

**A PROCESS BASED APPROACH SOFTWARE CERTIFICATION
MODEL FOR AGILE AND SECURE ENVIRONMENT**

SHAFINAH FARVIN PACKER MOHAMED

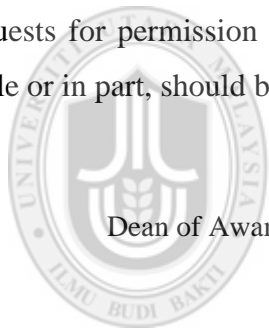


**DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2015**

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 :



Dean of Awang Had Salleh Graduate School of Arts and Sciences

UUM College of Arts and Sciences

Universiti Utara Malaysia

06010 UUM Sintok

Abstrak

Di dalam persekitaran perniagaan hari ini, proses perisian Agil dan selamat menjadi penting kerana kedua-dua proses ini dapat menghasilkan perisian yang berkualiti tinggi dan terjamin keselamatannya untuk dipasarkan dengan lebih cepat dan kos efektif. Malangnya, terdapat di kalangan para pengamal perisian yang tidak mengikuti amalan yang sesuai bagi kedua-dua proses ketika membangunkan perisian. Terdapat banyak kajian telah dijalankan untuk menilai kualiti proses perisian, walau bagaimanapun, tumpuan kajian tersebut hanya diberikan kepada proses perisian lazim. Tambahan pula, kajian yang sedia ada tidak mengambil kira nilai pemberat di dalam penilaian walaupun setiap kriteria penilaian mungkin mempunyai kepentingan yang berbeza. Oleh yang demikian, pensijilan perisian diperlukan untuk menjamin kualiti bagi proses perisian Agil dan selamat. Justeru, objektif tesis ini adalah untuk mencadangkan Model Pensijilan dan Penilaian Proses Perisian Lanjutan (ESPAC) yang memberi fokus kepada kedua-dua proses perisian ini dan mengambil kira nilai pemberat ketika menjalankan penilaian. Kajian ini telah dijalankan dalam empat fasa: 1) kajian teori untuk mengkaji faktor dan amalan yang mempengaruhi kualiti proses perisian Agil dan selamat serta teknik untuk memperuntukkan nilai pemberat, 2) kajian penerokaan yang disertai oleh 114 pengamal perisian untuk mengkaji amalan pembangunan perisian mereka, 3) pembangunan model pensijilan proses perisian lanjutan yang mengambil kira proses, manusia, teknologi, kekangan projek dan persekitaran serta menyediakan garis panduan pensijilan dan menggunakan Proses Hierarki Analitik (AHP) untuk memperuntukkan nilai pemberat dan 4) penentuan proses perisian Agil dan selamat serta AHP melalui kajian pakar, diikuti dengan pengesahsahihan terhadap tahap kepuasan dan praktikal model yang dicadangkan melalui perbincangan kumpulan berfokus. Keputusan pengesahsahihan menunjukkan bahawa Model ESPAC telah mencapai kepuasan pengamal perisian dan didapati praktikal untuk dilaksanakan di dalam persekitaran sebenar. Sumbangan kajian ini mencakupi perspektif Pensijilan dan Penilaian Proses Perisian dan Kriteria Berbilang Membuat Keputusan, serta perspektif praktikal dengan menyediakan satu mekanisme yang boleh digunakan oleh pengamal dan penilai perisian untuk menentukan tahap kualiti proses perisian dan membantu pelabur serta pelanggan dalam membuat keputusan pelaburan.

Kata kunci: Pensijilan proses perisian, Proses perisian Agil, Proses perisian selamat, Proses Hierarki Analitik, Model Pensijilan dan Penilaian Proses Perisian Lanjutan.

Abstract

In today's business environment, Agile and secure software processes are essential since they bring high quality and secured software to market faster and more cost-effectively. Unfortunately, some software practitioners are not following the proper practices of both processes when developing software. There exist various studies which assess the quality of software process; nevertheless, their focus is on the conventional software process. Furthermore, they do not consider weight values in the assessment although each evaluation criterion might have different importance. Consequently, software certification is needed to give conformance on the quality of Agile and secure software processes. Therefore, the objective of this thesis is to propose Extended Software Process Assessment and Certification Model (ESPAC) which addresses both software processes and considers the weight values during the assessment. The study is conducted in four phases: 1) theoretical study to examine the factors and practices that influence the quality of Agile and secure software processes and weight value allocation techniques, 2) an exploratory study which was participated by 114 software practitioners to investigate their current practices, 3) development of an enhanced software process certification model which considers process, people, technology, project constraint and environment, provides certification guideline and utilizes the Analytic Hierarchy Process (AHP) for weight values allocation and 4) verification of Agile and secure software processes and AHP through expert reviews followed by validation on satisfaction and practicality of the proposed model through focus group discussion. The validation result shows that ESPAC Model gained software practitioners' satisfaction and practical to be executed in the real environment. The contributions of this study straddle research perspectives of Software Process Assessment and Certification and Multiple Criteria Decision Making, and practical perspectives by providing software practitioners and assessors a mechanism to reveal the quality of software process and helps investors and customers in making investment decisions.

Keywords: Software process certification, Agile software process, Secure software processes, Analytic Hierarchy Process, Extended Software Process Assessment and Certification Model.

Acknowledgement

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

First and foremost all praise and thanks go to Allah for giving me the strength and patience to accomplish this study. Besides, completing this thesis would not have been possible without a number of people who offered their unfailing support throughout the period of the study.

I would like to express my sincerest thanks and deepest gratitude to my supervisors Assoc. Prof. Dr. Fauziah Baharom and Prof. Dato' Dr. Aziz Deraman for their excellent guidance, caring, patience, encouragement and sharing of all their research experiences throughout these challenging years.

My sincere thanks must also go to the members of viva committee: Prof. Dr. Suhaidi Hassan as the chairman, Prof. Dr. Saadiah Yahya from University Technology Mara (UiTM) and Assoc. Prof. Dr. Suhaimi Ibrahim from University Technology Malaysia (UTM) as the external examiners for sharing their constructive advices and comments to improve my thesis.

I would also like to extend my thanks to the Ministry of Higher Education Malaysia and Universiti Utara Malaysia for providing funds and opportunity to conduct this study. My high appreciation also goes to the School of Computing staff for their kind supports and comments. Also I would like to thank the knowledge and domain experts who provided their insights on this study. Their fruitful comments and suggestions are utmost important for my study.

On a more personal level, I would like to express my gratitude to my parents and my beloved family members for their continuous prayers, patience and supports throughout my four years plus of difficult endeavour. My gratitude also goes to all my colleagues in the PhD journey, especially for the discussions and suggestions on the better ways to perform my study.

Table of Contents

Permission to Use	i
Abstrak.....	ii
Abstract.....	iii
Acknowledgement	iv
Table of Contents.....	v
List of Tables	x
List of Figures.....	xii
List of Appendices	xiv
List of Abbreviations	xv
CHAPTER ONE INTRODUCTION	1
1.1 Overview.....	1
1.2 Background.....	1
1.3 Problem Statements	7
1.4 Research Questions.....	12
1.5 Objectives	13
1.6 Scope.....	13
1.7 Significance.....	14
1.8 Thesis Organization	16
CHAPTER TWO LITERATURE REVIEW	18
2.1 Introduction.....	18
2.2 Software Certification.....	18
2.2.1 Evaluation Theory: The Theory That Underpins Software Certification ..	21
2.2.2 Software Certification Process.....	22
2.2.3 Existing Process Based Software Certification Model	24
2.2.4 Current Issues in Software Process Certification	31
2.2.5 Software Process Certification for Agile Software Process	35
2.2.6 Software Process Certification for Secure Software Process	37
2.2.7 Factors that Influence the Quality of Agile and Secure Software Processes	39
2.2.8 The Agile and Secure Software Practices.....	53

2.3 Multiple Criteria Decision Making (MCDM).....	72
2.3.1 Analytic Hierarchy Process (AHP).....	77
2.3.2 Weighted Sum Method (WSM).....	80
2.4 Measurement Approach in Software Process Certification.....	82
2.5 Summary.....	85
CHAPTER THREE RESEARCH METHODOLOGY.....	86
3.1 Introduction.....	86
3.2 Research Design.....	86
3.3 Phase One: Theoretical Study.....	88
3.4 Phase Two: Exploratory Study.....	88
3.4.1 Instrument Design.....	89
3.4.2 Sampling for the Survey.....	90
3.4.3 Instrument Testing.....	91
3.4.4 Data Collection.....	91
3.4.5 Data Analysis.....	92
3.5 Phase Three: ESPAC Model Development.....	92
3.5.1 Defining the target.....	93
3.5.2 Defining the evaluation criteria.....	93
3.5.3 Building the Reference Standard.....	94
3.5.4 Determining the Data Gathering Techniques.....	94
3.5.5 Determining the Assessment Process.....	95
3.5.6 Determining the Synthesis Technique.....	95
3.5.7 Determining the Achievement Index.....	96
3.6 Phase Four: ESPAC Model Evaluation.....	96
3.6.1 Verification Stage.....	96
3.6.2 Validation Stage.....	98
3.7 Summary.....	100
CHAPTER FOUR EXPLORATORY STUDY.....	101
4.1 Introduction.....	101
4.2 Instrument Design.....	101
4.3 Sampling.....	103

4.4 Instrument Testing	104
4.5 Data Collection	105
4.6 Data Analysis	106
4.7 Findings.....	107
4.7.1 Demographic Information.....	107
4.7.1.1 Respondents’ Background.....	107
4.7.1.2 Organizational Background.....	109
4.7.2 Current Practices of Agile Software Process	109
4.7.2.1 Software Practitioners’ Familiarity of Agile	110
4.7.2.2 Level of Exposure to Agile	110
4.7.2.3 Years of Experience Implementing Agile	111
4.7.2.4 Number of Agile Team Members	111
4.7.2.5 Agile Methods	112
4.7.2.6 Benefits of Agile	112
4.7.2.7 Implementation of Agile Principles	113
4.7.3 Current Practices of Secure Software Process	114
4.7.3.1 Software Practitioners’ Familiarity of Secure Software Process	115
4.7.3.2 Common Attacks Prevention Technique.....	115
4.7.3.3 Security Trainings	116
4.7.3.4 Notations for Security Requirements	117
4.7.3.5 Security Requirement Elicitation Practice	117
4.7.3.6 Security Incidents Faced	118
4.7.4 Agile Software Practices that Influence the Quality of Software	119
4.7.5 Secure Software Practices that Influence the Quality of Software	123
4.7.6 Perceptions on The Importance of Agile and Secure Software Processes in Producing High Quality Software.	125
4.7.7 Characteristics of People Who Involve in Agile and Secure Software Processes.....	125
4.7.8 Current Practices of Software Certification.....	127
4.7.8.1 Software Practitioners’ Opinion on the Importance of Software Certification.....	127

4.7.8.2 The Implementation of Internal Assessment/Audit and the Techniques Used	128
4.7.8.3 The Use of Standards	129
4.8 Discussions	131
4.9 Summary	154
CHAPTER FIVE ESPAC MODEL DEVELOPMENT.....	155
5.1 Introduction.....	155
5.2 Overview of ESPAC Model.....	155
5.3 The Components of the ESPAC Model	161
5.3.1 Target	161
5.3.2 Evaluation Criteria	161
5.3.3 Reference Standard	165
5.3.4 Data Gathering Techniques	168
5.3.5 Assessment Process	169
5.3.6 Synthesis Technique	177
5.3.7 The Achievement Index.....	197
5.3.7.1 The Quality Levels	197
5.3.7.2 The Certification Level	197
5.4 Discussions	198
5.5 Summary	201
CHAPTER SIX ESPAC MODEL EVALUATION	202
6.1 Introduction.....	202
6.2 Verification through Expert Reviews	202
6.2.1 Experts for AHP Technique Verification	203
6.2.2 Experts for the Agile and Secure Software Processes	204
6.2.3 Results for the AHP Technique Verification.....	205
6.2.4 Results for the Factors, Sub Factors and the Agile and Secure Software Processes.....	206
6.3 Verification and Validation through Focus Group	209
6.3.1 Plan the Focus Group.....	210
6.3.1.1 Define the Objectives of the Focus Group	210

6.3.1.2	Participants Identification and Recruitment	210
6.3.1.3	Meeting Scheduling	211
6.3.1.4	Preparation of the Focus Group Interview Guide and Materials	212
6.3.1.5	Remind the Participants	212
6.3.2	Conduct the Focus Group	213
6.3.2.1	Obtain the Weight Values for Evaluation Criteria	215
6.3.2.2	The Agile and Secure Software Processes Verification	216
6.3.2.3	The ESPAC Model Validation	217
6.3.3	Data Analysis and Results Reporting	217
6.3.4	The Focus Group Discussion Findings	218
6.4	Validation Results and Discussions	221
6.4.1	Gain Satisfaction	222
6.4.2	Interface Satisfaction	224
6.4.3	Task Support Satisfaction	225
6.5	Summary	227
	CHAPTER SEVEN CONCLUSIONS	228
7.1	Introduction	228
7.2	Study Recapitulation	228
7.3	Contributions	233
7.3.1	The ESPAC Model	233
7.3.2	The Reference Standard	234
7.3.3	The Synthesis Technique	235
7.3.4	The AHP Technique Implementation through Planning Poker	236
7.3.5	Utilize the QFD for the Reference Standard	236
7.4	Limitations and Future Directions	237
7.5	Conclusions	239
	REFERENCES	241

List of Tables

Table 2.1 Achievement Grades Scale for ISO/IEC 15504 PA (Galín, 2004)	29
Table 2.2 Comparisons of Existing Software Process Certification Models and Standards....	30
Table 2.3 Lacking Issues in Existing Software Process Certification Models and Standards .	34
Table 2.4 The Factors Regarding Development Team	40
Table 2.5 The Factors Regarding Organization	41
Table 2.6 The Factors Regarding Customer	42
Table 2.7 The Factors Regarding Process.....	43
Table 2.8 The Organizational Factors.....	46
Table 2.9 The People Factors	48
Table 2.10 The Technical Factors.....	49
Table 2.11 Other Factors That Influence the Successful Implementation of Information Security	50
Table 2.12 Summary of Factors Influencing the Software Process Quality	51
Table 2.13 Factors, Sub Factors and The Evaluation Criteria	53
Table 2.14 The Agile Software Practices	55
Table 2.15 The Secure Software Practices	62
Table 2.16 The Skills Needed for Software Practitioners.....	69
Table 2.17 MCDM Techniques	73
Table 2.18 AHP Preference Scale (Saaty, 1990)	78
Table 2.19 The Existing AHP Studies on Evaluation.....	78
Table 2.20 The Existing WSM Studies on Evaluation	81
Table 3.1 Validation Criteria for ESPAC Model.....	99
Table 4.1 Overview of Respondents.....	106
Table 4.2 Respondents' Position in Company.....	108
Table 4.3 Respondents' Experience.....	108
Table 4.4 Sectors of Organization	109
Table 4.5 Interval Values.....	113
Table 4.6 Agile Principles Implementation	114
Table 4.7 Prevention Techniques from Common Attacks	115
Table 4.8 Percentages of Security Training Provided.....	116
Table 4.9 Notations for Security Requirements.....	117
Table 4.10 Eliciting Security Requirements Explicitly during Requirement Gathering.....	118

Table 4.11 Interval Values.....	119
Table 4.12 The Mean Values for Agile Software Practices.....	120
Table 4.13 The Mean Values for Secure Software Practices.....	123
Table 4.14 Team and Organizational Characteristics	126
Table 4.15 The Importance of Software Certification Based on Respondents' Familiarity ..	128
Table 4.16 Assessment Techniques	128
Table 4.17 The Use of Standards Based on Respondents' Experience.....	130
Table 5.1 The Comparisons of the SPAC and ESPAC Models.....	160
Table 5.2 The Assessed Factors.....	164
Table 5.3 The Data Gathering Techniques	168
Table 5.4 Descriptions of the Pre-Assessment Phase	171
Table 5.5 Descriptions of the Assessment Phase.....	173
Table 5.6 Descriptions of the Post-Assessment Phase.....	175
Table 5.7 The Pair Wise Matrix.....	180
Table 5.8 The Pair Wise Comparison Matrix for Level One.....	181
Table 5.9 The Pair wise Comparison Matrix for Level Two.....	181
Table 5.10 Summary of the Pair Wise Comparison Matrixes for the ESPAC Model.....	182
Table 5.11 Scale of Pair Wise Comparison (Saaty, 1990).....	182
Table 5.12 Pair Wise Comparisons for Sub Factors of Software Development.....	183
Table 5.13 Weight Values for Sub Factors of Software Development	186
Table 5.14 Random Index (Saaty, 1988 as cited in Padumadasa et al., 2009)	187
Table 5.15 The Consistency Vectors	188
Table 5.16 The Achievement Index.....	198
Table 6.1 Experts' Background	205
Table 6.2 Results for the AHP Verification.....	206
Table 6.3 Descriptions of Verification Criteria	207
Table 6.4 Summary of Experts' Comments.....	208
Table 6.5 The Reorganizing and Updating Actions.....	208
Table 6.6 Anonymized Overview of the Participants	219
Table 6.7 The Suggested Software Processes.....	221
Table 6.8 The Results of Evaluation for Gain Satisfaction Criteria	222
Table 6.9 The Results of Evaluation for Interface Satisfaction Criteria.....	224
Table 6.10 The Results of Evaluation for Task Support Satisfaction Criteria.....	225
Table L.1 The grouping of projects	340
Table L.2: Certification results	364

List of Figures

Figure 2.1. Certification scenario simplified in Fauziah (2008)	20
Figure 2.2. Software Process Assessment and Certification Model (SPAC) (Fauziah, 2008) 25	
Figure 2.3. The basic structure of HOQ (Cohen, 1995).....	83
Figure 3.1. Research framework	87
Figure 3.2. The structure of evaluation criteria.....	94
Figure 4.1. Level of exposure to Agile	110
Figure 4.2. Years of experience	111
Figure 4.3. Number of Agile team members	111
Figure 4.4. Agile methods being practiced	112
Figure 4.5. Benefits of Agile practices	112
Figure 4.6. The security incidents faced	119
Figure 4.7. The use of standards	129
Figure 5.1. The proposed ESPAC Model	158
Figure 5.2. The hierarchy tree of the evaluation criteria.....	163
Figure 5.3. The structure of the reference standard	166
Figure 5.4. The example of reference standard for requirement engineering in Agile	167
Figure 5.5. Assessment process of the ESPAC Model	170
Figure 5.6. Steps to calculate the weight values	179
Figure 5.8. The example of assessment form for Agile requirement engineering	192
Figure 5.9. Activities for performing the assessment	193
Figure 5.10. The part of hierarchy tree with score and global weight values	196
Figure 6.1. The meeting place setting	214
Figure 6.2. The selection of the pair wise comparison value by the participants	216
Figure 6.3. The process of verifying and validating the ESPAC model	217
Figure L.1. Assessment results for requirement engineering.....	342
Figure L.2. Assessment results for software design.....	343
Figure L.3. Assessment results for coding	344
Figure L.4. Assessment results for testing	345
Figure L.5. Assessment result for project management	346
Figure L.6. Assessment results for change management	347
Figure L.7. Assessment results for support process.....	348
Figure L.8. Assessment results for requirement engineering.....	350

Figure L.9. Assessment results for software design.....	351
Figure L.10. Assessment results for coding.....	352
Figure L.11. Assessment results for testing.....	353
Figure L.12. Assessment results for security management.....	354
Figure L.13. Assessment results for risk management	355
Figure L.14. Assessment results for support process.....	356
Figure L.15. Assessment results for technology	357
Figure L.16. Assessment results for software practitioners' characteristics.....	358
Figure L.17. Assessment results for organization and customers.....	360
Figure L.18. Assessment results for project constraint.....	362
Figure L.19. Assessment results for environment.....	363



List of Appendices

Appendix A The Objectives and Sources for Instrument Development.....	271
Appendix B The Instrument.....	274
Appendix C The Brochure	286
Appendix D Hierarchy Tree for Agile Software Process.....	287
Appendix E Hierarchy Tree for Secure Software Process	288
Appendix F Pair wise Comparison Form.....	289
Appendix G The AHP Verification Form.....	292
Appendix H The Verification and Assessment Form	293
Appendix I The Overall Verification Form	295
Appendix J The Validation Form	297
Appendix K The Assessment Form	299
Appendix L The Assessment and Certification Results.....	335
Appendix M The Local/Global/Ideal Weights.....	365
Appendix N The Global Weights and Scores for Agile Software Process	367
Appendix O Publications	369



UUM
Universiti Utara Malaysia

List of Abbreviations

AHP	Analytic Hierarchy Process
AM	Agile Modeling
ANC	Average of Normalized Columns
ASD	Adaptive Software Development
CI	Consistency Index
CLASP	Comprehensive, Lightweight Application Security Process
CR	Consistency Ratio
CMMI	Capability Maturity Model Integrated
DSDM	Dynamic Systems Development Method
ESPAC Model	Extended Software Process Assessment and Certification Model
FDD	Feature-Driven Development
GQM	Goal Question Metric
HOQ	House of Quality
ISO	International Organization for Standardization
LSPCM	Laquso Software Product Certification Model
MCDM	Multiple Criteria Decision Making
MS SDL	Microsoft Security Development Lifecycle
NGM	Normalization of the Geometric Mean of the Rows
NRA	Normalization of Row Average
NRC	Normalization of the Reciprocal Sum of Columns
OWASP	Open Web Application Security Project
QFD	Quality Function Deployment
SCAMPI	Standard CMMI Appraisal Method for Process Improvement
SCM _{prod}	Software Product Certification Model
SEI	Software Engineering Insitute
SPAC Model	Software Process Assessment and Certification Model
SPSS	Statistical Package for Social Science
SSE-CMM	System Security Engineering CMM
TDD	Test Driven Development
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
WSM	Weighted Sum Method
XP	Extreme Programming

CHAPTER ONE

INTRODUCTION

1.1 Overview

This chapter provides an introduction to the study which begins with the background of the study, followed by the discussion on the problem. Then, research questions are provided and used to construct the objectives. Finally, this chapter presents the scope as well as the significance of the research. This chapter is concluded with an overview of the remaining chapters of this thesis.

1.2 Background

The use for software has become indispensable in today's world since its usage has become more and more critical in every domain of our life. Surprisingly, as indicated by Jones and Bonsignour (2012), even though software is among the most widely used product in human history, its failure rate is one of the highest among any other products in human history. Consequently, customers are always concerned with the quality of the software produced for them, whether the software meets their needs and follows certain standards. On top of that, in today's business environment, the customers expect that the software can be produced in the market faster and have good security features. Nevertheless, complaints about customers' dissatisfactions on the software still exist even though the software developers claimed that the software they produced is in good quality (The Standish Group, 2013; Weber-Jahnke, 2011; Cerpa & Verner, 2009; Charette, 2005; Lindstrom & Jeffries, 2004).

Therefore, one way of getting conformance on the software quality is through certification (Heck, Klabbers & Eekelen, 2010; Aziz, Jamaiah, Fauziah, Amalina Farhi, & Abdul Razak, 2007). Referring to The International Organization for Standardization (ISO), certification is defined as *“the procedure by which a third party gives written assurance that a product, process or service conforms to a specified characteristics”* (ISO, 2015; Rae, Robert, & Hausen,1995). With certification, customers will feel more confident on the quality and dependability in selecting organizations when making investment because it involves independent assessment which will then reduce the possibility of software failure (Sun-Jen & Wen-Ming, 2006; Rae et al., 1995). This is supported by the outcome from a survey conducted by Fauziah, Aziz and Abdul Razak (2005) and the exploratory study conducted in this study. The findings from these studies provide evidence that software certification is certainly needed in order to give assurance on the quality of software in Malaysia.

Voas (1998) points out that there are three approaches in certifying the quality of software, which are: process, product and personnel. There are several studies focusing on the product approach such as Heck et al. (2010) and Jamaiah (2007). However, it is hard to determine the quality of newly developed software without implementing it for a certain period of time (Fauziah, Jamaiah, Aziz, & Abdul Razak, 2011; Heck et al., 2010; Sommerville, 2004). Thus, process approach is chosen in this study as an alternative to product approach. By applying software certification based on the process approach, customers are able to know about the quality level of software process in a particular organization. This can help them in making decision on whether to invest in a particular

organization or not as it reveals the capability of an organization in producing high quality software (Gonzalez, Rubio, Gonzalez, & Velthuis, 2010).

Software process is defined as “*set of activities undertaken to manage, develop and maintain software systems in order to produce a software system, executed by a group of people organized according to a given organizational structure and counting on the support of techno-conceptual tools*” (Acuna, Antonio, Ferre, Lopez, & Mate, 2000). The underlying idea behind this is, by certifying that the software process performed in a particular organization is well-defined, it reflects that the produced software also has good quality, as stated by Deming (1982) and Humphrey (1989) “*the quality of product is influenced by the quality of process used to develop it*”. Consequently, it is vital to conduct software certification based on process approach in order to ensure that software process has been effectively and efficiently performed.

A considerable amount of literature has been published on models and standards which assess the quality of software process. For example, the Capability Maturity Model Integrated (CMMI) (CMMI Product Team, 2010) and ISO/IEC 15504 (Mas, Fluxa, & Amengual, 2012; Van Loon, 2007; Galin, 2004; El Emam & Birk, 2000). However, as indicated by Acuna, Antonio, Ferre, Lopez and Mate (2000), the aim of these existing models and standards is more on assessing and improving the software process rather than providing mechanism for certification. Furthermore, these assessments focus more on the conventional software process (Gandomani & Hazura, 2013; Lami & Falcini, 2009; Diaz, Garbajosa, & Calvo-Manzano, 2009; Marcal et al., 2008; Salo & Abrahamsson, 2005). Sommerville (2007) defines conventional software process as

“specification-based software development, which is based on completely specifying the requirements up front then designing, building and testing the system with emphasize given on documentation rather than the software itself”.

Nonetheless, nowadays with the modern business environment which is fast-paced and ever-changing, incorporating agility during software development has become very essential as it brings high quality software to market faster and more cost-effectively (Mehta & Adlakha, 2012; Santos, Bermejo, Oliveira, & Tonelli, 2011; Lee & Xia, 2010; Pressman, 2010; Sommerville, 2007). Thus, developers are increasingly incorporating agility in their software process. By the year 2002, approximately 67% of pioneering software developers has used Extreme Programming which represents one of the Agile methods (VersionOne, 2011; Rico, Sayani, & Sone, 2009; Salo & Abrahamsson, 2008). This shows high acceptance of this approach in the industry. A similar outcome is found in the survey conducted by this study, whereby majority of the software practitioners (64%) are familiar with Agile. As defined by Boehm & Turner (2005), Agile is *“lightweight software development approach which emphasizes on iterative, incremental, self-organizing and emergent practices”*. Agile supports close collaboration between software development and business team, face-to-face communication, frequent delivery, changes acceptance and adaptive organizational capability.

Additionally, Erdogan, Meland and Mathieson (2010), Nunes, Belchior and Albuquerque (2010) and Suhazimah, Ainin and Ali Hussein (2009) indicate that the security aspect has gained high concern in today’s business environment since the current application environment has become more complex, distributed and easily exposed to malicious

attacks. Moreover, Jones and Bonsignour (2012), Kazemi, Khajouei and Nasrabadi (2012), Nunes et al. (2010) and Julia, Barnum, Ellison, McGraw and Mead (2008) highlight security aspect as the major concern among the customers since most computer applications nowadays store numerous confidential data. As a consequence of these situations, there exist numerous incidents that lead the customers to lose hundred million of dollars every year caused by frauds and computer crime activities.

Koi (2012) reveals that there are around 15,218 incidents reported through the Cyber999 Help Centre in Malaysia in year 2012, compared to only 8,095 incidents in 2010. On top of that, recently, Malaysia was attacked by a Latin American gang who hacked 14 ATM machines in Selangor, Johor and Malacca and ran away with approximately three millions ringgit in cash (Kumar, Cheng, Nadirah, & Natasya, 2014). For that reason, currently the quality of software does not only focus on maintainability, usability and functionality (Guceglioglu & Demirors, 2005), but also emphasizes security (O'Regan, 2014; Hui, Dongyan, Min, Weizhe, & Dongmin, 2014; Merkow & Raghavan, 2010; Voas, 2008; Julia et al., 2008; Offut, 2002).

Researchers believe that security activities should be considered from the very beginning of the software development lifecycle and continuous in all phases, rather than having it as a sub-segment of software development in order to produce software for secured environment (McGraw, 2011; Muniraman & Damodaran, 2007; Essafi, Labed, & Ghezala, 2006; Lipner, 2006). According to McGraw (2004), secure software process is *“about building secure software: designing software to be secure, making sure that software is secure, and educating software developers, architects, and users about how to*

build secure things”. Nevertheless, despite the importance of Agile and secure software processes in today’s business environment, so far the awareness on the software process certification related to both software processes is still limited. This is the reason why both the Agile and secure software processes are not included in the existing software process certification model.

Additionally, the existing software process certification models and standards also lack appropriate synthesis technique in the certification process. Fauziah (2008) observes that the existing software process certification models and standards do not include weight value in the certification process. On the other hand, it has conclusively been shown that when an assessment involves multiple criteria, they will have different importance. Therefore, these criteria should be weighted (Saaty, 2008; Brugha, 2004; Malczewski, 1999; Yoon & Hwang, 1995). Furthermore, Triantaphyllou and Mann (1995) and Yoon and Hwang (1995) posit that the process of assigning weight values to the evaluation criteria is significant especially when the qualitative information is needed from the decision makers. Accordingly, this study enhances the existing software process certification model to overcome the lacking issues by including the Agile and secure software processes which are essential in today’s business environment. Also, the synthesis technique is improved in order to produce more consistent and better quality of certification results.

1.3 Problem Statements

The existing software process certification models and standards have thus far focused primarily on determining the software process maturity of an organization and proposing improvement to their software process, as what has been applied by Capability Maturity Model Integrated (CMMI) (CMMI Product Team, 2010), ISO/IEC 15504 (Mas et al., 2012; Van Loon, 2007; Galin, 2004; El Emam & Birk 2000) and System Security Engineering-Capability Maturity Model (SSE-CMM) (Davis, 2013; Carnegie Mellon University, 2003) which is now accepted as ISO/IEC 21827 standard. On the other hand, even though the ISO/IEC 27001 series (ISO, 2015; Evans, Tsohou, Tryfonas, & Morgan, 2010; Humphreys, 2008) provide mechanism for certification, they only focus on the general software process. To date, only Software Process Assessment and Certification (SPAC) Model (Fauziah, 2008) provides certification by assessing and certifying software process that have been carried out effectively and efficiently by organizations. The subsequent subsections discuss several shortcomings of the SPAC Model which need further investigation.

1) The needs of software process certification model that give emphasis on the Agile and secure software processes

It is vital for the software practitioners in today's business environment to implement best practices of Agile and secure software processes towards producing high quality and secured software faster and more cost effectively. However, based on the observations made by previous studies, the software practitioners are left far behind the theoretical best practices. This was observed by Brooks back in the year 1987 (Cater-Steel, 2004;

McConnell, 2002), and still can be seen today (Fauziah et al., 2005; McConnell, 2002; Ludewig, 2001).

Numerous studies have been conducted to investigate the current practice of Agile. Among them are by VersionOne (2011) and West and Grant (2010) that investigated the status of the Agile adoption and practices in the software industry. Salo and Abrahamsson (2008) on the other hand, investigate the usefulness of XP and Scrum in the European embedded software development organizations. Furthermore, Santos et al. (2011) study the perception of software practitioners on the relationship of Agile with the quality of software. In Malaysia, Ani Liza (2012) investigates on the perception of software practitioners when adopting Agile.

Regarding the secure software process, the existing studies focus more on showing the importance of considering security measures in developing software. For instance, Whitehat Security has investigated the number of vulnerabilities in small, medium and large organizations (Whitehat Security, 2013), while National Cyber Security Alliance (National Cyber Security Alliance, 2012) has done a survey on the security trainings provided in software companies, the awareness of security initiatives and the security problems faced. On the other hand, Elahi, Yu, Tong and Lin (2011) and Wilander and Gustavsson (2005) have investigated the software practitioners' practices in security requirement engineering.

It appears from the aforementioned studies that most of the existing studies were conducted in Western countries (Sison, Jarzabek, Hock, Rivepiboon, & Hai, 2006).

However, the adoption level of best practices related to Agile and secure software processes is still scarce in Malaysia. Furthermore, instead of focusing on the software certification or software quality, these studies were more concerned on a particular Agile method such as XP or Scrum. Albeit study by Santos et al. (2011) relates Agile with software quality, it did not include the Agile Modeling (AM) method. In fact, research on the effectiveness of AM is still scarce (Erickson, Lyytinen, & Siau, 2005). Additionally, very few studies have been conducted to reveal the secure software process adopted by software practitioners (Tondel, Jensen & Rostad, 2010). Even though studies by Elahi et al. (2011) and Wilander and Gustavsson (2005) addressed this issue, they just focus on the security requirement engineering, which represents a small part of the software process. Hence, this has motivated this study to investigate the awareness on the importance of software process certification with relation to the Agile and secure software processes among software practitioners in Malaysia.

In addition, the existing software process certification models and standards are more focused on the conventional software process (Gandomani & Hazura, 2013; Lami & Falcini, 2009; Diaz et al, 2009; Marcal et al., 2008; Salo & Abrahamsson, 2005). The SPAC Model considers five factors that influence the quality of software process as the reference model, which are the process performed, quality of people involved, working environment, technology and project condition, which focuses on the conventional software process. The Capability Maturity Model Integrated (CMMI) (CMMI Product Team, 2010) on the other hand, includes project management, process management, engineering and support. The CMMI has also included guidelines and notes for software practitioners who implement the CMMI in the Agile environment (CMMI Product Team,

2010). However, those are only general guidelines and included for certain process area. Therefore, it is significant to investigate and include the Agile software process because the process and work products are different from the conventional software process (Abrahamsson, Oza, & Siponen, 2010; Pikkarainen & Mantynie, 2006). Pressman (2010) also highlights that agility should be included in the current software process to ensure the quality of software.

Similarly, even though secure software process has also become a determinant factor of high quality software, the existing security standard such as ISO/IEC 27001 tends to focus on information security management system and only focuses generally on the software development process. ISO/IEC 21827 (Davis, 2013; Carnegie Mellon University, 2003) on the other hand concentrates on security engineering, which includes the risk, engineering and assurance processes. In spite of that, the aim of this model is more towards achieving the maturity of system security management.

As discussed above, it should be noted that limited studies are available on both software processes in the existing software process certification models and standards. For that reason, this has motivated the present study. Moreover, these two software processes have been partially addressed by the two most influential software process models and standards providers; the Software Engineering Institute (SEI) and ISO. The SEI addresses agility issue through CMMI, while ISO provides standards which emphasize on security issue through ISO/IEC 27001 and ISO/IEC 21827. This shows that both software processes are important to be addressed in today's business environment.

2) Improving the synthesis technique in software certification

The synthesis technique used in the software certification of the existing software process certification models and standards need further improvement to produce better quality and consistency on the certification decision made. This is necessary because the software certification process involves multiple criteria assessment whereby each of the criteria must be weighted since they might have different importance (Saaty, 2008; Brugha, 2004; Malczewski, 1999; Yoon & Hwang, 1995). Weight can be defined as “*a value assigned to an evaluation criterion that indicates its importance relative to other criteria under consideration*” (Malczewski, 1999). Nevertheless, despite of its importance, little attention has been directed to consider weight values for the assessed criteria in the existing software process certification models and standards, including the SPAC Model (Fauziah, 2008).

Since software process assessment involves multiple criteria, the Multiple Criteria Decision Making (MCDM) technique is an appropriate technique for determining the weight for each assessed criteria. The MCDM refers to “*making preference decision over the available alternatives that are characterized by multiple, usually conflicting attributes*” (Triantaphyllo, 2000). Although the MCDM provides numerous techniques in determining the weight, the most widely used is the Analytic Hierarchy Process (AHP) (Ishizaka & Labib, 2011; Rao & Davim, 2008; Vaidya & Kumar, 2006).

The AHP technique has been successfully implemented in the evaluation domain. Zhou and Liang (2013) utilize the AHP to evaluate the network course in China, while Chen, Pham and Yuan (2013) employ the AHP to evaluate potential outsourcing partner. In

addition, the AHP has also been used in conducting evaluation on component-based software (Al-Tarawneh, 2014) and tender (Padumadasa, Colombo and Rehan, 2009). The AHP utilization increases the consistency of judgments (Liberatore & Nydick, 1997). Thus, this has motivated the study to improve the synthesis technique in the software certification process by incorporating weight values through the adaptation of the AHP technique.

Based on the initiative made by the SPAC Model, this study has overcome the above mentioned shortcomings by enhancing the existing software process certification model. With the proposed model, the software certification can be conducted in a broader aspect which suits the current business environment and needs, since it includes Agile and secure software processes; and improves the synthesis technique in software certification.

1.4 Research Questions

The research questions aimed to be answered at the end of this study are as follows:

1. What are the current practices of software process certification in relation to Agile and secure software processes?
2. How to enhance software process certification model by including the Agile and secure software processes?
3. What are the techniques that can be used to improve the synthesizing process in software certification?
4. How to evaluate the proposed software process certification model?

1.5 Objectives

With regards to the above discussed problems, the following objectives are outlined for this study:

1. To investigate the current practices of software process certification in relation to Agile and secure software processes.
2. To enhance software process certification model by including the Agile and secure software processes.
3. To improve the synthesis technique in software certification by using the Analytic Hierarchy Process (AHP).
4. To evaluate the enhanced software process certification model by using expert review and focus group.

1.6 Scope

The scope of this study includes the certification approach and the software processes. The followings are further descriptions on the certification approach and the software processes:

- Certification Approach Scope

Although there are three approaches in the software certification, this study only focuses on the software process approach. The choice is made because if the product approach were to be used, it is hard to determine the quality of newly developed software without implementing it for a certain period of time (Fauziah et al., 2011; Heck et al., 2010; Sommerville, 2004). The advantage of the process approach is that the investors are able

to know the quality of the software development process performed in a particular organization before making any decision on whether to invest in the organization or not since the organization's process is one of the fundamental assets of organization (Guceglioglu & Demirors, 2005). The process approach reveals the capability of the organization in producing high quality software, as it is believed that a software failure is a consequence from the process failure (Doernhoefer, 2006).

- Software processes scope

The proposed model focuses on the software processes which concern about the latest business environment and needs, which are Agile and secure software processes, rather than only focusing on the conventional approach. This needs to be highlighted because currently these two approaches has become the determinant factor for producing high quality software in today's business environment. The software processes included in the proposed model were identified from the theoretical and exploratory study.

1.7 Significance

By achieving the objectives of this study, a systematic approach to ensure the effectiveness and efficiency of software process is offered to the software practitioners and assessors. Besides that, it is beneficial to the investors and customers. On top of that, this study supports the body of knowledge in several fields. These are discussed further subsequently.

- Body of knowledge

The main aim of this study is to provide an assessment and certification model which focuses on the Agile and secure software processes. Consequently, by achieving this aim, the study contributes to the field of Software Engineering particularly in the Software Process Assessment and Certification area. In addition, the study also contributes to the field of MCDM. Since this study focuses on the Agile and secure software processes, thus the factors and practices that influence the quality of software process which focuses on these two software processes are revealed. Moreover, this study has incorporated the AHP technique for the synthesizing process, which is relatively new in the Software Process Assessment and Certification area.

- Software developers and assessors

For the software developers, the assessment and certification provides a mechanism to reveal the quality of software process currently being performed in their projects. Consequently, the outcome from the assessment and certification can be used to plan and improve their upcoming software processes. Additionally, as the proposed model provides a proper guideline for assessing and certifying software process, thus the software assessors can refer to the proposed model for assuring the quality of software process performed by an organization.

- Investors and customers

The proposed model can help investors and customers in making investment decisions. This is because, before making investment, investors and customers can get conformance on the quality of software process implemented in an organization, which directly

influences the quality of produced software in the organization. Thus, it reveals the capability of an organization. It will give them higher confidence level on the quality of organization or software which they will invest on.

1.8 Thesis Organization

This thesis consists of seven chapters including this chapter. The outline of the remaining chapters of the thesis is as follows:

- **Chapter Two: Literature Review**

This chapter presents a review on the existing studies in related area, which is software certification. The focus is given on factors and practices that influence the quality of software process by concentrating on the Agile and secure software processes. Additionally, focus is given on the synthesis technique. Outcome from the review is important for constructing instrument for the exploratory study and gives support for producing the proposed model.

- **Chapter Three: Research Methodology**

This chapter discusses the research methodology that was used to achieve the objectives of this study. It discusses the four phases that were conducted in order to construct the software process assessment and certification model which emphasize on the Agile and secure software processes in detail.

- **Chapter Four: Exploratory Study**

This chapter confers the outcomes obtained from the exploratory study conducted among software practitioners in Malaysia. It reveals their opinions and experiences on the software certification which relates to the Agile and secure software processes. The outcomes from this study have been used to support the construction of the proposed model.

- **Chapter Five: ESPAC Model Development**

This chapter discusses in detail about the proposed model. The discussion is organized based on the components of the proposed model, which are the target, evaluation criteria, reference standard, data gathering techniques, synthesis technique assessment process and Achievement Index.

- **Chapter Six: ESPAC Model Evaluation**

This chapter reports the evaluation of the proposed model through two stages, which are verification and validation. The verification was performed through expert review and focus group. Meanwhile, during the focus group, the proposed model was validated on its practicality and ability to meet users' satisfaction.

- **Chapter seven: Conclusions**

This chapter concludes the study by recapitulating the study. Then, the contributions of this study are highlighted. Finally the limitations of the study are addressed followed by the future directions in related field.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter describes the state-of-the-art concerning the software process certification issues. The discussion is started with overview of software certification in Section 2.2, whereby the concepts of software certification, the existing software certification models and the issues are explained. Then the factors that influence the quality of Agile and secure software processes are elaborated. In Section 2.3, the MCDM is described and continued with the measurement approach in software process certification in Section 2.4. This chapter is ended with the summary in Section 2.5.

2.2 Software Certification

Certification has become an established activity in many industries for decades, however, it is very hard to assess software as it is intangible. Nonetheless, the need for this activity has been realized caused by the dangers of software failure which are becoming more obvious. Fauziah et al. (2005) reported complaints in a study conducted at software development organizations in Malaysia. 75% of the organizations mentioned that their software products need to be improved after they were released, 55% of them faced problem with delivery time, 20% faced budgetary problems and 22.5% of them face problem with customers who are unhappy of the quality of software that they produce. More recently, study conducted by the Standish Group (Standish Group, 2013), revealed that even though the success rate of software projects is reported to be increased, from

37% in year 2010 to 39% in year 2012, there still exist projects which failed and challenged, 18% and 43% respectively.

In general, a software is considered as in good quality if it: 1) meets the expected requirements, 2) completed within budget, 3) completed on time, 4) completed in its entirety, 5) delivered together with a solid and thoroughly tested code, 6) can be used easily (Nasution & Weistroffer, 2009), with good security features (O'Regan, 2014; Jones & Bonsignour, 2012; Merkow & Raghavan, 2010; Offut, 2002; Mouratidis & Giorgini, 2007) and follows certain standards (Sommerville, 2007; Jamaiah, Fauziah, Aziz, & Abdul Razak, 2005). Investing on bad quality software will cause negative impacts to the users. Due to this, nowadays customers are more concerned with the quality of the software that they invest on (O'Regan, 2014; Jones & Bonsignour, 2012; Heck et al., 2010). Consequently, the customers need confirmation about the quality of software they are investing on. One way to have assurance on quality of software is by having certification.

Fabbrini, Fusani, and Lami (2006) have illustrated the basic scenario of certification, as simplified by Fauziah (2008), shown in Figure 2.1. Certification Body is an organism with internal rules, human resources and skills which are used to perform the certification procedures. Certification object is what is usually certified, which are process, product or people. Standard for certification process consist of well-known standards being applied worldwide. The usage depends on which certification object being certified and the rules implemented by certification body are same for all object of the same type. The standard

requirement relating to an object of interest (product, process or people) will be used by the Certification Body to assess the object.

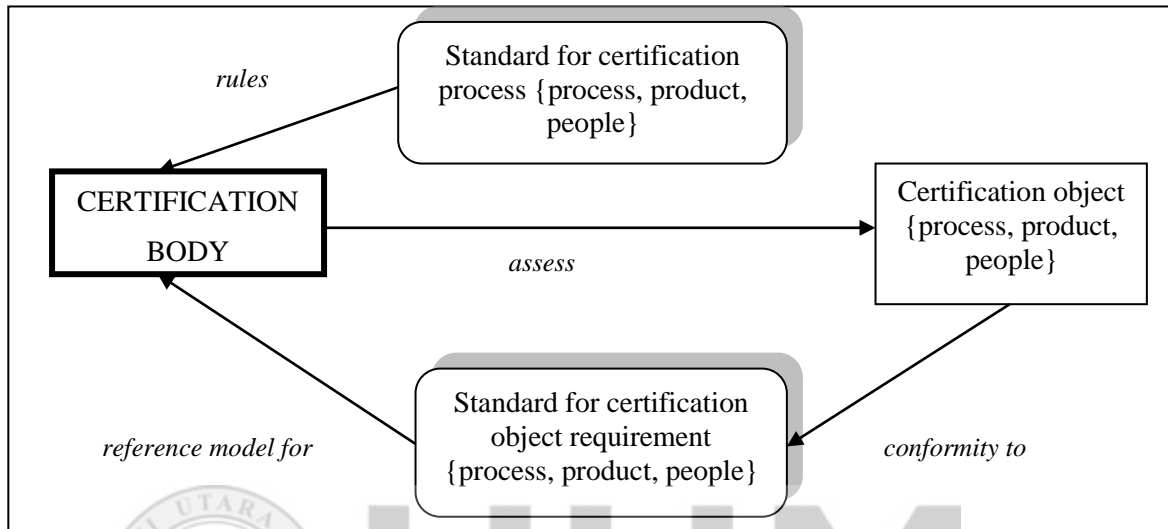


Figure 2.1. Certification scenario simplified in Fauziah (2008)

Besides providing a written statement about the level of software quality, there are many reasons for the need of software certification, to be exact it leads to a better quality and increased productivity of software product (Fabbrini et al., 2006), serves as a tool for improving development and maintenance processes (Galín, 2004), increases disciplines among development team by encouraging best practices in software development and emphasize standardization implementation (Tripp, 2002), increases the ability to compete in the market (Heck et al., 2010; Fauziah, Aziz, & Abdul Razak, 2007) and increases customers' trustworthiness towards the claim made by organizations about the quality of their software (Rathfelder, Groenda, & Reussner, 2008).

2.2.1 Evaluation Theory: The Theory That Underpins Software Certification

The key activity in software certification is assessing the certification object. Thus, the Evaluation Theory (Scriven, 1991) is very closely related. Generally, an evaluation involves: 1) identification of relevant standards of merit, worth or value, 2) investigation of the performance of targets (whatever is being assessed) on these standards and 3) integration or synthesis of the results to achieve an overall evaluation result or a set of association evaluation results. In conducting evaluation, six evaluation components need to be considered. They are common to any type of evaluation, regardless of the discipline, field or evaluand considered (Acuna et al., 2000). The six components are listed below (Al-Tarawneh, 2014; Zarour, 2009; Acuna et al., 2000; Ares, Garcia, Juristo, Lopez, & Moreno, 2000; Scriven, 1991):

1. Target: the object under evaluation.
2. Evaluation criteria: the characteristics of evaluated object.
3. Reference standard / Yardstick: the ideal target which is compared with the real target.
4. Data gathering techniques: techniques used to assess the criteria under analysis.
5. Synthesis technique: technique used to judge each criterion, and in general, to judge the target, obtaining the results of the evaluation.
6. Evaluation process: series of activities and tasks to be performed for the evaluation process.

Several researchers in software engineering field have used the Evaluation Theory as the benchmark for their studies (Al-Tarawneh, 2014; Zarour, 2009; Ares et al., 2000). Al-Tarawneh utilized the theory to develop an evaluation method for the Component off-The Shelf (COTS) software. On the other hand, Zarour (2009) develop a method to evaluate

lightweight software process assessment methods, while Ares et al. (2000) utilized the theory to construct a framework for software process assessment. Thus the Evaluation Theory was used in this study by adapting the six components to develop the proposed model.

2.2.2 Software Certification Process

Software certification process can be categorized by its methods and approaches. There are four methods in conducting software certification, which are self-certification, second-party certification, third-party certification (Voas, 1999; Vermesan, 1998) and collaborative assessment (Jamaiah, 2007; Fauziah, 2008). With self-certification, organizations declare by themselves that their product conforms to a certain standard. The advantage of this method is the certification process can be done faster, because the project is already understood by the developer (Voas, 1999). However, certification process using this method may cause biased assessment. Nevertheless, self-certification which involves self-assessment can cut the cost down since the assessment is performed within an organization (Ritchie & Dale, 2000).

For second-party certification, user will require a particular specified body to certify the product. Third-party certification on the other hand occurs when the certification process is carried out by a body which is independent of both user and organization (Vermesan, 1998). Alternatively, collaborative assessment is introduced by Fauziah (2008) and Jamaiah (2007) in their studies. This method is implemented by the users, developers and independent assessor collaboratively. They pointed out that there are advantages in the collaborative approach compared to other methods, which are: 1) it eliminates bias

assessment and evaluation of the product by including independent assessor in the team, 2) it removes unfairness evaluation by including the owner or users of the product to participate in the assessment process and 3) it accelerates the process because the team is familiar with the product and its' environment, and 4) it protects data confidentiality and privacy by only permitting direct users to have access to the software (Jamaiah, Aziz, & Abdul Razak, 2006). In this study, the collaborative self-assessment method is proposed by combining the self-certification/ self-assessment with the collaborative assessment, considering the advantages of both methods.

Voas (1998) pointed out that there are three approaches in certifying software, which are: process, product and personnel. Process approach ensures that the software was developed by following the development processes properly, while product approach assesses the software itself and personnel approach ensures that developers has specific skill sets. These three approaches are known as 'The Software Quality Certification Triangle' (Voas, 1998).

Although software certification can be conducted in three approaches, however based on the literature, most of the existing software certification models focus on the product approach. Among them are the Software Product Certification (SCM_*prod*) (Jamaiah, 2007) and Laquso Software Product Certification Model (LSPCM) (Heck et al., 2010). Furthermore, Alvaro, Almeida and Meira (2007) have produced Software Component Maturity Model, a certification model which is aimed for certifying software components. However, these models cover only the product quality, without taking into account management and development factors.

The only certification model found in the literature for personnel approach thus far is the People CMM (PCMM) (Curtis, Hefley, & Miller, 2009). Besides that, there are various certifications which certify the knowledge and competence of software engineers, for instance The Certified Tester and Certified Profesional for Requirements Engineering (Rathfelder et al., 2008).

On the other hand, there are a number of studies which assesses the quality of process (CMMI Product Team, 2010; Galin, 2004; Davis, 2013; ISO, 2015). They are discussed further in the next sub section since this study focuses on the software certification based on the quality of software process. This approach is chosen since the quality assessment for product based approach is hard to be practiced without implementing the software for a certain period of time. Thus, considering Deming's (1982) and Humphrey's (1989) premise; *'the quality of product is influenced by the quality of process used to develop it'*, this study focuses on the process approach.

2.2.3 Existing Process Based Software Certification Model

Although there are various models and standards which assess the quality of software process, their focus is on improving the software process (Acuna et al., 2000), except SPAC Model which emphasis on assessing and certifying the software process (Fauziah, 2008). Nevertheless, this model need further improvement since it does not include the software processes which are essential for producing high quality software in today's business environment: the Agile and secure software processes (Fauziah et al., 2011). Thus, this study adapted the SPAC Model as the basis model and utilized other software

process models and standards which give emphasis on these two software processes. Consequently, these models and standards are elaborated further subsequently.

i. Software Process Assessment and Certification Model (SPAC Model)

Fauziah (2008) has developed software certification model based on software development process approach, Software Process Assessment and Certification Model (SPAC), as depicted in Figure 2.2. The ultimate goal of this model is to assure that the proper software development processes have been carried out effectively and efficiently in order to meet the expected quality criteria, delivered on time and within budget. It is formulated based on existing models, which are Capability Maturity Model (CMM), ISO 9000, ISO/IEC 15504 and Bootstrap.

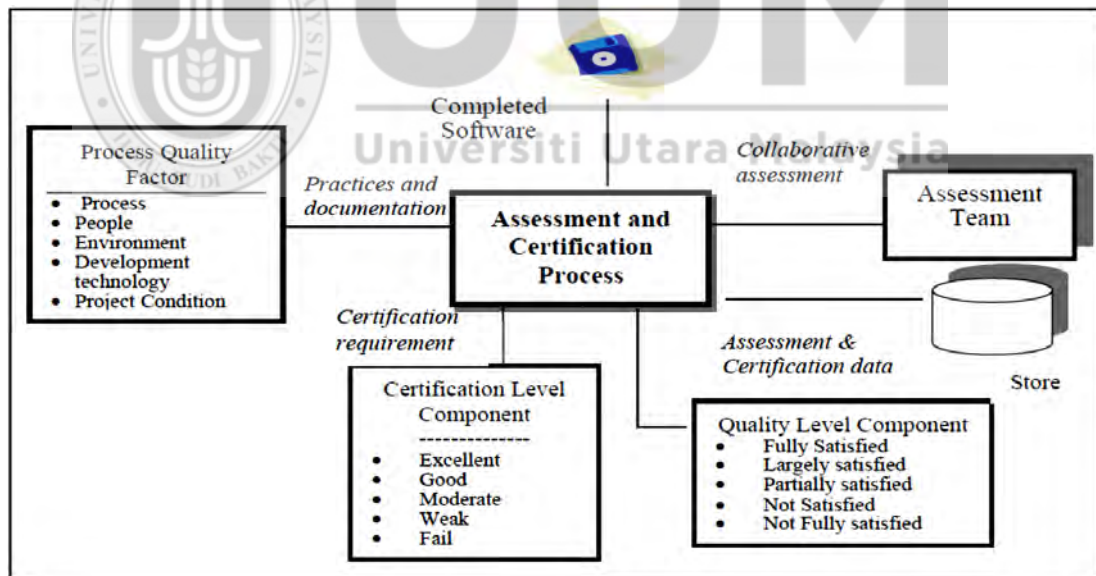


Figure 2.2. Software Process Assessment and Certification Model (SPAC) (Fauziah, 2008)

SPAC Model consists of seven (7) components: 1) completed software, 2) process quality factor, 3) certification level component, 4) quality level component, 5) assessment and

certification process, 6) assessment team and 7) repository. This model focuses in assessing five main factors that influence software quality: 1) the process performed, 2) the quality of people involved, 3) the working environment, 4) the use of development technology and 5) project condition. Each of these factors is decomposed into sub factors and measures. Nevertheless, this model only focuses on the conventional software process. Thus this study enhances the assessment by including the Agile and secure software processes.

Furthermore, two assessment methods are constructed in this model, which are quality assessment and certification determination. At the end, the certification level is determined by referring to the quality assessment. This model adopts the CGPA calculation method for its synthesis technique. However, the weights for evaluated factors are set to equal value. Thus, this study overcomes this shortcoming by addressing the weight values for the synthesis technique.

ii. Capability Maturity Model Integrated (CMMI)

CMMI (Rout, 2011; CMMI Product Team, 2010; Hoggerl & Sehorz, 2006) is the successor of the Capability Maturity Model (CMM) or Software CMM. In 2002, CMMI Version 1.1 was released, version 1.2 followed in August 2006, and version 1.3 was recently released in October 2010. CMMI is a process improvement approach by Software Engineering Institute at Carnegie Mellon University that provides organizations with the essential elements of effective processes to improve their performance.

Unlike the previous versions, CMMI version 1.3 considers its implementation for Agile practitioners (Philips & Shrum, 2010; CMMI Product Team, 2010). In this version, CMMI provides interpretation guideline on how to interpret Agile practices and includes notes to the relevant process areas. The process areas which include the guideline and notes are: configuration management, product integration, project monitoring and control, project planning, process and product quality assurance, requirement development, requirement management, risk management, technical solution and verification. Accordingly, the relevant Agile software practices are adapted from CMMI Version 1.3 to be used in this study.

iii. ISO/IEC 27001

ISO/IEC 27001 (ISO, 2015; Evans et al., 2010; Humphreys, 2008) originated from ISO/IEC 17799 standard, which is a comprehensive set of controls comprising best practices in information security management. It is intended for providing certification for organizations and aims to ensure information's confidentiality, integrity and availability.

The implementation of security management is based on the common management system model, the Plan-Do-Check-Act Model. It is ended with a third-party evaluation and certification awards. However, the ISO/IEC 27001 only certifies software development process in general and focuses more on the management of information security management. The relevant practices of this standard are adapted in the study. This standard consists of fourteen (14) sections, which are: information security policies, organization of information security, human resource security, asset management, access

control, cryptography, physical and environmental security, operations security, communications security, systems acquisition, development and maintenance, supplier relationships, information security incident management, information security aspects of business continuity management and compliance (ISO, 2015; ISECT, 2015).

iv. ISO/IEC 21827

ISO/IEC 21827 is an International Standard based on System Security Engineering CMM (SSE-CMM) (Davis, 2013; Davis, 2005; Carnegie Mellon University, 2003). It is a process reference model by Software Engineering Institute which describes the maturity level of an organization's security engineering. This standard provides security engineering activities that span the entire trusted product or secure system life cycle, including concept definition, requirements analysis, design, development, integration, installation, operations, maintenance and decommissioning. However, this standard is aimed for organizations to improve their security engineering. Nunes et al. (2010) pointed out that SSE-CMM does not restrict organizations to a particular process, whereas, organizations should use the model in its existing process. Thus, security engineering should be integrated with other engineering disciplines, such as software engineering. The relevant security practices, in particular the security management are adapted in this study.

v. ISO/IEC 15504

The ISO/IEC 15504 (Mas et al., 2012; Van Loon, 2007; Galin, 2004; El Emam & Birk 2000) is a software process assessment International standard which emerged due to the multiple models for software process assessment and improvement, which are the ISO

9000, Bootstrap, Trilium and CMM (O'Regan, 2014). It was formerly known as ISO SPICE (Software Process Improvement and Capability Determination). This standard is essential for the organizations to know its performance against the competitors in its market by assessing its process. Consequently improvements can be made based on the outcomes of the assessment.

The achievement levels required for each relevant attributes are determined based on four levels of rating scale, as described in Table 2.1. This Achievement Index is adapted in this study to determine the quality and certification levels. The assessment in ISO/IEC 15504 constitutes the engineering, customer-supplier, management, support and organization process. However, this standard focuses on software process improvement and disregards the human aspects in its assessment (Fauziah, 2008).

Table 2.1

Achievement Grades Scale for ISO/IEC 15504 PA (Galini, 2004)

Grade	Rating (%)	Achievements
F (Fully achieved)	86-100	Systematic and complete or almost complete performance of process attributes
L (Largely achieved)	51-85	Significant achievement and systematic approach are evident. Some areas of low performance exists
P (Partially achieved)	16-50	Some achievements and partial adoption of systematic approach are evident. Other aspects of process attributes are uncontrolled
N (Not achieved)	0-15	Little or no achievement of the process attributes

The existing models are compared in Table 2.2 based on their purposes, achievement levels, measurement aspects, data gathering techniques and the general descriptions.

Table 2.2

Comparisons of Existing Software Process Certification Models and Standards

Criteria	SPAC Model	CMMI	ISO/IEC 27001	ISO/IEC 21827	ISO/IEC 15504
Purpose	Certify the software process quality	Software process improvement	Certify the information security process	Security process improvement	Software process improvement
Achievement Levels	5 certification levels and 5 quality levels	5 maturity levels 4 capability levels	2 levels	5 levels	6 levels
Measurement aspects	<ul style="list-style-type: none"> • Process • People • Technology • Working Environment • Project Constraint 	<ul style="list-style-type: none"> • Project Management • Process Management • Engineering • Support 	14 sections of information security management	<ul style="list-style-type: none"> • Risk Process • Engineering Process • Assurance Process 	<ul style="list-style-type: none"> • Engineering • Customer-supplier • Management • Support and organization process
Data gathering techniques	<ul style="list-style-type: none"> • Document review • Interview • Observation 	<ul style="list-style-type: none"> • Document review • Interview 	<ul style="list-style-type: none"> • Document review • Interview • Observation 	<ul style="list-style-type: none"> • Document review • Interview 	<ul style="list-style-type: none"> • Document review • Interview • Observation
General descriptions	<ul style="list-style-type: none"> • Focuses on software lifecycle processes as well as people, environment and project condition as assessment factors • Uses the conventional software development 	<ul style="list-style-type: none"> • Focused more on the project management aspect • The assessment process starts from the beginning of development process 	<ul style="list-style-type: none"> • Focused on the security information management in organization • Does not include software development processes in detail 	<ul style="list-style-type: none"> • Aims at security process maturity and improvement • Focused more on system security management 	<ul style="list-style-type: none"> • Focused more on process maturity • Does not consider human aspects in the assessment

From the comparison made in Table 2.2, majority of the existing models focus on the software process improvement, rather than software process certification. Furthermore, they focus on the management aspect of assessment and only assess the software process in general. Though SPAC Model assesses the software process and other factors that influence the quality of software process, it only considers the conventional software process. On the other hand, albeit CMMI addresses the Agile software process, it is only a general guideline and intended for a certain process area. Moreover, The ISO/IEC 27001 standard focuses more on information security management system and only focus generally in software process. Additionally, the ISO/IEC 21827 is aimed for security maturity, while ISO/IEC 15504 focuses on process improvement and do not consider human aspects in its assessment. Based on the comparison made, the current issues in the software process certification are highlighted. Next section discusses the issues in detail.

2.2.4 Current Issues in Software Process Certification

Based on the comparison made in previous section, there exist two (2) issues which need to be addressed in the existing software process certification and standards in order to ensure that the software process certification is aligned with the current business needs. First issue is regarding the reference standard. As mentioned earlier, most of the existing software process certification models focus more on the conventional software development process in their assessment (Gandomani & Hazura, 2013; Lami & Falcini, 2009; Diaz, Garbajosa, & Calvo-Manzano, 2009; Marcal et al., 2008; Salo & Abrahamsson, 2005).

However, currently the software industry is facing challenges to bring software products to market as quickly as possible, with shorter development life cycles, lower cost and able to accept rapid requirement changes during development (Santos et al., 2011; Pressman, 2010; Pikkarainen, 2009; Dyba & Dingsoyr, 2008; Lan & Ramesh, 2007; Sommerville, 2004). Thus, the software practitioners are incorporating Agile software process during software development towards producing high quality software. This is because being Agile could bring high quality software to market faster and most cost-effective manner (Abbas, Gravell, & Wills, 2010; Pressman, 2010; Rico et al., 2009; Livermore, 2007). Several studies support that agility brings higher quality software, for instance Rico et al. (2009) and Charrate (2001).

Moreover, the software nowadays is exposed to malicious attack due to the application environment which is more complex and distributed (Erdogan et al., 2010), especially the Web enabled application. On top of that, most computer applications nowadays keep confidential data which is exposed to malicious attack (Jones & Bonsignour, 2012). There are many problems faced in Web based application, such as Website crashes and security breaches. These circumstances have influenced the level of system performance, quality and integrity of a system. Thus, in recent years, there are many serious computer crimes have been reported.

For instance, in 2009, Albert Gonzalez and two other coconspirators were charged to steal more than 130 million credit and debit cards numbers as well as account information from corporate organizations such as 7-Eleven and Hannaford Brothers using an 'SQL injection' attack (Marra, 2009). Similarly, the computer crime activities are increasing

highly in Malaysia. 6167 cases were reported in 2010 which caused RM 63 million loss. Among the frequent crime reported are in Internet banking, VoIP (Voice over Internet Protocol) and e-Commerce. Mostly it involves identity theft and fraud (Lee, 2011). On top of that, Malaysia has been listed in the Sophos Security Threat Report 2013 as the sixth most vulnerable country in the world for cyber crime, which involves the malware attacks in computer or smart phone (Bernama, 2013).

As a result, nowadays customers are concerned about the security level of the software produced to them, not only concerning on the usability, maintainability and functionality (O'Regan, 2014; Hui et al., 2014; Merkow & Raghavan, 2010; Voas, 2008; Julia et al., 2008; Offut, 2002), besides expecting the software could be delivered fast to them with the ability to accept change and move quickly (Isawi, 2011; Pikkarainen, 2009; Dyba & Dingsoyr, 2008; Sliger & Broderick, 2008; Boehm, 2008). Although these two software processes are very important in determining the quality of software, they are not considered in the reference standard of the existing software process certification model. As a consequent, there is a need to enhance the reference standard by considering these new approaches of developing software, as they give impact on delivering high quality software in today's business environment.

The second issue in the software process assessment and certification models is regarding the synthesis technique used during the software certification. The software certification involves with multiple criteria, as mentioned by Kroeger (2011) and Guceglioglu and Demirors (2005). Typically when the assessment involves multiple criteria, the factors will have different importance, thus it should be weighted (Saaty, 2008; Brugha, 2004;

Malczewski, 1999; Yoon & Hwang, 1995). In addition, determining the weight becomes vital particularly when it involves eliciting qualitative information from the decision makers (Triantaphyllou & Mann, 1995; Yoon & Hwang, 1995). Nevertheless, the synthesis technique used in the existing software process assessment and certification do not include weight values. Thus, the synthesis technique needs to be improved by incorporating weights to the assessed criteria in the proposed model. Table 2.3 compares the issues that are lacking in the existing software process certification models and standards.

Table 2.3

Lacking Issues in Existing Software Process Certification Models and Standards

Model	Reference standard		Synthesis Technique
	Agile Software Process	Secure Software Process	
SPAC Model	Not Addressed	Not Addressed	Does not consider weight value
CMMI 1.3	Partially Addressed	Not Addressed	Does not consider weight value
ISO/IEC 27001	Not Addressed	Partially Addressed	Does not consider weight value
ISO/IEC 21827	Not Addressed	Partially Addressed	Does not consider weight value
ISO/IEC 15504	Not Addressed	Not Addressed	Does not consider weight value

It follows from the table that the Agile software process has not been addressed by majority of the existing software process certification models and standards, although it is such an important aspect which determines the quality of developed software in today's business environment. However, CMMI version 1.3 addresses this aspect partially, by including notes to Agile practitioners who implement CMMI. Nevertheless, as

documented in CMMI Product Team (2010), it does not cover the whole process areas; rather it only covers several process areas such as requirement management and risk management. Moreover, the secure software process has not been included in existing software process certification model (Fauziah et al., 2011). Although security issue has been addressed in ISO/IEC 27001, they are only in general and do not include the software process in particular. On the other hand, ISO/IEC 21827 focuses more on system security engineering and intended for security process maturity.

Additionally, all of the software process certification models compared do not consider weight that should be allocated to the assessed criteria. This issue is important to be addressed as the weight value will influence the quality and consistency of the certification result.

Since the Agile and secure software processes are seen as essential for producing high quality software in today's business environment, they are considered as the reference standard for the proposed model. In addition, this study adapted the MCDM technique in order to obtain the weight values for the synthesis technique. They are discussed further in the subsequent sub sections.

2.2.5 Software Process Certification for Agile Software Process

The needs of incorporating the Agile software process in software process certification are revealed in the previous discussion. Agile software process is introduced recently as a consequence from the problems faced in conventional software process which is not flexible in accepting unstable and volatile requirements (Lohan, Conboy, & Lang, 2010;

Rico et al., 2009). Agile software process follows the twelve (12) principles and its values, as defined in the Agile Manifesto (2001). Additionally, it comprises of several methods.

Among the Agile methods are Adaptive Software Development (ASD), The Crystal Methodologies, Dynamic Systems Development Method (DSDM), Extreme Programming (XP), Feature-Driven Development (FDD), Agile Modeling (AM), Lean Software Development and Scrum (Abrahamsson et al., 2010). These methods focus on different phases in software development lifecycle. Some focus more on software development practices, such as XP and AM, some focus more on management of software development practices, such as Scrum. On the other hand, there are methods which fully support the software development lifecycle, for instance DSDM, while FDD is more suitable for requirement specification phase (Tarhan & Yilmaz, 2013; Abrahamsson et al., 2010). However, regardless of their focus, they have similarities whereby all of the methodologies are iterative, incremental, self-organizing and emergent (Stamelos & Sfetsos, 2007; Boehm & Turner, 2005; Lindvall et al., 2002).

In this study, the common Agile principles and values were investigated, as well as the XP (Wells, 2013; Beck 1999) and Scrum (Abrahamsson et al., 2010) practices, as they are the most popular and mostly being adopted (Fernandes & Almeida, 2010; Abrahamsson et al., 2010). They are often used together during software development, as Scrum focuses on project management, while XP focuses on software development (Maurer & Martel, 2002). Furthermore, they complement each other, as Fitzgerald, Hartnett, and Conboy (2006) found out in their studies. In addition, the Agile Modeling

also referred, since it is claimed to provide a methodology for effective Agile modeling and documentation (Ambler, 2014). By taking these methods into consideration, the practices are covered from a wider perspective instead of only focusing on the software development practices.

2.2.6 Software Process Certification for Secure Software Process

Secure software process has become an important software process in today's business environment, nevertheless it is not addressed in the existing software process assessment and certification model. It has become essential since the traditional perimeter defenses like firewalls, intrusion detection and anti-virus systems are unable to protect software since hackers are concerning on the software layer (Shafiq, Erwin, & Dunne, 2011; Muniraman & Damodaran, 2007). It is estimated that 80% of all breaches are application-related (Colley, 2009). Therefore, building, deploying, operating and using software which does not consider security during its development can be risky (Julia et al., 2008).

The researchers emphasis on building secure software, rather than securing software. Both of them differs in each other, whereby building secured software is meant for designing and implementing secured software. On the other hand, securing software means building software and then securing it after the software is completed (Goertze, 2009). Thus, security activities should be considered from the starting of software development and continuous in all phases, rather than having it as a sub-segment of software development. By incorporating security in later stages of software development as an afterthought, the risks of introducing security vulnerabilities into software will be higher (Shafiq et al., 2011; McGraw, 2006).

As consequence, many researchers have come out with software lifecycle models which support security activities throughout the lifecycle (Microsoft, 2012; Davis, 2013; OWASP, 2006; Fitcher & Von Solms, 2007; McGraw, 2006). Among the most prominent and used software lifecycle models which emphasize on security activities in industry to date are Comprehensive, Lightweight Application Security Project (CLASP) by Open Web Application Security Project (OWASP) (Merkow & Ragavhan, 2010; OWASP, 2006), Microsoft's Security Development Lifecycle (MS SDL) (Microsoft, 2012; Merkow & Ragavhan, 2010) and Cigital's Touchpoints (McGraw, 2011; Julia, 2008; McGraw, 2006), as cited by McGraw (2011) and De Win, Scandariato, Buyens, Gregoire and Joosen (2009).

DeWin et al. (2009) highlighted that these models provide an extensive set of activities which emphasize on secure software processes during the entire software development lifecycle. On top of that, they have gone through extensive validation. MS SDL has been used during the development of Vista project, CLASP was reviewed by several leading companies of OWASP consortium, while Touchpoints has been validated over time as it uses the experience from several industrial projects. There are observable differences among these three models, however they agree on three points: 1) supports security education throughout organization, 2) the most important activity in developing secured software is security management and 3) it is vital to implement best practices in order to succeed (Simpson, 2008; Howard & Lipner, 2006). Consequently, the security practices from these models are considered in this study.

2.2.7 Factors that Influence the Quality of Agile and Secure Software Processes

In order to perform assessment on the Agile and secure processes, the factors that influence their quality are identified, as discussed further subsequently.

i. Agile software process

Focusing on the Agile software process, the factors are concerned more on the people and process factors.

- People factor

Evidently, the people factor is the most imperative factor towards successful Agile implementation, since the nature of Agile emphasizes on individuals and interactions, customer collaboration and responding to changes. Commonly the issues regarding the people factor concentrates on the development team, organization and customers. They are discussed further:

- The development team

The attitude of the development team is very crucial, since they are the main people who perform the Agile practices. Consequently, it is essential that they are able to adapt with working practices (Lindvall et al., 2002), willing to learn continuously (Misra, Kumar, & Kumar, 2009), motivated (Rumpe & Schroder, 2002) and have societal culture such as being honest, collaborative and responsible (Misra et al., 2009). However, study by Misra et al. (2009) concluded that staff competent is not essential for the success of Agile. On the contrary, staff competent has been found as significant for Agile success (Franca et al., 2010; Tsun & Dac-Buu, 2008; Rumpe & Schroder, 2002; Lindvall et al., 2002). Furthermore, the development team must be motivated in order to perform Agile

successfully (Rumpe & Schroder, 2002). Pertaining to the Agile principle which is self-organization, it also has been found as the success factor (Franca et al., 2010; Tsun & Dac-Buu, 2008). The time taken to make decisions is essential (Misra et al., 2009), since everything needs to be performed fast. Thus, having good communication with the customers is also influential (Rumpe & Schroder, 2002), since they also involve in decision making process. Table 2.4 provides the summary.

Table 2.4

The Factors Regarding Development Team

Factors	Authors
Competent team members	Franca et al. (2010); Tsun & Dac-Buu (2008); Rumpe & Schroder (2002); Lindvall et al. (2002)
Competent team members is not influential	Misra et al. (2009)
Team's ability to adapt with working practices	Lindvall et al. (2002)
Team members that willing to learn continuously through informal traininigs	Misra et al. (2009)
Societal culture in team: honest, collaborative and responsible	Misra et al. (2009)
Motivated team	Rumpe & Schroder (2002)
Coherent, self organizing team	Franca et al. (2010); Tsun & Dac-Buu (2008)
Team is able to make fast decisions	Misra et al. (2009)
Team that have good communication with customers	Rumpe & Schroder (2002)

– The organization

Eventhough organization's support plays a significant role in order to ensure the success of Agile implementation, suprisingly study by Tsun and Dac-Buu (2008) found that it is not a significant factor for the success of Agile. However, studies of Misra et al. (2009) and Lindvall et al. (2002) concluded contradictly. For that reason, the organization must

support cooperative culture of management instead of hierarchical culture. Besides that, the organization must trust the decisions made by the development team without second-guessing their decision (Lindvall et al., 2002). On top of that, in order to encourage rapid communication, the organization must provide facilities that is appropriate for the Agile environment such as meeting rooms, co-located workstations and meeting room facilities (Lindvall et al., 2002). These factors are summarized in Table 2.5.

Table 2.5

The Factors Regarding Organization

Factors	Authors
Organization that supports cooperative culture of management	Misra et al. (2009); Lindvall et al. (2002)
Organization's support is not influential	Tsun & Dac-Buu (2008)
Organization that trusts the teams' decisions	Lindvall et al. (2002)
Organization that provides Agile environment facilities	Lindvall et al. (2002)

– The customer

Referring to the Agile Manifesto, customer collaboration is among the most influential requirement for the success of Agile. Thus, their satisfaction, collaboration (Misra et al., 2009), as well as commitment to the project is noteworthy (Misra et al., 2009; Lindvall et al., 2002). Therefore, good relationship with the customer must exist (Franca et al., 2010). However, the attitude of the customer is also influential to the success of Agile, whereby they must be able to give constant feedback (Lindvall et al., 2002). The factors related to customer are summed up in Table 2.6.

Table 2.6

The Factors Regarding Customer

Factors	Authors
Customers' satisfaction	Misra et al. (2009)
Customers' collaboration	Misra et al. (2009)
Customers' commitment	Misra et al. (2009); Tsun & Dac-Buu (2008)
Good customer relationship	Franca et al. (2010)
Customer able to give fast feedback	Lindvall et al. (2002)

- Process factor

Besides the people factor, the Agile process gives high influence on the success of Agile implementation. Study by Rumpe and Schroder (2002) concluded that testing, pair programming, tasks prioritization and achieving the goals as the success factor. Furthermore, proper Agile software engineering and Agile project management are also essential (Tsun & Dac-Buu, 2008). Among the management needed for the Agile environment are the Agile-oriented requirement management process and configuration management process (Franca et al., 2010). Also, correct integration testing (Franca et al., 2010), delivering the most important features first, as well as regular delivery has been found as vital (Franca et al., 2010; Tsun & Dac-Buu, 2008). Besides, the technical trainings are also important to be provided to the team members. Table 2.7 shows the summary of process factor.

Table 2.7

The Factors Regarding Process

Factors	Authors
Testing	Rumpe & Schroder (2002)
Pair programming	Rumpe & Schroder (2002)
Task prioritization	Rumpe & Schroder (2002)
Achieving the goals	Rumpe & Schroder (2002)
Proper Agile software engineering	Tsun & Dac-Buu (2008)
Follow good Agile project management	Tsun & Dac-Buu (2008)
Follow Agile-oriented requirement management process	Franca et al. (2010)
Follow Agile-oriented configuration management process	Franca et al. (2010)
Agile style delivery of software (regular and delivering most important features first)	Franca et al. (2010); Tsun & Dac-Buu (2008)
Correct integration testing	Franca et al. (2010)
Appropriate technical trainings provided	Tsun & Dac-Buu (2008)

Besides the process and people factors, study by Tsun and Dac-Buu (2008) also insists on the well-defined coding standards up-front. Furthermore, a detailed and realistic project schedule is also essential (Doherty, 2012).

ii. Secure software process

When concentrating on factors that influence the quality of security issue, most of the previous studies concentrated on the information security (Ai, Md Mahbubur and Leon, 2007), and there is a lack of empirical research in the security risk management area (Kotulic & Clark, 2004). Therefore, the factors that affecting the success of information security has been consulted in order to obtain the factors that influence the quality of secure software process. Previous literature (Von Solms & Von Solms, 2004; Dutta &

McRohan, 2002) concluded that information security is not mainly a technical problem, instead, it is a management or business issue.

As a consequence, the success factors of information security has been studied from the perspectives of organizational, people and technological aspects. Organizational aspects are the issues related to the managerial decisions. People aspects on the other hand, are issues related to cognition at the individual level, as well as culture and interaction with other people, while technological aspects involve technical solutions such as applications and protocols (Werlinger, Hawkey, & Besnosov, 2009). In addition, Bulgurcu, Cavusoglu and Benbasat (2010) stated that the success of information security can be accomplished by expending in both technical and socio-organizational resources. They are discussed further subsequently.

- Organizational factor

The organizational factor has been studied on the management support, policies, and organizational capabilities. The often cited success factor that influence the success of information security is the management support (Kazemi et al., 2012; Pierce, 2012; Kraemer, Carayon, & Clem, 2009; Ai et al., 2007; Torres et al., 2006; Knapp, Marshall, Rainer, & Ford, 2006; Kankanhalli et al., 2003; Fulford & Doherty, 2003). The respondent in the study of Knapp et al. (2006) stated that without management support and involvement, the creation, training and enforcement of the security policy will not be successful, as the employees will not take it seriously. Furthermore, Kankanhalli (2003) stated that with management support, the financial and technical resources are likely to be made available for security initiatives.

Furthermore, the implementation of security policy has been highlighted as a significant factor for the success of information security (Kazemi et al., 2012; Kraemer et al., 2009; Kraemer & Carayon, 2007; Ai et al., 2007; Fulford & Doherty, 2003). Therefore, studies have been conducted to investigate the employees' behaviour towards security compliance (Waly, Tassabehji, & Kamala, 2012; Bulgurcu et al., 2010; Pahnla, Siponen, & Mahmood, 2007). Bulgurcu et al. (2010) and Pahnla et al. (2007) concluded that employees' attitude, normative beliefs and habits have significant effect on their intention to comply with the security policy. Moreover, Pahnla et al. (2007) found that sanctions and rewards do not influence the intention to comply with the security policy. On contrary, Waly et al. (2012) found that communication, motivation, sanctions and reward are the most effective factors contributing to the application of information security policy in the business sector.

Besides, Hall, Sarkani, & Mazzuchi (2011) examined the impacts of organizational capabilities on the effective implementation of information security strategy. They concluded that the ability to develop awareness of the current and future threat environment, the ability to possess appropriate budget, and the ability to coordinate the budget to respond to information security threats, are positively associated with the effective implementation of information security strategy. Table 2.8 summarizes the factors related to organizational factor for the successful information security implementation.

Table 2.8

The Organizational Factors

Factors	Authors
Management support	Kazemi, Khajouei, & Nasrabadi (2012); Pierce (2012); Kraemer et al. (2009); Ai et al. (2007); Torres et al. (2006); Knapp (2006); Kankanhalli et al. (2003); Fulford & Doherty (2003)
Implementation of security policy	Kazemi et al. (2012); Kraemer et al. (2009); Kraemer & Carayon (2007); Ai et al. (2007); Fulford & Doherty (2003)
Intention to comply with security policy among staffs: staffs' attitude, normative beliefs and habits have significant effect on their intention to comply with the security policy	Bulgurcu et al. (2010); Pahnla et al. (2007)
Intention to comply with security policy among staffs: sanctions and rewards do not influence the intention to comply with the security policy	Pahnla et al. (2007)
Intention to comply with security policy among staffs: communication, motivation, sanctions and reward influence the intention to comply with the security policy	Waly et al. (2012)
Organizational capabilities: ability to develop awareness of the current and future threat environment, the ability to possess appropriate budget, and the ability to coordinate the budget to respond to information security threats, are positively associated with the effective implementation of information security strategy	Hall, Sarkani, & Mazzuchi (2011)

- People factor

From the view of people factor, previous studies investigated the awareness among the staffs, trainings provided to them, their competence and communication. The awareness on the security issue among the staffs plays a major role for the successful implementation of information security (Kazemi et al. 2012; Pierce, 2012; Siponen, Pahnla, & Mahmood, 2010; Ai et al., 2007; Lane, 2007; Torres et al., 2006). Bulgurcu et al. (2010) found that awareness among the staffs highly influences their willingness to

comply with the security initiatives. In addition, the training provided to them is vital (Kazemi et al., 2012; Pierce, 2012; Kraemer & Carayon, 2005). With the trainings provided to them, the staffs will be more responsible with their job and motivated on the work they are performing, which directly will influence the quality of their job (Kazemi et al., 2012). On top of that, the staffs must be competent so that all security risks are identified and mitigated (Ai et al., 2007; Torres et al., 2006)

Furthermore, communicating on the security risks is vital in order to create common perception and understanding on the security requirements. Koskosas and Paul (2004) found that communicating risks among the organization members plays significant role in the success of information security. Tsohou, Karyda, Kokolakis and Kiountouzis (2006) added that the security risks should be communicated among the stakeholders by using different strategies in order to reach common perception among the stakeholders. As evidence, Kraemer and Crayon (2007) concluded in the study that the communication breakdown can cause the failure in information security implementation. Moreover, study by Fulford and Doherty (2003) concluded that good understanding of security risks and security requirements plays a significant role in the successful implementation of information security. Therefore, the staffs need to get involved during the communication and get involved with the decision making process within the organization (Kraemer & Carayon, 2005). Table 2.9 recapitulates the discussed factors.

Table 2.9

The People Factors

Factors	Authors
Awareness on the security issue among the staffs	Kazemi, Khajouei, & Nasrabadi (2012); (Pierce, 2012); Ai et al. (2007); Lane (2007); Torres et al. (2006); Siponen et al. (2010)
Trainings provided to the staffs	Kazemi et al., (2012); Pierce (2012); Kraemer & Carayon (2005)
Staff competence	Ai et al. (2007); Torres et al. (2006)
Communicating on the security risks	Kraemer & Crayon (2007); Tsohou (2006) ; Koskosas & Paul (2004)
Good understanding of security risks and security requirements	Kraemer & Carayon (2005); Fulford & Doherty (2003)

- Technological factor

Technological aspect has been studied widely in the area of information security (Dunkerley & Tejay, 2011). Whitman and Mattord (2012) emphasized that access control is very important in information security since it restricts the access only to authorized personnel. Furthermore, access control has been highlighted as an important element in information security (La Reau, 2006). In the same manner, Torres et al. (2006) concluded that business connections, which are the external and internal connection to the organization's intranet, must be protected from unauthorized users. In addition, the use of software tools such as vulnerability assessment tools and intrusion detection tools can increase the effectiveness of information security implementation (Ai et al., 2007). Table 2.10 summarizes the factors.

Table 2.10

The Technical Factors

Factors	Authors
Access control	Torres et al. (2006); La Reau (2006); Whitman and Mattord (2012)
Use of software tools	Ai et al. (2007)

- Other factors

Besides the abovementioned factors, the previous studies also highlighted other factors that influence the success of information security which involves the process itself. The study of Ai et al. (2007) concluded that effective planning was found to have a paramount influence on the success of information security. Likewise, Torres et al. (2006) concluded that defining information security strategy is vital. By doing so, the plan of security goals are clearly defined. Furthermore, risk management has been found as a significant activity which must be performed in order to carry out a successful information security (Tohidi, 2011; Torres, 2006). La Reau (2006) convinced that information security is actually the process of risk management, whereby it is the ongoing process in identifying the risks and implementing mitigation plans to address them. Therefore, it has been emphasized as crucial to be considered from the beginning of software development lifecycle and continuous in all phases by numerous existing studies (Muniraman & Damodaran, 2007; Essafi et al., 2006; Lipner, 2006; McGraw, 2006).

In the study of Torres et al. (2006), the identified factors are: information system security architecture, information security integration, which aligns the information security activities with the business objectives, project accomplishment and law enforcement and compliance. In addition, another important success factor concluded in Fulford &

Doherty (2003) is the distribution of guidance on IT security policy. Also, the relationship between budget and information security performance has also been emphasized in previous studies (Ai et al., 2007; Torres et al., 2006). Table 2.11 summarizes the discussed factors.

Table 2.11

Other Factors That Influence the Successful Implementation of Information Security

Factors	Authors
Effective planning	Ai et al. (2007); Torres et al. (2006)
Risk management	Tohidi (2011); Torres (2006); La Reau (2006)
Information system security architecture, information security integration, project accomplishment and law enforcement and compliance	Torres et al. (2006)
Distribution of guidance on IT security policy	Fulford & Doherty (2003)
Security budget	Ai et al. (2007); Torres et al. (2006)

Based on the factors that influence the quality of the two software processes, they can be classified into five main factors, which are process, people, technology, project constraint and environment. This classification is similar to the software process quality factors proposed by Fauziah (2008). However, they are considered from the perspectives of Agile and secure software processes. The factors are summarized in Table 2.12.

Table 2.12

Summary of Factors Influencing the Software Process Quality

Factors	Agile Software Process Factors	Secure Software Process Factors
Process	<ul style="list-style-type: none"> - Pair programming - Task prioritization - Achieving the goals - Proper Agile software engineering - Follow good Agile project management - Follow Agile-oriented requirement management process - Follow Agile-oriented configuration management process - Agile style delivery of software (regular and delivering most important features first) - Correct integration testing - Appropriate technical trainings 	<ul style="list-style-type: none"> - Effective planning - Risk management - Information system security architecture - Information security integration - Project accomplishment - Law enforcement and compliance - Trainings provided to the staffs
People	<p>Development team:</p> <ul style="list-style-type: none"> - Competent team members - Team's ability to adapt with working practices - Team members that willing to learn continuously through informal traininigs - Societal culture in team: honest, collaborative and responsible - Motivated team - Coherent, self organizing team - Team is able to make fast decisions - Team that have good communication with customers <p>Organization:</p> <ul style="list-style-type: none"> - Organization that supports cooperative culture of management - Organization that trusts the teams' decisions - Organization that provides Agile environment facilities 	<p>Staffs:</p> <ul style="list-style-type: none"> - Awareness on the security issue among the staffs - Staff competence - Communicating on the security risks - Good understanding of security risks and security requirements - Staffs' attitude, normative beliefs and habits have significant effect on their intention to comply with the security policy <p>Organization:</p> <ul style="list-style-type: none"> - Management support - Implementation of security policy - Intention to comply with security policy among staffs: staffs' attitude, normative beliefs, habits, communication, motivation, santions and reward have significant effect on their intention to comply with the security policy - Organizational capabilities: ability to develop awareness of the

	<p>Customer:</p> <ul style="list-style-type: none"> - Customers' satisfaction - Customers' collaboration - Customers' commitment - Good customer relationship - Customer able to give fast feedback 	<p>current and future threat environment, the ability to possess appropriate budget, and the ability to coordinate the budget to respond to information security threats, are positively associated with the effective implementation of information security strategy.</p>
Technology	<ul style="list-style-type: none"> - Well-defined coding standards 	<ul style="list-style-type: none"> - Use of software tools - Access control
Constraint	<ul style="list-style-type: none"> - A detailed and realistic project schedule 	<ul style="list-style-type: none"> - Security budget
Environment	<ul style="list-style-type: none"> - Agile environment facilities 	<ul style="list-style-type: none"> - Authorized access

Referring to this study, the characteristics of evaluated object is the effectiveness and efficiency of software process. These characteristics are important in order to ensure the quality of software (Fauziah, 2008). The effectiveness of the software process is measured in terms of the level of completeness, consistency and accuracy of the process of developing products that fulfill users' expectation through the involvement of good quality people, use of appropriate technology and stability of working environment. On the other hand, the efficiency is measured in terms of the capability of the process to produce products according to estimated budget and time (Fauziah et al., 2011). Based on the five factors that influence the Agile and secure software processes as discussed in the previous section, the sub factors are defined by adapting from Fauziah (2008). The summary of the factors, sub factors, as well as the evaluation criteria is depicted in Table 2.13.

Table 2.13

Factors, Sub Factors and the Evaluation Criteria

Factors	Sub factors	Evaluation criteria
Software development process	Requirement engineering	Completeness Consistency Accuracy
	Design	
	Coding	
	Testing	
Management process	Project management	
	Change management	
	Security management	
	Risk management	
Support process	Staff initiative	
	Documentation	
	Resource management	
	Training	
People	Developer	
	Customer	Involvement
	Organization	
Technology	Tools and techniques	Completeness
	Standard and procedure	
Project constraint	Budget	Accuracy
	Schedule	
Environment	Working environment	Safety Comfort

2.2.8 The Agile and Secure Software Practices

Referring to the defined factors in previous section, this study has identified the Agile and secure software practices that need to be performed in order to produce high quality software. They are used as the reference standard in this study.

- The process factor for Agile software proces:

The Agile practices are gathered by referring to the Agile Principles and values and Agile methods which are XP, Scrum and Agile Modeling. Additionally, the relevant practices are adopted from CMMI. Furthermore, the findings from existing empirical studies which gather the opinions of software practitioners are also included. Among the studies which investigate the current practice of Agile are VersionOne (2011) and West and Grant (2010) which investigated the status of the Agile adoption and practices in the software industry. Furthermore, Salo and Abrahamsson (2008) investigate the usefulness of XP and Scrum in the European embedded software development organizations. In addition, Santos et al. (2011) study the perception of software practitioners on the relationship of Agile with the quality of software. In Malaysia, Ani Liza, Gravell and Wills (2012a) investigate on the perception of software practitioners when adopting Agile.

The Agile software practices which are considered as having high importance in the empirical studies are labeled as (+), while the ones with less importance are labeled as (-). Table 2.14 summarizes the Agile software practices and the resources. The main resources for Agile principles and values is Agile Manifesto (2001) while for XP and Scrum are Wells (2013), Abrahamsson et al. (2010), Dyba & Dingsoyr (2012) Cockburn and Highsmith (2001) and Beck (2001). Additionally, Ambler (2014) and Ambler (2006) are used mainly for the Agile Modeling. The symbol (√) is used to indicate that the practices are obtained from the main references, while additional references are listed in the 'Other References' column.

Table 2.14

The Agile Software Practices

Phases	Practices	References						Other References
		Principles	Values	XP	Scrum	AM	CMMI	
Requirement Engineering	Gathering requirements iteratively and incrementally	√	√					Williams, Rubin, & Cohn, (2010); Liu, Wang, & Gao (2010); Ramesh, Lan, & Baskerville (2010); Lan & Ramesh (2008) (+)
	Emphasizing on face-to-face communication	√	√					Williams et al. (2010); Liu et al. (2010); Ramesh et al. (2010); Lan & Ramesh (2008) (+)
	Identifying the scope at the beginning of the project to create initial prioritized stack of requirements	√			√			Ambler (2014); O'Sheedy & Sankaran (2013) (+)
	Producing product backlog and iteration backlog for ensuring the consistency and traceability of requirements				√			Salo & Abrahamsson (2008) (+)
	Using releases (working software) for validating requirements at the end of each iterations				√			Liu et al. (2010); Ramesh et al. (2010); Lan & Ramesh (2008) (+)
	The requirements are gathered with little detail in the beginning and detailed up during iterations	√	√					Williams et al. (2010) (+)
	Enabling customers to prioritize and reprioritize requirements throughout the development	√	√					Liu et al. (2010); Ramesh et al. (2010); Lan & Ramesh (2008) (+)

Phases	Practices	References						Other References
		Principles	Values	XP	Scrum	AM	CMMI	
Software Design	Implementing model storming					√		Ambler (2014) (+)
	Start designing with simple initial design and integrating it continuously	√	√	√				Rumpe and Schroder (2002); Tsun & Dac-Buu (2008); Sison & Yang (2007); Tessem (2003) (+)
	Designing with multiple models					√		(From main reference)
	Refactoring (reorganize) the design	√		√				Tsun & Dac-Buu (2008); Moser et al. (2008); Fowler (1999) (+)
	Using metaphor as architecture of the system			√				Begel & Nagappan (2007) (+) Rumpe & Schroder (2002); West & Grant (2010) (-)
	Creating an initial model at the beginning of iteration	√	√			√		Ambler (2014) (+)
	Producing just barely good enough artifacts (for situation at hand only)		√			√		Ambler (2014) (+)
Coding	Following coding/database/interface standards			√				VersionOne (2011); Williams et al. (2010); Salo and Abrahamsson (2008); Begel and Nagappan (2007); Tsun & Dac-Buu (2008); Sison & Yang (2007); Rumpe Schroder (2002) (+)
	Delivering the software frequently with increments of features	√	√	√				Franca et al. (2010); Sison & Yang (2007); Tsun & Dac-Buu (2007); Rumpe & Schroder (2002) (+)
	Having customers on-site to get continuous and immediate feedback from customer for clarification	√	√	√		√		Ani Liza, Gravell, & Wills (2012a); Tsun & Dac-Buu (2007); Tessem (2003) (+) Rumpe & Schroder (2002) (-)

Phases	Practices	References						Other References
		Principles	Values	XP	Scrum	AM	CMMI	
Coding	Deploying the software gradually in real environment	√	√					Williams & Erdogmus (2002) (+) VersionOne (2011) (-)
	Giving authority to team members to make changes at any part of the code			√				Williams et al. (2010); Salo & Abrahamsson (2008); Rumpe & Schröder (2002) (+)
	Implementing pair programming (two programmers working together)		√	√				Williams et al. (2010); Schindler (2008); Begel & Nagappan (2007); Tessem (2003); Rumpe & Schroder (2002) (+) VersionOne (2011); Salo & Abrahamsson (2008) (-)
	Implementing test driven development (TDD): write tests first, then write the code to pass the tests		√	√		√		Williams et al. (2010); Sanchez, Williams, & Maximilien (2007); Ambler (2006); Nagappan, Maximilien, Bhat, & Williams (2008); George & Williams (2004) (+) West & Grant (2010); Salo & Abrahamsson (2008); Begel & Nagappan (2007) (-)
	Integrating the newly produced code to system baseline frequently			√				VersionOne (2011); West & Grant (2010); Williams et al. (2010); Salo & Abrahamsson (2008); Begel & Nagappan (2007); Rumpe & Schroder (2002) (+)
	Determining code integration strategy and revising it						√	(From main reference)
	Producing deliverable documentation late		√			√		Ambler (2014) (+)
	Refactoring the code and database			√				Moser et al. (2008); Ambler (2006) (+) Alshayeb (2009) (-)

Phases	Practices	References						Other References
		Principles	Values	XP	Scrum	AM	CMMI	
Testing	Using acceptance tests to validate and verify user's requirements							Lan & Ramesh (2008) (+)
	Implementing user interface testing							Liu et al. (2010); VersionOne (2011) (+)
	Implementing database regression testing							Ambler (2006) (+)
	Producing executable specification		√			√		Ambler (2014) (+)
	Implementing automated tests							VersionOne (2011); Williams et al. (2010); Liu et al. (2010) (+)
	Implementing tests continuously throughout the development							VersionOne (2011); Liu et al. (2010) (+)
	Implementing frequent integration testing		√					Franca et al. (2010); Tsun & Dac-Buu (2008); Rumpe & Schroder (2002) (+)
	Acceptance tests are written or at least modeled by customers	√	√					Lan & Ramesh (2008); Paetsch et al. (2003) (+)
Project Management	Performing project planning jointly and continuously with team members	√	√	√	√			Liu et al. (2010); Lan & Ramesh (2008) (+) Salo & Abrahamsson (2008) (+)
	Conducting continuous review meetings at end of each iteration to demonstrate the latest version of software		√		√			Lan & Ramesh (2008) (+)
	Carrying out release meeting at the beginning of project to plan releases		√	√	√			Sison & Yang (2007) (+)
	Carrying out iteration meeting at the beginning of each iteration to plan iterations		√	√	√			Sison & Yang (2007) (+)

Phases	Practices	References						Other References
		Principles	Values	XP	Scrum	AM	CMMI	
Project Management	Carrying out daily stand-up meetings for daily plan		√	√	√			Li, Moe, & Dyba (2010); West & Grant (2010); Salo & Abrahamsson (2008); Sison & Yang (2007) (+)
	Planning and estimating (cost and schedule) are based on features/functions/stories of system		√	√	√			Wells (2013) (+)
	Enabling development team to re-estimate the time and velocity (speed of accomplishing tasks) of user stories		√					Liu et al. (2010); Ramesh et al. (2010); Lan & Ramesh (2008) (+)
	Conducting retrospective (postmortem) at end of each iteration to look back what worked well and what need to be improved	√	√		√			Abbas et al. (2010); Sison & Yang (2007) (+)
	Monitoring customer involvement and end-user in project activity		√				√	(From main references)
	Delivering the features with high priority first	√	√			√		Franca et al. (2010) (+)
	Revealing the current progress of iteration/sprint to everyone on sprint burn down chart		√		√			Williams et al. (2010), VersionOne (2011); West & Grant (2010) (+)
	Change Management	Controlling changes using product backlog (prioritized user stories)		√		√		
Not allowing changes once an iteration has begin, until the iteration ends			√		√			Blankenship et al. (2011); Schuh (2005) (+)
Assigning the individual who will be responsible for ensuring Change Management activities are implemented correctly							√	(From main references)

Phases	Practices	References						Other References
		Principles	Values	XP	Scrum	AM	CMMI	
	Automating the Change Management activities (e.g. building scripts)						√	(From main references)
Resource Management	Sufficient resources are allocated for the needed periods in order to accelerate the software development process				√			
Training	Technical and management trainings must be provided for the staffs				√			
Documentation	The amount of documents is minimized, by replacing them with informal documents, face to face communication and onsite customer		√			√		
	Emphasizing on single source information		√			√	Ambler (2014) (+)	
Staff Initiative	Ensuring that working hours do not exceed 40 hours per week	√		√			Salo & Abrahamsson (2008); Sison & Yang (2007); Rumpe & Schroder (2002) Sliger & Broderick (2008) (+)	

Indicators:

(√) : Practices obtained from the main references

Other references: Practices obtained from the additional references

- The process factor for secure software process

Even though there are several studies which focuses on secure software process, their focus is more on showing the criticality of considering security measures in developing software, for instance Whitehat website security investigated the number of

vulnerabilities in small, medium and large organizations (Tudor, 2013), while National Cyber Security Alliance (National Cyber Security Alliance, 2010) surveyed the security trainings provided in software companies, the awareness of security initiatives and the security problems they are facing. Nevertheless, these studies do not reveal the methods and processes adopted by software practitioners in eliciting, documenting and analyzing security requirements in the real environment (Elahi et al., 2011; Tondel et al., 2010).

Therefore, Elahi et al. (2011) and Wilander and Gustavsson (2005) investigated the software practitioners' practices in requirement engineering which focus on security. Also, Wilander and Gustavsson (2005) analyzed the requirement documents of 11 Swedish software projects. However, both studies only focus on the security requirement engineering, which is only a small phase of software process. Therefore, considering this limitation, the secure software development practices were gathered from the three most prominent secure software development lifecycle: the Touchpoints, MS SDL and CLASP, besides the security standards: the ISO/IEC 27001 and ISO/IEC 21827.

The main references for the Touchpoints are McGraw (2011), Julia (2008) and McGraw (2006), while for MS SDL the references are Microsoft (2012) and Merkow and Ragavhan (2010). Additionally, for CLASP, OWASP (2006) was referred. On the other hand for the ISO/IEC 27001, ISO (2015), Evans, Tsohou, Tryfonas and Morgan (2010) and Humphreys (2008) were referred, while the main references for the ISO/IEC 21827 are Davis (2013) and Carnegie Mellon University (2003). Besides, the available empirical studies are also taken into consideration. Table 2.15 summarizes the sources for the secure software process. The symbol (√) is used to indicate that the practices are

obtained from the main references, while additional references are listed in the ‘Other References’ column.

Table 2.15

The Secure Software Practices

Phases	Practices	References					Other references
		MS SDL	CLASP	Touchpoints	ISO/IEC 21827	ISO/IEC 27001	
Requirement Engineering	Updating security requirements iteratively, taking place as changes occur	√	√	√	√		Mead (2010)
	Documenting and maintaining a set of well-defined security requirements to prevent from introducing common attacks that occurred previously	√		√			Christian (2010); McGraw (2006)
	Obtaining security requirements explicitly	√	√	√		√	Wilander & Gustavsson (2005); Karpati, Sindre, & Opdahl (2011)
	Considering attackers’ perspective while eliciting security requirements	√	√	√	√		Mead (2010)
	The available guidelines, internal or external guidelines/standards/ policies, or established compliance requirements were referred while gathering security requirements		√				Elahi et al. (2011); OWASP (2006)
	The identified security requirements are validated with stakeholders	√	√	√			Christian (2010)
Software Design	Referring the latest lists of common attack patterns, vulnerabilities and threats in order to keep up-to-date with current trends		√	√			Julia et al. (2008)

Phases	Practices	References					Other references
		MS SDL	CLASP	Touchpoints	ISO/IEC 21827	ISO/IEC 27001	
Software Design	Defining the attack surfaces	√					(From main references)
	Identifying, classifying and rating the possible impacts, vulnerables and threats	√	√	√	√		
	The countermeasures are designed and documented	√	√	√			
	Modeling the possible threats	√	√	√			
	Performing an external (by someone outside the design team)	√	√				
	Referring to the secure coding guidelines	√	√	√			
Coding	Coding countermeasures for the identified threats	√	√	√		√	Evans et al., (2010); Ashbaugh (2009)
	Implementing pair programming to reduce vulnerability (with continuous review)	√					Ashbaugh (2009)
	Comparing outcome from automated and manual code review			√			Merkow & Raghavan (2010)
	The security features provided by programming language used are identified			√			(From main references)
	Creating test cases by focusing on the identified threats and vulnerabilities	√	√	√			

Phases	Practices	References					Other references
		MS SDL	CLASP	Touchpoints	ISO/IEC 21827	ISO/IEC 27001	
	Performing risk analysis again at the end of the phase to ensure all risks are mitigated and to consider remaining risks			√			(From main references)
	Performing penetration test (simulate attack from malicious outsiders)	√		√			
	Performing fuzz testing (use random data as input for tests)	√		√			
	Performing integration tests by focusing on the threats and vulnerabilities			√			Julia et al. (2008)
Security Management	Producing and revising security policy regularly		√		√	√	Ai et al. (2007); Tondel et al. (2008); Syed Irfan et al. (2010)
	Producing security plan	√					Ai et al. (2007)
	The security goals are established	√	√				(From main references)
	Defining the security roles and responsibilities up-front		√				
	The valuable assets that must be protected are identified					√	
Risk Management	Performing risk analysis iteratively throughout the software development to identify the possible threats, vulnerabilities and impacts of the application		√	√	√	√	Evans et al. (2010); Davis (2013)
	Planning mitigation strategy to countermeasure the identified threats, vulnerabilities and impacts	√	√	√	√	√	Evans et al. (2010); Davis (2013)

Phases	Practices	References					Other references
		MS SDL	CLASP	Touchpoints	ISO/IEC 21827	ISO/IEC 27001	
Risk Management	Ensuring the newly identified risks are reported and mitigated as soon as possible	√	√	√		√	Evans et al. (2010)
	Monitoring the identified threats, vulnerabilities and impacts throughout the development			√	√	√	Evans et al. (2010); Davis (2013)
Staff Initiatives	Provide rewards for successful security implementation			√			Waly et al. (2012)
Documentation	Preparing documentation for installing and operating the application securely	√	√				(From main references)
	Document and maintain a set of well-defined security requirements to prevent from introducing common attacks that occurred previously			√			Elahi et al. (2011)
	Share the security artifacts among the team members		√				Evans et al. (2010); Davis (2013); OWASP (2006)

Indicators:

(√) : Practices obtained from the main references

Other references: Practices obtained from the additional references

- The people factor

The people factors comprises of the software practitioners, organization and customer.

The related studies are presented subsequently.

- The software practitioners

Current software industry does not only acquire technical skills for the software practitioners (Abdul Rahman, Yusri, Mohd Adam, & Husnayati, 2010; Jiang & Klein, 1995), whereas it urges the software practitioners to have other various skill sets (Gallagher et al., 2011; Gallivan, Truex, & Kvasny, 2004). Existing studies commonly discusses the skills from the perspective of technical and non-technical skills. Khan and Kukalis (1990) concluded that technical and non-technical skills are important; nevertheless the technical skills are considered as less important compared to the non-technical skills. This is supported by Bolton (1986) in Ahmed, Capretz, Bouktif, & Campbell (2012) who reported that 80% of the workers who failed at work are caused by their inability to relate well with others, rather than having lack of technical skills.

The software practitioners are expected to have good non-technical skills such as interpersonal skills (Bassellier & Benbasat, 2003; Rodina & Zaitun, 2000; Mohd. Noah, Md. Mahbubur, Afzaal, & Awg Yussof, 1999), management skills, skills in business processes, leadership skills, global awareness (Gallagher et al., 2011; Abdul Rahman et al., 2010; Benamati & Mahaney, 2007; Rodina & Zaitun, 2000), teamwork (Azrina, Safura Zuriati, & Nafisah, 2012; Abdul Rahman et al., 2010), writing (Abdul Rahman et al., 2010), problem solving, listening, time management skills, ability to apply knowledge

(Azrina et al., 2012), negotiation and communication skills (Gallagher et al., 2011), as well as being creative (Sterling, 2003).

Management skill is defined as the ability to manage project and time management (Sanders & Curran, 1994), while interpersonal skill is a person's ability to convey information, thoughts, feelings and attitude (Mohd Noah et al., 1999). Additionally, the software practitioners are expected to manage the risks of the project (Gallagher et al., 2011). On the other hand, communication skill is described as the ability to write, talk, read, listen and make presentation (Leitheiser, 1992). Mohd Noah et al. (1999) clearly provide the skills needed to measure the interpersonal skill, which are the ability to work effectively as a member of a team, ability to listen to others, ability to manage own roles and responsibilities, ability to work alone to accomplish goals, ability to communicate in writing, ability to train others and ability to give oral presentations.

On the other hand, the technical skill is also valuable. It includes the use of hardware, software, telecommunication, database, advanced software development (Jiang & Klein, 1995). Among the technical skills cited in previous studies are the ability to design user friendly applications (Azrina et al., 2012; Rodina & Zaitun, 2000) and ability to do programming (Litecky et al., 2012; Azrina et al., 2012; Abdul Rahman et al., 2010; Benamati & Mahaney, 2007; Rodina & Zaitun, 2000).

Additionally, specifically for the Agile environment, the developers are supposed to have attitudes such as able to work in changing situations, willing to learn continuously (Schuh, 2005) and able to handle and respond to changes quickly (Agile Manifesto,

2001). Furthermore, the team should emphasize on face-to-face communication (Rao, Naidu, & Chakka, 2011; Liu et al., 2010; Lan & Ramesh, 2008; Coram & Bohner, 2005), cross functional team (Cohn & Ford, 2003), co-located (Tsun & Dac-Buu, 2008; Parsons, Ryu, & Lal, 2007; Lindval et al., 2002) and self-organized (Franca et al., 2010; Tsun & Dac-Buu, 2008; Moe, Dingsoyr, & Dyba, 2008). Besides, the team size should be small (O'Sheedy & Sankaran, 2013). Furthermore, the developers should be engaged with daily activities, only concerned with the progress of the entire iteration, responsible to the overall project's progress and ensure that news is spread between customers and team (Schuh, 2005), knowledgeable in Agile process, implement adaptive management style (Tsun & Dac-Buu, 2008), responsible to maintain relationship with customers (Tsun & Dac-Buu, 2008, Schuh, 2005), acts like a facilitator (Sliger, 2006; Schuh, 2005) and responsible to build team cohesion (Sliger, 2006).

Meanwhile, for the secure software process, the team is expected to adopt the security activities and become familiar with the security requirement of the system (Davis, 2013; Microsoft, 2012). Furthermore, common understanding about the security needs must be reached among all applicable parties, including customer (Microsoft, 2012; McGraw, 2006). Table 2.16 recapitulates the skills needed for software practitioners.

Table 2.16

The Skills Needed for Software Practitioners

Type	Skills	References
Non-technical skills	Interpersonal skills	Bassellier & Benbasat (2003); Rodina & Zaitun (2000); Noah et al. (1999)
	Management skills, communication skills, business skills, leadership skills, global awareness	Gallagher et al. (2011); Abdul Rahman et al. (2010); Benamati & Mahaney (2007); Rodina & Zaitun (2000)
	Teamwork	Azrina et al. (2012); Abdul Rahman et al. (2010)
	Writing skill	Abdul Rahman et al. (2010)
	Problem solving skills, listening skills, time management skills and ability to apply knowledge	Azrina et al. (2012)
Technical Skills	Use of hardware, software, telecommunication, database, advanced software development	Jiang & Klein (1995)
	Ability to design user friendly applications	Azrina et al. (2012); Rodina & Zaitun (2000)
	Ability to do programming	Litecky et al. (2012); Azrina et al. (2012); Abdul Rahman et al., (2010); Benamati & Mahaney (2007); Rodina & Zaitun (2000)
Agile software process	Able to work in changing situations	Schuh (2005)
	Emphasize on face-to face communication	Rao et al. (2011); Liu et al. (2010); Lan and Ramesh (2008); Coram & Bohner (2005)
	Cross functional team	Cohn & Ford (2003)
	Co-located	Tsun & Dac-Buu (2008); Parsons et al. (2007); Lindval et al. (2002)
	Self-organized	Franca et al. (2010); Tsun & Dac-Buu (2008); Moe et al. (2008)
	Small sized team	O'Sheedy & Sankaran (2013)
	Engaged with daily activities	Schuh (2005)
	Only concerned with the progress of the entire iteration	
	Responsible to ensure that news is spread between customer and team.	

Agile software process	Responsible to the overall project's progress	Schuh (2005)
	Knowledgeable in Agile process	Tsun & Dac-Buu(2008)
	Has adaptive management style	Tsun & Dac-Buu (2008)
	Responsible to maintain relationship with customers	Tsun & Dac-Buu(2008); Schuh (2005)
	Acts more like a facilitator than a foreman	Sliger (2006); Schuh (2005)
	Responsible to build team cohesion	Sliger (2006)
Secure software process	Adopt the security activities and become familiar with the security requirement of the system	Davis (2013); Microsoft (2012)
	Common understanding about the security needs must be reached among all applicable parties, including the customer	Microsoft (2012); McGraw (2006)

– The organization

The organization which adapts Agile should have essential characteristics that can support the implementation of Agile. Among them are encouraging customers' participation, providing cooperative organizational culture instead of hierarchical, provide facilities with proper Agile-style work environment and ensure that Agile way of software development is universally accepted (Sheffield & Lemetayer, 2013; Ani Liza et al., 2012b; Hoda, Noble, & Marshall, 2011; Tsun & Dac-Buu, 2008; Strode et al., 2009; Lindvall et al., 2002). Also, Sliger and Broderick (2008) point out that empowered team is important in the Agile team, thus organization should implement this. Furthermore, in order to ensure that security initiatives are successful, the organization must produce the security policy and ensures it is being implemented (Syed Irfan et al., 2010; Tondel et al., 2008). The organization also should provide a separate security team to engineer and evaluate the security of software (Microsoft, 2012; OWASP, 2006) and ensure that all members of the project team are aware of and involved with security engineering activities (Siponen et al., 2010; Syed Irfan et al., 2010; Lane, 2007; Ai et al., 2007; Torres et al., 2006) .

- The customers

The customers of Agile environment are expected to be able to give constant feedback (Lindvall et al., 2002), able to give commitment to the team (Misra et al., 2009; Tsun & Dac-Buu, 2008), able to present on-site (Tsun & Dac-Buu, 2008), as well as able to collaborate with the team (Misra et al., 2009), empowered to make decisions on behalf of other stakeholders (Boehm & Turner, 2003; Paetsch et al., 2003), knows the business domain and knowledgeable (Schuh, 2005; Boehm & Turner, 2003), do not feel afraid to be responsible to the decisions made, understands and appreciates objective of the project and willing to compromise (Schuh, 2005).

- The technology factor

The technology factor is comprised of tools and techniques, as well as the use of standard and procedure. The practices in this factor are the same from the perspective of Agile and secure software processes. The use of standard and procedure is obviously vital to ensure that the software process implemented correctly throughout the organization (Limaye, 2011; Wheeler & Duggins, 1998; Addison & Vallabh, 2002). On the other hand, the use of tools and technique is imperative to accelerate the development process (Ai et al., 2007; Yazrina et al., 2002).

- The project constraint factor

The project constraint factor contains the budget and schedule. Both are vital in order to ensure a project's success. Nevertheless, the schedule must be realistic to avoid under pressure work (Procaccino et al., 2005; Linberg, 1999). The schedule for Agile implementation is dynamic and accelerated in order to handle the fast changing

environment (Wells, 2013; Tsun & Dac-Buu, 2008). Also, the budget is not estimated upfront, rather, it is estimated based on the features (Wells, 2013; Tsun & Dac-Buu, 2008).

- The environment factor

The organization must provide necessary environmental facilities to encourage the successful implementation of projects (Lindvall et al., 2002; Linberg, 1999).

As discussed in Section 2.2.4, the second issue that needs to be addressed in the process based software certification models is the synthesis technique, whereby the weight values allocation need to be addressed for the evaluation criteria. Accordingly, the MCDM technique is suitable for that purpose. It is discussed in the next sub section.

2.3 Multiple Criteria Decision Making (MCDM)

MCDM refers to “*making preference decision over the available alternatives that are characterized by multiple, usually conflicting attributes*” (Triantaphyllou, 2000). In MCDM, the assignment of weight is an important step (Ishizaka & Labib, 2011; Brugha, 2004; Yoon & Hwang, 1995). Among the MCDM techniques commonly being used for evaluation are the Weighted Sum Method (WSM), Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), outranking methods fuzzy multiple criteria and Analytic Hierarchy Process (AHP). The brief descriptions of these techniques are provided in Table 2.17.

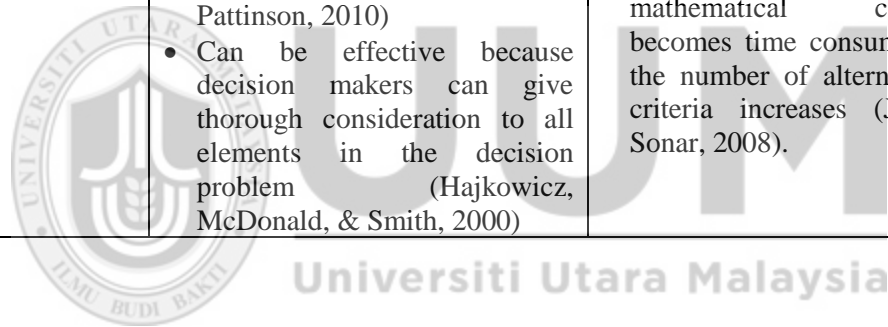
Table 2.17

MCDM Techniques

Techniques	Descriptions	Strengths	Weaknesses	Examples of existing studies using the MCDM techniques
WSM	<p>The score is calculated by multiplying the comparable rating of each attribute with the weight values assigned to the attributes and summing these values over all attributes</p>	<ul style="list-style-type: none"> • Easy to be used (Jadhav & Sonar, 2008) • It is a proportional linear transformation of the raw data which means that the relative order of magnitude of the standardized scores remains equal (Afshari, Mojahed, & Mohd Yusuff, 2010) 	<ul style="list-style-type: none"> • Weights to attribute are assigned arbitrary. • Common numerical scaling is required to obtain score in this method. (Jadav & Sonar, 2008) • Does not provide technique for determining weight • Inherent problem with the formula losing dependency information between attributes (Maxville, Armarego, & Lam, 2004) 	<p>Jain & Raj (2013); Savitha & Chandrasekar (2011); Afshari et al. (2010); Setiawan, Bouk, and Sasase (2008)</p>
TOPSIS	<p>TOPSIS was first introduced by Hwang and Yoon (1981). Referring to this technique, the best alternative is the one which is nearest to the positive ideal solution and the farthest from the negative ideal solution. This shows that an ideal solution is composed of all best values which can be attained by the evaluation criteria, while the negative ideal solution is comprised of all worst values which can be</p>	<ul style="list-style-type: none"> • It is a sound logic that represents the rationale of human choice. • It is a scalar value that accounts for both the best and worst alternatives simultaneously. • It provides a simple computation process that can be easily programmed into a spreadsheet. • The performance measures of all alternatives on attributes can be visualized on a polyhedron, at least for any two dimensions. (Shih, Shyur, & Lee, 2007) • Can be computed easily and easy to be understood since the final 	<ul style="list-style-type: none"> • Do not provide specific technique for weight allocation (Rao & Davim, 2008). • The operation of normalized decision matrix in which the normalized scale for each criterion is usually derived a narrow gap among the performed measures. That is, a narrow gap in the TOPSIS method is not good for ranking and cannot reflect the true dominance of alternatives. • The risk assessment for a decision maker is never 	<p>Chen, Lin, Lee, Chen, & Huang (2012); Joshi, Banwet, & Shankar (2011); Rao & Davim (2008); Gumus (2009); Ertugul & Karakasoglu (2009)</p>

	obtained by the evaluation criteria Benitez, Martin, & Roman (2007)	results are calculated based on the definite values given directly by the experts (Wang, Cheng, & Huang, 2009)	considered (Tsaur, 2011)	
Outranking methods	<p>Outranking methods orders the alternatives by finding the ones that outperform or dominate. This is done by differentiating the preferred alternatives with non preferred ones by establishing outranking relationships. The preferences are modelled by using binary values, for instance, A outranks B. When the comparisons are made between candidates on each attribute, the issue of units and attribute types are removed (Linkov & Moberg, 2012; Figueira, Greco & Ehrgott, 2005).</p>	<ul style="list-style-type: none"> • Able to consider both objective and subjective criteria. • Least amount of information required from the decision maker (Mollaghasemi and Pet-Edwards, 1997 as cited in Kunda, 2003). 	<ul style="list-style-type: none"> • Does not indicate how much an alternative outranks another alternative, for example ELECTRE I (Yatsalo et al., 2007; Mollaghasemi and Pet-Edwards, 1997 as cited in Kunda, 2003). • Issues with explaining the reasoning for decisions and a complete ranking may not be possible, for instance with ELECTRE I (Ruth, 2008; Kunda, 2003). • PROMETHEE does not provide specific guideline for determining weights (Behzadian, Kazemzadeh, Albadvi, & Aghdasi, 2010; Macharis, Springael, Brucker, & Verbeke, 2004). 	Taillandier & Stinckwich (2011); Dagdeviren (2008); Bollinger & Pictet (2008)
Fuzzy multiple-criteria	It handles the vagueness and uncertainty of users' thought and perception, where the linguistic terms of criteria are represented by the fuzzy numbers. It uses the linguistic terms to assign the weights of the criteria (Patil & Kant, 2014; Al Tarawneh, 2014; Ruth, 2008)	<ul style="list-style-type: none"> • Suitable for subjective problem • Beneficial to tackle the ambiguities involved in the process of linguistic estimation (Patil & Kant, 2014) 	<ul style="list-style-type: none"> • Associated with criteria whose values are not numbers, but words or sentence in natural language • Does not support the objective data 	Chou, & Cheng (2012); Gumus, Yayla, & Gurbuz (2011); Chou, Chang, & Shen (2008); Benitez et al. (2007)

AHP	<p>It decomposes the evaluation criteria and estimating the software alternatives using the hierarchy structure where the weights and the score of the alternatives are calculated through the pair wise comparisons.</p>	<ul style="list-style-type: none"> • The decision making problem are structured into a hierarchy, which helps in understanding and simplifying the problem. • Can be applied for individual or group decision making (Jadhav & Sonar, 2008) • The calculation method can be easily integrated into any software structure. • The calculation takes reasonable time to be completed. (Xuhua & Pattinson, 2010) • Can be effective because decision makers can give thorough consideration to all elements in the decision problem (Hajkowicz, McDonald, & Smith, 2000) 	<ul style="list-style-type: none"> • Pairwise comparison is arbitrary because they are subjectively interpreted (Eckman, 1989). • Relies on expertise and knowledge of decision makers (Xuhua & Pattinson, 2010). • If the number of alternatives, evaluation criteria or user requirements needs changes, the final score need to be calculated again (Jadhav & Sonar, 2008; Lin, Hsu, & Sheen, 2007). • The pairwise comparisons and mathematical calculations becomes time consuming when the number of alternatives and criteria increases (Jadhav & Sonar, 2008). 	<p>Al-Tarawneh (2014); Cay & Uyan (2013); Zhou & Liang (2013); Chen, Pham and Yuan (2013); Padumadasa, Colombo, & Rehan (2009); Kunda (2003); Lai, Wong, & Cheung (2002); Akarte, Surendra, Ravi, & Rangaraj (2001); Jung (2001)</p>
-----	---	--	--	--



This study adapted the AHP technique for assigning the weight values due to several reasons. The advantage of the AHP technique is that it provides a systematic approach for synthesizing information, by providing a structured hierarchy. The hierarchy of specific criteria and sub criteria helps the understanding of problem and simplify the problem by providing a better focus during the weight allocation for criteria and sub criteria (Ishizaka & Labib, 2011; Jadhav & Sonar, 2008). This is the advantage gained by this study, as it involves with numerous factors, sub factors and evaluation criteria.

By using the hierarchy three, the criteria are systematically organized. (Rafikul & Shuib, 2006). Also, it is the most widely used technique in various fields and has been considered as the most reliable one (Rao & Davim, 2008; Triantaphyllou & Mann, 1995). On top of these, AHP can increase the accuracy of the judgments, as the judgments are not made arbitrarily. This is because the accuracy of judgments made is largely influenced by the quality of input quantity (Crostack, Hackenbroich, Refflinghaus, & Winter, 2007).

Moreover, AHP is appropriate to be used for group decision making by reaching favorable agreement among the group members (Marjani, Soh, Majid, Mohd Sofian, Nur Surayyah, & Mohd Rizam, 2012; Lai et al., 2002; Liberatore & Nydick, 1997). This is important as the study involves group decision making in determining the weight values. On top of that, AHP includes the consistency checking in the judgment, which is essential in order to ensure that the judgments have been made consistently (Liberatore & Nydick, 1997). AHP is discussed in detail in the next sub section.

2.3.1 Analytic Hierarchy Process (AHP)

AHP (Saaty, 2008; Saaty, 1990) enables decision-makers to structure a multiple criteria decision making problem into a hierarchy, whereby there are at least three levels in a hierarchy. The overall goal of the problem is placed at the top, the evaluation criteria in the middle and the alternatives at the lowest level. However, in this study, the hierarchy tree only contains the goal and several levels of evaluation criteria, without the alternatives. This is because AHP is used only for obtaining the weight values. The weight values are obtained through pair wise comparisons which are performed among the evaluation criteria of each level.

From the pair wise comparisons, a normalized ranking is calculated by using the eigen value method. Besides, there are other simpler methods that can be used, which are the normalization of row average (NRA), normalization of the reciprocal sum of columns (NRC), average of normalized columns (ANC) and normalization of the geometric mean of the rows (NGM) (Hsiao, 2002). The basic steps involved in the AHP technique are as provided below (Vaidya & Kumar, 2006):

1. Identify the evaluation criteria that influence the quality of software process.
2. Structure the evaluation criteria in a hierarchy which comprises the factors, sub factors and evaluation criteria.
3. Construct pair wise matrixes which requires $n(n - 1)/2$ comparisons.
4. Compare the importance of factors/sub factors/evaluation criteria in each pair wise matrixes. The importance values are as depicted in Table 2.18.
6. Synthesize the pair wise comparisons to find the weights, consistency index (CI), consistency ratio (CR).
7. If the CR is less than 0.1 the weight is usable, otherwise the process is repeated.

Table 2.18

AHP Preference Scale (Saaty, 1990)

Intensity of Importance	Definition
1	Equal importance
3	Moderate importance of one over another
5	Essential or strong importance
7	Very strong importance
9	Extreme importance
2,4,6,8	Intermediate values

Review from literature shows that AHP has been widely used in different domains, such as planning, selecting the best alternative, resource allocations, resolving conflict and evaluation (Vaidya & Kumar, 2006). In the area of software process assessment, AHP has been used for the purpose of software process improvement (Jung, 2001), however, it has not been used for the software process certification. Table 2.19 summarizes the AHP usage in the assessment/evaluation, since this study focuses on the assessment.

Table 2.19

The Existing AHP Studies on Evaluation

Descriptions	Methods	Authors
This study utilized AHP for preference reallocation, which is the most important phase of land consolidation. At the same time, the traditional method of preference reallocation, which is interview-based reallocation model, was used.	ANC	Cay & Uyan (2013)
AHP was performed to evaluate the network course in China. The AHP technique was used for the evaluation.	NRA	Zhou & Liang (2013)
AHP was utilized to help firms in making decisions in evaluating potential outsourcing partner. There were four evaluation criteria used, which are: cost competitiveness, human resources, business and economic environments and	Eigenvalue	Chen, Pham, & Yuan (2013)

government policies and legal framework.

The component based software (COTS) is evaluated by using the AHP technique. There are five main criteria for evaluating the COTS software, which are quality, domain, architecture, operational environment, and vendor.	ANC	Al Tarawneh (2014)
AHP was performed to select and evaluate tender. In order to enhance the efficiency rate and accuracy of the final tender decision, the tendering process is automated by changing the manual process to the use of website. Therefore, the process can be performed 24/7 and this gives convenient to the suppliers and give flexibility to decision makers.	ANC	Padumadasa et al. (2009)
AHP is applied in a framework for evaluating and selecting component-off-the-shelf (COST) software components. This framework use the AHP technique to decompose the requirements into hierarchical criteria set and calculate the weight by using pair wise comparison.	Eigenvalue	Kunda (2003)
Evaluate and select three multimedia authorizing systems. They found that AHP is more preferable than Delphi, another group decision making approach. Besides, AHP is found to be more conducive to consensus to building in group decision setting.	Eigenvalue	Lai, Wong et al. (2002)
Evaluate supplier with 18 different criteria which are categorized into four groups, namely, product development capability, manufacturing capability, quality capability and cost and delivery.	NGM	Akarte et al. (2001)
AHP was utilized to rate the process attribute in the SPICE-based process assessment. Mainly AHP was utilized to solve boundary problems faced by the assessors. To compute the final score, the simple additive weighting technique was used.	NGM	Jung (2001)

Referring to Table 2.19, the commonly applied method for calculating the weight in AHP is the eigenvalue method. However, the NGM method is applied in this study since the approximation to the correct answer is higher (Coyle, 2004). Additionally, it is statistically better and easier to calculate (Ishizaka & Labib, 2011; Crawford & Williams, 1985). On top of that, geometric mean of rows and columns provide the same ranking due to the absence of rank reversals, which is not necessarily the case with eigen value method (Ishizaka & Labib, 2011). Also, it gives good approximation

(Hsiao, 2002). The next section discusses another MCDM technique used in this study, which is Weighted Sum Method (WSM).

2.3.2 Weighted Sum Method (WSM)

WSM (Mollaghasemi, 1997), which is also known as Simple Additive Weighting technique is one of the simplest methods in MCDM. It is a widely used method for calculating the final grade values in the multiple criteria problems (Kontos, Komilis, & Halvadakis, 2005). The total score for each alternative then can be computed by multiplying the comparable rating for each attribute by the importance weight assigned to the attribute and then summing these products over all the attributes. The sum of the weight allocated to each attribute must be 1.

Jadhav & Sonar (2008) pointed out that the main advantage of the WSM is its ease of use. However, this technique does not provide the technique for weight allocation explicitly (Jadav & Sonar, 2008; Maxville et al., 2004). Therefore, other weight allocation techniques need to be adapted, instead of assigning weights arbitrarily. The succeeding equation is used for the calculation.

$$A_i = \sum w_j \cdot x_{ij} \quad (2.1)$$

Where

A_i = Score for i^{th} alternative

w_j = Weight for j^{th} criterion

x_{ij} = Score of the i^{th} alternative in term of j^{th} criterion

Table 2.20 provides the existing studies related to the assessment/evaluation by using the WSM technique.

Table 2.20

The Existing WSM Studies on Evaluation

Descriptions	Technique for assigning weight	Authors
WSM was used to build a decision support system which evaluates alternatives in the procurement of goods based on particular criteria. The evaluation is made based on the benefit and cost criteria. Finally, the winner of the procurement is determined based on the achieved score.	No specific technique	Nugraha (2013)
This study combined the WSM with the weighted product method (WPM) and AHP in order to determine the flexibility in manufacturing sector, which is the Flexible Manufacturing System (FMS). The AHP is used to obtain the weights and the other two techniques are used for ranking.	AHP	Jain & Raj (2013)
This study applied the WSM to evaluate and select personnel for an organization, based on seven criteria. Finally the best personnel was selected.	AHP	Afshari et al. (2010)
The WSM is used to evaluate the suitable landfill site. There were four main criteria used for the evaluation. Each of these criteria has its sub criteria and spatial attributes.	AHP	Kontos et al. (2005)

Referring to the existing studies on WSM, majority of them utilized AHP as the technique for determining the weight values for the evaluation criteria. This confirms that both techniques are applicable to be used in the software process certification. Thus, as stated earlier, this study adapts the AHP for weight calculation. Additionally, to calculate the score of assessment and certification in the ESPAC Model, the WSM is adapted. The discussion is continued in next section with the measurement approach used for the software process certification.

2.4 Measurement Approach in Software Process Certification

Software process assessment and certification involves with measuring the effectiveness and efficiency of the software process. To measure successfully, the top-down approach is needed. This study adapted the Quality Function Deployment (QFD) approach as it is being used widely in various industries and businesses, including software development (Zultner, 1992). On top of that, it is an appropriate tool for making a consistent, non-intuitive decision-making processes with a structured approach (Bouchereau & Rowlands, 2000). Therefore, in this study, this approach assists in organizing the evaluation criteria and the assessed Agile and secure software processes in a structured manner.

QFD is a quality tool that helps to translate the Voice of the Customer (VoC) into new products systematically. It involves building the matrix which is House of Quality (HOQ), as depicted in Figure 2.3. The VoC (the WHATS) which is customers' requirements are matched with the appropriate technical response along the top (the HOWS). Therefore, the appropriate technical response for each customer's requirement can be organized systematically (Cohen, 1995). Additionally, each of the customers' requirements is assigned with the weight value, which is right after the customers' requirements on the left side.

The rating scales are represented as the relationship matrix, in the middle of HOQ. At the end, the technical ratings are calculated for each HOWs. It indicates the basic importance of the HOWs in relation to the WHATs. These values are obtained by using the weighted sum method (Chan & Wu, 2005; Park & Kim, 1998). Finally, the

importance ratings for the WHATs are determined, which is placed on the right hand side of the HOQ. The customer requirements which obtained higher relative importance should receive higher attention for future improvements (Chan & Wu, 2005). This study adapted the structure of HOQ to organize the evaluation criteria and the Agile and secure software practices.

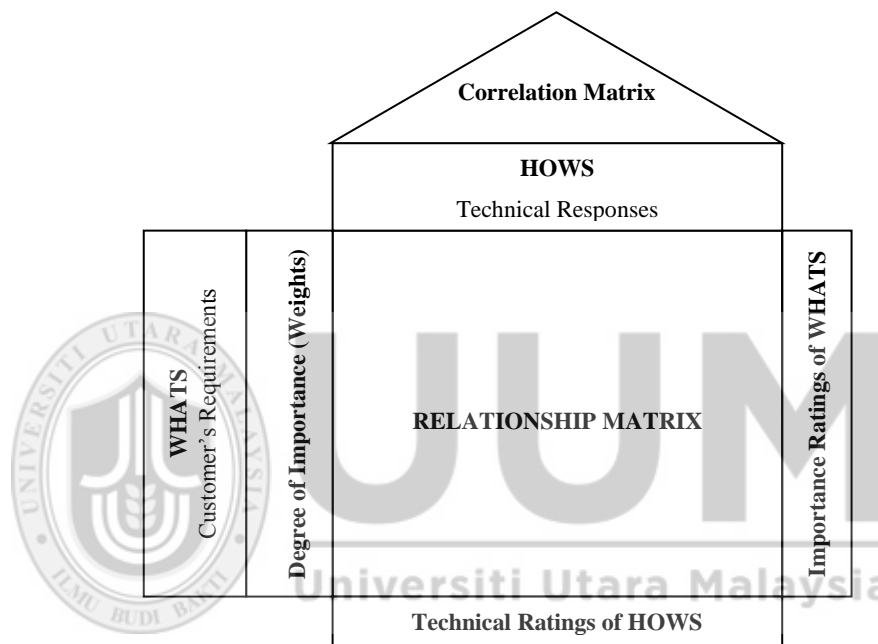


Figure 2.3. The basic structure of HOQ (Cohen, 1995)

The major functions of QFD have been prolonged from product development to wider fields such as quality management, product design, planning, engineering, decision making, management, manufacturing, customers' needs analysis, software systems, and services as well (Lai-Kow & Ming-Lu, 2002; Bouchereau and Rowlands, 2000). In the area of software process, it has been used for software process improvement by several studies; Richardson and Ryan (2001), Zultner (1992), SAP (Hierholzer,

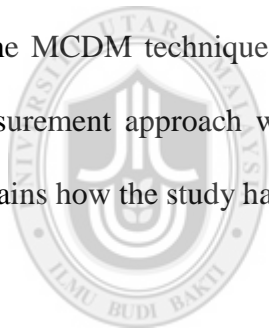
Herzwurm, & Schlang, 2003), Yan (2008) and more recently by Wei and Yonghui (2013).

On top of that, the QFD approach is also used for evaluations, for instance, quality performance assessment (Yumin & Jichao, 2006), evaluation of digital library (Garibay, Gutierrez, & Figueroa, 2010) and evaluation of technical textbook (Chen & Chen, 2001). Therefore, considering its ability to structure the evaluation criteria and the assessed practices in a structured manner and its appropriateness for assessment, as well as its suitability for software process, thus this approach was adapted for structuring the reference standard of the ESPAC Model.

One of the important and crucial steps in QFD is determining the importance of the weights for the customer requirements (Alinezad, Seif, & Esfandiari, 2013; Garibay et al., 2010). As mentioned earlier, AHP is the widely used and reliable technique for deriving weight values (Ishizaka & Labib, 2011; Jadhav & Sonar, 2008). The AHP and QFD have been used in combination in numerous studies, among them are Taghizadeh and Mohamadi (2013), Dai and Blackhurst (2012), De Felice and Petrillo (2011), Tu, Zhang, He, Zhang and Li (2011) and Hsiao (2002). Similar to the abovementioned studies, the AHP and the QFD are adapted in this study.

2.5 Summary

This chapter has successfully discussed about the existing work found in the literature regarding the software process certification and related issues. The discussion is started with the overview of software certification. The methods and approaches in software certification are discussed. Subsequently the discussion is further continued with the current issues in software process certification. It indicates the gaps identified in the literature which are addressed by this study. The first issue is the needs of incorporating the Agile and secure software processes in the reference standard, while the second issue is to allocate the weight value during the synthesis process in the assessment and certification process. The Agile and secure software processes, as well as the MCDM techniques are elaborated in detail. At the end, the software process measurement approach which is the QFD approach is explained. The next chapter explains how the study has been conducted.



UUM
Universiti Utara Malaysia

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

The previous chapter provides review on the literature. It gives understanding on the issues related to the domain of the study. This chapter discusses the method used and processes involved in answering the research questions and achieving the objectives as stated in Chapter 1. It starts by presenting the research design in Section 3.2, continues with the phases of the study from Section 3.3 until Section 3.6. The chapter is ended with a summary in Section 3.7.

3.2 Research Design

To achieve the aims of this study, deductive approach, which is also known as ‘top-down’ approach, was performed (Srivastava & Shailaja, 2011; Trochim, 2006). The deductive approach begins from general ideas and ends with specific conclusions, whereby the conclusions are made based on a known general premise or something known to be true (Cooper & Schindler, 2008). Using this approach, the theory and concept of software process certification were derived from the theoretical and the exploratory studies’ findings in order to construct the proposed model. Then, the proposed model was applied and validated by the potential users of the model who are the software practitioners.

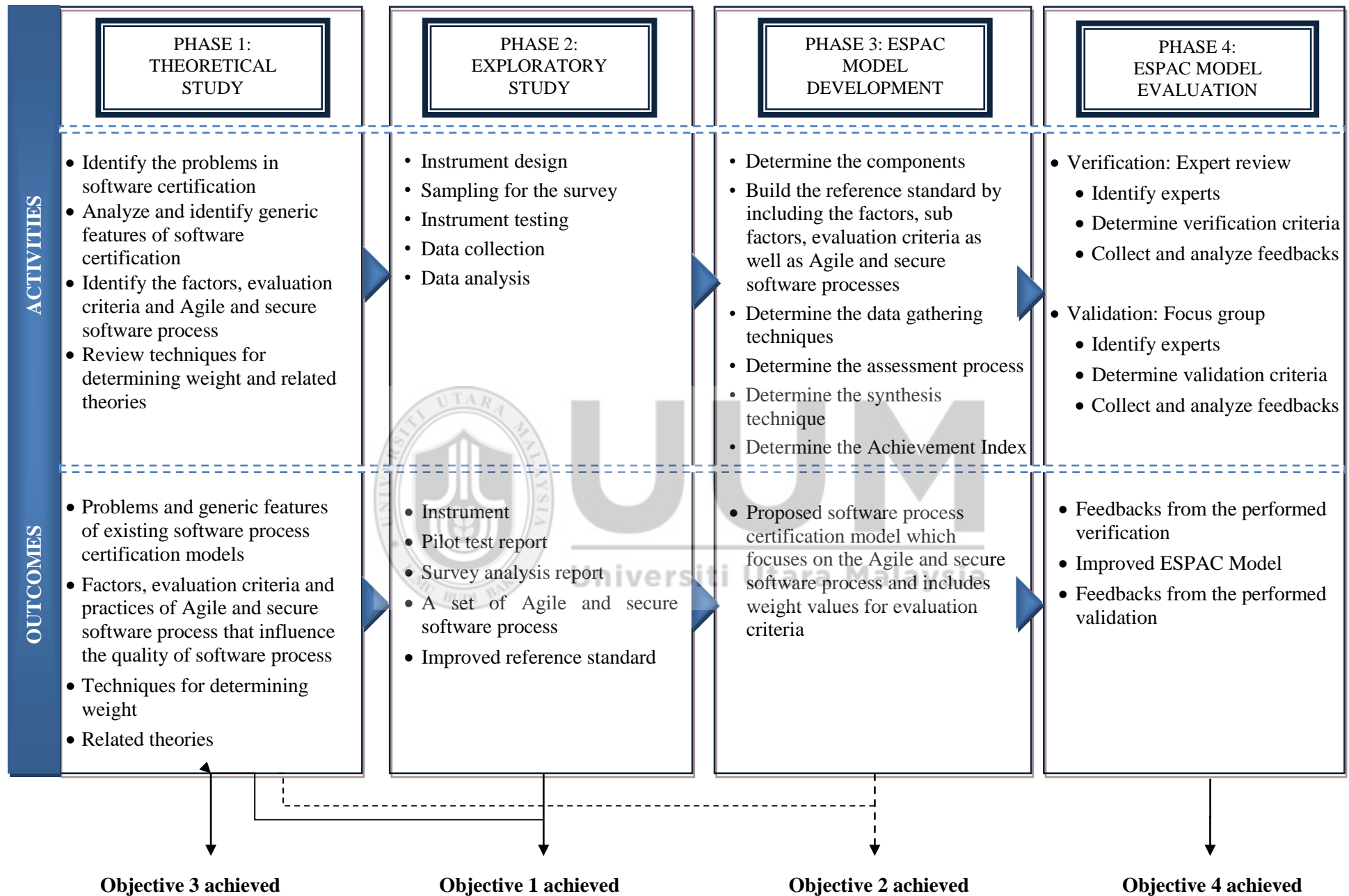


Figure 3.1. Research framework

Figure 3.1 depicts the research framework which illustrates the phases, activities involved and outcomes for each phase. Additionally, it also clearly illustrates the achievement of the objectives of the study. There were four phases in conducting this study, which are 1) Theoretical study, 2) Exploratory study, 3) ESPAC Model development and 4) ESPAC Model evaluation. Each phase is explained in detail in the next sub sections of this chapter.

3.3 Phase One: Theoretical Study

The theoretical study was performed by reviewing the literature in order to identify the issues and gaps related to the domain of the study. Consequently, the main ideas were gained through the literature by reading the printed and online references. Among them are journals, proceeding papers, standards documentation, books and unpublished thesis. From the knowledge gained, the problem and scope of the study were defined. Among the activities conducted were identifying the problems in software certification, analyzing and identifying the generic features of the existing software certification models, and identifying the factors, evaluation criteria and practices of Agile and secure software processes. Since software certification involves multiple criteria assessment, the techniques for determining weight and related theories were investigated from the MCDM techniques.

3.4 Phase Two: Exploratory Study

The second phase of the study is the exploratory study on the current practices of software certification in relation to the Agile and secure software processes among Malaysian software practitioners. More specifically, the objectives of the study are: 1) to

study the software practitioners' current practices of the Agile and secure software processes, 2) to investigate the software practitioners' opinion on the Agile and secure software processes that are important in producing high quality software, 3) to examine the software practitioners' opinion on the importance of the Agile and secure software processes in producing high quality software 4) to investigate the software practitioners' opinion on the good characteristics of those involved in the Agile and secure software processes, and 5) to inspect the software practitioners' awareness on the importance of software certification and its implementation.

The exploratory study was conducted using the quantitative approach, which is survey. The survey was used since it is a useful and powerful approach to measure opinions and awareness (Sekaran & Bougie, 2010). In order to realize this aim, the self-administered instrument was used because of several advantages such as cost effective, easy to analyze data, wider area coverage, and high degree of secrecy. Furthermore, the instrument allows more time for respondents to think, is perceived as more anonymous and reduces biasness as it is not influenced by personel qualities of researcher (Sekaran & Bougie, 2010; Cooper & Schindler, 2008). There were five main activities performed in conducting the exploratory study, as laid out in the next sub sections.

3.4.1 Instrument Design

The instrument was designed based on the guideline provided by Gay, Mills, & Airasian (2006) and Zikmund, Babin, Carr and Griffin (2010). Among the suggestions are as follows: the instrument should be attractive and concise, only consider the items that are related to the objectives of the study and use simple and understandable language. Any

leading and loaded questions must be avoided, as well as being specific to avoid ambiguity. The instrument should then be pilot tested to ensure it is not too lengthy.

The content of the instrument was established by referring to various theoretical findings and adapting several existing instruments which emphasize on the Agile and secure software processes. Among the existing instruments are those by VersionOne (2011), Corbucci et al. (2011) and Elahi et al. (2011) (more detail in Appendix A). There are four (4) main sections in the instrument: 1) demographic information, 2) Agile software process, 3) secure software process, and 4) the implementation of software certification. Refer to Section 4.2 for further explanation.

3.4.2 Sampling for the Survey

The purposive sampling (judgment) was used in this study, which involved the selection of unique sample with specific feature that is important for the study (Nardi, 2003). The sample was chosen among the software practitioners in Malaysia. The main constrain of selecting these software practitioners is that they are very busy people and cannot be reached easily. Consequently, this sampling technique is appropriate since it is intended to be used when only a limited number or category of people can be approached (Sekaran & Bougie, 2010). Furthermore, the sample is guaranteed to meet the objectives of the study since they are chosen based on specific characteristics (Zikmund et al., 2010). A total of 114 samples participated in this study, which should be acceptable, referring to Bailey's (2008) recommendation that the sample size of 100 is sufficient. Additionally, Fisher (2007) and Sekaran (2003) stated that the minimum sample acceptable size for statistical analysis is 30. Refer to Section 4.3 for further explanation.

3.4.3 Instrument Testing

The instrument was validated through a pilot study, which was answered by 32 software practitioners in Kedah and Penang. The number of appropriate subjects for the pilot study is suggested as between 25 to 100, which constitutes the subjects from the target population (Cooper & Schindler, 2011). The key purposes of conducting pilot study are to ensure the instrument's validity, completeness of the included items and readability (to avoid misinterpretation of questions). The pilot study helped to improve the instrument by simplifying the questions to be more readable and understandable, reorganizing the presentation of questions and reducing the number of questions. Additionally, the time taken to answer the instrument was determined as well. More details on the pilot study can be obtained in Section 4.4. The finalized draft of the instrument is provided in Appendix B.

3.4.4 Data Collection

The potential respondents were contacted through telephone to ask their willingness to participate in the study. Then, the instruments were delivered via mail postage or meeting them face-to-face. Additionally, an online survey was created and the link was emailed to the potential respondents who agreed to participate in the study. By using the online survey, the response rate was higher. Furthermore, it is beneficial since the turnaround time for results are shorter. Similarly, the turnaround time from drafting the instrument to the execution of the study is shorter since everything is performed online and accessible on real-time basis (Cooper & Schindler, 2011).

3.4.5 Data Analysis

Data obtained from the exploratory study were coded by using the Software Package for Social Science (SPSS) Version 14.0. The analysis was performed by using descriptive statistical analysis. Among the analysis used were frequency, mean and cross tabulation. The main findings from the exploratory study are discussed in Section 4.7.

3.5 Phase Three: ESPAC Model Development

The next phase was to develop the proposed model. The proposed model was constructed based on three main elements, which are: 1) the evaluation components as recommended by the Evaluation Theory (Scriven, 1991), 2) the outcomes from the theoretical study such as the problems and generic features of existing software process certification models, current software processes which influence the quality of software and the techniques for obtaining weights, and 3) the findings from the exploratory study which highlights the important Agile and secure software processes, characteristics of people who involve in both software processes, as well as data gathering techniques for certification. In addition, the existing software process certification models and standards were referred as the baseline models to get insight of the software process certification. Among them are the SPAC Model, CMMI version 1.3, ISO/IEC 15504, ISO/IEC 27001 and ISO/IEC 21827.

There are seven main components in the proposed model, which are the target, evaluation criteria, reference standard, data gathering techniques, assessment process, synthesis technique and Achievement Index. They are discussed further subsequently.

3.5.1 Defining the target

The target for the assessment and certification process is determined to decide on the scope of assessment. It is determined by referring to the previous studies which relates to the software process assessment and certification models. Thus, the target is the software process implemented in the project that has been completed and ready to be delivered to customers. In addition, the scope of the assessed software process is limited to the Agile and secure software processes.

3.5.2 Defining the evaluation criteria

The evaluation criteria were defined for assessing the described target, which are the Agile and secure software processes. Thus, the goal is to assess the quality of Agile and secure software processes. Additionally, other influencing factors taken into consideration are the people, technology, project constraint and environment. Each of them is decomposed into at least one sub factor that are measureable. For each sub factor, at least one evaluation criterion is defined. They are structured by using hierarchy tree, as depicted in Figure 3.2. This hierarchical structure is adapted from the AHP technique, whereby the goal is placed on top of the hierarchy tree and continues with the factors, sub factors and evaluation criteria in the subsequent levels. This hierarchy tree was used as the basis for developing the reference standard. Further explanation can be found in Section 5.3.2.

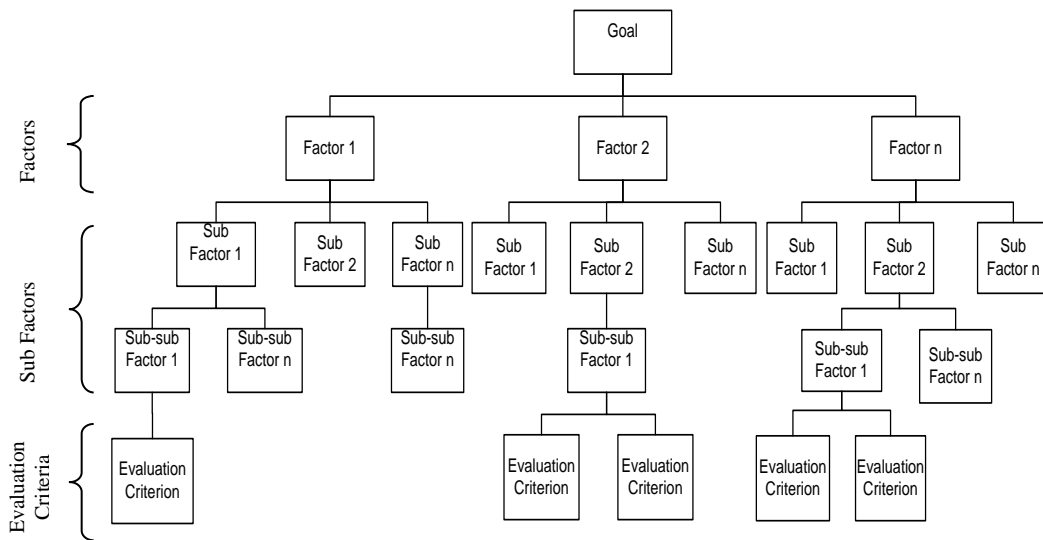


Figure 3.2. The structure of evaluation criteria

3.5.3 Building the Reference Standard

Based on the hierarchy of the evaluation criteria, the Agile and secure software practices were defined. Each evaluation criterion is assigned with appropriate practices to achieve the specified evaluation criterion. Furthermore, each evaluation criterion is assigned with weight value and the score achieved. They are arranged by adapting the Quality Function Deployment approach (Cohen, 1995; Zultner, 1992). More details are discussed in Section 5.3.3.

3.5.4 Determining the Data Gathering Techniques

The proposed model adapted multiple techniques for data gathering. These techniques were adopted from the existing software process certification model (Fauziah, 2008), as well as the outcomes from the exploratory study. Among the data gathering techniques are document review, interview and observation. They act as evident that are needed for

assessing and certifying software process which focuses on the Agile and secure software processes. More details are explained in Section 5.3.4.

3.5.5 Determining the Assessment Process

The assessment process gives guidance on how to perform the assessment and certification. Referring to Ares et al. (2000), a comprehensive and rigorous model should include the assessment process. Therefore, the assessment process was determined. There are three (3) main phases for conducting the software process certification, which are pre-assessment, assessment and post-assessment. Each of these phases has several processes and activities. Basically, the structure of the assessment process was adapted from the SCAMPI for CMMI version 1.3 (SCAMPI Upgrade Team, 2011), SPAC Model (Fauziah, 2008) and Lascelles and Peacock (1996). The ESPAC Model proposes the collaborative self-assessment method to perform the assessment. They are further discussed in Section 5.3.5.

3.5.6 Determining the Synthesis Technique

Another main contribution of this study is the synthesis technique that has been improved by incorporating weight allocation for the evaluation criteria. This is realized by adapting the AHP technique for allocating the weight values. This model suggests the ideal weight value for each evaluation criterion to the assessment team, which was obtained after conducting the proposed model validation (Refer to Section 6.3.4 for more details). Having the ideal weight values will be useful for the assessment team especially when they are novice in conducting software process assessment and certification. On the other hand, if the assessment team consists of experienced assessors, they can assign the weight

values based on their expertise and experience. This is to ensure flexibility in determining the weight. Moreover, the WSM was adapted for calculating the scores for the assessment. The detailed implementation of the synthesis technique is provided in Section 5.3.6.

3.5.7 Determining the Achievement Index

To determine the Achievement Index, the score ranges are adapted from the ISO/IEC 15504 (Galín, 2004; Jung, 2001) and Patel and Ramachandran (2009). There are two types of achievements, which are quality and certification levels. Further explanation can be obtained from Section 5.3.7.

3.6 Phase Four: ESPAC Model Evaluation

With the intention of ensuring that the ESPAC Model has been constructed conforming to its specification and ensuring that it performs according to the users' expectation, the evaluation was performed (Sommerville, 2007). It was evaluated through two stages which are verification and validation. They are discussed further in the next section.

3.6.1 Verification Stage

The verification is performed in order to verify whether the proposed model conforms to its specification (Sommerville, 2007) and ensures that all required components are present in right quantity (Chemuturi, 2011). In this study, the verification stage was intended to verify: i) the AHP technique used in the proposed model, and ii) the factors, sub factors and the Agile and secure software processes included in the proposed model. To accomplish this, the expert review was used because it can be easily conducted, costs

less and is faster. Moreover, it has been accepted as a significant way to detect and remove defects (Komuro & Komoda, 2008; Wiegers, 2002). Basically, there were three activities involved in verifying the proposed model:

i. Identifying the experts

The experts were chosen among the academicians (knowledge experts) by following the characteristics of experts as suggested by Rogers and Lopez (2002) and Hallowell and Gambatese (2010). The characteristics include i) currently attached to the field of the study under examination, ii) hold an advanced degree (PhD.), iii) faculty members at an accredited university, iv) authorship, and v) have at least 5 years of experience. Additionally, as the proposed model is intended to be used by the software practitioners, therefore, they were included as the domain experts to perform the verification as well as to give their insights from the real life environment point of view. The characteristics of the domain experts are discussed in the validation stage section.

ii. Determining the verification criteria

The AHP technique was verified by ensuring the correctness of performing the steps as well as the outcomes of the steps, which was adapted from Goerigk and Hoffmann (1999) and Moody (1998). Furthermore, the factors, sub factors, as well as the Agile and secure software processes were verified for their comprehensiveness, understandability, accurateness and organization. These criteria were adapted from previous studies (Al Tarawneh, 2014; Behkamal, Kahani, & Akbari, 2009; Kunda, 2003). The checklists which were used to obtain comments from the experts can be found in Appendix G, H and I.

iii. Collecting and analyzing the feedbacks

The knowledge experts' feedbacks were collected and analyzed for further improvements. Detailed explanation can be found in Section 6.2.3 and 6.2.4.

3.6.2 Validation Stage

Validation is the process of determining whether a model meets users' expectation, as well as whether it represents the real world accurately from the perspective of the proposed usage (Sommerville, 2007). Therefore, with the aim of revealing the practicality of the proposed model in the real working environment, a focus group discussion was conducted, which was participated by domain experts. The focus group technique was chosen as the validation approach because it can provide valuable feedbacks quickly, as well as can be conducted easily (Martakis & Daneva, 2013; Kontio, Bragge, & Lehlota, 2008; Krueger, 1994). On top of that, it is a convenient way to collect data concurrently from the software practitioners, who are really busy and cannot be reached easily (Martakis & Daneva, 2013). This approach also has been used in the field of software engineering for evaluation or obtaining practitioners' experience (Daneva & Ahituv, 2011; Mazza & Berre, 2007; Kontio, Lehlota, & Bragge, 2004). It is also suitable for confirmation studies (Krueger & Casey, 2008; Morgan, 1998).

The key steps for performing the focus group were adapted from Martakis and Daneva (2013), Daneva and Ahituv (2011), Mazza and Berre (2007), Morgan (1998) and Krueger (1994). The three main activities were carried out to perform the validation as described subsequently.

i. Identifying the participants

The participants of the focus group were chosen by using the purposive sampling (Liamputtong, 2011). They were chosen based on four characteristics: 1) Agile software practitioners, 2) work in Kuala Lumpur or nearby area 3) have experience in secure software process, 4) have software development experience for more than 3 years.

ii. Determining the validation criteria

The validation criteria for the proposed model were determined by adapting them from the studies of Al Tarawneh (2014) and Kunda (2003) to reveal the success of the proposed model, as listed in Table 3.1. The feedbacks on the validation of the proposed model are discussed in Section 6.4. Similar to the verification stage, checklists were used to obtain feedbacks from the experts (Refer Appendix J).

Table 3.1

Validation Criteria for ESPAC Model

Evaluation criteria	Variables
Gain satisfaction	<ul style="list-style-type: none"> • Perceived usefulness • Decision support satisfaction • Comparing with current method • Cost-effectiveness • 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 • Ability to produce usable results • Completeness • Ease to implementation • Understandability (easy to understand)

iii. Collecting and analyzing the feedbacks

The participants' feedbacks were collected, analyzed and reported. Detailed explanation can be found in Section 6.4.

3.7 Summary

This chapter has elaborated the methodology that was used in performing this study, which is the deductive approach. It consists of four phases: theoretical study, exploratory study, ESPAC Model development and ESPAC Model evaluation. Each of the phases was performed in order to achieve the objectives indicated in Chapter One. By executing those phases, the enhanced software process assessment and certification model has been developed. This model enables the assessment and certification to be performed in wider perspectives and matches current business needs. Additionally, the quality and consistency of certification decision is improved. The proposed model is targeted to be used by software practitioners as a mechanism to assess and certify their software process, besides to help investors and customers in making investment decisions. At the end, the proposed model was evaluated through verification and validation. The next chapter discusses about the exploratory study conducted in this study.

CHAPTER FOUR

EXPLORATORY STUDY

4.1 Introduction

This chapter discusses about the findings from the exploratory study conducted among software practitioners in Malaysia. The exploratory study aims to investigate the current practices of software certification in relation to Agile and secure software processes implemented in the Malaysian software industry. The findings from this exploratory study facilitated the development of the proposed software process assessment and certification model which focuses on the Agile and secure software processes.

The discussion in this chapter starts with the instrument design in Section 4.2, continues with the sampling, instrument testing, data collection and data analysis in Section 4.3, 4.4, 4.5 and 4.6 respectively. The findings are presented in Section 4.7 followed by discussions in Section 4.8. This chapter ends with a summary in Section 4.9.

4.2 Instrument Design

The contents of the instrument were obtained from previous works such as those from Elahi et al. (2011) and Misra et al. (2009). Additionally, findings from the theoretical study as discussed in Chapter 2 were also utilized. In general, 7-point semantic differential scale ranging from *Extremely Not Important* to *Extremely Important* was used in most of the questions (Zikmund et al., 2010; Nardi, 2003). This scale is used since it is reliable and valid for many research purposes, flexible, easy to be adapted, as well as quick and economical to administer and score, as mentioned by Kerlinger (1973) in

Thompson and Stapleton (1979). It is also an alternative to reduce the acquiescence bias which is found in Likert Scale (Friborg, Martinussen & Rosenvinge, 2006) besides reducing the survey completion time (Chin, Johnson, & Schwarts, 2008). In addition, multiple responses questions and yes/no questions were also included. The objectives and sources for each question are presented in Appendix A. The instrument for this study consists of 32 questions with sub questions, organized in four main sections:

Section I: Demographic Information

This section assesses the respondent's background such as their position in the organization, years of experience and the sector of the organization they are attached to.

This section was answered by all of the respondents.

Section II: Agile Software Process

Since this section focuses mainly on the Agile software process, it was answered only by the respondents who had prior knowledge in it. Most of the questions in this section relates to the software practitioners' perception on the importance of the Agile software process in producing high quality software, as well as their familiarity and experience with the approach. Among the questions are the Agile principles and the Agile practices that they perform. These questions are important towards providing an insight on the important Agile software practices that have influence on the quality of software.

Section III: Secure Software Process

Similar to Section II, this section was only answered by the respondents who had prior knowledge in secure software process. From this section, the respondents who did not

have preceding knowledge on the approach were directed to Section IV. Basically this section investigates the awareness regarding the secure software process among software practitioners in Malaysia. Additionally, the current practices of the secure software practices are also explored.

Section IV: The Implementation of Software Certification

This section seeks for the software practitioners' opinion on the importance of software certification. Furthermore, questions regarding the assessment or audit as well as the techniques being used for the assessment/audit are asked. Finally, the standards currently used in the respondents' organizations are also inquired.

4.3 Sampling

In this study, a non-probability sampling was used, which is the purposive (judgmental) sampling. The possible software organizations were obtained from 1) the list of software SME organizations in Malaysia, 2) the private and government companies attained through the Internet and 3) contacting friends who are working in the software industry. The potential samples were identified from Kuala Lumpur and Selangor, as these are the places where software development companies are most located in Malaysia (Ani Liza et al., 2012b; Mohd Hassan, Md. Mahbubur, & Noor Maizura, 1996). Moreover, Kedah has a big technology park which places a number of International software companies, while Penang has many software industries concentrated there (Ani Liza, 2012). Therefore, effort in distributing the instruments concentrated in these states.

4.4 Instrument Testing

The instrument went through several rounds of reviews and revisions after it was constructed to ensure that the content is comprehensive and appropriate. Additionally, the layout of the instrument needs to be friendly, with clear instructions and understandable language. These characteristics were used to validate the instrument through a pilot study before distributing it and collecting the actual data from the selected sample.

The pilot study involved thirty two (32) respondents. This number of respondents is appropriate since Cooper and Schindler (2011) suggests the size of pilot group may range from 25 to 100. The instruments were distributed face-to-face, which involved project managers, system analysts and programmers who have at least 3 years' of working experience in the related field. Through the pilot study, the ambiguities that might arise, difficulties that might be faced when answering the instrument, misinterpretation on the questions and incompleteness of the items in the instrument can be identified. Additionally, the time and motivation for answering the questions can also be looked into.

The pilot study respondents agreed that the questions make sense and covers the domain of the Agile and secure software processes, as well as software certification. However, there are some suggestions for improving the quality of the instrument. Among the suggestions were to simplify the questions to be more readable and understandable, reorganize the presentation of questions and reduce the number of questions since the time taken to answer was too long. Consequently, the instrument was refined based on

their feedbacks. Refer to Appendix B for the final version of the instrument after the refinement.

4.5 Data Collection

From the lists of organizations obtained, the researcher contacted all of the potential respondents through telephone or email and asked their willingness to participate in the survey, within July and August 2012 (230 of them). Out of the possible contacted respondents, only 169 organizations actually involved in software development and willing to participate, while the other 61 of them refused to participate.

A total of 114 instruments are usable for the study, which is acceptable according to Bailey (2008), Fisher (2007) and Sekaran (2003). Different approaches were used to gather the data, namely face-to-face meetings, mail postages and online survey. Face-to-face meeting was used to ensure that the respondents clearly understand each question and answer them properly. Furthermore, if they have doubts on any question, they can immediately request for clarification. However, most of the respondents preferred to answer the survey via online or mail postage rather than face-to-face. Therefore, an online survey was created by using the SurveyMonkey and the link was emailed to the corresponding respondents who agreed to participate in the survey. The instrument was posted for two months in the Internet (October to November 2012). By using this increasingly popular way of data collection, the response rate was higher compared to the face-to-face meeting, as well as reduced the cost and can be done faster.

One month was allocated for the respondents to return the instruments. Reminders were sent to the ones who failed to do so. Unfortunately, there were 35 unreturned instruments, whilst 20 were rejected due to incomplete answers. The overview of the respondents is provided in Table 4.1.

Table 4.1

Overview of Respondents

Details	Details	Percentage
Number of respondents willing to participate	169	100
Unreturned instruments	35	20.7
Face-to-face respondents	14	8.3
Online respondents	84	49.7
Mail postage respondents	16	9.5
Rejected/ Incomplete online survey	20	11.8
Total usable	114	67.5

4.6 Data Analysis

Data obtained from the survey were analyzed using descriptive statistical analysis, whereby it is not intended to explain or show any causal relationships between the variables. On the other hand, it focuses on describing the respondents' opinion or the frequency of certain events to occur (Oppenheim, 1992). Among the analysis used were frequency, mean and cross tabulation. The SPSS Version 14.0 software was used for this purpose.

4.7 Findings

This section aims to report the findings of the exploratory study. They are classified into eight (8) sub sections, which are 1) demographic information, 2) current practices of Agile software process, 3) current practices of secure software process, 4) Agile software practices that influence the quality of software, 5) secure software practices that influence the quality of software 6) perceptions on the importance of Agile and secure software processes in producing high quality software, 7) characteristics of people who involve in Agile and secure software processes and 8) current practices of software certification.

4.7.1 Demographic Information

This section is aimed for assessing the background of the respondents and the organization.

4.7.1.1 Respondents' Background

To understand the respondents' background, they were asked to indicate their position in the company and years of experience in software development. Table 4.2 portrays the frequency and percentages of respondents according to their positions. Majority of the respondents are programmers (40%), followed by system analysts (26%), project managers (13%) and team leaders (13%). The rest of them are the quality assurance/testers (7%) and security advisors (1%).

Table 4.2

Respondents' Position in Company

Positions	Frequency	Percentage
Programmers	45	40
System Analysts	30	26
Project Managers	15	13
Team Leaders	15	13
Quality Assurance/Testers	8	7
Security Advisors	1	1
Total	114	100

Cross tabulation analysis was used in order to classify the respondents based on their experience and position, as depicted in Table 4.3. Out of the 114 respondents, only 20 have experience more than 10 years. Most of the respondents (53 of them) have 1 to 5 years experience and among them, 28 are programmers.

Table 4.3

Respondents' Experience

Positions	<1 year	1-5 years	6-10 years	11-20 years	Total
Project Managers	2	4	4	5	15
Programmers	9	28	5	3	45
Quality Assurance/Testers	1	6	0	1	8
System Analysts	2	11	12	5	30
Security Advisors	1	0	0	0	1
Team Leaders	1	4	4	6	15
Total	16	53	25	20	114

4.7.1.2 Organizational Background

Table 4.4 lists the sectors of organization of the respondents. Software development sector is found to be in the highest ranking (42.1%). The ranking continues with education/training (22.8%), service and public administration (10.5%) and manufacturing (6.1%). Other sectors include consultation, telecommunication, health and social work, banking/financial/insurance and agriculture, hunting and forestry.

Table 4.4

Sectors of Organization

Organization's Sectors	Frequency	Percentage
Software Development	48	42.1
Education/Training	26	22.8
Service and Public Administration	12	10.5
Manufacturing	7	6.1
Consultation	6	5.3
Telecommunication	7	6.1
Health & Social Work	5	4.4
Banking/Financial/Insurance	2	1.8
Agriculture, Hunting & Forestry	1	0.9
Total	114	100

4.7.2 Current Practices of Agile Software Process

This section addresses the software practitioners' opinion and experiences regarding Agile software process by describing their familiarity, level of exposure, years of experience, the number Agile team members in their teams, Agile methods that they are familiar with, the benefits of Agile and the implementation of Agile principles.

4.7.2.1 Software Practitioners' Familiarity of Agile

To ensure the validity of the data, the respondents were asked whether they are familiar with the Agile. Majority of them (64%) claimed that they are familiar with Agile, whilst only 36% are not familiar.

4.7.2.2 Level of Exposure to Agile

Furthermore, they were asked about their experience in implementing Agile. 64% among the respondents are either current member of Agile (19%) or currently leading Agile team (12%) or previously were in Agile team (11%) or Agile coach (6%) or have heard about it, but not in depth (16%), while 36% of them have never heard about Agile. Figure 4.1 exhibits the analysis result. Therefore, the subsequent questions regarding Agile were answered by the respondents who have knowledge in it (73 of them), while the others continued answering the questions about the secure software process and software certification.

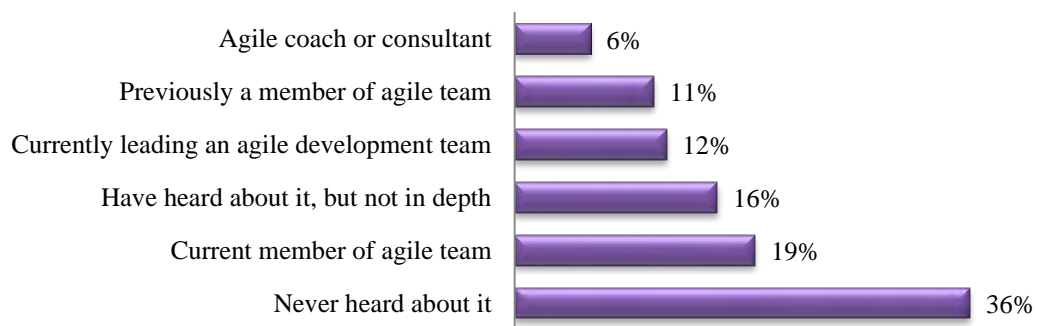


Figure 4.1. Level of exposure to Agile

4.7.2.3 Years of Experience Implementing Agile

The respondents were then asked about their years of experience in implementing Agile. Majority of them (78%) have experience in it for two years or less' duration. Only 7% have experience in it for more than five years while 15% have experience for three (3) to five (5) years (Refer to Figure 4.2).

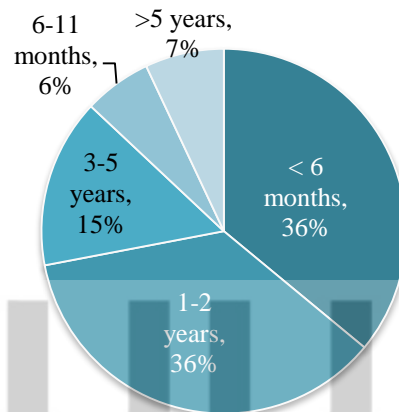


Figure 4.2. Years of experience

4.7.2.4 Number of Agile Team Members

The respondents were then asked about the number of team members in their team. Most of them work in a team with less than five members (37%) or five to ten members (34%), as shown in Figure 4.3.

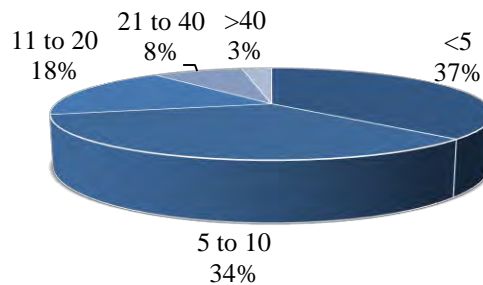


Figure 4.3. Number of Agile team members

4.7.2.5 Agile Methods

Figure 4.4 shows the Agile methods that the respondents are familiar with. They were allowed to choose more than one answer for this question. Most of them are familiar with Extreme Programming (XP) (52%), followed by Scrum (32%). The rest are familiar with DSDM, Crystal Family, FDD and Agile Modeling.

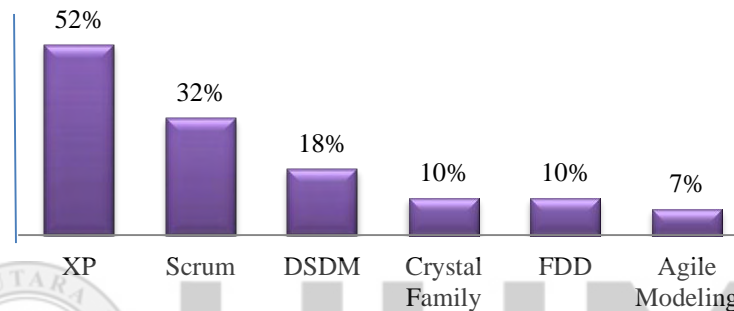


Figure 4.4. Agile methods being practiced

4.7.2.6 Benefits of Agile

Additionally, the respondents were asked about the benefits that they gain by practicing Agile. This question allowed multiple answers. It is apparent from Figure 4.5 that most of the respondents agree that Agile can enhance the ability to manage changing requirements (59%), increase productivity (55%) and accelerate time-to-market (48%).

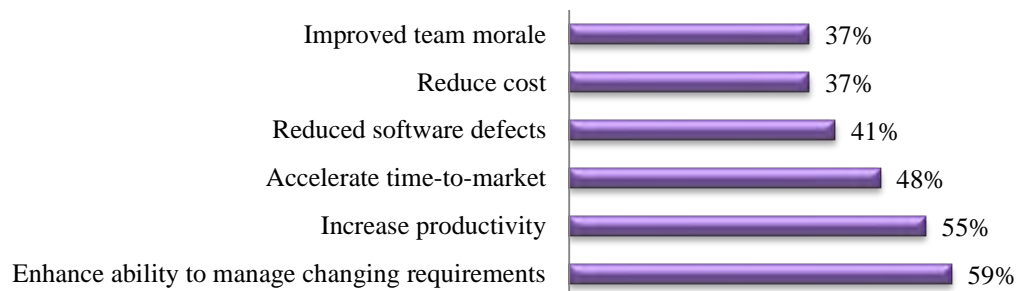


Figure 4.5. Benefits of Agile practices

4.7.2.7 Implementation of Agile Principles


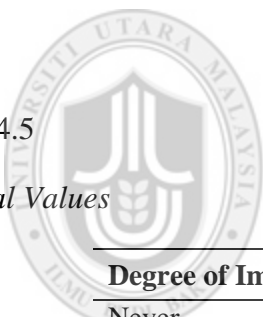
The respondents were then asked about the frequency of the Agile principles being practiced in their organizations. The 7-point numerical scale (Zikmund et al., 2010) was used for this question, which ranged from *Never* to *Every time*. This scale was then mapped to equal intervals. The interval ranges are calculated by using the following formula (Ismail, Abedlazeed, & Hussin, 2011):

$$\text{Interval ranges} = (n-1) / n \quad (4.1)$$

Where n is the maximum number in the used scale, which is to equal 7. Thus, the interval size of the consideration level between one through seven is 0.86, as depicted in Table 4.5.

Table 4.5

Interval Values



Degree of Importance (DI)	Interval Values
Never	1.00 – 1.86
Rarely	1.87 – 2.72
Occasionally	2.73 – 3.58
Sometimes	3.59 – 4.44
Frequently	4.45 - 5.30
Usually	5.31 - 6.16
Every time	6.17 - 7.00

As can be seen in Table 4.6, the results demonstrate that none of the Agile principles was used ‘*Every time*’ by the respondents. Nevertheless, majority of the principles were performed frequently. Only principles 1 to 5 were performed ‘*Usually*’.

Table 4.6

Agile Principles Implementation

Agile Principles	Mean	DI
1) Satisfy the customer through early and continuous delivery of valuable software	5.82	Usually
2) Emphasize on face-to-face conversation for conveying information to and within a development team	5.67	
3) Emphasize on simplicity throughout the development process (estimation, design, coding, etc)	5.34	
4) At regular intervals, the team reflects on how to become more effective in future iterations/sprints	5.33	
5) Continuous attention is given to technical excellence and good design	5.32	
6) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale	5.30	Frequently
7) Working software is the primary measure of progress	5.14	
8) The sponsors, developers, and customers maintain a sustainable development	5.12	
9) The projects are built around motivated individuals	5.08	
10) Customers work closely with the Agile team and are readily available	5.01	
11) Welcome changing requirements, even late in development	4.88	
12) Self-organized teams (<i>team members make their decisions and plans without depending on managers</i>)	4.45	

4.7.3 Current Practices of Secure Software Process

This section investigates the current practices of software practitioners regarding secure software process in terms of their familiarity with it, how they prevent from common attacks, the security trainings provided for them, the notations that they use to represent the security requirements, how they elicit security requirements and the security incidents that they faced.

4.7.3.1 Software Practitioners' Familiarity of Secure Software Process

From the 114 respondents, majority of the respondents (82%) have knowledge about the secure software process (93 of them). Consequently, the questions regarding the secure software process were only answered by the ones who have prior knowledge and experience on secure software process.

4.7.3.2 Common Attacks Prevention Technique

The respondents were asked about how they prevent from introducing common attacks that occurred previously. Surprisingly, majority of them do not consider the attacks that have happened in the past (41%). However, fortunately the remaining respondents refer to the document which records the security attacks that have occurred previously (37%), while 35% of them consult with the security experts. On the other hand, 34% of them look for well-known common security attacks and vulnerability databases. Table 4.7 shows the analysis result.

Table 4.7

Prevention Techniques from Common Attacks

Prevention Techniques	Frequency	Percentage
Do not consider attacks that have happened in the past	38	41
Refer to document which records the security attacks that have occurred previously	34	37
Consult with security experts to prevent common attacks	33	35
Look for well-known common security attacks in attack and vulnerability databases	32	34

4.7.3.3 Security Trainings

The respondents were then asked about the percentage of security trainings provided for the staff. Cross tabulation analysis was used in order to classify the respondents based on their position and amount of security training provided for them. Most of the respondents (38.7%) are provided with 25% or less security trainings in a year. Quite a big percentage is not provided with any security trainings (19.4%). Only 24.7% are provided with security trainings within 25 to 50 percent in a year. The result of analysis is depicted in Table 4.8. Meanwhile, the trainings are provided mostly for the programmers and system analysts, 41.9% and 28% respectively, while majority of the project managers who participated in this study are provided with security trainings yearly.

Table 4.8
Percentages of Security Training Provided

Positions	Percentages of Trainings per Year					Total
	None	25% or less	25% but less than 50%	50% but less than 75%	Greater than 75%	
Project Manager	1 (1.1%)	3 (3.2%)	3 (3.2%)	2 (2.2%)	1 (1.1%)	10 (10.8%)
Programmer	7 (7.5%)	14 (15.1%)	11 (11.8%)	2 (2.2%)	5 (5.4%)	39 (41.9%)
Quality Assurance/Tester	2 (2.2%)	3 (3.2%)	1 (1.1%)	0 (0%)	0 (0%)	6 (6.5%)
System Analyst	7 (7.5%)	10 (10.8%)	5 (5.4%)	2 (2.2%)	2 (2.2%)	26 (28%)
Security Advisor	0 (0%)	0 (0%)	1 (1.1%)	0 (0%)	0 (0%)	1 (1.1%)
Team leader	1 (1.1%)	6 (6.5%)	2 (2.2%)	0 (0%)	2 (2.2%)	11 (11.8%)
Total	18 (19.4%)	36 (38.7%)	23 (24.7%)	6 (6.5%)	10 (10.8%)	93 (100%)

4.7.3.4 Notations for Security Requirements

Table 4.9 represents the analysis result regarding the notations used to represent security requirements. Multiple answers were allowed for this question. Unfortunately, the analysis result found that majority of them (76%) do not document the security requirements, while 4% do not use any specific notation to represent the security requirements. Only 11% use abuse case, 10% use misuse case, 8% use attack tree, and 2% use misuser stories.

Table 4.9

Notations for Security Requirements

Notations	Frequency	Percentage
Do not document	71	76
Abuse case	10	11
Misuse case	9	10
Attack tree	7	8
No specific notation	4	4
Misuser stories	2	2

4.7.3.5 Security Requirement Elicitation Practice

The respondents were asked whether they elicit and document security requirements explicitly from early stage. 21.5% of the respondents discuss about the security requirement from early stage. Unfortunately, the requirements are not documented. However, 24% of them are aware of this, whereby they gather and document the security requirements explicitly during requirement gathering. Meanwhile, 32% of the respondents only deal with security issues during the implementation phase or after the

system being developed. On top of that, 22.5% do not even deal with the security requirements. Table 4.10 presents the analysis result.

Table 4.10

Eliciting Security Requirements Explicitly during Requirement Gathering

Answers	Frequency	Percentage
Security issues are only dealt during the implementation phase or after the system being developed	30	32
Security requirements are gathered and documented in the early stages of the projects before the development starts	22	24
Do not deal with security requirements	21	22.5
Security requirements are discussed from early stages but are not documented	20	21.5
Total	93	100

4.7.3.6 Security Incidents Faced

The respondents were asked about the security incidents that they faced. They were allowed to provide multiple answers for this question. The analysis result found that respondents face many security incidents, as depicted in Figure 4.6. The most common security incidents faced by them are password cracking (45%), followed by malicious code (39%) and SQL injection (35%). Other security incidents are spamming, denial of service, eavesdropping, spoofing. Only small percentage (9%) of them never face any security incidents.

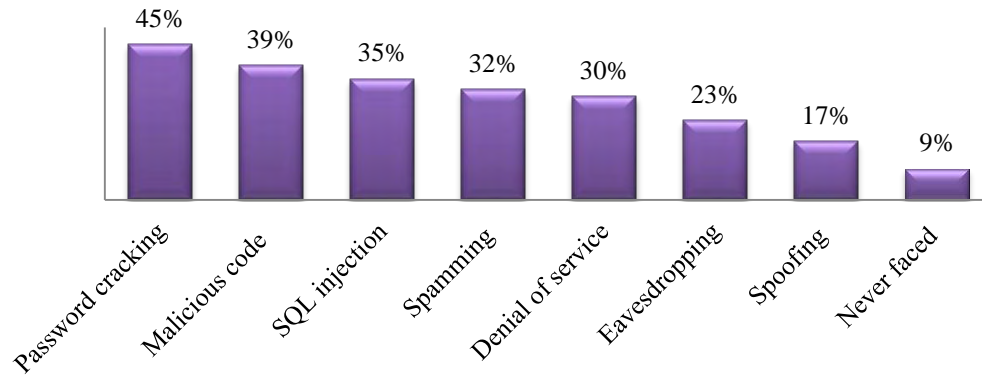


Figure 4.6. The security incidents faced

4.7.4 Agile Software Practices that Influence the Quality of Software

The respondents were further inquired about the Agile software practices that need to be performed in order to produce high quality software. They are categorized into requirement engineering, software design, coding, testing, project management and change management. The mean value for each practice is obtained from the analysis, as it represents the most selected answers in average. The 7-point numerical scale was used for this question, which ranges from *Extremely Not Important* to *Extremely Important*. The scale is then mapped to equal intervals, by using Equation 4.1. The interval values are depicted in Table 4.11.

Table 4.11

Interval Values

Degree of Importance (DI)	Interval Values
Extremely Not Important (ENI)	1.00 – 1.86
Not important (NI)	1.87 – 2.72
Less Important (LI)	2.73 – 3.58
Moderately Important (MI)	3.59 – 4.44
Important (I)	4.45 – 5.30
Very Important (VI)	5.31 – 6.16
Extremely Important (EI)	6.17 – 7.00

Table 4.12 exhibits the mean values obtained by the important practices in each phase of the Agile software process. Outcomes from the study show that mostly these Agile software processes obtained high consideration among the respondents, whereby the mean values are in the range of *Important to Very Important*.

Table 4.12

The Mean Values for Agile Software Practices

Phases	Practices	Mean	DI
Requirement Engineering	Identifying the scope at the beginning of the project to create initial prioritized stack of requirements	5.58	(VI)
	Gathering requirements iteratively and incrementally	5.51	
	Emphasizing on face-to-face communication	5.51	
	Producing product backlog and iteration backlog for ensuring the consistency and traceability of requirements	5.42	
	Emphasizing on single source information	5.30	(I)
	Using releases (working software) for validating requirements at the end of each iterations	5.23	
	The requirements are written on cards in short statements	5.11	
	Enabling development team to re-estimate the time and velocity (speed of accomplishing tasks) of user stories	5.03	
	Enabling customers to prioritize and reprioritize requirements throughout the development	4.42	
Software Design	Implementing model storming	5.36	(VI)
	Creating an initial model at the beginning of iteration	5.27	(I)
	Start designing with simple initial design and integrating it continuously	5.21	
	Producing just barely good enough artifacts (for situation at hand only)	5.07	

Phases	Practices	Mean	DI
	Refactoring (reorganize) the design	4.85	(I)
	Using metaphor as architecture of the system	4.64	
Coding	Delivering the features with high priority first	5.59	(VI)
	Following coding/database/interface standards	5.47	
	Delivering the software frequently with increments of features	5.42	
	Deploying the software gradually in real environment	5.42	
	Having customers on-site to get continuous and immediate feedback from customer for clarification	5.34	
	Integrating the newly produced code to system baseline frequently	5.14	(I)
	Determining code integration strategy and revising it	5.05	
	Implementing test driven development (TDD): write tests first, then write the code to pass the tests	5.01	
	Producing deliverable documentation late	5.01	
	Refactoring the code and database	4.85	
Implementing pair programming (two programmers working together)	4.79		
Giving authority to team members to make changes at any part of the code	4.77		
Testing	Implementing user interface testing	5.77	(VI)
	Using acceptance tests to validate and verify user's requirements	5.64	
	Implementing database regression testing	5.51	
	Producing executable specification	5.41	
	Acceptance tests are written or at least modeled by customers	5.26	(I)
	Implementing automated tests	5.23	

Phases	Practices	Mean	DI
	Implementing frequent integration testing	5.23	(I)
	Implementing tests continuously throughout the development	5.18	
Project Management	Performing project planning jointly and continuously with team members	5.41	(VI)
	Conducting continuous review meetings at end of each iteration to demonstrate the latest version of software	5.32	
	Planning and estimating (cost and schedule) are based on features/ functions/stories of system	5.23	(I)
	Revealing the current progress of iteration/sprint to everyone on sprint burn down chart	5.19	
	Carrying out release meeting at the beginning of project to plan releases	5.18	
	Ensuring that working hours do not exceed 40 hour per week (no overtime)	5.14	
	Carrying out iteration meeting at the beginning of each iteration to plan iterations	5.11	
	Conducting retrospective (postmortem) at end of each iteration to look back what worked well and what need to be improved	4.81	
	Monitoring customer involvement and end-user in project activity	4.71	
	Carrying out daily stand-up meetings for daily plan	4.68	
Change Management	Controlling changes using product backlog (prioritized user stories)	5.40	(VI)
	Assigning the individual who will be responsible for ensuring Change Management activities are implemented correctly	5.12	(I)
	Automating the Change Management activities (e.g. building scripts)	4.99	
	Not allowing changes once an iteration has begin, until the iteration ends	4.93	

4.7.5 Secure Software Practices that Influence the Quality of Software

Additionally, the respondents were asked about the secure software practices needed in order to produce high quality software, which are categorized into requirement engineering, software design, coding, testing, security management and risk management. Similar to Agile software practices, the 7-point numerical scale was used for this question and mapped to equal intervals, as shown in Table 4.11. Results were obtained by calculating the mean score gained by each secure software process, as depicted in Table 4.13. It shows that mostly the secure software processes obtained high consideration, whereby the mean values are in the range of *Important* to *Very Important*.

Table 4.13

The Mean Values for Secure Software Practices

Phases	Practices	Mean	DI
Requirement Engineering	Updating security requirements iteratively, taking place as changes occur	5.51	(VI)
	Documenting and maintaining a set of well-defined security requirements to prevent from introducing common attacks that occurred previously	5.47	
	Obtaining security requirements explicitly	5.41	
	Considering attackers' perspective while eliciting security requirements	5.39	
Software Design	Referring the latest lists of common attack patterns, vulnerabilities and threats in order to keep up-to-date with current trends	5.52	(VI)
	Documenting security requirements in a particular notation (e.g.: misuse case, attack tree)	5.33	
	Modeling the possible threats	5.23	(I)
	Performing an external (by someone outside the design team)	5.13	

Coding	Referring to the secure coding guidelines	5.40	(VI)
	Coding countermeasures for the identified threats	5.35	
	Preparing documents for installing and operating the application securely	5.32	
	Implementing pair programming to reduce vulnerability: continuous review	4.81	(I)
	Comparing outcome from automated and manual code review	4.72	
Testing	Performing integration tests focusing on the threats and vulnerabilities	5.31	(VI)
	Creating unit tests by focusing on identified threats and vulnerabilities	5.26	(I)
	Performing risk analysis again at the end of the phase to ensure all risks are mitigated and to consider remaining risks	5.23	
	Performing penetration test (simulate attack from malicious outsiders)	5.23	
	Performing fuzz testing (use random data as input for tests)	5.17	
Security Management	Sharing the produced artifacts among team members	5.63	(VI)
	Producing and revising security policy regularly	5.59	
	Ensuring that all members of the project team are aware of and involved with security engineering activities	5.43	
	Planning and documenting security plan	5.42	
	Defining the security roles and responsibilities up-front	5.34	
Risk Management	Performing risk analysis iteratively throughout the software development to identify the possible threats, vulnerabilities and impacts of application	5.35	(VI)
	Planning mitigation strategy to countermeasure the identified threats, vulnerabilities and impacts	5.32	
	Ensuring the newly identified risks are reported and mitigated as soon as possible	5.24	(I)
	Monitoring the identified threats, vulnerabilities and impacts throughout the development	5.18	

4.7.6 Perceptions on The Importance of Agile and Secure Software Processes in Producing High Quality Software.

The respondents who have knowledge about Agile were asked whether they agree that Agile can influence the quality of produced software. 96% of them answered 'Yes', while only 4% answered 'No'. Furthermore, the respondents who have prior knowledge on secure software process were asked whether they agree that it is important to consider secure software process and its implementation from early phases of software development in order to ensure software quality. Akin to the Agile software process, 96% of them answered 'Yes', while only 4% answered 'No'.

4.7.7 Characteristics of People Who Involve in Agile and Secure Software Processes

In this section, respondents were asked about the characteristics that should exist among the team and organization in order to successfully develop high quality software which concerns on the Agile and secure software processes. These characteristics are presented below. The 7-point numerical scale was used for this question, which ranged from *Extremely Not Important* to *Extremely Important*. The scale was then mapped to equal intervals, as presented in Table 4.11. Results are obtained by calculating the mean score gained by each characteristic, as depicted in Table 4.14. They obtained high consideration among the respondents, whereby the mean values are between *Very Important* and *Important*.

Table 4.14

Team and Organizational Characteristics

Types	Characteristics	Mean	DI
Teams	Emphasize on face-to-face communication	5.97	(VI)
	The team members consists of people with different functional expertise	5.96	
	Every team member adopts the security activities	5.71	
	Every team member being familiar with the security requirement of the system	5.70	
	Reach a common understanding about the security needs among all applicable parties, including the customer.	5.60	
	Small sized team	5.60	
	Co-located team	5.23	(I)
	Self-organized team	5.14	
Organization	Encourage customer participation and face-to-face communication	5.96	(VI)
	Provide basic security knowledge training for staff	5.80	
	Provide cooperative organizational culture instead of hierarchical	5.86	
	Provide facilities with proper Agile-style work environment	5.73	
	Ensure that Agile way of software development is universally accepted	5.71	
	Give rewards for successful security handling	5.57	
	Provide sufficient budget for security initiative	5.35	
	Provide a separate security team to engineer and evaluate the security of software	5.16	(I)

4.7.8 Current Practices of Software Certification

This section presents the software practitioners' opinion on the importance of software certification, as well as the internal assessment or audit that they performed, besides the assessment techniques that they used. Additionally, they were also asked about the use of standards in their organizations.

4.7.8.1 Software Practitioners' Opinion on the Importance of Software Certification

In this section, the respondents were asked whether they agree that certifying software process is necessary to improve and enhance the quality of software. Majority of them (86%) answered 'Yes', while only 14% answered 'No'. When further investigated through cross tabulation analysis, the results reveal that majority of the respondents who are familiar with Agile and secure software processes agree on the importance of software certification, 56% and 74% respectively. On the other hand, only small percentage from the respondents did not agree, same percentage (8%) from the respondents who are familiar with Agile and secure software processes.

In the meantime, the respondents who are not familiar with Agile and secure software processes also agreed on software certification's importance, 31% and 13% correspondingly. Additionally, only small percent of them did not agree on this, 5% of respondents who are not familiar with Agile and secure software processes. Table 4.15 provides the analysis results.

Table 4.15

The Importance of Software Certification Based on Respondents' Familiarity

Familiarity		Software Certification Importance		Total
		Yes	No	
Agile software process	Yes	64 (56%)	9 (8%)	73 (64%)
	No	35 (31%)	6 (5%)	41 (36%)
Secure software process	Yes	84 (74%)	9 (8%)	93 (82%)
	No	15 (13%)	6 (5%)	21 (18%)

4.7.8.2 The Implementation of Internal Assessment/Audit and the Techniques Used

Most of the respondents' companies did not implement internal assessment/audit on the software process (53%), while only 37% implemented it. The respondents who implement the assessment/audit provide the techniques used for the assessment. Table 4.16 shows the analysis result. Most of them use document review (35%), followed by observation (28%) and interview (21%). They are allowed to choose more than one answer for this question.

Table 4.16

Assessment Techniques

Assessment Techniques	Frequency	Percentage
Document review	41	35
Observation	33	28
Interview	24	21

4.7.8.3 The Use of Standards

The respondents were asked about the standards that their organization implementing. Unpredictably, almost half of them did not implement any standard (48%). Only 22% use ISO/IEC 9000, 15% use IEEE Standards, 5% use ISO/ IEC 27001 and 2% use CMMI, while the rest 8% use other standard. The analysis result is shown in Figure 4.7.

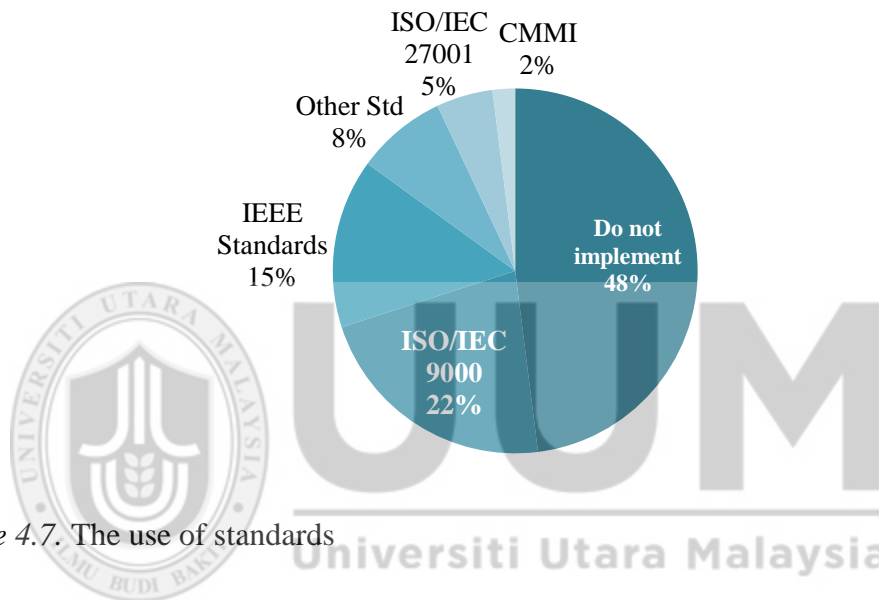


Figure 4.7. The use of standards

The above result is further detailed out by performing cross tabulation analysis to classify the use of standards based on the respondents' familiarity with the Agile and secure software processes. The analysis results are shown in Table 4.17.

Table 4.17

The Use of Standards Based on Respondents' Experience

Familiarity		Standards use						Total
		Do not implement	IEEE Standards	ISO/IEC 9000	ISO/IEC 27001	CMMI	Other International Standards	
Agile software process	Yes	33 (28.9%)	11 (9.7%)	20 (17.5%)	2 (1.8%)	2 (1.8%)	5 (4.4%)	73 (64%)
	No	22 (19.3%)	6 (5.3%)	5 (4.4%)	4 (3.5%)	0 (0%)	4 (3.5%)	41 (36%)
Secure software process	Yes	39 (34%)	17 (15%)	22 (19.3%)	5 (4.4%)	2 (1.8%)	8 (7.1%)	93 (81.6%)
	No	16 (14%)	0 (0%)	3 (2.6%)	1 (0.9%)	0 (0%)	1 (0.9%)	21 (18.4%)

It is apparent from the table that the most of the respondents who are familiar with Agile software process do not implement any standards (28.9%), while 17.5% implement ISO/IEC 9000 and 9.7% apply IEEE Standards. Meanwhile, CMMI and ISO/IEC 27001 are only implemented by 1.8% each. Also, 4.4% of them execute other International Standards. Similarly, most of the respondents who are familiar with secure software process do not implement any standards (34%). However, there are among them who implement ISO/IEC 9000 and IEEE Standards, 19.3% and 15% respectively, while 7.1% apply other International Standards, ISO/IEC 27001 (4.4%) and CMMI (1.8%).

Additionally, the respondents who are not familiar with Agile do not implement any standard (19.3%). Additionally, 5.3% of them implement IEEE Standards while 4.4% use ISO/IEC 9000. Furthermore, the same percentages of the respondents (3.5% each) utilize the ISO/IEC 27001 and other International Standards. On the other hand, 14% of the respondents who are not familiar with secure software process do not implement any standard, while 2.6% implement ISO/IEC 9000. Additionally, only small percent of them (0.9%) use ISO/IEC 27001 and other International Standards (0.9%).

4.8 Discussions

Generally, this study has investigated several issues regarding software certification which relates to Agile and secure processes. The findings of the study are discussed in the next sub sections according to the exploratory study's objectives.

Objective 1: To study the software practitioners' current practices of the Agile and secure software processes

- **Exposure on Agile software process**

Most of the respondents have 6 months' to 2 years' experience (Refer Figure 4.2). Additionally, there are among the respondents who have experienced using Agile software process for more than five years (7%). This result is opposed to the outcome from the study by Fauziah et al. (2005) ten years back, by which majority of the respondents were implementing the 'Waterfall' model during software development. This shows that Agile software process is gradually being implemented in the Malaysian software industry. This outcome is consistent with the study by Ani Liza et al. (2012b).

However, there still exist among the respondents who never heard about it (Refer Figure 4.1). The study by Ani Liza et al. (2011) supports this. It indicates that even though Agile software process is gradually being implemented among Malaysian software practitioners, there still exist among them who are not aware of this important software process.

- **The number of Agile team members**

Furthermore, most of the respondents work in smaller teams, with less than 10 people in a team (71%) (Refer Figure 4.3). This is consistent with the literature whereby smaller teams are more effective in producing high quality software. Having a larger team might pose great obstacles to fast communication and decision making in projects. This is essential for Agile as it involves frequent communication. Having more people will make communication tougher (Abbas et al., 2010; Lindvall et al., 2002).

- **Agile methods being implemented**

Most of the respondents are familiar with Extreme Programming (XP) (52%), followed by Scrum (32%) (Refer Figure 4.4). This result is supported by the findings from the study conducted by VersionOne (2011) and Abrahamsson et al. (2010). One of the main reasons is that XP and Scrum complement each other, since Scrum focuses on project management, while XP focuses on project development (Fernandes & Almeida, 2010). AM is the least implemented method, although it provides effective way of modeling and documenting in the Agile software process, as claimed by Ambler (2014).

- **Implementation of Agile principles**

Additionally, in implementing Agile, its basic principles should be followed. There are 12 principles common to all Agile methodologies (Agile manifesto, 2001). There are only five (5) Agile principles which achieved *Usually*. The least implemented is self-organizing team (Refer Table 4.6). These principles are essential to be considered by the software practitioners as they are the backbone of the Agile software process. By emphasizing the principles, the proper Agile values can be delivered (Williams, 2012). If

they are violated, it means that the Agile software process is not being implemented properly. However, the respondents did not follow the Agile principles constantly, since none of the principles was implemented as *Every time*.

- **Common attack prevention technique**

Majority of the respondents referred to the documents which record the previous attacks occurred, which is aligned with the findings from the study of Elahi et al. (2011). They also consulted security experts and looked for the common attacks from the attack and vulnerability database. However, almost half of the respondents did not make any security references while eliciting security requirements (Refer Table 4.7). This might cause the software practitioners to be outdated from the current threats, attacks and countermeasure available in the industry, as well as repeating the same threats which occurred in previous projects. Consequently, they faced a lot of security attacks, as discussed before.

- **The security trainings provided for the staff**

Trainings have been accepted as one of the major ways to create awareness on the security issues among the software practitioners. Less security trainings are provided for the respondents, whereby majority of them attended security trainings only for the percentage of 25% or less (Refer Table 4.8). On top of that, there still exist among them who did not receive any security trainings. This result is contradict with the findings in the study of Elahi et al. (2011), whereby majority of their respondents attended security trainings. Without attending proper trainings may lead to improper implementation of secure software process, since proper guideline on its actual implementation is not

received. Moreover, the security trainings were provided for majority of the project managers for at least the percentage of 25, which is essential. By providing trainings for them, they can understand the importance of secure software process and problems that might arise by neglecting it (Geer, 2010). Consequently, they will enforce the software development team to include secure software process during software development.

- **The notation used for representing security requirements**

Representing the security requirements in particular notation is vital in order to get good understanding about the requirement of proposed system. Yet, majority of the respondents do not even document the security requirements (Refer Table 4.9). In contrast, Elahi et al. (2011) indicated that their respondents used modeling notations widely. By neglecting this important software process, the software practitioners might ignore relevant threats that might surface in the proposed system. This is because by explicitly presenting the security requirements, analysts may get ideas about possible threats that not thought before (Sindre & Opdahl, 2001). Fortunately, there exist among them who use abuse case, use misuse case, attack tree, misuse stories, while some do not follow any specific notations.

- **Security incidents and consideration of security requirements from early stage**

Although the respondents faced many security incidents such as password cracking, malicious code attacks and SQL injection (Refer Figure 4.6), most of them did not consider security requirements from the early stage of software development (Refer Table 4.10). They only dealt with security requirements during the implementation phase or after the system being developed. This result is aligned with the outcomes of Elahi et

al. (2011), whereby most of their respondents left the security requirements undocumented and only consider them implicitly. However, according to Shafiq et al., (2011), incorporating security in later stages of software development as an afterthought will increase the risks of introducing security vulnerabilities into software. On the other hand, the outcome of Errata Security survey (Geer, 2010) found that half of the respondents gave high concern on security during software development.

Furthermore, 21.5% of the respondents discuss the security requirement from early stages. Yet, they do not document the requirements. Fortunately, 24% of the respondents gather and document the security requirements from early stage. This explains that some of the respondents are aware of the importance of security activities during software development, however some are not aware on this important software process.

Objective 2: To investigate the software practitioners' opinion on the Agile and secure software processes that are important in producing high quality software

- **Requirement engineering for Agile**

Following the Agile principles, Agile requirement engineering is performed iteratively and incrementally, in contrast with the conventional software development approach which emphasizes completed and well-defined requirements up-front (Wells, 2013; CMMI Product Team, 2010; Liu et al., 2010; Lan & Ramesh, 2008). In this way, the requirements evolve over time throughout the development. Furthermore, Agile give importance on face-to-face communication during the requirement elicitation, with minimal documentation. These software practices are identified as *Very Important* by the

respondents of this study that aligned with the previous studies by Liu et al. (2010), Williams et al. (2010), Ramesh et al. (2010), as well as Lan and Ramesh (2008). Furthermore, towards ensuring the consistency and traceability of requirements, the uses of product and iteration backlog have been agreed by the respondents as *Very Important*. Similar result is reported in Salo and Abrahamsson (2008). Additionally, the scope is identified at the beginning of the project to create initial prioritized stack of requirements, as emphasized by Ambler (2014) and performed by O'Sheedy and Sankaran (2013) in their studies.

Moreover, the requirements are gathered with little detail in the beginning of the project and detailed up during iterations through discussions and negotiations (Williams et al., 2010). In order to verify the requirements and show the progress to customers after completing each iteration, the working software (releases) are demonstrated to the customers. These software practices have been rated as important Agile requirement engineering practices for ensuring software quality by the respondents of this study. Similar results were obtained in the studies of Liu et al. (2010), Ramesh et al. (2010) and Lan and Ramesh (2008). However, even though the studies of Liu et al. (2010), Ramesh et al. (2010) and Lan and Ramesh (2008) found that their respondents appreciated when the customers continuously prioritized the requirements, the result of this study is contradict. The result obtained is *Moderately Important*.

In addition, emphasizing on the single source information also reduces the maintenance and traceability burden, as well as increases the consistency (Ambler, 2014). The documentations produced are minimized by documenting repeating information only

once, such as the business rules. This single source of information can be a reference in producing other documents, rather than repeating them again and again (Ambler, 2014). However, the importance of this software practice has not been studied previously.

- **Software design for Agile**

The Agile software process emphasizes on simple initial design which continuously evolve over time, in contrast with the traditional approach which design everything upfront. Simple design is one of the Agile software process's success factors concluded by Rumpe and Schroder (2002), as well as Tsun and Dac-Buu (2008), Sison and Yang (2007) and Tessem (2003) in their studies. Similarly, it was rated as *Important* in this study. Designing in simple way can be accomplished by producing just barely good enough artifacts/documents. This means to produce documentation for the situation in-hand only, rather than documenting the whole project. This is done by modeling and documenting during the iterations. The iteration modeling is implemented during each of the iteration planning meetings, whereby the requirements in a particular iteration are modeled. The detailed modeling is then implemented through model storming for the in-hand solution before the development. The issues that need to be resolved is identified and explored together in a small group. During this discussion, the models are sketched on the whiteboard or paper and made visible to everybody (Ambler, 2014). These practices have not been studied by previous studies. However, in this study, model storming attained *Very Important*, while iteration modeling and producing just barely good enough artifacts obtained *Important*.

Furthermore, refactoring is another important Agile software design practice. It is a valuable tool that can be used to improve the software design (Tsun and Dac-Buu, 2008; Moser et al., 2008; Fowler, 1999). This is also agreed by the respondents in this study. Besides, the importance of metaphor was revealed by the respondents in Begel and Nagappan (2008), as well as the result of this study. Conversely, studies by Rumpe and Schroder (2002) and West and Grant (2010) did not support this observation since this practice was least used.

- **Coding for Agile**

Similar to the requirement engineering and designing, coding in Agile is also implemented iteratively and incrementally. In addition, before starting the coding, all programmers have to agree upon a set of coding/database/interface standard that everybody will follow during development. This practice assists in giving better understanding on the code, improves communication and facilitates maintenance. Studies in VersionOne (2011), Williams et al. (2010), Salo and Abrahamsson (2008), Tsun and Dac-Buu (2008), Begel and Nagappan (2008), Sison and Yang (2007) and Rumpe and Schroder (2002) indicated that coding standard is a highly adopted practice among their respondents.

The next practice is about delivering software frequently with features increments. By doing so, the software can be demonstrated earlier to customers and enable them to review the software, identify defects and make adjustment for future requirements (Abrantes & Travassos, 2011). This practice has been considered as *Very Important* by the respondents in this study, as well as in the studies by Franca et al. (2010), Tsun and

Dac-Buu (2008), Sison and Yang (2007), Rumpe and Schroder (2002). This practice is closely related with deploying the software gradually in real environment, which gained high consideration in study by Williams and Erdogmus (2002). On contrary, in VersionOne's study (VersionOne, 2011), this practice was rated as low percentage. However, deploying the software gradually in the real environment reduces the risk of deploying all at once. Furthermore, the feedbacks can be obtained earlier (Agile Alliance, 2013).

In addition, having customer on-site facilitates in providing continuous and immediate feedback. It is one of the essential Agile principles and has high influence on the success of Agile (West & Grant, 2010; Misra et al., 2009; Alshayeb, 2009; Tsun & Dac-Buu, 2008; Nerur, Mahapatra, & Mangalaraj, 2005; Tessem, 2003). It denotes that customer should be a member in the development team so that the uncertainties can be cleared as soon as it occurs (Abrantes & Travassos, 2011). This is contradicting to the traditional software development approach, whereby the customers typically involved during the initial requirements elicitation. Only towards the end they will then give their feedback on the developed software (Nerur et al., 2005). Customer involvement has also been identified as an important practice in Asnawi et al. (2012b), Tsun and Dac-Buu (2008), Tessem (2003). On contrary, study by Rumpe and Schroder (2002) concluded that on-site customer was least implemented and hard to be performed.

Delivering features with high priority is one of the Agile principles. It can ensure that most of the important business values are to be delivered first (Paetsch et al., 2003). This Agile principle is highly considered among the respondents in this study. Moreover,

study by Franca et al. (2010) concludes that it is one of the factors that influence the success of the Agile implementation. All of the previously discussed coding practices obtained *Very Important* for this study. The practices which obtained *Important* are discussed next.

Agile emphasizes collective code ownership, whereby all programmers in a team are empowered to make any changes to any part of the code they are working on. This practice gained high consideration among the respondents in this study. Even previous studies by Williams et al. (2010), Salo and Abrahamsson (2008) and Rumpe and Schroder (2002) provide the same report. Moreover, pair programming is one of the most accepted and succeeded practices in the industry and academic. The most significant result is improvement in the quality of design and code, as reported by Sfetsos, Stamelos, Angelis and Deligiannis (2009), Begel and Nagappan (2008), Schindler (2008) and Canfora, Cimitile, Garcia, Piattini and Vissagio (2007). This practice encourages two programmers to work together when accomplishing their tasks which enable them to transfer their knowledge as well as to review the code permanently (Schindler, 2008). Studies by Williams et al. (2010), Begel & Nagappan (2008), Tessem (2003), Rumpe and Schroder (2002) found out that pair programming is among the top Agile practices. On the contrary, studies by VersionOne (2011) and Salo and Abrahamsson (2008) found that this practice is rated as the least practiced. Although the respondents of (Schindler, 2008) practiced pair programming, they did not use it regularly. They usually used it based on demand especially for complex code or debugging.

Test driven development (TDD) is a critical practice in producing high quality software (Sfetsos & Stamelos, 2010). The developers create the unit tests before writing the production code. Many studies have proven its ability to produce high quality software, such as Huang and Holcombe (2009), Gupta and Jalote (2007) and Desai, Janzen and Clements (2009). Meanwhile, Sanchez et al. (2007) stated that the complexity of code and design is reduced with this practice. Similarly, Desai et al. (2009) report the same in their study. Study by Williams et al. (2010), Nagappan et al. (2008), Sanchez et al. (2007), Ambler (2006), George and Williams (2004) signified that the TDD is an important practice, similar to this study. On the other hand, studies by West and Grant (2010), Begel & Nagappan (2008); Salo and Abrahamsson (2008) gave contradict reports.

Continuous integration of source code to the system baseline has been found as an important practice in VersionOne (2011), Williams et al. (2010), West and Grant (2010), Begel and Nagappan (2008), Salo and Abrahamsson (2008), Rumpe and Schroder (2002). By performing this practice, compatibility problems can be detected or avoided earlier (Wells, 2013). In fact, the definition and revision of the code integration strategy has been included in the CMMI Version 1.3 (CMMI Product Team, 2010), which indicates it as an important practice.

By practicing refactoring on code and database, the software will be easier to be understood, helps in finding bugs and performing program faster. The refactoring focuses on the internal code restructuring (attributes and methods) across existing classes, without changing its external behavior (Fowler, 1999). Moser and his co-researchers (2008)

pointed out that refactoring increases the software quality as well as improves the productivity. Additionally, this practice gained high consideration in study by Ambler (2006). On the other hand, Alshayeb (2009) indicated that refactoring does not influence the quality (adaptability, maintainability, understandability, reusability, and testability) of the developed software. As for this study, refactoring is highly considered.

As Agile involves frequent changes, the production of the deliverable documentations is deferred to the end of development. The documents are created just before delivering the software. In preparing documentation, this can be risky because the earlier details of the requirements or design might change (Ambler, 2014). However, the importance of this practice has not been studied in previous studies.

- **Testing for Agile**

The Agile testing practices attained either *Very Important* or *Important*, which indicates that the practices are important towards producing high quality software. Testing in the Agile environment is done continuously throughout the development, as reported by VersionOne (2011) and Liu et al. (2010). It involves unit, system integration, user interface, database regression, and user acceptance tests. This differs to the conventional approach which conducts testing after the implementation stage. Database regression testing attained high importance in the study of Ambler (2006), while user interface tests was emphasized in studies by VersionOne (2011) as well as Liu et al. (2010). Furthermore, the integration testing must be done frequently as performed by the respondents in Franca et al. (2010), Tsun and Dac-Buu (2008) and Rumpe and Schroder (2002).

Moreover, the user acceptance tests are written by the customers to assure that the systems fulfill their needs, as reported in (Lan & Ramesh, 2008; Paetsch et al., 2003). In cases where the customers do not have technical knowledge, the developers will help the customers in writing the acceptance tests. Referring to the study by Lan and Ramesh (2008), the acceptance tests acts as a mechanism to validate and verify user's requirements. In addition, Agile also emphasizes on automating these tests (Wells, 2013). This practice gained high consideration by the respondents in this study, as well as studies by VersionOne (2011), Liu et al. (2010), Williams et al. (2010). The well-written tests act as executable specification. For instance, unit test is a portion of technical documentation and acceptance test is part of requirement documentation (Ambler, 2014). However, the importance of this practice has not been studied for the real world implementation.

- **Project management for Agile**

Project management in Agile is different than that in the conventional software development approach. The project management consists of three planning levels which obtained high consideration in previous studies; release plan, iteration plan and daily plan (Li et al., 2010; West & Grant, 2010; Salo & Abrahamsson, 2008; Sison & Yang (2007). These planning are done iteratively and collaboratively, rather than planning the whole project up-front, as reported in (Liu et al., 2010; Lan & Ramesh, 2008). A study by Tessem (2003) indicates that conducting these planning leads to better estimation of the work size. However, in Salo and Abrahamsson (2008), the collaborative planning was rated as low. Additionally, the Agile project management emphasizes on the sprint review and retrospectives which are held at the end of a sprint (Blankenship et al., 2011;

Lan & Ramesh, 2008). The retrospective was found to be more beneficial when applied to small teams, participated by the whole team and when the comments are recorded (Abbas et al., 2010). Similarly, Sison and Yang (2007) concluded that retrospective is important. Additionally, Sliger and Broderick (2008) explained that the planning must be done according to the features/requirements. However, the importance of this practice has not been studied before. Furthermore, the developers are able to re-estimate the time and velocity of accomplishing the requirements (Liu et al., 2010; Ramesh et al., 2010; Lan & Ramesh, 2008).

In addition, the progress of the team should be revealed in an open space so that everyone is aware of the current progress of the project. This practice gained high consideration in this study, as well as in the studies by VersionOne (2011), Williams et al. (2010) and West and Grant (2010). At the same time, the working hours should not exceed 40 hours in a week to ensure productivity. This practice obtained high consideration among the respondents not only in this study but also in Rumpe and Schroder (2002), Salo and Abrahamsson (2008), and Sison and Yang (2007) studies. Furthermore, the involvement of customers and end-users is monitored by the management, as Agile enforces their collaboration with the development team (CMMI Product Team, 2010; Sliger & Broderick, 2008).

- **Change management for Agile**

Since Agile involves a lot of frequent changes, the changes need to be adapted, rather than controlling them. Thus, change management and traceability is imperative (Jyothi & Rao, 2011). Furthermore, a particular individual who will be responsible in managing the

changes must be identified (CMMI Product Team, 2010). To enable the change management activities to be more efficient, the change management activities are automated. For example, the use of automated tool for scripts creation. In order to avoid scope crepe, the changes are controlled by monitoring the product backlog and by restricting changes once the iteration starts (Ambler, 2014; Blankenship et al., 2011). However, previous studies that studied about the importance of these practices are hardly found.

- **Requirement engineering for secure software process**

Eliciting security requirements explicitly, accurately and consistently has been one of the most fundamental activities for engineering secured software (McGraw, 2004; Wilander & Gustavsson, 2005; Karpati et al., 2011). However, security requirements are mostly dealt when the system has been designed or put in operation (Mellado, Blanco, Sanchez, & Fernandez-Medina, 2010; Christian, 2010). Only low percentage of respondents (9%) admitted that they document security requirements explicitly in study by Elahi et al. (2010), while majority of them (59%) considered it implicitly. On top of that, 31% do not elicit security requirements at all. Furthermore, most of the researchers (McGraw, 2006; Microsoft, 2012; Mead, 2010 OWASP, 2006) stress that security requirements should be established from an attacker's perspective and updated iteratively as soon as changes occur. In addition, the security requirements must be documented and maintained for reuse purpose (Christian, 2010; McGraw, 2006). By doing so, it will help developers to improve the software security as well as learn from past mistakes. In addition, the available guidelines, internal or external guidelines/ standards/ policies, or established compliance requirements shall be referred while gathering security requirements (Elahi et

al., 2011; OWASP, 2006) and the identified security requirements shall be validated with stakeholders (Microsoft, 2012; Christian, 2010; McGraw, 2006; OWASP, 2006). Majority of the respondents in this study expressed that all of these practices as important towards producing secured software.

- **Software Design for secure software process**

Designing security is similarly important as eliciting security requirements explicitly (Karpati et al., 2011). MS SDL (Microsoft, 2012) emphasizes on defining the attack surfaces. Furthermore, during this phase, the possible impacts, vulnerable and threats must be identified, classified, rated (Davis, 2013; Microsoft, 2012; McGraw, 2006; OWASP, 2006) and the countermeasures are documented (Microsoft, 2012; McGraw, 2006; OWASP, 2006). This activity will be more efficient by performing external review (Microsoft, 2012; McGraw, 2006) and referring to the latest list of common attack from online database (Julia et al., 2008; McGraw, 2006; OWASP, 2006).

- **Coding for secure software process**

During this phase, the secure coding guideline should be referred to (Microsoft, 2012; McGraw, 2006; OWASP, 2006). There are websites which gives this guideline such as Software Engineering Institute (SEI). The most important part in this phase is coding the countermeasure for the identified risks- threats, vulnerabilities and impacts (Microsoft, 2012; Evans et al., 2010; Ashbaugh 2009; McGraw, 2006; OWASP, 2006). Besides, these codes must be reviewed with automated tools as well as manual review and both results should be compared (McGraw, 2006; Merkow & Raghavan, 2010). Also, the security features provided by programming language used are identified (McGraw,

2006). In addition, pair programming is useful to reduce vulnerability- by having continuous review (Ashbaugh, 2009). Besides producing security emphasized coding, CLASP (OWASP, 2006) insists of producing document for installing and operating the application securely.

- **Testing for secure software process**

Testing for secure software process is different from traditional testing as it emphasizes what an application should not do rather than what it should do (Julia et al., 2008). Thus, testing in secure software process must include testing the security functionality besides the standard functional testing. They are the fuzz test and penetration test (Microsoft, 2012; McGraw, 2006). Traditional tests such as unit tests and integration tests are performed as well, but focused more on the threats and vulnerabilities (Julia et al., 2008). Consequently, the test cases created focuses on the identified mitigation strategies (Microsoft, 2012; McGraw, 2006; OWASP, 2006). Additionally, to ensure all risks are mitigated and to consider other residual risks, McGraw (2006) includes analyzing the risks again at the end of testing phase.

- **Security management**

In managing the security, the usage of security policy is very important to ensure that appropriate controls are put in place (Ai et al., 2007; Tondel et al., 2008; Colley, 2009; Syed Irfan, Abdulrahman, & Khaled, 2010). Besides, it should be reviewed and revised regularly (Syed Irfan et al., 2010) as well as ensuring that it is being properly followed by the workers (Syed Irfan et al., 2010). Additionally, security plan is another means of

ensuring good security management (Ai et al., 2007; Microsoft, 2012). Furthermore, CLASP (OWASP, 2006) defines security roles upfront for the team members.

- **Risk management for secure software process**

Risk management is the main activity in secure software process (Davis, 2013). Basically all of the traditional risk management activities exist in this approach: risk identification, risk analysis, risk planning and risk monitoring (Sommerville, 2007). However, their concern is more on the threat, vulnerabilities and impacts (Davis, 2013; Evans et al., 2010; McGraw, 2006; OWASP, 2006). These activities are implemented iteratively throughout the software development and ensuring the newly identified risks are reported and mitigated as soon as possible (Microsoft, 2012; Evans et al., 2010; McGraw, 2006; OWASP, 2006). The mitigation strategy is planned to countermeasure the identified threats, vulnerabilities and impacts (Evans et al., 2010). Also, these threats, vulnerabilities and impacts are monitored throughout the software development (Davis, 2013; Evans et al., 2010).

The Agile and secure software practices which obtained the value of 4.45 and above for the Degree of Importance (DI) are deemed as important practices that influence the quality of software. Consequently, they are considered as the reference standard in the proposed model.

Objective 3: To examine the software practitioners' opinion on the importance of the Agile and secure software processes in producing high quality software

Based on the literature, Agile software process is important and should be included in current software process as it ensures that high quality software could be marketed faster in most cost-effective manner (Pressman, 2010; Rico et al., 2009). This issue has been supported by majority of the respondents who have experience in implementing Agile software process (96%). They also agreed this approach can enhance the ability to manage changing requirements, increase productivity and accelerate time-to-market. Besides, secure software process also has become a determinant factor for producing high quality software (O'Regan, 2014; Hui et al., 2014; Merkow & Raghavan, 2010; Voas, 2008; Julia et al., 2008; Offut, 2002). The respondents supported this, whereby majority of them (96%) agreed that considering security and its implementation from early phases of software development is essential. Accordingly, these have become as evidences to support the needs of incorporating both software processes in the proposed model.

Objective 4: To investigate the software practitioners' opinion on the good characteristics of those involved in the Agile and secure software processes

It is widely accepted that software development processes is highly influenced by the team performances, as it involves human interactions (Hazzan & Dubinsky, 2009). This is same with Agile software process, whereby emphasis is given on individuals and interactions, as well as customer collaboration and responds to changes, contradict with the the individual role assignment implemented in the conventional software

development (Nerur et al., 2005). The respondents were asked about the characteristics that should exist among the team and the support that should be given by organization in order to successfully develop high quality software.

The important factors that need to be considered in order to achieve high performance teams are through feedback and communication (Guzzo & Dickson, 1996). Therefore, Agile stresses face-to-face communication and this practice was reported as beneficial in studies of Rao et al. (2011), Liu et al. (2010), Lan and Ramesh (2008), Coram & Bohner (2005) and Paetsch et al. (2003). In order to make these possible, the team must be small sized, as concluded by O'Sheedy and Sankaran (2013) in their study. However, study by Misra and his colleagues (2009) reported as opposed. Furthermore, they should be placed at the same workplace area. This is essential for intense interaction and knowledge sharing (Lindvall et al., 2002). It is also agreed that team members must have high competence and expertise (Tsun & Dac-Buu, 2008; Parsons et al., 2007; Lindvall et al., 2002). On contrary, collocated team was found as not beneficial in Misra et al. (2009).

Furthermore, in Agile, the self-organized team is essential because it can make decisions and plans without depending on managers (Sliger & Broderick, 2008; Tsun & Dac-Buu, 2008; Cockburn & Highsmith, 2001; Agile Manifesto, 2001). This practice has been identified as able to influence the team effectiveness (Franca et al., 2010; Tsun & Dac-Buu, 2008; Moe et al., 2008). Moreover, to enable that the secure software process being implemented properly, team members play a significant role. All team members must be familiar with the security requirements of the system and reach a common understanding

about the security needs. This will help them to adopt the security activities (Microsoft, 2012; McGraw, 2006).

Organization plays an important role in enabling the implementation of Agile principle on the team (Ani Liza et al., 2012b; Mazni, Sharifah Lailee, & Azman, 2011; Strode et al., 2009; Misra et al., 2009; Tsun & Dac-Buu, 2008), as well as the implementation of secure software process (Geer, 2010). The organization must provide environment that supports Agile software process throughout the organization by providing cooperative organizational culture instead of hierarchical, encouraging face-to-face communication, ensuring that the Agile way of software development is universally accepted and offering facilities with proper Agile-style work environment (Sheffield & Lematayer, 2013; Tsun & Dac-Buu, 2008). In relation to the previous studies: Sheffield and Lematayer (2013), Hoda et al. (2011), Strode et al. (2009), Tsun and Dac-Buu (2008), Lindvall et al. (2002), this study also indicates that these are the important practices.

On the other hand, to successfully encourage on the security activities, organizations must give rewards for successful security handling, provide sufficient budget for security initiatives and provide basic security trainings for the staffs. Moreover, providing a separate security team to engineer and evaluate the security of software will be useful as well (Microsoft, 2012; OWASP, 2006).

Similar to Agile and secure software practices, the characteristics of team and organization which obtained the value of 4.45 and above for the Degree of Importance

(DI) are considered as important practices that influence the quality of software. Thus, they are considered as the reference standard in the proposed model.

Objective 5: To inspect the software practitioners' awareness on the importance of software certification and its implementation

- **Software practitioners' opinion regarding the importance of software certification**

As discussed in the findings, the respondents were facing security problems, which indicate that the quality of produced software is still low. Thus, one way to give conformance on the quality of software is through certification (Heck et al., 2010; Rathfelder et al., 2008). Majority of the respondents regardless their familiarities with Agile and secure software processes agree that software certification can lead to higher quality software. In the mean time, majority of the respondents also agree that Agile and secure software processes are essential in today's business environment. However, as discussed earlier, some of the software practitioners are not practicing proper practices of Agile and secure software processes. Consequently, a mechanism to assess and certify the Agile and secure software processes is needed. Thus, this supports the needs of producing a process based software certification model which focuses on the Agile and secure software processes.

- **The implementation of internal assessment/audit and the techniques used**

Although the assessment/audit is very important in order to monitor the quality of software process, surprisingly majority of the respondents did not implement internal

assessment/audit on the software process. This shows that the software processes were implemented as 'ad-hoc', without considering formal procedures and monitoring. However, the assessment can help in doing corrective or preventive actions, whereby the differences between the implemented processes can be differentiated with the actual. By doing so, the root causes can be detected and actions can be taken to eliminate them (Limaye, 2011). Without proper monitoring on the software process, it might cause the quality of produced software to be low, as stated by Deming (1982) and Humphrey (1979), *'the quality of process is influenced by the quality of produced product'*. Among the respondents who implement the assessment/audit, conduct internal assessment/audit by using document review, observation and interview, similar to the implementation of SPAC Model (Fauziah, 2008) (Refer Table 4.16). These techniques are considered in the proposed software process certification model as the data gathering techniques.

- **The use of standards**

The use of standards has become as the key for quality management (Sommerville, 2007). It is intended to ensure that the processes are implemented correctly throughout the organization in all circumstances (Limaye, 2011) and bring uniformity and control to the process of developing software product (Wheeler & Duggins, 1998). Surprisingly, almost half of the respondents do not implement any security standards (Refer Figure 4.7). Furthermore, when the results are classified based on respondents' familiarity with Agile and secure software processes, it indicates that most of respondents do not implement any standards regardless of their familiarity in both software processes. This shows that the use of standard is still being neglected by software practitioners in Malaysia.

4.9 Summary

This chapter has described the instrument design, sampling, instrument testing, data collection and analysis performed in the exploratory study. The study aims to investigate the current practices of software certification which relate to the Agile and secure software processes. Findings from the study reveal the implementation of the software certification, as well as the Agile and secure software processes among the software practitioners in Malaysia. The significance of both software processes in today's business environment is highlighted. Besides, the Agile and secure software processes which influence the quality of software are also revealed. On top of that, the needs of software certification in software industry have been disclosed as a consequence from the quality problems faced by the software practitioners. Accordingly, these findings form the basis for constructing the software process assessment and certification model which focus on Agile and secure software processes. Mainly the Agile and secure software processes as well as the characteristics of people who involve in the Agile and secure software processes are considered in the proposed model. Besides, the data gathering techniques for software process certification are adapted for the proposed model. The proposed model is discussed in the next chapter.

CHAPTER FIVE

ESPAC MODEL DEVELOPMENT

5.1 Introduction

Findings from the theoretical and exploratory studies reveal the main shortcomings in the existing software process certification such as lack of attention given on the Agile and secure software processes, as well as inappropriate synthesis technique used during the certification process. In this chapter, the solutions for the shortcomings are proposed as an enhanced software process certification model. The proposed model is known as Extended Software Process Assessment and Certification Model (ESPAC).

The chapter starts by describing the overview of the ESPAC Model in Section 5.2 whilst its components are explained in details in Section 5.3. Section 5.4 provides the discussion related to the development of the ESPAC Model and Section 5.5 ends the chapter with a summary.

5.2 Overview of ESPAC Model

The main aim of the ESPAC Model is to assess and certify the Agile and secure software processes. It is constructed by adapting the Evaluation Theory (Scriven, 1991) as the base theory besides the outcomes from the theoretical and exploratory studies. The Evaluation Theory consists of six (6) components which are target, evaluation criteria, reference standard, data gathering techniques, synthesis technique and assessment process. However, this study adapts the theory by including another component which is the

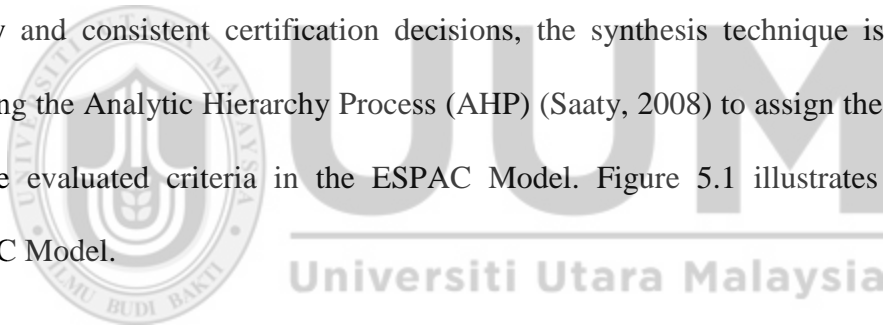
Achievement Index that consists of quality levels and certification level. Furthermore, when constructing the proposed model, the following existing studies are referred:

- SPAC Model (Fauziah, 2008): This model was referred as the base model, whereby most of the components are adapted. These include the target, evaluation criteria, reference standard, assessment process, and certification outcomes. One component is adopted which is the data gathering techniques, besides the practices for process origin, resource management, tools and techniques, standards & procedures, technical skills, knowledge, experience and environment.
- CMMI Version 1.3 (CMMI Product Team, 2010): This model was referred to obtain the relevant Agile software practices and the assessment process.
- ISO/IEC 15504 (Mas et al., 2012; Van Loon, 2007; Galin, 2004; El Emam & Birk 2000): This model was utilized by adapting its Achievement Index.
- Agile Manifesto (Agile Manifesto, 2001): The Agile principles and values are obtained.
- Agile methods (XP, Scrum and Agile Modeling) (Abrahamsson et al., 2010): The Agile software practices were gathered.
- The characteristics of people who involve in Agile and secure software processes: existing studies such as O'Sheedy & Sankaran (2013) and Litecky et al. (2012).
- ISO/IEC 27001 (ISO, 2015; Evans et al., 2010; Humphreys, 2008) and ISO/IEC 21827 (Davis, 2013; Carnegie Mellon University, 2003): These security standards were referred to obtain the relevant secure software practices.
- MS SDL (Microsoft, 2012; Merkow & Ragavhan, 2010), Touchpoints (McGraw, 2011; Julia, 2008; McGraw, 2006) and CLASP (Merkow & Ragavhan, 2010; OWASP, 2006): These models were referred to obtain the secure software practices.

Apart from the theoretical study, this study also conducted an exploratory study which gathers the opinion and perceptions of software practitioners in Malaysia on the software certification which relates to the Agile and secure software processes. Besides the Agile

and secure software processes which influence the quality of software, the outcome that obtained high consideration among the software practitioners is the characteristics of people who involve in these two software processes (Refer Chapter Four). These software processes and characteristics are used to build the reference standard of the ESPAC Model. The Quality Function Deployment (QFD) approach (Zultner, 1992; Cohen, 1995) is used to systematically organize the reference standard. Furthermore, the data gathering techniques used by the respondents are adopted in the proposed model.

Moreover, software certification involves multiple criteria assessment whereby each evaluation criterion might have different influence to the project. Thus, to produce better quality and consistent certification decisions, the synthesis technique is improved by adapting the Analytic Hierarchy Process (AHP) (Saaty, 2008) to assign the weight values for the evaluated criteria in the ESPAC Model. Figure 5.1 illustrates the proposed ESPAC Model.



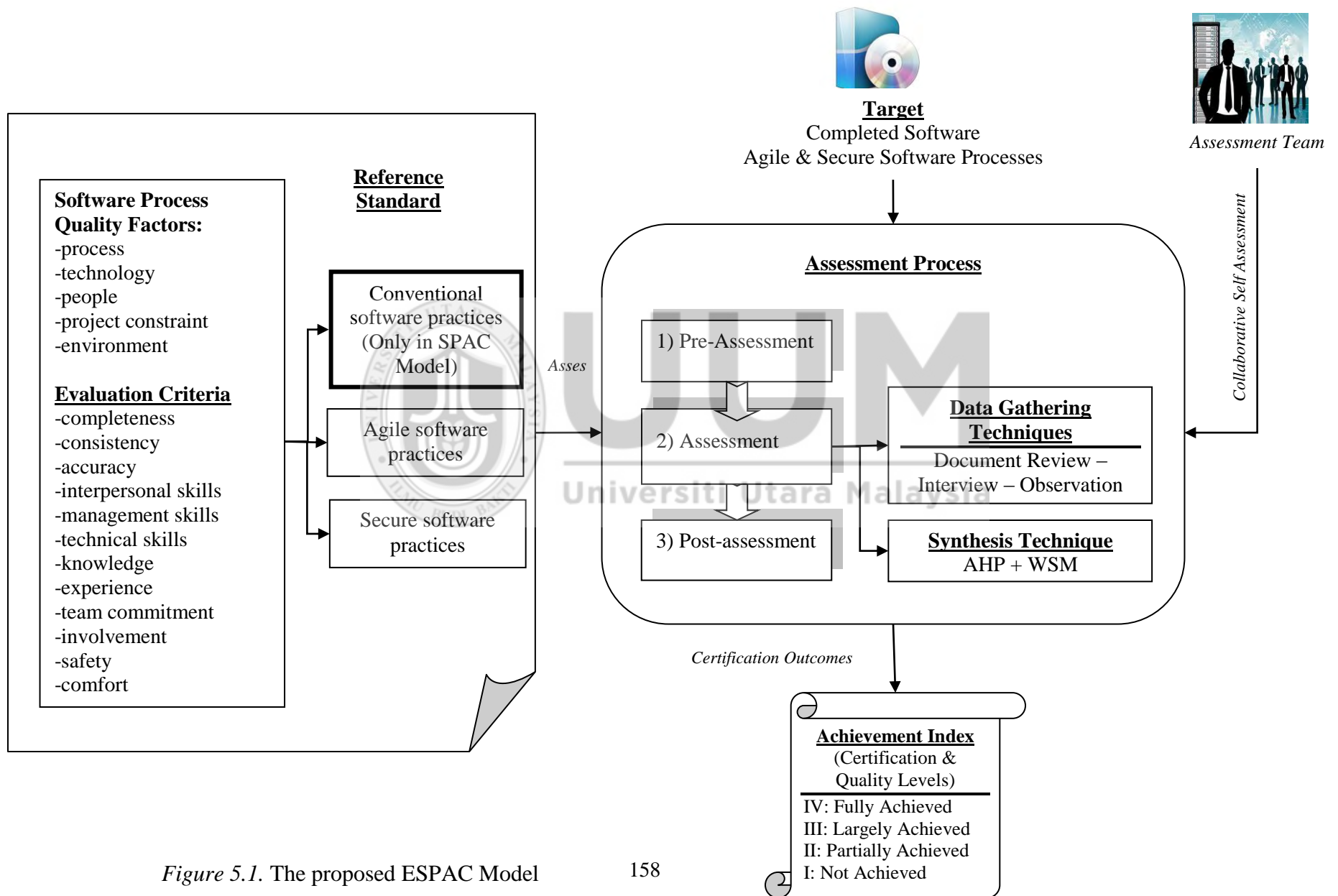


Figure 5.1. The proposed ESPAC Model

Most of the ESPAC Model components are adapted from the SPAC Model. The assessment target of the SPAC Model is similar to that of the ESPAC Model, whereby it is intended for a project that has been completed and ready to be delivered to customer. However, the difference is that the SPAC Model assesses the conventional software process, while the ESPAC Model assesses the Agile and secure software processes. Consequently, the reference standard in the SPAC Model only focus on conventional software process, whereas in the ESPAC Model, the Agile and secure software processes are considered. Basically, there is a slight difference between the sub factors considered in the reference standard for the conventional, Agile and secure software processes, specifically in the management process sub factors. For the conventional software process, the project, quality, risk, and configuration management, as well as the formal technical reviews are considered. Conversely, only change and project management are appropriate for the Agile software process, while the security and risk management are appropriate for the secure software process. On the other hand, the resource management, training, tools and techniques, standards and procedure, technical skills, knowledge, experience and environment, as well as process origin remain the same as in the SPAC Model.

The assessment process in the ESPAC Model is not only adapted from the SPAC Model but also from other studies (SCAMPI Upgrade Team, 2011, Lascelles & Peacock, 1996). On the other hand, the data gathering techniques are adopted from the SPAC Model and the outcome of the exploratory study. As for the assessment method, the SPAC Model uses the collaborative assessment method. This method is adapted in the ESPAC Model by proposing collaborative self-assessment method, which combines the collaborative and self-assessment methods.

Additionally, the synthesis technique in the ESPAC Model considers weight values for the evaluation criteria by using the AHP technique. Previously, the SPAC Model uses equal weight values. Even though the ESPAC Model uses the same outcomes as the SPAC Model which are the quality and certification levels, the calculations for obtaining both outcomes are different since the ESPAC Model considers weight values for the evaluation criteria. Furthermore, the achievement levels are determined based on the Achievement Index as adapted from the ISO/ IEC 15504. Table 5.1 summarizes the comparisons between both models. The discussions in the next section focus on the components of the ESPAC Model that have been enhanced from the SPAC Model.

Table 5.1
The Comparisons of the SPAC and ESPAC Models

Descriptions	SPAC Model	ESPAC Model
Target	<ul style="list-style-type: none"> • Completed project and ready to be delivered to customers • Focuses on conventional software process 	<ul style="list-style-type: none"> • Completed projects and ready to be delivered to customers • Focuses on Agile and secure software processes
Evaluation criteria & Reference standard	Focuses on conventional software process	Focuses on Agile and secure software processes
Assessment process	Three phases of assessment	Three phases of assessment, adapted from SPAC, SCAMPI for CMMI V1.3 and Lascelles and Peacock (1996)
Data gathering techniques	Document review, interview and observation	Adopt from the SPAC Model and outcome from exploratory study
Assessment method	Collaborative assessment method	Adapt from the SPAC Model and self-assessment method: collaborative self-assessment method
Synthesis Technique	Equal weight values	Define weight values by using AHP.
Certification outcomes	Quality levels and certification level	Same quality and certification levels, however the calculations are different, as well as the Achievement Index

5.3 The Components of the ESPAC Model

The ESPAC Model consists of seven (7) components which are target, evaluation criteria, reference standard, data gathering techniques, assessment process, synthesis technique and the Achievement Index. The next sub sections discuss in detail about the components.

5.3.1 Target

Referring to this study, the target is the software process implemented in the projects that have been completed and the software that is ready to be delivered to customers. Furthermore, the software process comprises of the Agile and secure software processes, since the aim of the ESPAC Model is to assess the quality of Agile and secure software processes.

5.3.2 Evaluation Criteria

In this component, the required evaluation criteria for evaluating the target are defined. The evaluation criteria describe WHAT should be assessed by the ESPAC Model. It comprises the factors that influence the quality of Agile and secure software processes (Refer to Section 2.2.7).

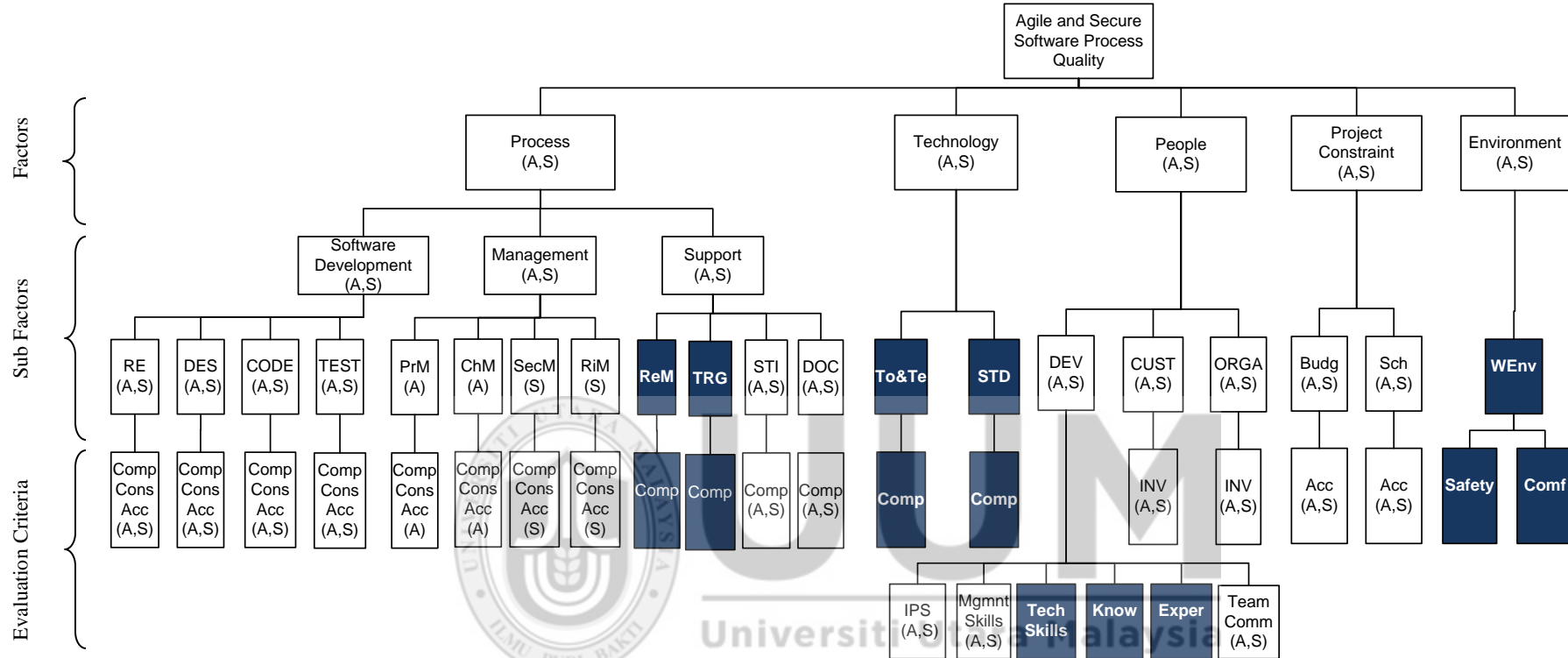
It is already known that if the process is in good quality, the produced product would also have high quality. However, given that the software process is implemented by people, there are other influential factors that can indirectly influence the software quality. Thus, to ensure the correctness of the assessment and certification outcomes, these factors have been taken into consideration.

Based on the literature findings in Chapter Two, the five factors that have influence on the quality of software are as follows:

- Process: the quality of implemented process
- Technology: the technology used during software development
- People: the quality of people who involved during the development
- Project constraints: the ability to produce software on-time and within budget.
- Environment: the safety and comfort of working environment where the software is developed

However, these factors cannot be measured directly, thus they are decomposed to sub factors and criteria. Each of the factors comprises of at least one sub factor. For each sub factor, at least one evaluation criterion is defined. Since the ESPAC Model focuses on the Agile and secure software process assessment, these factors, sub factors and evaluation criteria are considered from the perspectives of both software processes. The complete factors and evaluation criteria are provided in Figure 5.2. They are organized in a hierarchical structure, as adapted from the AHP technique.

There are three (3) types of factors/sub factors/evaluation criteria: 1) factors/sub factors/evaluation criteria for both Agile and secure software processes, labeled as (A, S), 2) sub factors/evaluation criteria that are solely for Agile or secure software processes, labeled as (A) or (S) respectively, 3) sub factors/evaluation criteria that contains same practices for conventional, Agile and secure software processes, shaded (assessment is performed only once for both Agile and secure software processes). The assessment is performed depending on the background of the assessed project, either it is developed by using Agile or secure software processes. On the other hand, if the project is developed by using the conventional approach, then the SPAC Model is used.



Legend:

- | | | | | |
|---------------------------|----------------------|--------------------------------|-----------------------------|-------------------------------|
| A: Agile Software Process | Comf: Comfort | IPS: Interpersonal Skills | RE: Requirement Engineering | Team Comm: Team Commitment |
| Acc: Accuracy | CUST: Customer | Know: Knowledge | RiM: Risk Management | To&Te: Tools & Techniques |
| Budg: Budget | DES: Software Design | Mgmt Skills: Management Skills | S: Secure Software Process | Tech Skills: Technical Skills |
| ChM: Change Management | DEV: Developer | ORGA: Organization | STI: Staff Initiative | TEST: Testing |
| CODE: Coding | DOC: Documentation | PrM: Project Management | Sch: Schedule | TRG: Training |
| Comp : Completeness | Exper: Experience | ReM: Resource Management | SecM: Security Management | WEnv: Working Environment |
| Cons: Consistency | INV: Involvement | STD: Standard & Procedure | ■ : Similar Practices | |

Figure 5.2. The hierarchy tree of the evaluation criteria

The Agile and secure software processes are assessed based on their effectiveness and efficiency. The effectiveness is assessed based on the completeness, consistency and accuracy of performing the software process in order to produce high quality software. The effectiveness is influenced by the characteristics of people who involved in the development, use of technology and working environment. On the other hand, the efficiency is assessed based on the capability of producing the software within the estimated budget and on-time (Fauziah, 2008). Referring to Figure 5.2, the descriptions for each factor are provided in Table 5.2.

Table 5.2

The Assessed Factors

Factors	Descriptions	Sub Factors
Process	Three types of processes are assessed: software development, management and support. The software development and management process are assessed in terms of completeness, consistency and accuracy, while support process is assessed in terms of completeness. Completeness of software process denotes the correctness in performing the process and the production of appropriate documentation. Consistency on the other hand refers to the use of standard and procedure, while accuracy indicates the use of tools, methods and technology.	<ul style="list-style-type: none"> • Software development <ul style="list-style-type: none"> – Requirement Engineering – Software Design – Coding – Testing • Management <ul style="list-style-type: none"> – Project Management – Change Management – Security Management – Risk Management • Support <ul style="list-style-type: none"> – Resource Management – Training – Staff Initiatives – Documentation
People	The developers are assessed in terms of interpersonal, management, and technical skills, knowledge, experience and team commitment, while the organization and customer are assessed based on the involvement.	<ul style="list-style-type: none"> • Developers • Customers • Organization

Technology	The technology is assessed in terms of completeness in the standard and procedure's implementation, as well as the completeness in the use of tools and technique.	<ul style="list-style-type: none"> • Standard & procedure • Tools & technique
Project constraint	The project constraint is assessed in terms of the accuracy in developing software within the budget and schedule.	<ul style="list-style-type: none"> • Budget • Schedule
Environment	The environment is assessed based on the safety and comfort of the organization's working environment.	<ul style="list-style-type: none"> • Working environment

5.3.3 Reference Standard

The reference standard is the benchmark used by the assessors to perform the assessment and certification process. The ESPAC Model does not only define WHAT need to be assessed through the evaluation criteria, but also HOW these evaluation criteria are assessed through the list of Agile and secure software practices. Each evaluation criterion is assigned with appropriate Agile and secure software practices that need to be performed towards achieving the specified evaluation criterion. By having this structure, the assessors are guided on what they should assess during the assessment.

In order to systematically organize the WHATs and HOWs, the Quality Function Deployment (QFD) approach is adapted, whereby the first phase of QFD is performed by developing the House of Quality (HOQ) (Cohen, 1995; Zultner, 1992). The other three phases of the QFD are not necessary for this study as the structures and analyzing methods are the same (Lai-Kow & Ming-Lu, 2005). There are five main areas in the HOQ adapted in this study as the reference standard: the WHATs,

HOWs, relationships between WHATs and HOWs, weight for each evaluation criterion and evaluation criteria scores, as shown in Figure 5.3

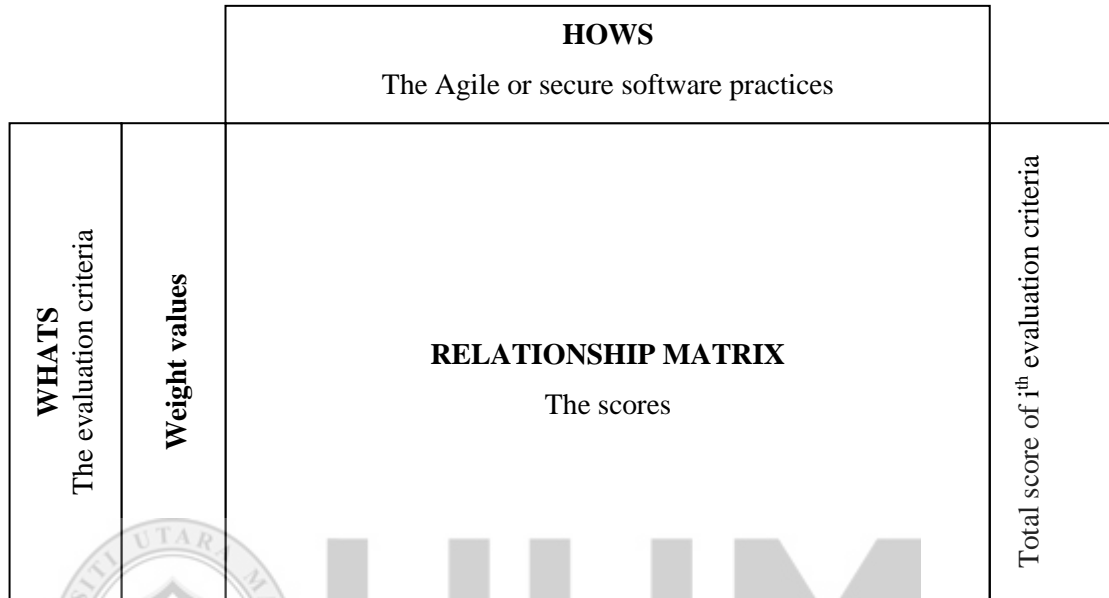


Figure 5.3. The structure of the reference standard

The first and second areas in the reference standard are the WHATs and HOWs. Each evaluation criterion represents the WHATs while the Agile and secure software processes that need to be performed represent the HOWs. There are 36 evaluation criteria for Agile as well as secure software processes defined in the ESPAC Model. Furthermore, there are 189 practices defined for Agile software process, whereas for secure software process, 146 practices are defined.

The third area in the reference standard is the area of relationship matrix among WHATs and HOWs. This area acts as the scoreboard for the assessment. A scale of five values is used; 1= *Never*, 2= *Rarely*, 3= *Sometimes*, 4= *Often*, and 5= *Always*, which is adapted from the Likert Scale (Zikmund et al., 2010). Using this scale, the

assessors assign the score for each assessed practice. The fourth area of the reference standard is the weight values. Generally, the weight values are determined arbitrarily in the QFD which may lead to the inconsistency and degrade of the quality of decisions made (Ho, 2008). Therefore, this model adapted the AHP technique to overcome this drawback. Section 5.3.6 presents the detailed explanation about the AHP implementation. The fifth area is the total scores obtained for each evaluation criterion. The calculation for obtaining these scores is also provided in Section 5.3.6. The example of the reference standard is illustrated in Figure 5.4.

Agile Requirement Engineering		WEIGHT VALUES FOR EVALUATION CRITERIA											
		The scope of project were identified at the beginning of project to create initial prioritized stack of requirements	Customers were available on-site for face-to-face discussions during requirement elicitation or at least can be easily reached through phone or skype or teleconferencing	The requirements were gathered, elaborated, analyzed and validated iteratively and incrementally	The requirements were written short statements using cards or tools	Requirements were prioritized and can be re-prioritized by customers throughout the development	The requirements were validated by customers in review meetings by using releases	The product backlog and iteration backlog were produced to ensure the consistency and traceability of requirements	Appropriate procedure is used to handle frequently changing requirements	The requirements were documented by following a particular standard	The requirements were gathered using particular method (s)	Appropriate tools were used to facilitate defining and translating the requirements	Particular notation (s) was/were used to represent the requirements
Completeness													
Consistency													
Accuracy													
		TOTAL SCORE OF EVALUATION CRITERIA											

Figure 5.4. The example of reference standard for Agile Requirement Engineering

5.3.4 Data Gathering Techniques

The data are gathered by using multiple techniques, which are the document review interview, and observation. These techniques are adopted from the SPAC Model. Furthermore, the outcomes from the exploratory study also reveal that these techniques are widely used among the software practitioners. Using multiple data-gathering techniques can improve the understanding for the assessment team and give better confirmation on the assessment made (SCAMPI Upgrade Team, 2011). The documents that are reviewed can be direct documents or indirect documents. Direct documents are the tangible outputs resulting directly from the implementation of a practice, while indirect documents are outputs produced as a consequence of performing some other related activities. Table 5.3 lists the data gathering techniques for each of the sub factors.

Table 5.3

The Data Gathering Techniques

Factors	Sub factors	Data gathering techniques
Software Development	Requirement engineering	Document review + Interview (for clarification)
	Design	
	Coding	
	Testing	
Management	Project management	
	Change management	
	Security management	
	Risk management	
Support	Staff initiative	
	Documentation	
	Resource management	
	Training	
People	Developer	Interview
	Customer	
	Organization	

Technology	Tools and techniques	Interview + Observation
	Standard and procedure	
Project constraint	Budget	Document review
	Schedule	
Environment	Working environment	Observation

5.3.5 Assessment Process

The assessment process is the series of activities that need to be performed during the certification process. Figure 5.5 depicts the assessment process of ESPAC Model. The activities and workflow of the assessment process is presented using one of the quality tools, which is the flow chart (Ishikawa, 1976 in Mach & Guaqueta, 2001). There are three (3) main phases in conducting the software process certification: pre-assessment, assessment and post-assessment. Each of the assessment phases consists of several processes and activities. These phases, processes and activities are adapted from the SCAMPI for CMMI version 1.3 (SCAMPI Upgrade Team, 2011), SPAC Model (Fauziah, 2008) and Lascelles and Peacock (1996). They are elaborated further subsequently.

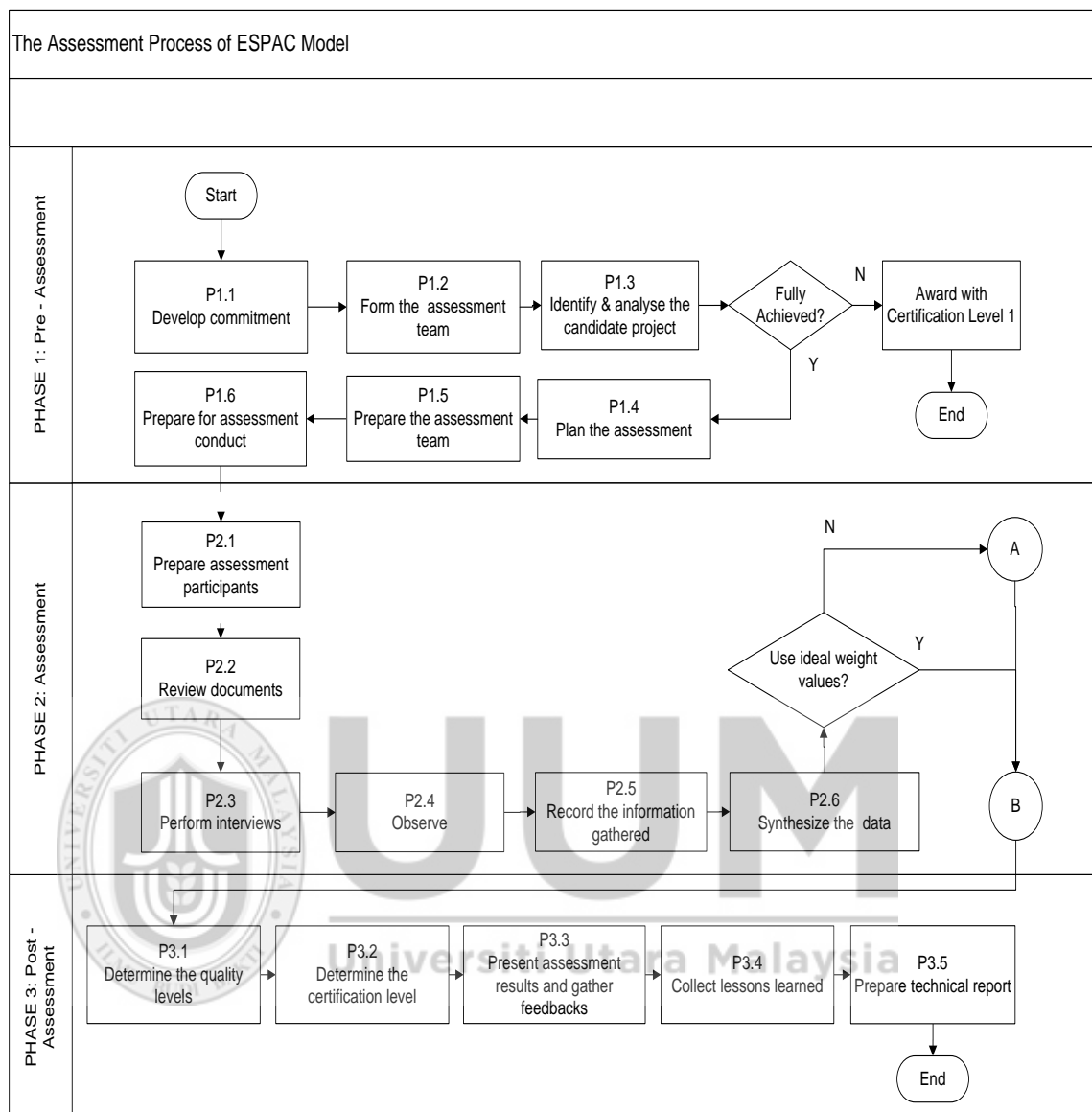


Figure 5.5. Assessment process of the ESPAC Model

1) Pre- assessment

This phase consists the process of developing commitment senior management on the use of self-assessment as a mechanism to ensure software quality. Then the assessment team is formed and then the assessment is planned and prepared. The input needed for this phase is the certification kit which will guide the

implementation of the assessment. At the end of this phase, the assessment plan is produced and the assessment forms are printed. At the same time, the project documentations are also gathered, which will be the input for the next phase. There are six (6) processes and seventeen (17) activities that need to be performed during this phase, as presented in Table 5.4.

Table 5.4

Descriptions of the Pre-Assessment Phase

Processes	Purposes	Activities
P1.1 Develop commitment	Establish organizational leaders' commitment.	<p>A1.1.1 Educate senior management on the use of self-assessment for the software process certification as a mechanism to ensure quality.</p> <p>A1.1.2 Obtain senior management's agreement to perform the assessment and certification.</p>
P1.2 Form the assessment team	Select the assessment team members which consist of project manager, assessors from other team and representative of the assessed team.	<p>A1.2.1 Identify the project manager for the assessment.</p> <p>A1.2.2 Identify the assessors among software practitioners from outside of the assessed project.</p> <p>A1.2.3 Identify the representative among the software practitioners of the assessed project.</p>
P1.3 Identify & analyse the candidate project	Determine whether the assessment can be conducted or not.	<p>A1.3.1 Identify the project that will be assessed.</p> <p>A1.3.2 Perform the 'Process Origin' assessment. If the score achieves 'Fully Achieved', then the assessment can be conducted. If the score is less than that, then the assessment cannot be conducted and the certification level I will be awarded.</p> <p>A1.3.3 Identify the background of the project in term of its software process (conventional/Agile/secure software processes)</p>

P1.4 Plan the assessment	Plan the assessment activities, the resources needed and the duration of assessment.	<p>A1.4.1 Describe the activities that will be performed.</p> <p>A1.4.2 Plan the logistics, for example access to rooms, equipments and supplies needed for administrative purposes.</p> <p>A1.4.3 Determine the assessment schedule.</p>
P1.5 Prepare the assessment team	Brief to the assessment team to make them familiar with the assessment plan, the ESPAC Model and the assessment process, as well as the tools and techniques that will be used during assessment.	<p>A1.5.1 The project manager briefs the assessment team:</p> <ul style="list-style-type: none"> i. Purpose and benefits of the ESPAC Model. ii. How the certification will be conducted? iii. Who will be involved in the assessment? iv. Decide whether to use the suggested ideal weight or to perform AHP. v. How to conduct the AHP technique in order to obtain the weights of evaluation criteria? (if they decide to assign their own weight values).
P1.6 Prepare for assessment and conduct	Prepare for assessment and ensure the readiness to conduct assessment, including confirmation on the availability of documents, staffs, logistics arrangements and assessment team's commitment.	<p>A1.6.1 Print out the assessment forms.</p> <p>A1.6.2 Select the documents that will be reviewed.</p> <p>A1.6.3 Select who will be interviewed (assessment participants) e.g. developers, customers, security advisors.</p> <p>A1.6.4 Ensure that the assessment logistics has been arranged and confirmed.</p> <p>A1.6.5 Ensure that the assessment team is available and prepared to conduct the assessment.</p>

2) Assessment

This phase is the main phase in the software process certification. The major activity is to assess the software process by comparing them with the reference standard. This is done by reviewing the documents, interviewing and observing. The score is assigned by referring to the five scales provided in the assessment form. Additionally, the weight for each evaluation criterion is obtained by performing the AHP technique (Refer Section 5.3.6). However, if the assessment team is not familiar with the AHP technique, the suggested ideal weight can be used (Refer Appendix M). The inputs for this phase are the assessment plan, assessment forms and the project's documentation. At the end of this phase, the team will have the score for evaluation criteria as well as the overall certification score in hand. There are six (6) processes and eleven (11) activities that need to be implemented during this phase, as provided in Table 5.5.

Table 5.5

Descriptions of the Assessment Phase

Processes	Purposes	Activities
P2.1 Prepare assessment participants	Ensure that the assessment participants are available to participate in the assessment and informed on the assessment that will be conducted, as well as the purposes and objectives.	A2.1.1 Check the availability of assessment participants. A2.1.2 Brief to the assessment participants on the assessment, its purposes and objectives.
P2.2 Review documents	Assess the documents produced during the development of the project. The documents can be the direct documents (tangible outputs resulting directly from the implementation of a practice), such as:	A2.2.1 Assess the documents.

- Product backlog
- Sprint backlog
- User stories
- High level requirements, architecture
- Iteration models
- Code
- Unit tests
- Acceptance tests
- Burn down charts

OR indirect documents (outputs produced as a consequence of performing some other related activities), such as:

- User manual
- Meeting minutes (review, retrospectives and planning meetings)
- Presentations

P2.3 Perform interviews	Interview the assessment participants to get information about the organization and also to clarify any information that could not be obtained by document review.	A2.3.1 Interview the assessment participants.
P2.4 Observe	Observe the working environment, whether the organization provides comfortable, suitable and safe environment.	A2.4.1 Observe the working environment. For Agile environment, the following can be observed: <ul style="list-style-type: none"> - Informative workspace (whiteboard sketch/burn down charts) - Team placement - Daily working routine
P2.5 Record the information gathered	Create lasting records of the information gathered.	A2.5.1 Record the existence or absence of reviewed documents. A2.5.2 Record the outcomes from the interviews. A2.5.3 Record the outcomes from the observation made.
P2.6 Synthesize the data	Assign weight for the evaluation criteria and allocate score for each assessed practices. Then, the score	A2.6.1 Obtain the weight values for each evaluation criterion (Refer to A2.6.1 for the implementation

for each evaluation criterion as well as the overall certification score are obtained (Detailed explanation in Section 5.3.6).

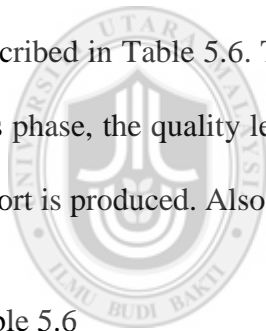
steps).

A2.6.2 Assign scores by using the assessment form (Refer to A2.6.2 for the detailed process).

A2.6.3 Calculate the score for evaluation criteria and overall certification score (Refer to A2.6.3 for detailed process)

3) Post- assessment

The final phase in the software process assessment and certification is the post-assessment. During this phase, the final outcomes from the assessment process are obtained. There are five (5) processes and nine (9) activities need to be performed, as described in Table 5.6. The input for this phase is the assessment data. At the end of this phase, the quality levels and certification level are determined and the technical report is produced. Also, the lessons learned are collected.



UUM
Universiti Utara Malaysia

Table 5.6

Descriptions of the Post-Assessment Phase

Processes	Purposes	Activities
P3.1 Determine the quality levels	The quality levels are determined based on the proposed Achievement Index, which are Level I to Level IV.	A3.1.1 Map the scores obtained for each evaluation criteria with the Achievement Index to determine the quality levels.
P3.2 Determine the certification level	The certification level is determined based on the overall score obtained. There are four levels of certification, which are Level I to Level IV.	A3.2.1 Map the scores obtained for overall assessment with the Achievement Index to determine the certification level.
P3.3 Present assessment results and	Present the assessment result to the assessment team as well as the organization and get	A3.3.1 Project manager presents the assessment result to the assessed team.

gather feedbacks	agreement on the result. The future improvements that can be made are suggested as well.	<p>A3.3.2 Get feedback from the assessed team on the assessment results.</p> <p>A3.3.3 Project manager presents to the top management on the assessment result.</p> <p>A3.3.4 Highlight on the practices that can be improved.</p>
P3.4 Collect lessons learned	Document important issues which were identified during the assessment for future improvement.	<p>A3.4.1 Identify the issues that worked well and what need to be improved.</p> <p>A3.4.2 Identify suggestions for improving the method or its execution.</p>
P3.5 Prepare technical report	Prepare a technical report for the team.	<p>A3.5.1 Prepare the technical report which consists of:</p> <ul style="list-style-type: none"> – The project’s profile. – The achieved scores, quality levels and certification level. – Suggestion of practices that can be improved.

The assessment is performed by the assessment team which consists of the organization’s own people. This is intended to reduce the cost since the assessment is only performed within the organization (Ritchie & Dale, 2000). The ESPAC Model proposes collaborative self-assessment method, which is adapted from self-assessment (Serkani, Mardi, Najafi, Jahanian, & Heart, 2013; Tari & Heras-Saizarbitoria, 2012; Lascelles & Peacock, 1996) and collaborative assessment (Fauziah et al., 2011; Jamaiah, 2007).

The assessment team is led by a project manager and composed of assessors from outside of the team being assessed (software practitioners from other software development team in the organization). This is aimed to eliminate biased assessment (Fauziah, 2008; Jamaiah, 2007; Fabbrini et al., 2006; Voas, 1999). Additionally, one

representative from the assessed team will co-operate as the assessment team to facilitate ideas exchanges between the assessment team (Lascelles & Peacock, 1996). Moreover, the assessment process can be accelerated since the project is already understood by the representative (Fauziah, 2008; Jamaiah, 2007; Voas, 1999; Vermesan, 1998).

The ESPAC Model calls for at least five assessors in a team, based on Byrnes and Philips (1996). The team leader should have technical and managerial experiences and have participated in two or more assessment and certification exercises as an assessor before being appointed as a team leader. Meanwhile, the assessors are selected among software practitioners who have experience in software development for at least five (5) years and have knowledge in assessment and certification.

5.3.6 Synthesis Technique

After all of the required information are collected, the synthesis technique is applied to synthesize the information gathered. Basically the synthesis technique in the ESPAC Model is aimed to improve the quality and consistency of the certification decision made whereby the weight value allocation is considered. There are three (3) main activities that need to be accomplished for synthesizing the information, as described in Table 5.5 (Activities A2.6.1. to A2.6.3). They are elaborated further subsequently.

A2.6.1 Obtain the weight values for each evaluation criterion

The first activity is to obtain the weight values for the evaluation criteria. The weight values should be assigned to each evaluation criterion since each of them might have different importance, depending on the assessment team's preference. This indicates the importance of an evaluation criterion relative to other evaluation criteria under consideration. For example, the requirement engineering phase is usually considered more important than the software design. On the other hand, coding is typically considered as more important than the requirement engineering. Thus more attention is given to the phases which are considered as more important.

The ESPAC Model suggests the ideal weight values for each evaluation criterion. However, they are flexible and can be changed based on the assessors' preference. These ideal weight values are obtained through the focus group discussion which was held in this study (Refer to Chapter Six for detailed explanations). They are gathered based on the opinions of seven software practitioners who have similar backgrounds. Based on their opinions, the same weight values can be used for the evaluation criteria of both Agile and secure software processes. Currently the weight values are suitable for web based projects, since the software practitioners assigned the weight values based on their experience in developing web based applications. The list of ideal weight values are provided in Appendix M.

Nevertheless, if the assessment team decided not to utilize the ideal weight values as suggested, the team has to obtain weight values for the evaluation criteria on its own. To accomplish this, the group AHP technique is applied, since it provides a

systematic technique to evaluate the priorities among the evaluated criteria (Saaty, 2008). The detailed steps for obtaining the weight values are provided in Figure 5.6 and discussed further in the next sub sections. The entry point for this process is synthesizing the data (P2.6). It starts from A2.6.1.1 by constructing pair wise matrix and ends with A2.6.1.5, whereby the global weight value for each evaluation criterion is obtained.

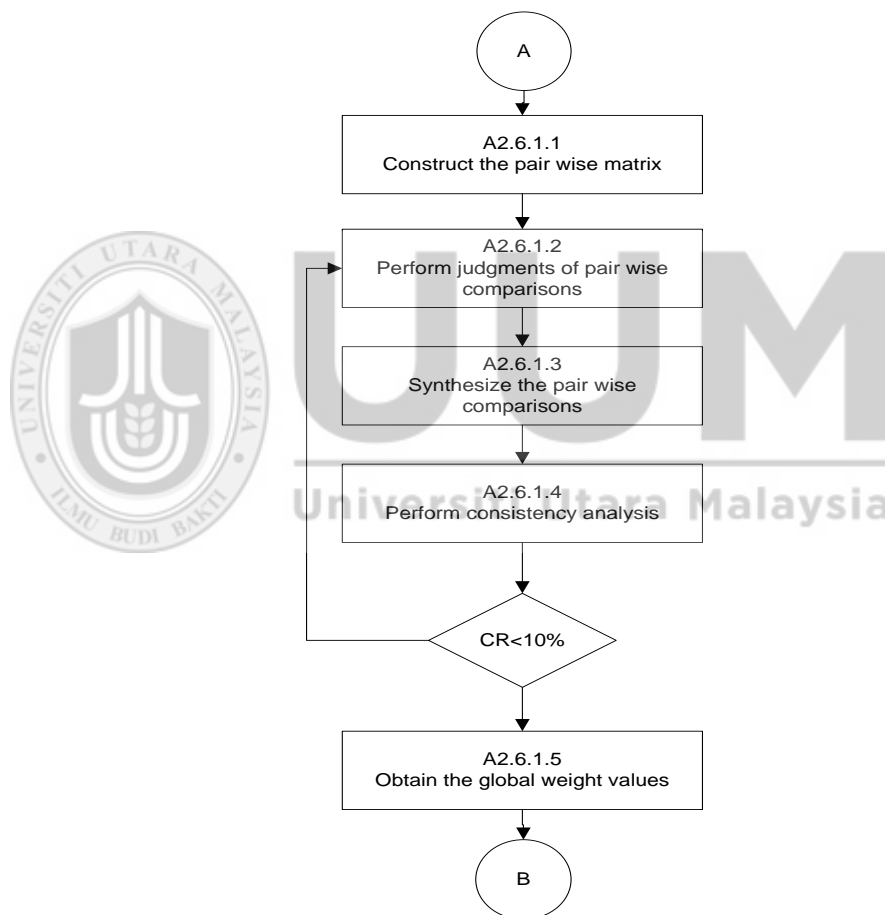


Figure 5.6. Steps to calculate the weight values

A2.6.1.1: Construct the pair wise matrix

The pair wise matrixes are used to perform the pair wise comparison. Accordingly, before performing the pair wise comparison, the pair wise matrixes need to be constructed for the assessed evaluation criteria. The evaluation criteria of the ESPAC Model are organized into two hierarchy trees whereby each of them consists of four levels. One tree composes evaluation criteria for the Agile software process (Refer Appendix D), while the other is for the secure software process (Refer Appendix E). Thus, to perform the pair wise comparisons, the sibling criteria at each level are compared in pairs to judge their importance. They are organized in matrix of two dimensions (square matrix) whereby the compared criteria are sorted vertically in the first column and horizontally in the first row of the matrix, as depicted in Table 5.7. The evaluation criteria are represented by $(C_1...C_n)$. The relative importance of each C_i in the column are compared to the C_j in the row, which are represented by a_{ij} by following the rules of $a_{ij} = 1/a_{ji}$ when $i \neq j$, and $a_{ii}=1$ when $i=j$.

Table 5.7

The Pair Wise Matrix

Criteria	C_1	C_2	..	C_n
C_1	1	$a_{1,2}$..	$a_{1,n}$
C_2	$a_{2,1}$	1	..	$a_{2,n}$
.
.
C_n	$a_{n,1}$	$a_{n,2}$	$a_{n,3}$	$a_{n,n}$

As an example, the five factors that influence the quality of software in the first level of the Agile software process hierarchy tree are compared using one pair wise comparison matrix, as shown in Table 5.8.

Table 5.8

The Pair Wise Comparison Matrix for Level One

Criteria	Process	Technology	People	Project Constraint	Environment
Process	1	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
Technology	$1/a_{2,1}$	1	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
People	$1/a_{3,1}$	$1/a_{2,3}$	1	$a_{3,4}$	$a_{3,5}$
Project Constraint	$1/a_{4,1}$	$1/a_{4,2}$	$1/a_{4,3}$	1	$a_{4,5}$
Environment	$1/a_{5,1}$	$1/a_{5,2}$	$1/a_{5,3}$	$1/a_{5,4}$	1

Furthermore, in the second level, the sub factors for the process are software development, management and support. These sub factors are compared to one another with respect to the process by using one pair wise matrix, as depicted in Table 5.9. The same procedure is used for the other factors/sub factors/evaluation criteria for the whole hierarchy tree.

Table 5.9

The Pair wise Comparison Matrix for Level Two

Criteria	Software Development	Management	Support
Software Development	1	$a_{1,2}$	$a_{1,3}$
Management	$1/a_{2,1}$	1	$a_{2,3}$
Support	$1/a_{3,1}$	$1/a_{3,2}$	1

There is a total of 32 pair wise comparison matrixes need to be implemented for both the Agile and secure software processes, as summarized in Table 5.10.

Table 5.10

Summary of the Pair Wise Comparison Matrixes for the ESPAC Model

Approaches	Description	Level 1	Level 2	Level 3	Level 4
Agile software process	Number of criteria	5	11	24	22
	Number of pair wise comparison matrixes	1	4	5	6
Secure software process	Number of criteria	5	11	24	22
	Number of pair wise comparisons matrixes	1	4	5	6

A2.6.1.2 Perform judgments of pair wise comparisons

To perform the judgments of a pair wise comparison, the relative importance of each two criteria in the matrix is compared, for example “is C_1 is more/equally/less important than/to C_2 by a factor of 2/3/4/5/6/7/8/9 (a_{ij})”. In order to determine the importance value for each pair wise comparison (a_{ij}), the scale of 1 to 9 by Saaty (1990) is used, as presented in Table 5.11.

Table 5.11

Scale of Pair Wise Comparison (Saaty, 1990)

Intensity of Importance	Definition
1	Equal importance
3	Moderate importance of one over another
5	Essential or strong importance
7	Very strong importance
9	Extreme importance
2,4,6,8	Intermediate values

The number of the pair wise comparisons that need to be performed for each matrix is determined by the following formula:

$$\text{Pair wise comparisons in each matrix} = n(n-1)/2 \quad (5.1)$$

where n is the number of criteria in the matrix.

For instance, there are five criteria listed in Table 5.8. Thus, $n=5$. Consequently, the number of pair wise comparisons that need to be performed can be determined by performing Equation 5.1, whereby: $5(5-1)/2 = 10$ pair wise comparisons.

Table 5.12 shows the pair wise comparison for the sub factors of the software development. There are four criteria, thus the number of pair wise comparison is $4(4-1)/2=6$. The diagonal elements of the matrix are assigned with value 1, since $a_{ii}=1$ when $i=j$. Comparisons are only made for the upper triangular matrix, as the lower triangular matrix comprise of the reciprocals of these values. In the first row of the matrix, the software design and testing are considered as 7 times more important compared to requirement engineering, while coding is considered as 8 times more important compared to requirement engineering.

Table 5.12

Pair Wise Comparisons for Sub Factors of Software Development

Software Development	RE	DES	CODE	TEST
Requirement engineering (RE)	1	7	8	7
Software Design (DES)	1/7	1	3	1
Coding (CODE)	1/8	1/3	1	1
Testing (TEST)	1/7	1	1	1

The process of performing judgment of each evaluation criteria is conducted by a group of assessors. During the session, the assessment team leader (project manager) shall raise each of the pair wise comparisons one by one, while probing open ended questions to the assessment team on their opinion about the evaluation criteria being compared. The assessors shall use the pair wise comparison form as a guideline to perform the judgments (Refer Appendix F). This representation of pair wise comparisons can help the assessors to complete the lengthy process of pair wise comparisons faster (Hajkowicz, McDonald, & Smith, 2000).

To perform the judgments, the ESPAC Model adapts the planning poker which is used in Agile environment for doing estimations (Dyba, Dingsoyr, & Moe, 2014; Mahnic & Hovelja, 2012). By using this technique, each assessor is provided with a set of cards which consists of numbers 1 to 9. These numbers represent the importance values which are used for making judgments of pair wise comparisons in AHP. When the team leader raises the pair wise comparison for particular criteria, the assessors shall discuss and exchange their experiences and opinions on the compared evaluation criteria. Then, each of them shall choose the importance value for the compared evaluation criteria from the cards. However, all selected values are kept private until all team members have chosen a card. Then when everyone is ready, the cards shall be revealed to the group at once. Thus, the chosen values can be observed clearly.

If consensus is reached among the assessors, the agreed importance value shall be chosen. However, when consensus is hard to be reached, the majority vote is

selected or the group may compromise on a preferred value. On the other hand, when consensus cannot be reached, and the group is not willing to choose the majority vote or compromise, then the average shall be calculated for the voted values by using geometric mean (Lai, Wong, & Cheung, 2002; Dyer & Forman, 1992). The geometric mean is used for aggregating the individual judgments (Forman & Peniwati, 1998). To calculate the geometric mean, the following formula is used:

$$GM = \sqrt[n]{a_1 * a_2 * a_3 * a_n} \quad (5.2)$$

Where

$$n = 1, 2, \dots, n$$

a_n = The importance value voted by n assessors

GM= Geometric mean for a_n

For instance, if there are five assessors in a team and each of them choose the importance value differently (5, 7, 5, 8, 9), the following calculation is performed:

$$GM = \sqrt[5]{5 * 7 * 5 * 8 * 9}$$

$$= 7$$

Accordingly, this value is accepted as the agreed importance value for a particular judgment.

A2.6.1.3: Synthesize the pair wise comparisons

Based on the pair wise comparison matrixes constructed, the weight values are calculated by using the Normalization of the Geometric Mean (NGM) is used (Akarte, Surendra, Ravi, & Rangaraj, 2001; Hsiao, 2002). By using this method, the approximation to the correct answer is higher (Coyle, 2004). Additionally, it is

statistically better and easier to calculate (Crawford & Williams, 1985). The n elements in each row are multiplied and the n^{th} root is calculated to obtain the weight values. Then, the resulting numbers are normalized. The equation for this method is provided subsequently.

$$w_i = \left(\prod_{j=1}^n a_{ij} \right)^{1/n} / \sum_{i=1}^n \left(\prod_{j=1}^n a_{ij} \right)^{\frac{1}{n}} \quad (5.3)$$

Where:

- w_i = Weight of evaluation criteria i
- i = 1,2,...., n
- j = 1,2,...., n
- a_{ij} = Pairwise comparison in matrix ij

Considering the pair wise comparison made in Table 5.12, the weight of RE is calculated as below:

$$\begin{aligned} \text{Weight for RE} &= (1*7*8*7)^{1/4} / (4.45+0.809+0.452+0.615) \\ &= 4.45 / 6.326 \\ &= 0.703 \end{aligned}$$

Table 5.13 shows the weight values obtained for the sub factors of software development.

Table 5.13

Weight Values for Sub Factors of Software Development

Software Development	RE	DES	CODE	TEST	n^{th} Root values	Weight values
RE	1	7	8	7	4.45	0.703
DES	1/7	1	3	1	0.809	0.128
CODE	1/8	1/3	1	1	0.452	0.071
TEST	1/7	1	1	1	0.615	0.097
Total					6.326	1.000

A2.6.1.4: Perform consistency analysis

As the pair wise comparison involves with subjective judgments, thus the Consistency Ratio (CR) needs to be calculated in order to eliminate the inconsistency of the judgments made. This is the advantage of using AHP, whereby the consistency of decisions can be revealed. Consequently, with the CR values, the certification decision will become more consistent. The acceptable CR value is less than 0.1 (Saaty, 1990). The CR is calculated by using the subsequent equations.

$$CR = \text{Consistency Index (CI)} / \text{Random Index (RI)} \quad (5.4)$$

Where CI is calculated using this formula:

$$CI = (\lambda_{\max} - n) / (n-1) \quad (5.5)$$

Where n = number of evaluation criteria in the matrix

λ_{\max} = the average value of consistency vectors

The RI is obtained for the appropriate value of 'n', as depicted in Table 5.14.

Table 5.14

Random Index (Saaty, 1988 as cited in Padumadasa et al., 2009)

n	1	2	3	4	5	6	7	8	9	10	11	12	13
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56

Considering the software development pair wise comparison in Table 5.12, λ_{\max} is calculated as provided subsequently.

Weight (RE)	PWC (RE)	Weight (DES)	PWC (DES)	Weight (CODE)	PWC (CODE)	Weight (TEST)	PWC (TEST)	Weighted sum vectors
0.703	1	0.128	7	0.071	8	0.097	7	2.846
	1/7		1		3		1	0.538
	1/8		1/3		1		1	0.299
	1/7		1		1		1	0.396

* PWC: Pair wise comparison of

First, the weighted sum vectors are calculated by multiplying the first weight value by the first column of the original pair wise comparison table (Refer to Table 5.12). Then, the second weight value is multiplied by the second column and so forth. Next, the values obtained are summed up over the rows. For instance, the weighted sum vector for the first row is calculated as below:

$$\begin{aligned} \text{Weighted sum vector (1}^{\text{st}} \text{ row)} &= (0.703*1) + (0.128*7) + (0.071*8) + (0.097*7) \\ &= 2.846 \end{aligned}$$

Similarly, they are 0.538, 0.299, and 0.396 for the 2nd, 3rd and 4th rows respectively.

Next, the consistency vectors are calculated by dividing the weighted sum vectors with the respective weight values obtained previously. They are depicted in Table 5.15.

Table 5.15

The Consistency Vectors

Criteria	Weighted sum vectors	Weight values	Consistency vectors
RE	2.846	0.703	4.048
DES	0.538	0.128	4.203
CODE	0.299	0.071	4.211
TEST	0.396	0.097	4.082

The average of consistency vectors are then calculated to obtain λ_{\max} , where

$$\lambda_{\max} = (4.048 + 4.203 + 4.211 + 4.082) / 4 = 4.136$$

Therefore,

$$\begin{aligned} \text{CI} &= (\lambda_{\max} - n) / (n-1) \\ &= (4.136 - 4) / (4-1) \\ &= 0.136/3 \\ &= 0.045 \end{aligned}$$

$$\begin{aligned} \text{CR} &= \text{CI} / \text{RI} \\ &= 0.045/0.90 \\ &= 0.05 \end{aligned}$$

As the CR value is less than 0.1, thus the pair wise comparison made is considered as consistent. However, if the CR value is more than 0.1, then the judgments on the criteria need to be performed again (A2.6.1.2).

A2.6.1.5: Obtain the global weight values

The weight values obtained in the previous activities are known as the local weight values. The final weight values are obtained by calculating the global weight values. If the CR value for the pair wise comparison is less than 0.1, then the global weight values can be calculated. They are obtained by multiplying the local weight value of a child by its parents' local weight values (the calculation starts from the lowest level to the first level of hierarchy tree). The equation for the global weight is provided subsequently.

$$GW_i = LW_i * \prod_{j=1}^n P_j \quad (5.6)$$

Where:

GW_i = Global weight value for i^{th} evaluation criteria

LW_i = Local weight value for i^{th} evaluation criteria

P_j = Local weight for j^{th} parents

i = 1,2.....,n

j = 1,2.....,n

For example, Equation 5.6 is used to obtain the global weight value for the completeness of requirement engineering (GW_{CompRE}) by multiplying the local weight value for completeness of requirement engineering with the local weights of its parents (requirement engineering, software development and process) as illustrated in Figure 5.7. In the same way, this calculation is performed to obtain the global weight values for other factors/sub factors/evaluation criteria. The complete local and global weight values can be obtained from Appendix M. These global weight values are the ideal weight values suggested by the ESPAC Model.

$$\begin{aligned} GW_{\text{CompRE}} &= 0.691 * 0.703 * 0.659 * 0.378 \\ &= 0.121 \end{aligned}$$

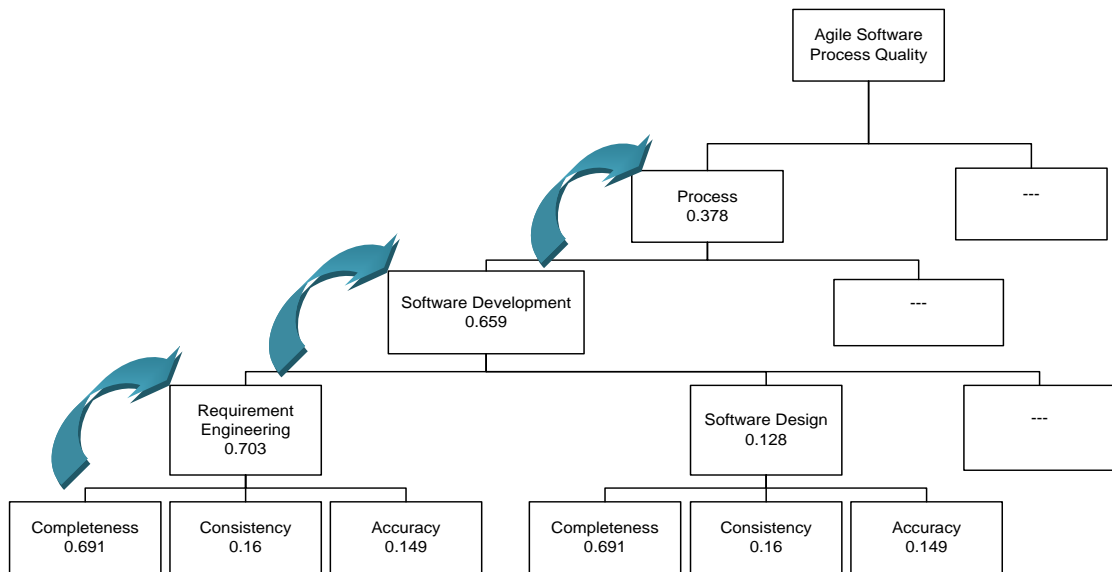


Figure 5.7. The part of hierarchy tree with local weight values

A2.6.2 Assign scores by using the assessment form

In the second activity, the scores are assigned for each assessed Agile and secure software processes. The reference standard is compared with the practices implemented by the organization. Each of these practices is assigned with the score based on the scale of *1= Never, 2= Rarely, 3= Sometimes, 4= Often, and 5= Always*. As mentioned in Section 5.3.3, the reference standard comprises of the WHATs, HOWs, relationships among WHATs and HOWs, the weight values for each evaluation criterion and the score for each evaluation criterion. This representation is used as the assessment form. Figure 5.8 shows the example of the assessment form.

Score: 1-Never 2-Rarely 3-Sometimes 4- Often 5-Always

EC	Weights	Practices	Scores
Completeness		The scope of project were identified at the beginning of project to create initial prioritized stack of requirements	1 2 3 4 5
		Customers were available on-site for face-to-face discussions during requirement elicitation or at least can be easily reached through phone or skype or teleconferencing	1 2 3 4 5
		The requirements were gathered, elaborated, analyzed and validated iteratively and incrementally	1 2 3 4 5
		The requirements were written short statements using cards or tools	1 2 3 4 5
		Requirements were prioritized and can be reprioritized by customers throughout the development	1 2 3 4 5
		The requirements were validated by customers in review meetings by using releases	1 2 3 4 5
		The product backlog and iteration backlog were produced to ensure the consistency and traceability of requirements	1 2 3 4 5
Total Score			
Consistency		Appropriate procedure is used to handle frequently changing requirements	1 2 3 4 5
		The requirements were documented by following a particular standard	1 2 3 4 5
Total Score			
Accuracy		The requirements were gathered using particular method (s)	1 2 3 4 5
		Appropriate tools were used to facilitate defining and translating the requirements	1 2 3 4 5
		Particular notation (s) was/were used to represent the requirements	1 2 3 4 5
Total Score			

**EC: Evaluation Criteria*

Figure 5.8. The example of assessment form for Agile Requirement Engineering

A2.6.3 Calculate the score for evaluation criteria and overall certification

The third activity in the synthesis technique is to calculate the score for evaluation criteria and overall certification score. This task is carried out through three (3) steps, as depicted in Figure 5.9. They are discussed further in the next sub sections.

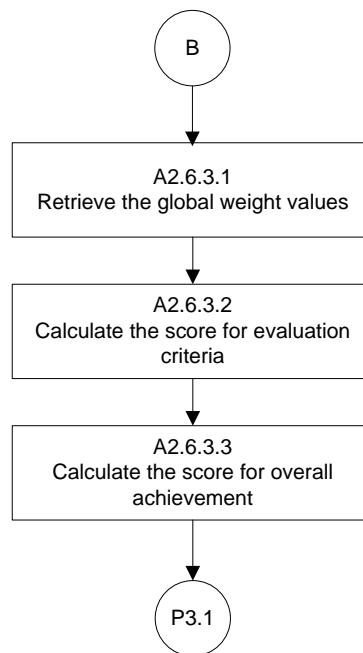


Figure 5.9. Activities for performing the assessment

A 2.6.3.1: Retrieve the global weight values

The global weight values are retrieved, either from the group discussion or the ideal weight values as suggested by the ESPAC Model.

A 2.6.3.2: Calculate the score for evaluation criteria

After the global weight values for each evaluation criteria are obtained, the scores of the evaluation criteria (the WHATS) are calculated. This is done by multiplying each global weight values of the evaluation criteria with the total score assigned for each practices (the HOWS). This equation adapts the WSM calculation (Park & Kim, 1998; Mollaghasemi, 1997). Then, the value is divided by the maximum score that can be obtained for a particular evaluation criterion to get the relative score. The maximum score is calculated by multiplying the global weight with 5 (the maximum

score for each HOWS), then multiplied with the number of HOWS. Finally, the value is aggregated by multiplying with 100 to get the percentage. The equation for this calculation is:

$$S_i = \left(GW_i * \left(\sum_{j=1}^n R_{ij} \right) \right) / (GW_i * H * 5) * 100 \quad (5.7)$$

Where:

- S_i = Score of i^{th} evaluation criteria
- GW_i = Global weight value for i^{th} evaluation criteria
- R_{ij} = The total score rating for each assessed practices in matrix ij
- H = The number of HOWS
- j = 1,2,...,n
- i = 1,2,...,n

As an example, as obtained in A2.6.1.5, the global weight value for the completeness of requirement engineering for the Agile software process is 0.121. Additionally, there is a total of seven (7) practices assessed for this evaluation criterion which obtained the score of 5, 3, 4, 5, 3, 4 and 3 respectively. Thus, the number of HOWS is 7. By using this information, the calculation is performed to obtain the score for the completeness of requirement engineering (S_{CompRE}) by using Equation 5.7, as shown accordingly. Finally, the score obtained for the completeness of requirement engineering is 77%.

$$\begin{aligned} S_{\text{CompRE}} &= (0.121 * (5+3+4+5+3+4+3)) / (0.121*7*5) * 100 \\ &= 3.267 / 4.235 * 100 \\ &= 77\% \end{aligned}$$

The same calculation is performed for the other evaluation criteria. The example of score obtained for each evaluation criterion is presented in Appendix N. They are the

scores obtained by Project C during the validation of ESPAC Model (Refer Chapter Six). By using these scores, the quality level of the completeness is determined by referring to the Achievement Index provided in Table 5.16. Thus, the achieved quality level for the completeness of requirement engineering in Agile software process is Level III (Largely Achieved).

A 2.6.3.3: Calculate the score for overall achievement

To obtain the overall achievement, the score for the root of the hierarchy tree is calculated. The calculation starts from the lowest level of the hierarchy tree to the top. In the beginning, only the scores of evaluation criteria in the lowest level are available, however the scores for their parents are not known. Thus, the score for each parent is calculated by multiplying the global weights of its children with their scores. Then, the multiplication results are summed up. The total of the sum is then divided by the total of global weight values of the children. The same calculation is performed for all parents until the root is reached. The equation for this calculation is as below:

$$S_p = \frac{\sum_{i=1}^n (GW_i * ScoreC_i)}{\sum_{i=1}^n GW_i} \quad (5.8)$$

Where:

- S_p = Parent's score
- $ScoreC_i$ = Score of i^{th} child (evaluation criteria/sub factors/factors)
- GW_i = Global weight value of i^{th} child
- i = 1,2,...,n

Figure 5.10 shows part of the hierarchy tree as provided in Appendix D. It is apparent that the requirement engineering is one of the parents; therefore the

children are the completeness, consistency and accuracy. In the beginning, only the scores of the completeness, consistency and accuracy are available (obtained from Equation 5.7). However, the score for the requirement engineering (their parent) is not known. Accordingly, to obtain the score for the requirement engineering (S_{RE}), Equation 5.8 is performed, as below:

$$S_{RE} = ((0.121*77) + (0.028*70) + (0.026*53.3)) / (0.121 + 0.028 + 0.026)$$

$$= 72.46\%$$

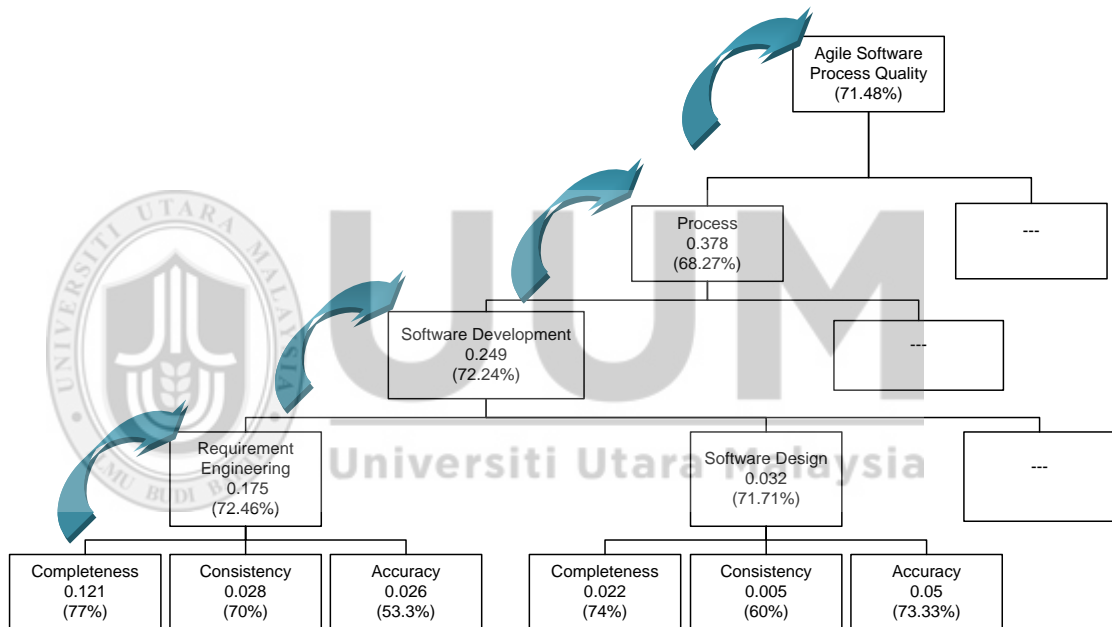


Figure 5.10. The part of hierarchy tree with score and global weight values

Likewise, the same procedure of calculation is performed to all parents in the hierarchy tree until the root is reached. The root's score indicates the overall certification score obtained. This score is then aligned with the Achievement Index (Refer Table 5.16) to determine the certification level. As per the example given, the certification level achieved for 71.48% (root's score) is Level III (Largely

Achieved). The complete global weight values and scores for each parent and child are provided in Appendix N.

5.3.7 The Achievement Index

The ESPAC Model produces the Achievement Index that can be used to determine the certification outcomes: quality level, which indicates the achievement level for each evaluation criterion, and certification level, which indicates the overall achievements for the assessed Agile and secure software processes. They are elaborated in the next sub sections. The Achievement Index is adapted from the ISO/IEC 15504 (Galín, 2004; Jung, 2001) and Patel and Ramachandran (2009).

5.3.7.1 The Quality Levels

Based on the score obtained for the evaluation criteria (S_i) by using Equation 5.7, the quality level is determined. The highest quality level is ‘Fully achieved’, obtained when the S_i value is between 86% and 100%, while the lowest quality level is ‘Not Achieved’, for the S_i value between 0% and 15%. The complete Achievement Index for the quality levels are presented in Table 5.16, as well as their descriptions.

5.3.7.2 The Certification Level

The certification level is determined for the overall achievements, whereby the score is obtained from the root of the hierarchy tree (by using Equation 5.8). The value is obtained by performing calculation for S_p in Equation 5.8. The levels of

achievement are similar to the quality levels, by referring to the Achievement Index as provided in Table 5.16.

Table 5.16

The Achievement Index

Score Values	Quality/ Certification Levels	Descriptions
$86 \leq S_i/S_p \leq 100$	Level IV Fully Achieved	This level indicates a fully satisfying achievement. The software processes were implemented effectively, systematically and perfectly or almost perfectly.
$51 \leq S_i/S_p \leq 85$	Level III Largely Achieved	This level indicates a largely satisfying achievement. The software processes were implemented quite systematically. However, some software processes of low performance exist.
$16 \leq S_i/S_p \leq 50$	Level II Partially Achieved	This level indicates a partially satisfying achievement. A systematic approach has been used; however almost all of the assessed software processes were not implemented properly.
$0 \leq S_i/S_p \leq 15$	Level I Not Achieved	This level indicates unsatisfying level of achievement. The software processes were not implemented systematically and below average. The methodology usage was neglected. The software process is considered as fail to achieve its goal.

5.4 Discussions

This chapter has described the ESPAC Model, which is a software process certification model that focuses on the Agile and secure software processes. The main aim of this model is to assess the effectiveness and efficiency of software process which concerns on both processes. By doing so, the quality of produced software can be revealed, as the quality of product is influenced by the quality of

process used to develop it (Humphrey, 1989; Deming, 1982). The assessment is conducted for the software that has been completed and ready to be delivered to customer.

The ESPAC Model is constructed by considering the outcomes from the theoretical and exploratory study, as well as considering the Evaluation Theory as the base theory. The SPAC Model is referred as the base model, as well as other existing software process standards and models such as CMMI Version 1.3, ISO/IEC 15504, ISO/IEC 21827 and ISO/IEC 27001. Furthermore, the main outcomes from the exploratory study which are the Agile and secure software practices and the characteristics of people who involve in both software processes are used to construct the reference standard. Also, the data gathering techniques are considered. There are seven (7) components in the model, which are adapted based on the Evaluation Theory: the target, evaluation criteria, reference standard, data gathering techniques, assessment process, synthesis technique and Achievement Index.

Basically, the major improvements are made on the reference standard and the synthesis technique. The reference standards of the existing software process certification models and standards focus more on the conventional software process. However in today's business environment, incorporating Agile and secure software processes are essential in order to produce high quality software. Therefore, these two software processes are included in the reference standard of ESPAC Model. The reference standard which consists of the evaluation criteria, scoreboard, Agile and secure software practices, weight values and total score are organized by adapting

the Quality Function Deployment (QFD) approach (Cohen, 1995; Zultner, 1992). By using this approach, the reference standard is organized systematically.

Additionally, the existing software process certification models and standards lack an appropriate synthesis technique in the certification process. This is because they do not include weight values in their assessment. However, weight value allocation is essential especially when the assessment includes multiple criteria and involves qualitative data. Consequently, the AHP technique is adapted in the synthesis technique to provide better quality and consistency on the certification decision made. Also, the WSM is adapted to calculate the score obtained for the certification.

Among the data gathering techniques included in the model are document reviews, interviews and observations. Using multiple approaches gives better understanding and confirmation on the assessment made. Furthermore, the assessment method is proposed as collaborative self-assessment. This means that the assessment team is comprised of software practitioners from other software development team from the same organization and a representative from the assessed team. This approach has several benefits such as cost-effective, accelerates the assessment process, encourages ideas exchange between assessment team members and produces unbiased certification results.

There are three phases in the assessment process, which are pre-assessment, assessment and post-assessment. Each of them has several processes and at least one activity. The outcomes of the certification model are the quality levels, which

indicates the achievement of each evaluation criterion and certification level, which indicates the overall achievement for the assessed software process. At the end of the certification process, a technical report is produced and presented to the top management and assessed team for future reference and improvement.

5.5 Summary

As a conclusion, this chapter has discussed the development of an enhanced software process certification model, which is the ESPAC Model. The proposed model has addressed the research problems, by including the Agile and secure software processes which are needed to produce high quality software in today's business environment as the reference standard. Furthermore, the synthesis technique has been improved by including AHP for determining weights of the evaluation criteria. The model is constructed based on the outcomes from the theoretical and exploratory studies, besides adapting the Evaluation Theory as the base theory. There are seven (7) components in the proposed model. By having this model, the assessment and certification can be performed in wider perspectives and matches the current business needs. Additionally, the quality and consistency of certification decision is improved. The next chapter elaborates on the evaluation of the ESPAC Model.

CHAPTER SIX

ESPAC MODEL EVALUATION

6.1 Introduction

The process of evaluating the proposed model was carried out by performing the verification and validation stages. The verification stage was performed by experts from academic field, who are the knowledge experts as well as software practitioners, who are the domain experts. Furthermore, the validation was carried out by the domain experts. The verification and validation which involved the domain experts were performed through focus group discussion. These stages are discussed further in this chapter.

This chapter starts with the discussion on verification through expert reviews in Section 6.2, continues with verification and validation through focus group in Section 6.3. Section 6.4 provides the validation results and discussions, while Section 6.5 ends the chapter with the summary.

6.2 Verification through Expert Reviews

Expert reviews have been accepted as a significant way to detect and remove defects (Komuro & Komoda, 2008; Wiegers, 2002). Therefore, this study adapted this technique for the verification process. Verification is intended to ensure that the proposed model conforms to its specification (Sommerville, 2007) and all required components are present in the right quantity (Chemuturi, 2011). The two main issues that need to be verified in this study are the correctness of the AHP technique

(Goerigk & Hoffmann, 1999; Moody, 1998) and the comprehensiveness, understandability, accurateness and organization of the factors, sub factors and Agile as well as the secure software processes (Al Tarawneh, 2014; Behkamal, Kahani, & Akbari, 2009; Kunda, 2003). Therefore, the potential experts who had knowledge and experience in MCDM, focusing on AHP, as well as those in software engineering, which focus on Agile and secure software processes were identified. They were chosen by following the characteristics of experts as suggested by Rogers and Lopez (2002) and Hallowell and Gambatese (2010), which is discussed in Chapter Three.

6.2.1 Experts for AHP Technique Verification

Seven experts of AHP were identified and contacted through e-mails. Three of them were willing to verify the technique, which is considered sufficient (Schneiderman, 1998; Nielsen & Molich, 1990). Consequently, appointments were made and face to face meetings were held. Basically, the following activities were conducted during the review sessions:

1. Researcher presented the overview of the study and the steps for performing the AHP to the experts.
2. The experts reviewed the steps of implementing the AHP and obtaining the weight values which were also presented in the related documents. The experts had the opportunity to ask questions to the researcher for further clarification.
3. The experts gave feedbacks by filling in the verification form.
4. Researcher then updated the calculation as suggested by the experts.

6.2.2 Experts for the Agile and Secure Software Processes

Eight knowledge experts from the software engineering field were identified as the potential experts. However, only four of them were willing to verify the Agile and secure software processes. Invitations to become experts for the study were sent through e-mails. The related documents were then sent to the experts who agreed to verify the Agile and secure software processes. Feedbacks were provided by them either through e-mails or face-to-face meeting. Unfortunately, out of the four experts, only three sent their feedbacks, which is sufficient for the purpose of expert review (Schneiderman, 1998; Nielsen & Molich, 1990). The following are activities involved during the expert review process:

1. Researcher presented the overview of the study or explained it through e-mail.
2. The experts would then review the factors, sub factors and the Agile and secure software processes.
3. The experts gave their comments by filling in the verification form.
4. The researcher would then update the software processes based on the experts' comments.

Besides the knowledge experts, this study also incorporated the domain experts from software industry since they are the potential users of the model and can give feedbacks based on their practices in the real world projects. Seven domain experts verified the factors, sub factors and the Agile and secure software processes included in the proposed model through focus group discussion. Table 6.1 summarizes the knowledge experts' background. The background about domain experts and the

activities related to the review are discussed in Section 6.3, since they verified the Agile and secure software processes through focus group discussion.

Table 6.1

Experts' Background

	ID	Qualifications	Expertise	Years of Experience	Institutions
AHP	Expert A	Ph. D	MCDM , Analytic Hierarchy Process, Analytic Network Process	19 years	International Islamic Universiti of Malaysia
	Expert B	Ph. D	MCDM, Performance Measurement	16 years	Universiti Utara Malaysia
	Expert C	Ph. D	MCDM, Decision making problems involving uncertainty	21 years	International Islamic Universiti of Malaysia
Agile and secure software processes	Expert D	Ph. D	Agile software development, data mining, empirical software engineering	12 years	Universiti Utara Malaysia
	Expert E	Ph. D	Requirements Engineering, Component-based Software Engineering, Modeling	14 years	International Islamic Universiti of Malaysia
	Expert F	Ph. D	Software maintenance, program comprehension, information seeking, programmers behavior	13 years	Universiti Putera Malaysia

6.2.3 Results for the AHP Technique Verification

The AHP technique was verified by ensuring the correctness of performing it, which is adapted from Goerigk and Hoffmann (1999) and Moody (1998). Correctness refers to whether the AHP technique which was performed in the study conforms to

the steps of AHP implementation. It starts from creating the hierarchy tree and finally ends with obtaining the weight values. There are five steps in the AHP implementation, as listed in Table 6.2. All of the experts agreed that the AHP technique was implemented correctly. Refer Appendix G for the verification form.

Table 6.2

Results for the AHP Verification

Steps	Expert A	Expert B	Expert C
1.The criteria have been arranged in the hierarchy trees correctly	Agree	Agree	Agree
2.The judgments of pair wise comparisons have been performed correctly	Agree	Agree	Agree
3.The pair wise comparisons have been synthesized correctly	Agree	Agree	Agree
4.The consistency of judgments have been analyzed correctly	Agree	Agree	Agree
5.The weight values have been obtained correctly	Agree	Agree	Agree
Overall comments:			
Expert A: The researcher is on the right track			
Expert B: The researcher is on the right track, however, different calculation technique can be used, for example by using eigen value method			
Expert C: The researcher has implemented the AHP technique correctly.			

6.2.4 Results for the Factors, Sub Factors and the Agile and Secure Software Processes

The factors, sub factors and the Agile and secure software processes were verified based on their comprehensiveness, understandability, accurateness and organization. These criteria are adapted from previous studies (Al Tarawneh, 2014; Behkamal, Kahani, & Akbari, 2009; Kunda, 2003). The descriptions of these criteria are as

listed in Table 6.3. The experts provided their feedbacks by filling in the checklist form (Refer Appendix H).

Table 6.3

Descriptions of Verification Criteria

Criteria	Descriptions
Comprehensiveness	This criterion shows that the required factors, sub factors and software processes for assessing the quality software process are included in the model.
Understandability	The criterion suggests that the factors, sub factors and software processes are decomposed clearly and unambiguously.
Accurateness	The criterion indicates that the factors, sub factors and software processes are adequately decomposed to achieve accurate assessment.
Organization	The criterion denotes that the factors, sub factors and software processes are organized well.

In a nutshell, all of the experts agreed that the factors, sub factors and the Agile and secure software processes are comprehensive, understandable and accurate. However, they had some comments on the organization of the sub factors and software practices. For example, Expert D suggested that the software practices should be organized based on the steps in the software development lifecycle.

Meanwhile, Expert E found that there were redundancies of software practices in the Agile and secure software processes. On the other hand, Expert F concluded that the sub factors that are similar between the Agile and secure software processes should be placed only once. In addition, some of terms or words used in the processes were inappropriate. For example, the term ‘release’ is more appropriate to be used in the Agile environment instead of ‘prototyping’. Hence, the Agile and secure software

practices were updated accordingly as suggested. Table 6.4 recapitulates the comments from the knowledge experts.

Table 6.4

Summary of Experts' Comments

Criteria	Expert D	Expert E	Expert F
Comprehensive-ness	Agreed	Agreed	Agreed
Understandable	Agreed	Agreed	Agreed
Accurate	Agreed	Agreed	Agreed
Well-organized	-May organize the software processes based on the steps in the software development lifecycle. -Some of the Agile software processes are included in the secure software process, and vice versa.	-Some of the software processes are redundant in the Agile and secure software processes.	-The similar factors among the Agile and secure software processes should be placed only once. For example, the tools and techniques as well as the resource management and training. -Some processes used inappropriate word, such as 'prototyping' instead of 'release'.

Based on the comments from the experts, the Agile and secure software processes were reorganized and updated. The actions taken are summarized in Table 6.5.

Table 6.5

The Reorganizing and Updating Actions

Descriptions	Problems/Actions taken
Resource management, training, tools and techniques, standards and procedure, technical skills, knowledge, experience, and environment (safety and comfort)	The assessed software practices for these sub factors and evaluation criteria are the same for the Agile and secure software processes/ Assessed only once
The requirements are written on cards in short	It is not necessary to use cards for user

statements	stories. It can also use the tools/ Updated
The requirements are validated by customers in the review meetings by using prototypes	'Prototype' is not appropriate with the Agile environment/ Updated
The features with high priorities are delivered first	Priority is considered in planning, not during coding/ Moved to Project Management
The schedule estimation is done by the developer who is going to implement a particular user story	The estimation is done by the team, not individually/ Updated
The development team was enabled to re-estimate the time and velocity of user stories	This software process is more related to planning / Moved to Project Management
The deliverable documentation were produced late	Redundant with documentation/ Removed
A common understanding about the security needs was reached among all applicable parties, including the customers.	Redundant with team commitment/ Removed

6.3 Verification and Validation through Focus Group

The ESPAC Model was validated through focus group, which was attended by the domain experts from various software organizations. Through the focus group discussion, the experts verified the Agile and secure software processes, performed the AHP technique to obtain the weight values for the evaluation criteria and validated the proposed model. The next sub sections discuss the implementation of the focus group, which constitutes the planning, executing, along with analyzing the data and reporting the result of the focus group (Martakis & Daneva, 2013; Daneva & Ahituv, 2011; Mazza & Berre, 2007; Morgan, 1998; Krueger, 1994).

6.3.1 Plan the Focus Group

A thorough planning is needed to effectively implement the focus group. The focus group planning involved five (5) activities which are defining the objectives, identifying the participants, scheduling the meeting, preparing the materials for the focus group and sending reminder to the participants. These activities are further elaborated in the next sub sections.

6.3.1.1 Define the Objectives of the Focus Group

Basically the objectives of the focus group are threefold. In particular, the objectives are as follows:

1. To obtain the weight values for the evaluation criteria by using AHP technique.
2. To verify the comprehensiveness, accuracy, understandability and organization of the factors, sub factors and Agile and secure software processes that are included in the software process certification model.
3. To validate the model based on its practicality to be implemented in the real environment and satisfaction (gain, interface and task support satisfactions)

6.3.1.2 Participants Identification and Recruitment

The participants were selected by using purposive sampling. They were chosen based on several common characteristics that they have (Liamputtong, 2011) such as: 1) Agile software practitioners, 2) work in Kuala Lumpur or nearby area 3) have experience in secure software process, 4) have software development experience for more than 3 years. Initially, the respondents of the exploratory study which was previously conducted in this study were approached through telephone and E-mails.

Unfortunately, only one of them was willing to participate. It was hard to get participation among the software practitioners as they were busy.

Since the focus group needs a range of six to ten participants (Morgan, 1998; Powel & Single, 1996; Krueger, 1994), alternative ways were used to gather the participants. The potential participants were approached through the places they tend to assemble, either virtually or actual meetings (Stewart, Shamdasani & Rook, 2007). Consequently, they were also approached through social networking groups such as the Agile Malaysia group in Face book and Scrum Malaysia Community by Google. The invitation was posted on the wall of these groups, as well as randomly emailed them on a personal basis. In addition, they were approached face-to-face during the Agile Symposium in Melaka, Scrum Master Training in Kuala Lumpur, APAC Agile & Lean Conference 2013 organized by Intel Malaysia (Penang Campus) and Agile Malaysia group monthly meet up which was held in one of the software development companies in Putrajaya. Brochures relaying about the focus group were distributed to them during these meet ups (Refer Appendix C). By using these various approaches, finally eight participants agreed to participate.

6.3.1.3 Meeting Scheduling

The suitable meeting place was identified and booked based on the guidelines provided by Stewart et al. (2007), Powel and Single (1996) and Krueger (1994). The place which was central for all of the participants was chosen, which is one of the hotels in Kuala Lumpur that provide meeting room facilities such as discussion table and LCD projector. Also, the meeting place is considered neutral, as it did not have

special significance to the participants and no bearing to the objectives of the focus group. Additionally, it provided pleasant and comfort environment for the participants. Furthermore, the focus group was scheduled on Saturday, which was convenient for the participants.

6.3.1.4 Preparation of the Focus Group Interview Guide and Materials

Prior to conducting the focus group, the interview guide was developed. The principles of preparing interview guides were adapted, whereby the discussion was planned to be started by general topic, which is the introduction of the study. Then, the next agenda was to obtain the weight values, continued with the verification and validation of the proposed model. These key sequential activities were determined based on their relative importance to the study, as suggested by the second principle of preparing interview guide (Stewart et al., 2007). Additionally, the materials that were used during the focus group session were prepared, namely the presentation slides, documents for the participants and numerical cards for the voting process (obtain the weight values by using AHP technique). In addition, certificates of participation were also prepared.

6.3.1.5 Remind the Participants

One day before the focus group was conducted, the participants were reminded about the session and their attendance was confirmed. This was to ensure that they would not miss the session as well as to make them feel their importance in attending the session (Stewart et al., 2007).

6.3.2 Conduct the Focus Group

The focus group was conducted on the scheduled day and time. However, one of the participants who agreed to come could not attend the session. Thus, only seven of participants turned up to attend the session, which is considered sufficient number of participants for a focus group (Morgan, 1998; Powell & Single, 1996; Krueger, 1994). In order to conduct the focus group, the guideline provided by Stewart et al. (2007), Powel and Single (1996) and Krueger (1994) were followed.

Upon arrival at the meeting room, the participants were greeted and a friendly contact was established in order to create rapport. This was done by having an informal conversation among the participants and moderators before the formal discussion begins. They were also served with refreshments. This was intended to make the participants feel comfortable and relaxed. On top of that, this enabled the moderators and participants to get to know each other.

In the formal session, the participants were seated in a U-Shaped discussion table to facilitate interactions. They were provided with the materials that needed for the session. Figure 6.1 exhibits the settings of the meeting place. Once all of the participants were seated, they were welcomed with a speech from the moderator. Then, the moderator introduced herself and the assistant moderators. In the same manner, each of the participants introduced themselves to the group. This is a useful way to build rapport and a good sense of building group cohesion (Liamputtong, 2011). Then, they were briefed about the objectives of the focus group. They were encouraged to express their experience and points of view freely and spontaneously.

The participants were also reminded that the data gathered from them will be confidential and only will be used strictly for the study purposes. Then, they were briefed about the ESPAC Model and the AHP Technique. During this presentation, the participants started to interact freely by clarifying the issues that they were not clear from the presentation.



Figure 6.1. The meeting place setting

The participants worked in three stages, first was to obtain the weight values for evaluation criteria. The second stage was to verify the Agile and secure software processes and the third stage was to validate the ESPAC Model. The discussion took approximately three hours to be completed. This duration is acceptable, even though the common duration is one to two hours (Liamputtong, 2011). In between the session, the participants were provided with lunch. Furthermore, at the end of the session, certificate of participation were presented to them. The activities involved during the focus group discussion are further discussed in the following sub sections.

6.3.2.1 Obtain the Weight Values for Evaluation Criteria

To obtain the weight values for the evaluation criteria, the group AHP technique was implemented, as the decisions were made in a group. The planning poker (Dyba, Dingsoyr, & Moe, 2014; Mahnic & Hovelja, 2012) which is used in Agile environment is adapted to simplify the AHP process. Each of the participants was given a set of card which contains numbers from 1 to 9. They represent the importance values which are used for making judgments of pair wise comparisons in AHP. The participants were also provided with a list of pair wise comparisons that need to be performed (Refer Appendix F). Referring to the document, the moderator raised each of the pair wise comparisons one by one, while probing open ended questions to the participants on their opinion about the evaluation criteria being compared. The participants discussed and exchanged their experiences and opinions. Then, they chose the importance value for the compared evaluation criteria from the cards. All selected values are kept private until all team members have chosen a card. Then once everyone is ready, the cards are revealed to the group simultaneously.

When consensus was reached among the participants, the agreed importance value was chosen. Nevertheless, when neither consensus could be reached, nor majority vote or compromise can be reached, then the average was calculated by using the geometric mean. The judgments made on the pair wise comparisons were then keyed-in the Excel file for each pair wise comparisons. The AHP technique was implemented to obtain the weight values. When any of the pair wise comparisons was not consistent, the process was repeated again until a consistent judgment was

obtained. The detailed steps of the AHP technique implementation is provided in Section 5.3.6. Figure 6.2 shows the participants raising the cards with their preferred values for a pair wise comparison.



Figure 6.2. The selection of the pair wise comparison value by the participants

6.3.2.2 The Agile and Secure Software Processes Verification

The second stage of the focus group was to verify the Agile and secure software processes. The participants were instructed to fill in the forms for verification. Appendix H presents the example of the form for the Agile requirement engineering process. The participants checked the Agile and secure software practices included in the model one by one. Then, they verified whether the Agile and secure software practices are comprehensive, understandable, accurate and well-organized (Refer Appendix I).

6.3.2.3 The ESPAC Model Validation

The third stage of the focus group was to validate the ESPAC Model. During this stage, the participants were asked to implement the proposed model by assessing one of the projects that they have completed. The assessment form was provided for them to assign the score for each software process. Example of this form is provided in Appendix H. Based on their experience in the project, they self- assess the project and assigned the score for each of the software processes in the proposed model. Figure 6.3 presents the verifying and validating process by the domain experts.



Figure 6.3. The process of verifying and validating the ESPAC model

6.3.3 Data Analysis and Results Reporting

After completing the focus group session with the participants, the data obtained from the focus group were analyzed. The verification of the factors, sub factors and

Agile and secure software processes was examined. Additionally, the total score for the assessment and certification exercise were calculated. Then, the quality levels as well as the certification level for each project were obtained. The outcomes were then reported in the technical reports by representing them in tables and charts. These technical reports were then emailed to the participants.

Based on the technical report provided, the participants were asked to validate whether they were satisfied with the ESPAC Model and determine its practicality to be implemented in the real environment. The participants then provided their feedbacks either through emails or phone calls. They filled in the validation form to provide the feedbacks (Refer Appendix J). The findings are discussed in the next sub sections.

6.3.4 The Focus Group Discussion Findings

This section describes the findings from the implementation of the ESPAC Model among the focus group participants. The participants' background is provided together with the weight values obtained for each evaluation criteria. Then, the verification results as well as the assessment and certification results are presented.

i. Participants' Background

Participants of the focus group included seven Agile practitioners from different organizations around the Kuala Lumpur area. They were the team leaders, Scrum Master, Application Lifecycle Program Manager, architect and programmers. All of them had experience in software development for more than three years. Basically

the participants work in private software development companies. Four out of the seven participants work in large companies, whereby the employees are more than 250 people. All of the participants had experience in Agile and secure software processes. Majority of them had three to five years' experience in Agile. The most popular Agile method used by them seemed to be Scrum and Extreme Programming (XP). In addition, Feature Driven Development (FDD), Test Drive Development (TDD), Agile Modeling (AM) as well as Lean Programming were also used by the participants. Table 6.6 presents the anonymized overview of the participants.

Table 6.6

Anonymized Overview of the Participants

ID	Positions	Size of Organization	Years of Software Development Experience	Years of Agile Experience	Agile Methods
A	Team Leader	>250	11-20	1-2	FDD
B	Scrum Master	>250	6-10	3-5	Scrum, XP, AM, Lean
C	Programmer	51-250	6-10	3-5	Scrum, XP
D	Architect	>250	11-20	>5	Scrum, XP
E	Team Leader	51-250	6-10	1-2	Scrum
F	Programmer	>250	6-10	3-5	Scrum, XP
G	Application Lifecycle Manager	20-50	11-20	3-5	Scrum, TDD

ii. The weight values obtained

As discussed in Section 6.3.2.1, the weight values were obtained through the planning poker. The outcomes from the discussion are the ideal weight values for the evaluation criteria of the ESPAC Model. The ideal weight values obtained can be referred to Appendix M. These ideal weight values are suggested for the potential users of ESPAC Model, whereby they can use these values during the assessment. However, the values are flexible. They can decide whether to use the suggested ideal weight values or obtained the values of their own by performing the AHP technique (Refer Section 5.3.6).

iii. Verification results

As mentioned before, the factors, sub factors and Agile and secure software processes were verified by the knowledge and domain experts. This section provides results from the verification performed by the domain experts. Majority of the software processes included in the model were accepted by them. They agreed that the Agile and secure software processes included in the model are comprehensive, understandable, accurate and well-organized. However, they gave suggestion of software processes that need to be updated and reorganized in the proposed model as well. These are described in Table 6.7.

Table 6.7

The Suggested Software Processes

Descriptions	Issues / Actions taken
Customers are available on-site for face-to-face discussions during requirement elicitation	Sometimes customers are not able to be on-site, however they can be reached through phone calls or Skype or teleconferencing / Updated
Able to communicate in various languages	This is not necessary for all projects / Removed
Budget Estimation	The budget estimation is performed by the customer, not the team. It is only communicated to the team clearly / Updated

iv. Assessment and certification results

The assessment and certification results for the seven projects are discussed by focusing on the background of the projects, the results obtained from the assessment and certification exercise and the certification levels achieved by the projects. They are elaborated further in Appendix L.

6.4 Validation Results and Discussions

After implementing the ESPAC Model and obtaining the results, the domain experts are satisfied with it and agree that the proposed model is practical to be implemented in the real world environment. They validated the proposed model based on a predefined set of criteria, which are adapted from previous studies (Al Tarawneh, 2014; Kunda, 2003). These criteria include gain, interface and task support satisfactions. Each of these criteria was assessed based on a set of variables. They are discussed further in the next sub sections.

6.4.1 Gain Satisfaction

Among the variables that were assessed for gain satisfaction are perceived usefulness, decision support satisfaction, current assessment initiatives comparison, cost-effectiveness, clarity and task appropriateness. These are described based on the comments from the participants. Table 6.8 provides the results.

Table 6.8

The Results of Evaluation for Gain Satisfaction Criteria

Variables	Results of evaluation/ explanation
Perceived usefulness	The participants pointed out that the proposed model is useful for their working environment. By having this model, they can know the current quality level of their software process. In addition, the outcomes from the assessment and certification process can guide them in improving their software process. Furthermore, Participant C and E appreciated the secure software process included in the reference standard, as nowadays security aspect need to be addressed during the development to ensure that the end product is secured. In addition, the software processes cover from start to the end of software development. Besides these benefits, Participant F added that this model can guide new organizations which plan to adapt the Agile software process.
Decision support satisfaction	The participants were satisfied with the decision that they made for the weight values, whereby it can reduce individual bias since the weight values were obtained in a team. The weight values are important as it will influence the score obtained for the certification. Furthermore, Participant F highlighted that the model allows equal participation since the decision on the weight values are obtained in a team. Thus, it supports the Agile environment because it allows interaction and active team, which is the core value of Agile. On the other hand, Participant C added that conflicts can be resolved by reaching consensus when deciding on the weight values in a team.

Comparison with current assessment initiatives	Participant A compared the proposed model to the ISO 9001: 2008 performed in his organization. It is found that the proposed model focuses on the software process, which is more suitable for software development organizations. Conversely, Participants B, C and G admitted that they did not perform assessment on their software process. Therefore the proposed model will be beneficial for them.
Cost-effectiveness	The proposed model is deemed as very cost-effective, as compared to other certification models such as ISO 9000: 2008 which costs a lot of money, as highlighted by Participant A. This is because the assessment and certification is performed by the software practitioners in the same organization. Therefore, it does not require any payment for other assessing organizations.
Clarity	The assessment phases and the activities provided in the proposed model are found to be very clear to the participants, whereby the phases clearly presents the activities that need to be performed, besides the data gathering techniques that can be used to perform the activities. Additionally, Participant F emphasized that the model clearly defines the certification levels, which gives guidance on the level of software processes.
Task appropriateness	The participants agreed that the proposed model is appropriate for assessing and certifying the software process systematically and effectively because it considers the approaches that are essential in order to produce high quality software in today's business environment: Agile and secure software processes. In addition, Participant C highlighted that the assessment and certification by using proposed model can be performed to any type of project. Furthermore, Participant F added that by having Agile and secure software processes as the reference standard in isolation, the assessment and certification can be performed based on the needs of the project.

6.4.2 Interface Satisfaction

The interface satisfaction was assessed based on five variables, which are perceived ease of use, internally consistent, organization (well organized), appropriate for audience and presentation (readable and usable format). The results of the evaluation are provided based on the participants' thoughts, as depicted in Table 6.9.

Table 6.9

The Results of Evaluation for Interface Satisfaction Criteria

Variables	Results of evaluation/explanation
Perceived ease of use	According to the responses of the participants, the proposed model was perceived as easy to be used because it uses a well-defined processes, activities, and techniques.
Internally consistent	The participants found out that the proposed model is internally consistent, mainly because the components complement each other. In particular, it starts with forming the assessment team, then analyzing the candidate project to ensure that it achieves the minimum score for performing the certification assessment. The process is then continued with proper planning and preparing the assessment team and assessment conduct, whereby the documents that will be reviewed are determined, the persons who will be interviewed are selected and the logistics are arranged. The assessment is performed after checking the availability of the participants' assessment. In addition, the AHP technique is performed in a group for obtaining consensus among the assessment team members on the weight values (optional).
Organization (Well organized)	The model is found to be well organized and structured where the sequence of the assessment processes and activities are organized in a clear and understandable manner.
Appropriate for audience	The proposed model is found to be appropriate for audience. Participant G added that the assessment could be performed faster and easier since it is

performed by a representative of the assessed team because the implementation of the project is already understood by the representative. This facilitates the understanding of the software processes performed among the assessment team. Furthermore, Participant F added that the proposed model also could be used as a guide for organizations which would like to adopt Agile software process, since it provides the best practices of Agile software process.

Presentation The proposed model is found to produce the results in a readable and (readable and useful format, which is the technical report. Participant F added, by having useful format) the technical report, it acts like a guidance for the team to improve their software process in upcoming projects.

6.4.3 Task Support Satisfaction

Ability to produce expected result, ability to produce relevant results, ability to produce usable results, completeness, ease of implementation and understandability (easy to understand) were the variables used to assess the task support satisfaction criterion. Table 6.10 describes the results of the evaluation.

Table 6.10

The Results of Evaluation for Task Support Satisfaction Criteria

Variables	Results of evaluation/explanation
Ability to produce expected results	The proposed model is able to produce expected results. The participants indicated that they were satisfied with the quality levels and certification level obtained for their projects. Participant D added that he was satisfied with the result obtained as the assessment used various techniques for assessment: document review, interview and observation. By doing so, the assessed software processes can be understood better before appropriate score can be assigned.

Ability to produce relevant results	The participants highlighted that the five factors which are assessed in the proposed model were sufficient to produce relevant results. Participant A stressed that the proposed model does not only consider the software development factor, it also included other factors that have influence on the quality of software process. Thus, the assessment is comprehensive enough. In addition, Participant F added that the proposed model assesses the core values and principles of Agile, consequently the results are relevant.
Ability to produce usable results	The proposed model is able to produce usable results. From the quality levels obtained, the participants were able to identify which software processes need to be improved. Furthermore, with the certification level obtained, they could know how well they performed the software processes, as stated by Participants C and D. Additionally, Participant F pointed out that by having the proposed model, the team could understand and perform the best practices in Agile software process.
Completeness	The proposed model is found to be adequate and sufficient in assessing the Agile and secure software processes in the real world environment. The participants indicated that the evaluation criteria used in the proposed model are sufficient for assessing the quality of both software processes.
Ease of implementation	The participants indicated that the proposed model is easy to be implemented, as it provides a series of activities as a guideline that can be followed easily. Additionally, the technique for obtaining the weight values is also explained with examples which can be understood easier.
Understandability (easy to understand)	The proposed model is found to be readable and understandable. The participants highlighted that the assessment phases and their activities are organized well, which leads to easy understanding. Moreover, as stated by Participant C, the steps to implement the AHP technique are easy to be understood and applied.

In a nutshell, based on the feedbacks from the domain experts, the proposed ESPAC Model gained satisfaction from them and deemed as practical to be implemented in the real world projects. Besides assessing the current level of software process being performed by an organization, this model also supports continuous improvements.

6.5 Summary

This chapter has discussed the evaluation of the proposed ESPAC Model, which consists of verification and validation stages. Verification was performed on the AHP technique and the Agile and secure software processes by the knowledge and domain experts. Based on the feedbacks, the ESPAC Model was improved. For the validation stage, a focus group discussion was conducted among seven domain experts. They implemented the ESPAC Model by assessing one of their completed projects. Based on the assessment, the certification results were obtained and presented in technical reports. Then, the experts validated the proposed model. In this case, their feedbacks indicated that they are satisfied with the proposed model, and suggested that the model is practical to be implemented in the real world environment. Furthermore, during the focus group discussion, the domain experts agreed upon the ideal weight values for evaluation criteria in the ESPAC Model. These ideal weight values can be used by the potential users of ESPAC Model during the assessment or can be changed based on their preferences. The next chapter concludes the study by highlighting the contributions based on the achieved objectives. Furthermore, the limitations and future directions of the study are also provided.

CHAPTER SEVEN

CONCLUSIONS

7.1 Introduction

This chapter concludes the study reported in this thesis. The discussion starts by recapitulating the study in Section 7.2, continues with the contributions in Section 7.3. The limitations and future directions of the study are described in Section 7.4. The chapter ends with the conclusions in Section 7.5.

7.2 Study Recapitulation

The main aim of this study is to enhance the software process assessment and certification model by addressing the Agile and secure software processes as well as improving the synthesis technique used in the software certification. This aim was achieved through four objectives which have been defined earlier in Section 1.4. The study is recapitulated based on these objectives accordingly.

Objective 1: To investigate the current practices of software process certification in relation to Agile and secure software process.

This objective was achieved through the exploratory study as discussed thoroughly in Chapter Four. This study has drawn attention to the current practices of software certification which relates to the Agile and secure software processes being implemented by Malaysian software practitioners. The outcome from the study revealed the significance of both software processes in today's business environment.

However, when further investigated their current practices, it was found that the respondents only followed the best practices of Agile and secure software processes occasionally. As a result, they faced a lot of problems in the quality of software that they produced. Conversely, towards producing high quality software within budget and schedule, the software process must be performed effectively and efficiently, by following the best practices. Furthermore, the respondents gave high consideration to most of the Agile and secure software processes which can influence the quality of software as well as the characteristics of the people involved in both software processes which are included in the study. Accordingly, those processes and characteristics were included as the reference standard of the ESPAC Model.

Additionally, the needs of software certification in software industry have been revealed. However, most of the respondents do not implement any standards regardless of their familiarity with the Agile and secure software processes. On the other hand, the use of standard is vital towards ensuring that the software process is implemented correctly throughout the organization. This explains that the software process is implemented as 'ad-hoc', without considering formal procedures and monitoring. Without proper monitoring on the software development process, the quality of produced software can be low. Consequently, a mechanism to assess and certify the Agile and secure software processes is needed. Thus, this supports the needs of producing a process based software certification model which focuses on the Agile and secure software processes in this study.

Objective 2: To enhance software process certification model by including the Agile and secure software processes.

This study has successfully enhanced a software process certification model which focuses on the Agile and secure software processes, as discussed in Chapter Five. The ESPAC Model was constructed based on the Evaluation Theory and the findings from theoretical and exploratory studies (Chapter Two and Four). The components of the proposed model were determined by adapting the Evaluation Theory, which are the target, evaluation criteria, reference standard, data gathering techniques, assessment process, synthesis technique and Achievement Index. Additionally, among the findings from the theoretical study comprise of the problems and generic features of the existing software process and certification models and standard. On the other hand, findings from the exploratory study reveals the Agile and secure software processes which influence the quality of software, as well as the characteristics of people who involve in these two software processes, besides the data gathering technique in performing the assessment and certification.

The enhancements are made on the reference standard and the synthesis technique used. The ESPAC Model assesses the software process and other influencing factors such as the people involved during the development, the technology used, the project constraint and the environment by focusing on the Agile and secure software processes. Each of these factors is decomposed to at least one sub factor that is measurable, while each of them has at least one evaluation criterion. For each evaluation criterion, appropriate Agile and secure software practices that need to be performed towards achieving the specified evaluation criterion are assigned. To

organize the reference standard systematically, the Quality Function Deployment (QFD) approach is adapted. Furthermore, to improve the synthesis technique, each of the evaluation criteria is assigned with different weight values by adapting the AHP technique. Consequently, the quality of the certification results is better and more consistent as the judgments are not made arbitrarily in AHP.

This model has clearly defined three phases in the assessment process, which are pre-assessment, assessment and post-assessment, which is carried out by using the collaborative self-assessment method. During the assessment, three techniques are used to gather the data which are document review, interview and observation. At the end of assessment and certification exercise, the quality levels and certification level are produced. They are determined based on a defined Achievement Index, which comprises of four levels. The comprehensive explanation on the proposed model is presented in Chapter Five.

Objective 3: To improve the synthesis technique in software certification by using Analytic Hierarchy Process.

The study has fulfilled this objective by improving the synthesis technique used in the software process certification. This is done by including weight values for the evaluation criteria. This is necessary since the assessment in software process certification involves multiple criteria. Different evaluation criteria might have different influence on a particular project. Therefore, weight value allocation is necessary. To accomplish this objective, the Analytic Hierarchy Process technique is adapted in determining the weight values for each evaluation criterion. The weight

values are obtained through group discussion which is participated by the assessment team. The detailed explanation on the AHP implementation is provided in Chapter Five.

Objective 4: To evaluate the enhanced software process certification model by using expert reviews and focus group

This objective was fulfilled by performing the evaluation in two stages, which are verification and validation. In the verification stage, the correctness of the AHP technique and the comprehensiveness, understandability, accurateness and organization of the Agile and secure software processes were verified by the experts. Results from the verification process revealed that the AHP technique is implemented correctly. Similarly, the Agile and secure software processes are found to be comprehensive, understandable and accurate. However, some modifications were performed on the organization of the software processes to make them well-organized. More details about the verification are described in Sections 6.2 and 6.3.

The second stage is the validation, which aims to reveal the practicality of the ESPAC Model in the real environment, besides to disclose software practitioners' satisfaction. The proposed model was validated by domain experts who are the software practitioners, according to a predefined set of criteria of satisfaction such as gain, interface, and task support. Taken as a whole, the findings as described in Section 6.4 indicate that the ESPAC Model gained the experts' satisfaction and considered practical to be implemented in the real environment. Detailed discussions about the evaluation of the proposed model are presented in Chapter Six.

7.3 Contributions

This study has several implications on the theory and practice, especially in the field of Software Process Assessment and Certification and MCDM. The contributions of the study can be outlined in two categories: main and extra. They are elaborated in the next sub sections. The main contributions of the study include the ESPAC Model, the reference standard and the synthesis technique.

7.3.1 The ESPAC Model

The main contribution of this study is the ESPAC Model. It was built based on the outcomes of the theoretical and exploratory studies. The existing software process assessment and certification model only focus on the conventional software process and uses equal weight values for the synthesis technique. Accordingly, this study has overcome these shortcomings by incorporating the Agile and secure software processes in the reference standard and adapting the AHP technique for weight value allocation.

To construct the model, the Evaluation Theory is used as the base theory. It consists of six main components. However, this study has added another component which is the Achievement Index. This index is used to determine the outcomes of the ESPAC Model which are the quality levels and certification level.

Moreover, the ESPAC Model proposes an improved method to perform the assessment. Previously there are four assessment methods, which are first-party, second-party, third-party and collaborative methods. The collaborative self-

assessment method is adapted from the collaborative assessment method and self-assessment approach. This method suggests that the assessment and certification process is performed internally in an organization to ensure cost-effectiveness. However, to avoid biasness, the assessment is performed by software practitioners from other team in the organization. Additionally, to facilitate ideas exchange and accelerate the assessment process, a representative from the assessed team becomes as one of the assessment team members.

Theoretically, this study contributes to the field of Software Engineering, particularly in the Software Process Assessment and Certification by providing a software process certification model for the Agile and secure software processes. Practically, by having the proposed model, software developers are able to reveal the quality of software process being performed in their projects in broader aspects, which are Agile and secure software processes. Furthermore, this model provides proper guidelines which can be used by the assessors to perform the assessment and certification. In addition, the investors and customers can get benefit from the ESPAC Model since it provides conformance on the quality of software that they invest on.

7.3.2 The Reference Standard

The main component of the ESPAC Model is the reference standard, which is used as a benchmark for performing the assessment. The existing software process and certification models do not address the Agile and secure software processes in their reference standards. However, these software processes have become determinant

factors in producing high quality software in today's business environment. Consequently, the reference standard of ESPAC Model which comprises of software process quality factors, evaluation criteria and practices are enhanced by addressing the Agile and secure software processes. By doing so, the assessment and certification can be performed in broader aspects which is suitable for the current business environment and needs. Furthermore, the assessment and certification can be customized based on the background of the project: whether the project was developed by using Agile or conventional software process, as well as whether the project requires high safety environment.

The enhancement made on the reference standard contributes to the Software Engineering, specifically to the Software Process Assessment and Certification area. Mainly, the factors and practices of Agile and secure software processes that influence high quality software are revealed.

7.3.3 The Synthesis Technique

The synthesis technique is another essential component of the ESPAC Model. It is used to synthesize the data gathered during the assessment and obtain the score of the assessment and certification. The synthesis technique has been improved by considering the weight values during the assessment. The weight value allocation is vital since the software process assessment involves multiple criteria, whereby each of the criteria might have different influence. Consequently, the AHP technique is adapted for this purpose. It is one of the most widely used MCDM techniques. The

AHP technique is chosen since the consistency of judgment can be increased, which indirectly influence the quality of the certification decision made.

The use of the AHP improves the certification outcomes in the field of Software Process Assessment and Certification since the existing studies only used equal weights for the evaluation criteria. This study also contributed to the MCDM field as it proposes the use of AHP in the Software Process Assessment and Certification area.

Besides the main contributions, there are two (2) extra contributions in this study, as elaborated subsequently.

7.3.4 The AHP Technique Implementation through Planning Poker

The planning poker technique which is used in Agile environment to do estimation has been adapted in this study. This technique is used to determine the importance values for evaluation criteria of the ESPAC Model (Refer Section 5.3.6 and Section 6.3.2.1). By doing so, the implementation of the AHP technique is simplified and consensus can be reached easier. The use of planning poker for the AHP implementation is reasonably another contribution in the field of Software Process Assessment and Certification.

7.3.5 Utilize the QFD for the Reference Standard

The reference standard of the ESPAC Model involves numerous evaluation criteria and practices. Thus, the QFD approach has been utilized to systematically organize

them. In the reference standard, each of the evaluation criteria is allocated with the Agile or secure software practices. Thus, it facilitates the process of aligning the evaluation criterion with its appropriate practices (Refer Section 5.3.3). This approach has been used for Software Process Improvement. However, it is fairly new for the Software Process Assessment and Certification field.

7.4 Limitations and Future Directions

The limitations faced and future efforts that can be performed towards enhancing this study are discussed below:

- The ESPAC Model involves numerous evaluation criteria and practices for the assessment and certification. Thus, careful attention had to be taken in order to organize them and to perform the calculation of the certification scores, as well as to perform the AHP technique. These processes were time consuming and need extra effort to be accomplished. Thus, in future, an automated tool support can be built to facilitate the procedure and tasks that need to be performed during assessment. Additionally, the errors made by human can be reduced and the assessment and certification process can be performed faster and easier.
- The evaluation criteria and practices of the ESPAC Model are suitable for today's business environment. However, some of them become obsolete from time to time especially when the Agile and secure software processes become more mature in the software industry. Consequently, they need to be managed

and these will consume a lot of time since they involve a lot of data. Therefore, future enhancement can be made to aid the process of storing the data in a systematic repository. By having this repository, the historical data related to the assessment and certification can be managed, organized and stored safely. Furthermore, the data can be retrieved easier at any time.

- Currently the ESPAC Model only focuses on the process approach. Therefore, the ESPAC Model can be enhanced in the future by including the assessment for the software product as well. By doing so, the assessment can provide broader certification results. In addition, the outcomes from both assessments can be compared and their relation can be revealed.
- The ESPAC Model considers Agile and secure software processes as the reference standard. Nevertheless, Agile software process is criticized that it is not a reliable software process especially for big projects and produces low quality software with minimal documentation. Additionally, the outcomes from the Exploratory Study (Refer Chapter 4) indicates that most of the software practitioners do not implement proper secure software process eventhough they are facing a lot of security incidents. On the other hand, the ESPAC Model has already included the best practices of Agile and secure software processes that must be followed in order to produce high quality software. Thus, future enhancement can be made by utilizing the reference standard to produce a methodology that supports continuous software process improvement for both software processes.

7.5 Conclusions

In a nutshell, the importance of the software quality is a major concern in the software industry since low quality software may cause many problems. Because of the uncertainty on the quality of the software, the customers are becoming concerned on the ability of an organization to produce high quality software. Consequently, software certification has been found as a way to give conformance on the quality of the produced software. Nonetheless, the existing software process certification models and standards emphasize more on the product approach. Meanwhile, even though there are several studies which have been conducted in the field of process based software certification, most of the studies are intended to produce models and standards for SPI, except the SPAC Model.

However, two lacking issues need to be addressed in this model: i) the reference standard which focuses on the conventional software process and do not address the Agile and secure processes which are essential to produce high quality software in today's business environment and ii) the synthesis technique which uses equal weight values for evaluation criteria, even though the software certification involves multiple criteria.

Therefore, this study has overcome these shortcomings by making enhancement on the software processes included in the reference standard which comprises of Agile and secure software processes. Besides, the AHP technique is utilized for weight value allocation for the evaluation criteria. As a result, the software certification can

be performed in broader aspects and produces better quality and consistency of certification results.

The model was validated by seven software practitioners. The feedbacks from them revealed that the ESPAC Model is beneficial to be implemented in the real world environment. Furthermore, they highly believe that it can achieve its objectives. Taken as a whole, the software practitioners are satisfied with the proposed model.



REFERENCES

- Abbas, N., Gravell, A. M., & Wills, G. B. (2010). The impact of organization, project and governance variables on software quality and project success. *Agile Conference*, 77-86. doi: 10.1109/AGILE.2010.16
- Abdul Rahman Ahlan, Yusri Arshad, Mohd Adam Suhaimi, & Husnayati Hussin. (2010). The Malaysia IT outsourcing industry skill-sets requirements of future IT graduates. *WSEAS Transactions on Computers*, 9(7), 738-747. Retrieved from <http://www.wseas.us/e-library/transactions/computers/2010/89-744.pdf>
- Abrahamsson, P., Oza, N., & Siponen, M. T. (2010). Agile software development methods: A comparative review, *Information and Software Technology*, 50 (9-10), 833-859. doi: 10.1007/978-3-642-12575-1_3
- Abrantes, J. F., & Travassos, G. H. (2011). Common Agile practices in software processes. *International Symposium on Empirical Software Engineering and Measurement*, 355-358. doi: 10.1109/ESEM.2011.47
- Acuna, S. T., Antonio, A. D., Ferre, X., Lopez, M., & Mate, L. (2000). The software process: modeling, evaluation and improvement. In Chang, S. K. *Handbook of Software Engineering and Knowledge Engineering* (pp. 193-237). River Edge: World Scientific Publishing Co. Pte. Ltd.
- Addison, T., & Vallabh, S. (2002). Controlling software project risks - an empirical study of methods used by experienced project managers. *Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology*, 128-140.
- Afshari, A., Mojahed, M., & Yusuff, R. M. (2010). Simple additive weighting approach to personnel selection problem. *International Journal of Innovation, Management and Technology*, 1(5), 511-515. Retrieved from <http://www.ijimt.org/papers/89-M474.pdf>
- Agile Alliance. (2013). *Continuous Deployment*. Retrieved from <http://guide.Agilealliance.org/guide/cd.html>
- Agile Manifesto. (2001). Retrieved from www.Agilemanifesto.org
- Ahmed, F., Capretz, L. F., Bouktif, S., & Campbell, P. (2012). Soft skills requirements in software development jobs: a cross-cultural empirical study. *Journal of Systems and Information Technology*, 14(1), 58-81. doi: <http://dx.doi.org/10.1108/13287261211221137>
- Ai, C. Y., Md Mahbubur Rahim, & Leon, M. (2007). Understanding factors affecting success of information security risk assessment: the case of an Australian higher educational institution. *Proceedings of PACIS*. Paper 74. Retrieved from <http://aisel.aisnet.org/pacis2007/74>
- Akarte, M. M., Surendra, N. V., Ravi, B., & Rangaraj, N. (2001). Web based casting supplier evaluation using analytical hierarchy process. *Journal of the*

- Operational Research Society*, 52, 511-522. Retrieved from <http://www.jstor.org/stable/253987>
- Alinejad, A., Seif, A., & Esfandiari, N. (2013). Supplier evaluation and selection with QFD and FAHP in a pharmaceutical company. *The International Journal of Advanced Manufacturing Technology*, 68(1-4), 355-364. doi: 10.1007/s00170-013-4733-3
- Alshayeb, M. (2009). Empirical investigation of refactoring effect on software quality. *Information and Software Technology*, 51(9), 1319-1326. doi: 10.1016/j.infsof.2009.04.002
- Al-Tarawneh, F. H. (2014). *A framework for COTS software evaluation and selection for COTS mismatches handling and non-functional requirements*. (Unpublished doctoral dissertation). Universiti Utara Malaysia, Kedah, Malaysia.
- Alvaro, A., Almeida, E. S., & Meira, S. L. (2007). A software component maturity model (SCMM). *33rd EUROMICRO Conference on Software Engineering and Advanced Applications*, 83-92. doi: 10.1109/EUROMICRO.2007.11
- Ambler, S. (2014). *Agile project planning tips*. Retrieved from <http://www.ambysoft.com/essays/AgileProjectPlanning.html>
- Ambler, S. W. (2006). Survey says: Agile works in practice. *Dr. Dobb's Journal*, 31(9), 62-64. Retrieved from <http://www.drdoobs.com/architecture-and-design/survey-says-Agile-works-in-practice/191800169?pgno=1>
- Ani Liza Asnawi, Gravell, A. M., & Wills, G. B. (2012a). Emergence of Agile methods: perceptions from software practitioners in Malaysia. *AGILE India*, 30-39. doi: 10.1109/AgileIndia.2012.14
- Ani Liza Asnawi, Gravell, A. M., & Wills, G. B., (2012b). Factor analysis: investigating important aspects for Agile adoption in Malaysia. *AGILE India*, 60-63. doi: 10.1109/AgileIndia.2012.13
- Ani Liza Asnawi. (2012). *Investigating adoption of and success factors for Agile software development in Malaysia*. (Doctoral dissertation). Retrieved from http://eprints.soton.ac.uk/340352/1.hasCoversheetVersion/PhD_Thesis_Ani_Liza_Asnawi.pdf
- Ani Liza Asnawi, Gravell, A. M., & Wills, G. B. (2011). Empirical investigation on Agile methods usage: issues identified from early adopters in Malaysia. In Sillitti, A., Hazzan, O., Bache, E., & Albaladejo, X. *Agile Processes in Software Engineering and Extreme Programming* (pp. 192-207). Berlin Heidelberg: SpringerLink Verlag.
- Ares, J., Garcia, R., Juristo, N., Lopez, M., & Moreno, A. M. (2000). A more rigorous and comprehensive approach to software process assessment. *Software Process: Improvement and Practice*, 3-30. John Wiley & Sons Ltd. doi:10.1002/(SICI)1099-1670(200003)5:1<3::AID-SPIP113>3.0.CO;2-T.

- Ashbaugh, D. A. (2009). *Security software development assessing and managing security risks*. Boca Raton: CRC Press.
- Aziz Deraman, Jamaiah Yahya, Fauziah Baharom, Amalina Farhi Ahmad Fadzlah, & Abdul Razak Hamdan. (2007). Continuous quality improvement in software certification environment. *Proceedings of the International Conference on Electrical Engineering and Informatics*, 11-17.
- Azrina, S., Safura, A. D., Zuriati, I., & Nafisah, A. (2012). Skills needed by IT graduates as perceived by Malaysian IT professionals. *Proceedings of International Conference on Management, Economics and Finance*, 224-230. Retrieved from http://globalresearch.com.my/proceeding/icmef2012_proceeding/018_078_ICMEF2012_Proceeding_PG0224_0230.pdf
- Bailey, K. (2008). *Methods of social research* (4th Edition). New York: Free Press.
- Bassellier, G., & Benbasat, I. (2004). Business competence of information technology professionals: conceptual development and influence on IT-business partnerships. *MIS quarterly*, 28(4), 673-694. doi: <http://www.jstor.org/stable/25148659>
- Beck, K. (1999). Embracing change with extreme programming, *IEEE Computer*, 70-77. doi: 10.1109/2.796139
- Begel, A., & Nagappan, N. (2008). Pair programming: what's in it for me? *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 120-128. doi: 10.1145/1414004.1414026
- 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. doi: 10.1016/j.infsof.2008.08.001
- Behzadian, M., Kazemzadeh, R. B., Albadvi, A., & Aghdasi, M. (2010). PROMETHEE: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research*, 200(1), 198-215. doi: 10.1016/j.ejor.2009.01.021
- Benamati, J. S., & Mahaney, R. C. (2007). Current and future entry-level IT workforce needs in organizations. *Proceedings of the 2007 ACM SIGMIS CPR Conference on Computer Personnel Research: The Global Information Technology Workforce*, 101-104. doi: 10.1145/1235000.1235024
- Benitez, J. M., Martin, J. C., & Roman, C. (2007). Using fuzzy number for measuring quality of service in the hotel industry. *Tourism Management*, 28(2), 544-555. doi: 10.1016/j.tourman.2006.04.018
- Bernama (2013, May 6). Malaysia sixth most vulnerable to cyber crime. *The Star*. Retrieved from <http://www.thestar.com.my/News/Nation/2013/05/16/Malaysia-sixth-most-vulnerable-to-cyber-crime/>

- Blankenship, J., Bussa, M., & Millett, S. (2011). Managing Agile projects with Scrum. In Blankenship, J., Bussa, M., & Millett, S. *Pro Agile .NET Development with Scrum* (pp 13-27). Apress.
- Boehm, B. (2008). Making a difference in the software century. *Computer*, 41(3), 32-38. doi: 10.1109/MC.2008.91
- Boehm, B., & Turner, R. (2005). Management challenges to implement Agile process in traditional development organizations, *Software*, 30-39. doi: 10.1109/MS.2005.129
- Boehm, B., & Turner, R. (2003). Observations on balancing discipline and agility. *Proceedings of the Agile Development Conference*, 32-39. doi: 10.1109/ADC.2003.1231450
- Bollinger, D., & Pictet, J. (2008). Multiple criteria decision analysis of treatment and land-filling technologies for waste incineration residues. *Omega*, 36(3), 418-428. doi: 10.1016/j.omega.2006.07.008
- Bouchereau, V., & Rowlands, H. (2000). Methods and techniques to help quality function deployment (QFD). *Benchmarking: An International Journal*, 7(1), 8-20. doi: <http://www.emeraldinsight.com/doi/pdfplus/10.1108/14635770010314891>
- Brugha, C. (2004). Structure of Multi-Criteria Decision-Making. *Journal of the Operational Research Society*, 55(11), 1156-1168. doi: 10.1057/palgrave.jors.2601777
- Bulgurcu, B., Cavusoglu, H., & Benbasat, I. (2010). Information security policy compliance: an empirical study of rationality-based beliefs and information security awareness. *MIS Quarterly*, 34(3), 523-548.
- Byrnes, P., & Phillips, M. (1996). *Software capability evaluation, version 3.0, method description* (Technical Report No. CMU/SEI-96-TR-002). Retrieved from <http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA309160>
- Canfora, G., Cimitile, A., Garcia, F., Piattini, M., & Visaggio, C. A. (2007). Evaluating performances of pair designing in industry. *Journal of Systems and Software*, 80(8), 1317-1327. doi: <http://dx.doi.org/10.1016/j.jss.2006.11.004>
- Carnegie Mellon University. (2003). System Security Engineering Capability Maturity Model SSE-CMM Version 3.0.
- Cater-Steel (2004). *An evaluation of software development practice and assessment-based process improvement in small software development firms*. (Doctoral Dissertation). Retrieved from https://eprints.usq.edu.au/1256/3/Cater-Steel_PhD_%28non_USQ%29_Main_document.pdf
- Cay, T., & Uyan, M. (2013). Evaluation of reallocation criteria in land consolidation studies using the Analytic Hierarchy Process (AHP). *Land Use Policy*, 30(1), 541-548. doi: 10.1016/j.landusepol.2012.04.023

- Cerpa, N., & Verner, J. M. (2009). Why did your software fail?. *Communication of ACM*, 52(12), 130-134. doi: 10.1145/1610252.161028
- Chan, L. K., & Wu, M. L. (2005). A systematic approach to Quality Function Deployment with a full illustrative example. *Omega*, 33(2), 119-139. doi: 10.1016/j.omega.2004.03.010
- Charrate, R. N. (2001). Fair fight? Agile versus heavy methodologies, *Agile Methodologies: the great debate*. Arlington: Cutter Consortium, 2(13).
- Chemuturi, M. (2011). *Mastering software quality assurance*. Florida: J.Ross Publishing.
- Chen, C. C., Lin, M. L., Lee, Y. T., Chen, T. T., & Huang, C. L. (2012). Selection best starting pitcher of the Chinese professional baseball league in 2010 using AHP and TOPSIS methods. In Sambath, S. & Zhu, E. *Frontiers in Computer Education* (pp. 643-649). Berlin Heidelberg: SpringerLink Verlag.
- Chen, J. K., Pham, V. K., & Yuan, B. J. (2013). Adopting AHP approach on evaluation and selection of outsourcing destination in East and Southeast Asia. *Technology Management in the IT-Driven Services (PICMET)*, 528-537. Retrieved from <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6641715>
- Chen, J., & Chen, J. C. (2001). QFD-based technical textbook evaluation—procedure and a case study. *Journal of Industrial Technology*, 18(1), 1-8.
- Chin, W. W., Johnson, N., & Schwarz, A. (2008). A fast form approach to measuring technology acceptance and other constructs. *MIS Quarterly*, 32(4), 687-703. Retrieved from <http://www.jstor.org/stable/25148867>
- Chou, S. Y., Chang, Y. H., & Shen, C. Y. (2008). A fuzzy Simple Additive Weighting system under group decision-making for facility location selection with objective/subjective attributes. *European Journal of Operational Research*, 189(1), 132-145. doi: 10.1016/j.ejor.2007.05.006
- Chou, W. C., & Cheng, Y. P. (2012). A hybrid fuzzy MCDM approach for evaluating website quality of professional accounting firms. *Expert Systems with Applications*, 39(3), 2783-2793. doi: 10.1016/j.eswa.2011.08.138
- Christian, T. (2010). *Security requirements reusability and the SQUARE methodology*. (Technical Note No. CMU/SEI-2010-TN-027). Retrieved from https://resources.sei.cmu.edu/asset_files/TechnicalNote/2010_004_001_15197.pdf
- CMMI Product Team. (2010). *CMMI for Development VI.3*. (Technical Report No. CMU/SEI-2010-TR-033). Retrieved from <http://www.sei.cmu.edu/reports/10tr033.pdf>
- Cockburn, A., & Highsmith, J. (2001). Agile software development: the people factor. *IEEE Computer*, 131-133. doi: 10.1109/2.963450

- Cohen, L. (1995). *Quality Function Deployment: how to make QFD work for you*. Reading, MA: Addison-Wesley.
- Cohn, M., & Ford, D. (2003). Introducing an Agile process to an organization. *IEEE Computer*, 36(6), 74-78. doi: 10.1109/MC.2003.1204378
- Colley, J. (2009). Why secure coding is not enough: professionals' perspective. In Pohlmann, N., Reimer, H. & Schneider, W. *ISSE 2009 Securing Electronic Business Processes* (pp. 302-311). Wiesbaden: Vieweg+Teubner Verlag.
- Cooper, D. R., & Schindler, P. S. (2011). *Business research methods*. New York: McGraw-Hill/Irwin.
- Coram, M., & Bohner, S. (2005). The impact of Agile methods on software project management. *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, 363-370. doi: 10.1109/ECBS.2005.68
- Corbucci, H., Goldman, A., Katayama, E., Kon, F., Melo, C., & Santos, V. (2011). Genesis and evolution of the Agile movement in Brazil- perspective from academia and industry. *25th Brazilian Symposium on Software Engineering*, 98-107. doi: 10.1109/SBES.2011.26
- Coyle, G. (2004). The Analytic Hierarchy Process (AHP). *Practical strategy: Structured tools and techniques*. Retrieved from http://www.booksites.net/download/coyle/student_files/AHP_Technique.pdf
- Crawford, G., & Williams, C. (1985). A note on the analysis of subjective judgment matrices. *Journal of Mathematical Psychology*, 29(4), 387-405. doi: 10.1016/0022-2496(85)90002-1
- Crostack, H. A., Hackenbroich, I., Refflinghaus, R., & Winter, D. (2007). Investigations into more exact weightings of customer demands in QFD. *Asian Journal on Quality*, 8(3), 71-80. doi: <http://dx.doi.org/10.1108/15982688200700026>
- Curtis, B., Hefley, B., & Miller, S. (2009). *People capability maturity model*. (Technical Report No. CMU/SEI-2009-TR-003). Retrieved from <http://www.sei.cmu.edu/reports/09tr003.pdf>
- Dagdeviren, M. (2008). Decision making in equipment selection: an integrated approach with AHP and PROMETHEE. *Journal of Intelligent Manufacturing*, 19(4), 397-406. doi: 10.1007/s10845-008-0091-7
- Dai, J., & Blackhurst, J. (2012). A four-phase AHP-QFD approach for supplier assessment: a sustainability perspective. *International Journal of Production Research*, 50(19), 5474-5490. doi: 10.1080/00207543.2011.639396
- Daneva, M., & Ahituv, N. (2011). What practitioners think of inter-organizational ERP requirements engineering practices: focus group results. *International Journal of Information System Modeling and Design*, 2(3), 49-74. doi: 10.4018/jismd.2011070103

- Davis, N. (2013). *Secure software development lifecycle process*. Retrieved from <https://buildsecurityin.us-cert.gov/articles/knowledge/sdlc-process/secure-software-development-life-cycle-processes>
- Davis, N. (2005). *Secure software development lifecycle processes: a technology scouting report*. Retrieved from <http://www.dtic.mil/dtic/tr/fulltext/u2/a447047.pdf>
- De Felice, F., & Petrillo, A. (2011). A multiple choice decision analysis: an integrated QFD-AHP model for the assessment of customer needs. *International Journal of Engineering, Science and Technology*, 2(9). Retrieved from <http://www.ajol.info/index.php/ijest/article/view/63849/51665>
- Deming, W. (1982). *Out of the crisis*. Cambridge, MA: MIT Center for Advanced Engineering Study.
- Desai, C., Janzen, D. S., & Clements, J. (2009). Implications of integrating test-driven development into CS1/CS2 curricula. *SIGCSE Bull.*, 41(1), 148-152. doi: 10.1145/1539024.1508921
- De Win, B., Scandariato, R., Buyens, K., Gregoire, J., & Joosen, W. (2009). On the secure software development process: CLASP, SDL and Touchpoints compared. *Information and Software Technology*, 51(7), 1152-1171. doi: 10.1016/j.infsof.2008.01.010
- Diaz, J., Garbajosa, J., & Calvo-Manzano, J. A. (2009). Mapping CMMI level 2 to Scrum practices: an experience report. *Software Process Improvement*, 93-104. doi: 10.1007/978-3-642-04133-4_8
- Doernhoefer. (2006). Surfing the Net for Software Engineering notes. *SIGSOFT Software Engineering Notes*, 31(1), 5–13. doi: 10.1145/1874391.1874395
- Doherty, M. J. (2012). Examining project manager insights of Agile and traditional success factors for Information Technology projects: A Q-Methodology study. (Report from Doctoral Dissertation). Retrieved from <http://www.asapm.org/articles/MJDoherty.pdf>
- Dunkerley, K. D., & Tejay, G. (2011). A confirmatory analysis of information systems security success factors. *Hawaii International Conference on System Sciences*, 1530-1605. doi: 10.1109/HICSS.2011.5
- Dutta, A., & McCrohan, K. (2002). Management's role in information security in a cyber economy. *California Management Review*, 45 (1). Retrieved from <http://irps.ucsd.edu/assets/001/501280.pdf>
- Dyba, T., & Dingsoyr, T. (2008). Empirical studies of Agile software development: a systematic review. *Informatics Software Technology*, 50(9-10), 833-859. doi: 10.1016/j.infsof.2008.01.006
- Dyba, T., Dingsoyr, T., & Moe, N. B. (2014). Agile project management. In Ruhe, G. & Wohlin, C. *Software Project Management in a Changing World* (pp. 277-300). Springer Berlin Heidelberg.

- Dyer, R. F., & Forman, E. H. (1992). Group decision support with the Analytic Hierarchy Process. *Decision Support Systems*, 8(2), 99-124. doi: 10.1016/0167-9236(92)90003-8
- Eckman, M. H. (1989). A counterpoint to the Analytic Hierarchy Process. *Medical Decision Making*, 9(1), 57-58. doi: 10.1177/0272989X8900900110
- Elahi, G., Yu, E., Tong, L., & Lin, L. (2011). Security requirements engineering in the wild: a survey of common practices. *IEEE Annual Computer Software and Applications Conference*, 314-319. doi: 10.1109/COMPSAC.2011.48
- El Emam, K., & Birk, A. (2000). Validating the ISO/IEC 15504 measures of software development process capability. *Journal of Systems and Software*, 51(2), 119-149. doi: 10.1016/S0164-1212(99)00117-X
- Erdogan, G., Meland, P.H., & Mathieson, D. (2010). Security testing in Agile web application development-a case study using the east methodology. In Sillitti, A., Martin, A., Xiao., F. W., & Whitworth, E. *Agile Processes in Software Engineering and Extreme Programming* (pp. 14-27). Berlin Heidelberg: SpringerLink Verlag.
- Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile Modeling, Agile software development, and Extreme Programming: the state of research. *Journal of Database Management*, 16(4), 88-100. doi: 10.4018/jdm.2005100105
- Ertugrul, I., & Karakasoglu, N. (2009). Performance evaluation of Turkish cement firms with Fuzzy Analytic Hierarchy Process and TOPSIS methods. *Expert Systems with Applications*, 36(1), 702-715. doi: 10.1016/j.eswa.2007.10.014
- Essafi, M., Labeled, L., & Ghezala, H. B. (2006). S2D-ProM: A strategy oriented process model for secure software development. *International Conference on Software Engineering Advances*. doi: 10.1109/ICSEA.2007.59.
- Evans, R., Tsohou, A., Tryfonas, T., & Morgan, T. (2010). Engineering secure systems with ISO 26702 and 27001. *5th International Conference on System of Systems Engineering (SoSE)*, 1-6. doi: 10.1109/SYSESE.2010.5544065
- Fabbrini, F., Fusani, M., & Lami, G. (2006). Basic concepts of software certification. *First International Workshop on Software Certification*. Retrieved from <http://www.cas.mcmaster.ca/sqrl/papers/SQRLreport37.pdf#page=10>
- Fauziah Baharom, Jamaiah Yahya, Aziz Deraman, & Abdul Razak Hamdan. (2011). SPQF: Software Process Quality Factor for software process assessment and certification. *International Conference on Electrical Engineering and Informatics*, 1-7. doi: 10.1109/ICEEI.2011.6021526
- Fauziah Baharom. (2008). *A software certification model based on development process quality assessment*. (Unpublished doctoral dissertation). Universiti Kebangsaan Malaysia, Selangor, Malaysia.
- Fauziah Baharom, Aziz Deraman, & Abdul Razak Hamdan. (2007). Introducing Software Process Assessment and Certification (SPAC) Model. *The 3rd Malaysian Software Engineering Conference*, 59-63.

- Fauziah Baharom, Aziz Deraman, & Abdul Razak Hamdan. (2005). A survey on the current practices of software development process in Malaysia. *Journal of ICT*, 57-76.
- Fernandes, J. M., & Almeida, M. (2010). Classification and comparison of Agile methods. *Seventh International Conference on the Quality of Information and Communications Technology*, 391-396. doi: 10.1109/QUATIC.2010.71
- Figueira, J., Greco, S., & Ehr Gott, M. (2005). *Multiple Criteria Decision Analysis state of the art of surveys*. New York: Springer.
- Fisher, C. M. (2007). *Researching and writing a dissertation: a guidebook for business students*. England: Prentice Hall.
- Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising Agile methods to software practices at Intel Shannon. *European Journal of Information System*, 15, 200-213. doi: 10.1057/palgrave.ejis.3000605.
- Forman, E., & Peniwati, K. (1998). Aggregating individual judgments and priorities with the Analytic Hierarchy Process. *European Journal of Operational Research*, 108, 165–169. doi: 10.1016/S0377-2217(97)00244-0
- Fowler M. (1999) *Refactoring improving the design of existing code*. Westford: Addison-Wesley.
- Franca, A. C. C., Silva, F. Q. B., & Sousa Mariz, L. M. R. (2010). An empirical study on the relationship between the use of Agile practices and the success of Scrum projects. *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 1-4. doi: 10.1145/1852786.1852835
- Friborg, O., Martinussen, M., & Rosenvinge, J. H. (2006). Likert-based vs. semantic differential-based scorings of positive psychological constructs: A psychometric comparison of two versions of a scale measuring resilience. *Personality and Individual Differences*, 40(5), 873-884. doi: 10.1016/j.paid.2005.08.015
- Fulford, H., & Doherty, N. F. (2003). The application of information security policies in large UK-based organizations: an exploratory investigation. *Information Management & Computer Security*, 11(3), 106-114. doi: http://dx.doi.org/10.1108/09685220310480381
- Futcher, L., & Von Solms, R. (2007). SecSDM: A Model for Integrating Security into the Software Development Life Cycle. *Fifth World Conference on Information Security Education*, 41-48. doi: 10.1007/978-0-387-73269-5_6
- Gallagher, K. P., Goles, T., Hawk, S., Simon, J. C., Kaiser, K. M., Beath, C. M., & Martz, W. B. (2011). A typology of requisite skills for Information Technology professionals. *44th Hawaii International Conference on System Sciences*, 1-10. doi: 10.1109/HICSS.2011.39
- Galín, D. (2004). *Software Quality Assurance*. England: Pearson Education Limited.

- Gallivan, M. J., Truex, D. P., & Kvasny, L. (2004). Changing patterns in IT skill sets 1988-2003: a content analysis of classified advertising. *ACM SIGMIS Database*, 35(3), 64-87. doi: 10.1145/1017114.1017121
- Gandomani, T. J., & Hazura Zulzalil. (2013). Compatibility of Agile software development methods and CMMI. *Indian Journal of Science and Technology*, 6(8), 5089-5094. doi: 10.17485/ijst/2013/v6i8/36349
- Garibay, C., Gutierrez, H., & Figueroa, A. (2010). Evaluation of a digital library by means of quality function deployment (QFD) and the Kano model. *The Journal of Academic Librarianship*, 36(2), 125-132. doi: 10.1016/j.acalib.2010.01.002
- Gay, L. R., Mills, G. E., & Airasian, P. (2006). *Educational research: Competencies for Analysis and Application* (8th Edition). Upper Saddle River, NJ: Pearson Merrill Prentice Hall.
- Geer, D. (2010). Are companies actually using secure development life cycles? *Computer*, 43(6), 12-16. doi: <http://doi.ieeecomputersociety.org/10.1109/MC.2010.159>
- George, B., & Williams, L. (2004). A structured experiment of Test-Driven Development. *Information and Software Technology*, 46(5), 337-342. doi: 10.1016/j.infsof.2003.09.011
- Goerigk, W., & Hoffmann, U. (1999). Rigorous compiler implementation correctness: how to prove the real thing correct. In Hutter, D., Stephan, W., Traverso, P. & Ullmann, M. *Applied Formal Methods-FM-Trends 98* (pp. 122-136). Berlin Heidelberg: SpringerLink Verlag.
- Goertze, K. M. (2009). Introduction to software security. Retrieved from <https://buildsecurityin.us-cert.gov/introduction-software-security>
- Gonzalez, L. S., Rubio, F. G., Gonzalez, F. R., & Velthuis, M. P. (2010). Measurement in business processes: a systematic review. *Business Process Management Journal*, 16(1), 114-134. doi: <http://dx.doi.org/10.1108/14637151011017976>
- Guceglioglu, A., & Demirors, O. (2005). Using software quality characteristics to measure business process quality. *Business Process Management*, 374-379. doi: 10.1007/11538394_26
- Gumus, A. T. (2009). Evaluation of hazardous waste transportation firms by using a two step fuzzy-AHP and TOPSIS methodology. *Expert Systems with Applications*, 36(2), 4067-4074. doi: 10.1016/j.eswa.2008.03.013
- Gumus, A. T., Yayla, A. Y., & Gurbuz, K. (2011). Performance evaluation of ERP implementation by using fuzzy MCDM. *International Symposium on Innovations in Intelligent Systems and Applications*. doi: 10.1109/INISTA.2011.5946114
- Gupta, A., & Jalote, P. (2007). An experimental evaluation of the effectiveness and efficiency of the Test Driven Development. *First International Symposium on*

- Empirical Software Engineering and Measurement*, 285 – 294. doi: 10.1109/ESEM.2007.41
- Guzzo, R. A., & Dickson M. W. (1996) Teams in organizations: recent research on performance and effectiveness. *Annual Review of Psychology*, 47, 307-338. doi: 10.1146/annurev.psych.47.1.307
- Hajkovicz, S. A., McDonald, G. T., & Smith, P. N. (2000). An evaluation of multiple objective decision support weighting techniques in natural resource management. *Journal of Environmental Planning and Management*, 43(4), 505-518. doi: 10.1080/713676575
- Hall, J. H., Sarkani, S., & Mazzuchi, T. A. (2011). Impacts of organizational capabilities in information security. *Information Management & Computer Security*, 19(3), 155-176. doi: <http://dx.doi.org/10.1108/09685221111153546>
- 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. doi: 10.1061/_ASCE_CO.1943-7862.0000137
- Hazzan, O., & Dubinsky, Y. (2009). Workshop on human aspects of Software Engineering. *Proceeding Of The 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, 725-726. doi: 10.1145/1639950.1639984
- Heck, P., Klabbers, M., & Eekelen, M. (2010). A software product certification model. *Software Quality Journal*, 18(1)37-55. doi: 10.1007/s11219-009-9080-0
- Hierholzer, A., Herzwurm, G., & Schlang, H. (2003). Applying QFD for software process improvement at SAP AG, Walldorf, Germany. *Proceedings of the Third Workshop on Software Quality*, 85-95. Retrieved from www.researchgate.net/publication/2817508_Applying_QFD_For_Software_Process_Improvement_At_SAP_AG_Walldorf_Germany/file/9fcfd50cb1d481fcd7.pdf
- Ho, W. (2008). Integrated Analytic Hierarchy Process and its applications—a literature review. *European Journal of Operational Research*, 186(1), 211-228. doi: 10.1016/j.ejor.2007.01.004
- Hoda, R., Noble, J., & Marshall, S. (2011). The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and Software Technology*, 53(5), 521-534. doi: 10.1016/j.infsof.2010.10.009
- Hoggerl, M., & Sehorz, B. (2006). *An introduction to CMMI and its assessment procedure*. Seminar for Computer Science, University of Salzburg, 1-17. Retrieved from http://softwareresearch.sbg.ac.at/fileadmin/src/docs/teaching/WS05/SaI/Paper_Hoeggerl_Sehorz.pdf
- Howard, M., & Lipner, S. (2006). *The Security Development Lifecycle SDL: a process for developing demonstrably more secure software*. Retrieved from

download.microsoft.com/download/f/c/7/fc7d048b-b7a5-4add-be2c-baaee38091e3/9780735622142_SecurityDevLifecycle_ch01.pdf

- Hsiao, S. W. (2002). Concurrent design method for developing a new product. *International Journal of Industrial Ergonomics*, 29(1), 41-55. doi: 10.1016/S0169-8141(01)00048-8
- Huang, L., & Holcombe, M. (2009). Empirical investigation towards the effectiveness of test first programming. *Information and Software Technology*, 51(1), 182-194. doi: <http://dx.doi.org/10.1016/j.infsof.2008.03.007>,
- Hui, H., Dongyan, L., Min, Z., Weizhe., & Dongmin, G. (2014). A coverage and slicing dependencies analysis for seeking software security defects. *The Scientific World Journal*. doi: <http://dx.doi.org/10.1155/2014/463912>
- Humphrey, W. (1989). *Managing the software process*. Mass: Addison-Wesley.
- Humphreys, E. (2008). Information security management standards: compliance, governance and risk management. *Information Security Technical Report*, 13(4), 247-255. doi: 10.1016/j.istr.2008.10.010
- Hwang, C., & Yoon, K. (1981). *Multiple Attribute Decision Making: methods and application*. New York: Springer
- Isawi, A. B. M. (2011). *Software development process improvement for small Palestinian software development companies*. (Master's thesis). Retrieved from http://scholar.najah.edu/sites/default/files/all-thesis/software_development_process_improvement_for_small_palestinian_software_development_companies.pdf
- ISECT (2015). *Information security standards*. Retrieved from <http://www.iso27001security.com>
- Ishizaka, A., & Labib, A. (2011). Review of the main developments in the Analytic Hierarchy Process. *Expert Systems with Applications*, 38(11), 14336-14345. doi: 10.1016/j.eswa.2011.04.143
- Ismail, W., Abedlazeed, N., & Hussin, Z. (2011). Epistemological beliefs of students at high schools: a survey study in Malaysia. *OIDA International Journal of Sustainable Development*, 2(08), 39-46. Retrieved from <http://ssrn.com/abstract=1974094>
- ISO (2015). *ISO Standards*. Retrieved from <https://www.iso.org>
- Jadhav, A. S., & Sonar, R. M. (2008) A hybrid system for selection of the software packages. *International Conference on Emerging Trends in Engineering and Technology*, 337-342. doi: 10.1109/ICETET.2008.7
- Jain, V., & Raj, T. (2013). Evaluation of flexibility in FMS using SAW and WPM. *Decision Science Letters*, 2(4), 223-230. doi: 10.5267/j.dsl.2013.06.003
- Jamaiah Haji Yahya, Fauziah Baharom, Aziz Deraman, & Abdul Razak. (2005). A conceptual framework for software certification. *KUTPM Journal of Technology and Management*, 3(2), 99-111.

- Jamaiah Haji Yahya, Aziz Deraman, & Abdul Razak Hamdan. (2006). A conceptual model for software product certification process, *Proceedings of Conference on Information Science, Technology and Management*.
- Jamaiah Yahya. (2007). *The development of software certification model based on product quality approach*. (Unpublished doctoral dissertation). Universiti Kebangsaan Malaysia, Selangor, Malaysia.
- Jiang, J. J., & Klein, G. (1995). Requisite technical skills for technical support analysts: A survey. *Computer Personnel*, 16(2), 12-20. doi: 10.1145/202896.202899 .
- Jones, C., & Bonsignour, O. (2012). *The economics of software quality*. Boston: Pearson Education.
- Joshi, R., Banwet, D., & Shankar, R. (2011). A Delphi-AHP-TOPSIS based benchmarking framework for performance improvement of a cold chain. *Expert Systems with Applications*, 38(8), 10170-10182. doi: 10.1016/j.eswa.2011.02.072
- Julia, H. A., Barnum, S., Ellison, R. J., McGraw, G., & Mead, N. R. (2008). *Software security engineering*. Boston: Addison-Wesley.
- Jung, H. W. (2001). Rating the process attribute utilizing AHP in SPICE-based process assessments. *Software Process: Improvement and Practice*, 6(2), 111-122. doi: 10.1002/spip.139
- Jyothi, V. E., & Rao, K. N. (2011). Effective implementation of Agile practices. *International Journal of Advanced Computer Science and Applications*, 2(3), 41-48. Retrieved from <http://www.Agilemethod.csie.ncu.edu.tw/Agilemethod/download/2011papers/2011%20Effective%20Implementation%20of%20Agile%20Practices%20-%20Ingenious%20and%20Organized%20Theoretical%20Framework/100522015%20E8%94%A1%E6%9D%B1%E7%A9%8E.pdf>
- Kankanhalli, A., Teo, H. H., Tan, B. C. Y., & Wei, K. K. (2003). An integrative study of information systems security effectiveness. *International Journal of Information Management*, 23(2), 139-154. doi: [http://dx.doi.org/10.1016/S0268-4012\(02\)00105-6](http://dx.doi.org/10.1016/S0268-4012(02)00105-6)
- Karpati, P., Sindre, G., & Opdahl, A. L. (2011). Characterising and analysing security requirements modelling initiatives. *Sixth International Conference on Availability, Reliability and Security*, 710-715. doi: 10.1109/ARES.2011.113
- Kazemi, M., Khajouei, H., & Nasrabadi, H. (2012). Evaluation of information security management system success factors: case study of municipal organization. *African Journal of Business Management*, 6(14), 4982-4989. doi: 10.5897/AJBM11.2323
- Khalane, T., & Tanner, M. (2013). Software quality assurance in Scrum: The need for concrete guidance on SQA strategies in meeting user expectations.

- International Conference on Adaptive Science and Technology*, 1-6. doi: 10.1109/ICASTech.2013.6707499
- Khan, M., & Kukalis, S. (1990). MIS professionals: education and performance. *Information & Management*, 19(4), 249-255. doi: 10.1016/0378-7206(90)90034-F
- Knapp, K. J., Marshall, T. E., Rainer, R. K., & Ford, F. N. (2006). Information security: management's effect on culture and policy. *Information Management & Computer Security*, 14(1), 24-36. doi: <http://dx.doi.org/10.1108/09685220610648355>
- Koi, K. L. (2012, January 11). 15200 cases of cyber crimes last year. *New Straits Times*. Retrieved from <http://www.nst.com.my/opinion/columnist/15-200-cases-of-cyber-crimes-last-year-1.30592>
- Komuro, M., & Komoda, N. (2008). An explanation model for quality improvement effect of peer reviews. *International Conference on Computational Intelligence for Modelling Control & Automation*. 1159-1164. doi: 10.1109/CIMCA.2008.187
- Kontio, J., Bragge, J., & Lehtola, L. (2008). The focus group method as an empirical tool in software engineering. In Shull, F., Singer, J., & Sjöberg, D. D. K. *Guide to advanced empirical software engineering* (pp. 93-116). London: SpringerLink Verlag.
- Kontio, J., Lehtola, L., & Bragge, J. (2004). Using the focus group method in software engineering: obtaining practitioner and user experiences. *International Symposium on Empirical Software Engineering*, 271-280. doi: 10.1109/ISESE.2004.1334914
- Kontos, T. D., Komilis, D. P., & Halvadakis, C. P. (2005). Siting MSW landfills with a spatial multiple criteria analysis methodology. *Waste management*, 25(8), 818-832. doi: 10.1016/j.wasman.2005.04.002
- Koskela, J. (2003). *Software configuration management in Agile methods*. (Research Report No. 514). Retrieved from <http://www2.vtt.fi/inf/pdf/publications/2003/P514.pdf>
- Koskosas, I. V., & Paul, R. J. (2004). The interrelationship and effect of culture and risk communication in setting internet banking security goals. *Proceedings of the 6th International Conference on Electronic Commerce*, 341-350. doi: 10.1145/1052220.1052264
- Kotulic, A. G., & Clark, J. G. (2004). Why there aren't more information security research studies. *Information & Management*, 41(5), 597-607. doi: 10.1016/j.im.2003.08.001
- Kraemer, S., Carayon, P., & Clem, J. (2009). Human and organizational factors in computer and information security: pathways to vulnerabilities. *Computers & Security*, 28(7), 509-520. doi: 10.1016/j.cose.2009.04.006

- Kraemer, S., & Carayon, P. (2007). Human errors and violations in computer and information security: The viewpoint of network administrators and security specialists. *Applied Ergonomics*, 38(2), 143-154. doi: 10.1016/j.apergo.2006.03.010
- Kraemer, S., & Carayon, P. (2005). Computer and information security culture: findings from two studies. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 1483-1488. doi: 10.1177/154193120504901605
- Kroeger, T. A. (2011). *Understanding the characteristics of quality for software engineering processes*. (Doctoral dissertation). Retrieved from http://ura.unisa.edu.au/view/action/singleViewer.do?dvs=1412756066477~544&locale=en_US&VIEWER_URL=/view/action/singleViewer.do?&DELIVERY_RULE_ID=10&adjacency=N&application=DIGITool-3&frameId=1&usePid1=true&usePid2=true
- Krueger, R. A. (1994). *Focus group a practical guide for applied research*. Thousand Oaks: SAGE Publications.
- Krueger, R. A., & Casey M. A. (2008). *Focus groups a practical guide for applied research*. Thousand Oaks: Sage Publications.
- Kumar, M., Cheng, N., Nadirah Rodzi, & Natasya Joibi. (2014, September 30). Gang steals RM3m from ATMs. *The Star Online*. Retrieved from <http://www.thestar.com.my/News/Nation/2014/09/30/Gang-steals-RM3m-from-ATMs-Thieves-use-malware-to-bypass-authentication-process/>
- Kunda, D. (2003). STACE: Social technical approach to COTS software evaluation. In Cechich, A., Piayyini, M., & Vallecillo, A. *Component-Based Software Quality* (pp. 64-84). Berlin Heidelberg: Springer-Verlag.
- Lai, V. S., Wong, B. K., & Cheung, W. (2002). Group decision making in a multiple criteria environment: a case using AHP in software selection. *European Journal of Operational Research*, 137, 134-144. doi: 10.1016/S0377-2217(01)00084-4
- Lai-Kow, C., & Ming-Lu, W. (2002). Quality Function Deployment: a literature review. *European Journal of Operational Research*, 143(3), 463-497. doi: 10.1016/S0377-2217(02)00178-9
- Lami, G., & Falcini, F. (2009). Is ISO/IEC 15504 Applicable to Agile methods? In Abrahamsson, P., Marchesi, M., & Maurer, F. *Agile Processes in Software Engineering and Extreme Programming* (pp. 130-135). Berlin Heidelberg: SpringerLink Verlag.
- Lan, C., & Ramesh, B. (2008). Agile requirements engineering practices: an empirical study. *IEEE Software*, 60-67. doi: 10.1109/MS.2008.1
- Lane, T. (2007). Information security management in Australian universities-an exploratory analysis. (Master's thesis). Retrieved from http://eprints.qut.edu.au/16486/1/Tim_Lane_Thesis.pdf

- LaReau, B. S. (2006). An engineer's primer on information security [White Paper]. Retrieved from Brent Scott LeReau: http://w.designsbylareau.com/pdf/AnEngineersPrimerOnInformationSecurity_.pdf
- Lascelles, D., & Peacock, R. (1996). *Self-assessment for business excellence*. Berkshire: McGraw-Hill.
- Lee, H. B. (2011, July 26). RM 63 juta rugi angkara jenayah siber. *Utusan Malaysia*. Retrieved from http://www.utusan.com.my/utusan/info.asp?y=2011&dt=0726&pub=Utusan_Malaysia&sec=Jenayah&pg=je_01.htm
- Lee, G., & Xia, W. (2010). Toward Agile: An integrated analysis of quantitative and qualitative field data on software development agility. *MIS Quarterly*, 34(1), 87-114.
- Leitheiser, R. L. (1992). MIS skills for the 1990s: a survey of MIS managers' perceptions. *Journal of Management Information Systems*, 9(1), 69-91. Retrieved from <http://www.jstor.org/discover/10.2307/40398019?uid=3738672&uid=2&uid=4&sid=21104154371041>
- Li, J., Moe, N. B., & Dyba, T. (2010). Transition from a plan-driven process to Scrum: a longitudinal case study on software quality. *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. doi: 10.1145/1852786.1852804
- Liamputtong, P. (2011). *Focus group methodology principles and practices*. London: SAGE Publication.
- Liberatore, M. J., & Nydick, R. L. (1997). Group decision making in higher education using the Analytic Hierarchy Process. In Liberatore, M. & Nydick, R. L. *Research in Higher Education* (pp. 593-614). Netherlands: Springer.
- Limaye, M. (2011). *Software Quality Assurance*. New Delhi: Tata McGraw-Hill.
- Lin, H., Y., Hsu, P. Y., & Sheen, G. J. (2007). A fuzzy-based decision-making procedure for data warehouse system selection. *Expert systems with applications*, 32(3), 939-953. doi: 10.1016/j.eswa.2006.01.031
- Linberg, K. R. (1999). Software developer perceptions about software project failure: a case study. *The Journal of Systems and Software* (49): 177-192. doi: 10.1016/S0164-1212(99)00094-1
- Lindstrom, L., & Jeffries, R. (2004). Extreme programming and Agile software development methodologies. *Information Systems Management*, 21(3), 41-52. doi: 10.1201/1078/44432.21.3.20040601/82476.7
- Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., . . . Zelkowitz, M. (2002). Empirical findings in Agile methods. In Wells, D. & Williams, L. *Proceedings of Extreme Programming and Agile Methods*, Extreme

- Programming and Agile Methods — XP/Agile Universe 2002 (pp. 197-207). Berlin Heidelberg: Springer.
- Linkov, I., & Moberg, E. (2012). *Multi-criteria decision analysis environmental applications and case studies*. New York: Taylor & Francis Group.
- Lipner, S. (2006). The trustworthy computing security development lifecycle. *20th Annual Computer Security Applications Conference*, 2-13. doi: 10.1109/CSAC.2004.41
- Liu, J., Wang, Q., & Gao, L. (2010). Application of Agile requirement engineering in modest-sized information systems development. *Second WRI World Congress on Software Engineering*, 207-210. doi: 10.1109/WCSE.2010.105
- Litecky, C., Igou, A. J., & Aken, A. (2012). Skills in the management oriented IS and enterprise system job markets. *Proceedings of the 50th annual conference on Computers and People Research*, 35-44. doi: 10.1145/2214091.2214104
- Livermore, J. A. (2007). Factors that impact implementing an Agile software development methodology. *Proceedings of SoutheastCon*, 82-86. doi: 10.1109/SECON.2007.342860
- Lohan, G., Conboy, K., & Lang, M. (2010). Beyond budgeting and Agile software development: A conceptual framework for the performance management of Agile software development teams. *International Journal of Information Systems*, 1-13. Retrieved from http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1158&context=icis2010_submissions
- Ludewig, J. (2000). 10 Years Back, 10 Years Ahead. In Wilhelm, R. *Software Engineering in the Years 2000 Minus and Plus Ten* (pp. 102-111). Berlin Heidelberg: Springer Berlin Heidelberg.
- Mach, P., & Guaqueta, J. (2001). Utilization of the seven Ishikawa tools (old tools) in the Six Sigma strategy. *24th International Spring Seminar on Electronics Technology: Concurrent Engineering in Electronic Packaging*, 51-55. doi: 10.1109/ISSE.2001.931009
- Macharis, C., Springael, J., De Brucker, K., & Verbeke, A. (2004). PROMETHEE and AHP: The design of operational synergies in multicriteria analysis: strengthening PROMETHEE with ideas of AHP. *European Journal of Operational Research*, 153(2), 307-317. doi: 10.1016/S0377-2217(03)00153-X
- Mahnic, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software*, 85(9), 2086-2095. doi: 10.1016/j.jss.2012.04.005
- Malczewski, J. (1999). *GIS and multicriteria decision analysis*. New York: John Wiley & Sons.
- Marcal, A. S. C., de Freitas, B. C. C., Soares, F. S. F., Furtado, M. E. S., Maciel, T. M., & Belchior, A. D. (2008). Blending Scrum practices and CMMI project management process areas. In Marcal, A. S. C., de Freitas, B. C., Soares, F. S.

- F., Furtado, M. E. S., Maciel, T. M., & Belchior, A. D. *Innovations in Systems and Software Engineering* (pp. 17-29). Springer-Verlag
- Marjani, M. E., Soh, K. G., Majid, M., Mohd Sofian, O. F., Nur Surayyah, M. A., & Mohd Rizam, A. B. (2012). Usage of group decision making approach in karate agility test selection. *Proceedings of the International Symposium on the Analytic Hierarchy Process*, 1-11. Retrieved from <http://www.isahp.org/uploads/59.pdf>
- Marra, R. J. (2009, August 17). Three men indicted for hacking into five corporate entities, including heartland, 7-Eleven, and Hannaford, with over 130 million credit and debit card numbers stolen. *United States Department of Justice Online News*. Retrieved from <http://www.usdoj.gov/usao/nj/press/>
- Martakis, A., & Daneva, M. (2013). Handling requirements dependencies in Agile projects: A focus group with Agile software development practitioners. *Seventh International Conference on Research Challenges in Information Science*, 1-11. doi: 10.1109/RCIS.2013.6577679
- Maruping, L. M., Venkatesh, V., & Agarwal, R. (2009). A control theory perspective on Agile methodology use and changing user requirements. *Information Systems Research*, 20(3), 377-399. doi: 10.1287/isre.1090.0238
- Mas, A., Fluxa, B., & Amengual, E. (2012). Lessons learned from an ISO/IEC 15504 SPI programme in a company. *Journal of Software: Evolution and Process*, 24(5), 493-500. doi: 10.1002/smr.501
- Maurer, F., & Martel, S. (2002). Extreme programming. rapid development for Web-based applications. *Internet Computing*, 6(1), 86-90. doi: 10.1109/4236.989006
- Maxville, V., Armarego, J., & Lam, C. P. (2004). Intelligent component selection. *Proceedings of Computer Software and Applications Conference*, 244-249. doi: 10.1109/CMPSAC.2004.1342839
- Mazni Omar, Sharifah-Lailee Abdullah, & Azman Yassin. (2011). The impact of Agile approach on software engineering teams. *American Journal of Economics and Business Administration*, 3(1), 12-17. doi: 10.3844/ajebasp.2011.12.17
- Mazza, R., & Berre, A. (2007). Focus group methodology for evaluating information visualization techniques and tools. *11th International Conference Information Visualization*, 74-80. doi: 10.1109/IV.2007.51
- McConnell, S. (2000). Closing the gap. *Software, IEEE*. 1(19). doi: <http://doi.ieeecomputersociety.org/10.1109/MS.2002.976933>
- McGraw, G. (2011). Technology transfer: A software security marketplace case study. *Software, IEEE*, 28(5), 9-11. doi: 10.1109/MS.2011.110
- McGraw, G. (2006). *Building security in*. Boston: Pearson Education.

- McGraw, G. (2004). Software security. *Security & Privacy, IEEE*, 2(2), 80-83. doi: 10.1109/MSECP.2004.1281254
- Mead, N. R. (2010). Security requirement engineering. Retrieved from <https://buildsecurityin.us-cert.gov/articles/best-practices/requirements-engineering/security-requirements-engineering>
- Mehta, M., & Adlakha, N. (2012). Manifestation of Agile methods for prompt software development: a review. *International Journal of Research in IT & Management*, 2(2), 249-255. Retrieved from <http://www.euroasiapub.org/IJRIM/Feb2012/paper3.pdf>
- Mellado, D., Blanco, C., Sanchez, L. E., & Fernandez-Medina, E. (2010). A systematic review of security requirements engineering. *Computer Standards & Interfaces*, 32(4), 153-165. doi: 10.1016/j.csi.2010.01.006
- Merkow, S. M. & Raghavan, L. (2010). *Secure and resilient software development*. Boca Raton: Auerbach Publications.
- Microsoft. (2012). Microsoft Security Development Lifecycle SDL Process Guidance Version 5.2. Retrieved from <http://www.microsoft.com/en-my/download/confirmation.aspx?id=29884>
- Misra, S., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting Agile software. *The Journal of Systems and Software*, 82, 1869–1890. doi:10.1016/j.jss.2009.05.052
- Mollaghasemi, M. (1997). *Technical briefing: making multiple-objective decisions*. California: IEEE Computer Society Press.
- Moe, N. B., Dingsoyr, T., & Dyba, T. (2008). Understanding self-organizing teams in Agile software development. *19th Australian Conference on Software Engineering*, 76-85. doi: 10.1109/ASWEC.2008.4483195
- Mohd Hassan Selamat, Md. Mahbubur Rahim, & Noor Maizura Mohamad Noor. (1996). Perceptions of selected Malaysian information systems practitioners towards software prototyping: An exploratory study. *Malaysian Journal of Computer Science*, 9 (2), 14-28. Retrieved from http://icmsm2009.um.edu.my/filebank/published_article/1688/12.pdf
- Mohd. Noah A. Rahman, Md. Mahbubur Rahim, Afzaal H. Seyal, & Awg Yussof Awg Mohamed. (1999). Interpersonal skill requirements for fresh computer programmers: expectation of Brunei-based organizations. *Malaysian Journal of Computer Science*, 12(2), 10-18. Retrieved from www.researchgate.net/publication/241032508_INTERPERSONAL_SKIL_REQUIREMENTS_FOR_FRESH_COMPUTER_PROGRAMMERS_EXPECTATION_OF_BRUNEI-BASED_ORGANISATIONS/file/72e7e52c0192835.pdf
- Moody, D. L. (1998). Metrics for evaluating the quality of Entity Relationship Models. In Tok-Wang, L., Ram, S., & Mong, L.L. *Conceptual Modeling—ER '98* (pp. 211-225). Berlin Heidelberg: Springer Berlin Heidelberg.
- Morgan, D. L. (1998). *Planning focus groups*. Thousand Oaks: SAGE Publications.

- 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. In Meyer, B., Nawrocki, J. R., & Walter, B. *Balancing Agility and Formalism in Software Engineering*, (pp. 252-266). Berlin:Springer Berlin Heidelberg.
- Muniraman, C., & Damodaran, M. (2007). A practical approach to include security in software development. *Issues in Information Systems*, 8(2), 193-199. Retrieved from http://iacis.org/iis/2007/Muniraman_Damodaran.pdf
- Nagappan, N., Maximilien, E. M., Bhat, T., & Williams, L. (2008). Realizing quality improvement through Test Driven Development: results and experiences of four industrial teams. *Empirical Software Engineering*, 13(3), 289-302. doi: 10.1007/s10664-008-9062-z
- Nardi, P. M. (2003). *Doing survey research—a guide to quantitative methods*. Boston: Pearson Education.
- Nasution, M. F., & Weistroffer, H. R. (2009). Documentation in systems development: a significant criterion for project success. *Proceedings of the 42nd Hawaii International Conference on System Sciences*, 1-9. doi: 10.1109/HICSS.2009.167
- National Cyber Security Alliance. (2012). National small business study. Retrieved from <https://www.staysafeonline.org>
- Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to Agile methodologies. *Communications of ACM*, 48(5), 72-78. doi: 10.1145/1060710.1060712
- Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People*, 249-256. doi: 10.1145/97243.97281
- Nugraha, F. (2013). Decision support system for evaluation procurement of goods with Simple Additive Weighting Method (SAW). *International Conference on Information Systems for Business Competitiveness*, 211-215. Retrieved from http://eprints.undip.ac.id/41795/1/38-_Fajar_Nugraha.pdf
- Nunes, F. J. B., Belchior, A. D., & Albuquerque, A. B. (2010). Security engineering approach to support software security. *6th World Congress on Services*. 48-55. doi:10.1109/SERVICES.2010.37
- Offut, M. (2002). Quality attributes of web software applications. *IEEE Software*, 19(2), 25-32. doi:10.1109/52.991329
- Oppenheim, A. N. (1992). *Questionnaire design, interviewing and attitude measurement*. London: Pinter Publishers.
- O'Regan, G. (2014). Software Process Improvement. In O'Regan, G. *Introduction to software quality* (pp.199-209). Switzerland: Springer International Publishing.
- 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. Retrieved from www.atlantispress.com/php/download_paper.php?id=9613
- OWASP. (2006). CLASP best practices. Retrieved from https://www.owasp.org/index.php/Category:CLASP_Best_Practice
- Padumadasa, E. U., Colombo, S., & Rehan, S. (2009). Investigation in to Decision Support Systems and Multiple Criteria Decision Making to develop a Web-based tender management system. *Proceedings of the International Symposium on the Analytic Hierarchy Process*, 1-17. Retrieved from http://www.isahp.org/2009Proceedings/Final_Papers/66_Padumadasa_EvaluatingTenderOffers_REV_FIN.pdf
- 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*, 308-313. doi: 10.1109/ENABL.2003.1231428
- Pahnla, S., Siponen, M., & Mahmood, A. (2007). Employees' behavior towards IS security policy compliance. *Proceedings of the 40th Hawaii International Conference on System Sciences*, 156-166. doi: 10.1109/HICSS.2007.206
- Park, T., & Kim, K. J. (1998). Determination of an optimal set of design requirements using House of Quality. *Journal of Operations Management*, 16(5), 569-581. doi: 10.1016/S0272-6963(97)00029-6
- Parsons, D., Ryu, H., & Lal, R. (2007). The impact of methods and techniques on outcomes from Agile software development projects. In McMaster, T., Wastell, D., Ferneley, E., & DeGross, J. I. *Organizational Dynamics of Technology-Based Innovation: Diversifying the Research Agenda* (pp. 235-249). US: Springer.
- Patel, C., & Ramachandran, M. (2009). Agile Maturity Model (AMM): a Software Process Improvement framework for Agile software development practices. *International Journal of Software Engineering*, 2(1), 3-28. Retrieved from http://www.ijse.org.eg/content/vol2/no1/vol2_no1_1.pdf
- Patil, S. K., & Kant, R. (2014). A fuzzy AHP-TOPSIS framework for ranking the solutions of Knowledge Management adoption in supply chain to overcome its barriers. *Expert Systems with Applications*, 41(2), 679-693. doi: 10.1016/j.eswa.2013.07.093
- Phillips, M., & Shrum, S. (2010). Process improvement for all: what to expect from CMMI Version 1.3. *Crosstalk--The Journal of Defense Software Engineering*. Retrieved from [http://www.cs.cmu.edu/~bam/uicourse/2011hasd/Phillips%202010%20-%20What%20to%20Expect%20from%20CMMI%20Version%201.3%20\(Crosstalk\).pdf](http://www.cs.cmu.edu/~bam/uicourse/2011hasd/Phillips%202010%20-%20What%20to%20Expect%20from%20CMMI%20Version%201.3%20(Crosstalk).pdf)
- Pierce, R. E. (2012). *Key factors in the success of an organization's information security culture: A quantitative study and analysis*. (Doctoral dissertation). Retrieved from <http://search.proquest.com/docview/1143268791>

- Pikkarainen, M. (2009). Towards a better understanding of CMMI and Agile integration-multiple case study of four companies. In Bomarius, F., Oivo, M., Jaring, P., & Abrahamsson, P. *Product-Focused Software Process Improvement* (pp. 401-415). Berlin Heidelberg: Springer.
- Pikkarainen, M., & Mantyniemi, A. (2006). An approach for using CMMI in Agile software development assessments: experiences from three case studies. *The SPICE 2006 Conference*. Retrieved from http://Agile.vtt.fi/docs/publications/2006/2006_Agile_cmimi_camera_ready.pdf
- Powell, R. A., & Single, H. M. (1996). Focus groups. *International Journal for Quality in Health Care*, 8(5), 499-504. doi: 10.1093/intqhc/8.5.499
- Pressman, R. S. (2010). *Software Engineering a practitioner's approach 7th Ed.* New York: McGraw-Hill Higher Education.
- Procaccino, J. D., Verner, J. M., Shelfer, K. M., & Gefen, D. (2005). What do software practitioners really think about project success: an exploratory study. *The Journal of Systems and Software* (78): 194-203. doi: 10.1016/j.jss.2004.12.011
- Rae, A., Robert, P., & Hausen, H. L. (1995). *Software evaluation for certification principles, practice and legal liability*. England: McGraw-Hill.
- Rafikul, I., & Shuib, M. R. (2006). Employee performance evaluation by the AHP: A case study. *Asia Pacific Management Review*, 11(3), 163-176. Retrieved from <http://apmr.management.ncku.edu.tw/comm/updown/DW0711300438.pdf>
- Ramesh, B., Lan, C., & Baskerville, R. (2010). Agile Requirements Engineering practices and challenges: an empirical study. *Information Systems Journal*, 20(5), 449-480. doi: 10.1111/j.1365-2575.2007.00259.x
- Rao, K. N., Naidu, G. K., & Chakka, P. (2011). A Study of the Agile software development methods, applicability and implications in industry. *International Journal of Software Engineering & Its Applications*, 5(2), 35-45. Reterived from http://www.sersc.org/journals/IJSEIA/vol5_no2_2011/4.pdf
- Rao, R., & Davim, J. (2008). A decision-making framework model for material selection using a combined Multiple Attribute Decision-Making method. *The International Journal of Advanced Manufacturing Technology*, 35(7-8), 751-760. doi: 10.1007/s00170-006-0752-7
- Rathfelder, C., Groenda, H., & Reussner, R. (2008). Software industrialization and architecture certification. *Proceedings of Industrialization of Software Management*. 169-180. Retrieved from <http://subs.emis.de/LNI/Proceedings/Proceedings139/P-139.pdf#page=170>
- Richardson, I., & Ryan, K. (2001). Software Process Improvements in a very small company. *Software Quality Professional*, 3(2), 23-35. Retrieved from <http://www.itu.dk/~katten/speciale/Software%20Process%20Improvements%20in%20a%20Very%20Small%20Company.pdf>

- Rico, D., Sayani, H., & Sone, S. (2009). *The business value of Agile software methods*. Fort Lauderdale: J.Ross.
- Ritchie, L., & Dale, B. G. (2000). Self-assessment using the business excellence model: a study of practice and process. *International Journal of Production Economics*, 66(3), 241-254. doi: 10.1016/S0925-5273(99)00130-9
- Rodina Ahmad, & Zaitun Abu Bakar. (2000). Information Systems skills requirements in Malaysia. *Malaysian Journal of Computer Science*, 13 (2), 64-69. Retrieved from http://e-journal.um.edu.my/filebank/published_article/1772/96.pdf
- Rogers, M. R., & Lopez, E. C. (2002). Identifying critical cross-cultural school psychology competencies. *Journal of School Psychology*, 40(2), 115-141. doi:10.1016/S0022-4405(02)00093-6
- Rout, T. (2011). High levels of process capability in CMMI and ISO/IEC 15504. In O'Connor, R. V., Rout, T., McCaffery, F., & Dorling, A. *Software Process Improvement and Capability Determination* (pp. 197-199). Berlin Heidelberg: Springer Berlin Heidelberg
- Rumpe, B., & Schroder, A. (2002). Quantitative survey on Extreme Programming projects. *Third International Conference on Extreme Programming and Flexible Processes in Software Engineering*, 26-30. Retrieved from <http://www.se-rwth.de/~rumpe/publications/Quantitative-Survey-on-Extreme-Programming-Projects.pdf>
- Ruth, N. (2008). *A Multi Criteria Decision Making support to software selection*. (Master's thesis). Retrieved from <http://hdl.handle.net/10570/784>
- Saaty, T. L. (2008). Decision making with the Analytic Hierarchy Process. *International Journal of Services Sciences*, 1(1/2008), 83-98. doi: 10.1504/IJSSci.2008.01759
- Saaty, T. L. (1990). How to make a decision: the Analytic Hierarchy Process, *European Journal of Operation Research*, 48 (1), 9-26. doi: 10.1016/0377-2217(90)90057-I
- Salo, O., & Abrahamsson, P. (2008). Agile methods in European embedded software development organizations: A survey study of Extreme Programming and Scrum, *IET Software*, 2(1), 58-64. doi: 10.1049/iet-sen:20070038
- Salo, O., & Abrahamsson, P. (2005). Integrating Agile software development and Software Process Improvement: a longitudinal case study. *International Symposium on Empirical Software Engineering*, 193-202. doi: 10.1109/ISESE.2005.1541828
- Sanchez, J. C., Williams, L., & Maximilien, E. M. (2007). On the sustained use of a Test-Driven Development practice at IBM. *Agile Conference*, 5-14. doi: 10.1109/AGILE.2007.43
- Sanders, J., & Curran, E. (1994). *Software Quality: A framework for success in software development and support*. Wokingham: Addison-Wesley.

- Santos, M. D. A., Bermejo, P. H. D. S., Oliveira, M. S. D., & Tonelli, A. O. (2011). Agile practices: an assessment of perception of value of professionals on the quality criteria in performance of projects. *Journal of Software Engineering and Applications*, 700-709. doi:10.4236/jsea.2011.412082
- Savitha, K., & Chandrasekar, C. (2011). Vertical handover decision schemes using SAW and WPM for network selection in heterogeneous wireless networks. *Global Journal of Computer Science and Technology*. 11(9). Retrieved from <http://arxiv.org/ftp/arxiv/papers/1109/1109.4490.pdf>
- SCAMPI Upgrade Team. (2011). Standard CMMI® appraisal method for process improvement (SCAMPISM) A, Version 1.3: Method Definition Document Handbook
- Schindler, C. (2008). Agile software development methods and practices in Austrian IT-industry: results of an empirical study. *International Conference on Computational Intelligence for Modelling Control & Automation*, 321-326. doi: 10.1109/CIMCA.2008.100
- Schneiderman, B. (1998). *Designing the user interface: strategies of effective Human-Computer Interaction 3rd edition*. Boston: Addison-Wesley Longman.
- Schuh, P. (2005). *Integrating Agile development in the real world*. Hingham: Charles River Media.
- Scriven, M. (1991). *Evaluation thesaurus: fourth edition*. Newbury Park: Sage Publications.
- Sekaran, U., & Bougie, R. (2010). *Research methods for business*. New York: John Wiley & Sons.
- Sekaran, U. (2003). *Research methods for business* (4th edition). New York, USA: John Wiley & Sons.
- Serkani, E. S., Mardi, M., Najafi, E., Jahanian, K., & Herat, A. T. (2013). Using AHP and ANP approaches for selecting improvement projects of Iranian Excellence Model in healthcare sector. *African Journal of Business Management*, 7(23). Retrieved from http://www.academicjournals.org/article/article1380702998_Serkani%20et%20al.pdf
- Setiawan, F. P., Bouk, S. H., & Sasase, I. (2008). An optimum multiple metrics gateway selection mechanism in MANET and infrastructure networks integration. *IEEE Wireless Communications and Networking Conference*, 2229 – 2234. doi: 10.1109/WCNC.2008.394
- Sfetsos, P., Stamelos, I., Angelis, L., & Deligiannis, I. (2009). An experimental investigation of personality types impact on pair effectiveness in pair programming. *Empirical Software Engineering*, 14(2), 187-226. doi: 10.1007/s10664-008-9093-5
- Sfetsos, P., & Stamelos, I. (2010). Empirical studies on quality in Agile practices: a systematic literature review. *Proceedings of the 2010 Seventh International*

- Conference on the Quality of Information and Communications Technology*, 44-53. doi: 10.1109/QUATIC.2010.17
- Shafiq Hussain, S., Erwin, H., & Dunne, P. (2011). Threat modeling using formal methods: A new approach to develop secure web applications. *7th International Conference of Emerging Technologies*. 1-5. doi: 10.1109/ICET.2011.6048492
- Sheffield, J., & Lematayer, J. (2013). Factors associated with the software development agility of successful projects. *International Journal of Project Management*, 31(3), 459-472. doi: 10.1016/j.ijproman.2012.09.011
- Shih, H. S., Shyur, H. J., & Lee, E. S. (2007). An extension of TOPSIS for group decision making. *Mathematical and Computer Modelling*, 45(7-8), 801-813. doi: 10.1016/j.mcm.2006.03.023
- Simpson, S. (2008). Fundamental practices for secure software development: A guide to the most effective secure development practices in use today: SAFECODE. Retrieved from <http://www.safecode.org>, 2008
- Sindre, G., & Opdahl, A. L. (2001). Capturing security requirements through misuse cases. Retrieved from <http://www.nik.no/2001/21-sindre.pdf>
- Siponen, M., Pahnla, S., & Mahmood, M. (2010). Compliance with information security policies: an empirical investigation. *Computer*. 43(2), 64–71. doi: 10.1109/MC.2010.35
- Sison, R., & Yang, T. (2007). Use of Agile methods and practices in the Philippines. *14th Asia-Pacific Software Engineering Conference*, 462-469. doi: 10.1109/ASPEC.2007.35
- Sison, R., Jarzabek, S., Hock, O. S., Rivepiboon, W., & Hai, N. N. (2006). Software practices in five ASEAN countries: an exploratory study. *Proceedings of the 28th International Conference on Software engineering*, 628-631. doi: 10.1145/1134285.1134378
- Sliger, M., & Broderick, S. (2008). *The software project manager's bridge to agility*. Boston: Addison-Wesley.
- Sliger, M. (2006). A project manager's survival guide to going Agile. Retrieved from http://www.rallydev.com/documents/rally_survival_guide.pdf
- Sommerville, I. (2004). *Software Engineering 7th Ed*. Harlow: Pearson Education Limited.
- Sommerville, I. (2007). *Software Engineering 8th Ed*. Harlow: Pearson Education Limited.
- Srivastava, T. N., & Shailaja, R. (2011). *Business research methodology*. New Delhi: Tata McGrawHill Education Private Limited.
- Stamelos, I. G., & Sfetsos, P. (2007). *Agile software development quality assurance*: IGI Global.

- Sterling, G. D., & Brinthaup, T. M. (2003). Faculty and industry conceptions of successful computer programmers. *Journal of Information Systems Education*, 14(4), 417-424. Retrieved from [http://jise.org/Volume14/14-4/Pdf/14\(4\)-417.pdf](http://jise.org/Volume14/14-4/Pdf/14(4)-417.pdf)
- Stewart, D. W., Shamdasani, P. N., & Rook, D. W. (2007). *Focus groups theory and practices*. Thousand Oaks: Sage Publications.
- Strode, D. E., Huff, S. L., & Tretiakov, A. (2009). The impact of organizational culture on Agile method use. *42nd Hawaii International Conference on System Sciences*, 1-9. doi: 10.1109/HICSS.2009.436
- Suhazimah Dzazali, Ainin Sulaiman, & Ali Hussein Zolait. (2009). Information security landscape and maturity level: case study of Malaysian public service (MPS) organizations. *Government Information Quarterly*, 26(4), 584-593. doi: 10.1016/j.giq.2009.04.004
- Sun-Jen, H., & Wen-Ming, H. (2006). Selection priority of process areas based on CMMI continuous representation. *Information & Management*, 43(3), 297-307. doi: 10.1016/j.im.2005.08.003
- Syed Irfan Nabi, Abdulrahman A. Mirza, & Khaled Alghathbar. (2010). Information assurance in Saudi organizations- an empirical study. In Tai-Hoon, K., Wai-Chi, F., Muhammad Khurram Khan, Arnett, K. P., Heau-jo, K., & Slezak, D. *Security technology, disaster recovery and business continuity* (pp. 18-28). Berlin Heidelberg: Springer Berlin Heidelberg
- Taghizadeh, H., & Mohamadi, P. (2013). Identifying educational services quality using Quality Function Deployment model (QFD) and, Analytic Hierarchy Process (AHP). *African Journal of Business Management*, 7(15), 1250-1257. doi: 10.5897/AJBM10.1613
- Taillandier, P., & Stinckwich, S. (2011). Using the PROMETHEE Multi-Criteria Decision Making method to define new exploration strategies for rescue robots. *IEEE International Symposium on the Safety, Security, and Rescue Robotics*, 321-326. doi: 10.1109/SSRR.2011.6106747
- Tarhan, A., & Yilmaz, S. G. (2013). Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process. *Information and Software Technology*, 56(5), 477-494. doi: 10.1016/j.infsof.2013.12.002
- Tarí, J. J., & Heras-Saizarbitoria, I. (2012). The self-assessment process and impacts on performance: A case study. *International Journal for Quality Research*, 6(4). Retrieved from <http://www.ijqr.net/journal/v6-n4/5.pdf>
- Tessem, B. (2003). Experiences in learning XP practices: A qualitative study. In Marchesi, M. & Succi, G. *Extreme Programming and Agile Processes in Software Engineering* (pp. 131-137). Berlin Heidelberg: Springer Berlin Heidelberg.

- The Standish Group. (2013). Chaos manifesto 2013, think big, act small. Retrieved from <http://www.versionone.com/assets/img/files/ChaosManifesto2013.pdf>
- Thompson, B., & Stapleton, J. C. (1979). A method for validating semantic differential referents. *The Journal of Experimental Educational*, 48 (2), 110-113. Retrieved from <http://www.jstor.org/stable/20151324>
- Tohidi, H. (2011). The role of risk management in IT systems of organizations. *Procedia Computer Science*, 3, 881-887. doi: 10.1016/j.procs.2010.12.144
- Tondel, I. A., Jensen, J., & Rstad, L. (2010). Combining misuse cases with attack trees and security activity models. *International Conference on Availability, Reliability, and Security*, 438-445. doi: 10.1109/ARES.2010.101
- Tondel, I. A., Jaatun, M. G., & Meland, P. H. (2008). Security requirements for the rest of us: A survey. *IEEE Software*, 25(1), 20-27. doi: 10.1109/MS.2008.19
- Torres, J., Sarriegi, J., Santos, J., & Serrano, N. (2006). Managing information systems security: critical success factors and indicators to measure effectiveness. In Katsikas, S. K., Lopez, J., Backes, M., Gritzalis, S., & Preneel, B. *Information Security* (pp. 530-545). Berlin Heidelberg: Springer Berlin Heidelberg
- Triantaphyllou, E. (2000). *Multi-Criteria Decision Making methods: a comparative study*. Netherlands: Kluwer Academic Publishers.
- Triantaphyllou, E., & Mann, S. H. (1995). Using the Analytic Hierarchy Process for decision making in engineering applications: some challenges. *International Journal of Industrial Engineering: Applications and Practice*, 2(1), 35-44. Retrieved from http://bit.csc.lsu.edu/trianta/Journal_PAPERS1/AHPapls1.pdf
- Tripp, L. L. (2002). Benefits of certification. *Computer*, 35(6), 31-33. doi: 10.1109/MC.2002.1009164
- Trochim, W. M. (2006). *The research methods knowledge base*. Retrieved from <http://www.socialresearchmethods.net/kb/dedind.php>
- Tsaur, R. C. (2011). Decision risk analysis for an interval TOPSIS method. *Applied Mathematics and Computation*, 218(8), 4295-4304. doi: 10.1016/j.amc.2011.10.001
- Tsohou, A., Karyda, M., Kokolakis, S., & Kiountouzis, E. (2006). Formulating information systems risk management strategies through cultural theory. *Information Management & Computer Security*, 14(3), 198-217. doi: <http://dx.doi.org/10.1108/09685220610670378>
- 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. doi:10.1016/j.jss.2007.08.020
- Tu, N., Zhang, T., He, Q., Zhang, H., & Li, Y. (2011). Applying combined AHP-QFD method in new product development: A case study in developing new

- sports earphone. *International Conference on Management Science and Industrial Engineering*, 80-85. doi: 10.1109/MSIE.2011.5707520
- Tudor, J. (2013). Web application vulnerability statistics 2013. [White Paper]. Retrieved from Context Information Security: http://www.contextis.com/documents/70/Web_Application_Vulnerability_Statistics_Appendix_-_June_2013.pdf
- Vaidya, O. S., & Kumar, S. (2006). Analytic Hierarchy Process: An overview of applications, *European Journal of Operational Research*, 169(1), 1-29. doi:10.1016/j.ejor.2004.04.028.
- Van Loon, H. (2007). *Process assessment and improvement a practical guide*. Switzerland: Springer.
- Vermesan, A. I. (1998). Software certification for industry-verification and validation issues in expert systems. *Proceedings of Database and Expert Systems Applications*, 3-14. doi: 10.1109/DEXA.1998.707373
- VersionOne. (2011). State of Agile survey. Retrieved from http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf
- Voas, J. (1998). The software quality certification triangle: crosstalk. *The Journal of Defense software engineering*, 12-14. Retrieved from <http://www.crosstalkonline.org/storage/issue-archives/1998/199811/199811-Voas.pdf>
- Voas, J. (1999). User participation-based software certification. In Vermesan, A. & Coenen, F. *Validation and Verification of Knowledge Based Systems* (pp. 267-276). US: Springer-Verlag.
- Voas, J. (2008). Software quality unpeeled. *STSC CrossTalk*, 27-30. Retrieved from <http://www.cellar.com.br/ICSI08/Artigo/Jeffrey%20Voas.pdf>
- Von Solms, B., & Von Solms, R. (2004). The 10 deadly sins of information security management. *Computers & Security*, 23(5), 371-376. doi: 10.1016/j.cose.2004.05.002
- Waly, N., Tassabehji, R., & Kamala, M. (2012). Improving organisational information security management: The impact of training and awareness. *9th International Conference on Embedded Software and Systems High Performance Computing and Communication*, 1270-1275. doi: 10.1109/HPCC.2012.187
- Wang, J. W., Cheng, C. H., & Huang, K. C. (2009). Fuzzy hierarchical TOPSIS for supplier selection. *Applied Soft Computing*, 9(1), 377-386. doi: 10.1016/j.asoc.2008.04.014
- Weber-Jahnke, J. H. (2011). A preliminary study of apparent causes and outcomes of reported failures with patient management software. *3rd Workshop on Software Engineering in Health Care*, 5-8. doi: 10.1145/1987993.1987996

- Wei, X., & Yonghui, C. (2013). Comments on Software Process Improvement methodologies using QFD. *Applied Mathematics & Information Sciences*, 7(3), 1137-1143. Retrieved from <http://t.naturalspublishing.com/files/published/68vp82lzf77ig9.pdf>
- Wells, D. (2013). Extreme Programming. Retrieved from <http://www.extremeprogramming.org>
- Werlinger, R., Hawkey, K., & Beznosov, K. (2009). An integrated view of human, organizational, and technological challenges of IT security management. *Information Management & Computer Security*, 17(1), 4-19. doi: <http://dx.doi.org/10.1108/09685220910944722>
- West, D., & Grant, T. (2010). Agile development: mainstream adoption has changed agility. Forrester Research.
- Wheeler, S., & Duggins, S. (1998). Improving software quality. *Proceedings of the Southeast Regional Conference*. 300-309. doi: 10.1145/275295.275375
- Whitehat Security. (2013). Website security statistics report, WhiteHat Security, Santa Clara, California: Whitehat Security. Retrieved from https://www.whitehatsec.com/assets/WPstatsReport_052013.pdf
- Whitman, M., & Mattord, H. J. (2012). *Principles of information security*. Thomson Course Technology. Boston: Course Technology.
- Wieggers, K. E. (2002). Seven truths about peer reviews. *Cutter IT Journal*. Retrieved from http://www.processimpact.com/articles/seven_truths.pdf
- Wilander, J., & Gustavsson, J. (2005). Security requirements—A field study of current practice. *Symposium on Requirement Engineering for Information Security*. Retrieved from http://www.ida.liu.se/labs/pelab/publications/documents/2005/08_wilander_sreis.pdf
- Williams, L. (2012). What Agile teams think of Agile principles. *Communications of the ACM*, 55(4), 71-76. doi: 10.1145/2133806.2133823
- Williams, L., Rubin, K., & Cohn, M. (2010). Driving process improvement via comparative agility assessment. *Agile Conference*, 3-10. doi: 10.1109/AGILE.2010.12
- Williams, L., & Erdogmus, H. (2002). On the economic feasibility of pair programming. *International Workshop on Economics-Driven Software Engineering Research*. Retrieved from <http://collaboration.csc.ncsu.edu/laurie/Papers/EDSER02WilliamsErdogmus.pdf>
- Xuhua, J., & Pattinson, C. (2010). AHP implemented security assessment and security weight verification. *International Conference on Social Computing*, 1026-1031. doi: 10.1109/SocialCom.2010.153

- Yan, S. (2008). *Business oriented software process improvement based on CMM and CMMI using QFD*. (Doctoral dissertation). Retrieved from https://mospace.umsystem.edu/xmlui/bitstream/handle/10355/26432/Sun_2008.pdf?sequence=1
- Yatsalo, B. I., Kiker, G. A., Kim, J., Bridges, T. S., Seager, T. P., Gardner, K., . . . Linkov, I. (2007). Application of multicriteria decision analysis tools to two contaminated sediment case studies. *Integrated Environmental Assessment and Management*, 3(2), 223-233. doi: 10.1897/IEAM_2006-036.1
- Yoon, K. & Hwang, C. (1995). *Multiple Attribute Decision-Making: An introduction*. Thousand Oaks: Sage Publisher.
- Yazrina Yahya, Maryati Mohd Yusof, Mohammed Yusof, & Nazlia Omar. (2002). The use of Information System development methodology in Malaysia. *Jurnal Antarabangsa (Teknologi Maklumat)*, 15-34.
- Yumin, L., & Jichao, X. (2006). QFD Model for quality performance self-assessment. *Asian Journal on Quality*, 7(1), 112-127. doi: <http://dx.doi.org/10.1108/15982688200600008>
- Zarour Mohammad. (2009). *Methods to evaluate lightweight software process assessment methods based on evaluation theory and engineering design concepts*. (Doctoral dissertation). Retrieved from http://espace.etsmtl.ca/92/1/ZAROUR_Mohammad.pdf
- Zhou, Z., & Liang, K. (2013). Network course evaluation system based on AHP theory. In Wenjiang, Du. *Informatics and Management Science II* (pp. 569-575). London: Springer London.
- Zikmund, W. G., Babin, B. J., Carr, J. C., & Griffin, M. (2010). *Business research methods, 8th edition*. South-Western: Cengage Learning.
- Zultner, R. E. (1992). *Quality Function Deployment (QFD) for software*. American Programmer.