1. MO-PT structure

The front-end represents the interface that determines the interaction between the users and the system which created using the PHP language. On the other hand, the back-end represents the database and the server. The database for the MOGWD methodology was created using my SQL application. The database used to save the values of the metrics and calculation results. Furthermore, the tool should be uploaded to a host server on the internet. In this case study, the tool is only used to

calculate metrics and presenting the results. For more details for the MO-PT refer to Appendix L. The next section will discuss the results of the case study.

6.3.1.1 Case Study: Developing the CMS Web application by Firm “A”

Firm “A” was established in Jordan in 2010. It focuses on the Internet consulting and development. This firm helps clients to create and implement full-service digital business solutions. The main objective of this firm is to produce highly scalable business solutions and rich user experiences. In addition, it deals with a simple static or a fully dynamic Web application or e-commerce site. This firm has 24 employees working with managing and developing the Web application products. Firm “A” has one manager, three project or team leaders and 20 developers. In this case study, Content Management System (CMS) has been developed using MOGWD methodology. The CMS is created to allow the customer to manage their website. This system automatically generates navigation elements, makes the content searchable and indexable, track the users and manage their security settings. A CMS consists of three layers namely; presentation, application and database layer (see Figure 6.2).

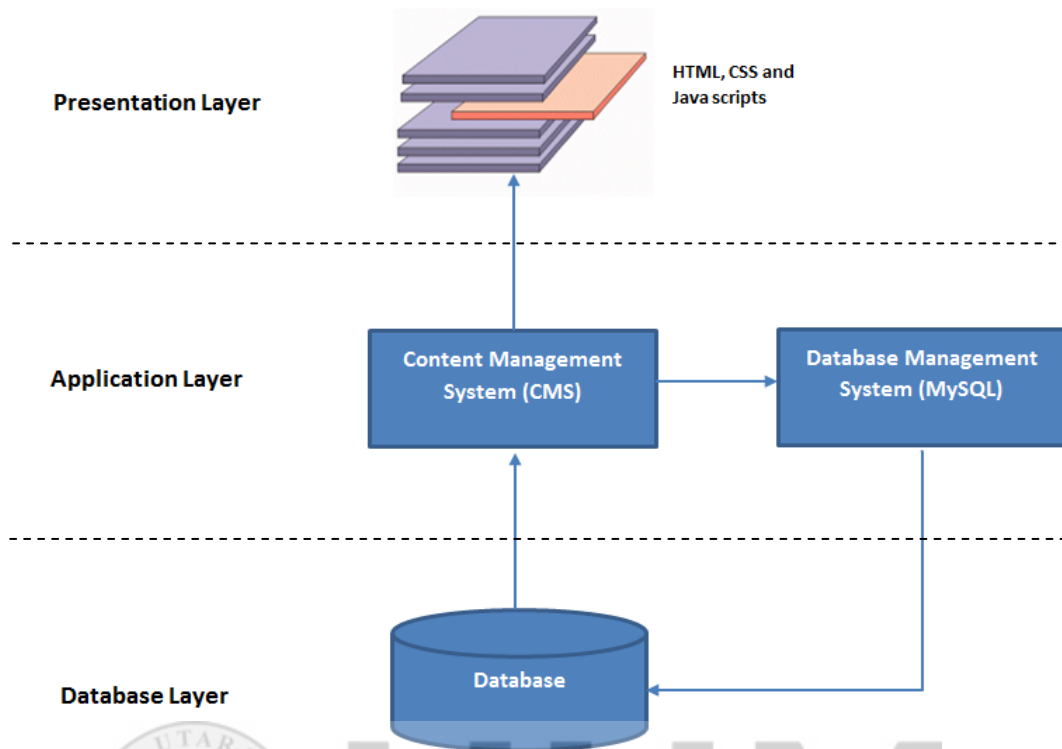


Figure 6.2. CMS layers

The presentation layer (user layer) is created using the HTML, Cascading Style Sheets (CSS) and Java scripts. This layer allows the content manager to manage the creation, modification, and removal of content their Web site. The application layer (developer layers) is the middle layer between the user and the database. The layer consists of the CMS and database management system (MySQL). The lowest layer of this system is the database which includes all the manipulations that were made by the database management system.

The team was given 3 days to study and understand the descriptions of the MOGWD before implementing it. After that, a meeting was held with the manager to clarify any misunderstanding or ambiguity of the MOGWD methodology.

The next sections describe the details of the MOGWD methodology related phases in developing and monitoring the quality of the CMS application.

6.3.1.1.1 Plan

Referring to section 5.4.1, the plan phase consists of the following management, development and monitoring planning.

A. Management planning

Three sub activities involved in the management planning; staffing, training and controlling. The staffing is defining each member role and responsibilities. Seven team members were assigned to perform this project; these members are: master, PO, two programmers, tester and two GOMM member. For the training session, each team member knows the activity that he plays and how to perform it. Controlling involved with accelerating the process and keeping it Agile. In this sub activity, the master should define the practices to keep the process Agile. These practices are shown in Appendix G.

B. Development planning

This activity includes five sub activities that performed in two meetings. The first two sub activities are: create the product backlog and perform the Web design method, these sub activities were performed in the first planning meeting. On the other hand, three sub activities were performed at the Do planning meeting, these

sub activities are: select the items that will be entered for the next Do (iteration), split the large items (if any) to smaller items and estimate the items.

In the first planning meeting the PO order the items of the product backlog and presents the list of them to the team as shown in Table 6.7. Twenty items were identified to be developed.

Table 6.7

List of product backlog items

Product backlog no.	Description
PB1	Detailing database design
PB2	Building unit tests
PB3	Securing Service
PB4	Log in page
PB5	Manage account page
PB6	Adding / Edit account page
PB7	Traceability of access.
PB8	Management System sections.
PB9	Manage the news
PB10	Manage articles
PB11	Manage ads
PB12	Manage the book
PB13	Manage the news sent
PB14	Manage the Archives
PB15	Manage Comments and activated and stopped
PB16	Control the breaking news appearance
PB17	Full control on-site
PB18	Statistics of the site and the number of visits
PB19	Tape news
PB20	Resetting password option

For performing the Web design method, the team sat together to specify the requirement for designing CMS. Five actions were involved with performing the

Web design method, namely, requirement analysis, conceptual design, navigational design, implementation and construction.

Requirements analysis is performed by taking the product backlog items that are related to the design. Conceptual design defines the modules, classes, and Web pages that need to be designed. Navigational design determined the number of links and the sitemap for the application. Implementation describes the interface items to be used for designing the whole application. The construction is to select the items to be entered into the Do phase. The outputs of performing the Web design method are shown in Table 6.8.

Table 6.8
The Main Outputs of Web Design Method

Web design action	Item	Outputs
Requirement analysis	# of modules (product backlog items)	20
Conceptual design	# of web pages	119
	#of dynamic pages	17
	# of classes	86
Navigational design	# of internal links	68
	# of direct links	6
Interface design	# of interface items	71

After finishing the first planning meeting, the PO and development team (DT) hold another meeting called Do (iteration) planning meeting. The aim of this meeting is to select product backlog items for the next Do (iteration). The items were priorities or ordered by the PO and estimated by the DT. The DT decided to divide the items into two iterations. The outputs of this meeting are shown Table 6.9.

Table 6.9

The Outputs of the Do Planning Meeting

Iteration no.	Duration	Product backlog item	Estimated Time	Priority
Iteration one	23Nov- 8 Dec (14 days)	PB1: Detailing database design.	2	High
		PB2: Building unit tests.	3	
		PB3: Securing Service.	2	↓
		PB4: Log in page	0.5	
		PB5: Manage account page.	2	
		PB6: Adding / Edit account page.	0.5	↓
		PB7: Traceability of access.	1.5	
		PB8: Management System sections.	0.5	↓
		PB9: Manage the news.	0.5	
		PB10: Manage articles.	0.5	
		PB11: Manage ads.	0.5	↓
		PB12: Manage the book.	0.5	low
Iteration two	9 Dec- 24 Dec (14 days)	PB13: Manage the news sent.	0.5	High
		PB14: Manage the Archives.	1	
		PB15: Manage Comments and activated and stopped.	1	↓
		PB16: Control the breaking news appearance.	2.5	
		PB17: Full control on-site.	3	↓
		PB18: Statistics of the site.	3	
		PB19: Tape news.	1.5	↓
		PB20: Resetting password option	1.5	Low

In addition, the meeting has other results related to the construction action of the web design method (see Table 6.10).

Table 6.10

Construction Action, Results

Web design action	Item	Iteration 1	Iteration 2	Total
Requirement analysis	# of modules (product backlog items)	12	8	20
Conceptual design	# of web pages	68	51	119
	#of dynamic pages	7	10	17
	# of classes	47	39	86
Navigational design	# of internal links	32	34	68
	# of direct links	3	3	6

Interface design	# of interface items	39	32	71
------------------	----------------------	----	----	----

C. Monitoring planning

For the monitoring planning, the team used the monitoring goals, questions and metrics that were defined by the researcher. In addition, the team should produce the monitoring plan that includes data collection procedure and data collection instrument. Moreover the MOGWD methodology allows the team to prioritize the monitoring goals. In this case, the researcher took the organization agreement to measure all the monitoring goals without excluding. Two team members were involved in the monitoring, one for collecting data from the team and the other member presents the report using the MO-PT to the management and describes the improvement actions that should be taken. After finishing the plan phase, the Do phase begins.

6.3.1.1.2 Do phase

This phase includes the activities of building CMS. These activities are: analysis, design, code, daily reviewing and iteration reviewing.

A. Analysis and design

After the Do items were analysis, the conceptual design, the navigation design and content design were created in this activity. The master ensured that the programmers follow the simple design practice. Two programmers were involved in

this activity. The outcomes of this activity are interface design, navigation design, content design and monitoring data. After the design finished the GOMM member collected the design data from the programmers.

B. Code

In this activity, the programmers started coding the two iterations sequentially; they took into their consideration the application of coding standards, code ownership, pair programming and continuous integration practices. After the code activity finished, the GOMM member asked the programmer to fill the coding data in the checklist.

C. Test

The code is tested regularly using the unit tests. Each feature of the system is tested individually, and then integrated to the system. The tester followed the TDD practice during the testing process. The tester also was asked to fill the testing data in the checklist.

D. Daily reviewing

Daily meetings were conducted by the master and DT. The aim of the meetings was to know what the team has done and what they will do in the next day. Data were collected from the PO during this meeting. However, the short duration of the

meeting did not allow doing that in every daily meeting. Therefore, the researcher took the DT agreement to collect the data at last daily meeting of each iteration.

E. DO (iteration) reviewing

This meeting is held by the master, PO, DT and MT on the last day of the Do (iteration). The outcomes of this meeting are the increment by applying small release duration time (2 weeks) and continuous integration practices that helped the team to add the increment to the system in the next phase. This meeting also provides a time for the GOMM team members to collect data from the master. The main outcome of the first Do reviewing meeting was not completed at the first iteration, therefore, another iteration was needed to accomplish CMS product.

6.3.1.1.3 Check phase

This phase includes three activities namely, collect, store and analyze the metric. The researcher took the monitoring data from the firm “X” and entered the data to MO-PT. The data analysis is supported and performed by the MO-PT.

6.3.1.1.4 Act phase

This phase includes three activities which are: safe the increment to the repository, integrate into the system and the final release.

A. Save increment to the repository

After the first iteration has been finished, the increment has been saved in the requirement repository by the team. In this activity, the PO provided monitoring data to the GOMM member.

B. Integrate with the system

Once the increment is saved to the requirement repository, it has been integrated with the system to make a new version of the product by performing continuous integration practice. The PO is in charge of announcing that all product backlog items have been achieved by making an agreement with the DT. In addition, the programmer was also required to fill the last section of the monitoring data in the checklist.

After finishing the development activities the monitoring report was presented. In this activity, the main role of the MO-PT is appeared. The data were analyzed and the report was issued from the master page by clicking on *view report* in the iteration page (see Figure 6.3).

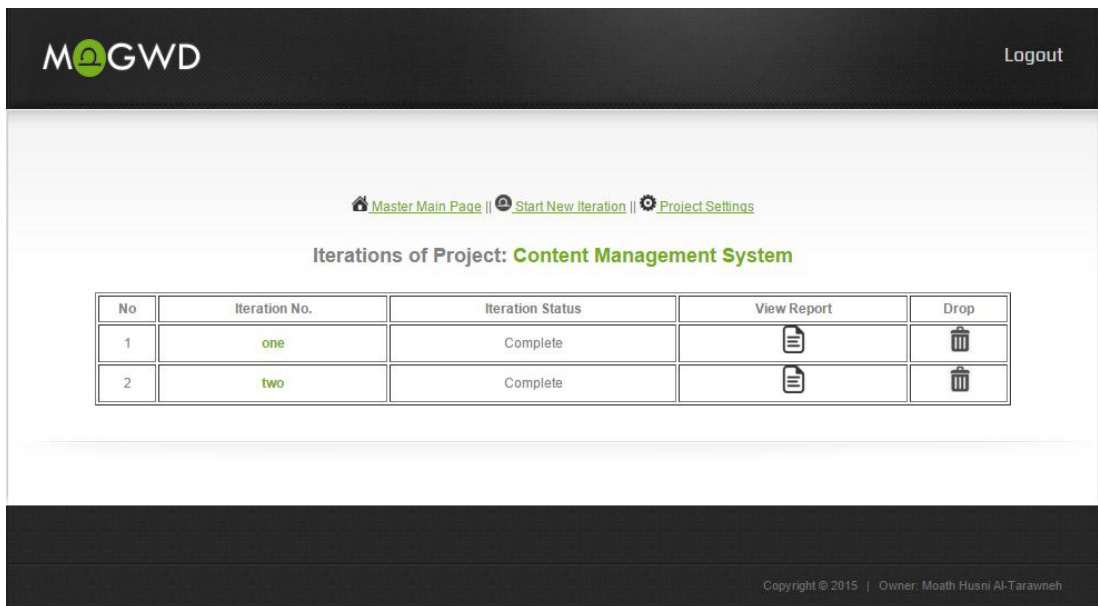


Figure 6.3. View report

Based on the above figure, each iteration has a monitoring report. Each report consists of quantitative metrics and qualitative metrics results. The report includes the indicators and action of improvement if needed for each metric. A snapshot of the quantitative and qualitative results that attain from the MO-PT is shown in Figure 6.4.

Results of Quantitative Metrics

Activity	Metric No.	Metric Name	Value	Indicator
Design	M1.2.1.3	Impact Factor	0.5	Need to improve Action
	M4.6.1	Navigability	40 %	Acceptable
	M4.7.1	Changeability	10.3 %	Perfect
	M4.7.3	Stability	33.3 %	Acceptable
	M5.1.2	Reused web pages	14.7 %	Acceptable
Code	M1.2.1.1	LOC Progress	46.8 %	Perfect
	M1.2.1.2	Web pages progress	52.3 %	Perfect
	M4.5.1	Effort for locating each fault (h)	2	Acceptable
	M4.5.2	Effort for fixing each fault (h)	3	Need to improve Action
	M5.1.1	Reused LOC percentage	15.3 %	Acceptable
	M2.1.1	Programmer Productivity	1.4	Need to improve Action
	PM3.5.1	Collective Ownership (number of team members who have the power to change the code)	3	Acceptable

Results of Qualitative Metrics

Qualitative Factor	Percent	Indicator
Requirement Completeness	87.5	Fully achieved
Requirement Consistency	75	Largely achieved
Requirement Accuracy	91.7	Fully achieved
Applicability	93.8	Fully achieved
Changeability	87.5	Fully achieved
Supportability	87.5	Fully achieved
Design Completeness	89.3	Fully achieved
Design Consistency	93.8	Fully achieved
Design Accuracy	83.3	Largely achieved
Coding Completeness	90.6	Fully achieved
Coding Consistency	85	Fully achieved

Figure 6.4. Quantitative and Qualitative Results

If any metric has the indicator “need to improve”, the MO-PT shows the action button beside the indicator. By clicking on that action button, a pop up message will be shown telling the team the action that should be taken (see Figure 6.5).

Code	M1.2.1.2	web pages progress	52.3 %	Perfect
	M4.5.1	Effort for locating each fault (h)	2	Acceptable
	M4.5.2	Effort for		Need to improve <input type="button" value="Action"/>
	M5.1.1	Reused		Acceptable
	M2.1.1	Program		Need to improve <input type="button" value="Action"/>
	PM3.5.1	Collectiv have the power to change the code)		Acceptable
Test	M1.3.1.1	Test Progress	53.8 %	Perfect
	PM2.1.1	Refactoring	10.9 %	Acceptable

Implement pair programming practice

Figure 6.5. Action message

After the project ended, the GOMM member merged the two iteration results into one report which has been presented to the management. The report showed the quantitative and the qualitative results. Table 6.11 shows the quantitative results.



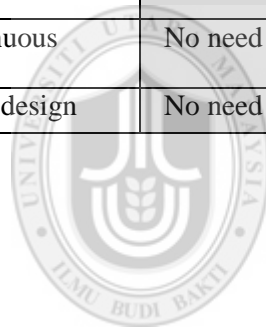
Table 6.11

Quantitative results

Phase	Activity	Metric #	Iteration one	Indicator	Action	Iteration two	Indicator	Action
Do	Design	M1.2.1.3: Impact factor	0.5	Need to improve	Increasing the internal links or reducing the web pages	0.7	Need to improve	Increasing the internal links or reducing the web pages
		M4.6.1: Navigability	40%	Acceptable	-	32%	Acceptable	-
		M4.7.1: Changeability	10%	Perfect	-	14%	Perfect	-
		M4.7.3: Stability	33%	Acceptable	-	33%	Acceptable	-
		M5.1.2: Reused web pages	15%	Acceptable	-	14%	Acceptable	-
	Code	M1.2.1.1: LOC progress	47%	Perfect	-	88%	Perfect	-
		M1.2.1.2: Web pages progress	52%	Perfect	-	91%	Perfect	-
		M5.1.1: Reused LOC percentage	15%	Acceptable	-	20%	Acceptable	-
		M2.1.1: Programmer productivity	1.4	Need to improve	Increase the programmer productivity	2.6	Need to improve	Increase the programmer productivity
		PM3.5.1: Collective ownership (number of teams who change the code)	3	Acceptable	-	3	Acceptable	-
	Test	M1.3.1.1: Test progress	54%	Perfect	-	95.4%	Perfect	-
		PM2.1.1:Refactoring	11%	Acceptable	-	15%	Acceptable	-
	Daily	M1.1.1.1: Backlog item's	60%	Perfect	-	100%	Perfect	-

reviewing	progress						
	M3.2.1:Do budget spent	92%	Acceptable	-	95%	Acceptable	-
	M3.3.1: Total budget spent	46%	Acceptable	-	93%	Acceptable	-
	PM1.2.1:Daily meeting	12	Acceptable	-	12	Acceptable	-
Iteration reviewing	M3.1.1: Dollars spent to fix the post to release problems	120		-	166	-	-
	M4.2.1: Defect density	0.2%	Perfect	-	0.1%	Perfect	-
	M4.3.1: Defect detection quality	73%	Perfect	-	75%	Perfect	-
	M4.4.2: Mean time to find defects	10	Need to improve	Ensure the application of the pair programming, continuous integration and refactoring practices	7	Need to improve	Ensure the application of the pair programming, continuous integration and refactoring practices
	M4.4.3: Mean time between two defects	22	Need to improve	Ensure the application of the pair programming, continuous integration and refactoring practices	16	Need to improve	Ensure the application of the pair programming, continuous integration and refactoring practices
	M4.4.4: Mean time to recovery	5	Need to improve	Ensure the application of the pair programming, continuous integration and refactoring practices	5.5	Need to improve	Ensure the application of the pair programming, continuous integration and refactoring practices

		M5.2.1: Consumed time percentage	31%	Acceptable	-	62%	Acceptable	-
		PM1.3.1: Review meeting	1	Acceptable	-	1	Acceptable	-
		PM3.4.1: Metaphor	7	Acceptable	-	8	Acceptable	-
		M4.4.1: Reliability (defects /execution Time)	20%	Very good	-	14%	Perfect	-
Act	Save the increment to the repository	PM3.1.1: Small release	No need		-	6.4%	Acceptable	-
	Integrate with the system	PM3.2.1: Continuous integration	No need		-	84%	Very good	-
		PM3.3.1:simple design	No need		-	-14034	-	-



UUM
Universiti Utara Malaysia

Based on Table 6.11, it is clearly shown that most of the quantitative metrics which have been used in the case study achieved the acceptance level. However, five metrics need to be improved namely, M1.2.1.3, M2.1.1, M4.4.2, M4.4.3 and M4.4.4. The result of metric M1.2.1.3 can be improved by increasing the internal links or by reducing the Web pages number. The metric result has been improved from 0.5 to 0.7 in the second iteration, but it still needs to reach 1.0. Regarding to M2.1.1 which related to the programmer productivity was improved from 1.4 to 2.6 in the second iteration but it has not reached 3. For this metric, the team accepted the results but they said that they will take it into their consideration in the future. For the metrics M4.4.2, M4.4.3 and M4.4.4 the team enhanced the results of the metrics during the two iterations; however they did not reach the acceptable level. Therefore, the team mentioned that these results are acceptable from their point of view and they will improve it in the future. Table 6.12 shows the results of the qualitative metrics.

Table 6.12

Qualitative results

Factor	Iteration one percentage	Indicator	Iteration two percentage	Indicator
Requirement completeness	88%	Fully achieved	83%	Largely achieved
Requirement consistency	75%	Largely achieved	88%	Fully achieved
Requirement accuracy	92%	Fully achieved	92%	Fully achieved
Design completeness	89%	Fully achieved	89%	Fully achieved
Design consistency	94%	Fully achieved	94%	Fully achieved
Design accuracy	83%	Largely achieved	92%	Fully achieved
Coding completeness	91%	Fully achieved	81%	Largely achieved
Coding consistency	85%	Fully achieved	88%	Fully achieved
Coding accuracy	100%	Fully achieved	88%	Fully achieved
Testing completeness	88%	Fully achieved	83%	Largely achieved
Testing consistency	83%	Largely achieved	75%	Largely achieved
Testing accuracy	100%	Fully achieved	88%	Fully achieved
Project management completeness	95%	Fully achieved	90%	Fully achieved

Project management consistency	93%	Fully achieved	93%	Fully achieved
Project management accuracy	75%	Largely achieved	100%	Fully achieved
Tailorability	92%	Fully achieved	92%	Fully achieved
Flexibility	88%	Fully achieved	100%	Fully achieved
Compatibility	88%	Fully achieved	100%	Fully achieved
Accessibility	82%	Largely achieved	89%	Fully achieved
Applicability	94%	Fully achieved	94%	Fully achieved
Changeability	88%	Fully achieved	88%	Fully achieved
Supportability	88%	Fully achieved	81%	Largely achieved

It is clearly shown in the above table that all the factors fully or largely achieved during the two iterations. Therefore, no improvements need to be taken. After finishing the project the team members answered the validation form (see Appendix K). The results of the validation will be discussed in the next section.

6.3.1.2 Validation results

The validation was conducted through an interview with the MOGWD team. The team answered the validation form that was constructed based on a set of evaluation factors as shown in Chapter Three, Section 3.4. These factors are: gain satisfaction, interface satisfactions, task support satisfaction, perceived usefulness and perceived ease of use. Each factor has certain items. The team was asked to rank the level of these items achievement. Therefore, Five Likert scales ranging from strongly disagree (value 1) to strongly agree (value 5) were used to describe the level of achievement of the items. The results were calculated by getting the mean score for each item and selecting the appropriate interval that represent the actual mean. An appropriate interval scale was required to represent all levels of achievement. Table 6.13 shows the mean interval presentation and the achievement level.

Table 6.13

Representations of the achievement levels

Mean interval presentation	Achievement level
From 1 to 1.80	Not achieved
From 1.81 to 2.60	Very limited achievement
From 2.61 to 3.40	Partially achieved
From 3.41 to 4.20	Largely achieved
From 4.21 to 5	Fully achieved

Table 6.14 shows the validation results.

Table 6.14

Validation results

Gain satisfaction			
Item	Mean	Overall mean	Achievement level
Decision support satisfaction: is the MOGDW methodology helps the management to take a well-defined decision based on the process and product monitoring?	4.3	4.3	Fully achieved
Comparison with the current development method: is the MOGDW methodology better than the old development that you used in terms of the structure and achieve results?	4.3		
Clarity (clear and illuminate the process): Is the MOGDW process clear to the development team, where each phase clearly presents the required inputs, outputs, methods or practices, and activities?	4.4		
Task Appropriateness: Are the phases and activities that presented in the MOGDW methodology appropriate for developing and monitoring web application in your company, and is the flow of the process presented in a systematic and effective way?	4.1		
Interface satisfaction			
Item	Mean	Overall mean	Achievement level
Internally consistent: the MOGDW methodology is internally consistent?	4	4.1	Largely achieved
Organization (well organized): the component of MOGDW methodology well organized and structured that makes the process is easy to perform?	4.1		

Appropriate for audience: is the MOGDW methodology appropriate for the audience. Those audiences are referred to the development and the monitoring team in the Small Software firms?	4.3		
Presentation: is the results presented by performing the MOGDW process produced in a readable and useful format?	4.1		
Task support satisfaction			
Item	Mean	Overall mean	Achievement level
Ability to produce expected results: is the MOGDW methodology able to produce expected results?	4.3	4.4	Fully achieved
Completeness (adequate or sufficient): is the MOGDW methodology adequate and sufficient for developing web application in your organization.	4.4		
Ease of implementation: is the process of the MOGDW methodology easy to implement?	4.4		
Perceived usefulness			
Item	Mean	Overall mean	Achievement level
Using MOGDW methodology enables you to accomplish your tasks more quickly.	4.3	4.4	Fully achieved
Using MOGDW methodology improve the performance of your work	4.1		
Using MOGDW methodology makes performing your tasks easier	4.6		
MOGDW methodology is useful to your work	4.3		
Using MOGDW methodology increases your productivity	4.9		
Perceived ease of use			
Item	Mean	Overall mean	Achievement level
Learning the MOGDW methodology is easy for you	4.6	4.5	Fully achieved
Do you find it easy to use MOGDW methodology to do what want to do	4.4		
The MOGDW methodology is flexible to interact with	4.7		
Your interactions with the MOGDW methodology clear and understandable	4.3		
It is easy for you to become skilful in using MOGDW methodology	4.3		
The MOGDW methodology is easy to use	4.7		

Results in Table 6.14 show that four factors gained “fully achieved” level. These factors are: gain satisfaction, task support satisfaction, perceived usefulness and

perceived ease of use. However, the interface satisfaction attained “largely achieved” level. Consequently, it can be concluded that the MOGWD found effective and applicable to be used in Jordanian SSF. Nevertheless, the team claimed that using the MO-PT should reduce the number of GOMM to one member and the number of the goals should be reduced by using the goal prioritizing.

6.3.2 Validation based on the yardstick method

The main objective of the validation process is to demonstrate the correctness of the proposed methodology for its intended purpose, the comparison with existing baseline methods is also considered as a reliable and the perfect way to validate a model, this method called yardstick validation (Carson, 2002). Yardstick approach is used usually with other approaches to increase the trustworthiness of the model or the methodology that was proposed (Sargent, 2011). Specifically, if the model’s components are compared and found that they match with baseline models in the same field, it will be considered as proof to the validity of that model (Carson, 2002; Sargent, 2011).

The yardstick validation was performed through the following steps:

Step 1: Identify the baseline methods. As discussed in Chapter Two, there are six studies taken as a baseline for the comparison as they combined XP and Scrum. These studies are: Mar and Schwaber (2002), Fitzgerald et al. (2006), Clutterbuck et al. (2009), Qureshi (2011), Jyothi and Rao (2011) and Temprado and Bendito (2010).

Step 2: Determine the comparison criteria and compare the baseline methods and MOGWD based on them. The comparison criteria were determined based on the criteria of building a good methodology in Chapter Two. These criteria are: process style (Iterative or sequential), deal with changing requirements and small teams, well defined components (model, process, rules, guidelines, practices and activities), should have the suitable measurement mechanism for monitoring the quality of the development process and the final product and should be built based on a specific theory. Table 6.15 shows the methods used and whether they achieved the criteria.

Table 6.15

Baseline Models and Comparison Criteria.

Models	Mar and Schwaber (2002)	Fitzgerald et al. (2006)	Clutterbuck et al. (2009)	Qureshi (2011)	Jyothi and Rao (2011)	Temprado and Bendito (2010)	MOGWD methodology
Iterative style	√	√	√	√	√	√	√
Well-defined components	<	<	<	<	<	<	√
Deals with design complexity	<	<	<	<	<	<	√
Monitor the quality of process and product	×	×	×	×	×	×	√
Built based on specific theory	×	√	×	×	×	×	√

(√) means satisfy the criterion, (×) means not satisfied the criterion and (<) partially satisfy the criterion.

Based on the above table, it is clearly shown that the MOGWD satisfied all the comparison criteria. However, the baseline method did not satisfy all the criteria except the iterative development style. Consequently, two criteria were partially

stratified by the baseline methods, namely well-defined components and dealing the Web application design complexity as they still use simple design activity process and not define the components of the methods. Moreover, all the baseline methods did not use any monitoring mechanism that measures the quality of process and product. Finally, just the MOGWD methodology and Fitzgerald et al. (2006) method used a specific theory for creating the methods. However, Fitzgerald et al. (2006) did not depend on the Agile principles when they create the methods. Based on this comparison the strength and weaknesses of the baseline methods and MOGWD are determined in the next step.

Step 3: Determine the strength and weaknesses of the baseline methods and MOGWD methodology. As mentioned earlier, the strengths and weaknesses for each method were determined based comparison that conducted in the previous step. Table 6.16 presents the comparison between the MOGWD methodology and the baseline methods based on their strengths and weaknesses.

Table 6.16

Yardstick validation

	Strengths	Weaknesses
MOGWD methodology	<ul style="list-style-type: none"> • Iterative process. • Combine XP and Scrum based on the Agile principles achieved. • Nine XP practices used which are: simple design, collective ownership, pair programming, metaphor, coding standards, TDD, continuous integration, refactoring and small release. • Enhance the design phase of the combined XP and Scrum. • Use measurement mechanisms for monitoring the quality of the process and product. • Activities, methods, practices, tools and team structure are clearly defined and organized using the PDCA method. 	<ul style="list-style-type: none"> • The methodology can be used for Web applications development. • The methodology specified for SSF in Jordan. • Just seven factors used for monitoring the quality of the process.
Mar and Schwaber (2002)	<ul style="list-style-type: none"> • Iterative process. • Combined XP and Scrum based on practices. • Seven practices used in this combination which are: simple design, collective ownership, pair programming, coding standards, TDD, continuous integration and refactoring. 	<ul style="list-style-type: none"> • No theory used in the combination. • Agile principles are not taken into account in the combination process. • Design phase still simple. • No measurement mechanism used. • Just process and practices were discussed in this study • Metaphor and small release are not integrated in the combination. However, these two practices are very important to fulfil the Agile principles.
Fitzgerald et al. (2006)	<ul style="list-style-type: none"> • Iterative process. • Combined XP and Scrum based on practices used method engineering. • Six practices used in this combination which are: simple design, collective ownership, pair programming, coding standards, TDD and refactoring 	<ul style="list-style-type: none"> • Agile principles are not taken into account in the combination process. • Design phase still simple. • No measurement mechanism used. • Just process and practices were discussed in this study. • Metaphor, small release and continuous integration are not integrated

		in the combination. However, these two practices are very important to fulfil the Agile principles.
Clutterbuck et al. (2009)	<ul style="list-style-type: none"> • Iterative process. • Combined XP and Scrum based on practices. • Seven practices used in this combination which are: Simple Design, Collective Ownership, Pair Programming, Coding Standards, TDD, Continuous Integration and Refactoring. 	<ul style="list-style-type: none"> • No theory used in the combination. • Agile principles are not taken into account in the combination process. • Design phase still simple. • No measurement mechanism used. • Just process and practices were discussed in this study. • Metaphor and small release are not integrated in the combination. However, these two practices are very important to fulfil the Agile principles.
Qureshi (2011)	<ul style="list-style-type: none"> • Iterative process. • Combined XP and Scrum based on practices. • Seven practices used in this combination which are: Simple Design, Collective Ownership, Pair Programming, Coding Standards, TDD, Continuous Integration and Refactoring. 	<ul style="list-style-type: none"> • No theory used in the combination. • Agile principles are not taken into account in the combination process. • Design phase still simple. • No measurement mechanism used. • Just process and practices were discussed in this study. • Metaphor and small release are not integrated in the combination. However, these two practices are very important to fulfil the Agile principles.
Jyothi and Rao (2011)	<ul style="list-style-type: none"> • Iterative process. • Combined XP and Scrum based on practice. • Four practices used in this combination which are: collective ownership, pair programming, continuous integration and Refactoring. 	<ul style="list-style-type: none"> • No theory used in the combination. • Agile principles are not taken into account in the combination process. • Design phase still simple. • No measurement mechanism used. • Just process and practices were discussed in this study. • Metaphor, small release, simple design, coding standards and TDD are not integrated in the combination. However, these two practices are very important to fulfil the Agile principles.

Temprado and Bendito (2010)	<ul style="list-style-type: none"> • Iterative process. • Combined XP and Scrum based practice. • Five practices used in this combination which are: pair programming, TDD, onsite customer, coding standards and Refactoring. 	<ul style="list-style-type: none"> • No theory used in the combination. • Agile principles are not taken into account in the combination process. • Design phase still simple. • No measurement mechanism used. • Just process and practices were discussed in this study. • Metaphor, small release, simple design, collective ownership and continuous integration are not integrated in the combination. However, these two practices are very important to fulfil the Agile principles
------------------------------------	---	--

Table 6.16 shows that the MOGWD have less number of weaknesses and maximum number strengths among the baseline methods. However, the baseline methods still have critical weaknesses that may affect the quality of the Web application these weaknesses are: using simple design activity, and they did not any mechanism for monitoring the quality of the process and product. In addition, no base line method takes into account the Agile principles on combining XP and Scrum. Moreover, all the baseline methods focus on the process and practices and neglecting the other components such as tools and team structure, that may affect the completeness of their methods. Finally, nine XP development practices were used in the MOGWD methodology. Where, the number of practices that used in the baseline methods range from 4-7 practices.

6.4 Summary

This chapter presented the evaluation process of MOGWD Methodology for SSF that was conducted through the verification and validation process. The verification process was conducted using expert review method. Eight experts verified the new methodology. Delphi technique was used to describe the verification process through three rounds of modifications in order to improve the methodology. In the final round, the methodology components were modified according to the expert's suggestions. And finally, get the experts' agreement that the modifications fulfilled their suggestions.

For the validation, two approaches were used the case study and yardstick validation. Regarding to the case study, one case study conducted in Jordan was implemented the MOGWD methodology. It was found that MOGWD was effective and applicable to be used in SSF. However, the team recommended to reduce the monitoring team as the methodology proposed the MO-PT.

In the yard stick validation, the MOGWD methodology was compared with the baseline methods in the field to the show the strengths and the weaknesses of the proposed methodology. The comparison conducted using specified criteria. The criteria were taken from the criteria of the good methodology. The output validation shows that the MOGWD methodology satisfies all the comparison criteria and has less number of weaknesses among the baseline methods.

CHAPTER SEVEN

CONCLUSION

7.1 Introduction

This chapter concludes the research findings. It includes an overview of results, research contributions, methodology limitations, and future work.

7.2 Overview of Results

The main goal of this research was to develop a new Web application development methodology that emphasized on monitoring. The research was performed through a theoretical study, survey, methodology development, and methodology evaluation.

These phases are described phases as below:

7.2.1 Theoretical study

This research started with the reviewing literature that related to software engineering, software development, software measurement, Web applications development and development practices for SSF. In this phase, problems that currently faced developers in SSF were highlighted. The findings of this phase were used to formalize the research problem and research objectives as well as gain knowledge on the state of the art that related to Web application development in SSF. This phase involved with five steps: identify the most suitable methods and practices for SSF, determine a suitable measurement method for SSF, define the common activities for designing Web applications, identify the best Web application development and measurement practices for SSF and identify the criteria of good

methodology. The main outcomes of this phase are suitable Agile development methods and practices for SSF, suitable measurement method for SSF, common Web application design steps, best Web application development and measurement practices and criteria of the good development methodology.

7.2.2 Survey

This survey aims to determine the real characteristics of SSF in Jordan. In addition, it is conducted to examine the need of new methodology for developing Web applications in SSF. Moreover, it also investigates the current development and measurement practices of Web application development in SSF. In performing this phase, survey approach was adopted and questionnaire was used to be a data collection instrument.

The results of this phase have been achieved and presented in Chapter Four, section 4.6. In short, the main results are summarized as follows:

7.2.2.1 SSF Characteristics

The majority of SSF in Jordan is private sectors and they have 10 to 30 employees. Consequently, all developers have ten or less than ten years of experience and few managers and team leaders have more than ten years' experience.

7.2.2.2 Development issues

According to the development issues respondents indicate that the majority of them are not using any method for developing Web applications. This means that there is a need for a new methodology to develop Web application for SSF. The most common methods that they are familiar with are Waterfall, XP and Scrum.

In terms of the test, the test type that is normally used by the respondents are unit test, acceptance test, system test and code coverage test and most of the developers perform the testing process at the end of the coding phase of the development.

The most components that had been reused by the SSF in Jordan are source code, templates and modules.

7.2.2.3 Measurement issues

The majority of respondents do not use any measurements during the development process, whereas there is a minimal percentage use line of code measurement type and use GQM as a measurement method after the coding phase, which means there is a lack of using measurements during the development process. Consequently, respondents were asked about why they are not using any specific measurements or method, the majority of them explain that because nobody in the company familiar with measurements type and methods. In addition, they indicate that using a specific measurement and using measurements need specific trained team to be performed.

7.2.2.4 The current Web applications development and measurement practices

The degree of applying the important Web applications development practices was low in case of just three among seventeen practices were applied in the small software in Jordan as well as three are partially applied. This means there is a lack of applying the development and measurement practices inside these companies. Consequently, these practices are related directly to enhance and improve the development process such as requirement, test, quality measurement and management. Therefore, that there is a need to a development methodology that deploys the important practices in order to get a high quality Web application.

7.2.3 Methodology Construction

The aim of this phase is to construct a new monitoring Agile based Web application methodology for SSF. The methodology focused on continuous quality monitoring of process and product. The methodology also handles the XP and Scrum limitations. Based on the theoretical and the survey findings, XP and Scrum were identified as the suitable methods for SSF. Therefore, this study proposed a new methodology to handle the XP and Scrum limitations by extending Scrum method with important XP elements. Then enhance the Extended Agile method by adding Web design method and in corporate GOMM for monitoring process and product. After carrying out these this study adapts the PDCA method to organize the components and guide the development and measurement process.

7.2.4 Methodology Evaluation

The evaluation of the MOGDW was performed in two stages: verification and validation.

7.2.4.1 Verification

The verification phase aims to verify the completeness, understandability, feasibility of the MOGDW methodology components. The verification process was performed using the experts review approach joined with the Delphi technique. After three rounds of reviewing, the results of verification show that MOGDW methodology is acceptable with some modifications and the improved version of MOGDW methodology.

7.2.4.2 Validation

The aim of the validation is to confirm that MOGDW methodology is effective and applicable in SSF. Accordingly, two approaches were used which are: case study and yardstick. Regarding to the case study, one case study in Jordan was chosen. The firm implements the MOGDW methodology for developing CMS. The methodology was validated based on various factors such as gain satisfaction, interface satisfaction, task support satisfaction, perceived usefulness and perceived ease of use. The findings show that MOGDW is applicable and effective for developing and monitoring the quality of the Web application in the real life.

In addition, this study used yardstick validation to increase the reliability of the validation stage. The aim of conducting the yardstick validation is to check

MOGWD methodology validity comparing with other valid methods in the field. Accordingly, six baseline methods have been identified. The comparison criteria were specified based on the good criteria of building new methodology for developing Web application in SSF. These criteria are: process style (Iterative or sequential), deal with changing requirements and small teams, well defined components (model, process, rules, guidelines, practices and activities), should have the suitable measurement mechanism for monitoring the quality of the development process and the final product and should be built based on a specific theory. The results of the yard stick validation showed that the MOGWD has more strengths than previous methods.

7.3 Research contributions

The contributions of this study are MOGWD methodology, Extended Agile method, Web Design method, GOMM, MO-PT software and survey findings.

7.3.1 MOGWD methodology

The main aim of this methodology is to produce high quality Web applications. The MOGWD methodology focuses on continuous quality monitoring of the development process and the product. In doing so, The MOGWD adapts the PDCA method phases to guide the process of management, development and monitoring and organize the methodology components namely, activities, methods, practices, tools and team structure.

In addition, the new MOGWD methodology combined the Extended Agile Method with Web design method and GOMM methods, to cope with the uniqueness and individuality that is specific to Web applications and deals with the small software firm's limitations.

The new MOGWD methodology enhanced the design phase of the Extended Agile method by adding Web design method to deal with the Web application design complexity. Furthermore, it enhanced the Agile development practices by adding the Web design practices and the measurement practices.

This methodology has a measurement mechanism that includes quantitative and qualitative metrics to monitor the quality of process and product.

7.3.2 Extended Agile method with Web design method

This method aims to overcome the XP and Scrum limitations. Based on the discussion in Chapter Two this method was created by extending the Scrum method with XP important elements. These elements are: XP iteration activities, XP core and supported practices and XP iteration team. The XP iteration activities were adapted from XP as these activities should be performed during 2 weeks of time that may accelerate the process. Whereas, the core and supported XP practice were merged to the Extended Agile method in order to ensure the application of the Agile principles. Finally, at least two programmers and one tester were used to perform the iteration activities. However, the Extended Agile method still has limitations. One of the

limitations is the simple design phase that was covered by adding Web design method.

The simple Web design method merged to the Extended Agile method to meet the complexity of Web application design. This method produced a better design phase of the new methodology. This method integrated to the planning phase of the new methodology and used once per Web application. The simple design method created based on the existing Web design methods. The steps of the method are: requirements analysis, conceptual design, navigational design, implementation design (interface) and construction.

7.3.3 GOMM

To ensure the quality of process and product, several metrics were integrated into the Extended Agile method to reduce defects, time and rework of the development life cycle. The measurement mechanism was performed by using the GOMM. This method used a set of goals, questions and quantitative and qualitative metrics that monitor the quality of process and product. The GOMM monitors the process quality and the Web application product quality. Twenty quantitative and qualitative goals identified for monitoring the quality of the process. Ten quantitative goals related to the process activities, development and management practices and staff productivity. Ten qualitative goals identified for monitoring the process quality factors that related to process effectiveness, adaptability, compatibility, accessibility, applicability, changeability and supportability. Whereas, Monitoring the Web application product is done by achieving three quantitative goals: cost, quality and time.

7.3.4 Survey results

The main goal of this survey is to investigate the current development and measurement practices for SSF in Jordan.

The survey offers a view on the development and measurement practices in SSF, particularly in Jordan. Therefore, this research is useful as it extracts and ranks the most important practices that affect development and measurement process in SSF. This research is useful and beneficial to other researchers. Researchers will find this study useful for its contribution in literature and survey findings related to Web applications in SSF.

7.4 Limitations of the Research

Despite the achievement, this study has some limitations. Among these are:

7.4.1 Lack of the Related Researches

There is a lack of researches that combine the Agile development methods to GQM. Therefore, it was a challenge to combine Agile development with GQM. In this regard, many related publications on Agile and software measurement were utilized in this study in order to carry out this combination. In addition, the related studies did not show how XP and Scrum methods can be extended and integrated with GQM. Therefore, it was difficult to search for literature on extending the combined XP and Scrum method to cover all the Agile principles and monitor the quality of product and process for SSF, as these firms need to have lightweight processes in their development processes. Due to these obstacles, some quality factors are not

included in the new MOGWD methodology because it may take time to deploy. Therefore, future research can be continued to address the missing specific factors.

7.4.2 Limited Scope in the Evaluation Processes

During the verification process, the expert review comprising of eight members was performed. The experts include four knowledge experts (two from Malaysia and two from Jordan) and four domain experts (one from Malaysia and three from Jordan). Accordingly, the verification and validation process was carried out based on the characteristics of a limited number of Jordanian SSF. In future, the verification and validation can be made more extensive by including SSF from other countries in order to assess the comprehensiveness of the research results.

7.5 Future Work

The MOGWD methodology presented in this study is the starting point for working towards collaboration between Agile methods and GQM. During the course of the research, several potential directions for future investigation were identified. Some of these are meant to address the current limitations of this study. Sections 7.5.1, 7.5.2 and 7.5.3 highlight the potential directions for future work.

7.5.1 Add more quality factors for the process

The MOGWD methodology supports the specific goals related to seven process quality factors. These factors are effectiveness, adaptability, compatibility, accessibility, applicability, changeability and supportability. Therefore, future

research can incorporate other factors based on the organization recommendations. The Web application development and measurement in this methodology can be used as a guideline for developing better applications. On the other hand the GOMM that used in the MOGWD is mainly focused on the process and product. Therefore, future research can incorporate suitable measurement for the people and technology. Moreover, the measurement process should be refined by referring to the existing standard for software process assessment such as ISO 15504.

7.5.2 Using other Agile Practices or methods for the Extended Agile method

The construction of the MOGWD methodology was based on the Extended Agile Method and GOMM. In this regard, there is possible avenue for further research to examine the use of other Agile methods such DSDM, LSD, AM and AUP (RUP). These are all effective methods that could be used by SSDFs and may be extended to large organizations. Therefore, the combination of some new Agile methods will offer better development practices that are suitable for large organizations.

7.5.3 Extend the MOGWD to include other Key process areas

Currently, the MOGWD methodology focused on development and monitoring Web applications practices in SSF. The practices covered by XP, Scrum and Web design method. These practices should be analyzed in order to improve the maturity of the methodology by including other key process areas. Therefore, it will be fruitful if the future research can identify what maturity level does this methodology achieve and improve it by adding the identified Key process areas.

7.6 Summary

This research started from the need to have a suitable methodology for development and measurement in SSF. These firms suffer from problems during the development of their products. This is because Web application products were developed in a chaotic manner. A more suitable methodology integrating the Agile methods with an appropriate measurement method was developed to address the SSF' needs. The thesis reports on the development of the new methodology.



REFERENCES

- Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods. *Relatório Técnico, Finlândia*.
- Abran, A., Moore, J. W., Dupuis, R., Dupuis, R., & Tripp, L. L. (2004). *Guide to the Software Engineering Body of Knowledge–SWEBOK, trial version*. IEEE-Computer Society Press.
- Abrantes, J. F., & Travassos, G. H. (2011). Common Agile Practices in Software Processes. In *Proceedings of the Fifth International Symposium on Empirical Software Engineering and Measurement (ESEM)*, (pp. 355–358).
- Ahmad, F. (2008). Presage, context, process and product: Influencing variables in literature instruction in an ESL Context. *GEMA Online Journal of Language Studies*, 8(1), 1-21.
- Ahmad, M., Yousef al-Tarawneh, M., & Bashah Mat Ali, A. (2012). Software process improvement in small software development firms. *Global Journal on Technology*, 1.
- Ahmad, R., Li, Z., & Azam, F. (2005, September). Web engineering: a new emerging discipline. In *Emerging Technologies, 2005*. In *Proceedings of the IEEE Symposium on Emerging Technologies* (pp. 445-450). IEEE.
- Aleksy, M., Gitzel, R., & Schwind, M. (2004). Developing Web Applications for Small and Medium-sized Enterprises-An Experience Report. In *GI Jahrestagung (1)* (pp. 282-286).
- Alexandre, S., Renault, A., & Habra, N. (2006, August). OWPL: a gradual approach for software process improvement in SMEs. In *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA'06)* (pp. 328-335). IEEE.
- Ali, R., Iqbal, S., Shahzad, S., Qadeer, M. Z., & Khan, U. A. (2011). Use Of Reinforcement Practices in the Educational Institutions and its Impacts on Student Motivation. *International Journal Of Academic Research*, 3(1), 960-963.
- Allen, P., Ramachandran, M., & Abushama, H. (2003, November). PRISMS: an approach to software process improvement for small to medium enterprises. In *Proceedings of the Third International Conference On Quality Software*

(QSIC'03) (pp. 211-214). IEEE.

- Altarawneh, H., & El Shiekh, A. (2008, August). A theoretical agile process framework for web applications development in small software firms. In *Proceedings of the Sixth International Conference on Software Engineering Research Management and Applications, SERA'08* (pp. 125-132). IEEE.
- Al-Tarawneh, M. Y. (2013). *Harmonizing CMMI-DEV 1.2 and XP Method to Improve The Software Development Processes in Small Software Development Firms* (Doctoral dissertation, Universiti Utara Malaysia).
- al-Tarawneh, M. Y., Abdullah, M. S., & Ali, A. B. M. (2011). A proposed methodology for establishing software process development improvement for small software development firms. *Procedia Computer Science*, 3, 893-897.
- Anacleto, A., von Wangenheim, C., Salviano, C., & Savi, R. (2004). *A method for process assessment in small software companies*. Paper presented at the 4th international SPICE conference on process assessment and improvement, Portugal, 2004.
- Androcec, D., & Dobrovic, Z. (2012, June 25-28). Creating hybrid software engineering methods by means of metamodels. In *Proceedings of 34th, International Conference on Information Technology Interfaces (ITI)*, Cavat, Croatia.
- Ardimento, P., Baldassarre, M. T., Caivano, D., & Visaggio, G. (2004). Multiview framework for goal oriented measurement plan design. In *Proceeding of 5th international conference on the Product Focused Software Process Improvement (PROFES 2004)*, (pp. 159-173), Kansai, Japan
- Avison, D. E., Lau, F., Myers, M. D., & Nielsen, P. A. (1999). Action research. *Communications of the ACM*, 42(1), 94-97.
- Awad, M. A. (2005). *A comparison between agile and traditional software development methodologies*. Honours program thesis, University of Western Australia.
- Azuma, M., & Mole, D. (1994). Software management practice and metrics in the european community and japan: Some results of a survey. *Journal of Systems and Software*, 26(1), 5-18.
- Baharom, F. (2008). A software certification model based on development process quality assessment. Unpublished doctoral dissertation, Universiti Kebangsaan Malaysia.

- Baharom, F., Deraman, A., & Hamdan, A. (2006). A survey on the current practices of software development process in Malaysia. *Journal of ICT, 4*, 57-76.
- Baharom, F., Yahaya, J., Deraman, A., & Hamdan, A. R. (2011, July). SPQF: Software Process Quality Factor. In *Proceedings of Electrical Engineering and Informatics (ICEEI)*, (pp. 1-7). IEEE. Bandung, Indonesia
- Balasubramanian, V., Bieber, M., & Lsakowitz, T. (1996). Systematic hypermedia design. *Information Systems Working Papers Series*, Stern School of Business, NYU.
- Barna, P., Frasinca, F., Houben, G. J., & Vdovjak, R. (2003, April). Methodologies for web information system design. In *Proceedings of the International Conference on Information Technology: Computers and Communications (ITCC 2003)* (pp. 420–424). Las Vegas, NV, USA. IEEE Computer Society
- Barnett L., & Schwaber C.E. (2004). Adopting Agile Development Processes: *Improve Time-To-Benefits for Software Projects*. Trends, Forrester Research, March 2004.
- Barry, C. & Lang, M. (2001) A Survey of Multimedia and Web Development Techniques and Methodology Usage. *IEEE Multimedia*, 8 (2), 52-60.
- Basili, V. R. (1992). Software modeling and measurement: the Goal/Question/Metric paradigm. Tech. Rep. CS-TR- 2956, Department of Computer Science, University of Maryland, College Park, MD 20742, Sept. 1992.
- Basili, V. R., & Selby, R. W. (1987). Comparing the effectiveness of software testing strategies., *IEEE Transactions on Software Engineering*, 13(12), 1278-1296.
- Basili, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*, 1(4), 390-396.
- Basili, V., Caldiera, G., & Rombach, H. D. (1994). *Encyclopedia of Software Engineering*, chap. Goal Question Metric Approach, (pp. 528{532). John Wiley & Sons, Inc.
- Baskerville, R., & Pries-Heje, J. (2002). Information Systems Development@ Internet Speed: A New Paradigm In The Making!. *Proceedings of the European Conference on Information Systems (ECIS 2002)*, (pp. 282-291). Wrycza, Gdansk, University of Gdansk

- Beck, K. (1999). Embracing change with extreme programming. *Computer*, 32(10), 70-77.
- Beck, K. (2000). *Extreme programming explained: embrace change*. Addison-Wesley Professional.
- Begel, A., & Nagappan, N. (2008, October). Pair programming: what's in it for me?. In *Proceedings of the second international symposium on empirical software engineering and measurement, (ESEM)* (pp 120–128). ACM, Kaiserslautern, Germany
- Behkamal, B., Kahani, M., & Akbari, M. K. (2009). Customizing ISO 9126 quality model for evaluation of B2B applications. *Information and Software Technology*, 51(3), 599-609.
- Bell, D. (2001). *Software engineering: A programming approach*. Prentice Hall International (UK) Ltd..
- Bellettini, C., Marchetto, A., & Trentini, A. (2004, March). WebUml: reverse engineering of web applications. In *Proceedings of the 2004 ACM symposium on Applied computing*, (pp. 1662–1669). New York, NY, USA: ACM
- Berardi, E., & Santillo, L. (2010). COSMIC-based Project Management in Agile Software Development and Mapping onto related CMMI-DEV Process Areas. In *Proceedings of International Workshop on Software Measurement–IWSM*, Stuttgart (Germany),
- Bertoa, M. F., Troya, J. M., & Vallecillo, A. (2006). Measuring the usability of software components. *Journal of Systems and Software*, 79(3), 427-439.
- Bidad, C. D., & Campiseño, E. R. (2010). Community Extension Services OF SUCs IN REGION IX: Basis For A Sustainable Community Enhancement Program. *E-International Scientific International Scientific International Scientific Research Journal Research Journal*, 235-243.
- Blackburn, J. D., Scudder, G. D., & Van Wassenhove, L. N. (1996). Improving speed and productivity of software development: a global survey of software developers. *IEEE Transactions on Software Engineering*, 22(12), 875-885.
- Blaxter, L., Hughes, C., & Tight, M. (2010). *How to research*. McGraw-Hill International.
- Bocij, P., Chaffey, D., Greasley, A., & Hickie, S. (1999). *Business Information Systems. Technology, Development and Management*.
- Boehm, B. (2006, May). A view of 20th and 21st century software engineering.

In Proceedings *28th International Conference on Software Engineering (ICSE)* pp. 12–29. Shanghai, China.

- Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5), 61-72.
- Bourque, P., Dupuis, R., Abran, A., Moore, J. W., & Tripp, L. (1999). The guide to the software engineering body of knowledge. *Software, IEEE*, 16(6), 35-44.
- Briand, L. C., Differding, C. M., & Rombach, H. D. (1996). Practical guidelines for measurement-based process improvement. *Software Process Improvement and Practice*, 2(4), 253-280.
- Briand, L. C., Morasca, S., & Basili, V. R. (2002). An operational process for goal-driven definition of measures., *IEEE Transactions on Software Engineering*, 28(12), 1106-1125.
- Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4), 275-280.
- Brown, N. (1999). High-Leverage Best Practices: What Hot Companies Are Doing to Stay Ahead. *Cutter IT Journal*, 12(9), 4-9.
- Bryman, A. (2001). *Social research methods*. Oxford: Oxford University Press.
- Bryman, A., & Bell, E. (2007). *Business research methods*. USA: Oxford University Press.
- Bucci, G., Campanai, M., & Cignoni, G. A. (2001). Rapid Assessment to Solicit Process Improvement in Small and Medium-Sized Organizations. *Software Quality Professional Magazine*, 4(1), 33-41.
- Caldiera, V. R. B. G., & Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of Software Engineering*, 2(1994), 528-532.
- Calero, C., Ruiz, J., & Piattini, M. (2005). Classifying web metrics using the web quality model. *Online Information Review*, 29(3), 227-248.
- Cao, L., & Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *Software, IEEE*, 25(1), 60-67.
- Carson, J. S., II. (2002, Dec). Model verification and validation. *In Proceeding of the 2002 Winter Simulation Conference*, (pp. 52-58 Vol.1). USA: IEEE Computer Society.

- Cater-Steel, A. (2004). *An evaluation of software development practice and assessment-based process improvement in small software development firms* (Doctoral dissertation, Griffith University).
- Cater-Steel, A. P. (2001). Process improvement in four small software companies. In *Proceedings of the 13th Australian Software Engineering Conference (ASWEC'01)* (pp. 262-272). IEEE.
- Ceri, S., Fraternali, P., & Bongio, A. (2000). Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks*, 33(1), 137-157.
- Chatfeild, c. and Collins, A.J. (1990). *Introduction to Multivariate Analysis*. Chapman and Hall publication. London.
- Cho, J. (2009). A hybrid software development method for large-scale projects: rational unified process with scrum. *Issues in Information Systems*, 10(2), 340–348.
- Clutterbuck, P., Rowlands, T., & Seamons, O. (2009). A case study of SME web application development effectiveness via Agile methods. *Electronic Journal Information Systems Evaluation Volume*, 12(1), 13-26.
- Conallen, J. (1999). Modeling Web application architectures with UML. *Communications of the ACM*, 42(10), 63-70.
- Cook, Dave and Les Dupaix.(1998). *Life Cycle Reviews from a Software Engineering Perspective*, Presented at the 1998 Software Technology Conference, May 1998, Salt Lake City, Utah.
- Costagliola, G., Ferrucci, F., & Francese, R. (2002). Web engineering: Models and methodologies for the design of hypermedia applications. *Handbook of Software Engineering & Knowledge Engineering*, 2, 181–199.
- Creswell, J. W. (2003) *Research Design: Qualitative, Quantitative and Mixed Methods Approaches*, 2nd Ed., sage publication, California.
- da Rocha, A. R. C., Montoni, M., Weber, K. C., & de Araújo, E. E. R. (2007, September). A Nationwide Program for Software Process Improvement in Brazil. In *Proceedings of the 6th International Conference on the Quality of Information and Communications Technology (QUATIC'2007)* (pp. 449-460). Lisbon New University, Lisbon, Portugal.
- Dangle, K., Larsen, P., Shaw, M., & Zelkowitz, M. (2005). Software process improvement in small organizations: A case study. *IEEE Software*, 22(6), 68-75.

- Daskalantonakis, M. K. (1992). A practical view of software measurement and implementation experiences within Motorola. *IEEE Transactions on Software Engineering*, 18(11), 998-1010.
- Daskalantonakis, Michael K. A practical view of software measurement and implementation experiences within Motorola. *IEEE Transactions on Software Engineering*, 18(11), 998-1010
- Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, 13(3), 319-340.
- de Cesare, S., Iacovelli, N., Merico, A., Patel, C., & Lycett, M. (2004). Tailoring software development methodologies in practice: A case study. *Journal of Computing and Information Technology*, 16(3), 157-168.
- De Troyer, O. M. F., & Leune, C. J. (1998). WSDM: a user centered design method for Web sites. *Computer Networks and ISDN systems*, 30(1), 85-94.
- DeGrace, P., & Stahl, L. (1990). Wicked problems, righteous solutions: a catalogue of modern software engineering paradigms: Prentice Hall.
- Deshpande, Y., Murugesan, S., Ginige, A., Hansen, S., Schwabe, D., Gaedke, M., White, B. (2002), WEB ENGINEERING, *Journal of Web Engineering*, 1(1), 003-017.
- Diaz-Ley, M., Garcia, F., & Piattini, M. (2008). Implementing a software measurement program in small and medium enterprises: a suitable framework. *IET Software*, 2(5), 417-436.
- Dingsøyr, T., Nerur, S., Balijepally, V. G., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(2012), 1213-1221.
- Distante, D., Rossi, G., Canfora, G., & Tilley, S. (2007). A comprehensive design model for integrating business processes in Web applications. *International Journal of Web Engineering and Technology*, 3(1), 43-72.
- Dumke, R. R., & Foltin, E. (1998, March). Metrics-based evaluation of object-oriented software development methods. In *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering (CSMR '98)* (pp. 193-196). Florence, Italy
- Dutta, S., Van Wassenhove, L. N., & Kulandaiswamy, S. (1998). Benchmarking European software management practices. *Communications of the ACM*, 41(6), 77-86.

- Dyba, T. (2000). An instrument for measuring the key factors of success in software process improvement. *Empirical software engineering*, 5(4), 357-390.
- Easterbrook, S., Singer, J., Storey, M. A., & Damian, D. (2008). Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering* (pp. 285-311). Springer London.
- El Emam, K., & Madhavji, N. H. (1995, March). A field study of requirements engineering practices in information systems development. In *Proceedings of 2nd international symposium on requirements engineering* (pp.68-80). York, England, IEEE CS Press,
- El Sheikh, A., & Tarawneh, H. (2007, September). A survey of web engineering practice in small Jordanian web development firms. In *proceedings of the 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers* (pp. 481-489). ACM.
- Eldai, O. I., Ali, A. H. M. H., & Raviraja, S. (2008). Towards a new methodology for developing web-based systems. *World Academy of Science and Technology*, 46(2008), 190-195.
- Esaki, K., Ichinose, Y., & Yamada, S. (2012). Statistical Analysis of Process Monitoring Data for Software Process Improvement and Its Application. *American Journal of Operations Research*, 2(1), 43-50.
- ESI (1997) *Software Best Practice Questionnaire: Analysis of Results*, European Software Institute, Bilbao.
- Fayad, M., Laitinen, M., & Ward, R. (2000). Thinking objectively: software engineering in the small. *Communications of the ACM*, 43(3), 115-118.
- Feiler, P. H., & Humphrey, W. S. (1993, February). Software process development and enactment: Concepts and definitions. In *Proceedings of the 2nd International Conference on the Software Process* (pp. 28-40)., Berlin, IEEE Computer Society Press, Los Alamitos, CA,
- Fernandes, J. M., & Almeida, M. (2010). *Classification and Comparison of Agile Methods*. Paper presented at the 7th International Conference on the Quality of Information and Communications Technology, ICQICT.391-369
- Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems*, 15(2), 200-213.
- Fitzgerald, B., Russo, N. L., & O'Kane, T. (2003). Software development method tailoring at Motorola. *Communications of the ACM*, 46(4), 64-70.

- Floyd, C. (1984). A systematic look at prototyping. In *Approaches to prototyping* (pp. 1-18). Springer Berlin Heidelberg.
- Fraternali, P. (1999). Tools and approaches for developing data-intensive Web applications: a survey. *ACM Computing Surveys (CSUR)*, 31(3), 227-263.
- Fritzsche, M., & Keil, P. (2007). Agile methods and CMMI: compatibility or conflict?. *e-Infomatica Software Engineering Journal*, 1(1), 9-26.
- Gaedke, M., & Gräf, G. (2001, May). Development and evolution of web-applications using the web composition process model. In *proceedings of the 9th International World Wide Web Conference on Web Engineering*, Amsterdam, the Netherlands.
- Garzotto, F., Paolini, P., & Schwabe, D. (1991, September). HDM—a model for the design of hypertext applications. In *Proceedings of the third annual ACM conference on Hypertext* (pp. 313-328). ACM.
- Genero, M., Poels, G., & Piattini, M. (2008). Defining and validating metrics for assessing the understandability of entity–relationship diagrams. *Data & Knowledge Engineering*, 64(3), 534-557.
- Gilb, T., & Graham, D. (1995). Software inspection. *ACM SIGSOFT Software Engineering Notes*, 20(5), 90.
- Ginige, A., & Murugesan, S. (2001). Web engineering: An introduction. *IEEE MultiMedia*, 8(1), 14-18.
- Govers, C. P. M. (1996). What and how about quality function deployment (QFD). *International Journal of Production Economics*, 46(1996), 575-585.
- Graf, K (2005). Addressing Challenges in Application Security, *A WatchFire whitepaper*, Retrieved august 14, 2010, from <http://www-01.ibm.com/software/rational/offerings/websecurity>.
- Greenfield, J., & Short, K. (2003, October). Software factories: assembling applications with patterns, models, frameworks and tools. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (pp. 16-27). ACM.
- Grobbelaar, S. S. (2007). *R&D in the National system of innovation: A system dynamics model* (Doctoral dissertation, University of Pretoria).
- Guceglioglu, A. S., & Demirsors, O. (2011, July). The application of a new process quality measurement model for software process improvement initiatives. In *proceedings of the 11th International Conference on Quality Software*, 2011, (pp. 112-120).

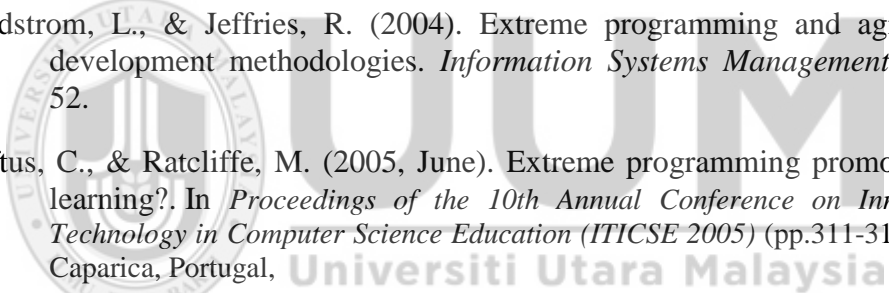
- Hallowell, M. R., & Gambatese, J. A. (2010). Qualitative research: application of the Delphi method to CEM research. *Journal of construction engineering and management*, 136(1), 99-107.
- Hassan, A. E., & Holt, R. C. (2003, November). Migrating web frameworks using water transformations. In *Proceedings of COMPSAC 2003: International Computer Software and Application Conference*, Dallas, Texas, USA, Nov. 2003.
- Henderson-Sellers, B. (1995). Who needs an object-oriented methodology anyway?. *Journal of Object Oriented Programming*, 8(6), 6-8.
- Hjerling, J., & Ljungqvist, P. (2004). Capability Maturity Model Integration (CMMI) and Agile Methods. *A course Paper, unpublished*.
- Hofer, C. (2002). Software development in Austria: results of an empirical study among small and very small enterprises. In *Proceedings of the 28th Euromicro Conference (EUROMICRO'02)* (pp. 361–366). IEEE Computer Society,
- Howcroft, D., & Carroll, J. (2000, July). In *Proceedings of the Eighth European Conference on Information Systems*, Vienna.
- Huang, W., Li, R., Maple, C., Yang, H., Foskett, D., & Cleaver, V. (2008, August). Web Application Development Lifecycle for Small Medium-Sized Enterprises (SMEs)(Short Paper). In *Proceeding of the Eighth International Conference on the Quality Software (QSIC'08)* (pp. 247-252). Oxford, UK IEEE Computer Society
- Humphrey W.,(1995) A Discipline for Software Engineering, Addison-Wesley, England, 1995.
- Humphrey, W.S.(2000). The Personal Software Process (PSP), Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-2000-TR- 022, ESC-TR-2000-022, December 2000.
- Iivari, J. (1989). A methodology for IS development as organizational change: A pragmatic contingency approach. *Systems Development for Human Progress, North-Holland, Amsterdam*, 197-217.
- Imreh, R., & Raisinghani, M. S. (2011). Impact of Agile Software Development on Quality within Information Technology Organizations. *Journal of Emerging Trends in Computing and Information Sciences*, 2(10). 460-475.
- Isakowitz, T., Stohr, E., & Balasubramanian, P. (1995). RMM: a methodology for structured hypermedia design. *Communications of the ACM*, 38(8), 34-44.
- Itkonen,J., Kristian ,R., and Lassenius,C.,(2005) *Towards Understanding Quality Assurance in Agile Software Development*, Paper presented at the

International Conference on Agility Management (ICAM 2005), Helsinki, July 2005.

- Jani, H. M. (2011, October). Intellectual capacity building in higher education: Quality assurance and management. In *Proceeding 5th International Conference on of the Information Science and Service Science (NISS), 2011 New Trends in* (pp. 361-366). IEEE.
- Javdani, T., Zulzalil, H., Ghani, A. A. A., Sultan, A. B. M., & Parizi, R. M. (2012). On the Current Measurement Practices in Agile Software Development. *IJCSI International Journal of Computer Science Issues*, 9(4), 127-133.
- Jeffries, R., Anderson, A., & Hendrickson, C. (2000). *Extreme programming installed*: Addison-Wesley Longman Publishing Co., Second Edition. Prentice Hall Inc. Boston, MA, USA.
- Jiang, L., & Eberlein, A. (2008, May). Towards a framework for understanding the relationships between classical software engineering and agile methodologies. In *Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral* (pp. 9-14). ACM.
- Johnson, R. A., & Wichern, D. W. (1992). *Applied multivariate statistical analysis* (Vol. 4). Englewood Cliffs, NJ: Prentice hall.
- Jones, C. (2003). Implementing a Successful Measurement. *IT Metrics and Benchmarking: Part II*, 16(11), 12-18.
- Jun, L., Qiuzhen, W., & Lin, G. (2010, December). Application of agile requirement engineering in modest-sized information systems development. In *Proceedings of the Second World Congress on Software Engineering (WCSE), 2010* (Vol. 2, pp. 207-210). IEEE.
- Jyothi, V. E., & Rao, K. N. (2011). Effective Implementation of Agile Practices. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 2(3), 41-48.
- Kaner, C. (2004). Software engineering metrics: What do they measure and how do we know?. *Software Engineering Metric*, 8(5), 1-6.
- Kao, Y. W., Lin, C. F., Cheng, K. Y., Yuan, S. M., & Tsai, C. T. (2010, October). A PCDA-based critical exception management system in semiconductor industry. In *proceeding of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, (pp. 417-420). IEEE.
- Kassou, M., & Kjiri, L. (2012). A Goal Question Metric Approach for Evaluating Security in a Service Oriented Architecture Context. *IJCSI International Journal of Computer Science Issues*, 9(4), 238- 249.

- Kaur, G., & Bahl, K. (2014). Software Reliability, Metrics, Reliability Improvement Using Agile Process. *IJISET*, 1(3), 143-145.
- Kautz, K., & Nørbjerg, J. (2003, June). Persistent problems in information systems development. The case of the world wide web. In *Proceedings of 11th European Conference on Information Systems (ECIS)* (pp. 919-926). Naples, Italy, June 16-21,
- Kettelerij, R. (2006). *Designing A Measurement Programme For Software Development Projects*. Master thesis, Universtiet van Amsterdam, Amsterdam.
- Kim, J. A., Choi, S. Y., & Jung, R. (2007). Process improvement with process management agent. *International Journal of Software Engineering and Its Applications*, 1(1). 37-52.
- Kirakowski, J. (2000). Questionnaires in usability engineering .www.ucc.ie/hfrg/resources/qfaq1.html (accessed Jul 2010).
- Kirk, D., & Tempero, E. (2012, December). *Software Development Practices in New Zealand*. In *Proceedings of the Nineteenth Asia-Pacific Software Engineering Conference (APSEC 2012)*, Hong Kong.
- Kitchenham, B. A., & Mendes, E. (2004, May). A comparison of cross-company and within-company effort estimation models for web applications. In *proceedings of the 8th International Conference on Empirical Assessment in Software Engineering* (pp. 47-55). Edinburgh, Scotland, UK
- Kitchenham, B. A., & Pickard, L. M. (1998). *SIGSOFT Softw. Eng. Notes*, 23(1), 24-26.
- Kitchenham, B., Linkman, S., & Law, D. (1997). DESMET: a methodology for evaluating software engineering methods and tools. *Computing & Control Engineering Journal*, 8(3), 120-126.
- Knauber, P., Muthig, D., Schmid, K., & Widen, T. (2000). Applying product line concepts in small and medium-sized companies. *IEEE Software*, 17(5), 88-95.
- Knight, A., & Dai, N. (2002). Objects and the Web. *IEEE software*, 19(2), 51-59.
- Koblenz, C.(2003) *Maintenance Activities in Software Process Models: Theory and Case Study Practice*. University of Koblenz Landau Campus Koblenz., Mastre thesis.
- KOCH N.(2001). *Software Engineering for Adaptive Hypermedia Applications*, PhD. Thesis, Reihe Softwaretechnik 12, Uni-Druck Publishing Company, Munich.

- Koch, N. (1999). A comparative study of methods for hypermedia development. *Ludwig-Maximilians-University Munich, Institute of Computer Science*.
- Koch, N., & Kraus, A. (2002, June). The expressive power of uml-based web engineering. In *proceedings of the Second International Workshop on Web-oriented Software Technology (IWWOST02)*.
- Kolski, C. (1998). A “call for answers” around the proposition of an HCI-enriched model. *ACM SIGSOFT Software Engineering Notes*, 23(3), 93-96.
- Kroeger, T. A. (2011). *Understanding the Characteristics of Quality for Software Engineering Processes*. Doctor of Philosophy, University of South Australia.
- Kroeger, T. A., Davidson, N. J., & Cook, S. C. (2014). Understanding the characteristics of quality for software engineering processes: A Grounded Theory investigation. *Information and Software Technology*, 56(2), 252-271.
- KRUEGER, C. W. (1992). Software Reuse. *ACM Computing Surveys*, 24(2), 132-183.
- Kulas, H. (2012). *Product Metrics in Agile Software Development* (Doctoral dissertation, Master's Thesis, Univ. of Tampere, Finland).
- Kumar, G., & Bhatia, P. K. (2012). Impact of Agile Methodology on Software Development Process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, 2(4), 46-50.
- Kunda, D. (2002). *A social-technical approach to selecting software supporting COTS-Based Systems*, Unpublished Doctoral Thesis, Department of Computer Science, University of York, York, UK.
- Kunwar, S. (2013). *Metamodelling and Evaluating Extreme Programming*. Master Thesis, University of Tampere. Finland.
- Lang, M. (2002, June). Hypermedia systems development: do we really need new methods. In *Proceedings of the Informing Science and IT Education Conference*, (pp. 883-891). *Cork, Ireland*
- Laporte, C. Y., Renault, A., Desharnais, J. M., Habra, N., Abou El Fattah, M., & Bamba, J. C. (2005, May). Initiating software process improvement in small enterprises: Experiment with micro-evaluation framework. In *Proceedings of International Conference on Software Development*, (pp. 153-163). University of Iceland, Reykjavik, Iceland, May 27-June 1, 2005,
- Larman C. (2003). *Agile and iterative development: a manager's guide*. Addison Wesley.
- Lassenius, C.(2008). Software Process Improvement. <http://www.soberit.hut.fi/T->

- Lee, H., Lee, C., & Yoo, C. (1998, January). A scenario-based object-oriented methodology for developing hypermedia information systems. In *Proceedings of the 31st Annual Conference on Systems Science* (pp. 47-56). IEEE.
- Leveson, N. G., Heimdahl, M. P. E., Hildreth, H., & Reese, J. D. (1994). Requirements specification for process-control systems. *IEEE Transactions on Software Engineering*, 20(9), 684-707
- Li, J., Moe, N. B., & Dybå, T. (2010, September). Transition from a plan-driven process to scrum: a longitudinal case study on software quality. In *Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement* (p. 13). ACM.
- Lilburne, B., Devkota, P., & Khan, K. M. (2004). *Measuring quality metrics for web applications*. Paper presented at the Innovations Through Information Technology: 2004 Information Resources Management Association International Conference, New Orleans, Louisiana, USA, May 23-26, 2004.
- Lindstrom, L., & Jeffries, R. (2004). Extreme programming and agile software development methodologies. *Information Systems Management*, 21(3), 41-52.
- Loftus, C., & Ratcliffe, M. (2005, June). Extreme programming promotes extreme learning?. In *Proceedings of the 10th Annual Conference on Innovation and Technology in Computer Science Education (ITICSE 2005)* (pp.311-315). Monte de Caparica, Portugal,  Universiti Utara Malaysia
- Lowe, D., & Henderson-Sellers, B. (2001). Characteristics of web development processes. In *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR' 2001)*.
- Lyytinen, K., & Rose, G. (2005). How Agile is Agile Enough? Toward a Theory of Agility in Software Development. *Business Agility and Information Technology Diffusion*, 203-225.
- Mar, K., & Schwaber, K. (2002). Scrum with XP. *Informat. com*.
- Marinelarena, V. K. I. (2014). *Agile Methodologies and Software Process Improvement Maturity Models, Current State of Practice in Small and Medium Enterprises*. Master thesis. Blekinge Institute of Technology. Sweden .
- McCaffery, F, Wilkie, FG, McFall, D & Lester, N (2004), Northern Ireland software industry survey, in *Proceedings of 4th International SPICE Conference on*

- Process Assessment and Improvement* (pp. 159-61). Lisbon, Portugal,
- McCarthy, J. (1995). *Dynamics of software development* (Vol. 3). Redmond, Washington: Microsoft Press.
- Mccarthy, R. V., & Aronson, J. E. (2001). Activating consumer response: a model for web site design strategy. *Journal of Computer information systems*, 41(2), 2-8.
- McCurley, J., Zubrow, D., Dekkers, C.(2008). Measures and Measurement for Secure Software Development. *Build security in*. Retrieved august 14, 2010, from <https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/measurement/227-BSI.html>
- McDonald, A., & Welland, R. (2001, May). Web engineering in practice. In *Proceedings of the fourth WWW10 Workshop on Web Engineering* (pp. 21-30).
- McDonald.A and Welland. R., (2001b), 'A Survey of Web Engineering in Practice', *Department of Computing Science Technical Report R-2001-79, University of Glasgow, Scotland*, 1 March 2001.
- Mendes, E., Mosley, N., & Counsell, S. (2003, April). Investigating early web size measures for web cost estimation. In *Proceedings of EASE'2003 Conference*, Keele, April, 2003, (pp 1-22).
- Moniruzzaman, A. B. M., & Hossain, D. S. A. (2013). Comparative Study on Agile software development methodologies. *Global Journal of Computer Science and Technology Software & Data Engineering*. 13(7).4-18.
- Montero, S., Díaz, P., & Aedo, I. (2003, February). A Framework for the Analysis and Comparison of Hypermedia Design Methods. In: *Proceedings of The IASTED International Conference on Software Engineering (SE'2003)*, (pp. 1053-1058).
- Moody, D. L., Sindre, G., Brasethvik, T., & Sølvsberg, A. (2003, May). Evaluating the quality of information models: empirical testing of a conceptual model quality framework. In *Proceedings of the 25th International Conference of Software Engineering*, (pp. 295-305). Oregon State University, Portland, Oregon USA: IEEE Computer Society.
- Morasca, S.(1999). Software measurement. *Handbook. of Software Engineering and Knowledge Engineering*, 2, 239-276.
- Temprado, E., & Ruz Bendito, E. (2010). *Lean Software Development and Agile Methodologies for a small Software development organization*. Master thesis, university of Boras, Sweden.

- Morse, J. M. (2003). Principles of mixed methods and multimethod research design. *Handbook of mixed methods in social and behavioral research*, 189-208.
- Moser, R., Abrahamsson, P., Pedrycz, W., Sillitti, A., & Succi, G. (2008). A case study on the impact of refactoring on quality and productivity in an agile team. *Balancing Agility and Formalism in Software Engineering*, 252-266.
- Munassar, N. M. A., & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. *IJCSI International Journal of Computer Science Issues*, 7(5), 94-101.
- Murphy, T., & Cormican, K. (2012). An analysis of non-observance of best practice in a software measurement program. *Procedia Technology*, 5(2012), 50-58.
- Murry, J. W., & Hammons, J. O. (1995). Delphi-a versatile methodology for conducting qualitative research. *Review of Higher Education*, 18(4), 423-436.
- Murugesan, S., Deshpande, Y., Hansen, S., & Ginige, A. (2001). Web engineering: A new discipline for development of web-based systems. In *proceedings of the First ICSE Workshop on Web Engineering, International Conference on Software Engineering* (pp. 3-13). Springer Berlin Heidelberg.
- Nachmias, F., & Nachmias, D. (1996). *Research Methods in the Social Sciences*, 5th Edition, Aenold a member of the Hodder Headline Group, London.
- Næsset, L. R., & Bhargava, A. (2003). Electronic Process Guides in Connection With the Use of RUP at ConsultIT. *The Norwegian University of Science and Technology (NTNU)*.
- Naqvi, S. (2007). *A Semi-Autonomous On-Line Chemotherapy Prescription System*, Master, Thesis. Department of Computer Science, Memorial Newfoundland Uni, Canada.
- Nawaz, A., & Malik, K. (2008). *Software Testing Process in Agile Development*, Computer Science Master Thesis. Comp Science Dept. School of Engineering Blekinge Institute of Technology, Sweden, 2008.
- Neuman, W. (2003). *Social research methods: Qualitative and quantitative approaches*: Pearson Education
- O'Sheedy, D., & Sankaran, S. (2013). Agile Project Management for IT Projects in SMEs: A Framework and Success Factors. *The International Technology Management Review*, 3(3), 187-195.
- Okoli, C., & Carillo, K. (2012). The best of adaptive and predictive methodologies: open source software development, a balance between agility and discipline. *International Journal of Information Technology and Management*, 11(1),

153-166.

- Owens, D. M., & Khazanchi, D. (2009). Software Quality Assurance. *Handbook of Research on Technology Project Management, Planning, and Operations*. ISBN, 1965131010.
- Paetsch, F., Eberlein, A. & Maurer, F. (2003). Requirements engineering and agile software development. *Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, (pp. 308 -313).
- Pallant, J. (2007). *SPSS Survival Manual: A Step by Step Guide to Data Analysis Using SPSS for Windows Version 15: 3rd ed.* Open University Press
- Park, R. E., Goethert, W. B., & Florac, W. A. (1996). *Goal-Driven Software Measurement – A Guidebook*. Tech. Rep. CMU/SEI-96-HB-002, Software Engineering Institute, Carnegie Mellon University, August 1996.
- Park, Y., Park, H., Choi, H., & Baik, J. (2006, July). A study on the application of six sigma tools to PSP/TSP for process improvement. In: *proceeding of the 5th International Conference on Computer and Information Science, Component-Based Software Engineering, Software Architecture and Reuse. ICIS-COMSAR* (pp. 174-179). IEEE.
- Pathak, S., Pateriya, P., & Pal, P. (2012). A Case Study on Software Development Projects in Academic Knowledge Centers using SCRUM. *International Journal of Computer Applications*, 43(10), 20-24.
- Patton, M. (2002). *Qualitative research and evaluation methods*: Sage Publications, Inc.
- Paul, C. (1995). Software testing: A craftsman's approach. *CRC Press Inc.*
- Pinsonneault, A., & Kraemer, K. L. (1993). Survey research methodology in management information systems: an assessment. *Journal of management information systems*, 10(2),75-105.
- Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An agile toolkit*. Boston: Addison-Wesley Professional.
- Powell, T., Jones, D., & Cutts, D. (1998). *Web site engineering: beyond Web page design*: Prentice-Hall, Inc. Upper Saddle River, NJ, USA.
- Preciado, J. C., Linaje, M., Sanchez, F., & Comai, S. (2005, September). Necessity of methodologies to model Rich Internet Applications. In *Proceedings of the 7th IEEE International Symposium on Web Site Evolution*, (pp. 7-13). IEEE.
- Pressman, R. (2009). *Software Engineering: A Practitioner's Approach*, 7th Edition, McGraw-Hill Education.

- Preuninger, R.D. (2006). *The advantages of implementing software engineering process models*, Master Thesis. Faculty of the Graduate School, Texas At Arlington Uni, USA.
- Punch, K. (2005). *Introduction to social research: Quantitative and qualitative approaches*: Sage Publications Ltd.
- Pusatli, O. T., & Misra, S. (2011). Software Measurement Activities in Small and Medium Enterprises: an Empirical Assessment. *Acta Polytechnica Hungarica*, 8(5), 21-42.
- Quaglia, E. J., & Tocantins, C. A. (2011, December). Simulation projects management using Scrum. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, (pp. 3421-3430). IEEE.
- Qumer, A., & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and Software Technology*, 50(4), 280-295.
- Qureshi, M. R. J. (2011). Empirical Evaluation of the Proposed eXSCRUM Model: Results of a Case Study. *IJCSI*, 8(3), 150-157.
- Ralyté, J., Deneckère, R., & Rolland, C. (2003, January). Towards a generic model for situational method engineering. In *proceedings of 15th International Conference on the Advanced Information Systems Engineering (CAiSE 2003)* (pp. 95-110). Klagenfurt, Austria
- Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20(5), 449-480.
- Redouane, A. (2002, October). Guidelines for Improving the Development of Web-Based Applications. In *Proceedings of the Fourth International Workshop on Web Site Evolution (WSE'02)* (p. 93). IEEE Computer Society.
- Redouane, A. (2004, August). Towards a new method for the development of web-based applications. In *proceedings of the Third IEEE International Conference on Cognitive Informatics* (pp. 116-122). IEEE.
- Ribeiro, F. L., & Fernandes, M. T. (2010). Exploring agile methods in construction small and medium enterprises: a case study. *Journal of Enterprise Information Management*, 23(2), 161-180.
- Richardson, I., & von Wangenheim, C. (2007). Guest Editors' Introduction: Why are Small Software Organizations Different? *IEEE Software*, 24(1), 18-22.
- Rico, D. F., Sayani, H. H., & Sone, S. (2009). *The business value of agile software methods: maximizing ROI with just-in-time processes and documentation*. J.

Ross Publishing.

- Robson C. (1993) *Real World Research. A Resource for Social Scientists and Practitioner-Researchers*. Blackwell Publishers, Oxford.
- Rodríguez, D., Harrison, R., & Satpathy, M. (2002). A generic model and tool support for assessing and improving Web processes. In *Proceedings of the Eighth IEEE Symposium on Software Metrics (METRICS.02)* (pp. 141-151). IEEE.
- Rombach, H. D., & Basili, V. R. (1991). Practical benefits of goal-oriented measurement. *Software reliability and metrics*, 217-235.
- Rowe, G., & Wright, G. (1999). The Delphi technique as a forecasting tool: issues and analysis. *International Journal of Forecasting*, 15(4), 353-375.
- Royce, W. W. (1970, August). Managing the development of large software systems. In *proceedings of IEEE WESCON*, 26(8), 118-127.
- Rumbaugh J. (1995). What is a method?. *Journal of Object Oriented Programming*, 8(6), 10-16.
- Rumpe, B., & Schröder, A. (2002, May). Quantitative survey on extreme programming projects. In: *Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2002)*, (pp. 95-100). Alghero, Italy
- Runeson, P., & Isacson, P. (1998, August). Software quality assurance-concepts and misconceptions. In *Proceedings of the 24th Euromicro Conference (EUROMICRO 1998)*, Vasteras, Sweden, IEEE.
- Russo, N. L., & Graham, B. R. (1999, August). A first step in developing a Web application design methodology: understanding the environment. In *Proceedings of the Sixth International Conference on Information Systems Methodology* (pp. 24-33). Salford University, Manchester, UK, Springer, London, UK
- Saiedian, H., & Carr, N. (1997). Characterizing a software process maturity model for small organizations. *ACM SIGICE Bulletin*, 23(1), 2-11.
- Salo, O. (2006). *Enabling software process improvement in agile software development teams and organizations*. Ph.D. dissertation, University of Oulu, Finland.
- Salo, O., & Abrahamsson, P. (2008). Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum. *IET Software*, 2 (1), 58-64.
- Sanchez, J. C., Williams, L., & Maximilien, E. M. (2007, August). On the sustained

- use of a test-driven development practice at IBM. In *Proceedings of the Agile Conference (AGILE)*, (pp. 5-14). IEEE.
- Santos, G., Montoni, M., Vasconcellos, J., Figueiredo, S., Cabral, R., Cerdeiral, C. & Rocha, A. R. (2007, September). Implementing software process improvement initiatives in small and medium-size enterprises in Brazil. In *Proceedings of the Sixth International Conference on the Quality of Information and Communications Technology (QUATIC 2007)*, (pp. 187-198). IEEE.
- Sargent, R. G. (2012). Verification and validation of simulation models. *Journal of Simulation*, 7(1), 12-24.
- SARGUT, K. (2003). Application of Statistical Process Control to Software Development Processes via Control Charts. Master Thesis, THE MIDDLE EAST TECHNICAL UNIVERSITY.
- Sato, D., Bassi, D., Bravo, M., Goldman, A., & Kon, F. (2006). Experiences tracking agile projects: an empirical study. *Journal of the Brazilian Computer Society*, 12(3), 45-64.
- Savolainen, P., Sihvonen, H. M., & Ahonen, J. J. (2007). SPI with lightweight software process modeling in a small software company. In *Software Process Improvement* (pp. 71-81). Springer Berlin Heidelberg.
- Schneider, J. G., & Vasa, R. (2006, April). Agile practices in software development-experiences from student projects. In *Proceedings of the 2006 Australian Software Engineering Conference (ASWEC'06)*, Sydney, Australia. IEEE Computer Society.
- Scholtz, J., & Steves, M. P. (2004, November). A framework for real-world software system evaluations. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work* (pp. 600-603). ACM.
- Schwabe, D., & Rossi, G. (1995). The object-oriented hypermedia design model. *Communications of the ACM*, 38(8), 46.
- Schwabe, D., & Rossi, G. (1998, June). Developing hypermedia applications using OOHD. In *Workshop on Hypermedia Development Process, Methods and Models, (Hypertext '98)*, Pittsburg, USA
- Schwaber, K. and Beedle.(2001). *Agile Software Development with Scrum: Upper Saddle River: Prentice Hall*. New Jersey 2001
- Seidman, I. (1991). *Interviewing as qualitative research: A guide for researchers in education and the social sciences*: Teachers College Press New York.
- Sekaran, U. & Bougie, R. (2010). *Research methods for business: A skill building*

Approach (5th ed.). New York: John Wiley & Sons.

- Serrano, M. A., Montes de Oca, C., & Cedillo, K. (2003, November). An experience on using the team software process for implementing the Capability Maturity Model for software in a small organization. In *Proceedings of the Third International Conference on Quality Software (QSIC'03)*, (pp. 327-334). IEEE.
- Sison, R., & Yang, T. (2007). *Use of Agile Methods and Practices in the Philippines*. Paper presented at the 14th Asia-Pacific Software Engineering Conference (APSEC'07), Nagoya, Japan.
- Sjoberg, D. I., Dyba, T., & Jorgensen, M. (2007, May). The future of empirical methods in software engineering research In *Proceedings of the 29th International Conference on Software Engineering (ICSE'07)*, (pp. 358-378). *Future of Software Engineering (FOSE'07)*, Minneapolis, Minnesota, USA IEEE Computer Society.
- Software Management Guide, Vol. I, Software Technology Support Center, October 1993, p. 23.
- Software Program Managers Network (SPMN). 1999. 16 Critical Software Practices™ for Performance-based Management. <<http://www.spmn.com/critical-software-practices.html>>
- Sorensen, R. (1995). A comparison of software development methodologies. *CrossTalk*, 8(1), 10-13.
- Sørungård, S., & Sindre, G. (1995, July). Aspects of process quality. In *Proceedings of the 4th International Conference on Software Quality, Dundee, Scotland* (pp. 4-5).
- Spasibenko, N., & Alite, B. (2009). *Project Suitability for Agile methodologies*. Master.Thesis, Umeå School of Business, Sweden.
- Stojanovic, Z., Dahanayake, A., & Sol, H. (2003). Modeling and Architectural Design in Agile Development Methodologies. In *Proceedings of the 8th International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'03)*, (pp.1-10). Velden, Austria.
- Strode, D. (2006). *Agile methods: a comparative analysis*. Paper presented at the 19th Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ 2006), Wellington, New Zealand.
- Stutzke, R. D. (2005, November). Measuring and Estimating Process Performance. In *5th Annual CMMI Technology Users Group Meeting Denver, Colorado*.

- Sulayman, M., & Mendes, E. (2009). A systematic literature review of software process improvement in small and medium web companies. In *Proceedings of the International Conference on Advanced Software Engineering and Its Applications (ASEA'09)*, (pp. 1–8). Jeju Island, Korea, Springer. doi:10.1007/978-3-642-10619-4_1
- Sulayman, M., & Mendes, E. (2010, March). Quantitative assessments of key success factors in software process improvement for small and medium web companies. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (pp. 2319-2323). ACM.
- Sulayman, M., Urquhart, C., Mendes, E., & Seidel, S. (2012). Software process improvement success factors for small and medium Web companies: A qualitative study. *Information and Software Technology*, 54(2012), 479-500.
- Suwanya, S., & Kurutach, W. (2008, July). An analysis of software process improvement for sustainable development in Thailand. In *Proceedings of 2008 IEEE 8th International Conference on Computer and Information Technology*, Sydney, Australia (pp. 724-729). IEEE.
- Tabachnick, B., & Fidell, L. (2007). Using multivariate analysis. Using multivariate analysis. 5th Edition Allyn & Bacon; Needham Heights MA.
- Tan, M., & Yap, C. Y. (1995). Impact of organisational maturity on software quality. In *Software Quality and Productivity: Theory, practice, education and training*, Editors M. Lee, B. Barta, & P. Juliff, Chapman and Hall for IFIP, London, (pp. 231-234). Springer US.
- Tarafdar, M., & Zhang, J. (2008). Determinants of reach and loyalty-a study of Website performance and implications for Website design. *Journal of Computer Information Systems*, 48(2), 16-24.
- Tarawneh , H& Allahawiah, S (2009). Web applications Development and Software Process Improvement in Small Software Firms: a Review. In *Proceedings of the 4th international Conference on Information Technology (ICIT 2009)*, al zaytoonah University of Jordan.
- Tessem, B. (2003). Experiences in learning xp practices: A qualitative study. In *Proceedings of the Fourth International Conference on Extreme Programming and Agile Processes in Software Engineering* (pp. 131-137). Springer Berlin Heidelberg.
- Thacker, B. H., Anderson, M. C., Senseny, P. E., & Rodriguez, E. A. (2006). The role of nondeterminism in model verification and validation. *International Journal of Materials and Product Technology*, 25(1), 144-163.
- Theunissen, W. H., Boake, A., & Kourie, D. G. (2005, July). In search of the sweet spot: agile open collaborative corporate software development. In *Proceedings of the 2005 annual research conference of the South African*

institute of computer scientists and information technologists on IT research in developing countries (pp. 268-277). South African Institute for Computer Scientists and Information Technologists.

- Tofan, D., Galster, M., Avgeriou, P., & Weyns, D. (2011, April). Software engineering researchers' attitudes on case studies and experiments: An exploratory survey. In *proceedings of the 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*, (pp. 91-95). IET.
- Toffolon, C., & Dakhli, S. (1998, March). Software artifacts reuse and maintenance: an organizational framework. In *Proceedings of the 2nd Euro micro Conference on Software Maintenance and Reengineering (CSMR)*, (pp. 228-233). March 1998, Palazzo degli Affari, Italy IEEE.
- Trochim, W. M. (2006). *Qualitative measures. Research Measures Knowledge Base*, (pp.361-9433). Retrieved on 13 May 2013 from <http://www.socialresearchmethods.net/kb/qualval.php>.
- Tsai, H. L., & Cheung, D. (1999). A monitoring framework for software project development. In *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials (IPMM'99)*, (pp. 1079-1085). IEEE.
- Tsun, C. Dac-Buu, C. (2008). A survey study of critical success factors in agile software projects. *The Journal of Systems and Software*, 81(6), 961–971.
- Tu, H., Sun, W., & Zhang, Y. (2009). *The Research on Software Metrics and Software Complexity Metrics*. Paper presented at International Forum on the Computer Science-Technology and Applications IFCSTA '09.
- Turk, D., France, R., Rumpe, B. (2002). *Limitations of Agile Software Processes*, paper presented at the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, Sardinia, Italy, 43-46.
- Turk, D., Robert, F., & Rumpe, B. (2005). Assumptions underlying agile software-development processes. *Journal of Database Management (JDM)*, 16(4), 62-87.
- Tyrrell, S. (2000). The many dimensions of the software process. *Crossroads*, 6 (4), 22-26.
- Upender, B. (2005, July). Staying agile in government software projects. In *Proceedings of the Agile Development Conference (ADC'05)* (pp. 153-159). IEEE.

- Väänänen, M. (2008), “*evaluating agile methods and their implementations*,” Master thesis, Information System Competence, Applied Sciences UNI, Finland
- Van Solingen, R. (2002). The goal/question/metric approach. *Encyclopedia of Software Engineering—2 Volume Set*, 578-583.
- Van Solingen, R., & Berghout, E. (1999). *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*: McGraw-Hill.
- Van Solingen, R., & Berghout, E. (2001). *Integrating goal-oriented measurement in industrial software engineering: industrial experiences with and additions to the Goal/Question/Metric method (GQM)*.paper presented at the 7th International Software Metrics Symposium, London, UK,
- Varkoi, T., & Mäkinen, T. (2000, October). Software process improvement initiation in small organisations. In *Proceedings of the 3rd European Software Measurement Conference, FESMAAEMES*, Madrid, Spain.
- Visconti, M., & Cook, C. R. (2004). An ideal process model for agile methods. In *Proceedings of 5th International Conference on Product Focused Software Process Improvement PROFES 2004*, (pp.431-441), Lecture Notes in Computer Science, 3009,
- Von Wangenheim, C. G., Punter, T., & Anacleto, A. (2003). Software measurement for small and medium enterprises. In *Proceeding of the 7th International Conference on Empirical Assessment in Software Engineering (EASE)*. Keele University, Staffordshire, UK.
- Vriens, C. (2003, June). Certifying for CMM Level 2 and ISO9001 with XP@ Scrum. In *Proceedings of the Agile Development Conference*, (pp. 120-124). IEEE.
- Weiss, D. (1994). GQM plus heuristics better than brainstorming. *IEEE Software*, 11(1), 8-9.
- West, D., Grant, T., Gerush, M., & D’silva, D. (2010). Agile development: Mainstream adoption has changed agility. *Forrester Research*, 2, 41.
- Whitgift, D. (1991). *Methods and tools for software configuration management*: John Wiley & Sons, Inc. New York, NY, USA.
- Whitson, G. (2006). WebHelix: another web engineering process. *Journal of Computing Sciences in Colleges*, 21(5), 21-27.
- Wikipedia. (2011). List of software development philosophies. Retrieved Dec. 08, 2011, from http://en.wikipedia.org/wiki/List_of_software_development_philosophies#Software_development_philosophies

- Williams, L. (2012). What agile teams think of agile principles. *Communications of the ACM*, 55(4), 71-76.
- Williams, L., & Erdogmus, H. (2002, May). On the economic feasibility of pair programming. In *Proceedings of the International Workshop on Economics-Driven Software Engineering Research EDSE*, Orlando, Florida, USA.
- Williams, L., Rubin, K., & Cohn, M. (2010, August). Driving process improvement via comparative agility assessment. In *Proceedings of the Agile 2010 Conference*, (pp. 3-10). Piscataway, NJ IEEE.
- Wills, G. B., Abbas, N., Chandrasekharan, R., Crowder, R. M., Gilbert, L., Howard, Y. M. & Walters, R. J. (2007, September). An agile hypertext design methodology. In *Proceedings of the eighteenth conference on Hypertext and hypermedia* (pp. 181-184). ACM.
- Winger, A. R. (1994). Is Big Really Bad? *Business Economics*. 29(3), 38-42.
- Withers, D. H. (2000, December). Some fundamental issues in model building: software engineering best practices applied to the modeling process. In *Proceedings of the 32nd conference on Winter simulation* (pp. 432-439).
- Wohlin, C., Höst, M., & Henningsson, K. (2006). Empirical research methods in Web and software Engineering. In *Web engineering* (pp. 409-430). Springer Berlin Heidelberg.
- Wong, B., & Hasan, S. (2006). Software Process Improvement In Bangladesh. In *Software Engineering Research and Practice* (pp. 246-252).
- Wu, Y., & Offutt, J. (2002). Modeling and testing Web-based applications. *Technical Report*, George Mason University 2002.
- Xu, Y., Lin, Z., & Foster, W. (2003). Agile Methodology in CMM Framework: an Approach to Success for Software Companies in China. In *Proceedings of the Global Information Technology Management GITM*. Calgary, Alberta, Canada.
- Yin, R. (2003). *Case Study Research Design and Methods* (3rd edition). London, UK: Sage.
- ZAROUR, M. (2009). *Methods to evaluate lightweight software process assessment methods based on evaluation theory and engineering design principles*, PhD Thesis, Universite du Quebec, Canada
- Zelenka, P. (2006). Modern methods of web applications analysis and design. *ZEMEDLSKA EKONOMIKA-PRAHA*-, 52(4), 152.
- Zelkowitz, M., & Wallace, D. (1998). Experimental models for validating technology. *Computer*, 31(5), 23-31.

Appendix A

Questionnaire

Web Application Development Methodology for Small Software Firms
College of Arts and Science
Universiti Utara Malaysia

SURVEY FORM
WEB APPLICATIONS DEVELOPMENT AND MEASUREMENT PRACTICES
IN SMALL SOFTWARE FIRMS

The objectives of this survey are:

- 1- To determine the real characteristics of small software firms in Jordan.
- 2- To examine the need of new methodology for developing web applications in small software firms.
- 3- To study the current practices of web applications development in small software firms.
- 4- To study the measurement practices those have been performed currently in small software firms.

Instructions:

This survey takes 20-30 minutes to complete. Please read the questions carefully which categorize into four sections as shown below.

SECTION I : RESPONDENT BACKGROUND
SECTION II : ORGANIZATION BACKGROUND
SECTION III : DEVELOPMENT AND MEASUREMENT ISSUES
SECTION IV : WEB APPLICATIONS DEVELOPMENT AND MEASUREMENT PRACTICES

Please tick (✓) in the boxes unless stated otherwise. You are advised to answer the questions depending on your knowledge and your responsibility. The data from this survey is strictly confidential and for the purpose of this research only. Thank you for sharing your opinions with us.

Respondent Name:	
Telephone No:	
Company email:	Company Website:
Date:	

Section I: (Respondent Background) (3 Questions)

1- What is your current position?

- Project or Team Leader
- Manager
- Technical Member
- Software Engineering Process Group Member
- Other (please specify).....

2- Which of these activities is your current position involved?

- Software Requirements
- Software Quality Assurance
- Software Design
- Configuration Management
- Code and Unit Test
- Software Process Improvement
- Test and Integration
- Others (please specify).....

3- How many years have you worked in web application development?

- Less than 3 years
- 3 -10 years
- 10-20 years
- More than 20 years

Section II: (Organization Background) (3 Questions)

1- Type of organization:

- Government Agency
- Private Sector

2- Is your organization a software development company?

- Yes
- No (don't complete answering the questionnaire)



3 – What is your organization size?

- Less than 10 people
- 10 – 30 people
- 31-50 people
- More than 50 people (don't complete answering the questionnaire)

Section III (Software Development and Measurement Practices) (22 Questions)

1- Application domain

- Business information systems (general)
- E-learning
- E-banking
- Web engineering tools
- E-commerce
- Personal web pages
- E-business (general)
- Other (please specify).....

2- What type of software philosophy is your organization familiar with?

- Code and fix
- Waterfall
- Agile software development
- Your own philosophy (ad hoc)
- Other (please specify).....

3- Which of these software development methods is used by your organization?

- | | |
|---|---|
| <input type="checkbox"/> Waterfall | <input type="checkbox"/> Agile modeling (AM) |
| <input type="checkbox"/> V- Model | <input type="checkbox"/> Incremental |
| <input type="checkbox"/> Rational Unified Process (RUP) | <input type="checkbox"/> Prototyping |
| <input type="checkbox"/> Spiral model | <input type="checkbox"/> Enterprise Unified Process (EUP) |
| <input type="checkbox"/> Agile Unified Process (AUP) | <input type="checkbox"/> Extreme Programming (XP) |
| <input type="checkbox"/> Dynamic System Development Method (DSDM) | <input type="checkbox"/> Scrum |
| <input type="checkbox"/> Feature Driven Development (FDD) | <input type="checkbox"/> Other (please specify)..... |
| <input type="checkbox"/> No Specific Methodology Used | |

4- Which of these software development methods are you familiar with?
(you may choose more than one answer)

- | | |
|---|---|
| <input type="checkbox"/> Waterfall | <input type="checkbox"/> Agile modeling (AM) |
| <input type="checkbox"/> V- Model | <input type="checkbox"/> Incremental |
| <input type="checkbox"/> Rational Unified Process (RUP) | <input type="checkbox"/> Prototyping |
| <input type="checkbox"/> Spiral model | <input type="checkbox"/> Enterprise Unified Process (EUP) |
| <input type="checkbox"/> Agile Unified Process (AUP) | <input type="checkbox"/> Extreme Programming (XP) |
| <input type="checkbox"/> Dynamic System Development Method (DSDM) | <input type="checkbox"/> Scrum |
| <input type="checkbox"/> Feature Driven Development (FDD) | <input type="checkbox"/> Other (please specify)..... |

5- Within the life cycle, is there any prototyping method was used

- Yes
- No (skip question 6)

6- How is this prototyping method performed in your organization?

- | | |
|--|--|
| <input type="checkbox"/> 4GL environment | <input type="checkbox"/> Throw-away prototype |
| <input type="checkbox"/> Evolutionary prototype | <input type="checkbox"/> User Interface |
| <input type="checkbox"/> No Specific Prototyping Method Used | <input type="checkbox"/> Other (please specify)..... |

7- What kind of requirements collection methods does your organization use?

- | | |
|---|--|
| <input type="checkbox"/> Questionnaires | <input type="checkbox"/> Document reviews |
| <input type="checkbox"/> Interviews | <input type="checkbox"/> Use case scenarios |
| <input type="checkbox"/> Observations | <input type="checkbox"/> Other (please specify)..... |

8- Which of the software requirement analysis methods does your organization use?

- Structure Analysis and Design (SAD)
- Structured System Analysis and Design Method (SSADM)
- Structured Requirements Definition
- Object Oriented Analysis (OOA)
- Other (please specify).....

9- In what notation is the requirement specification presented?
(you may choose more than one answer)

- Formal (e.g.: Z, VDM)
- Semi formal (e.g.: DFD, ERD, STD)
- Informal (Natural Language)
- No specific notation
- Other (please specify).....

10- What kind of programming languages does the development team use in this organization?
(you may choose more than one answer)

- 3GL (i.e.: COBOL, FORTRAN, C)
- 4GL (i.e.: Informix, DbaseIV, SQL)
- Visual languages (i.e. V.B,Delphi)
- Object oriented (i.e. Java,C++)
- Other (please specify).....

11- Which kinds of tests are required by your organization?
(you may choose more than one answer)

- Unit Tests
- System Testing
- Acceptance Tests
- Usability Testing
- Beta Testing
- No tests are required
- Integration Tests
- Code Coverage Tests
- Regression Testing
- Alpha Testing
- Functional Testing
- Other (please specify).....

12- At what stage of the development activities does the testing process begin with?

- The end of the coding phase.
- Early as soon as possible software projects were acquiring.
- Documentation or element that can be tested.
- While integrating major software modules.
- When implementing the final acceptance testing .
- Other (please specify).....

13- If you don't use any method for developing web application in your company please specify why?

- Nobody inside the organization familiar with any type of web applications development methods
- Using any development methods takes a lot of time.
- Consume a lot of money
- Need specific team to be performed
- Need specific training to be performed
- Other (please specify).....

14- Does your organization encourage the software reuse?

- Yes
- No (skip question 14)

15- What kinds of components does this organization reuse?
(you may choose more than one answer)

- | | |
|--|--|
| <input type="checkbox"/> Source Code | <input type="checkbox"/> Feasibility Studies |
| <input type="checkbox"/> Media | <input type="checkbox"/> Web pages |
| <input type="checkbox"/> Templates (use case, project plan etc) | <input type="checkbox"/> Design Document |
| <input type="checkbox"/> User Documentation/Specification | <input type="checkbox"/> Data |
| <input type="checkbox"/> Modules (Web services, technical components etc.) | <input type="checkbox"/> Navigation |
| <input type="checkbox"/> Cost benefits calculators and estimates | <input type="checkbox"/> Test Data |
| <input type="checkbox"/> No software reuse required | <input type="checkbox"/> Other (please specify)..... |

16- What kind of quality assurance activities does your organization perform?
(you may choose more than one answer)

- Testing of Web-based Applications
- Code review (formal or informal)
- Development Process Audit
- Configuration Management Audit
- Functional Configuration Audit
- Physical Configuration Audit
- Version Description Document or equivalent
- No assurance activities are performed (skip question 16)

17-Who is responsible for performing the quality assurance activities?

- Project team
- Software Assurance Group
- Other Assurance Group (outside)
- Others (please specify).....

18- Does your organization perform software development measurement?

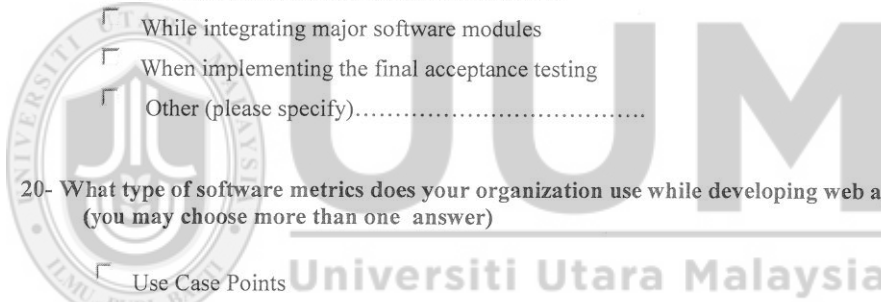
- Yes
- No (skip questions 18, 19 and 20)

19- At what stage of the development process does your organization perform the software measurement?

- The end of the coding phase
- Early as soon as possible software projects were acquiring
- Documentation or element that can be tested
- While integrating major software modules
- When implementing the final acceptance testing
- Other (please specify).....

20- What type of software metrics does your organization use while developing web applications?
(you may choose more than one answer)

- Use Case Points
- Constructive Cost Model (COCOMO)
- Function Points
- Pages Counts
- Line of Code (LOC)
- Links Count
- No Specific Metric Used
- Other (please specify).....



21- What type of software measurement method does your organization use to perform the measurement program?

- Goal/Question/Metric (GQM)
- Practical Systems and Software Measurement (PSM)
- Quality Function Deployment (QFD)
- No Specific Method Used
- Other (please specify).....

22- If your organization does not perform software measurement during the development process, could you explain why?

- Nobody inside the company familiar with software measurement
- Take a lot of time to employ software measurement
- Consume a lot of money
- Need specific team to perform
- You organization is not a ware to perform software measurement during the development
- Other (please specify).....


Section IV: (Web Applications Development and Measurement Practices) (17 Questions)


No	Question	Strongly Agree	Agree	Don't know	Disagree	Strongly Disagree
4.1	Does your development process of Web applications copes with time pressure?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4.2	Does your development process of Web applications clarify that all involved in this process understand their roles and responsibilities?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4.3	Does the development team ensure that the development process must be performed with minimum design and quick prototype?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4.4	Does each web project have a nominated web project manager?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4.5	Does your web project plan project plan perform the budget estimation?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4.6	Are the requirements collected directly from the user or and the manager?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- 4.7 Are design notations used in web design?
- 4.8 Does the development process ensure that all components of the web application such as page, code, site, navigation and services are being tested by test cases generated according to requirement specifications?
- 4.9 Is the testing process carried out or performed by the development process team?
- 4.10 Do the developers pay attention to the quality management and standards such as usability and user interface design when developing web applications in your company?
- 4.11 Is independent testing conducted by users (or appropriate representatives) under the guidance of Software Quality Assurance before any system or enhancement goes live?
- 4.12 Is there a procedure for controlling changes to the web applications requirements, designs and accompanying documentation?
- 4.13 Is a change control function established for each web project?
- 4.14 Is there a documented procedure for estimating web applications size (such as "Lines of Source Code") and thus for using productivity measures?
- 4.15 Is a formal procedure used to produce web development effort, schedule, and cost estimates?
- 4.16 Is there a required training program for all newly-appointed web managers which is designed to familiarize them with in-house web project management procedures?
- 4.17 Is there a procedure for maintaining awareness of the state-of-the-art in case of web engineering technology?

Appendix B

Questionnaire Face Validity Cover Letter



 **UNIVERSITI UTARA MALAYSIA**
06010 UUM Sintok, Kedah Darul Aman, Malaysia. Tel: 604 - 928 4000

“KEDAH SEJAHTERA”

8 December 2010

**Norhasimah Bt Mustafa
Noorfaizalfarid Bin Mohd Noor
Mohammad Zahid bin Che Man**

Dear Sir/Mdm,

APPOINTMENT AS A DOMAIN EXPERT

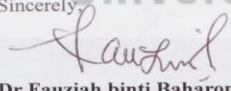
This letter confirms that Mo'ath Husni Ahmad Altarawneh (91413) is a PhD student at College of Arts and Sciences, Universiti Utara Malaysia. He is doing a research under my supervision. The research aims to propose a new methodology for Web application development in small software firms. One of the objectives in his research is to understand the practices related to Web application development that currently performed by software practitioners. This objective can be achieved through a survey and before he distributes the questionnaire to the actual respondents, we need your help in giving us some feedback and comments related to the validity and understandability of the questionnaire.

Therefore, it is kindly appreciated for your cooperation and participation in providing him with the above matter. Please accept our sincere gratitude and feel free to contact me, if you need further information.




Thank you

“ILMU BUDI BAKTI”
Universiti Utara Malaysia

Sincerely,



Dr Fauziah binti Baharom
Senior Lecturer (Information Technology)
College Arts and Sciences
Universiti Utara Malaysia
06010 Sintok Kedah Darul Aman,
Malaysia
Office Tel: 04-928 7010
Email: fauziah@uum.edu.my

   **MSC**
—MALAYSIA—
Status Institution

Appendix C

Expert cover letter



PUSAT PENGAJIAN PENGKOMPUTERAN
SCHOOL OF COMPUTING
College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM SINTOK
KEDAH DARUL AMAN
MALAYSIA



Tel: 604-928 5056/5058/5060
Faks (Fax): 604-928 5067
Laman Web (Web): www.soc.uum.edu.my

“KEDAH AMAN MAKMUR – BERSAMA MEMACU TRANSFORMASI”

9 June 2014

TO WHOM IT MAY CONCERN,

Sir/Madam,

APPOINTMENT AS EXPERT REVIEWER

This letter confirms that Moath Husni Altarawneh (91413) is a PhD student at School of Computing, College of Arts and Sciences, Universiti Utara Malaysia (UUM). Currently, he is doing a research under my supervision. The research aims to propose a new “Monitoring Oriented Agile Based Web Applications Development Methodology for Small Software Firms”. One of the objectives of his research is to verify components of the proposed methodology which could be achieved through an expert review. Thus to achieve the objective, we need your help in giving us some feedbacks and comments regarding the correctness and understandability of the methodology.

It is highly appreciated if you could give your kind cooperation and participation in providing her with the above mentioned matter. Kindly accept our sincere gratitude and feel free to contact me if you need further information.

Thank you.

“ILMU BUDI BAKTI”

Yours Sincerely

Dr. Fauziah Baharom

Assoc. Professor
School of Computing
College of Arts and Sciences
Universiti Utara Malaysia
06010 Sintok
Kedah.

HP: +6019-4741666

Email: fauziah@uum.edu.my

Universiti Pengurusan Terkemuka
The Eminent Management University



Appendix D

Knowledge expert questionnaire

Reviewing the proposed new monitoring oriented agile based web application development methodology for small software firms

PhD Student: Moath Husni Altarawneh
School of computing
College of Arts and Sciences
Universiti Utara Malaysia
Kedah, Malaysia
Tarawneh80@yahoo.com

The research aims to propose a new monitoring oriented agile based web application development methodology for small software firms. One of the objectives of this research is to verify the components of the methodology. This could be achieved through an expert review. Your answers to the following questions will serve as useful feedback on the methodology's comprehensiveness, understandability and feasibility. Your kind cooperation and participation in answering the questions is highly appreciated and will be treated as strictly confidential.

=====

The new methodology consists of five (5) components: activities, methods, practices, tools and team structure.

You are required to give answers related to each component.

1. Activities and methods

The proposed methodology process consists of two sides: development and measurement. Development process performed based on the combined XP and Scrum method and web design prototype. A measurement process performed based on the goal oriented monitoring method (GOMM).

1.1 Development activities: planning, development and integration

a) Are the activities of the development process correct, clear and feasible?

.....
.....
.....
.....
.....

b) Do the activities cover all the stages of building Web applications?

.....
.....
.....
.....

c) Are the web design method steps correct, clear and feasible?



UUM

Universiti Utara Malaysia

d) Is it applicable to use the planning phase to perform the web design method?

.....
.....
.....
.....
.....

e) Are the requirement repository activities (save, reuse and trace requirements.)
correct, clear and feasible?

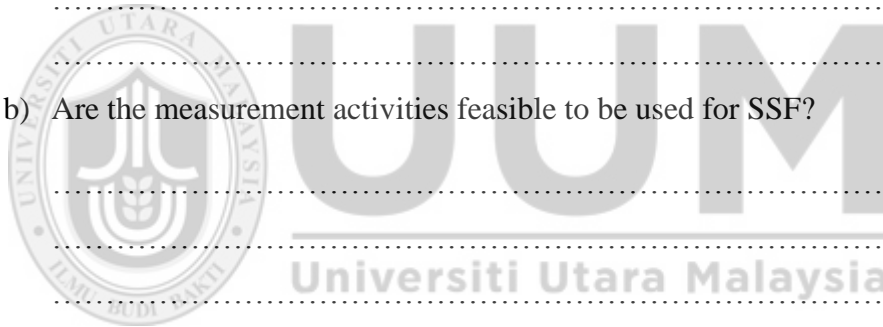
.....
.....
.....
.....
.....

1.2 Measurement process: planning, definition and feedback

a) Are the measurement activities correct and clear?

.....
.....
.....

b) Are the measurement activities feasible to be used for SSF?



.....
.....

c) Do the measurement activities cover all stages of building Web applications?

.....
.....
.....
.....

d) Please fill up the following table to determine the correctness and the clearness of goals. Use "√" for **YES** and **leave the space blank** for **NO**.

Goal description	Goal	Correct	Clear
Process activities	G1.1:To analyze requirement status for the purpose of monitoring with respect to no. of requirements completed from the viewpoint of GOMM member		
	G1.2:To analyze the design status for the purpose of monitoring with respect to no of SLOC, no of web pages and total no. of links from the viewpoint of GOMM member		
	G1.3To analyze the testing process for the purpose of monitoring with respect to current size of test status from the viewpoint of GOMM member		
Practices	PG1:To analyze common Scrum practices (core) for the purpose of Monitoring with respect to Scrum meetings from the viewpoint of GOMM member		
	PG2: To analyze the common XP practices (core) for the purpose of monitoring with respect to pair programming, TDD, refactoring, coding standards, from the viewpoint of GOMM member		
	PG3:To analyze the supported XP practices for the purpose of monitoring with respect to small release, continuous integration, simple design, metaphor and collective ownership from the viewpoint of GOMM member		
Productivity	G2:To analyze productivity tracking for the purpose of monitoring with respect to value of staff productivity from the viewpoint of GOMM member		
Process quality	QG1: To analyze the process completeness for the purpose of monitoring with respect to process activities requirements, design, coding, testing and project management from the viewpoint GOMM member through a questionnaire		
	QG2: To analyze the process consistency for the purpose of monitoring with respect to process activities requirements, design, coding, testing and project management from the viewpoint of GOMM member through a questionnaire		
	QG3: To analyze the process accuracy For		

	the purpose of Monitoring With respect to process activities requirements, design, coding, testing and project management From viewpoint of GOMM member through a questionnaire		
	QG4: To analyze the process tailorability for the purpose of monitoring with respect to tailorability practices from the viewpoint of GOMM member through a questionnaire		
	QG5: to analyze the process flexibility for the purpose of monitoring with respect to flexibility practices from the viewpoint of GOMM member through a questionnaire		
	QG6: To analyze the process compatibility for the purpose of monitoring with respect to compatibility practices from the viewpoint of GOMM member through a questionnaire		
	QG7: To analyze the process accessibility for the purpose of monitoring with respect to accessibility practices from the viewpoint of GOMM member through a questionnaire		
	QG8: To analyze the process applicability for the purpose of monitoring with respect to applicability practices from the viewpoint of GOMM member through a questionnaire		
	QG9: analyze the process changeability for the purpose of monitoring with respect to changeability practices from the viewpoint of GOMM member through a questionnaire		
	QG10: to analyze the process supportability for the purpose of monitoring with respect to supportability practices from the viewpoint of GOMM member through a questionnaire		
Cost	To analyze development process cost for the purpose of monitoring and controlling with respect to the cost of fix ,cost of activity and project budget from the viewpoint of GOMM member		
Quality	To analyze quality aspects for the purpose of monitoring With respect to security, product reliability, usability and maintainability from the viewpoint of GOMM member		
Time	To analyze development life cycle time for the purpose of monitoring With respect to reuse artifacts, time for each iteration, project velocity from the view point of GOMM member		

e) Please fill up the following table to determine the correctness and the clearness of the questions. Use "√" for **YES** and **leave the space blank** for **NO**.

Metric type	Questions	Correct	Clear
Process activities	Q1.1.1: What is the current size of the requirements status?		
	Q1.2.1: What is the current size of the design status?		
	Q1.3.1: What is the current size of the test status?		
Practices	PQ1.1: How to measure the iteration planning meeting?		
	PQ1.2: How to measure the daily meeting?		
	PQ1.3: How to measure the iteration review meeting?		
	PQ2.1: Does the coding stage performed by two programmers simultaneously?		
	PQ2.2: how to monitor the TDD practice?		
	PQ2.3: Does the duplicated code removed to decrease ambiguity and redundancy, and improve communication and adding flexibility?		
	PQ2.4: Does the development team follow a coding standard?		
	PQ3.1: Is every iteration release with small size of code?		
	PQ3.2: Does the new created release reflecting all the changes?		
	PQ3.3: Is the architecture and the code (including the unit tests) as simple as possible?		
	PQ3.4: Does the System created by set of Metaphors between the client and programmers?		
	PQ3.5: Do all team members are owners of the code (can make changes on the code)?		
	Productivity	Q2.1: What is the value of productivity of the project staff?	
Completeness	QQ1.1: what is the degree of requirement completeness?		

	QQ1.2: what is the degree of design completeness?		
	QQ1.3: What the degree of coding completeness?		
	QQ1.4: What the degree of testing completeness?		
	QQ1.5: What the degree of project management completeness?		
Consistency	QQ2.1: what is the degree of requirement consistency?		
	QQ2.2: what is the degree of design consistency?		
	QQ2.3: What the degree of coding consistency?		
	QQ2.4: What the degree of testing consistency?		
	QQ2.5: What the degree of project management consistency?		
Accuracy	QQ3.1: what is the degree of requirement accuracy?		
	QQ3.2: what is the degree of design accuracy?		
	QQ3.3: What the degree of coding accuracy?		
	QQ3.4: What the degree of testing accuracy?		
	QQ3.5: What the degree of project management accuracy?		
Tailorability	QQ4.1: what is the degree of process tailorability?		
Flexibility	QQ5.1: what is the degree of process flexibility?		
Compatibility	QQ6.1: what is the degree of process compatibility?		
Accessibility	QQ7.1: what is the degree of process accessibility?		
Applicability	QQ8.1: what is the degree of process applicability?		
Changeability	QQ9.1: what is the degree of process changeability?		
Supportability	QQ10.1: what is the degree of process supportability?		
Cost	Q3.1: What is the cost of fix post to release problem in a month?		
	Q3.2: What is the current cost by activity for each software product?		

	Q3.3: What is the current budget status of the project?		
Quality	Q4.1: What the distribution of failure after delivery?		
	Q4.2: What is the defect density?		
	Q4.3: What is the quality of the defect detection process?		
	Q4.4: What is the product reliability?		
	Q4.5: What is the total effort in hours spent in locating the fault vs. total effort spent for fixing the fault?		
	Q4.6: how to monitor the usability of Web application?		
	Q4.7 how to monitor Web application's maintainability?		
Time	Q5.1: What is the percentage of reuse artifacts?		
	Q5.2: What is the development time by activity for each Web application product?		
	Q5.3: What is the Project velocity?		

f) Please fill up the following table to determine the metric correctness, clearness and feasibility. Use "√" for **YES** and **leave the space blank** for **NO**.

Metric type	Metric	Correct	Clear	Feasible
Process activities	M1.1.1.1: Number of product backlog items completed to date / Total Number of requirements planned.			
	M1.2.1.1: Number of LOC completed to date / Total Number of SLOC planned.			
	M1.2.1.2: Number of Web Pages to date / Total Number of Web Page planned.			
	M1.2.1.3: Total Number of internal links / Number of Web pages.			
	M1.3.1.1: Number of test completed to date / Total Number of test planned.			
	M1.3.1.2 number of testing line of code / total number lines of code			
Practices	PM1.1.1: Number of iteration planning meetings per one application.			
	PM1.2.1: Number of daily meetings per one application?			
	PM1.3.1: Number of review meetings done per one application?			
	PM 2.1.1: Number of programmers.			

	PM2.2.1: Number of tests completed to date /Total Number of tests planned.			
	PM2.2.2: Number of testing line of code / total number lines of code.			
	PM2.3.1: Number of lines of duplicated code removed / total line of code per iteration.			
	PM2.4.1: Adherence of coding standard (High, Low).			
	PM3.1.1: (Number of LOC of the first release - the LOC of the next release) / total NLOC			
	PM 3.2.1: Total number of line of code added, removed and updated) / total line of code for the previous iteration.			
	PM3.3.1: (Number of LOC of the current release - total LOC) / Total LOC			
	PM3.4.1: Number of meetings between development team and the client?			
	PM3.5.1Number of team members who made changes in the code.			
Productivity	M2.1.1: Number of LOC for staff in month.			
Cost	M3.1.1: Dollar cost related to fix post to release problems.			
	M3.2.1: Number of dollars spent to date for activity i /Number of dollars estimated for activity.			
	M3.3.1Number of total dollars spent to date / Number of total dollars estimated.			
Quality	M4.1.1: Severity classification for each detected failure (fatal, major, minor and other).			
	M4.2.1: Number of iteration i defects / metric for size in sprint i(LOC).			
	M4.3.1: Number of pre-release defects of in iteration / (Number of pre-release + post-release defects).			
	M4.4.1: Number of defects / execution time.			
	M4.5.1: Effort in hours for locating each fault.			
	M4.5.2: Efforts in hours for fixing the fault.			
	M4.6.1 No. of page links/ total number of internal links (navigability)			
	M4.6.2 Response time			
	M4.6.3 Memory space			
	M4.7.1dynamic pages/ total no. of			

	pages (changeability) should be low			
	M4.7.2 dynamic testing LOC/ total LOC testability should be low			
	M4.7.3 1/ no of direct links (stability) should be high			
Time	M5.1.1: Number SLOC of reusing code / Number of SLOC completed to date.			
	M5.1.2: Number of reuse Web pages / total Web Page number.			
	M5.2.1: Elapsed time / estimated time.			

g) Please fill up the following table determine the metric correctness, clearness, and feasibility. Use "√" for **YES** and **leave the space blank** for **NO**.

Metric type	Metric	Correct	Clear	Feasible
Completeness	Q.M1.1.1: Customers or P.O were available on-site for face-to-face discussions during requirement elicitation			
	Q.M1.1.2: The scope of project was identified at the beginning of a project to create initial prioritized stack of requirements			
	Q.M1.1.3: The requirements were validated by customers in review meetings by using prototype/release			
	Q.M1.1.4: Requirements were prioritized and can be reprioritized by customers throughout the development			
	Q.M1.1.5: The development team was enabled to re-estimate the time and velocity of user stories			
	Q.M1.1.6: The requirements were written on cards in short statement			
	Q.M1.2.1: Model storming was performed (architecture, interface, data structure and algorithm)			
	Q.M1.2.2: The architecture designs were produced			
	Q.M1.2.3: The interface designs were produced			
	Q.M1.2.4: The data structure was produced			
	Q.M1.2.5: The algorithms were produced			
	Q.M1.2.6: Iteration modelling was performed at beginning of each iterations			

Q.M1.2.7: The designs were documented			
Q.M1.3.1: Reuse of software components was encouraged			
Q.M1.3.2: Detailed explanations on the functions and variables were included in the code			
Q.M1.3.3: The code was produced and integrated to system baseline iteratively and incrementally			
Q.M1.3.4: The software was delivered frequently with increments of features			
Q.M1.3.5: Customer involved with the team for giving immediate feedbacks			
Q.M1.3.6: The features with high priority were delivered first			
Q.M1.3.7: The software was deployed gradually in real environment			
Q.M1.3.8: The deliverable documentation were produced late			
Q.M1.4.1: Tests were automated			
Q.M1.4.2: Tests were performed continuously throughout the development			
Q.M1.4.3: Frequent integration tests were performed			
Q.M1.4.4: Unit tests were performed to ensure that all requirements were fulfilled			
Q.M1.4.5: User interfaces were tested			
Q.M1.4.6: Database regression testing were performed			
Q.M1.4.7: Customer wrote the user acceptance tests according to stories/features			
Q.M1.4.8: Acceptance tests were used to validate and verify user's requirements			
Q.M1.4.9: Results of the tests were documented			
Q.M1.4.10: Results from automated tests were compared to manual tests			
Q.M1.5.1: The project was started with a clear scope, goals and objectives			
Q.M1.5.2: Planning for the project was performed collaboratively with team members			
Q.M1.5.3: The current progress of the			

	iteration / sprint was revealed to everyone on sprint burn down chart			
	Q.M1.5.4: Customer and end-user involvement were monitored in project activity			
	Q.M1.5.5: The project plan was documented for in-hand problems			
Consistency	Q.M2.1.1: Appropriate procedure is used to handle frequently changing requirements			
	Q.M2.1.2: The requirements were documented by following a particular standard			
	Q.M2.2.1: Appropriate procedure was used to handle frequently changing designs			
	Q.M2.2.2: The design was documented by following a particular standard			
	Q.M2.2.3: Software designs were refactored frequently			
	Q.M2.2.4: Metaphor was used for determining architecture of the system			
	Q.M2.3.1: Appropriate procedure was used to ensure that the code was developed based on the requirements and design			
	Q.M2.3.2: Appropriate procedure was used to handle frequently changing code			
	Q.M2.3.3: Appropriate procedure was used to deliver the software releases to customers			
	Q.M2.3.4: Appropriate code integration strategy was followed			
	Q.M2.3.5: Appropriate coding/ interface/ database standards were followed			
	Q.M2.3.6: Team members had authority to make changes at any part of the code			
	Q.M2.3.7: Pair programming was performed			
	Q.M2.3.8: Failing unit tests were developed before the code was written (TDD)			
Q.M2.3.9: Rigorous code and database refactoring were implemented				
Q.M2.3.10: Code integration strategy				

	was established and revised			
	Q.M2.4.1: The testing results were documented by following a particular standard			
	Q.M2.4.2: Appropriate procedure was followed for implementing automated tests			
	Q.M2.4.3: Appropriate procedure was followed for implementing integration tests			
	Q.M2.4.4: Appropriate procedure was followed for implementing interface tests			
	Q.M2.4.5: Appropriate procedure was followed for implementing user acceptance tests			
	Q.M2.4.6: Appropriate procedure was followed for implementing database regression tests			
	M.Q2.5.1: Appropriate procedure was used to plan the project (estimation and work breakdown)			
	M.Q2.5.2: The project plan was documented by following a particular standard			
	M.Q2.5.3: Release meetings were conducted at the beginning of the project and each release to create release plan			
	M.Q2.5.4: Iteration meetings were conducted at the beginning of each iteration to plan the iteration			
	M.Q2.5.5: Daily stand-up meetings were conducted for daily plan			
	M.Q2.5.6: Continuous review meetings were conducted at end of each iterations to demonstrate the latest version of web application			
	M.Q2.5.7: Retrospectives were conducted at end of each iteration			
Accuracy	Q.M3.1.1: The requirements were gathered using a particular method			
	Q.M3.1.2: Appropriate tools were used to facilitate requirement gathering activities			
	Q.M3.1.3: A particular notation was used to represent the requirements			
	Q.M3.2.1: Software was designed by following a particular method			
	Q.M3.2.2: Appropriate tools were			

	used to facilitate design activities			
	Q.M3.2.3: A particular notation was used to represent the design			
	Q.M3.3.1: Appropriate tools were used for bug tracking			
	Q.M3.3.2: Appropriate programming language was used			
	Q.M3.4.1: Appropriate tools were used to facilitate testing activities			
	Q.M3.4.2: Appropriate techniques or methods were followed for the implemented tests			
	Q.M3.5.1: Appropriate tools were used to facilitate the planning activities			
Tailorability	Q.M4.1.1: Is the development of web application performed using the integration of XP and Scrum			
	Q.M4.1.2: Is the using of the web design method and the measurement mechanism performed with affecting the process performance			
	Q.M4.1.3: Is the integration of Scrum, XP and GOMM easy to be performed in your organization			
Flexibility	Q.M5.1.1: Is any team member can vary the process performance for a specific need			
	Q.M5.1.2: Is this variation performed without requiring affecting the process it self			
Compatibility	Q.M6.1.1: Is the development of web application performed by interact with more than one process			
	Q.M6.1.2: Is this interact done easily and clear			
Accessibility	Q.M7.1.1: Is there a strategic established for training in the organization			
	Q.M7.1.2: Is the determine of the training is the responsibility of the organization			
	Q.M7.1.3: Is there training tactical plan in the organization			
	Q.M7.1.4: is there a record of the organization training			
	Q.M7.1.5: Is there any way to assess the organization training			
	Q.M7.1.6: Is the process practitioner			

	can access the process electronically not by hard copy in training			
	Q.M7.1.7: Is the process described graphically not textually			
Applicability	Q.M8.1.1: Is there a define process for each project from start up until end			
	Q.M8.1.2: Is there a measurement program used for estimate and plan the project activity			
	Q.M8.1.3: Is the project managed based to a specific plan			
	Q.M8.1.4: Is there a contribute product, measures, and experience for the future project			
Changeability	Q.M9.1.1: is there a way to Determine requirement change Sources and Categories.			
	Q.M9.1.2: is there a strategy Established for requirement change			
	Q.M9.1.3: is there a way to Evaluate, Categorize, and Prioritize these changes			
	Q.M9.1.4: is the team Develop and implement change management Plans			
Supportability	Q.M10.1.1: Is there an agreement establish and maintain between the supplier and the organization for supporting the any item.			
	Q.M10.1.2: Is the selection of the suppliers based on their ability of satisfying a specific requirement			
	Q.M10.1.3: Is the acquired product from the supplier evaluated from the organization before accepting it			
	Q.M10.1.4: Is the organization ensure that the agreement satisfied before accepting the acquired product			

- If there are too many metrics, what type of metrics do you find important during the process? Please prioritize the goals of web application product (Process, Cost, Time, Quality, Productivity and Practice), and the process quality factors (Changeability, Applicability, Accessibility, Compatibility, Flexibility, Tailorability, Accuracy, Consistency and Completeness) according to their importance to the organization.

.....
.....
.....
.....
.....
.....

- Do you have any recommended metrics to improve the monitoring of web application product and process quality factors?

.....
.....
.....
.....
.....

1.3 Practices

This component consists of two types of practices, XP practices and Scrum practices. Nine cores XP practices were used in this methodology, namely: Collective ownership, TDD, Refactoring, Coding standards, Small release, Continuous integration, Metaphor, Simple design and Pair programming. In addition, four Scrum practices were included: first planning meeting Iteration review meeting, Daily meeting and Iteration planning meeting.

- a) Are the nine XP practices feasible to be used together in the development phase?

.....
.....
.....
.....
.....

- b) Is the use of the four Scrum meetings feasible to be used?

.....
.....
.....
.....
.....

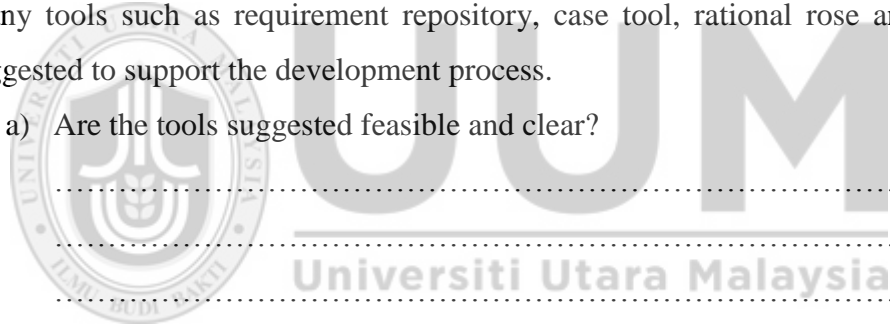
- c) Do the XP and Scrum practices are comprehensive to fulfill the application of agile principles?

.....
.....
.....
.....

1.4 Tools

Many tools such as requirement repository, case tool, rational rose and TDD are suggested to support the development process.

- a) Are the tools suggested feasible and clear?



.....
.....
.....

1.5 Team

The team structure of the proposed methodology consists of seven persons, including the customer. Each member has a specific role and responsibility. The members are master, one product owner, two programmers, one tester, customer and GOMM member.

- a) Are the roles and responsibilities of the team comprehensive and feasible?

.....
.....
.....

.....
.....

b) Are roles and responsibilities of the team correct and clear?

.....
.....
.....
.....

c) Can one person (GOMM member) perform the measurement process?

.....
.....
.....
.....

1.6 General overview

a) After reviewing the proposed methodology do you find extended based strategy steps for creating the methodology is clear and correct?

.....
.....
.....
.....

b) After reviewing the proposed methodology do you find the components feasible to be used for SSF?

.....
.....
.....
.....

c) Are the components consistent and well organized?

.....
.....
.....
.....
.....

d) Please state any suggestion or improvement that you may have.

.....
.....
.....



Appendix E
Domain expert questionnaire

**Reviewing the Proposed New Monitoring Oriented Agile Based Web
Application Development Methodology for Small Software Firms (D**

PHD Student: Moath Husni Altarawneh
School of Computing
College of Arts and Sciences
Universiti Utara Malaysia
Kedah, Malaysia
Tarawneh80@yahoo.com

The research aims to propose a new monitoring oriented agile based web application development methodology for small software firms. One of the objectives of this research is to verify the components of the methodology. This could be achieved through an expert review. Your answers to the following questions will serve as useful feedback on the methodology's comprehensiveness, understandability and feasibility. Your kind cooperation and participation in answering the questions is highly appreciated and will be treated as strictly confidential.

=====

The new methodology consists of five (5) components: activities, methods, practices, tools and team structure.

You are required to give answers related to each component.

1. Activities and methods

The proposed methodology process consists of two sides: development and measurement. Development process performed based on the combined XP and Scrum method and web design prototype. A measurement process performed based on the goal oriented monitoring method (GOMM).

1.1 Development activities: planning, development and integration

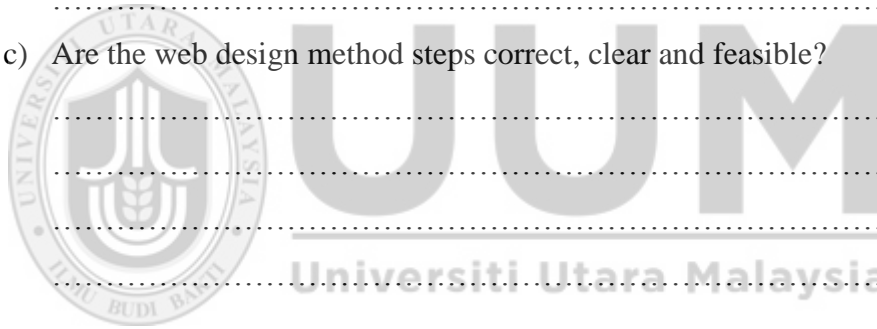
a) Are the activities of the development process correct, clear and feasible?

.....
.....
.....
.....
.....

b) Do the activities cover all the stages of building Web applications?

.....
.....
.....
.....
.....

c) Are the web design method steps correct, clear and feasible?



.....
.....
.....
.....
.....

d) Is it feasible to use the planning phase to perform the web design method?

.....
.....
.....
.....
.....

e) Are the requirement repository activities (save, reuse and trace requirements.)
clear and correct?

.....
.....
.....
.....
.....

1.2 Measurement activities: planning, definition and feedback

a) Are the measurement activities correct and clear?

.....
.....
.....
.....
.....

b) Are the measurement activities feasible to be used for SSF?



UUM
Universiti Utara Malaysia

.....
.....
.....
.....
.....

c) Do the measurement activities cover all stages of building Web applications?

.....
.....
.....
.....
.....

d) Please fill up the following table to determine the metric correctness, clearness and feasibility. Use "✓" for **YES** and **leave the space blank** for **NO**.

Metric type	Metric	Correct	Clear	Feasible
Process activities	M1.1.1.1: Number of product backlog items completed to date / Total Number of requirements planned.			
	M1.2.1.1: Number of LOC completed to date / Total Number of SLOC planned.			
	M1.2.1.2: Number of Web Pages to date / Total Number of Web Page planned.			
	M1.2.1.3: Total Number of internal links / Number of Web pages.			
	M1.3.1.1: Number of test completed to date / Total Number of test planned.			
	M1.3.1.2 number of testing line of code / total number lines of code			
Practices	PM1.1.1: Number of iteration planning meetings per one application.			
	PM1.2.1: Number of daily meetings per one application?			
	PM1.3.1: Number of review meetings done per one application?			
	PM 2.1.1: Number of programmers.			
	PM2.2.1: Number of tests completed to date /Total Number of tests planned.			
	PM2.2.2: Number of testing line of code / total number lines of code.			
	PM2.3.1: Number of lines of duplicated code removed / total line of code per iteration.			
	PM2.4.1: Adherence of coding standard (High, Low).			
	PM3.1.1: (Number of LOC of the first release - the LOC of the next release) / total NLOC			
	PM 3.2.1: Total number of line of code added, removed and updated) / total line of code of the previous iteration.			
	PM3.3.1: (Number of LOC of the current release - total LOC) / Total LOC			
	PM3.4.1: Number of meetings between development team and the client?			
	PM3.5.1 Number of team members who made changes in the code.			
Productivity	M2.1.1: Number of LOC for staff in month.			
Cost	M3.1.1: Dollar cost related to fix post to release problems.			
	M3.2.1: Number of dollars spent to date for activity i /Number of dollars			

	estimated for activity.			
	M3.3.1 Number of total dollars spent to date / Number of total dollars estimated.			
Quality	M4.1.1: Severity classification for each detected failure (fatal, major, minor and other).			
	M4.2.1: Number of iteration i defects / metric for size in sprint i(LOC).			
	M4.3.1: Number of pre-release defects of in iteration / (Number of pre-release + post-release defects).			
	M4.4.1: Number of defects / execution time.			
	M4.5.1: Effort in hours for locating each fault.			
	M4.5.2: Efforts in hours for fixing the fault.			
	M4.6.1 No. of page links/ total number of internal links (navigability)			
	M4.6.2 Response time			
	M4.6.3 Memory space			
	M4.7.1 dynamic pages/ total no. of pages (changeability) should be low			
	M4.7.2 dynamic testing LOC/ total LOC testability should be low			
	M4.7.3 1/ no of direct links (stability) should be high			
	Time	M5.1.1: Number SLOC of reusing code / Number of SLOC completed to date.		
M5.1.2: Number of reuse Web pages / total Web Pages number.				
M5.2.1: Elapsed time / estimated time.				

e) Please fill up the following table to determine the metric correctness, clearness, ease of use and applicability. Use "√" for **YES** and **leave the space blank** for **NO**.

Metric type	Metric no	Correct	Clear	Feasible
Completeness	Q.M1.1.1: Customers or P.O were available on-site for face-to-face discussions during requirement elicitation			
	Q.M1.1.2: The scope of project were identified at the beginning of project to create initial prioritized stack of requirements			
	Q.M1.1.3: The requirements were validated by customers in review meetings by using			

	prototype/release			
	Q.M1.1.4: Requirements were prioritized and can be reprioritized by customers throughout the development			
	Q.M1.1.5The development team was enabled to re-estimate the time and velocity of user stories			
	Q.M1.1.6:The requirements were written on cards in short statement			
	Q.M1.2.1:Model storming was performed (architecture, interface, data structure and algorithm)			
	Q.M1.2.2: The architecture designs were produced			
	Q.M1.2.3: The interface designs were produced			
	Q.M1.2.4: The data structure was produced			
	Q.M1.2.5:The algorithms were produced			
	Q.M1.2.6:Iteration modelling was performed at beginning of each iterations			
	Q.M1.2.7:The designs were documented			
	Q.M1.3.1: Reuse of software components was encouraged			
	Q.M1.3.2: Detailed explanations on the functions and variables were included in the code			
	Q.M1.3.3:The code was produced and integrated to system baseline iteratively and incrementally			
	Q.M1.3.4:The software was delivered frequently with increments of features			
	Q.M1.3.5:Customer involved with the team for giving immediate feedbacks			
	Q.M1.3.6:The features with high priority were delivered first			
	Q.M1.3.7:The software was deployed gradually in real			

	environment			
	Q.M1.3.8: The deliverable documentation were produced late			
	Q.M1.4.1: Tests were automated			
	Q.M1.4.2: Tests were performed continuously throughout the development			
	Q.M1.4.3: Frequent integration tests were performed			
	Q.M1.4.4: Unit tests were performed to ensure that all requirements were fulfilled			
	Q.M1.4.5: User interfaces were tested			
	Q.M1.4.6: Database regression testing were performed			
	Q.M1.4.7: Customer wrote the user acceptance tests according to stories/features			
	Q.M1.4.8: Acceptance tests were used to validate and verify user's requirements			
	Q.M1.4.9: Results of the tests were documented			
	Q.M1.4.10: Results from automated tests were compared to manual tests			
	Q.M1.5.1: The project was started with a clear scope, goals and objectives			
	Q.M1.5.2: Planning for the project was performed collaboratively with team members			
	Q.M1.5.3: The current progress of iteration/sprint was revealed to everyone on sprint burn down chart			
	Q.M1.5.4: Customer and end-user involvement were monitored in project activity			
	Q.M1.5.5: The project plan was documented for in-hand problems			
Consistency	Q.M2.1.1: Appropriate procedure is used to handle frequently changing			

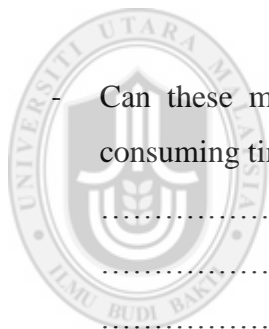
	requirements			
	Q.M2.1.2: The requirements were documented by following a particular standard			
	Q.M2.2.1: Appropriate procedure was used to handle frequently changing designs			
	Q.M2.2.2: The design was documented by following a particular standard			
	Q.M2.2.3: Software designs were refactored frequently			
	Q.M2.2.4: Metaphor was used for determining architecture of the system			
	Q.M2.3.1: Appropriate procedure was used to ensure that the code were developed based on the requirements and design			
	Q.M2.3.2: Appropriate procedure was used to handle frequently changing code			
	Q.M2.3.3: Appropriate procedure was used to deliver the software releases to customers			
	Q.M2.3.4: Appropriate code integration strategy was followed			
	Q.M2.3.5: Appropriate coding/ interface/ database standards were followed			
	Q.M2.3.6: Team members had authority to make changes at any part of the code			
	Q.M2.3.7: Pair programming was performed			
	Q.M2.3.8: Failing unit tests were developed before the code was written (TDD)			
	Q.M2.3.9: Rigorous code and database refactoring were implemented			
	Q.M2.3.10: Code integration strategy was established and revised			
	Q.M2.4.1: The testing results were documented by following a particular standard			

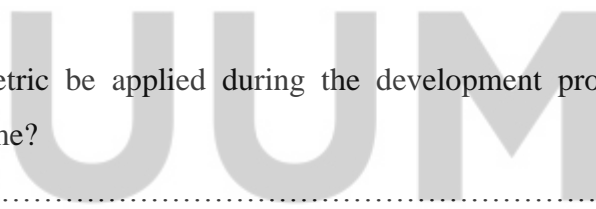
	Q.M2.4.2: Appropriate procedure was followed for implementing automated tests			
	Q.M2.4.3: Appropriate procedure was followed for implementing integration tests			
	Q.M2.4.4: Appropriate procedure was followed for implementing interface tests			
	Q.M2.4.5: Appropriate procedure was followed for implementing user acceptance tests			
	Q.M2.4.6: Appropriate procedure was followed for implementing database regression tests			
	M.Q2.5.1: Appropriate procedure was used to plan the project (estimation and work breakdown)			
	M.Q2.5.2: The project plan was documented by following a particular standard			
	M.Q2.5.3: Release meetings were conducted at the beginning of project and each release to create release plan			
	M.Q2.5.4: Iteration meetings were conducted at the beginning of each iterations to plan the iteration			
	M.Q2.5.5: Daily stand-up meetings were conducted for daily plan			
	M.Q2.5.6: Continuous review meetings were conducted at end of each iterations to demonstrate the latest version of software			
	M.Q2.5.7: Retrospectives were conducted at end of each iteration			
Accuracy	Q.M3.1.1: The requirements were gathered using a particular method			
	Q.M3.1.2: Appropriate tools were used to facilitate requirement gathering activities			

	Q.M3.1.3: A particular notation was used to represent the requirements			
	Q.M3.2.1: Software was designed by following a particular method			
	Q.M3.2.2: Appropriate tools were used to facilitate design activities			
	Q.M3.2.3: A particular notation was used to represent the design			
	Q.M3.3.1: Appropriate tools were used for bug tracking			
	Q.M3.3.2: Appropriate programming language was used			
	Q.M3.4.1: Appropriate tools were used to facilitate testing activities			
	Q.M3.4.2: Appropriate techniques or methods were followed for the implemented tests			
	Q.M3.5.1: Appropriate tools were used to facilitate the planning activities			
Tailorability	Q.M4.1.1: Is the development of web application performed using the integration of XP and Scrum			
	Q.M4.1.2: Is the using of the web design method and the measurement mechanism performed with affecting the process performance			
	Q.M4.1.3: Is the integration of Scrum, XP and GOMM easy to be performed in your organization			
Flexibility	Q.M5.1.1: Is any team member can vary the process performance for a specific need			
	Q.M5.1.2: Is this variation performed without requiring affecting the process it self			
Compatibility	Q.M6.1.1: Is the development of web application performed by interact with more than one			

	process			
	Q.M6.1.2: Is this interact done easily and clear			
Accessibility	Q.M7.1.1: Is there a strategic established for training in the organization			
	Q.M7.1.2: Is the determine of the training is the responsibility of the organization			
	Q.M7.1.3: Is there training tactical plan in the organization			
	Q.M7.1.4: is there a record of the organization training			
	Q.M7.1.5: Is there any way to assess the training organization			
	Q.M7.1.6: Is the process practitioner can access the process electronically, not by hard copy in training			
	Q.M7.1.7: Is the process described graphically not textually			
Applicability	Q.M8.1.1: Is there a define process for each project from start up until the end			
	Q.M8.1.2: Is there a measurement program used for estimate and plan the project activity			
	Q.M8.1.3: Is the project managed based on a specific plan			
	Q.M8.1.4: Is there a contribute product, measures, and experience for the future project			
Changeability	Q.M9.1.1: is there a way to Determine requirement change Sources and Categories.			
	Q.M9.1.2: is there a strategy Established for requirement change			
	Q.M9.1.3: is there a way to Evaluate, Categorize, and Prioritize these changes			
	Q.M9.1.4: is the team Develop			

	and implement change management Plans			
Supportability	Q.M10.1.1: Is there an agreement establish and maintain between the supplier and the organization for supporting the any item.			
	Q.M10.1.2: Is the selection of the suppliers based on their ability of satisfying a specific requirements			
	Q.M10.1.3: Is the acquired product from the supplier evaluated from the organization before accepting it			
	Q.M10.1.4: Is the organization ensure that the agreement satisfied before accepting the acquired product			




 Universiti Utara Malaysia

- Can these metric be applied during the development process without consuming time?

.....

.....

.....

.....

- If there are too many metrics, what type of metrics do you find important during the process? Please prioritize the goals of web application product (Process, Cost, Time, Quality, Productivity and Practice), and the process quality factors (Changeability, Applicability, Accessibility, Compatibility, Flexibility, Tailorability, Accuracy, Consistency and Completeness) according to their importance to the organization.

.....

.....

.....

.....

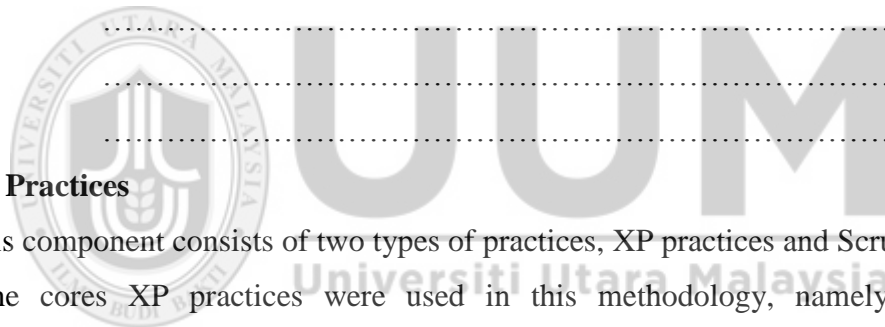
.....
.....
.....

- Is the time of applying these metrics during development appropriate?

.....
.....
.....
.....

- Do you have any recommended metrics to improve the monitoring of web application product and process quality factors?

.....
.....
.....
.....



1.3 Practices

This component consists of two types of practices, XP practices and Scrum practices. Nine cores XP practices were used in this methodology, namely: Collective ownership, TDD, Refactoring, Coding standards, Small release, Continuous integration, Metaphor, Simple design and Pair programming. In addition, four Scrum practices were included: first planning meeting Iteration review meeting, Daily meeting and Iteration planning meeting.

- a) Are the nine XP practices feasible to be used together in the development phase?

.....
.....
.....
.....

- b) Is the use of the four Scrum meetings feasible?

.....
.....
.....
.....
.....

1.4 Tools

Many tools such as requirement repository, case tool, rational rose and TDD are suggested to support the development process.

- a) Are the tools suggested feasible and clear?

.....
.....
.....
.....
.....

1.5 Team

The team structure of the proposed methodology consists of seven persons, including the customer. Each member has a specific role and responsibility. The members are master, one product owner, two programmers, one tester, customer and GOMM member.

- a) Are the roles and responsibilities of the team comprehensive and feasible?

.....
.....
.....
.....

- b) Are roles and responsibilities of the team correct and clear?

.....
.....
.....
.....

- c) Can one person (GOMM member) do the measurement process?

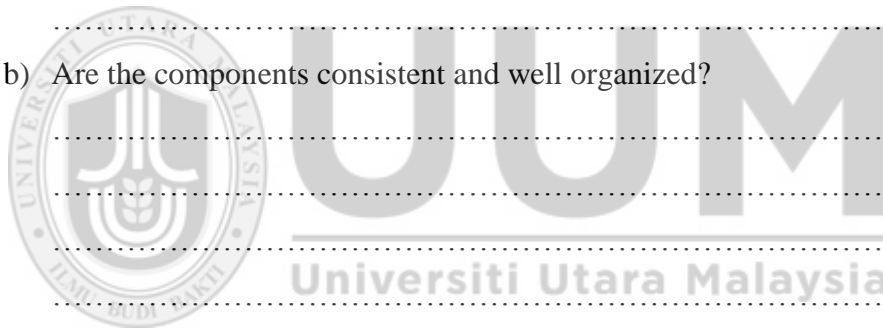
.....
.....
.....
.....
.....

1.6 General overview

a) After reviewing the proposed methodology do you find the components feasible to be used for SSF?

.....
.....
.....
.....

b) Are the components consistent and well organized?



.....
.....

c) Please state any suggestion or improvement that you may have.

.....
.....
.....
.....
.....
.....
.....

Appendix F

Questions Objective, Content and Source

Sections	Objectives	Questions	Contents	Sources
I: Respondents background	To assess the qualification of respondents	1	Position in company	Baharom(2006), El-Sheikh and Tarawneh(2007).
		2	Positions activities	El-Sheikh and Tarawneh (2007)
		3	Years of experience	Baharom (2006), El-Sheikh and Tarawneh (2007) .
II: Organisation background	To study the organizations background	1	Type of organization	Baharom (2006).
		2	Sector of organization (yes or no)	Baharom (2006).
		3	Organization size	El-Sheikh and Tarawneh, (2007)
III: Development and measurement practices	To investigate the software development practices in small software firms	1	Application type	
		2	Philosophy type	
		3	Development method used	Baharom (2006).
		4	Development methods that developers familiar with	
		5	Prototyping method used (yes or no)	
		6	Type of prototyping	Baharom (2006).
		7	Requirement collection method	Baharom (2006).
		8	Requirement analysis method	Baharom (2006).

		9	Requirement specification notation	Baharom (2006).	
		10	Programming language	Baharom (2006).	
		11	Testing type	Baharom (2006).	
		12	Testing process stage	Baharom (2006).	
		13	Reasons of not using any method		
		14	Encourage software reuse	Baharom (2006).	
		15	Software reuse type	Baharom (2006).	
		16	Quality assurance activities	El-Sheikh and Tarawneh, (2007)	
		17	Who perform quality assurance?	El-Sheikh and Tarawneh, (2007)	
	To investigate the software measurement practices in small software firms	18	Performing measurement yes or no		
		19	Measurement stage		
		20	Metrics type		
		21	Measurement method		
		22	Why not using measurement		
	III: Web application development and measurement practices	To investigate the current web application development and measurement practices in small software firms	1	Time pressure	El-Sheikh and Tarawneh (2007), McDonald and Welland (2001a)
			2	Process role and responsibilities	El-Sheikh and Tarawneh (2007), McDonald and Welland (2001a)
			3	Minimum design and quick prototype	El-Sheikh and Tarawneh (2007)
4			Each project has manager	El-Sheikh and Tarawneh, (2007), McDonald and Welland (2001a)	
5			Budget estimation	El-Sheikh and Tarawneh (2007), McDonald and Welland (2001a)	
6			Requirement source(user or manager)	El-Sheikh and Tarawneh (2007), McDonald and Welland (2001a)	
7			Design notation	El-Sheikh and Tarawneh (2007), McDonald and Welland (2001a)	

		8	Testing case generated based on the requirement specifications	El-Sheikh& andTarawneh (2007), McDonald and Welland (2001a)
		9	Testing process carried out by the development team	El-Sheikh and Tarawneh (2007), McDonald and Welland (2001a)
		10	Quality management and standard	El-Sheikh& andTarawneh, (2007), McDonald and Welland (2001a)
		11	Testing conducted by user under the guidance of quality assurance member	El-Sheikh& and Tarawneh (2007), McDonald and Welland (2001a)
		12	Change management	El-Sheikh and Tarawneh (2007), McDonald and Welland (2001a)
		13	Change control functions for each project	El-Sheikh and Tarawneh (2007), McDonald and Welland (2001a)
		14	Web application Estimation size procedure (LOC)	El-Sheikh and Tarawneh (2007), McDonald and Welland (2001a)
		15	Effort, size and cost procedure or method	El-Sheikh and Tarawneh (2007), McDonald and Welland (2001a)
		16	Training program	El-Sheikh and Tarawneh (2007), McDonald & Welland (2001a)
		17	Awareness of web application state of the art	El-Sheikh and Tarawneh (2007), McDonald & Welland (2001a)

Appendix G

Practices

Development Practices

The new methodology concentrates on the development practices that emphasis on building high quality Web application in small software firms. Thus, these practices should take into account Web application characteristics and small software firm limitations. As a result set of recommended development practices must take place during the development process:

- Use an iterative development process for small teams to cope with time pressure and requirement changing environment (Scrum).
- Use the Scrum product backlog for collecting the requirements.
- Use a simple (conceptual) design method that extracted from the current Web design methods.
- Use several XP development practices to ensure short life cycle, respond to change, test all Web application components and simple design. These practices are categorized into two types: core practices and supported practices.

Core XP Practices: these practices called core because they used from all previous studies that combine between XP and Scrum. These practices are:

- **Pair Programming:** This practice consists in having two programmers working simultaneously on the same computer.
- **TDD:** All implemented features must be covered by unit tests, which must all be always satisfied, in an effort to eliminate unit-level and regression bugs during development.
- **Refactoring:** aims to simplify the implemented code by removing ambiguity and redundancy, improving communication and adding flexibility.
- **Coding standards:** following the Coding standard practice allows programmers to interpret of the executed code in an easy way.

Supported XP practices: this type of practices found to be very important to fulfill all the agile principles and need to be integrated in the proposed methodology to ensure that the process still agile. These practices are:

- Continuous Integration: After a new feature is implemented or the code is adjusted, and all tests are successfully executed, a new release should be created reflecting all the changes.
- Metaphor: System is created by a set of Metaphors between the client and programmers, who allow describing features to be implemented, by creating a common vision of the client and the technical team on how the product should work.
- Small release: system version released daily, but at least monthly.
- Simple Design: The architecture and the code (including the unit tests) should be as simple as possible. No need for complexity and extra coding.
- Collective Ownership: Each team member is encouraged to perform all necessary changes in the code. Thus, all team members are owners of the code. This practice avoids unnecessary waits for third party changes in the code.

Management Practices

These practices were extracted from the previous studies and described as a core Scrum practices to be used in the combination method. These practices are:

- Scrum Master: is responsible to ensure that project is performed according to the practices, rules and values of Scrum on one hand. On the other hand, ensure that project progress as planned.
- First planning meeting: The first planning meeting is performed just once per the Web application. The master, the PO and the DT select a set of items of the Product Backlog to be implemented during the development process. At this meeting, the simple design method should be created to be composed into tasks to be performed by the DT. These tasks will integrate the iteration backlog.
- Daily Do (iteration) meeting: this meeting will be held daily by the DT, lasting 15 minutes or less. These meetings aim to analyze the progress of the project and the unexpected issues that may delay the project, by identifying

the work undertaken since the last meeting and the work to be performed until the next one.

- Do (Iteration) planning meeting: this meeting conducted by the master, development team and PO to determine the selected requirements for the next iteration.
- Do (Iteration) Review Meeting: a review meeting is held at the end of each iteration. This meeting will be attended by all master, PO and DT. Full explanations of all new release features are performed. Results related to this iteration will be presented to the management.

Measurement Practices

This set of practices is to ensure that the measurement program is performed in the right way in order to monitor the quality of product and process. These practices are:

- The measurement mechanism should measure all the development process phases and practices.
- A measurement mechanism should be goal oriented.
- The measurement mechanism should use qualitative and quantitative metrics.
- GOMM defines particular goals, refine these goals into questions and provide metrics to answer the desired questions.
- Goals prioritizing practice: Development team should priorities the measurement goals based on the organization and user demands.
- Two developers assign as GOMM members for performing the measurement process (one for collecting the data and the other for analyze the results and print the feedback report).
- Self-preparing data: Each team member should prepare the data he owns for the GOMM member to accelerate the data collection process.

Appendix H

Tools

Phase	Tool name	Aim
Planning phase	User story	Collecting requirements
	Web design prototype	Support the design phase
	Requirement repository	Reuse any existing code or model for the next iteration
Development phase	Rational rose	Supporting the analysis activity
	Code base	Saving the code
	ArgoUWE	Supporting the design
	Casper SJ	Supporting the testing
	Burn down chart tool	Supporting the daily meeting and Do (iteration) review meeting
	Task board	Support the daily and Do (iteration) review meetings
Integration and maintenance phase	Requirement repository	Save the increment to the repository and trace the requirement status
	Burn down chart task board	Declaring the final version of web application



UUM

Universiti Utara Malaysia

Appendix I

Quantitative metric check list

Phase	Activity	Metric #	Inputs	Calculation	Remarks	Outputs	Data owner
Plan	Identifying the product backlog items		<ul style="list-style-type: none"> - #Product backlog items, - Estimated LOC - Estimated cost - Estimated time - Estimated #of Web pages 				
	F.Prioritize the items G. Split the items H. Estimate the items	PM1.1.1	# of Do items number	# of Do items number			Master and PO
Do	Design	M1.2.1.3	<ul style="list-style-type: none"> - #of internal links - #of web pages 	#of internal links / #of web pages	#Internal links: total number of internal links, not including dynamically generated links.		Programmer
		M4.6.1	<ul style="list-style-type: none"> - # of page links - Total # of links 	# of page links / Total # of links	0.3 That means this page 30% navigable.		
		M4.7.1	<ul style="list-style-type: none"> - # of dynamic pages - Total # of pages 	# of dynamic pages / total # of pages (low)	0.5 This reflects the changeability of the product it means its 50% changeable and it should be low. The low changeability is,		

				the better.		
	M4.7.3	- # of direct links	1/# of direct links	0.5 means that this product is 50% stable and should be and should be high.		
	M5.1.2	- #of reused web pages - Total #of web pages	#of reused web pages/Total #of web pages	0.44 means that 44% of the web pages in this product were reused from previous applications.		
Code	M1.2.1.1	- LOC completed to date - Total LOC	LOC completed to date / Total LOC	0.70 means that the progress of the coding is 70 %.		Programmer
	M1.2.1.2	- # of web pages to date - Total # of web pages	# of web pages to date /Total # of web pages	0.55 means that 55 % of the webpages were created by the team.		
	M4.5.1	- Effort in hour for locating each fault	Effort in hour for locating each fault	4 means it takes 4 hours to locate the fault.		
	M4.5.2	- Effort in hour for fixing each fault	Effort in hour for fixing each fault	3 mean it takes three hours to fix the specific fault.		
	M5.1.1	- # of reused LOC - Total LOC	# of reused LOC / Total LOC	0.35 means that the team reused 35% of their code.		
	M2.1.1	- # of KLOC for the programmer in the month	# of KLOC for the programmer in the month	30 means the programmer has produced 30000 lines of code monthly.		
	PM3.5.1	- # of team members who made changes on the code	# of team members who made changes on the code	2 means two members have the power to change the code.		

Test	M1.3.1.1	<ul style="list-style-type: none"> - # of test completed to date - Total # of planned test 	# of test completed to date / Total # of planned test	0.45 means that 45% of the planned tests were completed.		Tester
	PM2.1.1	<ul style="list-style-type: none"> - #of duplicated LOC removed - Total LOC 	#of duplicated LOC removed / Total LOC	0.30 means that 30% of the code is removed as duplicated code. That means the ratio should be low to get quality code.		
Daily reviewing	M1.1.1.1	<ul style="list-style-type: none"> - # of product backlog items completed to date - Total # of product backlog planned 	# of product backlog items completed to date / Total # of product backlog planned	0.7 means that 70% of the product item completed based on the planned no. of items.		PO
	M3.2.1	<ul style="list-style-type: none"> - # of dollars spent for each activity - # of dollars estimated for the activity 	# of dollars spent for each activity / # of dollars estimated for the activity	300/500 means that 60% of the budget of the this activity was consumed.		
	M3.3.1	<ul style="list-style-type: none"> - Total # of Dollars spent - Estimated cost in dollars 	Total # of Dollars spent /Estimated cost in dollars	600/1000 means that 60% of the budget consumed.		
	M5.3.1	<ul style="list-style-type: none"> - # of completed product backlog items - Total # of product backlog items 	# of completed product backlog items / Total # of items	0.5 means that 50% of the items were completed.		

		PM1.2.1	- # of daily meeting	# of daily meeting per one application	15 mean the meeting conducted 15 times.		
	Iteration reviewing	M3.1.1	- Dollars spent to fix post to release problems	Dollars spent to fix post to release problems	100		Master
		M4.2.1	- # of Do defects - LOC for the DO	# of Do defects / LOC for the DO	Should close to zero for the better execution.		
		M4.3.1	- # of pre-release defect of the DO - # of post-release defects of the DO	# of pre-release defect of the DO/ # of pre-release+ post-release defects of the DO	The result will be ranged from 0 to 1, and the perfect result should be nearer to 1 because that means the post defects were reduced.		
		M4.4.2	- Mean time to find defect	Mean time to find defect	If it =200, means that one failure can be expected every 200 time units		
		M4.4.3	- Mean time between two defects	Mean time between two defects	30 indicates that once the failure occurs, the next failure is expected to occur only after 30 hours.		
		M4.4.4	- Mean time to recover	Mean time to recover	The average time it takes to track the errors causing the failure & to fix them.		
		M5.2.1	- Elapsed time - Estimated time	Elapsed time / Estimated time	0.3 means that the product consumed the 30 % of the time		
		PM1.3.1	- # of review meeting	# of review meeting per one application	5 means that reeving meeting conducted I this application 5 times.		
		PM3.4.1	- # of meeting between DT and client	# of meeting between DT and client	4 means the team conducted 5 meetings with client.		
		M4.4.1	- # of defects	# of defects /	0.04 means that the defects occurs 4 times		

			- Execution time	Execution time	in 100 units of time.		
Act	Save the increment to the repository	PM3.1.1	- LOC of the first release - LOC of the current release - Total LOC	(LOC of the first release - LOC of the current release) / Total LOC	Reflects the small release practices and it should be low for example -0.02 means this release smaller the previous with 2%.		PO
	Integrate with the system	PM3.2.1	- LOC added, removed and updated - LOC of the iteration	LOC added, removed and updated / LOC of the iteration	This percent is good to be nearer to 100% as it indicates that the continuous integration used in the process.		Programmer
		PM3.3.1	- # of LOC of the current release - Total LOC	# of LOC of the current release – Total LOC	2000 means that the difference LOC between the total and current LOC is 2000.		
	Final release		-				

Appendix J

Qualitative metrics

Completeness metrics

Activities	Metrics	0	1	2	3	4	Data owner
Requirement completeness	Q.M1.1.1: Customers or P.O was available on-site for face-to-face discussions during the requirement elicitation						PO
	Q.M1.1.2: The scope of project was identified at the beginning of a project to create initial prioritized product backlog items						
	Q.M1.1.3: The requirements were validated by customers in review meetings by using prototype/release						
	Q.M1.1.4: Requirements were prioritized and can be reprioritized by customers throughout the development						
	Q.M1.1.5: The development team was enabled to re-estimate the time and velocity of user stories						
	Q.M1.1.6: The requirements were written on cards in a short statement						
Design completeness	Q.M1.2.1: Model storming was performed (architecture, interface, data structure and algorithm)						Programmer
	Q.M1.2.2: The architecture designs were produced						
	Q.M1.2.3: The interface designs were produced						
	Q.M1.2.4: The data structure was produced						
	Q.M1.2.5: The algorithms were produced						
	Q.M1.2.6: Iteration modeling was performed at the beginning of each iteration						
	Q.M1.2.7: The designs were documented						
Coding completeness	Q.M1.3.1: Reuse of software components was encouraged						Programmer
	Q.M1.3.2: Detailed explanations of the functions and variables were included in the code						
	Q.M1.3.3: The code was produced and integrated to system baseline iteratively and incrementally						
	Q.M1.3.4: Web application was delivered frequently with increments of features						
	Q.M1.3.5: Customer involved with the team for giving immediate feedbacks						
	Q.M1.3.6: The features with high priority were delivered first						
	Q.M1.3.7: Web application was deployed gradually in real environment						
	Q.M1.3.8: The deliverable documentation was produced late						
Testing completeness	Q.M1.4.1: Tests were automated						Tester
	Q.M1.4.2: Tests were performed continuously throughout the development						
	Q.M1.4.3: Frequent integration tests were performed						

	Q.M1.4.4: Unit tests were performed to ensure that all requirements were fulfilled						
	Q.M1.4.5: User interfaces were tested						
	Q.M1.4.6: Database regression testing was performed						
	Q.M1.4.7: Customer (P.O) wrote the user acceptance tests according to stories/features						
	Q.M1.4.8: Acceptance tests were used to validate and verify user's requirements						
	Q.M1.4.9: Results of the tests were documented						
	Q.M1.4.10: Results from the automated tests were compared to the manual tests						
Project management completeness	Q.M1.5.1: The project was started with a clear scope, goals and objectives						Master
	Q.M1.5.2: Planning for the project was performed collaboratively with team members						
	Q.M1.5.3: The current progress of iteration was revealed to everyone on iteration burn down chart						
	Q.M1.5.4: Customer and end-user involvement were monitored in project activity						
	Q.M1.5.5: The project plan was documented						

Consistency metrics

Activities	Metrics	0	1	2	3	4	Data owner
Requirement consistency	Q.M2.1.1: Appropriate procedure is used to handle frequently changing requirements						PO
	Q.M2.1.2: The requirements were documented by following a particular standard						
Design consistency	Q.M2.2.1: Appropriate procedure was used to handle frequently changing designs						Programmer
	Q.M2.2.2: The design was documented by following a particular standard						
	Q.M2.2.3: Web application designs were refactored frequently						
	Q.M2.2.4: Metaphor was used for determining the architecture of the system						
Coding consistency	Q.M2.3.1: Appropriate procedure was used to ensure that the code was developed based on the requirements and design						Programmer
	Q.M2.3.2: Appropriate procedure was used to handle frequently changing code						
	Q.M2.3.3: Appropriate procedure was used to deliver the Web application releases to customers						
	Q.M2.3.4: Appropriate code integration strategy was followed						
	Q.M2.3.5: Appropriate coding/ interface/ database standards were followed						
	Q.M2.3.6: Team members had authority to make changes in any part of the code						
	Q.M2.3.7: Pair programming was performed						

	Q.M2.3.8: Failing unit tests were developed before the code was written (TDD)							
	Q.M2.3.9: Rigorous code and database refactoring were implemented							
	Q.M2.3.10: Code integration strategy was established and revised							
Testing consistency	Q.M2.4.1: The testing results were documented by following a particular standard							Tester
	Q.M2.4.2: Appropriate procedure was followed for implementing automated tests							
	Q.M2.4.3: Appropriate procedure was followed for implementing integration tests							
	Q.M2.4.4: Appropriate procedure was followed for implementing interface tests							
	Q.M2.4.5: Appropriate procedure was followed for implementing user acceptance tests							
	Q.M2.4.6: Appropriate procedure was followed for implementing database regression tests							
Project management consistency	Q.M 2.5.1: Appropriate procedure was used to plan the project (estimation and work breakdown)							Master
	Q.M 2.5.2: The project plan was documented by following a particular standard							
	Q.M 2.5.3: Release meetings were conducted at the beginning of the project and each release to create release plan							
	Q.M 2.5.4: Iteration meetings were conducted at the beginning of each iteration to plan the iteration							
	Q.M 2.5.5: Daily stand-up meetings were conducted for daily plan							
	Q.M 2.5.6: Continuous review meetings were conducted at the end of each iteration to demonstrate the latest version of the Web application							
	Q.M 2.5.7: Retrospectives were conducted at the end of each iteration							

Accuracy metrics

Activities	Metrics	0	1	2	3	4	Data owner
Requirement accuracy	Q.M3.1.1: Requirements were gathered using customer card						PO
	Q.M3.1.2: Appropriate tools were used to facilitate requirements gathering activities						
	Q.M3.1.3: A particular notation was used to represent the requirements						
Design accuracy	Q.M3.2.1: Web application was designed by following Web design method steps						Programmer
	Q.M3.2.2: Appropriate tools were used to facilitate design activities						
	Q.M3.2.3: A particular notation was used to represent the design						
Coding	Q.M3.3.1: Appropriate tools were used for bug tracking						Programmer

accuracy	Q.M3.3.2: Appropriate programming language was used							
Testing accuracy	Q.M3.4.1: Appropriate tools were used to facilitate testing activities							Tester
	Q.M3.4.2: Appropriate techniques or methods were followed for the implemented tests							
Project management accuracy	Q.M3.5.1: Appropriate tools were used to facilitate the planning activities							Master

Tailorability metrics

Metrics	0	1	2	3	4	Data owner
Q.M4.1.1: Is the development of the Web application performed using the integration of the XP and Scrum?						Master
Q.M4.1.2: Is the using of the Web design method and measurement process performed without affecting the process performance?						
Q.M4.1.3: Is the integration of the Scrum, XP and GOMM easy to be performed in the organization?						

Flexibility metrics

Metrics	0	1	2	3	4	Data owner
Q.M5.1.1: Is any team member can vary the process performance for a specific need?						Master
Q.M5.1.2: Is this variation performed without requiring affecting the process itself?						

Compatibility metrics

Metrics	0	1	2	3	4	Data owner
Q.M6.1.1: Is the development of Web application performed by interacting with measurement and development process.						Master
Q.M6.1.2: Is this interaction between the team and the process done easily and clearly						

Accessibility metrics

Metrics	0	1	2	3	4	Data owner
Q.M7.1.1: Is there a strategic established for training in the organization?						Master
Q.M7.1.2: Is determining of the training is the responsibility of the organization?						
Q.M7.1.3: Is there any training and tactical plan in the organization?						
Q.M7.1.4: Is there a record of the training organization?						
Q.M7.1.5: Is there any way to assess the training organization?						
Q.M7.1.6: Is the process practitioner able to access the training process electronically, not by hard copy?						
Q.M7.1.7: Is the process described graphically not textually?						

Applicability metrics

Metrics	0	1	2	3	4	Data owner
Q.M8.1.1: Is there a defined process for each project from start up until the end?						PO
Q.M8.1.2: Is there a measurement mechanism used to estimate and plan the project activities?						
Q.M8.1.3: Is the project managed based on a specific plan?						
Q.M8.1.4: Is the contributed product, modules, code and measures saved to be used for the future project?						

Changeability metrics

Metrics	0	1	2	3	4	Data owner
Q.M9.1.1: is there a way to determine the change requirement sources and categories?						PO
Q.M9.1.2: Is there a strategy established for change requirement?						
Q.M9.1.3: Is there a way to evaluate, categorize, and prioritize these changes?						
Q.M9.1.4: Is the team going to develop and implement change management plans?						

Supportability metrics

Metrics	0	1	2	3	4	Data owner
Q.M10.1.1: Is there an agreement established and maintained between the supplier and the organization for supporting any item?						PO
Q.M10.1.2: Is the selection of the suppliers based on their ability of satisfying a specific requirement?						
Q.M10.1.3: Is the acquired product from the supplier evaluated from the organization before accepting it?						
Q.M10.1.4: Is the organization ensures that the agreement satisfied before accepting the acquired product?						

Appendix K

Validation form

Note: Please give a score from 1 to 5 for the following items where, 1 = strongly disagree, 2 = disagree, 3=don't know, 4 = agree and 5 = strongly agree.

Gain Satisfaction

Items	1	2	3	4	5
Decision support satisfaction: is the MOGWD methodology helps the management to take a well-defined decision based on the process and product monitoring?					
Comparison with the current development method: is the MOGWD methodology better than the old development that you used in terms of the structure and achieve results?					
Clarity (clear and illuminate the process): Is the MOGWD process clear to the development team, where each phase clearly presents the required inputs, outputs, methods or practices, and activities?					
Task Appropriateness: is the phases and activities that presented in the MOGWD methodology appropriate for developing and monitoring web application in your company, and is the flow of the process presented in a systematic and effective way?					

Interface Satisfaction

Items	1	2	3	4	5
Internally consistent: the MOGWD methodology is internally consistent?					
Organization (well organized): the components of MOGWD methodology well organized and structured that makes the process is easy to perform?.					
Appropriate for audience: is the MOGWD methodology appropriate for the audience. Those audiences are referred to the development and the monitoring team in the Small Software firms?					
Presentation: is the results presented by performing the MOGWD process produced in a readable and useful format?					

Task Support Satisfaction

Items	1	2	3	4	5
Ability to produce expected results: is the MOGWD methodology able to produce expected results?					
Completeness (adequate or sufficient): is the MOGWD methodology adequate and sufficient for developing web application in your organization.					
Ease of implementation: is the process of the MOGWD methodology easy to implement?					

Perceived Usefulness

Items	1	2	3	4	5
Using MOGWD methodology enables you to accomplish your tasks more quickly.					
Using MOGWD methodology improve the performance of your work					
Using MOGWD methodology makes performing your tasks easier					
MOGWD methodology is useful to your work					
Using MOGWD methodology increases your productivity					

Perceived ease of use

Items	1	2	3	4	5
Learning the MOGWD methodology is easy for you					
Do you find it easy to use MOGWD methodology to do what want to do					
The MOGWD methodology is flexible to interact with					
Your interactions with the MOGWD methodology clear and understandable					
It is easy for you to become skillful in using MOGWD methodology					
The MOGWD methodology is easy to use					



UUM
Universiti Utara Malaysia

Appendix L

MO-PT

The main page of the MO-PT consists of three tabs and one button as shown Figure 1.



Figure 1. Main page

The home tab returns the user to the main page, the overview tab gives a summary about the MOGWD methodology and the contact tab gives information about the author. On the other hand, the start button leads to the login page. Each user of the MO-PT has authentication to access the system based on his identity. The tool gives each development team member a specific user name and password.

The users of the MO-PT are master, product owner, programmer and tester. The tool starts working after the master logged in as shown in Figure 2.

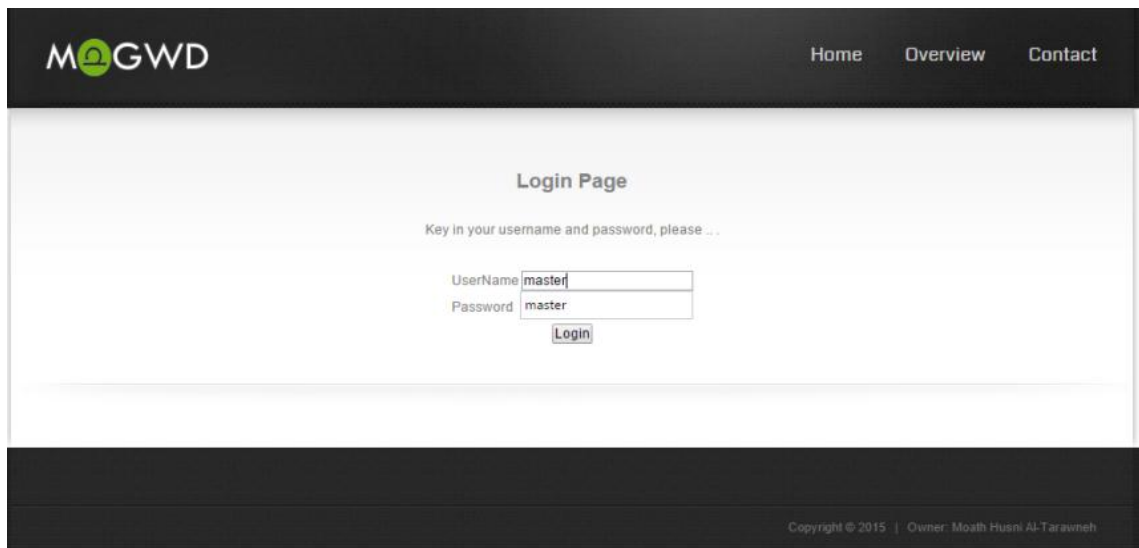


Figure 2. Login page

After login the team member will enter the data for the metrics that he/she responsible for. The MO-PT clarifies the team member and the data that he is responsible to enter as shown in Table 1.

Table 1. Team member activities in MO-PT

Phase	Activity	The data	Type of data		The team member involved
			Quantitative	Qualitative	
Plan	Planning meeting	Requirement completeness		√	PO
		Requirement consistency		√	
		Requirement accuracy		√	
		Applicability		√	
		Changeability		√	
		Supportability		√	
Do	Design	Design quantitative metrics	√		Programmer
		Design completeness		√	
		Design consistency		√	
		Design accuracy		√	
	Code	Code quantitative metrics	√		Programmer
		Code completeness		√	
		Code consistency		√	
		Code accuracy		√	
	Testing	Testing quantitative metrics	√		Tester
		Testing completeness		√	

		Testing consistency		√	
		Testing accuracy		√	
	Daily reviewing	Daily reviewing quantitative metrics	√		PO
	Iteration review meeting	Project management, quantitative metrics	√		Master
		Project management completeness		√	
		Project management consistency		√	
		Project management accuracy		√	
		Tailorability		√	
		Flexibility		√	
		Compatibility		√	
		Accessibility		√	
Act	Save the increment	Save the increment quantitative metrics	√		PO
	Integrate the increment	Integrate the increment Quantitative metrics	√		

Based on the above table, each team member knew the data that he owns and at what time he should enter into the system during the development of the system.

Plan phase

As mentioned before, MO-PT will be used to support the monitoring process. The master will be the administrator of the MO-PT and he will be assisted by the GOMM member. In the plan phase, master is responsible for the following activities: create a new project, start new iteration, iteration reviewing, activate the current activities and view report as shown in Figure 3.

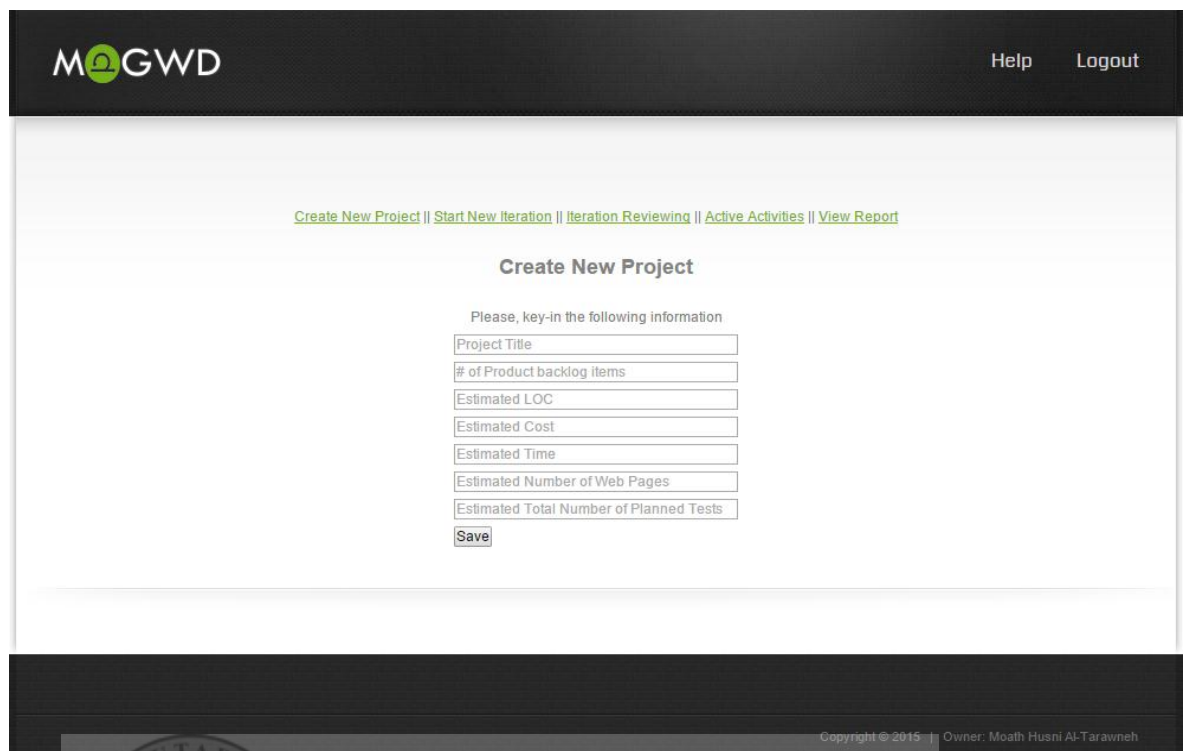


Figure 3. Master page

- **Create new project**

After the first planning meeting, the master required to access the MO-PT and create the project on the system. This will be done by clicking on create new project tab and enter the important information related to the project. The information includes: project title, number of backlog items, estimated LOC, estimated cost, estimated time, estimated number of pages, estimated total number of tests. Then click the save button.

- **Start a new iteration**

The master will start the iteration after the iteration planning meeting. After creating the project on the system the master should start a new iteration by entering the required information for the iteration, such as: the number Do items, estimated Do budget and estimated Do time (see Figure 4).

The screenshot displays the MOGWD web application interface. At the top left is the MOGWD logo, and at the top right is a 'Logout' link. The main content area features a breadcrumb trail 'Master Main Page' with a home icon. Below this is the heading 'Start New Iteration'. A prompt reads 'Please, key-in the following information'. The form contains three input fields: '# of Do Items number', 'Estimated Do Budget', and 'Estimated Do Time'. A 'Save' button is positioned below the third field. The footer contains the text 'Copyright © 2015 | Owner: Moath Husni Al-Tarawneh'.

Figure 4. Start new iteration

- **Iteration reviewing**

The master also required to enter his own data regarding to the quantitative and qualitative metrics in the iteration reviewing activity.

- **Activate the current activities**

The master also responsible for activating the current activity, for example, if the master activates the test activity, thus only the tester can enter the data at the current situation while the PO cannot enter the data of the next activity (daily reviewing) because it is not active yet, which useful to ensure the sequence of the activities (see Figure 5).

Project Settings

Please select the current active iteration:	Iteration two ▼
Please tick the active activities:	
<input checked="" type="checkbox"/> planning	
<input checked="" type="checkbox"/> design	
<input checked="" type="checkbox"/> code	
<input checked="" type="checkbox"/> test	
<input checked="" type="checkbox"/> daily reviewing	
<input checked="" type="checkbox"/> iteration reviewing	
<input checked="" type="checkbox"/> save the increment	
<input checked="" type="checkbox"/> integration	
Is the current project completed?	<input checked="" type="radio"/> Yes <input type="radio"/> No
<input type="button" value="Save"/>	

Figure 5. Activate the current activity

- **View report**

After the iteration or the project ends, the master can view the report for the iteration and the project to the management. The report includes the results of performing the quantitative and qualitative metrics during the development process.

After creating the iteration, the master activates planning activity. Consequently, the PO entered the data that he owns on the system as shown in Figure 6.

[Planning Activity](#) | [Daily Reviewing Activity](#) | [Save the Increment Activity](#)

Planning Activity

Please, tick your level of agreement with the following statements:

Activity	Metric	0	1	2	3	4
Requirement completeness	Q.M1.1.1: Customers or P.O was available on-site for face-to-face discussions during the requirement elicitation	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Q.M1.1.2: The scope of project was identified at the beginning of a project to create initial prioritized product backlog items	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Q.M1.1.3: The requirements were validated by customers in review meetings by using prototype/release	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Q.M1.1.4: Requirements were prioritized and can be reprioritized by customers throughout the development	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Q.M1.1.5: The development team was enabled to re-estimate the time and velocity of user stories	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Q.M1.1.6: The requirements were written on cards in a short statement	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Requirement consistency	Q.M2.1.1: Appropriate procedure is used to handle frequently changing requirements	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Q.M2.1.2: The requirements were documented by following a particular standard	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 6. Planning activity data collection (PO)

After finishing the plan phase, the Do phase begins.

Do phase

This phase consists of several activities such as: design, code, test, daily meeting and reviewing meeting.

- **Design**

During the design activity, the GOMM member helped the programmer to enter the design data after he logged into the MO-PT system (see Figure 7).

[Design Activity](#) | [Code Activity](#) | [Integration Activity](#)

Design Activity

Step 1: Quantitative Metrics

Please, key-in the following inputs:

Metric	Inputs
M1.2.1.3	# of internal links
	# of web pages
M4.6.1	# of page links (internal+external)
	# of total links
M4.7.1	# of dynamic pages
M4.7.3	# of direct links
M5.1.2	# of reused web pages

Step 2: Qualitative Metrics

Please, tick your level of agreement with the following statements:

Activity	Metric	0	1	2	3	4
	Q.M1.2.1:Model storming was performed (architecture, interface, data structure and algorithm)	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 7. Design activity data collection (programmer)

- **Code**

After the code activity finished, the GOMM member helped the programmer to enter the code data after he logged into the MO-PT system (see Figure 8).

[Design Activity](#) || [Code Activity](#) || [Integration Activity](#)

Code Activity

Step 1: Quantitative Metrics

Please, key-in the following inputs:

Metric	Inputs
M1.2.1.1	LOC completed to date
M1.2.1.2	# of web pages to date
M4.5.1	Effort in hour for locating each fault
M4.5.2	Effort in hour for fixing each fault
M5.1.1	# of reused LOC
	Total LOC to date
M2.1.1	# of KLOC for the programmer in the month
PM3.5.1	# of team members who made changes on the code

Step 2: Qualitative Metrics

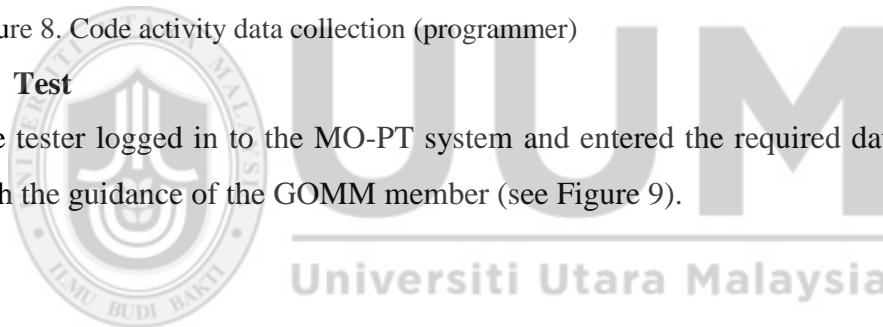
Please, tick your level of agreement with the following statements:

Activity	Metric	0	1	2	3	4
	Q.M1.3.1: Reuse of software components was encouraged	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 8. Code activity data collection (programmer)

- **Test**

The tester logged in to the MO-PT system and entered the required data for testing with the guidance of the GOMM member (see Figure 9).



MAGWD Help Logout

[Test Activity](#)

Test Activity

Step 1: Quantitative Metrics

Please, key-in the following inputs:

Metric	Inputs
M1.3.1.1	# of test completed to date
PM2.1.1	# of duplicated LOC removed
	Total LOC to date

Step 2: Qualitative Metrics

Please, tick your level of agreement with the following statements:

Activity	Metric	0	1	2	3	4
	Q.M1.4.1: Tests were automated	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Q.M1.4.2: Tests were performed continuously throughout the development	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Q.M1.4.3: Frequent integration tests were performed	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Q.M1.4.4: Unit tests were performed to ensure that all requirements were fulfilled	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 9. Test activity data collection (tester)

- **Daily reviewing**

PO entered the required data at the last daily meeting (see Figure 10).

The screenshot shows a web application interface for 'MOGWD'. At the top right, there are links for 'Help' and 'Logout'. Below the header, there are navigation links: 'Planning Activity', 'Daily Reviewing Activity', and 'Save the Increment Activity'. The main heading is 'Daily Reviewing Activity'. Below this, it says 'Please key-in the following inputs:'. A table with two columns, 'Metric' and 'Inputs', contains the following data:

Metric	Inputs
M1.1.1.1	# of product backlog items completed to date
	Total # of product backlog planned
M3.2.1	# of dollars spent for Do
M3.3.1	Total # of Dollars spent for the whole product
PM1.2.1	# of daily meetings
<input type="button" value="Save"/>	

At the bottom right of the page, there is a copyright notice: 'Copyright © 2015 | Owner: Moath Husni Al-Tarawneh'.

Figure 10. Daily reviewing activity data collection (PO)

- **Iteration reviewing meeting**

This meeting also provides a time for the GOMM team members to help the master to enter the required data (see Figure 11).

MOGWD Help Logout

[Master Main Page](#) | [Iterations Page](#)

Iteration Reviewing Activity

Step 1: Quantitative Metrics

Please, key-in the following information

Metric	Inputs
M3.1.1	Dollars spent to fix post to release problems
M4.2.1	# of Do defects LOC for the DO
M4.3.1	# of pre-release defect of the DO # of post-release defects of the DO
M4.4.2	Mean time to find defect
M4.4.3	Mean time between two defects
M4.4.4	Mean time to recover
M5.2.1	Elapsed time Estimated time
M5.2.2	Current DO time Estimated DO time
PM1.3.1	# of review meeting

Figure 11. Do reviewing activity data collection (Master)

Check phase

After completing the data collection the analysis of the data will be performed by MO-PT. The results of the analysis will be presented in the act phase.

Act phase

This phase includes the following activities:

- Save the increment

The PO entered the required data for monitoring this activity as shown in Figure 12.

MOGWD Help Logout

[Planning Activity](#) | [Daily Reviewing Activity](#) | [Save the Increment Activity](#)

Save the increment to the repository

Please key-in the following inputs:

Metric	Inputs
PM3.1.1	LOC of the first release
	LOC of the current release
	Total LOC
<input type="button" value="Save"/>	

Copyright © 2015 | Owner: Moath Husni Al-Tarawneh

Figure 12. Save the increment activity data collection (PO)

- Integrate with the system

The programmer is also required to enter the data for monitoring in this activity as shown in Figure 13.

MOGWD Help Logout

[Design Activity](#) | [Code Activity](#) | [Integration Activity](#)

Integrate with the system

Please, key-in the following inputs:

Metric	Inputs
PM3.2.1	LOC added, removed and updated
	LOC of the previous iteration
PM3.3.1	# of LOC of the current release
	Total LOC
<input type="button" value="Save"/>	

Copyright © 2015 | Owner: Moath Husni Al-Tarawneh

Figure 13. Integrate with the system activity data collection (programmer)

- **View report**

After the development activities the monitoring report will be presented. The GOMM asked the master to print the report from his page on the MO-PT by clicking on *view report* in the iteration page (see Figure 14).

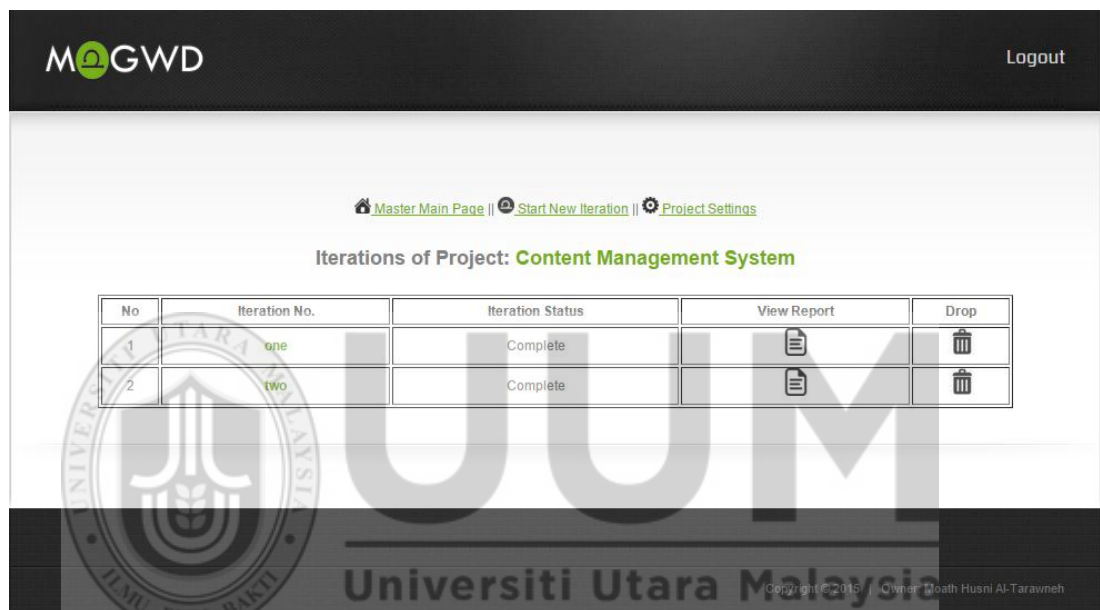


Figure 14. View report

Based on the figure, each iteration has a monitoring report. Each report consists of quantitative metrics and qualitative metrics results. The report includes the indicators and action of improvement if needed for each metric.

If any metric has the indicator “need to improve”, the MO-PT shows the action button beside the indicator. By clicking on that action button, a pop up message will be shown telling the team the action that should be taken (see Figure 15).

Code	M1.2.1.2	web pages progress	52.3 %	Perfect
	M4.5.1	Effort for locating each fault (h)	2	Acceptable
	M4.5.2	Effort for		Need to improve Action
	M5.1.1	Reused		Acceptable
	M2.1.1	Program		Need to improve Action
	PM3.5.1	Collectiv have the power to change the code)		Acceptable
Test	M1.3.1.1	Test Progress	53.8 %	Perfect
	PM2.1.1	Refactoring	10.9 %	Acceptable

Implement pair programming practice

OK

Figure 15. Action message

