HARMONIZING CMMI-DEV1.2 AND XP METHOD TO IMPROVE THE SOFTWARE DEVELOPMENT PROCESSES IN SMALL SOFTWARE DEVELOPMENT FIRMS

MEJHEM YOUSEF AL-TARAWNEH

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2013

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

Kebanyakan organisasi yang membangunkan perisian komputer adalah firma kecil, dan mereka telah menyedari akan keperluan untuk mengurus dan meningkatkan aktiviti pembangunan dan pengurusan perisian komputer. Model dan piawaian Penambahbaikan Proses Perisian (SPI) yang tradisional didapati tidak realistik bagi firma kecil kerana kos yang tinggi, sumber yang terhad dan tempoh serahan projek yang ketat. Oleh itu, firma kecil memerlukan kaedah pembangunan perisian yang mudah serta model SPI yang sesuai bagi mengurus dan meningkatkan proses pembangunan dan pengurusan perisian. Kajian ini bertujuan untuk membangunkan suatu rangka kerja proses penambahbaikan pembangunan perisian yang sesuai untuk Firma Pembangunan Perisian Kecil (SSDFs) berasaskan kaedah Pengaturcaraan Ekstrem (XP) dan model Model Integrasi Kematangan Keupayaan untuk Pembangunan versi 1.2 (CMMI-Dev1.2). Terdapat empat tahap dalam pembangunan rangka kerja ini iaitu: (1) menjajarkan setiap amalan XP dengan matlamat khusus Bidang Proses Utama (KPAs) CMMI-Dev1.2; (2) membangunkan rangka kerja proses penambahbaikan pembangunan perisian yang dicadangkan dengan mengguna pakai kaedah XP melalui pengadaptasian Pendekatan Berasaskan Penambahan (EBA), CMMI-Dev1.2 dan elemen generik daripada rangka kerja SPI; (3) mengesahkan kesesuaian rangka kerja yang dicadangkan dengan KPAs CMMI-Dev1.2 melalui kaedah kumpulan berfokus yang dipadankan dengan teknik Delphi; dan (4) mengesahkan rangka kerja yang telah diubah suai dengan menggunakan soal selidik CMMI-Dev1.2 sebagai item utama untuk mengesahkan kesesuaian rangka kerja tersebut untuk SSDFs, serta menjalankan dua kajian kes bagi mengesahkan kebolehlaksanaan dan keberkesanan rangka kerja ini bagi firma tersebut. Hasil menjajarkan amalan XP kepada KPAs CMMI-Dev1.2 menunjukkan bahawa dua belas KPAs disokong oleh amalan XP, lapan KPAs sebahagiannya disokong oleh amalan XP, dan dua KPAs tidak disokong oleh amalan-amalan XP. Sumbangan utama kajian ini adalah: penambahbaikan rangka kerja proses pembangunan perisian untuk SSDFs, mendapatkan lebih pemahaman tentang cara untuk membina rangka kerja, dan peningkatan kualiti bagi proses pembangunan perisian. Masih terdapat ruang untuk membuat kajian lanjutan iaitu dengan memenuhi beberapa lompong tertentu dalam amalan KPAs, meneliti amalan kaedah *agile* yang lain dan menggunakan CMMI-Dev1.3 untuk memperbaiki rangka kerja ini, serta menjalankan lebih banyak kajian kes.

Kata kunci: Penambahbaikan proses perisian, Pengaturcaraan ekstrem, Model integrasi kematangan keupayaan untuk Pembangunan versi 1.2, Firma pembangunan perisian kecil

Abstract

Most software development organizations are small firms, and they have realized the need to manage and improve their software development and management activities. Traditional Software Process Improvement (SPI) models and standards are not realistic for these firms because of high cost, limited resources and strict project deadlines. Therefore, these firms need a lightweight software development method and an appropriate SPI model to manage and improve their software development and management processes. This study aims to construct a suitable software development process improvement framework for Small Software Development Firms (SSDFs) based on eXtreme Programming (XP) method and Capability Maturity Model Integration for Development Version 1.2 (CMMI-Dev1.2) model. Four stages are involved in developing the framework: (1) aligning XP practices to the specific goals of CMMI-Dev1.2 Key Process Areas (KPAs); (2) developing the proposed software development process improvement framework based on extending XP method by adapting the Extension-Based Approach (EBA), CMMI-Dev1.2, and generic elements of the SPI framework; (3) verifying the compatibility of the proposed framework to the KPAs of CMMI-Dev1.2 by using focus group method coupled with Delphi technique; and (4) validating the modified framework by using CMMI-Dev1.2 questionnaire as a main item to validate the suitability of the modified framework for SSDFs, and conducting two case studies to validate the applicability and effectiveness of this framework for these firms. The result of aligning XP practices to the KPAs of CMMI-Dev1.2 shows that twelve KPAs are largely supported by XP practices, eight KPAs are partially supported by XP practices, and two KPAs are not-supported by XP practices. The main contributions of this study are: software development process improvement framework for SSDFs, elicit better understanding of how to construct the framework, and quality improvement of the software development processes. There are possible avenues for extending this research to fulfil the missing specific practices of several KPAs, examining other agile practices and using CMMI-Dev1.3 to improve the framework, and conducting more case studies.

Keywords: Software process improvement, eXtreme programming, Capability maturity Model integration for development Version 1.2, Small software development firms.

Acknowledgement

By the name of ALLAH, The Most Gracious, and The Most Merciful.

In this occasion I would like to express my gratitude to a number of people whose admission, permission, and assistance contribute to finish my long story with PhD.

My deepest and warmest gratitude to my supervisor Assoc. Prof. Dr. Mohd Syazwan Abdullah for his assistance and patience in ensuring that the study reached completion. I am grateful for his understanding, advice, encouragement, and for making me confident in my work with timely feedback., I also would like to thank my informal supervisor Assoc. Prof. Abdul Bashah Mat Ali for his continuous guidance, fruitful feedback, moral support, and sharing of all his research experiences throughout these challenging years.

My appreciation to Prof. Dr. Abdul Razak Yaakub as chairman of viva committee, and I would like to present my deep thank to Assoc. Prof. Dr. Wan Mohd Nasir Wan Kadir from University Teknologi Malaysia (UTM) as external examiner, and Assoc. Prof. Dr. Fauziah Baharom as internal examiner for the useful comments and suggestions to improve my thesis.

On a more personal level, I would also like to express my gratitude to my parents and my beloved family members for patience and support throughout my three years plus of difficult endeavor. I guess they are the most who suffered throughout this period. My gratitude also goes to all my colleagues in the PhD journey; among them are Feras, Moath, and Omar Tarawneh, Ali and Wa'el Naimat, and Feras Zain, and many others, specifically for the discussions and sometimes the heated arguments on the better ways to perform my research. They were not only contributing constructive ideas on my research work, but some of them have also read parts of my thesis.

Thank You All Very Much

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

SPI	Software Process Improvement
SEI	Software Engineering Institute
CMM	Capability Maturity Model
CMMs	Capability Maturity Model & Capability Maturity Model Integration Versions
CMMI	Capability Maturity Model Integration
CMMI-Dev1.2	Capability Maturity Model Integration For Development Version 1.2
KPAs	Key Process Areas
SPA	Software Process Assessment
SPICE	Software Process Improvement and Capability Determination
ISO	International Organization for Standardization
IEC	International Electro-technical Commission
EIA	Electronic Industries Alliance
IEEE	Institute of Electrical and Electronics Engineers
SMEs	Small and Medium Enterprises
SME	Situational Method Engineering
SDP	Software Development Process
SDPI	Software Development Process Improvement

SSDFs	Small Software Development Firms
DSDM	Dynamic Systems Development Method
ASD	Adaptive Software Development
FDD	Feature-Driven Development
AM	Agile Modeling
SPM	Software Process Matrix
ASPE-MSC	An Approach for Software Process Establishment in Micro and Small Companies
PRISMS	An Approach to Software Process Improvement for Small to Medium Enterprises
iFLAP	Improvement Framework Utilizing Light Weight Assessment and Improvement Planning
MARES	A Methodology for Software Process Assessment in Small Software Companies
FAME	Fraunhofer IESE Assessment Method
TOPS	Toward Organized Process in SMEs
RAPID	Rapid Assessment for Process Improvement for Software Development
EPA	Express Process Appraisal
SPINI	Software Process Improvement Initiation Framework
S3mAssess	S3m Mini-Assessment Method
EBA	Extension-Based Approach
IPPD	Integrated Product And Process Development

M.V	Mean Value
S.D	Standard Deviation
C.V	Curriculum Vitae
SEPG	Software Engineering Process Group
Freq	Frequency
WCF	Widows Communication Foundation
UI	User Interface
BLL	Business Logic layer
DAL	Data Access Layer
LAN	Local Area Network
TCP	Transmission Control Protocol
VB	Visual Basic
TFS	Team Foundation Server
T	Task

CHAPTER ONE
INTRODUCTION

This chapter provides an overview of the research in this study. It presents the background of the research area and the problem statement of this study. The research question, research objectives, and the scope of this study are also highlighted in the chapter. The chapter also presents the research strategy of the study, followed by the expected contributions of the research. This chapter ends with an overview of the thesis structure.

1.1 Background

Software industry is considered as one of the most important and rapidly growing sectors all over the world. In this regard, software development firms need to be highly focused to be able to develop high quality software products, taking into account the time, cost, scope, and resources. Accordingly, these firms need to have a suitable software development process model to manage their processes in a systematic way. Somerville (2011) defines the software development process model as ”*a simplified representation of a software process. Each process model represents a process from a particular perspective, and thus provides only partial information about that process”.*

The quality of software development process directly affects the quality of the software product. In this respect, it is important for software development firms to improve their software processes to meet the challenges of continuously changing user requirements to satisfy the customer’s needs within the time constraints and maintaining high quality

products. As such, software industry has realized that Software Process Improvement (SPI) is very significant and imperative in order to achieve high quality software products (Pourkomeylian, 2002; BAe, 2007; Nawazish Khokhar et al., 2010).

SPI is the processes of improving the organizations capability to achieve the desired software quality based on well defined processes to improve the organizational capabilities to deliver quality software (Sharma & Sharma, 2012). There are two main traditional SPI models which are: Capability Maturity Model (CMM) (Paulk et al., 1993) and Capability Maturity Model Integration (CMMI) (CMMI Product Team, 2006), and also there are a number of traditional SPI standards such as International Organisation for Standardization 9000 (ISO 9000 series) (Haase, 1996), International Organization for Standardization/ International Electro-technical Commission (ISO/IEC 12207) (Singh, 1996), BOOSTRAP (Kuvaja et al., 1995), and ISO/IEC-15504 Software Process Improvement and Capability Determination (SPICE) (El Emam et al., 1999).

These traditional SPI models and standards were developed to improve the software development processes in large and very large firms (Allen et al., 2003; BAe, 2007; Zhang & Shao, 2011). However, these SPI models and standards are difficult to be directly implemented within the context of most Small Software Development Firms (SSDFs) (Mishra & Mishra, 2009; Gruner & Zyl, 2011). This implementation difficulty is due to: inexperienced staff, lack of defined SPI implementation methodology, lack of SPI awareness, lack of support, lack of resources, organizational politics, and time constraint (Cater-Steel, 2004a; Zarour, 2009; Ibrahim & Ali, 2011).

SSDFs are the software development firms which consist of ten to fifty employees (Laporte et al., 2005; Allison, 2010). These firms represent a high proportion of software firms in most countries all over the world (Richardson & Wangenheim, 2007; Gruner & Zyl, 2011). Most of these firms do not use specific software development process methods in developing the software products due to the lack of awareness on well-defined development processes (Johannesen, 2004; Altarawneh & Amro, 2008; Ali & Ibrahim, 2010). The reason to this is that most of them are using ad-hoc manner for software development (McFarlane & Biktasheva, 2008; Koznov, 2011). Furthermore, most of these firms have insufficient understanding of currently used software development best practices (El Sheikh & Tarawneh, 2007; Valdes et al., 2011).

Nevertheless, SPI in SSDFs is still possible, where some regional initiatives of SPI were developed for these firms such as gradual approach for SPI in Small and Medium Enterprises (SMEs) in Belgium, known as OWPL; Approach for Software Process Establishment in Micro and Small Companies (ASPE-MSC) in Brazil; SPI for SMEs in Britain, known as PRISMS; Improvement Framework Utilizing Lightweight Assessment and Improvement Planning (iFLAP) in Sweden; approach for SPI in SMEs in Spain, known as MESOPYME; Modelo de Procesos para la industria de Software (MoProSoft) in Mexico; and Brazilian software process model in Brazil, known as MPS (Mishra & Mishra, 2009; Isawi, 2011).

However, these regional initiatives are not suitable for SSDFs all over the world, as they were developed based on the characteristics, environments, and infrastructures of firms in these specific countries where the models originated (Isawi, 2011; Mishra & Mishra,

2009). Furthermore, the developments of these initiatives were based on simplifying the SPI traditional models without identifying the suitable software development practices that would achieve global quality level, where these initiatives focused on "what to do for improvement" and ignored "how to do the improvement" (Mishra & Mishra, 2009).

Pikkarainen (2008), Mongkolnam et al. (2009), and Lina and Dan (2012) have indicated the need for a suitable software development process improvement framework for SSDFs. The framework will allow these firms in knowing "what to do for improvement" by the SPI model and "how to do the improvement" by software development best practices (Sison, 2006; Pikkarainen, 2008; Garcia et al., 2010b). Shackel (1991) defines the framework as *"a collection of methods, mechanisms, and processes combined to solve problems, these components work together collaboratively to achieve the specified goal".*

Nowadays, CMMI has become increasingly important to all aspects of software industry (Pikkarainen, 2008; Alshammari & Ahmad, 2010). CMMI for Development Version 1.2 (CMMI-Dev1.2) was written especially for the software industry to guide the software improvement processes (Galinac, 2008; Hashmi & Baik, 2008), and it is the most comprehensive SPI model which is more compliant with relevant SPI models and standards (CMMI Product Team, 2006; Mongkolnam et al., 2009). In addition, Pikkarainen (2008) and Garcia et al. (2010a) argued that CMMI-Dev1.2 is a beneficial approach for identifying the key weaknesses of a software development process in SSDFs which need immediate attention and improvement especially with agile development methods.

Agile methods are a lightweight, efficient, low–risk, flexible, predictable, scientific, and fun way to develop the software product (Beck, 2000). These methods are most suitable for SSDFs compared to the traditional software development methods. EXtreme Programming (XP) method (Beck, 2000) is the most popular and effective method for SSDFs compared to other agile methods such as SCRUM (Alegra & Bastarrica, 2006; Zoysa, 2011). In this respect, Dyba and Dingsoyr (2008) reported that 79% of the empirical reports focused on the use of the XP or SCRUM methods in general, where 76% of the reports related to use of the XP and only 3% to SCRUM practices. In addition, Pikkarainen (2008) and Erharuyi (2007) argued that XP method can help SSDFs in the implementation of SPI, where XP method conforms to level two in CMMI, while SCRUM only conforms to level one in CMMI.

As known, the XP method only applies to a small projects (Beck, 2000), while CMMI applies to the organizations (CMMI Product Team, 2006). Therefore, the most limitations of XP method from software organizational perspective of CMMI is related to the management responsibilities such as quality objectives, organizational training, documentation, and sub-contractor management (Fritzsche & Keil, 2007; Deep, 2012).

Nevertheless, Zoysa (2011), and Lina and Dan (2012) argued that CMMI-Dev1.2 model and XP practices could be used as a combined approach to integrate the best abilities of both. Furthermore, Anderson (2005) and Fritzsche and Keil (2007) Mehrfard et al. (2010) indicated that CMMI-Dev1.2 is a suitable way to improve the software process of XP method, where high levels of CMMI would be possible to be achieved by extending XP method.

1.2 Problem Statement

The integration between XP method and CMMI is very important for SSDFs to help them in developing high quality software products (Baker, 2005). However, there are insufficient studies of how CMMI-Dev1.2 model and XP method practices can be used together to improve the software development processes (Sidky, 2007; Pikkarainen, 2008). Therefore, there is a lack of approaches that really integrates these aspects together, as there is no real integration work carried out, but it rather focuses on mapping XP method to CMMI KPAs (Lina & Dan, 2012).

The overlap and conflict between XP method and the KPAs of CMMI-Dev1.2 had been discussed by several researchers; however there are discrepancies in their results. These discrepancies are resulted from the different ways used in these alignments, where Omran (2008) used the main objective of each KPA as a main item to do the alignment, while Elshafey and Galal-Edeen (2008), and Fritzsche and Keil (2007) used the specific goals of each KPA as main items to do the alignment. Therefore, there is a lack of the comprehensive and systematic alignment of XP practices to CMMI-Dev1.2 (Pikkarainen & Mantyniemi, 2010).

In addition, many researchers such as Vriens (2003), Cohen et al. (2004), and Anderson (2005) argued that most of XP projects that truly follow XP practices could be assessed at level two or three of CMMI. However, these studies did not clearly show how XP method can be extended and integrated with CMMI-Dev1.2 to achieve the suitable KPAs of each level (Fritzsche & Keil, 2007; Zoysa, 2011). In this respect, Elshafey and

Galal-Edeen (2008) and Lina and Dan (2012) indicted the need for extending XP method to fulfill the suitable KPAs of CMMI-Dev1.2.

As highlighted in Section 1.1, XP method only applies to small projects, while SPI traditional models and standards apply to the organizations. Therefore, several organizational limitations in XP method are considered as the main obstacles in achieving high level of these software quality models such as CMMI-Dev1.2. These limitations are related to the management issues, which are:

- **Quality Objectives Problems**

  Quality Assurance (QA) is a planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements (IEEE Std, 1998). XP method focuses on building high quality into the product rather than relying on a quality process that verifies a product after development (Beck, 2000). In this regard, Hashmi and Baik (2007) argued that the QA in XP method is partially achieved by different practices like testing, re-factoring, system metaphor, and pair programming. Therefore, there is lack of process quality measures and analytical data in XP method, as the only aim is to complete each project quickly and then to start all over again with the next (Balkanski, 2003; Khalaf & Al-Jedaiah, 2008). Furthermore, XP method does not have rigorous procedures for the resolution of non-compliance issues and there are no recording for QA activities (Fritzsche & Keil, 2007).

In the term of process performance, XP method has only few quantitative measurements such as test-driven development (Paulk, 2001), but there is no explanation about what happens with testing, such as how many tests pass and fail (Vitoria, 2004). In addition, the testing activities in XP are mainly based on test cases and do not provide documented evidence how these testing activities can be planned, scheduled, and carried out throughout the software life cycle (Qasaimeh & Abran, 2010). Therefore, there is a lack for metrics that control the process performance quantitatively to ensure the high quality of the software development processes (Martinsson, 2002).

- **Organizational Training Problems**

  Organizational training process aims to develop the skills and knowledge of people so they can perform their roles effectively and efficiently (CMM Product Team, 2002). In addition, Highsmith and Cockburn (2004) argued that *"if the people on the project are good trained, they can use almost any process and accomplish their assignments"*.

  Altarawneh and El Shiekh (2008), and Deep (2012) argued that the successful implementation of XP practices are depended on trained and experienced developers. However, in XP method; just pair programming is especially attractive as a means of transferring expertise from experienced to less experienced developers (Beck, 2000), and this is not enough to have trained team (Poole & Huisman, 2002). Furthermore, there are deficiencies regarding the establishment of records and the assessment of training effectiveness in XP method which is

makes it difficult to know the capabilities and experiences of development teams for incoming projects (Fritzsche & Keil, 2007).

- **Documentation Problems**

  Requirements documentation plays an important role in the development and maintenance in the software process (Sengodan, 2003), where the requirements document should serve to: (1) communicate requirements among customers, users, analysts, and designers; (2) support system-testing activities, and (3) control the evolution of the system (Davis, 1993). In XP method, the requirement documentation is just simply supplied by the user stories (Beck, 2000). These stories do not take into account the system requirements or any of the technical details needed during development (Vitoria, 2004), and this is limiting the opportunities and advantages of reusability (Qureshi, 2011).

  In addition, Qasaimeh and Abran (2010) argued that it is not clear how XP method can check traceability problems of the requirements back to the final product; because the traceability from customer requirements to code is not defined in the XP method (Vitoria, 2004). Therefore, there is a lack of documentation though the development lifecycle in XP method, which makes it difficult to maintain the developed system using eXtreme methodology, and the same requirements specifications cannot be used for incoming projects that contains similar requirements (Abrahamsson et al., 2002; Erharuyi, 2007; Nisa, 2012).

- **Sub-Contractor Management Problems**

  Outsourcing of software development tasks to sub-contractors is often based on contracts that precisely stipulate what is required of the subcontractor. This process may be an iterative, incremental approach, but the sub-contractor may have to make the process predictive by specifying the number of iterations and the deliverables of each iteration in order to complete (Turk et al., 2002). XP practices do not support the software sub-contracting (Martinsson, 2002), as these practices focus only on the development processes (Beck, 2000).

  In this respect, Turk et al. (2002) and Mnkandla (2008) argued that it is important to improve the management practices of XP method to be suitable for outsourcing process, as an XP project needs great support from its stakeholders for its success. In addition, Fritzsche and Keil (2007) believed that XP method can be extended to fulfill the required practices of this process, but there is a need to take into account the agility value of XP method, because the involvement of suppliers could be problematic for agility if it hinders iterative development.

Therefore, based on the problems highlighted in this section, there is a need to construct an appropriate software development process improvement framework for SSDFs which enables the integration between XP method and CMMI-Dev1.2 by extending XP method to fulfill the suitable KPAs of CMMI-Dev1.2; taking into account the generic element of SPI framework and the lightness of developed framework's components to ensure it's suitability for SSDFs.

1.3 Research Question

The problem of this research as highlighted in Section 1.2 can be summarized as: SPI traditional models and standards are not suitable to be implemented directly by SSDFs. Thus, there is a need to construct a software development process improvement framework for SSDFs to manage and improve their software development activities by integrating XP method as a lightweight software development method and CMMI-Dev1.2 as a SPI model.

Therefore, the main question of this study is *"How to construct a software development process improvement framework by integrating XP method and CMMI-Dev1.2 model to improve the software development activities of SSDFs?".*

The research question of this study can be divided into sub-questions as follows:

1. What is the extent of KPAs achievements of CMMI-Dev1.2 by XP method?
2. How CMMI-Dev1.2 can be integrated with XP method for SSDFs?
3. How to make developed framework compatible to CMMI-Dev1.2 KPAs?
4. How the framework can be made applicable for SSDFs?

1.4 Research Objectives

The main aim of this research is to construct a software development process improvement framework for SSDFs based on integrating XP method and CMMI-Dev1.2 model. This aim is supported by the following objectives:

1. to identify the coverage of XP practices to the specific goals of CMMI-Dev1.2 KPAs.
2. to develop the software development process improvement framework for SSDFs based on extending XP method that adheres to the suitable KPAs of CMMI-Dev1.2 model and the generic elements of SPI framework.
3. to verify the compatibility of the proposed framework to CMMI-Dev1.2 KPAs by using focus group method coupled with Delphi technique.
4. to validate the applicability of the framework by using CMMI-Dev1.2 questionnaire and case studies.

1.5 Research Scope

The primary concern of the study is to construct a software development process improvement framework for SSDFs. In constructing this framework, two generic elements have been used, which are: a suitable SPI model to know "what-to-do for improvement", and appropriate lightweight software development method to know "how-to-do the improvement".

In this research, XP method has been used as a lightweight software development method in the development framework, as this method is the most popular, useful, and effective lightweight development method of software development in SSDFs. In addition, XP is more compatible to SPI models such as CMMI compared to other popular lightweight methods such as SCRUM method, as the XP practices can conform to level two or three of CMMI, while SCRUM only conforms to level one in CMMI.

As for the SPI model, the CMMI-Dev1.2 model has been chosen as a generic element in the development framework, as this model was written especially for the software industry to guide the software development improvement. It is the most comprehensive SPI and more fully complies with relevant traditional SPI models and standards such as CMM, SPICE, ISO/IEC 12207, and ISO-9000 series (Chrissis et al., 2003; CMMI Product Team, 2006; Mongkolnam et al., 2009). In addition, CMMI-Dev1.2 provides a comprehensive integrated solution for development and maintenance activities applied to products and services. Furthermore, CMMI-Dev1.2 and XP method support each other, as CMMI-Dev1.2 is a suitable way to improve the software process of XP method (Fritzsche & Keil, 2007).

In the validation process of the modified framework, Jordan has been chosen in this study to ensure the suitability and applicability of this framework for SSDFs. This is because most of Jordanian software development firms are small and they have the same generic problems with software development and improvement processes (El Sheikh & Tarawneh, 2007). In addition, it was easy to access these firms, where the same language (native language of the researcher) helped in working with these firms during the validation process.

Even though the validation process of this framework was conducted by Jordanian SSDFs, the framework also could be applicable to other countries. This is due to the fact establishment of the framework was based on a standards XP method and CMMI-Dev1.2 model. In addition, XP method was extended based on generic phases of the

popular software development methodologies such as Waterfall, Spiral, Incremental, and Prototyping and verified based on the principles of XP method.

1.6 Research Strategy

The research process in this study consists of four main stages aimed to achieve the research objectives. These stages are illustrated in Figure 1.1. In Stage One, CMMI-Dev1.2 and XP method were used as main inputs to identify the coverage and missing specific goals of CMMI-Dev1.2 KPAs by XP practices. In Stage Two, the XP method was extended to cover the missing specific goals of partially and not-supported KPAs. Extension-Based Approach (EBA) has been adapted to extend the XP method based on the related literatures of CMM/ CMMI (CMMs) models and XP method, and the popular software development methodologies. Then, the proposed Extended-XP method, the generic elements of SPI framework, and CMMI-Dev1.2 were used to establish the proposed software development process improvement framework. In Stage Three, the focus group method coupled with Delphi technique was used to verify the compatibility of the proposed framework to CMMI-Dev1.2 KPAs. In Stage Four, two approaches were used in validating the modified framework. The first approach is a quantitative research method that involved survey method to validate the suitability of this framework for SSDFs by using CMMI questionnaires; while the second approach is a qualitative research method that involved two case studies to validate the applicability and effectiveness of implementing the framework by SSDFs.

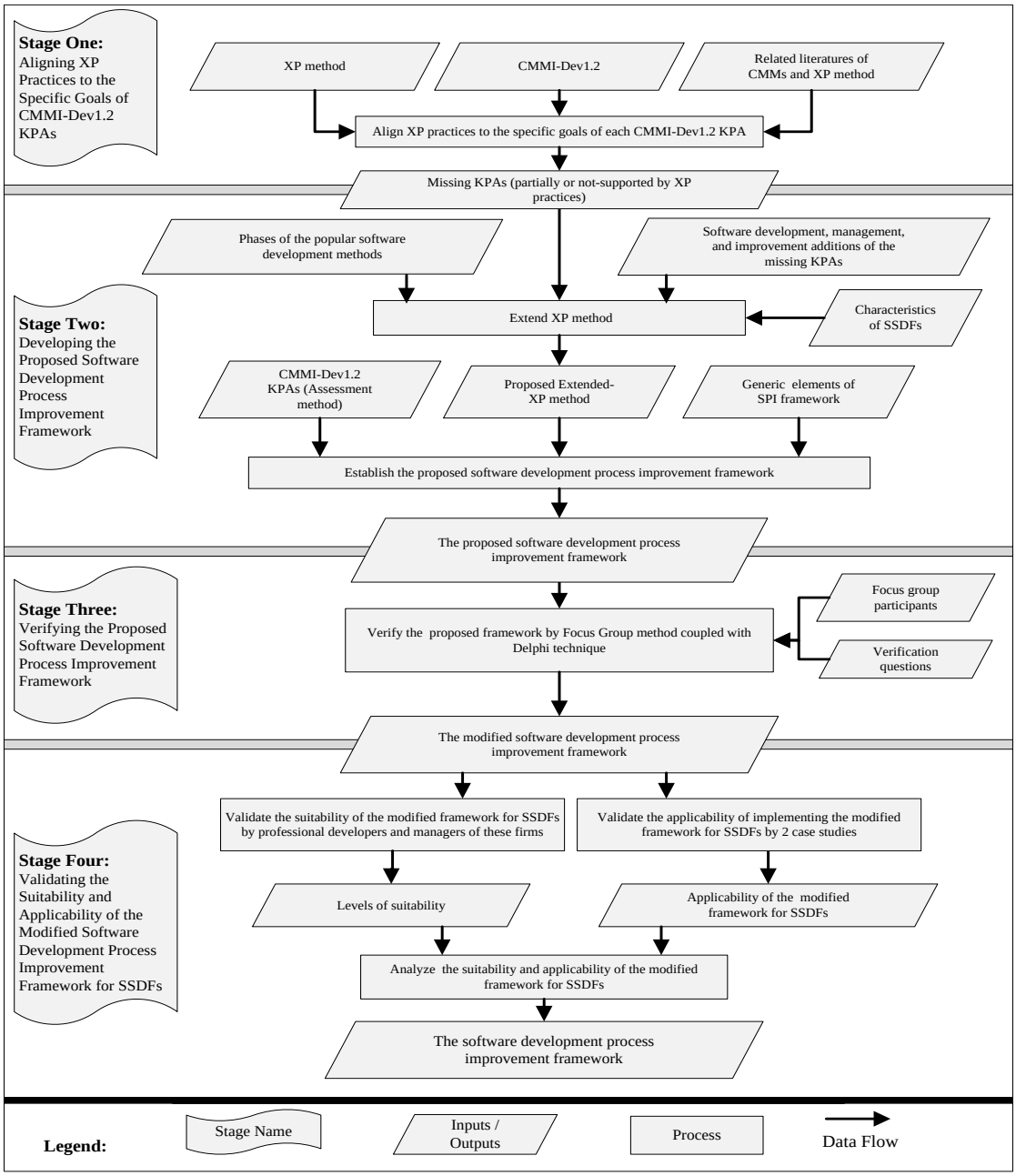

Figure 1.1: Research Strategy

1.7 Contributions

The high level goal of this research is to construct a software development process improvement framework for SSDFs. Therefore, this research contributes toward the

field of software engineering, particularity in that area of SPI. The specific contributions of this research are:

- Demonstrated a new software development process improvement framework for SSDFs based on XP method and CMMI-Dev1.2 model in order to help these firms in managing and improving their software development processes in systematic way.

- Provided new evidence of integrating the CMMI-Dev1.2 and XP method by using the practices of the XP method as the main items in achieving the specific goals of CMMI-Dev1.2 KPAs. This research clearly shows how the integration between CMMI-Dev1.2 model and XP practices can help SSDFs in improving the software development processes.

- Increased the ability of SSDFs to achieve high level of CMMI-Dev1.2 certification by implementing the framework as the framework is compatible to all the KPAs of CMMI-Dev1.2, except the "*organization innovation and deployment"* KPA of level 5.

- Elicited a better understanding of how to construct the framework, especially the processes of adapting the Extension-Based Approach (EBA) to extend XP method and the processes of integration the Extended-XP method with CMMI-Dev1.2 based on modifying the generic element of SPI framework.

- Demonstrated a comprehensive alignment of XP method to CMMI-Dev1.2. This further supports the need for increased attention to be given to the improvement of software development processes by CMMI-Dev1.2 model and XP method. In addition, the results of this alignment have a straightforward and simple guideline to identify suitable development improvement processes for firms of all sizes.

- Increased the right understanding of the project team during the software development lifecycle by identifying their roles specifically, and training them on the best way to achieve the goals of these roles. These processes enable the project members to be very familiar to the current roles, which is increased the productivity of the team members during the software development lifecycle.

1.8 Thesis Organization

- **Chapter One**

  This chapter begins with the background of the problem. Then, the problem statement of this research is discussed. This chapter also presents the research question, research objectives, and the scope of this study. The research strategy used in this research and the expected contributions are presented. Finally, this chapter presents the organizations of the thesis chapters.

- **Chapter Two**

  This chapter gives an overview about software process and SPI. It also discusses the characteristics and problems of these firms, the software development best

practices of these firms, and the difficulty of implementing the SPI traditional models in SSDFs. The chapter also discusses the popular regional SPI initiatives for SSDFs and highlights the popular lightweight assessment methods that have been developed for these firms. This chapter also focuses on CMMI-Dev1.2 model that is used as a baseline improvement model in this study. This chapter also gives an overview of the software development process methods, specifically XP, which was used as a baseline software development method in this research. At the end of this chapter, the relationship between XP and CMMI-Dev1.2 are presented.

- **Chapter Three**

  This chapter presents the research methodology used to achieve the research objectives. It gives an explanation of the four stages used to construct the software development process improvement framework for SSDFs.

- **Chapter Four**

  This chapter presents the stages of developing the proposed software development process improvement framework, which are: aligning XP method to the specific goals of CMMI-Dev1.2 KPAs to know the coverage and missing KPAs, developing the proposed software development process improvement framework based on CMMI-Dev1.2, proposed Extended-XP method, and the generic elements of SPI framework.

- **Chapter Five**

  This chapter presents the verification process of the proposed framework. It presents the three rounds which were conducted to verify the proposed framework through focus group method coupled with Delphi technique. In addition, the chapter also explains the modified software development process improvement framework and the modified Extended-XP method.

- **Chapter Six**

  This chapter presents the validation process of the modified framework by using CMMI-Dev1.2 questionnaires with professional developers and managers to validate the suitability of implementing this framework for SSDFs. The chapter also presents the implementation of the modified framework through two case studies. Finally, the evaluation process that was used to evaluate the effectiveness of the modified framework for SSDFs is discussed.

- **Chapter Seven**

  The final chapter concludes this research based on the research stages used in constructing the software development process improvement framework for SSDFs. This chapter also presents the contributions and the limitations of the study. The chapter ends with the suggested directions for future work in the area of software development and improvement processes.

CHAPTER TWO
SOFTWARE PROCESS IMPROVEMENT AND DEVELOPMENT FOR SMALL SOFTWARE DEVELOPMENT FIRMS

This chapter gives an overview about software process and SPI. This chapter also discusses the general characteristics of SSDFs, problems faced by SSDFs, software development best practices for SSDFs, regional SPI initiatives for SSDFs, and the popular lightweight assessment methods that can be used by SSDFs. The chapter also focuses on CMMI-Dev1.2 model and XP method that were used as baselines in developing the desired framework in this study. Finally, the related works of aligning XP method to CMMI-Dev1.2 are discussed.

2.1 Introduction

Currently, the software industry represents an important economical activity for every country; it offers multiple possibilities for business and it promises to be a great opportunity all over the world. As such, software firms need to have suitable software development methods to manage their software development activities. These methods are the systematic and predefined way the firm's works in general to produce software (Kähkönen, 2005).

Software processes play an important role in helping project teams in software development organizations by providing the suitable organizational stability and good control (Glass, 1995; Wong & Hasan, 2007; Xie et al., 2010). There are many definitions of the software process; nevertheless all of these definitions have the same

aim of helping software engineers to develop software of high quality. In this respect, Saiedian and Carr (1997) define the software process *as "a set of tools, practices, and methods to produce software products according to specific plan"*, while Pressman (2005) defines the software process as "a framework of tasks to build high quality software". In addition, Humphrey and Kellner (1989) summarize the software process as "*the technical and management framework established for applying tools, methods, and people to the software task".*

The ever demanding use of software in all aspects of our life is evident, and due to this; the development costs of the software have increased. This has resulted in software systems that are complex and require complex processes to manage (Allen et al., 2003; Habib, 2009). Therefore, software development firms need to improve their software processes to meet the challenges of continuously changing user requirements to satisfy the customer's needs within the time constraints, while maintaining high quality products (El Emam & Briand, 1997; BAe, 2007). For these reasons, SPI traditional models and standards were developed to manage the organizational capabilities by improving the existing development processes to deliver high quality software within limited time and cost (BAe, 2007; Dagnino, 2009; Baruah, 2012a).

2.2 Software Process Improvement (SPI)

SPI can be defined in many ways, but all of these definitions have similar meaning. Wang and King (2000) define the SPI as "*a systemic procedure for improving the performance of an existing process system by changing or updating the process*", while Sommerville (2011) argues that the SPI is used to understand the current processes and

make changes on the process to improve the product quality, reduce cost or accelerate schedules. Mathiassen et al. (2005) believes that the SPI is "*the primary approach to improving software quality and reliability, employees and customer satisfaction, and return on investment*". Recently, Unterkalmsteiner et al. (2011) define the SPI as *"a systematic approach to increase the efficiency and effectiveness of a software development organization and to enhance software products"*. Based on these definitions, it can be concluded that the SPI is an approach for improving the organizational capability of the software development processes in software firms to achieve high quality software.

Pourkomeylian (2002) and Savcenko and Tanveer (2009) summarize that the main objective of SPI is to improve the organizations capability to achieve the software quality depending on the defined processes or systematic procedures adopted to improve the organizational capabilities to deliver quality software. As shown in Figure 2.1, there are four generic elements for the SPI framework (Rout, 2002; cited by Pressman, 2009), which are:

- **Software Process:** a set of tools, practices, and methods to produce software products according to specific plan.
- **Software Process Assessment**: this element is used to assess the current state of the software process and is done by implementing the suitable assessment methods

- **Capability Determination:** this element is used to know the capability level of the software process and motivates an organization to do process improvement by identifying the capability and risks of a process
- **Improvement Strategy:** based on the capability determination results, the improvement strategy will identify the changes which should be made to the process.

In this study, it is important to take into account these elements as main components in developing the proposed software development process improvement framework for SSDFs.

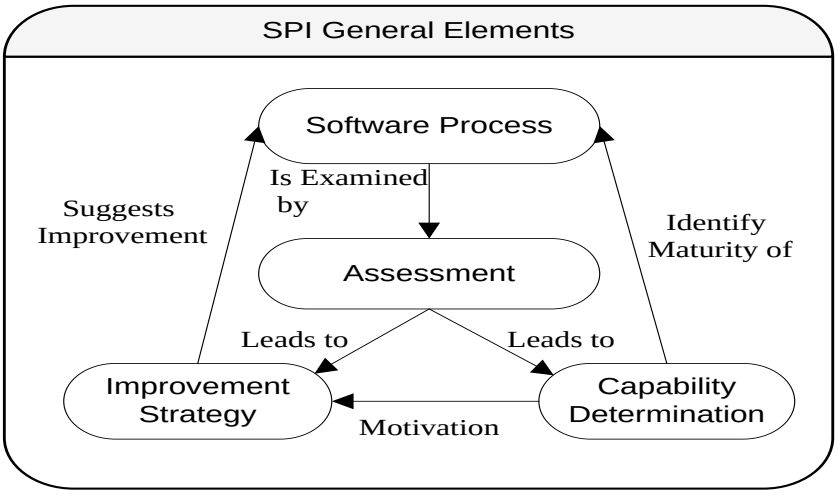

Figure 2.1: Generic Elements of SPI Framework, adopted from (Rout, 2002; cited by Pressman, 2009)

The popular SPI traditional models and standards such as: CMM, CMMI, ISO 9000 series, BOOTSTRAP, ISO/IEC 12207, and SPICE were developed to improve the software development processes in large and very large firms (Allen et al., 2003; BAe,

2007; Zhang & Shao, 2011). However, direct implementation of SPI traditional models and standards by SSDFs which represent the majority of software development firms all over the world are generally not possible (Richardson & Wangenheim, 2007; Gruner & Zyl, 2011). This is because they are not capable of investing the high cost of implementing these programs (Alexandre et al., 2006; Mishra & Mishra, 2009; Gruner & Zyl, 2011). Furthermore, limited resources and strict deadlines to complete the projects compound the difficulty to implement SPI programs, which can also affect quality issues in software project (Zarour, 2009; Ibrahim & Ali, 2011). Section 2.3 discusses the important issues of SSDFs.

2.3 Small Software Development Firms (SSDFs)

SSDFs represent a high proportion of software firms in most countries all over the world (Thorn, 2009; Baruah, 2012a). These firms play an important role in the economy of these countries compared to the larger software firms (Basri & O'Conno, 2011), as they develop a large portion of the needed software applications, offer many job opportunities, exploit new technologies and innovative (Vahaniitty & Rautiainen, 2005; Savolainen et al., 2007; Makitalo-Keinonen et al., 2011). In addition, these firms are believed to provide an impetus to the economic progress of developing countries and its importance is gaining widespread recognition (Palani & Mohideen, 2012).

As for the size of SSDFs, there is no fixed number of the employees to decide the size of SSDFs (Balandis & Laurinskait, 2005; Gruner & Zyl, 2011), and this number differs between countries. Some studies believe the number of employees in SSDFs to be fewer than 50 (Fayad et al., 2000; Carter-Steel, 2001; Da Rocha et al., 2007), and fewer than

60 employees (Laporte et al., 2005). Hofer (2002) and Allison (2010) indicate that the size of SSDFs is between 10 to 50 employees. Based on that, it can be concluded that the average of this number is usually between 10 to 50 employees, and this average is used in this study when referring to SSDFs.

SSDFs can not apply the same software development methodologies or techniques of large software development firms without any modification and optimization due to the major differences between these firms such as the limitation of resources and business issues (Fruhling & Vreede, 2006; Mishra & Mishra, 2009). In addition, best practices proven in large firms might be too expensive or time consuming to be performed in smaller software firms, where agile methods are more applicable for SSDFs, compared to the traditional development models (plan-driven approach) that are more suitable for large software firms (Sommerville, 2007).

2.3.1 SSDFs Characteristics and Problems

There are differences between the characteristics of SSDFs compared to other sized software firms in terms of formalization, centralization, complexity and personnel ratios (Carter-Steel, 2004a). Based on the related literatures of the characteristics of SSDFs, the following can be concluded:

- SSDFs are typically characterized by a flat organizational structure, where most of these firms do not have standards definition of software processes (Makitalo-Keinonen et al., 2011; Nawazish Khokhar et al., 2010; Ibrahim & Ali, 2011).

- SSDfs have less formalized decision-making structures and procedures (Carter-Steel, 2004a; BAe, 2007; Rivas et al., 2008).
- SSDFs have the features that enable the employees to be responsive and flexible, where they provide more freedom for employees to depart from the rules (Carter-Steel, 2004b; Habra el al., 2008; Thorn, 2009).
- SSDFs neglect of the training compared with large software firms (Johannesen, 2004, Carter-Steel, 2004b; Habra et al., 2008).
- The personal involvement of employees in SSDFs encourages motivation and commitment because the employees identify with the company's mission (Daft 1998; Carter-Steel, 2004a).
- SSDFs have faster employment growth rates and generate more new jobs than giant ones (Anacleto et al., 2004; Carter-Steel, 2004a; Savolainen et al., 2007).

As for the problems faced by SSDFs, there are several obstacles facing these firms throughout the development period of the software products. These problems are related to the management of resources, methods and techniques used and human aspects (Hofer, 2002; BAe, 2007; Habra et al., 2008). By reviewing the related literatures of SSDFs, the following can be deduced as the common problems faced by SSDFs:

- Lack of awareness of the well-defined development processes by these firms (Ali & Ibrahim, 2010; Gruner & Zyl, 2011). Therefore, most of them are using ad-hoc manner in developing their software products (Altarawneh & El Shiekh 2008; Koznov, 2011).

- Most of SSDFs have insufficient understanding of currently used software development best practices (El Sheikh & Tarawneh, 2007; Jantunen, 2010; Valdes et al., 2011). Therefore, the cost of developing the software products in these firms are always high, low customer's satisfaction, and the actual time in developing the software products usually exceeds the estimated time (Hofer, 2002; Alexandre at el., 2006; Altarawneh & Amro, 2008).
- Lack of project management and planning practices. Therefore, learning and knowledge management practices are rarely observed (Alexandre et al., 2006; Savolainen et al., 2007; Gruner & Zyl, 2011).
- Most of SSDFs have limited resources for business development (Savolainen et al., 2007; Nawazish Khokhar et al., 2010; Gruner & Zyl, 2011).

2.3.2 Software Development Best Practices for SSDFs

Best practice is the effective technical or management practice which is used to improve the productivity and predictability of cost and schedule (Withers, 2000). Laudon and Laudon (2004) defined the best practices as "the most successful solutions or problem-solving methods that have been developed by a specific organization or industry and are widely recognized as excellent, and recommended by most practitioners and experts in the field". In addition, Laugen et al. (2005) summarized the best practice as *"the basic principle of the best practice thinking is that operations philosophies, concepts and techniques should be driven by competitive benchmarks and business excellence models to improve an organization's competitiveness through the development of people, processes and technology"*. Based on these definitions, it can be concluded the best

practice is a management or technical practice that is widely recognized as effective and excellent practice which is recommended by most practitioners and experts in the field. Fogle (2001) indicated the criteria for identification of the best practices such as:

- **Existence:** at least, the practice must have been observed in one organization.
- **Importance:** the practice is important to an effective process.
- **Effectiveness**: in the practitioner's opinion, the practice should work well where it is used.
- **Tangible benefit:** there is a real benefit to the organization that conducts this practice.

Jones (1996) and Yourdon (1997) argued that many software development projects fail in different ways and it appears that most of them fail because of a combination of several roots such as: inaccurate understanding of end-user needs; inability to deal with changing requirements; modules that don not fit together software that is hard to maintain or extend; late discovery of serious project flaws; poor software quality; unacceptable software performance; ad hoc requirements management; ambiguous and imprecise communication; undetected inconsistencies in requirements, designs, and implementations; insufficient testing; and failure to attack risk. In this regard, Baharom et al. (2006) in their study about the current practices of the software development processes in Malaysia pointed out that the lack of awareness in using good software development practices lead to occurrence of quality problems. Therefore, it is important to treat these root causes by identifying the software development best practices to

develop and maintain quality software in a repeatable and predictable way (Jones, 1996).

Booch (1998) argued that the best practices of software development are: develop software iteratively, manage requirements, use component-based architectures, visually model software, verify software quality, and control changes to software. In addition, the Airline software council which is sponsored by the Department of Defense (DOD) listed to the sixteen software engineering best practices (Brown, 1999), and these practices had been categorized by Software Program Managers Network (SPMN) (Evans, 2001) into three groups, which are:

- **Project Integrity:** Adopt continuous risk management, estimate cost and schedule empirically, use metrics to manage, rack earned value, track defects against quality targets, and treat people as the most important resource.
- **Construction Integrity:** Adopt life cycle configuration management, manage and trace requirements, use system-based software design, ensure data and database interoperability, define and control interfaces, design twice (code once), and assess reuse risks and costs.
- **Product Stability, Integrity:** Inspect requirements and design, manage testing as a continuous process, and compile and smoke test frequently.

Based on the review of literatures on software development best practices of SSDFs, the following practices are the basis software development best practices for SSDFs:

- **Short-Development-Lifecycle:** the time to deliver the project should be short (Al Hussaini, 2006; McDonald & Welland, 2004). The successful development lifecycle should be less than three months, where the short period of development lifecycle can handle the unexpected time pressures.
- **Multidisciplinary Development Team:** it is important for all involved developers to understand their roles during the development lifecycle (El-Sheikh & Tarawneh, 2007; McDonald & Welland, 2004). The development process must include all the required developers to build a successful solution; this will help to know how to resolve conflict in the best interests of the project in question.
- **Maintenance:** the maintenance phase helps in improving software product (Al Hussaini, 2006; McDonald & Welland, 2004). It is certainly necessary for ensuring the proper maintenance and update of the deliverables.
- **Project Management:** project management is very important in the development lifecycle in SSDFs, where it is responsible to ensure that experiments are performed according to defined procedures, while making progress in the context of a schedule and a budget (Baxter et al., 2006; El-Sheikh & Tarawneh, 2007).
- **Delivery of Bespoke Solutions:** it is important to handle the development of software components, the development of data, and the inter-dependencies between them (McDonald & Welland, 2004).
- **Small Software Team:** during the development lifecycle; the development team should be small to avoid arising conflict that will lead to poor development (Al

Hussaini, 2006; McDonald & Welland, 2004). Different small teams of developers need to communicate amongst their peers, where this will help to ensure the consistency and prevent the duplications of effort amongst the team.

- **Requirements and Rigorous Testing Against Requirements:** McDonald & Welland (2004) indicated to the importance of knowing the required issues that are needed to address the process solution, and there is need to test the success of the deliverables in tackling these issues. Furthermore, iterative process is important to help in backtracking to handle the changing of the requirements (Haung et al., 2008).

2.3.3 Difficulties of Implementing SPI Traditional Models and Standards by SSDFs

Both large and SSDFs are faced by problems in managing and improving their software development processes, dealing with rapid technology advances, maintaining their products and operating in a global software environment (Makitalo-Keinonen et al., 2011). SPI traditional models and standards were developed especially for large firms and they need high investment and many other requirements; however most SSDFs could not afford the direct implementation of these models (Alexandre et al., 2006; Garcia et al., 2010a; Baruah, 2012a). Furthermore, these firms suffer from lack of understanding in the success factors of SPI and do not have enough people to perform all the SPI activities (Guerrero & Eterovic, 2004). Therefore, they find themselves to be very far from implementing formal SPI traditional models and standards, and also perceive these models as expensive and time consuming (Cater-Steel, 2004b; Oktaba & Piattini, 2008).

In this respect, Kalpana and Jeyakumar (2011) pointed out the problems faced by SSDFs in implementing the SPI traditional models (i.e. CMMI) such as: excessive documentation; extensive number of specific practices; requirement of extensive resources; high training costs; practices independent of project type; lack of guidance in satisfying project and development team needs; and expensive compliance effort, both in time and money.

In addition, Nawazish Khokhar et al. (2010) indicated that the Critical Barriers (CBs) faced by SSDFs in implementing the traditional SPI models, such as: (1) organizational structure: SSDFs operate with very limited resources in terms people and cost; therefore they lack expertise in the field of SPI; (2) SPI understanding: there are lack of awareness of the basic purpose of process improvement, where they focus on their own priorities for process improvements; and (3) project management: the practices of project management are as per the customer and organization needs, where these firms follow ad-hoc project management in their environment. Furthermore, Ibrahim and Ali (2011) argued that these CBs are: lack of communication; lack of resources; complicated framework; SPI activities gets in the work; and lack of SPI knowledge.

Therefore, it can be concluded that the problem faced by SSDFs in implementing the traditional SPI models and standards can be classified into economic problems and organizational problems as follows:

- **Economic Problems:** SSDFs suffer from the lack of financial support (Xie, 2011), where these firms try to satisfy the customers without the funding to pay

enough attention to the software quality and documentation processes. As such, most of these firms can not improve their software processes, because they do not have the financial support (Zarour, 2009). Furthermore, these firms could not spend some time period to get the benefits of the process improvement by the traditional SPI models, where they are always looking for fast Return Of Investment (ROI) to stay in business (Cater-Steel, 2004a; Ali & Ibrahim, 2010).

- **Organizational Problems:** SSDFs usually operate in flat structure, where they do not have official definition for roles, responsibilities and process (Xie, 2011). Therefore, they can not control the software development process cycle, and this mainly refers to a lack of the needed resources (Zarour, 2009). In addition, they do not have sufficient understanding of currently used software development practices (Jantunen, 2010), where they are using Ad-Hoc manner to develop their software products (Ali & Ibrahim, 2010). Furthermore, these firms mostly depend on the individual skills of their employees instead of a standardized development process, and this is very risky for the survival of these firms (Zarour, 2009). In addition, SSDFs suffer a lack of good experience in SPI and they are not aware of their process capability (Nawazish Khokhar et al., 2010).

Due to the inability of the direct implementation of the SPI traditional models and standards by SSDFs, some regional SPI initiatives have been developed to help these firms in improving their software development processes. Section 2.3.4 presents some of these popular initiatives.

2.3.4 Regional SPI Initiatives for SSDFs

Given the importance of SSDFs and the difficulty of directly incorporating SPI traditional models and standards by these firms; some regional initiatives have been developed to improve the software processes in these firms (Mishra & Mishra, 2009; Garcia et al., 2010b; Baruah, 2012b). Table 2.1 shows the popular SPI initiatives of SSDFs.

Table 2.1: Popular Regional SPI Initiatives of SSDFs

Models/ Methods	Description	Limitations
OWPL (Belgium) (Habra et al., 2008; Stambollian, 2006)	In this model, which is based on the SPICE method, an adviser interviews a stakeholder from the firm in order to identify current issues. This process addresses three core questions: (1) What is the current process?; (2) What improvements need to be made?; and (3) How these changes can be implemented?	- Sometimes, it is difficult to properly collect answers using Micro-evaluation in this model, as the tool's reference grids appear to be very unclear. - This model ignores the development practices, and focuses on some practices of SPICE model. - The reliability of conclusions was based on one interview of 45 minutes; based on the interviewee's vision. Therefore, the user should remain cautious before extensive investment. - This model was evaluated by single team or single project. Therefore, the implementation of the other teams or projects was not evaluated.
ASPE -MSC (Brazil) (Von Wangenheim et al., 2006; Hauck et al., 2008)	This model is based on multiple methods, and engages participation from an outside consulting process engineer (PE), an advisor, and a representative from the firm. This multi-step method first seeks to identify problems based on current needs and processes. The PE is trained, and after these problems are identified, an improvement plan is created and implemented.	- There is insufficient information on the applicability, and tailoring of solution alternatives in certain contexts is available. - This model does not cover the processes of software development. - There is a need for external consultant and assistant for process establishment. Therefore, it is costly for normal small software firms. - There is a need for high experience to assess current capabilities, SPI planning and implementation. - Evaluating this model by two case

		studies requires a lot of additional training and explanations for the representatives of the organization. This occurs because changes are made in the sequence of activities and new templates are created to assist in implementing the model.
PRISMS (Britain) (Allen et al., 2003)	This method seeks to identify the problem as well as provide solutions. The method incorporates the GQM paradigm into the CMM model. A workshop is conducted to develop a plan for implementing improvements and involves participation from an outside consultant as well as a representative from the firm. A web-based self assessment can also be used.	- There is need for significant experience to assess current process, and identify KPAs for improvement. In addition, development and implementation of process improvement plan require experienced persons. - Considerable experience is needed to identify current process model and process improvement plan. - This model helps the developer to identify the weak processes that need to be improved, but without identifying the suitable practices to improve. - This model is based on the improvement KPAs of CMM, where this model is old compared to CMMI.
iFLAP (Sweden) (Pettersson et al., 2008)	This method uses an inductive approach, and can be targeted to improve isolated problems, or a larger process-wide problem. Both a consultant and representative from the firm participate in the process which involves carrying out a series of workshops, where the assessor works with selected representatives from the firm to better understand how the processes are used. Other process documentations are also used to identify problems and solutions.	- This model ignores the management in the improvement steps. - There is need for external expert party in the assessment and planning phases, where this is costly for normal small software firms. - During the evaluation of this model by two case studies, it can be concluded that the project teams have difficulty in separating the required improvement issues that should be addressed. - This model focuses on the process assessment and requires improvement issues, as well as development practices for the required improvement issues.

MESOPYME (Spain) (Calvo-Manzano et al., 2002)	This method uses a CMM model, but emphasizes the creation of *action packages* which can be implemented to solve problems addressed during the assessment phase of the model. The CMM model is used to identify the areas where improvements are needed, and the solutions are developed by quality experts based on their investigations of the firms' processes.	- In this model, there is need for expert assessor to assess the current software process. - The software development team is involved in just implementing the improvement issues, and there is no evidence about who decides which process needs to be improved. - There is a need for expert assistance to develop action packages based on the organizations' business goals and current capabilities. - In the assessment phase, CMM is used to choose one to three processes that need to be improved. This means that the improvement process will be conducted just for the important one to three processes that need to be improved.
MPS (Brazil) (Santos et al., 2007; Boas et al., 2010)	This model was developed based on ISO/IEC 12207 and ISO/IEC 15504, and constitutes three components: PS Reference Model; MPS Assessment Method, and MPS Business Model. The focus of the MPS Model is on small settings, since it provides mechanisms to facilitate SPI implementation of the most critical software processes.	- In the software firms that never follow a process, it was difficult to implement this model without external help. - In evaluating this model, it was difficult for the project teams to adapt their practices as needed by this model. - Most team members were not satisfied and were bored with the way of improving this model. Therefore, some stakeholders, mainly customers, may not be interested in the establishment of formal commitment. - Most of the cases that had implemented this model had reached just the first level of improvement. As such, it is not useful for small firms that have limited resources to follow this model.
MoProSoft (Mexico) (Oktaba et al., 2007; Calvo-Manzano Villalan et al., 2002)	This model was developed for small firms as a first step in achieving CMM level. It helps small firms in the possibility of implementing the SPI practices that were developed for large firms. This model consists of four stages which are: commitment to improvement; software process assessment (CMM); infrastructure and Action Plan; and SPI implementation.	- In implementation of this model, there is a need for expert support and metrics application to manage process evolution, and this is costly for small firms. - In the initial phases of an improvement project, time is wasted due to the uncertainties associated with the new way of working. - This model was developed based on CMM, and there is no evidence of the best practices of software development for small firms being used.

As shown in Table 2.1, there are several initiatives developed to help SSDFs in improving there software process. Based on the characteristics and limitations of these initiatives, the following can be concluded:

- All these initiatives were developed based on the characteristics, environments, and infrastructures of these firms in the specific countries (Mishra & Mishra, 2009; Cruz Mendoza, 2009). Therefore, these initiatives are not suitable for all SSDFs around the world (Isawi, 2011).

- All these initiatives present the KPAs of SPI which are suitable for SSDFs. Nevertheless, they do not support a suitable software development practices to help these firms in adopting these initiatives. Therefore, these initiatives are considered as a simplified of the traditional SPI model to know "what to do" for improvement, but they did not explain "how to do" the improvement.

Accordingly, both of SPI model and software development method which contains software development best practices should be used in constructing the SPI model for SSDFs. This will help these firms to know "what to do" by the SPI model and "how to do" by software development best practices.

2.3.5 Software Process Assessment for SSDFs

Humphrey (1993) defines the software process assessment as "*a diagnostic tool to aid organizational improvement to provide a clear and factual understanding of the organization's state of software practice, to identify key areas for improvement, and to initiate actions to make these improvements"*. In addition, Zarour (2009) argued that the

software process assessment can be used to determine the capability levels or to views the current status of the software process in the software firms. Zahran (1998) indicated that the software process assessment is responsible for understanding and determining the organization's current software engineering practices, and to learn how the organization works; identifying strengths, weaknesses and key areas for SPI; and facilitating the initiation of process improvement activities.

CMMI Product Team (2010) argued that the required assessment is based on the circumstance; sometimes self assessments, initial appraisals, quick-look or mini appraisals, or external appraisals are appropriate; at other times a formal benchmarking appraisal is appropriate. In this respect, Simila et al. (1994) pointed out to the three types of assessments based on who plays the main role in an assessment process, which are:

- **First-party assessment or self- assessment:** this type refers primarily to a situation where the assessment is performed internally inside the software firm to identify the software process capability and initiate an action plan for SPI.
- **Second-party assessment:** in this assessment, there are external assessors are used to assess the software process in the firm to fulfill the specific contract requirements.
- **Third-party assessment or capability determination:** this type is performed by an independent third-party company to evaluate the software process in the software firm in order to enter contracts or produce software products. In addition, it is used occasionally to provide fulfillment of certification according to a selected standard.

Several methods are available to assess the maturity and capability of a software development process based on well-known software process assessment and improvement frameworks such as CMMI and ISO/IEC-15504 (Zarour, 2009). Unfortunately, many researchers consider that the traditional software process assessment models such as CMM, CMMI, and ISO/IEC 15504 are too large to be implemented in SSDFs (Alexandre et al., 2006; BAe, 2007; Mishra & Mishra, 2009). In addition, Cater-Steel (2002) argued that the cost of the formal assessment is beyond the means of most SSDFs, as the lack of resources limit the implementation of these models. Furthermore, Santos et al. (2007) pointed out that it is more important to keep the assessment cost as low as possible for SSDFs with the maximum coverage of relevant processes.

Humphrey (1993) believes that the self-assessments are another form of SEI assessment, where the self-assessment teams are composed of software professionals from the organization being assessed. Therefore, given the limited resources of SSDFs; self-assessment is suitable to be implemented by these firms, where the low-cost mini-assessments are effective for SPI in these firms (Cater-Steel, 2004b). In addition, Von Wangenheim et al. (2004) argued that completing questionnaire through an interview is a suitable technique to assess the current software processes in SSDFs, as most of these firms have a lack of software engineering knowledge. Furthermore, Kalpana and Jeyakumar (2011) pointed out that the self-assessment is suitable to be conducted by SSDFs by using CMMI questionnaires to help these firms in scaling their capability levels in each process area.

In this respect, many researchers have studied process assessment and improvement in SSDFs and tried to develop some assessment methods to be able to assess the software process of these firms, and usually these methods are called "lightweight software process assessment methods" (Zarour, 2009). Some of the popular lightweight assessment methods are: A Methodology for Software Process Assessment in Small Software Companies (MARES) (Anacleto et al., 2004; Von Wangenheim et al., 2004); Toward Organized Process in SMEs (TOPS) (Cignoni, 1999); Fraunhofer IESE Assessment Method (FAME) (Beitz et al., 1999); Rapid Assessment for Process Improvement for Software Development (RAPID) (Rout et al., 2000; Bucci, 2001); Software Process Matrix (SPM) (Richardson, 2001); Express Appraisal Process (EAP) (Wilkie et al., 2007); and Micro-Evaluation (Habra et al., 1999). Table 2.2 highlights some of the popular SPA methods.

Table 2.2: Some of the Popular Lightweight SPA Methods

Methods Criteria	**MARES**	**TOPS**	**FAME**	**RAPID**	**SPM**	**EAP**	**Micro-Evaluation**
Geographic origin/ Spread	Brazil	Italy	Germany	Australia	Ireland	Ireland	Belgium
Scientific origin	ISO 15504	ISO 15504	ISO 15504/ Bootstrap	ISO 15504	Quality Function Deployment	CMMI Compliant with the ARC 1.1	OWPL
Application region	Regional	Regional	Regional	Regional	Regional	Regional	Belgium/ Quebec/ France
Analysis techniques	Interview	Interview	Questionnaire	Interview	Questionnaire	Interview	Short Interview
Assessment duration	1 day	Half a day	NA	1 day	NA	1 day	Half hour
Tool support	NA	Paper forms	Data collection, analyses and rating tools	Paper Forms	NA	Paper forms + data collection	Paper forms + Excel sheet

By analyzing the highlighted SPA methods in Table 2.2, the following can be concluded:

- All of these lightweight assessment methods were developed to assess the software process in SSDFs of certain countries, where the development of these methods was based on the environment of these countries. Therefore, these methods can not be used in other regions.
- Interviews and questionnaires were used as lightweight techniques by these methods to assess the current software process in SSDFs. In addition, cheap support tools had been used in these methods to collect the assessment data.
- The time of assessment process in these methods is always less than one day.

As such, it is important to take into account the use of lightweight technique within short-time period and cheap support tools during the assessment stage in constructing the framework of this study, as SSDFs could not afford high cost of assessment.

As mentioned earlier in Section 1.1, CMMI-Dev1.2 was chosen as a SPI model in constructing the software development process improvement framework for SSDFs in this study. Therefore, Self-Assessment by Pre-Assessment CMMI-questionnaire will be suitable as an assessment technique in the proposed framework of this study to assess the current software development processes in SSDFs, where these firms suffer financial problems that prevent them in conducting the assessment by external party (Cater-Steel, 2004). Section 2.4 presents the history of CMM/ CMMI (CMMs) models.

2.4 History of CMM/ CMMI (CMMs) Models

CMMs models were developed by Software Engineering Institute (SEI) at Carnegie Mellon University to improve the processes in an organization, where these models contain the essential elements of effective processes for one or more disciplines and describe an evolutionary improvement path from ad hoc, immature processes to disciplined, mature processes with improved quality and effectiveness (CMMI Product Team, 2010).

As shown in Figure 2.2, CMM for software V1.1 (1993) is the first release of CMMs, while CMMI-Dev1.3 is the newest release of CMMs which was developed to ensure consistency among all three models and improve high maturity material in all CMMs models. In addition, Figure 2.2 shows that the CMMI is a collection of previous CMM models to sort out the problem of these models. This had been done by combining CMM models into a single improvement framework was intended for use by organizations in their pursuit of enterprise-wide process improvement.

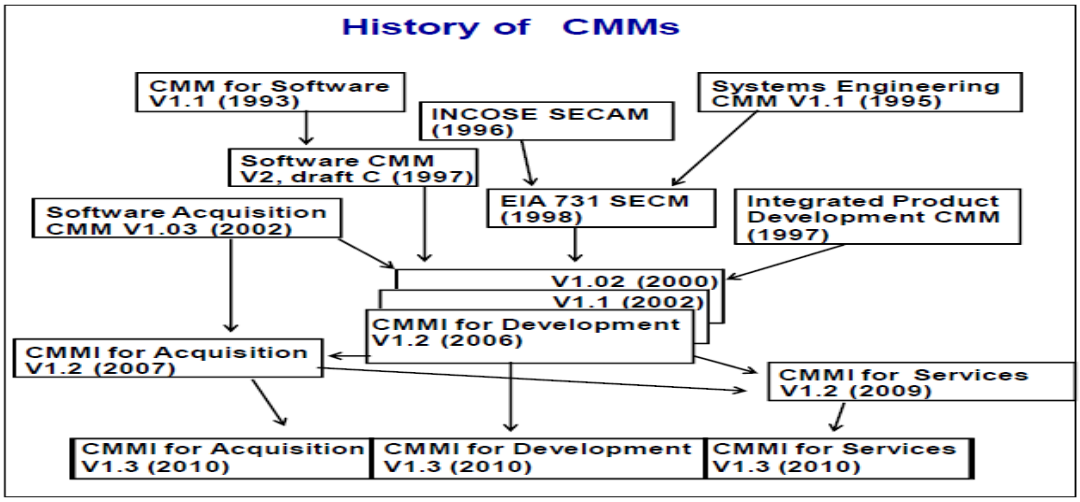

Figure 2.2: CMMs History, adopted from (CMMI Product Team, 2010)

CMMI for Development (CMMI-Dev) has become increasingly important to all aspects of software industry (Pikkarainen, 2008; Alshammari & Ahmad, 2010). In this regard, Bush and Dunaway (2005) indicated that CMMI-Dev has been broadly used for assessing and improving the organizational maturity and process capability throughout the world, where they have confidence in CMMI-Dev because of its extensive descriptions of how the various good practices fit together, as this model improves upon the best practices of other improvement models in many important ways (Goldenson & Gibson, 2003).

Even though CMMI-Dev1.3 (CMMI Product Team, 2010) is the newest version of the CMMI generations, CMMI-Dev1.2 has been chosen in this study as a main element in constructing the software development process improvement framework for SSDFs, as CMMI-Dev1.2 has been broadly used for assessing and improving the organizational maturity and process capability of most software development firms in the world (Mishra & Mishra, 2009; Pikkarainen, 2008). On the other hand, the CMMI-Dev1.3 is a new release and the usage of this model is still scarce (Isawi, 2011). In addition, this study focuses on the KPAs of CMMI-Dev1.2, where these areas are similar to the KPAs of CMMI-Dev1.3 (CMMI Product Team, 2010). Therefore, it can be argued that the developed framework is related also to the KPAs of CMMI-Dev1.3. Section 2.4.1 discusses the CMMI-Dev1.2 model and the reasons of choosing this model in this study.

2.4.1 CMMI-Dev1.2

CMMI-Dev1.2 is a continuation and update of CMMI-Dev1.1 and has been facilitated by the concept of CMMI constellations, where a set of core components can be

augmented by additional material to provide application-specific models with highly common content (CMMI Product Team, 2006). This model consists of four categories of KPAs (CMMI Product Team, 2006), which are:

- **Process Management:** process management areas contain the cross-project activities related to defining, planning, deploying, implementing, monitoring, controlling, appraising, measuring, and improving processes. These process areas are organizational process focus, organizational process definition + integration product and process development (IPPD), organizational training, organizational process performance, organizational innovation and deployment.
- **Project Management:** project management process areas cover the project management activities related to planning, monitoring, and controlling the project. These process areas are project planning, project monitoring and control, supplier agreement management, integrated project management + (IPPD), risk management, and quantitative project management.
- **Engineering:** engineering process areas cover the development and maintenance activities that are shared across engineering disciplines. These process areas are requirements development, requirements management, technical solution, product integration, verification, and validation.
- **Support:** support process areas cover the activities that support product development and maintenance. These process areas are configuration management, process and product quality assurance, measurement and analysis, decision analysis and resolution, causal analysis and resolution.

Generally, CMMI-Dev1.2 is not ready to be used directly by SSDFs (Mishra& Mishra, 2009; Valdes et al., 2011). Nevertheless, several researches (Mongkolnam, 2009; Tosun et al., 2009; Garcia et al., 2010a) indicated that the CMMI-Dev1.2 can be useful and more applicable for SSDFs compared to other SPI traditional models and standards, as these firms could spend their limited resources on the most striking problems to achieve the suitable KPAs of CMMI-Dev1.2.

In addition, there are several reasons for choosing the CMMI-Dev1.2 as a baseline improvement model in this research and these are: (1) CMMI-Dev1.2 has been used to guide the software development improvement (Diez et al., 2007; Galinac, 2008; Garcia et al., 2010a); (2) CMMI-Dev1.2 is the most comprehensive software improvement model and is highly compliance with relevant traditional SPI models and standards (CMMI Product Team, 2006; Mongkolnam et al., 2009); (3) CMMI-Dev1.2 provides a comprehensive integrated solution for development and maintenance activities applied to products and services, and it is considered to be one of the best known models that focuses on SPI for achieving quality software in SSDFs (Garcia et al., 2010a); and (4) CMMI-Dev1.2 is useful for identifying the key weaknesses of a software development processes (Pikkarainen, 2008).

Given the comprehensiveness, popularity, and advantages of the CMMI-Dev1.2 compared to other SPI traditional models and standards, this model was chosen in this study as a baseline model from the aspect of improvement models to develop the software development process improvement framework for SSDFs. On the other hand,

there is a need for suitable software development method to be combined with CMMI-Dev1.2 to help in developing the desired framework. Section 2.5 explains the popular software development methods and the reasons for choosing XP method as a suitable development method to be used in constructing the desired framework.

2.5 Software Development Process Models

Software development process model is "*an abstract representation of a process that presents the description of a process from some particular perspective"* (Sommerville, 2007). In addition, Bell (2001) explains that the software process model as "*a plan of action for software development with its requirements, tools and steps to create the software product"*. Boehm (1988) argues that the main objective of the software development process models is to identify the order of stages for software development and evolution by establishing the transition between the steps of development. Based on these definitions and the objective of the software development model, this software process model can be defined as a systemic plan which contains all the required components to describe the way of developing the software product.

There are many classifications of software development process models based on different authors' perspectives (Kuhlmann, 2003). In general, the software development process models can be classified into two main groups; traditional and agility models (Preuninger, 2006; Sommerville, 2007; Pressman, 2009). Traditional software development process models (plan-driven approach) such as waterfall (Royce, 1987) and spiral (Boehm, 1988) are more suitable for large software development firms, where agile methods (Beck, 2000) are more applicable for SSDFs. Therefore, it is useful to use

a lightweight software development method in developing the desired framework of this research such as agile method (Fruhling & Vreede, 2006). Section 2.5.1 discusses the popular agile methods and the reasons of choosing XP method in this study.

2.5.1 Agile Methods

Agile software development methods represent new approaches for planning and managing software projects. Agile development differs from the traditional plan-driven approaches as it puts less emphasis on up-front plans and strict plan-based control and more emphasis on mechanisms for change management during the project (Cockburn & Highsmith, 2001). The emergence of agile methods began in the mid 1990s, and these methods are considered the newest for software development methods (Salo, 2006). Figure 2.3 shows the evolution of software development models.

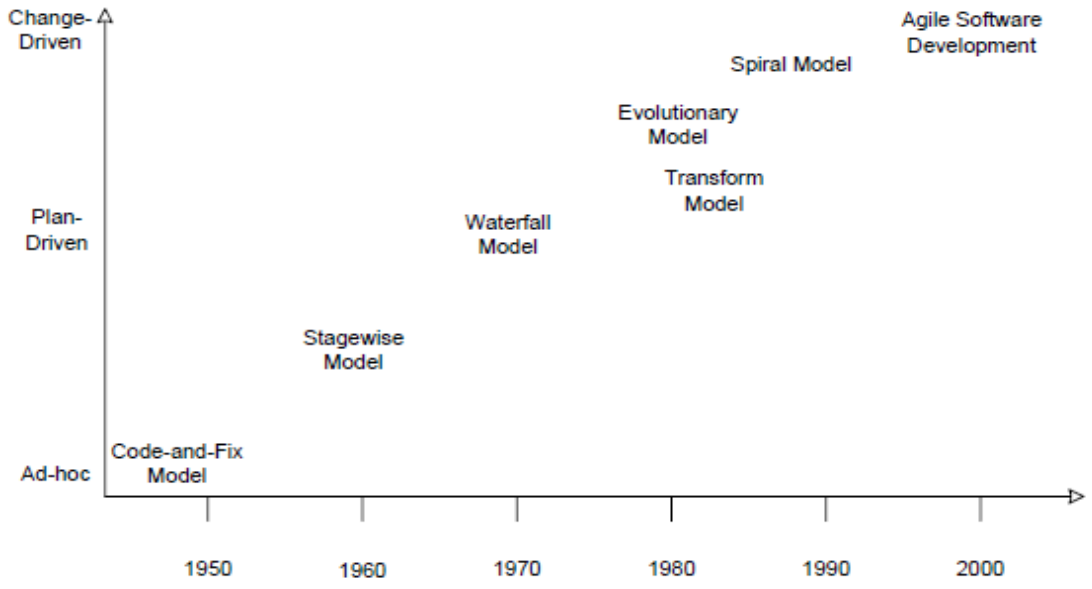

Figure 2.3: The Evolution of Software Process Models, adopted from (Salo, 2006)

Agile development methods are designed to address the problem of delivering high-quality software on time under constantly and rapidly changing requirements in business and IT environments (Stojanovic et al., 2003). Furthermore, these methods help to solve several critical problems faced by software projects, such as: (1) schedule slips: the software is not ready when the deadline comes; (2) project cancelled: projects are cancelled after a long period without ever going into production; (3) systems go sour: the defect rate increases after the system has been put into production; (4) defect rate: the defect rate of the software product is so high that it is never used; (5) business misunderstood: the software never solves the business problem for which it was originally posed; and (6) false feature rich: the software has many features which are fun to program but which do not have any added value from a customer perspective. These problems can be solved by following the manifesto of agile software development. It provides advice on how to focus on the development on people, working software, customer collaboration and increase an organizations ability to respond to changes (Beck, 2000).

Agile methods offer many approaches to improve the software development process (Karlstrom & Runeson, 2006). According to Pressman (2009), Abrahamsson et al. (2002), and Xu et al. (2003), the popular agile methods are Extreme Programming (XP), SCRUM, Crystal, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), and Feature-Driven Development (FDD). Table 2.3 highlights the general information and the scope of using agile development methods. Even thought both XP and SCRUM methods are the two popular and effective agile development

methods (Beck, 2000; Abrahamsson et al., 2002; Boehm, 2006), however just XP method has been chosen in this study as a software development method in constructing the software development process improvement framework for SSDFs because XP method is considered as a more compatible software development method for CMMI model compared to other agile methods such as SCRUM (Erharuyi, 2007; Fritzsche & Keil, 2007).

Table 2.3: Comparison of Agile Development Methods, expanded from (Abrahamsson et al., 2002)

Methods	General Information	Scope of Use
XP	XP is the most lightweight popular method in agile software development methods and has some characteristics such as customer-driven, frequent release, pair programming.	Good for small and medium size team, 3—20 people.
SCRUM	SCRUM from the popular agile software development methods that focus on agile project management. This method derived from the strategy of rugby game.	Suitable for small team. < 10 people.
CRYSTAL	A set of methods. Suggest development cycle within 4 months. Emphasis on communications, and allow adoption of other agile methods.	Not good for life-critical system. Up to 40 person's local development.
ASD	Emphasis on incremental, iterative development.	Focus on developing large system. No built-in limitation.
DSDM	Application of controls to RAD. Emphasis on time and resource.	Team size between 2 and 6, multiple teams exist. Can be used in large system, if the system can be splitted into components.
FDD	Focus on design and building phase. Emphasis iterative development. Needs other supporting approaches.	It is suitable for the development of large software project.

XP is popular in agile software development methods as the life cycle in XP can be executed quickly compared with other traditional methods like waterfall (Alite &

Spasibenko, 2008). Figure 2.4 shows the flexibility (how they accept change) and quality (defects and accuracy of the product) of XP method compared to other software development methods.

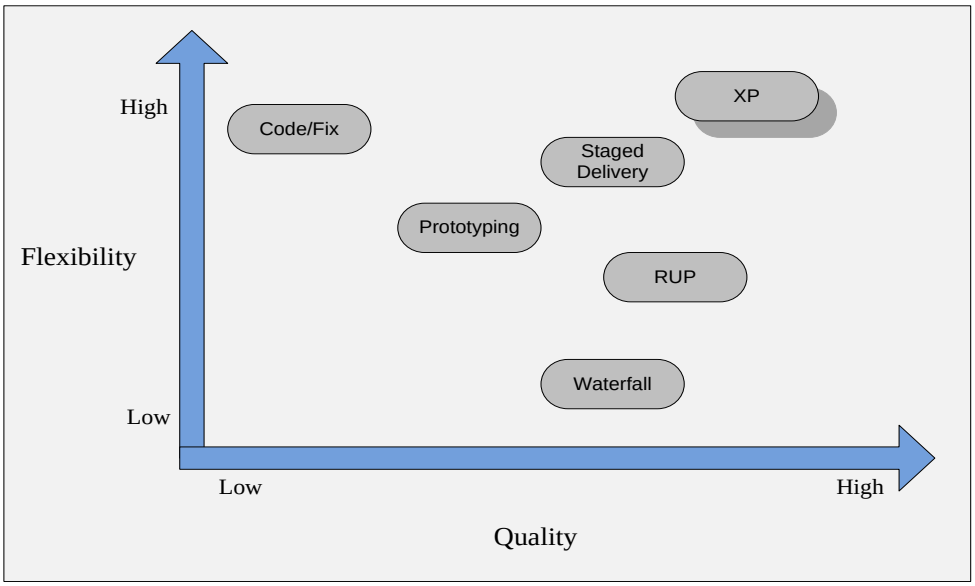

Figure 2.4: Comparison of the Methodologies, adopted from Baird (2002)

XP method involves the customers from the beginning of process to help understand the desired requirements, and uses pair programming to reduce the number of mistakes and share the knowledge between the team members (Loftus & Ratcliffe, 2005; Preuninger, 2006). Furthermore, XP does not require a lot of tools through the development stages (Stojanovic et al., 2003), and it also implements the quality assurance practices through the iteration (Nawaz & Malik, 2008). Therefore, the XP can be more useful and effective in SSDFs (Alite & Spasibenko, 2008; Beck, 2000; Abrahamsson et al., 2002).

In addition, there are other reasons for the choosing of XP as a software development method in this study such as: XP is better applicable for small, medium-scale and less complex projects and it is the most widely used agile methods and also one of the most prominent approaches that adheres to agile principles; XP practices work tightly together by carefully applying different practices over time that will eventually lead to improvement; and XP is an easy model for learning; XP can be easily adapted with changing requirements (Lindvall et al., 2004; Alegr & Bastarrica, 2006; Altarawneh & Shiekh, 2008). In addition, Fritzsche and Keil (2007), Pikkarainen (2008), and Erharuyi (2007) argued that XP is the lightweight process model that can help SSDFs in the implementation of SPI, and they believed that XP achieves SPI better than other agile methods as it conforms to level two in CMMI, while SCRUM only conforms just to level one in CMMI. Furthermore, Anderson (2005), and Fritzsche and Keil (2007) argued that the CMMI-Dev1.2 level 5 would be possible to be achieved by extending XP method.

2.5.2 Extreme Programming (XP) Method

XP is an agile method originally presented by Beck (2000) and it is the most popular method in agile software development methods (Jeffries et al., 2001). Regarding the flexibility and agility of XP; this method is called extreme, whereas it can take good things to develop the software and applies these things extremely (Beck, 2000). In addition, this method is used for business where time is important, when risk of a long project can not be taken, when requirement are not known earlier (Devesh et al., 2011). XP is a lightweight method with four key values (Beck, 2000) as follows:

- **Communication:** XP facilitates correct communication, which is needed to employ the defined XP practices.
- **Simplicity:** the team's goal of implementing software remains as simple as possible. This value is also connected to communication. If the code is simple, it is also easier to communicate to other people.
- **Feedback:** this value mainly relates to customer collaborations. It means that the team should receive concrete feedback on their work on a daily, weekly or monthly basis. This value also has a strong relation to communication. For example, Beck (2000) argues: "*the more feedback you have, the easier it is to communicate"*.
- **Courage:** this value helps to solve new technical challenges and to make new innovations. Moreover, communication value facilitates courage in teams because it opens the opportunity for new technical experiments.

Based on these values, Beck (2000) created the five core principles of XP method, which are:

- **Rapid Feedback:** Rapid feedback facilitates rapid responds thereby improving the design, coding and system delivery target date. Regular feedback and responses create room for system improvement by constant steering and reviewing changes. Thus, rapid feedback brings simplicity and lows the cost change (Hightower, 2004).
- **Assume Simplicity:** Assume simplicity requires designs or coding be concerned with current needs instead of bothering your self in design to take of future

needs; it requires design to be tailored to current iteration alone as customer requirements changes; and it is advisable to design to accommodate changes per iteration. Hightower (2004) defines assuming simplicity as *"treating every problem as a simple problem until proven otherwise".*

- **Making Incremental Changes:** Changes should effect gradually and performed do it step by step. Incremental change creates feedback that allows learning and improvement before taking another step, thereby minimizing risk. Hightower (2004) argued that incremental change fits well with simplicity and do not over-design a system.
- **Embrace Change:** XP developers should always expect change and be ready to embrace it. It becomes very easy to embrace this change because XP delivers business values to customer incrementally. This creates room for customer to request for change and furnishes with feedback.
- **Do Quality Work:** This principle encourages delivery of quality code or system that meets customer's needs. Once customer's needs are satisfy, it brings happiness to all stockholders, not just happiness but more business.

In detail, Sections 2.5.2.1 to 2.5.2.5 discuss XP phases, XP practices, XP roles, the strengths and weaknesses of XP method, and the coverage of software development basic best practices of SSDFs by XP method.

2.5.2.1 XP Phases

XP method consists of six separate phases. Figure 2.5 shows the life cycle of XP method. According to Beck (2000), Larman (2003), and Coram and Bohner (2005), these phases are:

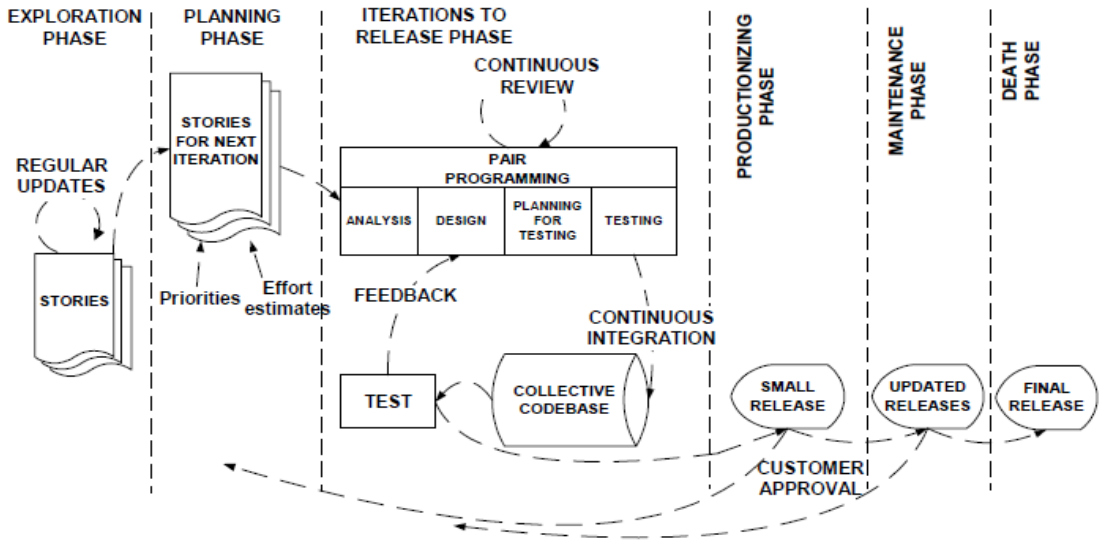

Figure 2.5: XP Life Cycle, adopted from Abrahamsson et al. (2002)

- **Exploration Phase:** in this phase; the customer writes "story cards" to describe the required features that are needed to be added into the program, where each story card contains one feature. Then, the developers familiarize themselves with the tools, practices and technologies that are going to be used in the project. After that, the team of developers tests the technology and also they develop a prototype to explore the architecture possibilities. This phase takes from few weeks to few months, depending on how well the programmers know the technology.

- **Planning Phase (known as planning game)** in this phase; developers estimate for each card how long it would take to implement this card and based on these estimations, customers and developers decide together about the prioritization of each card and agree together about the contents of the first release, and also the schedule for each of the features. Therefore, a release plan/schedule is finally set up which says which feature will be implemented in each release. This phase takes a couple of days and the first release usually takes no more than two months.
- **Iterations to Release Phase:** this phase includes several iterations before the first release (the schedule set in the planning phase is broken down to a number of iterations, these iterations create one or more functions of the system in each one of them), where each iteration takes one to four weeks to implement. Furthermore, the design as well as the coding is done, but before any line of code is written, first a unit test to test these lines has to be developed by the programmers. In the first iteration, a system with the architecture of the whole system is created. In addition, customer's functional tests are run at the end of each iteration. Finally, as soon as the developed features are tested by the developers (probably by automated unit tests), they are given to the customer and thereby, the next phase is entered. After the last iteration the system is ready for production.
- **Productionizing Phase:** this phase consists of extra testing and performance checks, where the customer performs functional tests and validates if the product works as intended. Then, if new requirements are elicited, they are either

included directly or a new story card is created which will be considered in the following release planning. Furthermore, new changes may be found and they might still be included in the current release. Finally, the postponed ideas and suggestions are documented for the later implementation e.g. in the maintenance phase.

- **Maintenance Phase:** after the first release is productionized and taken into use, the XP project has to keep the system running whilst implementing new features. This requires the effort of the customer support tasks also, which may decelerate the implementation pace of the new features. Moreover, the customer is supported by (probably new) team members whose task is to ensure that certain customer requests for, i.e. improvement suggestions are considered. The maintenance phase may require incorporating new people into the project team and changing the team structure.
- **Death Phase:** in the final phase, the system will undergo the final release; or the system will be broken down for some reasons such as when the customers do not have new stories to implement or when the system can not satisfy the customers needs, as well as when the system is too expensive for modification.

2.5.2.2 XP Practices

XP method is extreme in the sense that it takes many well-known software development best practices drawn from already existing development methodologies (Agarwal & Umphress, 2008). According to Beck (2000) and Jeffries et al. (2001), it can be concluded that the XP consists of twelve practices as follows:

- **Planning Game:** this practice means a set of rules and moves that may be used to simplify the release planning process, and it is closed interactions between customers and programmers. Planning game can be divided in two parts which are: (1) Release Planning: in this part the customer defines what kind of features are wanted in the product and prioritize them, then programmers make an estimation of each feature. The initial estimations could not be so accurate, but then with continuous iterations and reviews, they become more accurate, i.e. priorities and estimations. When the priorities and estimations are added to each feature, a release plan for the project can be done; and (2) Iteration Planning: this is when the customer and programmers meet together to deliver working software every two weeks. The level of detail is bigger than the release plan, the customer shows which of the features of the release plan he or she wants for the next two weeks. Furthermore, programmers divide the features in tasks and estimate their tasks, whereas the first tasks from the customer's side is to determine the scope of the project, priority of the features, composition of releases, dates of releases, and the second tasks from the programmer's side is to determine the estimations of the features, technical consequences, process, and detailed scheduling.
- **Small Releases:** this practice means all releases should be a small as possible, but with the maximum quantity of business features developed, whereas short cycles are used to reduce the risk when a project fails to produce business value to the customer, and also helps in reducing planning problems and the problem with changing requirements during the development process. Moreover,

frequency is important as well depending on which kind of software is delivered. At the end of every iteration; software is visible, and given to the customer.

- **System Metaphor:** both the customer and the programmers share a story based on a metaphor that guides all development by describing the functionality of the system. Additionally, the team shares some common understanding from their past experiences. A metaphor should helps everyone on the project to understand the basic elements and their relationships, where metaphor is similar to what other people call `an architecture', but with the addition that requires the XP team to follow some way of cohesion.
- **Simple Design:** the design should be kept simple through the developments, using the developer's test-driven development and re-factoring, whereas XP fits the design for the present system features ready for future changes in an incremental or iterative way. Therefore, XP design should begin without thinking of infrastructure, where the right design in XP can run all the tests, has no redundancies, and has the fewest possible classes and methods Moreover, XP focus on solving today's problems and every piece in the design must be able to justify its existence.
- **Design Improvement (Formerly Re-factoring):** re-factoring is a process of changing a software system in such a way that it does not alter the system behavior of the code yet improves it internal structure. Doing design improvement in an XP project is a practice where the programmers delete duplicate codes. In addition, programmers should increase cohesion and decrease coupling. Therefore, re-factoring should be made when there is something wrong

in the code, such as: classes that are too long, methods are too long and duplicate codes. Moreover, design improvement should be done every hour or half hour, followed by testing of what was done and this is done to keep the design as simple as possible at all times. Accordingly, the changes of the structure are verified with automated tests which help the programmers to get feedback on the changes.

- **Test-Driven Development (Programmers Tests + Customer Tests):** Testing is an essential part of XP; especially the automated tests, a feature without an automated test does not exist. The programmers write the unit tests and the customer writes the functional tests. Test-driven development can be divided in two parts, which are: (1) Programmer Tests: programmers should create the tests first and then code. The first test should fail, because no codes have been created, and then the programmers should create the code to pass the test, and then turn the cycle to add one more test followed by the code. One of the benefits of extreme programming is that 100% of the code is tested; and (2) Customer Tests: each user story that represents a feature in the XP development has an associated acceptance test that is determined by the XP customer and implemented by the team. Moreover, the correctness of the systems is shown to the customer when all tests are passed. Consequently the application is continually growing and evolving.
- **Pair Programming:** the production of codes is written with two people using one computer. One of them has control of the keyboard/mouse and creates the code, and the other is continuously assuring quality by watching, trying to

understand, asking questions, looking for alternative approaches, and helping to avoid defects. If pairs are switched through the team knowledge is shared to everyone working in the XP team. Therefore, individual's skills are improved because the pair should switch at least once per day.

- **Collective Code Ownership:** everybody in a XP project takes responsibility for the code in the whole system. Any improvements or new ideas can be added anywhere in the code, where this can be made partly due to the automated tests in XP. Moreover, unknown repercussions will be detected by the automated tests and the programmers can modify the code more freely. Therefore, this practice increases quality of the code and reduces faults.
- **Continuous Integration:** changes to the code are integrated at least once a day. The pair programmers are responsible for integrating their own code and automated tests are run to ensure that the system is working at 100 %. If the tests fail, the pair can undo their changes and start over. Therefore, this practice keeps the system never far from a production state. Moreover, the pair should check that their changes do not affect another part of the system developed by another pair of programmers. In addition, one machine can be used only for integration issues for one pair of programmers.
- **On-site Customer:** a customer needs to be available to determine and prioritize the requirements. This is one of the few requirements in XP and it helps to improve the software business value. However, the programmers can get input from the customer immediately instead of speculating. Quick changes to the focus of the development can also be made when necessary

- **Coding Standards:** coding rules exist and are followed by the programmers. Therefore, this practice keeps the code consistent and easy for the entire team to read. Re-factoring and all the codes in the system look coherent and harmonious. Furthermore, this practice helps the XP team to understand all the codes that have been written as basis for the practice of collective ownership.
- **Sustainable Pace (Formerly 40-Hour Weekly):** this practice means that the team members work hard at a pace that they can go along with for the time being. However, overtime is a symptom of a serious problem in an XP project.

2.5.2.3 XP Roles

There are several different roles with different tasks and purposes in XP method. These roles are used during the software process. Beck (2000) classified seven roles in an XP method as follows.

- **Programmers:** XP programmer should practice design improvements, simple design, learn pair programming, test-driven development, make the estimations from the use stories, and determine what are the tasks that should be undertaken in order to develop each use story. Furthermore, programmers write test and keep the program code as simple and definite as possible. However, the first issue of making XP successful is to communicate and coordinate with other programmers and team members.
- **Customer:** customer is responsible for writing use stories, defining the customer tests, writing the functional tests and determining the priorities for each use stories that should be explained to the team. During XP lifecycle; the customer is

important to answer any questions about the user stories and to verify the system, and also he/she decides when each requirement is satisfied.

- **Tester:** tester helps the customer to choose functional tests. He is also responsible for implementing and running the functional tests. Furthermore, he/she executes the integration tests and makes some graphs, in a manner to show the XP team the results.
- **Tracker:** he/she estimates the project velocity and uses the feedback from the programmers by asking and listening to what they are doing in the current moment. He/she should be careful to not interrupt the project too many times. A tracker should be able to tell if anything needs to be changed to follow the current plans or a new plan is needed. He/she also traces the progress of each iteration and evaluate whether the goal is reachable within the given resource and time constraints or if any changes are needed in the process.
- **Coach:** coach is responsible for the project as a whole. Accordingly, he/she ensure that the project goes along the right path by keeping people working on the current features for the actual iteration. However, coach is responsible to identify what practice might help when problems occur, what the ideas behind XP are, and how to relate these to the project. Thus, a coach should help the team to reveal mistakes without too much interference and steering. Therefore, a coach should have a good comprehension of XP method and the project than the rest of the team.

- **Consultant:** this role is represented by an external member possessing specific technical knowledge in a specific area, where the consultant is responsible for helping the team to solve their problems if there is need for external knowledge.
- **Big Boss:** big boss is responsible for checking of the team performance, and explains to the team if there is need to change something and explains why that change is needed. Furthermore, big boss makes the decision. Therefore, a big boss communicates with the XP team to determine the current situation, and to distinguish any difficulties or deficiencies in the process. If an XP team does not produce what they should, a big boss can step in and help them.

2.5.2.4 Strengths and Weaknesses of XP Method

Many researchers indicate the strengths and weaknesses of XP method. Based on these literatures, Table 2.4 summarizes the common strengths and weaknesses of XP.

Table 2.4: Strengths and weaknesses of XP method

Strengths of XP Method	**XP method helps the software industry for shorter release of functional software, where the customers are always contacted to ask for the highest priority features in the software.**	**Beck, 2000; Fruhling & Vreede, 2006; Xu, 2009.**
	XP method saves the project against the cancellation with the help of periodic releases.	Beck, 2000; Guha et al., 2011.
	XP method always focuses on the highest priority tasks; therefore false features are not prioritized during the development of the software, as it gives the freedom to the developers and testers to give their feedbacks upon the release time and cost of the software which will helpful for interaction with the clients via the business people.	Beck, 2000; Munassar & Govardhan, 2010; Xu, 2009.
	XP method is more flexible and includes more explicitly the needs and intentions of all project participants.	Beck, 2000; Fruhling & Vreede, 2006; Xu, 2009.

	By test driven development practices, XP method resulting in less errors and acceptance of changing requirements.	Beck, 2000; Fruhling & Vreede, 2006; Munassar & Govardhan, 2010.
Weaknesses of XP Method	XP method is suited for single project, developed and maintained by a single team. It cannot be implemented in the system where developers don't work well with each other and like to work on their own.	Beck, 2000; Guha et al., 2011; Hneif & Hock Ow, 2009.
	XP method is not suitable for medium and large scale projects.	Munassar & Govardhan, 2010; Mushtaq, 2012; Hneif & Hock Ow, 2009.
	XP method is not suitable to be implemented in an environment where a customer or manager insists on a complete specification or design before they begin programming.	Beck, 2000, Turk et al., 2002; Xu, 2009.
	Lack of project management practices.	Beck, 2000; Turk et al., 2002; Mushtaq, 2012.
	Lack of documentation though the development lifecycle.	Qureshi, 2011; Munassar & Govardhan, 2010; Guha et al., 2011; Paulk, 2001.
	Developers must be experienced.	Paulk, 2001; Munassar & Govardhan, 2010.

2.5.2.5 Coverage of XP Practices to Basic Best Software Development Practices of SSDFs

Based on the basic software development best practices of SSDFs that are discussed in Section 2.3.2 and based on the descriptions of XP method practices and roles that are discussed earlier in Sections 2.5.2.2 and 2.5.2.3, it can be concluded that the XP method is consistent with the software development basic best practices of SSDFs as shows in Table 2.5.

Table 2.5: Coverage of Software Development Baisc Best Practices in SSDFs by XP Practices and Roles

Basic Best Practices	XP Practices and Roles
Short-Development-Lifecycle	Small releases practice (short iterations and short releases).
Multidisciplinary Development Team	Collective code ownership and system metaphor practices. Tracker and coach roles.
Maintenance	Small releases, on-site customer, and test driven development practices. Coach and tracker roles.
Project Management	Planning game practice (release planning and iteration planning). Tracker and coach roles.
Small Software Team	The XP team is between 3-20 persons.
Delivery of Bespoke Solutions	On-site customer, planning game (iteration planning), continuous integration, and small releases practices.
Requirements and Rigorous Testing Against Requirements	Test driven development (programmer's tests and customer tests), small releases, and on-site customer practices.

2.6 The Relationship between CMMI-Dev1.2 and XP Method

Several studies discussed to what extent the CMMI-Dev1.2 KPAs can be covered by XP method (refer to Appendix A, CMMI-Dev1.2 KPAs) such as: Fritzsche and Keil (2007), Omran (2008), and Elshafey and Galal-Edeen (2008). In conducting the coverage of XP practices to the CMMI-Dev1.2 KPAs by these studies, five scales were used by Fritzsche and Keil (2007), while three scales were used by Omran (2008), and Elshafey and Galal-Edeen (2008). Table 2.6 shows the scales used by these studies.

Table 2.6: Scales of Coverage XP Practices to CMMI-Dev1.2 KPAs

References	Scale of Comparison
Fritzsche & Keil (2007)	• Conflicting (–):XP practices can not cover the process area's components • Not addressed (0): XP practices do not cover the process area's components. • Partially supported (+): XP practices satisfy some of the process area's components. • Supported (++): XP practices satisfy most of the process area's components. • Largely supported (+++): XP practices satisfy the major part of the process area's components.
Omran (2008)	• (++): process area is largely addressed by XP practices. • (+): process area is partially addressed by XP practices. • (--): process area is not addressed by XP practices.
Elshafey & Galal-Edeen (2008)	• Supported (S): when most parts of the process area is supported by XP practices that will help enhance or accelerate its implementation. • Partially Supported (P.S): when only a small part of the process area is covered by an XP practice, it can't help implementing this process area on its own other non XP practices will be needed. • Not Supported (N.S): when process area is not addressed by XP method.

As shown in Table 2.6, it can be concluded that the descriptions of these scales focus on common three levels, which are:

- Largely Support (L.S): XP practices largely support the specific goals of the KPA.
- Partially Support (P.S): XP practices partially support the specific goals of the KPA.
- Not-Support (N.S): XP practices do not support or not applicable for the specific goals of the KPA.

Based on the common three levels, Table 2.7 unites the different scales which used by these studies into common three levels. Accordingly, Table 2.8 presents the comparisons results of the three studies based on the common three levels (refer to Appendix B, detailed coverage results of XP practices to CMMI-Dev1.2 KPAs).

Table 2.7: Scale of Coverage XP Practices to CMMI KPAs

	References		
Three Scales	**Fritzsche & Keil (2007)**	**Omran (2008)**	**Elshafey & Galal-Edeen (2008)**
Largely Supported (L.S): XP practices largely support the specific goals of the KPA.	(+++)	(++)	Supported
Partially Supported (P.S): XP practices partially support the specific goals of the KPA.	(++) OR (+)	(+)	Partially Supported
Not Supported (N.S): XP practices do not support or not applicable for the specific goals of the KPA.	(-) OR (0)	(--)	Not Supported

Table 2.8: Coverage Results of XP Practices to CMMI-Dev1.2 KPAs

CMMI-Dev1.2 KPAs	Fritzsche & Keil (2007)	Omran (2008)	Elshafey & Galal-Edeen (2008)
Requirement Management	L.S	L.S	L.S
Project Planning	L.S	L.S	L.S
Project Monitoring and Control	L.S	L.S	L.S
Supplier Agreement Management	N.S	N.S	N.S
Measurement and Analysis	P.S	L.S	P.S
Process and Product Quality assurance	P.S	P.S	N.S
Configuration Management	L.S	P.S	P.S
Requirements Development	P.S	L.S	P.S
Technical Solution	L.S	L.S	L.S
Product Integration	L.S	L.S	P.S
Verification	L.S	L.S	L.S
Validation	L.S	L.S	L.S
Organizational Process Focus	N.S	P.S	N.S
Organizational Process Definition +IPPD	N.S	P.S	N.S
Organizational Training	P.S	L.S	P.S
Integrated Project Management +IPPD	P.S	P.S	P.S
Risk Management	L.S	P.S	L.S
Decision Analysis and Resolution	N.S	P.S	N.S
Organizational Process Performance	N.S	P.S	N.S
Quantitative Project Management	N.S	N.S	N.S
Organizational Innovation and Deployment	N.S	P.S	N.S
Causal Analysis and Resolution	N.S	P.S	N.S

Legend:

- Largely Supported (L.S): XP practices largely support the specific goals of the KPA.
- Partially Supported (P.S): XP practices partially support the specific goals of the KPA.
- Not-Supported (N.S): XP practices do not support or are not applicable for the specific goals of the KPA.

As shown in Table 2.8, it can be concluded that there are nine KPAs of CMMI-Dev1.2 that have the same coverage by XP practice in the three studies, which are: requirement management (L.S), project planning (L.S), project monitoring and control (L.S), supplier agreement management (N.S), technical solution (L.S), verification (L.S), validation (L.S), integrated project management +IPPD (P.S), and quantitative project management (N.S). Nevertheless, the remaining thirteen KPAs have different coverage by XP practices in the three studies, which are: measurement and analysis, process and product quality assurance, configuration management, requirements development, product integration, organizational process focus, organizational process definition +IPPD, organizational training, decision analysis and resolution, risk management, organizational process performance, organizational innovation and deployment, and causal analysis and resolution.

2.7 Conclusion

Software process is a set of tools, practices, and methods to produce software products. SPI is a systemic procedure for improving the performance of an existing process system by changing or updating the process. There are several popular SPI traditional models and standards which were developed for large and very large firms such as: CMMs, ISO 9000 series, BOOTSTRAP, ISO/IEC 12207 and SPICE. These SPI models and standards are difficult to be directly implemented within the context of most SSDFs. This is important as SSDFs represent a high proportion of software firms in most countries all over the world.

SSDFs firms have different characteristics compared with other larger software firms especially in terms of the number of employees, capital, methods, techniques, and software development activities. SSDFs suffer from the lack or recourses, lack of control, lack of project management, lack of awareness of the well-defined development processes, and lack of risk management. Due to these obstacles, SSDFs could not implement the SPI traditional model and standards directly. Therefore, some regional SPI models for SSDFs were developed to help these firms in improving the software development processes. However, all of these models were developed to be suitable for specific regions and they can not be global for all the SSDFs around the world, and also do not support the SSDFs with the software development practices.

In the term of SPA, there are several popular lightweight assessment methods which were developed for SSDFs. Based on the descriptions of these method, it can be concluded that the self-assessment by pre-assessment is a suitable technique to be used for SPA in SSDFs, as these firms focus on low-cost mini-assessments as they cannot afford the cost of the external party to assess their software processes.

CMMI is the comprehensive and newest software improvement model of the SEI where this model complies with relevant traditional models and standards. CMMI-Dev1.2 provides a guideline for improvement for the software process in the organizations, and it is written specially for the software industry to describe the software processes in details. Therefore, CMMI-Dev1.2 model is used as a baseline model in this study.

Software development process model is an abstract representation of a process that presents the description of a process from some particular perspective and there are many representations of these models that aim to develop the software products. Some researchers classify these models according to their own perspectives. This chapter categorized these models into two main groups; software development process traditional models and agility methods models. XP was used as a baseline software development method in this study because it is suitable for SSDFs and more compatible to CMMI-Dev1.2 compared to other software development models.

This chapter also discusses several studies that indicate the coverage ratios of XP method to the KPAs of CMMI-Dev1.2. Based on these studies, it can be concluded that there are differences in the used scales between these studies, some of the results of these studies are dissimilar, and the XP practices support some of the KPAs of CMMI-Dev1.2. In addition, these studies can be used to identify to which extent the KPAs of CMMI-Dev1.2 can be achieved by XP practices.

Therefore, there is a need to have a software development process improvement framework for SSDFs to know “what to improve”, and “how to improve” the software development processes of these firms. CMMI-Dev1.2 and XP method are chosen to be integrated to achieve the main goal of this study. Chapter 3 shows how to construct the desired framework for SSDFs.

CHAPTER THREE
RESEARCH METHODOLOGY

This chapter describes the methodology used in this research to achieve the research objectives. It explains the stages used in constructing of the software development process improvement framework which include: aligning XP practices to CMMI-Dev1.2 KPAs; tasks of developing the proposed framework; tasks of verifying the proposed framework; and the two approaches used in validating this framework. In addition, this chapter presents the adaptation of the Extension-Based Approach (EBA) which was used to extend the XP method. Finally, it presents the focus group method and Delphi technique that were used in the verification process.

3.1 Introduction

Research methodology comprises the methods and techniques used by researchers in carrying out the research; for example: data collection techniques, and data processing techniques and instruments (Kothari, 1985). Ramsin (2006) used the stages strategy to develop software modeling analysis methodology. In his research strategy, each stage has goals and tasks to achieve the overall goals. In this research, stages strategy is useful and suitable because the objectives of this research are to be achieved sequentially in stages. Therefore, the methodology of this research is used in four stages, which aim to construct a suitable software development process improvement framework for SSDFs. This chapter discusses the four stages, starting with aligning XP practices to the specific goals of CMMI-Dev1.2 KPAs as Stage One. It explains the steps of developing the proposed framework based on extending XP method and the generic elements of SPI in

Stage Two. Pursuant to Stage Two, this chapter also explains the verification process of the proposed framework in Stage Three. Finally, this chapter describes the process used to validate the suitability and applicability of the modified framework for SSDFs in Stage Four. Sections 3.2 to 3.5 elaborate the four stages in detail.

3.2 Stage One: Aligning XP Practices to the Specific Goals of CMMI-Dev1.2 KPAs

This stage aimed to identify the coverage ratio of XP method to CMMI-Dev1.2, based on aligning the XP practices to the specific goals of CMMI-Dev1.2 KPAs. This alignment was based on the specific goals of CMMI-Dev1.2, because all the generic goals are repetitive throughout the specific goals (Vasiljevic & Skoog, 2003; CMMI Product Team, 2006). Pikkarainen (2008), in his work on the same field, used the same way to map four KPAs of CMMI-Dev1.2 to agile practices to develop his framework. Accordingly, the specific practices of each specific goal of CMMI-Dev1.2 KPAs were used as main items in aligning XP method to CMMI-Dev1.2. As a result of this alignment, the coverage and missing specific goals of each KPA was known and used as inputs in Stage Two. Figure 3.1 shows the steps of Stage One.

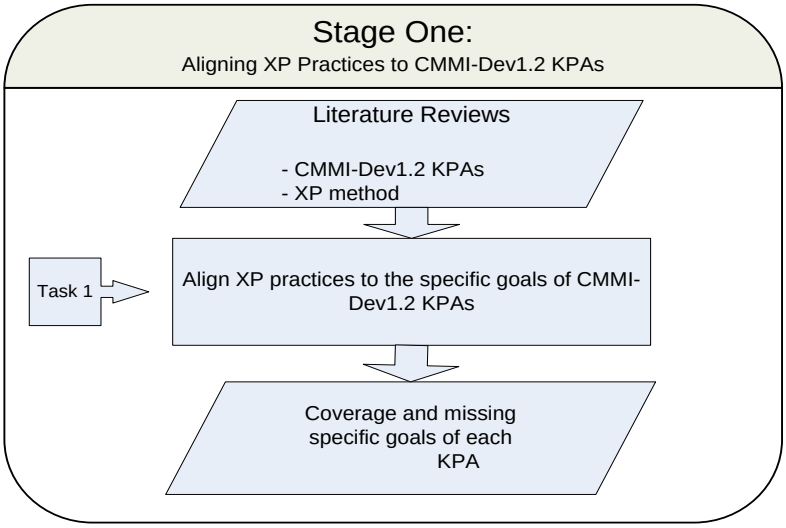

Figure 3.1: Aligning XP Practices to CMMI-Dev1.2 KPAs

Goal:

- To identify the coverage and missing specific goals of CMMI-Dev1.2 KPAs by XP practices.

Tasks:

- Task 1: this task is based on the related literature of CMMs and XP method, such as: Paulk (2001), Martinsson (2002), Koch (2003), Fritzsche and Keil (2007), Elshafey and Galal-Edeen (2008), Omran (2008), CMMI Product Team (2006), Jeffries et al. (2002), and Beck (2000). Although CMMI-Dev1.2 is different from CMM, the comparison of the XP method to CMM helps by providing good insight about the relationship between CMMI-Dev1.2 and the XP method, as the KPAs of CMM are already included in the KPAs of CMMI-Dev1.2 (Fritzsche & Keil, 2007).

Task 1 aligns XP practices to the specific goals of CMMI-Dev1.2 KPAs; taking into account the achievement of the specific practices of each specific goal by the same or different way of CMMI-Dev1.2, so as to know the coverage and missing specific goals of each KPA by XP practices.

As mentioned in Section 2.6, different scales have been used by other researchers to conduct the coverage ratio of XP practices to CMMI-Dev1.2 KPAs. Therefore, based on the descriptions of these scales, it was concluded that all of these scales focus on three common scales, which are: (1) largely supported: XP practices largely support the specific goals of the KPA; (2) partially supported: XP practices partially support the specific goals of the KPA; and (3) not-supported: XP practices do not support or are not applicable for the specific goals of the KPA. As a result of Task 1, the missing specific goals of partially and not-supported KPAs were known and used as inputs in Stage Two. Section 4.2 discusses the results of aligning XP practices to the KPAs of CMMI-Dev1.2.

3.3 Stage Two: Developing the Proposed Software Development Process Improvement Framework

In this stage, the Situational Method Engineering (SME) theory was used to extend XP method. The EBA of the SME theory was suitable for this study to extend XP method based on the KPAs of CMMI-Dev1.2, as this approach was developed for extending the existing methods to achieve specific issues. Details of SME and EBA are discussed in Section 3.6. Section 4.3 explains the adaptation of this approach in this study. In this

regard, the missing specific goals of partially and not-supported CMMI-Dev1.2 KPAs and related literature were used as inputs to determine the required development, management, and improvement additions, which are needed to cover these missing specific goals by extending XP method.

Given the importance of generalizing the new phases of the Extended-XP method, the phases of the popular software development methods such as Waterfall, Spiral, Incremental, and Prototyping (Preuninger, 2006; Sommerville, 2007), were compared with the phases of XP method to identify the related development activities. Based on the required additions that had to be added to XP method and the relations between the phases of the generic software development methods with the phases of XP method, the phases of the XP method were extended to be used as comprehensive phases of the popular software development methods.

Then, the required software development, management and improvement additions were entered to the new phases of the Extended-XP method to cover the related missing KPAs of CMMI-Dev1.2. As such, the Extended XP-method was known and used as a main element in developing the proposed software development process improvement framework. Accordingly, CMMI-Dev1.2 KPAs, the proposed Extended-XP method, and the generic elements of SPI framework (software process, assessment, capability determination, and improvement strategy) were used as inputs to establish the proposed framework. Then, the generic elements of the SPI framework were modified to suit the software development and improvement issues. Figure 3.2 shows the tasks of developing the proposed framework.

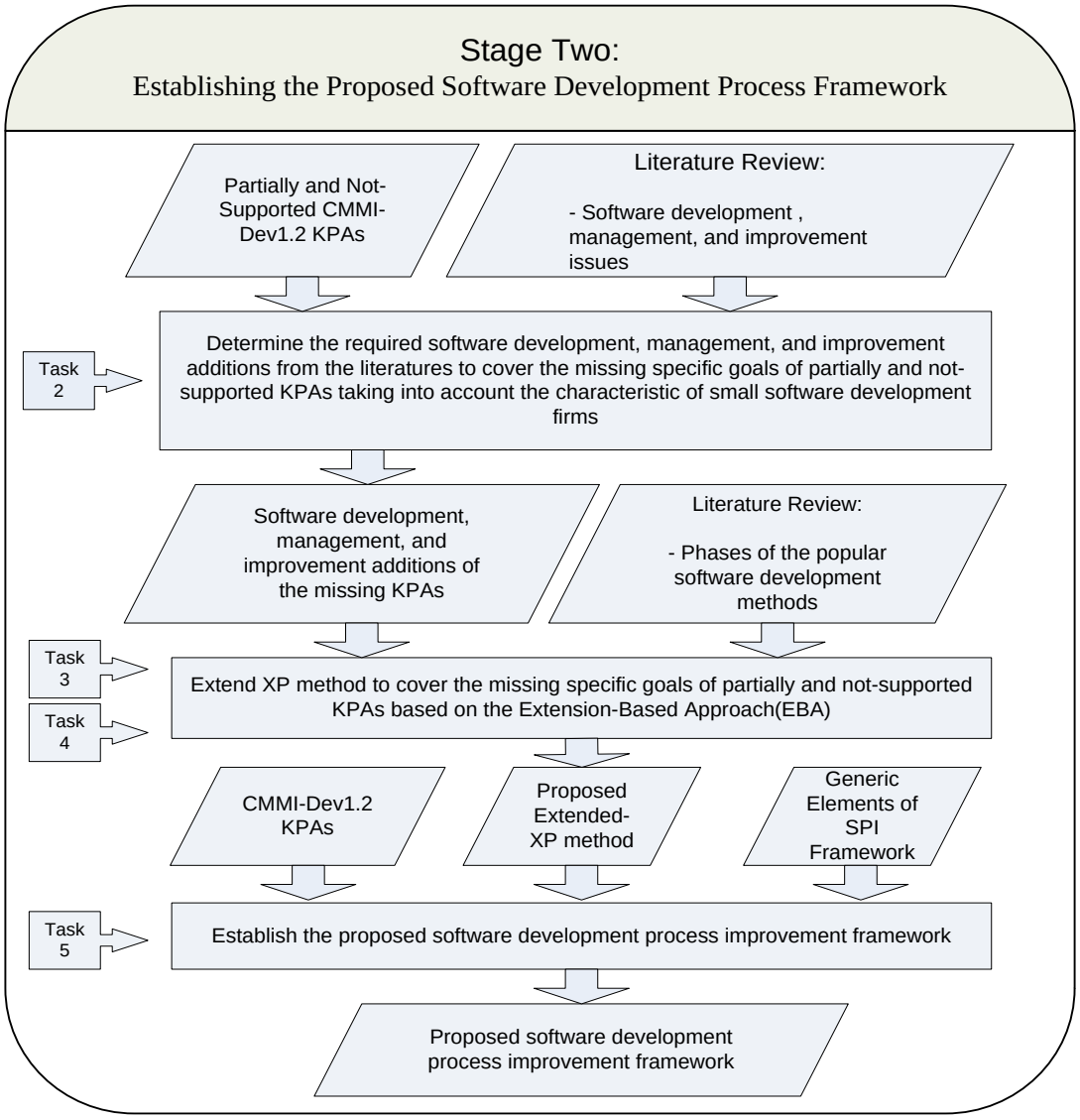

Figure 3.2: Developing the Proposed Software Development Process Improvement Framework

Goals:

- To extend XP method by adapting the EBA.
- To establish the proposed software development process improvement framework for SSDFs.

Tasks:

- Task 2: as a result of Task 1, the missing specific goals of CMMI-Dev1.2 KPAs were known. Based on that, Task 2 aimed to determine the required development, management, and improvement additions that were needed to cover the missing specific goals. This was done by studying the related previous works of Stephens (2001), Martinsson (2002), CMMI Project Team (2006), Vitoria (2004), Fritzsche and Keil (2007), Hearty (2008), Altarawneh and Shiekh (2008), and Omran (2008) to extract the required additions, taking into account the general characteristics of SSDFs, as discussed in Section 4.3.1. Accordingly, these additions were used as inputs in Tasks 3 and 4.

- Task 3: in this task, the phases of the popular software development methods (Waterfall, Spiral, Incremental, and Prototyping) were compared based on the phases of XP method. Accordingly, these relations and the required software development, management, and improvement additions that were found in Task 2, were used to extract comprehensive phases and these phases were used as generic phases of the proposed Extended-XP method, as discussed in Section 4.3.2. Then, these generic phases were used as inputs in Task 4.

- Task 4: based on the results of Tasks 2 and 3, the required development, management, and improvement additions and the generic phases of Extended-XP method were known. Accordingly, the required additions were entered and distributed into the phase of the proposed Extended-XP method, taking into account the suitable positions of these additions during the software

development lifecycle, as discussed in Section 4.3.2. Thus, the proposed Extended-XP method was known and used as input in Task 5.

- Task 5: this task started to modify the generic elements of the SPI framework to suit the software development and improvement issues, as discussed in Section 4.4.1 As such, the proposed Extended-XP was used as a software development method instead of the improvement strategy, while the CMMI-Dev1.2 model was used as an assessment model in establishing the proposed framework. Then, the proposed framework was used as inputs in Stage Three.

3.4 Stage Three: Verifying the Proposed Software Development Process Improvement Framework

As a result of Stage Two, the proposed software development process improvement framework for SSDFs was developed. Stage Three aimed to verify the compatibility of the proposed framework to the specific goals of CMMI-Dev1.2 KPAs, to verify the commitment of the proposed Extended-XP method to XP values, to verify the suitability of the proposed framework and the proposed Extended-XP structures for the software development process improvement issues in SSDFs, and to verify the proposed Extended-XP roles for their practices. In this respect, focus group method coupled with Delphi technique was used in three rounds as the verification process. Details of the focus group verification are discussed in Section 3.7. Based on the results of the verification process, the proposed framework was modified and used as input in Stage Four. Figure 3.3 shows the verification process steps. Chapter 5 discusses the verification process.

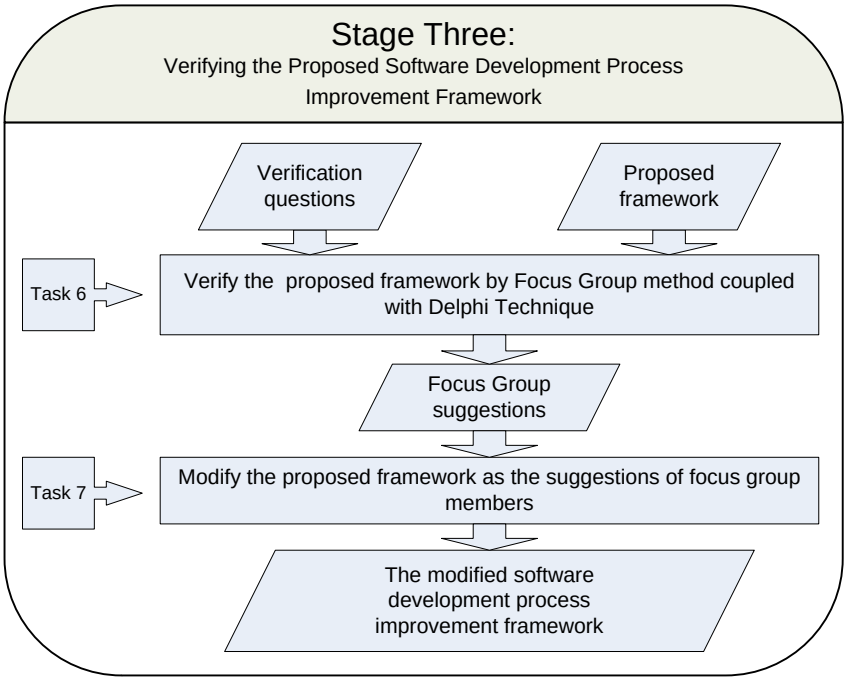

Figure 3.3: Steps of Verifying the Proposed Framework

Goals:

- To verify the compatibility of the proposed framework to the specific goals of CMMI-Dev1.2 KPAs.
- To verify the commitment of the proposed Extended-XP method to XP values.
- To verify the suitability of the proposed framework and the proposed Extended-XP method roles for their practices.
- To verify the suitability of the proposed framework and the proposed Extended-XP structures for the software development process improvement issues in SSDFs.

Tasks:

- Task 6: As a result of Task 5, the proposed software development process improvement framework was developed. This task aimed to verify the framework by focus group method coupled with Delphi Technique.

  Prior to starting the verification process, the verification questionnaire was pre-tested to ensure that the verification questions are clear, can be understood by the respondents, are comprehensive, sufficient, and suitable to achieve the aim of this research. This was done by using face-to-face interviews with related expert researchers and professional developers from the SSDFs. As a result, there were just minor corrections to these questions (refer to Appendix C, verification questionnaire).

  Accordingly, the questions and the required data were used as inputs to the proposed framework in the first round of verification process. As a result of the first round of the verification process, the answers and suggestions of focus group members were known and used as inputs in Task 7. Section 5.5.1 discusses the results of the first verification round.

- Task 7: this task aims to gather the answers and suggestions as a result of Task 6, in order to know the required modifications that had to be done to the proposed framework. Subsequently, the proposed framework was modified to suit the focus group members' suggestions. Sections 5.5.2 and 5.5.3 discuss the results of the second and third verification rounds. Then, the modified

framework was used as input in Stage Four to validate the suitability of the framework for SSDFs, and to validate the applicability and effectiveness of implementing this framework for these firms.

3.5 Stage Four: Validating the Modified Software Development Process Improvement Framework for SSDFs

As a result of Task 7, the proposed framework was modified based on the answers and suggestions of focus group members. This stage aims to validate the suitability of the modified software development process improvement framework for SSDFs, and also to validate the applicability and effectiveness of this framework by these firms. Chapter 6 discusses the validation process. In this respect, two approaches were used to validate the modified framework:

- Quantitative research method that involves survey was used to validate the suitability of this framework for SSDFs. Al-Allaf (2008), in her work about SPI model for large software firms, used CMMI questionnaires as a testing method to validate the suitability of her proposed model for these firms, where this questionnaire was validated by SEI. In this research, the questionnaire validation consists of two parts, as follows:
    - Part one: this part aimed to identify the demographic information of the respondents, such as: current position, current work, size of firm, and software experience. The demographic questions of this part were already validated, as these are adopted from El Sheikh and Tarawneh

(2007) and Cater-Steel (2004a), who are working in the same field of software improvement.

- Part two: this part aimed to validate the suitability of this framework for SSDFs. In this part, CMMI-Dev1.2 questionnaire was used to ask the professional developers and managers of SSDFs to rate the levels of the suitability of the framework for their firms, based on XP practices and the software development, improvement, and management additions, that were used in the framework to cover the specific goals of the suitable CMMI-Dev1.2 KPAs.

  The scale used in this questionnaire consisted of scaled-response items rating from 1 to 5, where 1= Strongly Unsuitable and 5= Strongly Suitable. This scale was used to obtain more accurate results. This scale was adopted from Vasiljevic and Skoog (2003), who have already validated and used this scale as they work in the same field of this study (refer to Appendix D, validation questionnaire).

- Qualitative research method that involved two case studies was used to validate the applicability and effectiveness of implementing the modified framework by SSDFs.

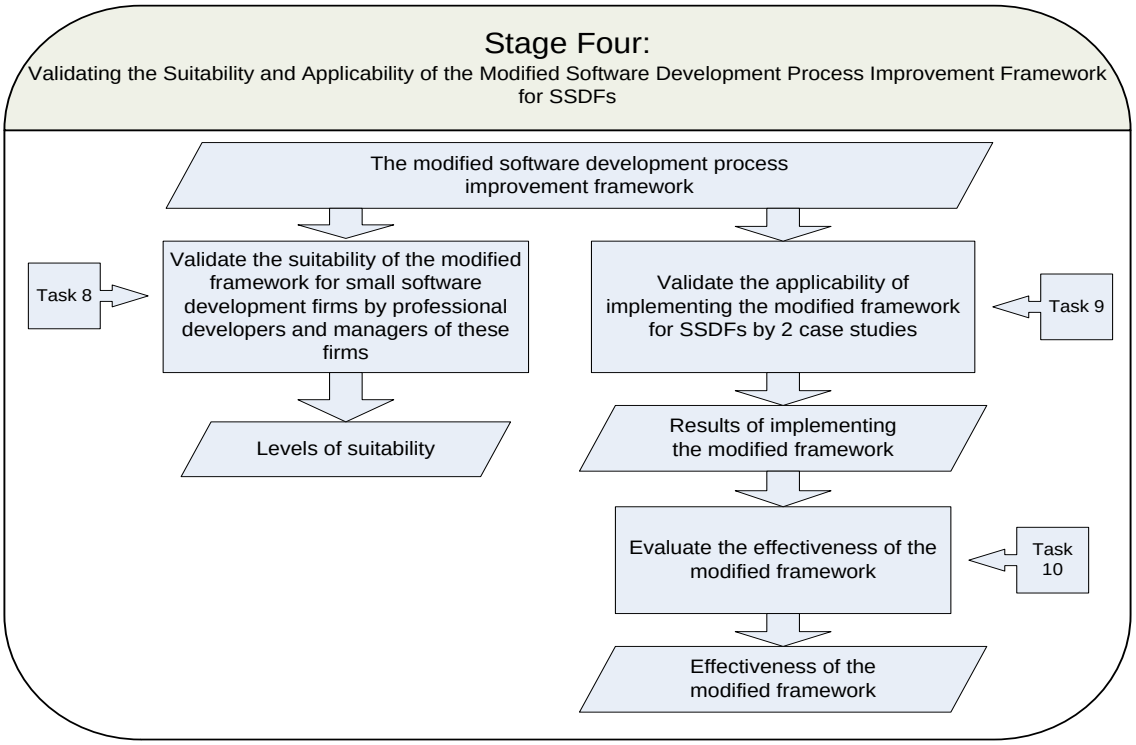

Figure 3.4 Steps of the Validation Process

Goals:

- To validate the suitability of the modified framework for SSDFs.
- To validate the applicability and effectiveness of implementing the modified framework for SSDFs.

Tasks:

- Task 8: as a result of Task 7, the proposed framework was modified according to the suggestions of focus group members. Accordingly, Task 8 aimed to validate the suitability of the modified framework for SSDFs. Using the information obtained from the Jordanian Ministry of Industry and Trade, it was convenient to access the addresses of some SSDFs in Jordan.

Accordingly, CMMI-Dev1.2 questionnaires were distributed and collected from the professional developers and managers who were working in Jordanian SSDFs. Based on the data from these questionnaires; Statistical Package for the Social Sciences (SPSS) program was used to calculate the main values of the suitability of this framework for their firms. As a result, the suitability of the modified framework for SSDFs was known and is discussed in Section 6.2.

- Task 9: referring to modified framework which resulted from Task 7, this task aimed to validate the applicability of implementing this framework by Jordanian SSDFs. In this regard, case study method was used in this study to validate the modified framework, for several reasons, such as: (1) the examination of the data is most often conducted within the context of its use (Yin, 1984; Zainal, 2007); (2) the variations in terms of intrinsic, instrumental and collective approaches to case studies allow for both quantitative and qualitative analyses of the data (Block, 1986; Yin, 1984); and (3) case study helps to explore or describe the data in real-life environment, and also helps to explain the complexities of real-life situations which may not be captured through experimental or survey research (Zainal, 2007).

  In this respect, the registration record of the software development firms in Jordanian Ministry of Industry and Trade was the convenient way to access the addresses of SSDFs in Jordan. Accordingly, several meetings were held with the managers of some firms to get their agreement for implementing the modified framework in their firms. Based on these meetings, two firms agreed to

implement the modified framework in developing their projects. Thus, the modified framework was implemented by the two Jordanian SSDFs, as discussed in Section 6.3.

- Task 10: As a result of Task 9, the modified framework was implemented by two Jordanian SSDFs. Task 10 aims to evaluate the effectiveness of the framework for SSDFs. In this task, interview method was used as a data collection method to evaluate the modified framework. The primary purpose of the interview approach is to understand the meanings that interviewees attach to issues and situations, in contexts that are not structured in advance by the researcher's assumptions (Easterby-Smith et al., 1991). Kunda (2001) adopted the interview approach to evaluate his framework which was developed for the software engineering field. In this regard, it was suitable in this study to conduct interviews with the project team members of the two SSDFs, to evaluate the effectiveness of the modified framework for SSDFs.

  Several studies discuss the criteria that are needed to evaluate the effectiveness of implementing software methods, models, and frameworks, such as Kitchenham (1998), and Garrity and Sanders (1998). Kitchenham (1998), in evaluating his method, used three major measures of success: basic evaluation, use evaluation and gain evaluation. Basic evaluation is concerned with quality of the component documentation, for example, completeness, readability and understandability of the component description. Use validation is concerned with the quality of the component, for example, whether the component is easy to

implement and “helpful”. Gain validation is concerned with the benefits delivered by the component, for example, whether the component is cost-effective and supports decision making.

In addition, Garrity and Sanders (1998) argued that when measuring the success of an information system, it is important to include the organizational and social-technical viewpoints, by using three major measures of success: task support, quality of life support satisfaction, interface satisfaction and decision support satisfaction.

As mentioned in the previous works of Kitchenham (1998), and Garrity and Sanders (1998), about the evaluation criteria, it can be concluded that there are three common evaluation criteria which are: gain validation, interface satisfaction, and task support satisfaction. These common criteria had been used by Kunda (2001) in evaluating his research which was carried out in the field of software engineering. Therefore, these common criteria are suitable and useful to be used in evaluating the modified framework in this study (refer to Appendix E, evaluation criteria questionnaire).

Accordingly, these common evaluation criteria were used as main variables in evaluating the effectiveness of the modified software development process improvement framework, by interviewing the project teams who participated in implementing the developed framework by these firms, as discussed in Section 6.4.

3.6 Situational Method Engineering (SME) Theory

Method engineering theory is a set of approaches to create software development methods that are specifically attuned to organizations or projects (Brinkkemper, 1996). Ralyté et al. (2003), Bajec et al. (2007), and (Weerd, 2009) argued that the Situational Method Engineering (SME) theory is the popular method engineering theory which is focused mostly on constructing or adapting a method for a certain issue. According to Ralyté et al. (2003), SME can be distinguished to four different approaches based on their starting point, which are:

- **The Extension-Based Approach (EBA)** focuses on an existing method and provides novel additions to it. The objective of this approach is to enhance a method with new features that are helpful to meet the requirements of the project.
- **The Paradigm-Based Approach** takes a meta-model that belongs to a certain theoretical framework as starting point. This meta-model is specialized, abstracted or adapted according to the objective of the project.
- **The Ad-Hoc Approach** is concerned with the construction of a novel method 'from scratch'. This strategy is required when necessary method fragments or Meta models are not available. This can be the case when the project deals with a new application domain that is not yet covered by a specific method or when the project characteristics differ significantly from former situations.
- **The Assembly-Based Approach** reuses method fragments to construct a new method. This approach assumes that these fragments have been detached from existing methods, provided with a description and stored in a method base.

Based on the specific characteristics of a project these fragments can be selected from this repository and assembled by following predefined rule.

As discussed in Section 3.3, the first two goals in Stage Two aimed to extend XP method to fulfill the missing KPAs of CMMI-Dev1.2. In this regard, the EBA has been selected to be used as a main approach in extending XP method, where this approach was developed for extending the existing methods to achieve specific issues compared to the others approaches.

EBA guides method engineer by providing extension patterns that help identifying typical extension situations and provide guidelines to perform the required extension. This approach consists of two main processes: (1) specify extension requirement: the requirement elicitation helps the method engineer to construct a map representing the method extension requirements; and (2) select & apply a pattern: by using process extension strategy as a guidelines which help to match the requirements map intentions and strategies with the pattern situation (Ralyté et al., 2003; Ralyté et al., 2004). Figure 3.5 shows the flow of the EBA for SME.

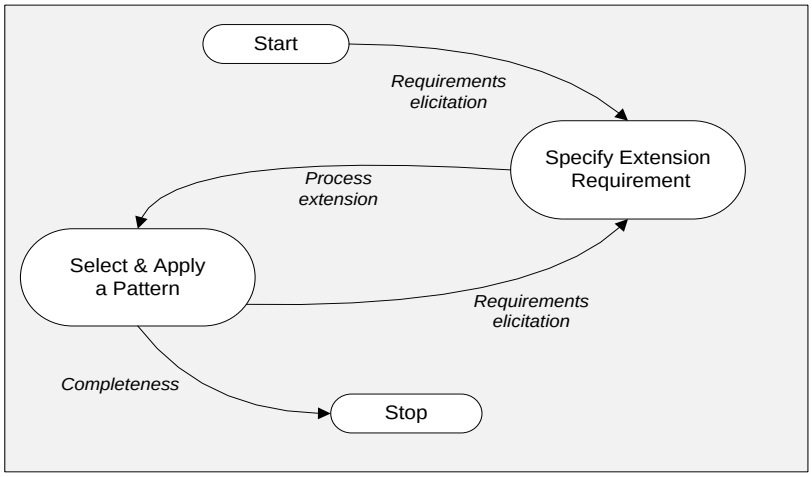

Figure 3.5: EBA for SME (Ralyté et al., 2003)

The EBA had been used by several researchers in several domains such as:

- Gerami and Ramsin (2011) used EBA to develop the aspect-oriented extension framework by extending the old version of the ASD methodology. The proposed framework covers the entire software development lifecycle by determining which basic features an aspect-oriented agile development process should possess, and can therefore also act as a benchmark for the evaluation and comparison of different methodologies. In addition, they argued that the EBA can be used for extending agile methodologies. Furthermore, they indicated the importance of using the agile values to ensure that the extension does not adversely affect agility.
- Lee et al. (2007) used EBA to develop the data warehouse portal module by extending the data warehouse queries into portals. In this work, the extension to the original data warehouse query tool was developed in order to support the required functionality to integrate data warehouse queries into portals as a separate module that can communicate with both the portal and the data

warehouse query tool. The usage of this approach can include complex capabilities to satisfy the portal characteristics. However, this approach requires the development of an additional module (i.e., extension) and needs to be able to interface seamlessly with the portal as well as the data warehouse query tool.

- Thapa et al. (2011) adapted a UML EBA that allows developing systems with domain-specific security and time-related requirements to develop an approach to verifying security and timing properties in UML models. In addition, the UML extension has been popularly utilized to tailor the UML to specific domains, where the extension was given in the form of UML profiles.
- Kuan et al. (2008) adapted the EBA to develop an algorithm to locate groups. In their research, the transitive extension method has been used to derive substructures, or communities, within social networks. Results showed that this method was fairly effective in finding community of friends. However, this method does not provide insight into how these communities are formed.

In this study, the EBA has been adapted to be used in extending the XP method. Figure 3.6 shows the processes of the adapted EBA.

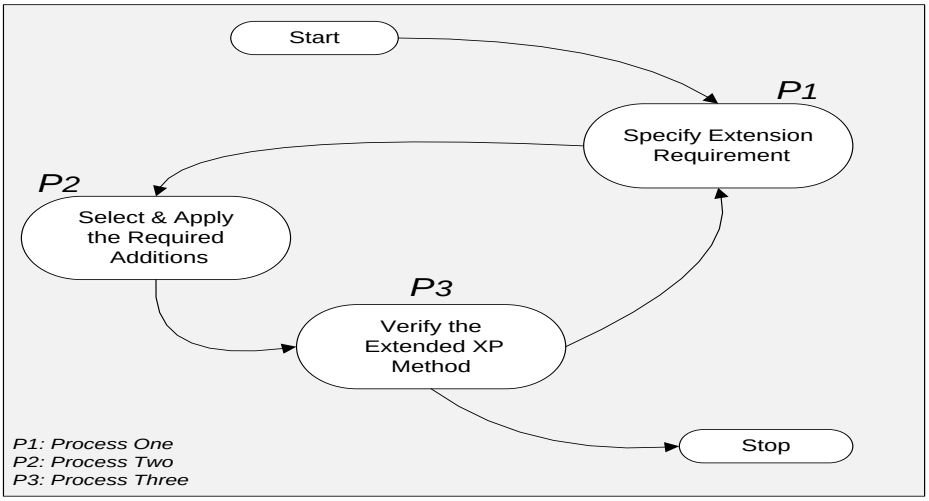

Figure 3.6: Adapting EBA in to Extend XP Method (adapted from Ralyté et al., 2003)

As shown in Figure 3.6, three main processes (P1, P2, and P3) are used in extending XP method, which are:

- **P1** (Specify extension requirements): this process aims to extract the required software development, management, and improvement addition that are needed to cover the partially and not-supported KPAs of CMMI-Dev1.2. Section 4.3.1 discusses these required additions.

- **P2** (Select & apply the required additions): this process aims to extract the new phases of the proposed Extended-XP method and harmonize these phases to be a comprehensive for all the popular software development methodologies. In addition, to distribute the required software development, management, and improvement additions into the new phases of the proposed Extended-XP method based on the need for these additions during the software development lifecycle. Section 4.3.2 discusses this process.

- **P3** (Verify the Extended-XP method): this process aims to verify the commitment of the proposed Extended-XP method to the principles of XP method. This process is very important to keep the extremely way of the software development lifecycle (Gerami & Ramsin, 2011). In this respect, XP values that reflect the XP principles (Beck, 2000) were used as a main question during the verification process which is discussed in Section 5.5.1.2.

3.7 Focus Group Coupled with Delphi Technique

Some researchers, such as Badoo and Hall (2002), Bechams et al. (2003), and Basri and O'Connor (2011) working in the software improvement field have used the focus group method as a testing method to verify their works. In this study, the focus group method coupled with Delphi technique was used to verify the proposed framework. Sections 3.7.1 to 3.7.2 illustrate focus group method and Delphi technique.

3.7.1 Focus Group Method

Focus group is one of the qualitative methods which enable the researcher to verify the topics of his research with experts. There are many definition of a focus group in the literature, where Powell et al. (1996) defines a focus group as *"a group of individuals selected and assembled by researchers to discuss and comment on, from personal experience, the topic that is the subject of the research".* Furthermore, Morgan (1997) defines the focus group as *"interaction within the group based on topics that are supplied by the researcher".*

Focus group is where a number of experts meet in the same place and same time to discuss amongst them and with the researcher about the research topics that enable the researcher to verify his work. In this study, focus group is a suitable method to verify the proposed framework for several reasons as Krueger and Casey (2000), Kontio et al. (2004), and Mazza and Berre (2007) have pointed out:

- Discovery of new insights: focus group method enables the researcher to discover new issues during the dissections between the participants.
- Cost-efficiency: the researcher can to meet a group of people at the same time and same place, where some researchers consider this method as the cheapest method for data collection.
- Depth of interview: focus group enables the participants to discuss deeply about the research issues, and this enables the researcher to obtain correct answers as much as possible.
- Business benefits to participants: the participants may get some benefits during the focus group sessions, where they will get new ideas and perhaps they will discuss with others to get new business for their companies.

Furthermore, Krueger and Casey (2000), Kontio et al. (2004), and Mazza and Berre (2007) also mentioned that there are several general steps in preparing the focus group as follows:

- Define the research problem: in this step there is the need to define and determine the research problem that will be discussed in the first focus group session to enable the participants to understand the importance of this research. In this study, the research problem is "there is a need for software

development process improvement framework to help SSDFs in developing, managing, and improving their software development processes in a systemic way".

- Prepare the needed questions: before the first session, there is the need to determine the needed questions that will help to verify the proposed framework. In this study, the verification questionnaire was prepared and is discussed in Section 5.3.
- Plan the focus group events: the number of issues to be covered needs to be limited so that sufficient time can be allocated for the participants to comprehend the issue and have a meaningful discussion and interaction about them. In this study, the verification schedule was prepared and is discussed in Section 5.4.
- Select the participants: the selection of participants is very important to get the right answers as far as possible. Therefore, it is important to select the highly experienced people to discuss the activities of the proposed framework. In this study, three expert researchers were selected as members of the focus group sessions for this study because they have good knowledge of CMMI-Dev1.2 and XP method, and also have publications in the fields of software development and improvement.

Based on the Jordanian Ministry of Industry and Trade, it was suitable to obtain the addresses of some SSDFs in Jordan. Through phone conversation, three professional developers, two professional managers, and two members of the software engineering process group who are working in different

SSDFs in Jordan agreed to participate in verifying the proposed framework through focus group sessions. Section 5.2 discusses the selection of the focus group members in detail.

- Conduct the focus group session: The focus group session needs to be carefully managed for time, while still making sure that all main contributions can be made during the allocated time, and each of the research topics are presented one after another. Furthermore, there are several types of data captured during the session, such as: taking notes during the session, audio, video, white papers or keyboard recording. In this study, white papers were used during all sessions to document the suggestions of the focus group members.

3.7.2 Delphi Technique

The Delphi technique, mainly developed by Dalkey and Helmer (1963) at the Rand Corporation in the 1950s, is a widely used and accepted method for achieving convergence of opinion concerning real-world knowledge solicited from experts within certain topic areas. This technique has and will continue to be an important data collection methodology with a wide variety of applications and uses for people who want to gather information from those who are immersed and imbedded in the topic of interest and can provide real-time and real-world knowledge. Furthermore, Delphi technique provides more opportunities to researchers than survey research, where the components of the Delphi technique include the communication process, a group of experts, and feedback (Stitt-Gohdes & Crews, 2004). Theoretically, the Delphi process can be continuously iterated to achieve the main goal of verification. Three iterations are

often sufficient to collect the needed information and to reach a consensus in most cases (Ludwig, 1997). Nevertheless, Walker and Selfe (1996) argue that most studies use two to three rounds.

In this research, Delphi technique was used as the main technique to arrange the focus group method through three rounds to verify the proposed framework, where the first round had three sessions aimed to answer the verification questions and to record the suggestions of the focus group members about the required modifications for the proposed framework. The second round aimed to modify the proposed framework based on the answers and suggestions from the focus group members that were made in the first round. Finally, the third round aimed to present the modified framework to the focus group members to make sure that all the suitable suggestions had been taken into account in the framework and also to check if there is need for further modifications. Section 5.5 discusses the results of the verification rounds.

3.8 Conclusion

There are four stages used to construct the software development process improvement framework for SSDFs. Stage One aimed to identify the coverage and missing specific goals of CMMI-Dev1.2 KPAs by XP method. In Stage Two, the EBA has been adapted to extend XP method, where the related previous works were studied to determine the required software development, management, and improvement additions to cover the related KPAs of CMMI-Dev1.2. Then, based on the required additions and also based on the phases of the general software development methods, the XP method was

extended. Subsequently, the proposed Extended-XP method and CMMI-Dev1.2 with the generic elements of SPI were used to develop the proposed framework.

In Stage Three, the focus group method coupled with Delphi Technique was used to verify the compatibility of the proposed framework to the specific goals of CMMI-Dev1.2 KPAs, to verify the commitment of the proposed Extended-XP method to XP values, to verify the suitability of the proposed framework and the proposed Extended-XP structures for the software development process improvement issues in SSDFs, and to verify the proposed Extended-XP roles for their practices.

In Stage Four, two approaches were used to validate the modified framework. The first approach involved the quantitative research method using a survey that was used to validate the suitability of this framework for SSDFs by using CMMI-Dev1.2 questionnaire. The second approach used a qualitative research method that involved two case studies to validate the applicability of implementing the modified framework by SSDFs. At the end of Stage Four, the common evaluation criteria were used to evaluate the effectiveness of the framework for the two firms that implemented this framework.

CHAPTER FOUR
DEVELOPMENT THE PROPOSED SOFTWARE DEVELOPMENT PROCESS IMPROVEMENT FRAMEWORK

This chapter presents the steps used in this study to develop the proposed software development process improvement framework for SSDFs. These steps start with aligning the XP practices to the specific goals of CMMI-Dev1.2 KPAs to know the coverage and missing KPAs. The chapter also presents the processes of adapting the EBA used to extend XP method. At the end of this chapter, the foundation and the processes of establishing the proposed software development process improvement framework is presented.

4.1 Introduction

This study aims to construct a suitable software development process improvement framework for SSDFs based on CMMI-Dev1.2 as a SPI model and the XP as a software development method, taking into account the generic elements of SPI framework. At the beginning of this chapter, the alignment results of XP method to the specific goals of CMMI-Dev1.2 KPAs are presented. Based on these results, the EBA has been adapted to extend XP method. In this respect, the required software development, management, and improvement additions that are needed to be added to the XP method to fulfil the missing KPAs of CMMI-Dev1.2 are highlighted. Subsequently, these required additions with the generic phases of the popular software development methodologies were used to extend the XP method to fulfil the missing KPAs of CMMI-Dev1.2 to be used as general phases compared to the popular methodologies. Accordingly, the proposed

Extended-XP method, CMMI-Dev1.2, and the generic elements of SPI framework were used as main items to establish the proposed software development process improvement framework for SSDFs.

4.2 Aligning XP Practices to the KPAs of CMMI-Dev1.2

The first objective in achieving the main aim of this study is to identify the coverage of XP practices to the specific goals of CMMI-Dev1.2 KPAs. Therefore, specific goals of each CMMI-Dev1.2 KPAs were used as the main items to know the coverage ratio of these goals by the XP practices. In doing this alignment, the description of CMMI-Dev1.2 (CMMI Product Team, 2006) and the description of the XP method (Beck, 2000; Jeffries et al., 2002) were used as the main references.

Prior to starting this alignment, it is important to determine the suitable scales of supporting the XP practices to the KPAs CMMI-Dev1.2. In this respect, three scales were selected to perform the alignment XP practices to the KPAs of CMMI-Dev1.2, because of the common use of these scales in related studies as discussed in Section 2.6. This scale consists of:

- Largely Supported (L.S): XP practices largely support the specific goals of the KPA.
- Partially Supported (P.S): XP practices partially or implicitly support the specific goals of the KPA.
- Not-Supported (N.S): XP practices do not support or not applicable for the specific goals of the KPA.

Sections 4.2.1 to 4.2.4 discuss the alignment of XP practices to the KPAs of CMMI-Dev1.2 based on the specific goals of each KPA.

4.2.1 Aligning XP Practices to the Level 2 KPAs of CMMI-Dev1.2

Level 2 (Managed process) is a process that has the basic infrastructure in place to support the process. It is planned and executed in accordance with policy; employs skilled people who have adequate resources to produce controlled outputs; involves relevant stakeholders; is monitored, controlled, and reviewed; and is evaluated for adherence to its process description. The process discipline reflected by level 2 helps to ensure that existing practices are retained during times of stress. This level consists of seven KPAs as follows:

- **Requirements Management - P.S**

  The purpose of this process area is *"to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products"* (CMMI Product Team, 2006).

  As shown in Table 4.1, it can be concluded that the specific goal the requirement management KPA is partially supported by some of XP practices, which are: on-site customer, planning game, continuous integration, metaphor, and small releases.

However, XP method does not have data repository to keep the required data of user stories.

Table 4.1: Coverage of the XP Practices to Requirement Management KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Manage Requirements: Requirements are managed and inconsistencies with project plans and work products are identified. **Specific Practices:** • Obtain an understanding of requirements. • Obtain commitment to requirements. • Manage requirements changes. • Maintain bidirectional traceability of requirements. • Identify inconsistencies between project work and requirements.	User story cards are used for collecting customer requirements that describe the features to be added into the program, where each story card contains one feature. Accordingly, the programmers divide the features into tasks, as well as estimate their tasks which consist of two tasks: (1) customer task: include determining the scope of the project, priority of the features, composition of releases, dates of releases; and (2) programmer tasks: include the estimations of the features, technical consequences, process, and detailed scheduling. Moreover, the understanding of these requirements is obtained through the integration of the customer into the team. The intensive communication between the customer and the development team help in identifying the contents of each release. Therefore, iteration to release enables the project team to know the current state of the project and help the customer to identify and communicate further requirements, because small releases help in integrating the feedback on customer expectations and needs. In addition, the requirements of the project can be quickly exchanged and discussed during the small releases, where the metaphor and user stories enable the collaboration between the customer and developers to check the status of the requirements. Small releases help to conduct the consistency between the requirements and other work products. In addition, user stories, functional test, and unit test help in detecting the inconsistencies between the project work and the requirements. However, traceability of the requirements is not largely supported by XP method, because there is no data repository in XP to keep the records of previous story cards and old versions of the documentation. XP method does not have data repository to keep the required data of user stories. Therefore, the “bidirectional traceability of requirements” practice is not largely supported by XP method.

- **Project Planning - L.S**

The purpose of this process area is "*to establish and maintain plans that define project activities"* (CMMI Product Team, 2006).

As shown in Table 4.2, it can be concluded that all the specific goals of the project planning KPA are largely supported by some of XP practices, which are: planning game, small releases, on-site customer, and metaphor.

Table 4.2: Coverage of the XP Practices to Project Planning KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Establish Estimates: Estimates of project planning parameters are established and maintained. **Specific Practices:** • Estimate the scope of the project • Establish estimates of work product and task attributes • Define project lifecycle • Determine estimates of effort and cost	By planning game practice, the customer illustrates which of the features he/ she wants for the next release; then the programmers will divide the features into tasks as well as estimate their tasks, where the customer tasks consist of: defining the project scope, determining the priority of the features, in addition to composition of releases and the dates of releases. In contrast, the programmer's tasks concentrate more on: estimations of the features, technical consequences, process, and detailed scheduling. In addition, in XP lifecycle, the software team involved in early planning and integrated into the commitment process by estimating the effort involved to implement the customer stories. Therefore, the estimate of stories and tasks are established and can be corrected during the project. Furthermore, the iteration to release practice helps in increasing the estimation precision.
SG 2. Develop a Project Plan: A project plan is established and maintained as the basis for managing the project. **Specific Practices:** • Establish the budget and schedule • Identify project risks • Plan for data management • Plan for project resources • Plan for needed knowledge and skills	In the exploration phase and based on the user stories, the developers familiarize themselves with the required tools, practices and technologies that are going to be used in the project. After that, the teams of developers test the technology and also develop a prototype to explore the architecture possibilities in developing a prototype. In addition, collective ownership practice supports the involvement of all relevant stakeholders in the planning phase, and this help in increasing the commitment to the iteration plans. The project plan is established through several releases and iterations that evolve throughout the project. Therefore, the risks are identified, training needs are planned, and the involvements of all the teams are assured if XP is applied correctly. In addition, incremental and evolutionary XP lifecycle help the developers to identify and manage risks efficiently. Furthermore, planning game practice is responsible for establishing the project schedule, budget,

• Plan stakeholder involvement • Establish the project plan	and plan for each iteration.
SG 3. Obtain Commitment to the Plan: Commitments to the project plan are established and maintained. **Specific Practices:** • Review plans that affect the project • Reconcile work and resource levels • Obtain plan commitment	Commitment to the release and iteration plans is obtained through the high involvement and responsibility of all team members. In addition, the tracker is responsible for tracing the progress of each iteration and evaluating whether the goal is reachable with the given resource and within the time constraints, or if any changes are needed in the process. Furthermore, Coach is responsible to ensure that the project goes along the right path by keeping people working on the current features for the actual iteration.

- **Project Monitoring and Control - L.S**

  The purpose of this process area is *"to provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance deviates significantly from the plan"* (CMMI Product Team, 2006).

  As shown in Table 4.3, it can be concluded that the all the specific goals of the project monitoring and control KPA are largely supported by some of XP practices, which are: on-site customer, test-driven development, collective ownership, and small releases.

Table 4.3: Coverage of the XP Practices to Project Monitoring and Control KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Monitor Project Against Plan: Actual performance and progress of the project are monitored against the project plan. **Specific Practices:** • Monitor project planning Parameters • Monitor commitments • Monitor project risks • Monitor data management • Monitor stakeholder involvement • Conduct progress reviews • Conduct milestone reviews	The tracker gives feedback on the development by monitoring the schedule and estimates. He traces the estimates made by the team (e.g. effort estimates) and gives feedback on how accurate they are in order to improve future estimations and also traces the progress of each iteration and evaluates whether the goal is reachable within the given resource and time constraints or if any changes are needed in the process. In addition tracker is responsible to collect metrics for the project performance during the iteration, and it is a common for XP projects to use a spreadsheet application or simpler tools like pen and paper for tracking project-planning metrics such as estimates and actual achievements as well as the overall project plan. A big visual chart and conducting the project velocity support the commitments of the stories during the small releases. Therefore, this commitments process enable the clear expectations between the customer and other project team at the tactical level, and also increase the flexibility at the project's strategic level. Therefore, the information on the project's progress is gathered by the use of measures and the milestones are checked against the schedule via functional tests. XP method enables the coordination and collaboration with relevant stakeholders by integrating developers, customer, testers, and management, and also by establishing a self-organizing cross-functional team in which all relevant stakeholders. In addition, collective ownership practice helps in integrating all the team members in the project work. The intensive communications between the customer and developers handle the changes that are needed during the iteration to the software commitments, or user stories are handled, and this can be done with assistance from the coach.
SG 2. Manage Corrective Action to Closure: Corrective actions are managed to closure when the project's performance or results deviate significantly from the plan. **Specific Practices:** • Analyze issues • Manage corrective	Short iteration and regular commitments helps in monitoring and managing the project against the baseline, and also offer opportunities to make the required adjustments. Therefore, corrective actions can include adjustments to the method, and also of the functionality that will be realized based on the on-site customer practice, where there is intensive communication among the team members and the customer helps to convey the information. Coach is responsible to ensures that the programmers are working efficiently and effectively, and also to find a solution for the problems faced by the programmers as soon as possible. In addition, tracker traces the progress of each iteration and evaluates whether the goal is reachable or not, and gives feedback on how accurate they are in order to improve future estimations or if any changes are needed in the

action • Take corrective action	process. As such, tracker is responsible for informing the results of daily meetings to check the status of each iteration against the plan. Furthermore, the big visual chart which used during XP lifecycle supports this specific goal, where the velocity of the project is stated clearly as well as commitments (stories) for small releases. This visual chart is usually developed by both the customers and XP teams.

- **Supplier Agreement Management - N.S**

The purpose of this process area is *"to manage the acquisition of products from suppliers for which there exists a formal agreement"* (CMMI Product Team, 2006).

Paulk (2001), Martinsson (2002), Koch (2003), Fritzsche and Keil (2007), Elshafey and Galal-Edeen (2008), and Omran (2008) have indicated that this KPA is not supported by XP method. In this respect, Fritzsche and Keil (2007) argued that this KPA is not addressed by XP, because the involving suppliers could be problematic; but they believe that the method can be extended to fulfill the goals of this process area with keeping the agility of XP method. In addition, Omran (2008) believed that this KPA seems to consume lot of the resources from small teams. Furthermore, Martinsson (2002) pointed out that the XP method does not mentioned to this KPA and nothing in particular that prevents it. Therefore, it can be concluded that this KPA is not supported by XP method and there is need to extend XP method to cover this process area with the keeping of the agility of XP values.

- **Measurement and Analysis - P.S**

The purpose of this process area is *"to develop and sustain a measurement capability that is used to support management information needs"* (CMMI Product Team, 2006).

As shown in Table 4.4, it can be concluded that the specific goals of the measurement and analysis KPA are partially supported by on-site customer and test driven development practices. However, XP method does not have data repository to keep the measurement data.

Table 4.4: Coverage of the XP Practices to Measurements and analysis KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Align Measurement and Analysis Activities: Measurement objectives and activities are aligned with identified information needs and objectives. **Specific Practices:** • Establish measurement objectives • Specify Measures • Specify data collection and storage procedures • Specify analysis procedures	Metric is the basic management tool in XP method, where the big visual chat is used to publish the results of the metric to the project team. One recommended XP metric is project velocity (the number of stories of a given size that developers can implement in an iteration). Furthermore, measurements and analysis procedures are defined by the tracker based on: (1) tracing the estimate made by the team (e.g. effort estimates) and gives feedback on how accurate they are in order to improve future estimations, and also the tracker should be careful to not interrupt the project too many times; and (2) tracing the progress of each iteration and evaluates whether the goal can be reached within the given resources and at a certain time, or if any changes are required in the process.

SG 2. Provide Measurement Results: Measurement results, which address identified information needs and objectives, are provided. **Specific Practices:** • Collect measurement data • Analyze measurement data • Store data and results • Communicate results	Tracker is responsible for gathering the information of the project's progress by estimating the project velocity and uses the feedback from the programmers by asking and listening to what they are doing in the current moment. Accordingly, the intensive communications help to convey the important data of measurements results within the team members. In addition, functional test is used to check the milestones are against the schedule. Furthermore, wall charts are used in XP method by tracker to convey the results of analyzing the measurements data. However, XP method does not have repository to store the measurement data.

- **Process and Product Quality Assurance - P.S**

The purpose of this process area is *"to provide staff and management with objective insight into processes and associated work products"* (CMMI Product Team, 2006).

As shown in Table 4.5, it can be concluded that the specific goals of the process and product quality assurance KPA are partially supported by some of XP practices, which are: continuous integration, test driven development, and pair programming practices. However, XP method does not demand an explicit evaluation of processes, work products and services against the applicable process descriptions. In addition, there are no strict guidelines for the resolutions of noncompliance issues and for establishing records of quality assurance activities.

Table 4.5: Coverage of the XP Practices to Process and Product Quality Assurance KPA

Specific Goal &Practices	Coverage by XP
SG 1. Objectively Evaluate Processes and Work Products: Adherence of the performed process and associated work products and services to applicable process descriptions, standards, and procedures is objectively evaluated. **Specific Practices:** • Objectively evaluate processes • Objectively evaluate work products and services	The planning of the quality assurance's activities is clearly satisfied by pair programming, continuous integration, and test driven development practices. In addition, the quality is central in the regular programming sessions. Coach is responsible for guiding the team in applying XP method in the right. Accordingly, the quality issues can be easily communicated in an XP team. In addition, XP assures the quality on a social level through peer pressure, which is often very successful in assuring the conformance of the standards. However, XP method does not demand an explicit evaluation of processes, work products and services against the applicable process descriptions (Fritzsche & Keil, 2007; Martinsson, 2002).
SG 2. Provide Objective Insight: Noncompliance issues are objectively tracked and communicated, and resolution is ensured. **Specific Practices:** • Communicate and ensure resolution of noncompliance issues • Establish records	The correctness of the systems is to be shown to the customer when all tests have passed. Consequently, the application is continually growing and evolving, where the intensive communication between team members helps to resolute the results of used quality assurance activities. In addition, posting a graph is always used by the project team to present the results of quality assurance such as the results of test-failures of each release. However, there are no strict guidelines for the resolutions of noncompliance issues and for establishing records of quality assurance activities.

- **Configuration Management - L.S**

  The purpose of this process area is *"to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits"* (CMMI Product Team, 2006).

As shown in Table 4.6, it can be concluded that the all the specific goals of the configuration management KPA are largely supported by some of XP practices, which are: planning game, continuous integration, re-factoring, on-site customer, test-driven development, coding standard, collective ownership, and small releases.

Table 4.6: Coverage of the XP Practices to Configuration Management KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Establish Baselines: Baselines of identified work products are established. Specific practices to establish baselines are covered by this specific goal. **Specific Practices:** • Identify configuration items • Establish a configuration management system • Create or release baselines	Code, design, tests and requirements are considered the items of configuration in XP method. In addition, the iteration to releases supports strong baselines mechanisms and careful version control of the code and other release components. Furthermore, the using of a configuration management system is recommended by continuous integration, collective ownership, and small releases. Furthermore, the release baselines are always established through the functional tests and at the end of iteration.
SG 2. Track and Control Changes: Changes to the work products under configuration management are tracked and controlled. **Specific Practices:** • Track change requests • Control configuration items	Pair programming, tests, and on-site customer support are used for tracking and controlling the changes. Moreover, re-factoring practice pushes the source code in the direction of a larger baseline, with more classes and codes in common.
SG 3. Establish Integrity: Integrity of baselines is established and maintained. **Specific Practices:** • Establish configuration management records • Perform configuration audits	Continuous integration practice keeps the system never far from a production state, where the pair should check that their changes do not affect another part of the system developed by another pair of programmers. In addition, coding standard practice keeps the code consistent and easy for the entire team to read, and also helps the XP team to understand all the codes that have been written as basis for the practice of collective ownership. Therefore, the code is easy to read as a coding standard and therefore has its own descriptions. Pair programming, on-site customer and test driven development are informally performing the audits.

4.2.2 Aligning XP Practices to the Level 3 KPAs of CMMI-Dev1.2

Level 3 (Defined process) is a process that is tailored from the organization's set of standard processes according to the organization's tailoring guidelines, and contributes work products, measures, and other process improvement information to the organizational process assets. This level consists of eleven process areas as follows:

- **Requirements Development - P.S**

  The purpose of this process area is *"to produce and analyze customer, product, and product-component requirements"* (CMMI Product Team, 2006).

  As shown in Table 4.7, it can be concluded that the specific goals the requirement development KPA are partially supported by some practices of XP method, which are: planning game, on-site customer, small releases, and test-driven development. However, there is a problem of non-keeping of the requirements specifications by XP method; therefore the requirements specifications remain vague.

Table 4.7: Coverage of the XP Practices to Requirement Development KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Develop Customer Requirements: Stakeholder needs, expectations, constraints, and interfaces are collected and translated into customer requirements. **Specific Practices:** • Elicit needs • Develop the customer requirements	Story cards and functional test are used to specify the user requirements that are elicited by customer, where the developers are always supporting the customer in doing these tasks. In addition, the customer decides which of the features are to be included in each iteration, and also conduct the functional tests at the end of iteration. Thereafter, the developers are responsible for determining the estimations of the features, technical consequences, process, and detailed scheduling. Furthermore, on-site customer, user stories, and iterative development are directly supporting the eliciting and developing of the user requirements.

SG 2. Develop Product Requirements: Customer requirements are refined and elaborated to develop product and product component requirements. **Specific Practices:** • Establish product and product component requirements • Allocate product component requirements • Identify interface requirements	Developers are responsible to refine the customer requirements into product requirement by using the specified task cards. Therefore, analysis, design, planning and testing of the application are used to refine the user requirements to product components requirements during each iteration. Furthermore, the developers are responsible for developing a prototype to explore the architecture possibilities. Based on that, customer and programmers work together in order to design the interface of the product components, where the running test is used to ensure the Interface compatibility at each integration step.
SG 3. Analyze and Validate Requirements: The requirements are analyzed and validated, and a definition of required functionality is developed. **Specific Practices:** • Establish operational concepts and scenarios • Establish a definition of required functionality • Analyze requirements • Analyze requirements to achieve balance • Validate requirements	Pair programming is responsible for analyzing the requirement during the planning phase with assistance from the on-site customer during requirements elicitation. In addition, small releases and on-site customer practices enable the constant analysis and validation of the requirements. Furthermore, test driven development supports the understanding and validating of the requirements. In addition, the functional tests support the establishment of the operational concepts and scenarios. At the end of each iteration, or during the iteration planning meeting, the team's demonstration of the current state of the project, also assists the customer into further specifying and communicating future requirements. However, there is problem of non-keeping of the requirements specifications by XP method.

- **Technical Solution - L.S**

The purpose of this process area is "*to design, develop, and implement solutions to requirements"* (CMMI Product Team, 2006).

As shown in Table 4.8, it can be concluded that the all the specific goals of the technical solution KPA are largely supported by some of XP practices, which are:

simple design, coding standards, on-site customer, pair programming, metaphor, and re-factoring.

Table 4.8: Coverage of the XP Practices to Technical Solution KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Select Product Component Solutions: Product or product component solutions are selected from alternative solutions. **Specific Practices:** • Develop alternative solutions and selection criteria • Select product component solutions	Prototype explores the alternative solutions at the beginning of the project, while re-factoring and iterative development enable the exploration of these solutions through the project. In addition, during the pair programming practice; it is continuously looking for alternative approaches to assure the quality of software product. Re-factoring practice should be used to restructure the system by removing duplications, improving communications, simplifying and adding flexibility if there are any inaccuracies in the code. Furthermore, metaphor, iterative solutions, and test-driven development practices lead to a high quality of technical solutions.
SG 2. Develop the Design: Product or product component designs are developed. **Specific Practices:** • Design the product or product component • Establish a technical data package • Design interfaces using criteria • Perform make, buy, or reuse analyses	Code is used as a design document in XP method, where this design is carried out iteratively. In addition, simple design practices help in making the code design as simple as possible. By coding standard practice, the code should be clear to everybody in the project, in order that all the team members can make changes to it. In addition, Coding standards help the XP team to understand all the codes that have been written as basis for the practice of collective ownership. Moreover, the standard should not be imposed on the team and the code should be written only once as a rule. By simple design, the XP fits the design for the present system features, and is ready for future changes in an incremental or iterative way, where XP focuses on solving today's problems and every piece in the design must be able to justify its existence. By re-factoring practice, the changes of the structure are verified with automated tests which help the programmers to get feedback on the changes.
SG 3. Implement the Product Design: Product components, and associated support	Re-factoring, coding standards, and pair programming are used in implementing the required features that are written by the customers.

documentation, are implemented from their designs. **Specific Practices:** • Implement the design • Develop product support documentation	Pair programming practice enables the reviewing, designing, and testing of the whole code, where the developers start almost immediately to code in a very simple way, trying to maintain the code to be as simple as possible and the coding rules exist and are followed by the programmers. Furthermore, product support documentation is developed if it is requested by the customer. Furthermore, re-factoring, coding standards, pair programming, test driven development and continuous improvement help to enhance the implementation phase and ensure better quality.

- **Product Integration - L.S**

The purpose of this process area is *"to assemble the product from the product components, ensure that the product is integrated, functions properly, and deliver the product"* (CMMI Product Team, 2006).

As shown in Table 4.9, it can be concluded that the specific goals of the product integration KPA are largely supported by some of XP practices, which are: continuous integration, simple design, coding standards, on-site customer, pair programming, and re-factoring.

Table 4.9: Coverage of the XP Practices to Product Integration KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Prepare for Product Integration: Preparation for product integration is conducted. **Specific Practices:** • Determine integration sequence • Establish the product integration environment • Establish product integration procedures and criteria	Continuous integration enables the integration of the changes to the code very often. In addition, the pair should check that their changes do not affect another part of the system developed by another pair of programmers. Coding standards practice helps the XP team to understand all the codes that have been written as basis for the practice of collective ownership to detect the unknown repercussions by automated tests; therefore, this practice increases quality of the code and reduces faults.

SG 2. Ensure Interface Compatibility: The product component interfaces, both internal and external, are compatible. **Specific Practices:** • Review interface descriptions for completeness • Manage interfaces	The functional tests are created by the customer and run at the end of each iteration to ensure interface compatibility. Therefore on-site customer, small iterations, and test-driven development enable the reviewing and managing of the interfaces.
SG 3. Assemble Product Components and Deliver the Product: Verified product components are assembled and the integrated, verified, and a validated product is delivered. **Specific Practices:** • Confirm readiness of product components for integration • Assemble product components • Evaluate assembled product components • Package and deliver the product or product component	By continuous integration, pair programming is responsible for integrating their own code and automated tests are run to ensure that the system is working at 100 %, as a new piece of code is integrated into the code-base as soon as it is ready. In addition, this practice keeps the system never far from a production state. In this respect, one machine should be used only for integration issues for each pair of programmers. Therefore, continuous integration and on-site customer practices help in assembling the product components and deliver the product. During the iterations, the team integrates their work regularly and eliminates the bugs. Therefore, at the end of each iteration, a fully programmed, tested and production worthy version of the system is delivered.

- **Verification - L.S**

The purpose of this process area is *"to ensure that selected work products meet their specified requirements"* (CMMI Product Team, 2006).

As shown in Table 4.10, it can be concluded that the specific goals of the verification KPA are largely supported by some of XP practices, which are: small releases, on-site customer, pair programming, re-factoring, test-driven development, collective ownership, and coding standards.

Table 4.10: Coverage of the XP Practices to Verification KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Prepare for Verification: Preparation for verification is conducted. **Specific Practices:** • Select work products for verification • Establish the verification environment • Establish verification procedures and criteria	Intensive communications between the project team (including the customer) support the preparing and executing of the verification process, whereas each user story that represents a feature in the XP development has associated an acceptance test, which is determined by the XP customer and implemented by the team. Furthermore, a test-first approach and all tests have to be written before the code. Therefore, test driven development and unit test support the enhancement of the verification process by increasing the probability of meeting the verified work to the specified requirements.
SG 2. Perform Peer Reviews: Peer reviews are performed on selected work products. **Specific Practices:** • Prepare for peer reviews • Conduct peer reviews • Analyze peer review data	Pair programming practice dictates that the design, coding, and testing are done by two people working together; therefore, this practice is the most effective of peer reviews possible. In addition, pair programming adopts defensive concepts found in code reading and literate programming. Furthermore, collective ownership and coding standards imply constant peer reviews.
SG 3. Verify Selected Work Products: Selected work products are verified against their specified requirements. **Specific Practices:** • Analyze verification results • Perform verification	Pair programming and testing are considered the main methods for verification process, both of which are performed constantly. In addition, test-driven development helps in the incorporation of defects identification. Small releases enable the customer to test the product; therefore, the correctness of the systems is shown to the customer when all tests are passed. Consequently, the application is continually growing and evolving.

- **Validation - L.S**

The purpose of this process area is *"to demonstrate that a product or product component fulfills its intended use when placed in its intended environment"* (CMMI Product Team, 2006).

As shown in Table 4.11, it can be concluded that the all the specific goals of the validation KPA goals are largely supported by some of XP practices, which are: small releases, on-site customer, pair programming, and test-driven development.

Table 4.11: Coverage of the XP Practices to Validation KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Prepare for Validation: Preparation for validation is conducted. **Specific Practices:** • Select products for validation • Establish the validation environment • Establish validation procedures and criteria	The main objective of validations is performed in XP project through customer participation and frequent releases. In addition, the acceptance by customer is the important criterion for validation. Furthermore, iterations to release, test-driven development, and on-site customer support the activities of the validation process.
SG 2. Validate Product or Product Components: Establish and maintain procedures and criteria for validation. **Specific Practices:** • Perform validation • Analyze validation results	The software products are always validated by the on-site customer, because the customer is integrated into the team; therefore he can validate the software product at the end of each iteration. Test-driven development (especially, customer tests) enables the developer to check the needed requirements and if there is a need for additional requirements. Thereafter, the correctness of the systems is shown to the customer when all tests are passed. Therefore, the participation of customer improves the chances that the product is suitable for use in its intended operating environments.

- **Organizational Process Focus - N.S**

  The purpose of this process area is *"to plan and implement organizational process improvement based on a thorough understanding of the current strengths and weaknesses of the organization's processes and process assets"* (CMMI Product Team, 2006).

  Several researchers (Koch, 2003; Fritzsche & Keil, 2007; and Elshafey & Galal-Edeen, 2008) indicated that this KPA is not addressed by XP method. In addition, Martinsson (2002), Koch (2003), and Omran (2008) argued that XP method does not addresses organization process focus at the organizational level, where XP method focuses on the software engineering process rather than organizational infrastructure issues. Therefore, it can be concluded that this KPA is not supported by XP method, as this process area is related to the organization while XP only applies to a project.

- **Organizational Process Definition + IPPD - P.S**

  The purpose of this process area is *"to establish and maintain a usable set of organizational process assets"* (CMMI Product Team, 2006).

  As shown in Table 4.12, it can be concluded that the specific goals of the Organizational Process Definition + IPPD KPA goals are partially supported by some of XP practices, which are: planning game, on-site customer, and metaphor. However, XP method does not addresses the organizational assets of process definition.

Table 4.12: Coverage of the XP Practices to Organizational Process Definition + IPPD KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Establish Organizational Process Assets: A set of organizational process assets is established and maintained. **Specific Practices:** • Establish standard processes • Establish lifecycle model descriptions • Establish tailoring criteria and guidelines • Establish the organization's measurement repository • Establish the organization's process asset library • Establish work environment standards	By specifying the roles of each member in the team and by XP literature and through various internet resources; it will be easy for the team to know and understand their specified roles during the XP development life cycle. Moreover, the communication between all team members is one of the XP values, whereas the XP contributes a lot to the project members' integration and their close collaboration. Thus, they can benefit from the experience of each other during the development. In addition, the metaphor is responsible for guiding all the activities of the development lifecycle by describing the functionality of the system to help everyone on the project to understand the basic elements and their relationships. Furthermore, the team shares their common understanding from their past experiences. However, organizational assets are outside the scope of the XP method (Omran, 2008).
SG 2. Enable IPPD Management: Organizational rules and guidelines, which govern the operation of integrated teams, are provided. **Specific Practices:** • Establish empowerment mechanisms • Establish rules and guidelines for integrated teams • Balance team and home organization responsibilities	Planning game practice provides the initial planning at the beginning of each iteration. The big boss communicates with the XP team to determine the current situation and to distinguish any difficulties or deficiencies in the process, whereas if an XP team does not produce what they should, the big boss can step in and help them. In addition, the metaphor should help everyone on the project to understand the basic elements and their relationships. Intensive communication, on-site customers and pair programming support the integration between product and process development. Furthermore, the coach is responsible for resolving the issues that may occur within the team, while the tracker role supports tracking inter-group issues.

- **Organizational Training - P.S**

The purpose of this process area is *"to develop the skills and knowledge of people so they can perform their roles effectively and efficiently"* (CMMI Product Team, 2006).

As shown in Table 4.13, it can be concluded that the specific goals of the organizational training KPA are partially supported by some of the XP practices, which are: planning game, collective ownership practices, on-site customer, and metaphor). However, there are deficiencies regarding the assessment of training effectiveness and establishment of training in XP method.

Table 4.13: Coverage of the XP Practices to Organizational Training KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Establish an Organizational Training Capability: A training capability, which supports the organization's management and technical roles, is established and maintained. **Specific Practices:** • Establish the strategic training needs • Determine which training needs are the responsibility of the organization • Establish an organizational training tactical plan • Establish training capability	This specific goal is implicitly supported by pair programming practice, because this practice is responsible for: (1) teaching the team members (especially the new) the skills and intricacies required in the actual project; (2) performing the software development and maintenance activities such as design, code, and test; (3) giving the programmers a tool for sharing and circulating knowledge within the team, and (4) circulating knowledge about new technologies. The training is carried out explicitly during the exploration phase and implicitly during the whole project through coaching and pair programming. Therefore, XP method enhances the organization's training capabilities.

SG 2. Provide Necessary Training: Training necessary for individuals to perform their roles effectively is provided. **Specific Practices:** • Deliver training • Establish training records • Assess training effectiveness	No one can complete his tasks in XP method without organizational training for individual development, where this is the concept of collective ownership. In this respect, pair programming practice and coach role are responsible for delivering the training at the explorations phase and during the whole project. However, there are deficiencies regarding the assessment of training effectiveness and establishment of training.

- **Integrated Project Management + IPPD - L.S**

  The purpose of this process area is *"to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process that is tailored from the organization's set of standard processes"* (CMMI Product Team, 2006).

  As shown in Table 4.14, it can be concluded that the all the specific goals of the integrated project management + IPPD KPA are largely supported by some of XP practices, which are: metaphor, collective ownership, small releases, planning game, on-site customer, and pair programming.

Table 4.14: Coverage of the XP Practices to Integrated Project Management + IPPD KPA

Specific Goal &Practices	Coverage by XP
SG 1. Use the Project's Defined Process: The project is conducted using a defined process that is tailored from the organization's set of standard processes. **Specific Practices:** • Establish the project's defined	XP method defines practices and roles for the development project. Therefore, XP method is based on careful compliance with the XP practices and roles. In addition, planning game, visual charts, and iterative developments support this specific goal, where the customer sets the priority order for the stories, as well as reaches an agreement with the programmers on the contents of the first small release, of the features, technical consequences, process, and the detailed scheduling.

process • Use organizational process assets for planning project activities • Establish the project's work environment • Integrate plans • Manage the project using the integrated plans • Contribute to the organizational process assets	Moreover, the coach enables the project to keep on the right path as well as work to keep people focused on the current features for the actual iteration. In addition, the tracker traces the progress of each iteration and evaluates whether the goal is reachable within the given resource and time constraints, or if any changes are needed in the process. Furthermore, pair programming, collective ownership of the code and the focus on cooperation and communication support the governing of the team operation.
SG 2. Coordinate and Collaborate with Relevant Stakeholders: Coordination and collaboration of the project with relevant stakeholders are conducted. **Specific Practices:** • Manage stakeholder involvement • Manage dependencies • Resolve coordination issues	The developers, testers, customers, and all the relevant stakeholders are integrated and coordinated in XP method, where everybody in a XP project takes responsibility for the code in the whole system. In addition, the metaphor helps everyone on the project to understand the basic elements and their relationships. Furthermore, XP method enables the democracy in the leadership mechanisms between the development team members. Nevertheless, the big boss and the customer have authority to decide on high level issues.
SG 3. Apply IPPD Principles: The project is managed using IPPD principles. **Specific Practices:** • Establish the project's shared vision • Establish the integrated team structure • Allocate requirements to integrated teams	Communication between all team members is one of the XP values. XP contributes a lot to the project members' integration and their close collaboration. Furthermore, the collaboration and intensive communication within the team help to establish a shared vision. In addition, XP method supports the establishment of a self-organizing cross-functional team in which all integrated relevant stakeholders are involved.

- **Risk Management - L.S**

The purpose of this process area is *"to identify potential problems before they occur, so that risk-handling activities may be planned and invoked as needed across the life*

of the product or project to mitigate adverse impacts on achieving objectives" (CMMI Product Team, 2006).

As shown in Table 4.15, it can be concluded that the specific goals of the risk management KPA are largely supported by some of XP practices, which are: small releases, pair programming, on-site customer, simple design, re-factoring, coding standards, continuous integration, simple design, and test driven development.

Table 4.15: Coverage of the XP Practices to Risk Management KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Prepare for Risk Management: Preparation for risk management is conducted. Preparation is conducted by establishing and maintaining a strategy for identifying, analyzing, and mitigating risks. **Specific Practices:** • Determine risk sources and categories • Define risk parameters • Establish a risk management strategy	The XP method does not explicitly state how the specific practices of this goal are to be conducted. Nevertheless, XP project makes some sort of preparation to avoid the risks based on pair programming, re-factoring, coding standards, continuous integration, simple design, and test driven development practices. In addition, the idea behind small releases is to get the system in production on time in order to get constant feedback from the customer, as well as to avoid risks, and minimize effort necessary to change the effect. Thus, the risks are discussed at the end-of-iteration.
SG 2. Identify and Analyze Risks: Risks are identified and analyzed to determine their relative importance. **Specific Practices:** • Identify risks • Evaluate, categorize, and prioritize risks	During the planning game practice, the XP method enforces the identifications and analysis of risks, where developers are making technical decisions (evaluating risk factors and estimating the effort). In addition, during design activity; the spike solution is always created to reduce risks of technical problems. Therefore, these solutions encourage the project teams to address the difficult or unknown aspects of an effort first in order to uncover potential problems as soon as possible. A spike solution is *"a preliminary or experimental effort to prove that a specific technology or an approach will actually work in a particular situation"* (Baker & Thomas, 2007).

SG 3. Mitigate Risks: Risks are handled and mitigated, where appropriate, to reduce adverse impacts to achieve objectives. **Specific Practices:** • Develop risk mitigation plans • Implement risk mitigation plans	Short iterations are a potent instrument to mitigate risks. In addition, the intensive communications enable the project team to identify and mitigate the risks. Therefore, the incremental and evolutionary XP lifecycle enable the developers to identify, mitigate, and manage the risks efficiently.

- **Decision Analysis and Resolution - L.S**

  The purpose of this process area is *"to analyze possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria"* (CMMI Product Team, 2006).

  As shown in Table 4.16, it can be concluded that the XP method largely supports the specific goal of the decision analysis and resolution KPA in a different way of CMMI-Dev1.2 suggests by XP practices and roles which are: pair programming, simple design, and on-site customer practices, and big boss role.

Table 4.16: Coverage of the XP Practices to Decision Analysis and Resolution KPA

Specific Goal &Practices	Coverage by XP
SG 1. Evaluate Alternatives: Decisions are based on an evaluation of alternatives using established criteria. Issues requiring a formal evaluation process may be identified at any time. **Specific Practices:** • Establish guidelines for decision analysis • Establish evaluation criteria • Identify alternative solutions • Select evaluation methods • Evaluate alternatives • Select solutions	The ability to adapt quickly to new situation is preferred by agile methods over a formal evaluation process; whereas, planning where you need to plan; designing what is important; and coding what can pass the unit tests; and user-story, are the main activities in XP that depend on the tacit knowledge of the XP team. Therefore, XP identifies and evaluates alternatives informally and not in the way CMMI suggests, where coding can be used to figure out the most suitable solution. XP would advocate having several alternatives to a programming problem. One should simply code all solutions and determine with automated tests which solution is most suitable, where simple design is in much a direct translation of the XP value as it is for simplicity in a software practice. It calls for the programmers to make design decisions for the current problem and the on-site customer helps to make decisions and answer the programmers' questions. Furthermore, the big boss is responsible to make the decisions, where a big boss communicates with the XP team to determine the current situation, and to distinguish any difficulties or deficiencies in the process. If an XP team does not produce what they should, a big boss can step in and help them. Therefore, it can be concluded that this specific goal is achieved by XP method in a different way from CMMI.

4.2.3 Aligning XP Practices to the Level 4 KPAs of CMMI-Dev1.2

Level 4 (Quantitatively managed) is a process that is controlled using statistical and other quantitative techniques. Quantitative objectives for quality and process performance are established and used as criteria in managing the process. Quality and process performance is understood in statistical terms and is managed throughout the life of the process. This level consists of the following two process areas:

- **Organizational Process Performance - P.S**

The purpose of this process area is *"to establish and maintain a quantitative understanding of the performance of the organization's set of standard processes in support of quality and process-performance objectives, and to provide the process performance data, baselines, and models to quantitatively manage the organization's projects"* (CMMI Product Team, 2006).

As shown in Table 4.17, it can be concluded that the specific goal of the of organizational process performance KPA is partially supported by XP practices, which are: planning game, small releases, and re-factoring. However, XP method focuses on individual rather than issues that are as process oriented as this processes area, where the important metrics of processes performance are not covered by XP method.

Table 4.17: Coverage of the XP Practices to Organizational Process Performance KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Establish Performance Baselines and Models: Baselines and models, which characterize the expected process performance of the organization's set of standard processes, are established and maintained. **Specific Practices:** • Select processes • Establish process performance measures	This process area is strongly related with the process and product quality assurance process area. In planning game, the XP team implies schedules exactly as much work (or as many "units") as the team's average velocity to extract the project team's velocity, where the velocity serves as the foundation in updating the project plan to be realistic and adhere to the historical performance of the team. The tracker gives feedback in XP method by tracing the estimates made by the team, tracing the progress of each iteration and evaluating whether the goal is reachable within the given resource and time constraints, or if there are needs for any changes in the process. In addition, the tracker is responsible for collecting the metrics of the project performance at the end of each iteration (or more often if required), and conveying the results of these metrics to the project team by using a spreadsheet application which always consists of the

• Establish quality and process-performance objectives • Establish process-performance baselines • Establish process-performance models	estimates and actual achievements as well as the overall project plan. Furthermore, the coach's role is to make sure that the programmers are working efficiently and effectively, and in instances where a programmer is not on par with his abilities or estimates, it is important for the coach to find a solution for this as soon as possible. However, XP method focuses on individuals rather than issues that are process oriented. The important metrics of processes performance are not covered by XP method.

- **Quantitative Project Management - L.S**

The purpose of this process area is *"to quantitatively manage the project's defined process to achieve the project's established quality and process-performance objectives"* (CMMI Product Team, 2006).

As shown in Table 4.18, it can be concluded that the specific goal of the quantitative project management KPA is largely supported by XP practices, but in a different way of CMMI-Dev1.2 suggests. These practices are: on-site customer, planning game, test driven development, re-factoring, continuous integration, collective code ownership, and metaphor practices.

Table 4.18: Coverage of the XP Practices to Quantitative Project Management KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Quantitatively Manage the Project: The project is quantitatively managed using quality and process-performance objectives. **Specific Practices:** • Establish the project's objectives	In general, statistical methods focus on defined processes that rely on the law of big numbers and on averaging out effects in large teams. Nevertheless, XP method is used by SSDFs that have small software projects; therefore, the specific goals of this KPA can be achieved by XP practices without achieving all specific practices. In this respect, several metrics are included by default for any XP team in the form of user story and engineering task estimates (define the requirement, architecture, design, coding, and testing) in conjunction with how many units of work the team and each individual

• Compose the defined process • Select the sub-processes that will be statistically managed • Manage project performance	programmer have been able to deliver during each iteration. Furthermore, the objectives of this process area can be achieved by XP practices as follows: (1) planning game and system metaphor estimate new iteration size and development time; (2) system metaphor, pair programming, test driven development, and continuous integration that analyzes cause and effect relationships in terms of productivity or number of defects; and (3) on-site customer, re-factoring, continuous integration, and collective code ownership to calculate the variance in productivity or quality across different system iterations. In addition, a tracker is responsible to trace the estimates made by the team and give feedback on how accurate they are in order to improve future estimations, and also to trace the progress of each iteration and evaluate whether the goal is reachable with the given resource and within time constraints. Accordingly, the specific goals of this KPA are supported by XP method in a different way from CMMI specific practices.
SG 2. Statistically Manage Sub-process Performance: The performance of selected sub- processes within the project's defined process is statistically managed. **Specific Practices:** • Select measures and analytic techniques • Apply statistical methods to understand variation • Monitor performance of the selected sub-processes • Record statistical management data	

4.2.4 Aligning XP Practices to the Level 5 KPAs of CMMI-Dev1.2

Level 5 (Optimizing process) is a quantitatively managed process that is improved based on an understanding of the common causes of variation inherent in the process. The focus of an optimizing process is to continually improve the range of process performance through both incremental and innovative improvements. This level consists of two process areas as follows:

- **Organizational Innovation and Deployment - P.S**

  The purpose of this process area is *"to select and deploy incremental and innovative improvements that measurably improve the organization's processes and technologies. The improvements support the organization's quality and process-*

performance objectives as derived from the organization's business objectives" (CMMI Product Team, 2006).

As shown in Table 4.19, it can be concluded that the specific goals of the organizational innovation and deployment KPA are partially supports by re-factoring practice, and feedback and simplicity values. However, the process improvements and adaptations in XP method are made only within projects and are not documented, because XP method does not have improvement strategy to improve the software process.

Table 4.19: Coverage of the XP Practices to Organizational Innovation and Deployment KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Select Improvements: Process and technology improvements, which contribute to meeting quality and process-performance objectives, are selected. **Specific Practices:** • Collect and analyze improvement proposals • Identify and analyze innovations • Pilot improvements • Select improvements for deployment	Re-factoring practice supports the incremental improvements to improve the processes at the team level, where the simplicity and feedback values enable these improvements by the project team. However, in XP method; the process improvements and adaptations are made only within projects and are not documented, because XP method does not have improvement strategy to improve the software process.
SG 2. Deploy Improvements: Measurable improvements to the organization's processes and technologies are continually and systematically deployed. **Specific Practices:** • Plan the deployment • Manage the deployment • Measure improvement effects	

- **Causal Analysis and Resolution - L.S**

The purpose of this process area is *"to identify causes of defects and other problems and to take action to prevent them from occurring in the future"* (CMMI Product Team, 2006).

As shown in Table 4.20, it can be concluded that the specific goals of the causal analysis and resolution KPA are largely supports by XP practices, which are: on-site customer, planning game, test driven development, and continuous integration.

Table 4.20: Coverage of the XP Practices to Causal Analysis and Resolution KPA

Specific Goal &Practices	**Coverage by XP**
SG 1. Determine Causes of Defects: Root causes of defects and other problems are systematically determined. **Specific Practices:** • Select defect data for analysis • Analyze causes	Defect prevention is addressed through some practices of XP method such as: test driven development, continuous integration, and pair programming. In addition, the feedback during rapid cycle supports the defect prevention. XP method focuses on stable source code and baselines, but if there is any spotted defect through the programming or integration, a test case will be written to reproduce the defect. By trying to find common denominator in the code, similar bugs might be identified. In addition, by programmers' tests of the test-driven development practices, there is a need to create the tests first and then code (test-first design), and then turn the cycle to add one more test followed by the code. Therefore, the test-first design and writing acceptance tests verify that the software is fully functional and without defects. Accordingly, it can be concluded that main goals of this process area are supported by XP method.
SG 2. Address Causes of Defects: Root causes of defects and other problems are systematically addressed to prevent their future occurrence. **Specific Practices:** • Implement the action proposals • Evaluate the effect of changes • Record data	

4.2.5 Summary of Alignment XP practices to the Specific Goals of CMMI-Dev1.2

Based on the results of theses alignments, Table 4.21 summarizes the results of coverage ratios of XP practices to CMMI-Dev1.2 KPAs.

Table 4.21: Coverage ratios or XP practices to CMMI-Dev1.2

No.	CMMI-Dev1.2 KPAs	Coverage Ratio
1	Project Planning	L.S
2	Project Monitoring And Control	L.S
3	Configuration Management	L.S
4	Technical Solution	L.S
5	Product Integration	L.S
6	Verification	L.S
7	Validation	L.S
8	Integrated Project Management +IPPD	L.S
9	Risk Management	L.S
10	Decision Analysis And Resolution	L.S
11	Quantitative Project Management	L.S
12	Causal Analysis And Resolution	L.S
1	Requirement Management	P.S
2	Measurement And Analysis	P.S
3	Process And Product Quality Assurance	P.S
4	Requirements Development	P.S
5	Organizational Process Definition + IPPD	P.S
6	Organizational Training	P.S
7	Organizational Process Performance	P.S
8	Organizational Innovation And Deployment	P.S
1	Supplier Agreement Management	N.S
2	Organizational Process Focus	N.S

As it shown in Table 4.21, it can be concluded that the KPAs of CMMI-Dev1.2 can be largely supported, partially supported, or not-supported by the practices of the XP method as follows:

- **Largely Supported KPAs**

  There are twelve KPAs of CMMI-Dev1.2 that are largely supported by the XP method. These are: project planning; project monitoring and control; configuration management; technical solution; product integration; verification; validation; integrated project management +IPPD; risk management; decision analysis and resolution; quantitative project management; and causal analysis and resolution.

- **Partially Supported KPAs**

  There are eight KPAs of CMMI-Dev1.2 that are partially supported by the XP method, and these are: requirement management; measurement and analysis; process and product quality assurance; requirements development; organizational process definition +IPPD; organizational training; organizational process performance; and organizational innovation and deployment.

- **Not-Supported KPAs**

  There are two KPAs of CMMI-Dev1.2 that are not supported by the XP method, and these are: supplier agreement management and organizational process focus.

Based on the supported levels of this alignment, there is a need to cover the partially and not-supported KPAs of CMMI-Dev1.2 by adding new related software development,

management, and improvement additions to the XP method and taking into account the suitability of these activities for SSDFs. Section 4.3 explains the adaptation of the EPA processes of extending XP method.

4.3 Adapting the Extension-Based Approach (EBA) to Extend XP Method

In this study, the EBA has been adapted for extending XP method to fulfill the partially and not-supported KPAs that are discussed in Section 4.2.

As discussed in Section 3.6, three main processes were used by the adapted EBA in extending XP method, which are:

- **P1:** Specify the extension requirements. Section 4.3.1 discusses this process.
- **P2:** Select & apply the required additions. Section 4.3.2 discusses this process.
- **P3:** Verify the Extended-XP method. Section 5.5.1.2 discusses this process.

4.3.1 The Required Additions to Fulfill the Partially and Not-Supported CMMI-Dev1.2 KPAs

Researchers such as Stephens (2001), Martinsson (2002), Vitoria (2004), Fritzsche and Keil (2007), Hearty (2008), and Omran (2008) indicate to the required software development, management, and improvement additions to fulfill the partially and not-supported CMMI-Dev1.2 KPAs by extending XP method. Therefore, this research used the useful suggestions for extending XP method to fulfill the missing KPAs of CMMI-Dev1.2. In addition, the descriptions of XP method (Beck, 2000; Jeffries et al., 2002) and CMMI-Dev1.2 (CMMI Product Team, 2006) were taken into account during the

extension of XP method to ensure that the required additions were acceptable for XP principles and to know the requirements of the specific goals of CMMI-Dev1.2 KPAs. Sections 4.3.1.1 and 4.3.1.2 discuss the required additions that had to be included in covering the partially and not-supported KPAs.

4.3.1.1 Covering the Partially Supported KPAs

This level consists of eight KPAs of CMM-Dev1.2 that have been partially supported by XP practices. The following points illustrate the required additions that are needed to fulfill the specific goals for each partially supported KPA:

- **Requirement Management**

  As shown in Section 4.2.1 about the requirement management KPA, XP method does not have repository to keep the required data of user stories. In this respect, Stephens (2001) argued that the user requirements in XP method are written in a high level of abstraction using use stories; therefore there are problems in tracing the status of these requirements especially by test-driven development practices. In addition, Vitoria (2004) believed that the traceability is not developed from the requirements to the code in XP lifecycle. Therefore, there is a need to relate the XP phases with the necessary requirement's specifications (Stephens, 2001; Vitoria, 2004).

  Furthermore, Fritzsche and Keil (2007) argued that there is a need to create simple project repository as a tool before the exploration phase of XP method to be responsible for keeping all the customer's requirements, modifying the changes on

these requirements and documenting the status of each requirement, that include the related functions, interfaces, objects, people, processes, and work products of the requirements. Thus, this repository enables the project team to trace the user requirements at any needed time. In this respect, Qureshi (2011) argued that Microsoft Office could be suitable to be used by XP team as a simple repository during the development lifecycle.

- **Measurement and Analyses**

  As shown in Section 4.2.1 about the measurement and analysis, XP method does not have repository to store the measurement data. In this respect, Vitoria (2004), and Fritzsche and Keil (2007) argued that measurements of the process can be developed in XP method by using the velocity of every developer using XP lanner; however the results of these measurements are not documented. Therefore, there is a need to keep the measurements data (Vitoria, 2004; Wong & Hasan, 2007), and this can be done by using a simple project repository at the end of each release to store the measurements data. Furthermore, this KPA is related to the requirement management KPA; because both require the presence of a data repository.

- **Process And Product Quality Assurance**

  As shown in Section 4.2.1 about the process and product quality assurance KPA, XP does not demand an explicit evaluation of processes, work products, and services against the applicable process descriptions. Therefore, there is a need for several metrics at the end of iteration to release of each project to achieve the scrutiny of the software products and development process activities (Stephens,

2001; Vitoria, 2004). In this respect, Hearty (2008) and Martinsson (2002) indicated to the required metrics that could be appropriate for objectively verifying the products and the processes, which are:

- Release plan adherence.
- Percentage of test cases that are running successfully (number of successful test cases/ numbers of total test cases).
- Percentage of acceptance tests that are running successfully (number of successful acceptance tests/ number of total acceptance tests).
- Length of pair programming sessions (average time of each pair programming session).
- Project velocity (actual time of the implemented user stories of all iteration / estimated time of all user stories of all iterations).

Additionally, there is a need to keep the results of these metrics in data storage to be used as guidance for incoming projects. The responsibility for these activities could very well lie on the coach, with the assistance of the tracker (Fritzsche & Keil, 2007). In addition, it is important to keep these responsibilities separated from the developers and testers. The coach or tracker that wants to take on the tasks of process and product quality assurance measurements should therefore not be assigned for programming tasks within the same project (Martinsson, 2002).

Furthermore, in XP method there are no strict guidelines for the resolution of non-compliance issues and establishing records of quality assurance activities (Fritzsche & Keil, 2007). Therefore, there is need to convey the metrics through defined channel to the affected parties and senior management. In this respect, Martinsson (2002) suggests that the results of these metrics are most conveniently posted to the

team using a white-board or an exposed wall in a central location during the development process. When metrics out of the ordinary occur, it is important for the coach to communicate these findings, either directly to the affected party or during the daily stand-up meeting so that these issues can be resolved. If no satisfactory solution is found, the issue is brought to the customer or project manager, depending on the nature of the issue. If no solution can be found at this level, senior management will be presented with the issue.

- **Requirement Development**

  As shown in Section 4.2.2 about requirement development KPA, there is problem of non-keeping of the requirement specification (Fritzsche & Keil, 2007; Omran, 2008). Therefore, this KPA is related to the requirement management KPA; because both require the presence of a data repository. In this respect, the user requirement specifications are need to be kept in a data repository at the exploration phase of XP method to help customers and developers to develop the customer's requirements and keep the changed customer requirements in this repository to know the current status of each requirement (Altarawneh & Shiekh, 2008).

- **Organizational Process Definition +IPPD**

  As shown in Section 4.2.2 about the organization process definition +IPPD KPA, the organizational assets of process definition are outside the scope of the XP method itself (Fritzsche & Keil, 2007). Therefore, this process area is most easily to be supported by buying copies of the XP books (i.e. Extreme Programming Explained, Extreme Programming Installed, and Planning Extreme Programming)

and making these books available within the firm during the development lifecycle (Boehm & Turner, 2003; Martinsson, 2002). In addition, Paulk (2001) believed that the various XP-related books, articles, courses, and Web sites will be suitable for the organization process definition.

As for the additions which will be added to extend the XP method in this study, there is a need also to support the team members with the descriptions of the Extended-XP method. Thus, they can use this description during the Extended-XP development lifecycle to enable them to apply their roles in the right way.

- **Organizational Training**

  As shown in Section 4.2.2 about the organizational training, there are deficiencies regarding the establishment of records and the assessment of training effectiveness (Fritzsche & Keil, 2007). Therefore, there is the need to have simple training process before the first phase XP method to train and educate the project team about the right implementation of the XP development lifecycle (Altarawneh & Shiekh, 2008, Vitoria, 2004).

  Humphrey (1998) mentioned to the importance of Software Engineering Process Group (SEPG) members for establishing the definition, control, training, and improvement tasks needed to launch an improvement program. Therefore, the responsibility for performing and coordinating these activities once again fall upon the SEPG who are responsible for arranging the required organizational training, conducting the needed training for the project team, assessing the training

effectiveness, and recording the data in the data repository (CMMI Product Team, 2006; Martinsson, 2002).

- **Organizational Process Performance**

  As shown in Section 4.2.3 about the organization process performance, XP method focuses on individuals rather than issues that are process oriented in this process area, where the important metrics of process performance are not covered by XP method (Elshafey & Galal-Edeen, 2008; Fritzsche & Keil, 2007).

  This process area is strongly related with the process and product quality assurance process area. Therefore, to fulfill the specific goal of this KPA, there is a need to have simple metrics at the end of each release, where the responsibility for these metrics could very well lie on the coach, with the assistance of the tracker (Martinsson, 2002). These metrics are: (1) calculating the differences between estimated and actual time spent on user stories or tasks; (2) calculating the velocity of the project; and (3) calculating the number of failed acceptance tests (Vitoria, 2004; Hearty, 2008). Furthermore, Martinsson (2002) indicated to the need for keeping the results of these metrics in project repository for making future estimates of similar user stories

- **Organizational Innovation And Deployment**

  As shown in Section 4.2.4 about the organizational innovation and deployment, the process improvements and adaptations in XP method are made only within projects, not documented, and not propagated to the whole firm (Elshafey & Galal-

Edeen, 2008; Omran, 2008). Therefore, XP method does not have improvement strategy to improve the software process (Fritzsche & Keil, 2007).

In order to fulfill the specific goals of this KPA, there are needs for several improvement practices (CMMI Product Team, 2006; Martinsson, 2002), which are: (1) establishing simple training and incentive programs before the first phase of the development method that making everyone within the organization is aware of their responsibility to identify the process improvement opportunities; (2) teaching the project team about writing proposals of their improvement suggestions for the current software development process; and (3) examining and analyzing the suggested process improvement opportunities as a whole and testing the important suggestions in pilot projects to select which ones to implement. Thus, the useful improvements will be made to the organization's standard software process and the defined software processes, as well as communicated through training courses within the firm. Accordingly, the modified software process will be ready to be used for the next projects.

In addition, there is a need for SEPG members consisting of software engineering representatives to be responsible for carrying out the activities of improvement the process (Martinsson, 2002).

4.3.1.2 Covering the Not-Supported KPAs

This group consists of three KPAs of CMM-Dev1.2 that are not supported by XP method. Accordingly, there are needs for new activities to fulfill all the KPAs of this group. The following points explain the needed activities to fulfill these process areas.

- **Supplier Agreement Management**

  This KPA addresses the acquisition of significant products and product components not delivered to the project's customer but are used to develop and maintain the products or services such as: development tools and test environments. However, XP method does not address this KPA (Elshafey & Galal-Edeen, 2008; Omran, 2008); because this KPA is does not deal with the development process, while the XP method only addresses development processes.

  In order to fulfill the specific goals of this KPA, there is need to have supplier agreement management process after the exploration phase in XP method to help in managing and selecting the required product, product components, and services that are explored by programmers. CMMI Product Team (2006) argued that the required supplying process is suitable to be implemented by the SEPG members of the software firms. Therefore, there are needs for several activities to achieve the specific goals of this KPA (CMMI Product Team, 2006), which are: (1) Determining the type of acquisition that will be used for the products to be acquired; (2) Selecting suppliers; (3) Establishing and maintaining agreements with suppliers; (4) Executing the supplier agreement; (5) Monitoring selected supplier

processes; (6) Evaluating selected supplier work products; (7) Accepting delivery of acquired products; and (8) Transitioning acquired products to the project.

- **Organization Process Focus**

  This KPA is not addressed by XP method, because this KPA is related to the organization, while XP method just applies to a project (Elshafey & Galal-Edeen, 2008; Vitoria, 2004). In addition, Fritzsche and Keil (2007) indicated that the improvements in XP method are often executed during the current project, therefore the other projects can benefit if people are moved between projects. In addition, this KPA is related to the large organizations which have too many projects. In this case, not all the people can benefit from a particular project's experience, where the information is not permanent since people can retire or change organization (Fritzsche & Keil, 2007).

  As for SSDFs, to achieve the main goals of this KPA, there is a need to establish organization repository for extracting the best practices of the current project to use for incoming projects or by institutionalizing the exchange of lessons learnt between projects (Fritsch & Keil, 2007). In addition, it will be suitable for SPEG members in these firms are in carrying out the tasks of managing the process used at an organizational level (CMM Product Team, 2002; Martinsson, 2002).

As results of the works discussed in sections 4.3.1.1 and 4.3.1.2 about the required software development, management, and improvement additions need to be added to the XP method to fulfill the partially and not-supported KPAs of CMMI-Dev1.2, Table 4.22

summarizes these additions and presents the suggested position of these additions based on XP method phases.

Table 4.22: Required Additions to Fulfil the Partially and Not-Supported KPAs of CMMI-Dev1.2

Partially & Not-Supported KPAs	**Required Software Development, Management, and Improvement Additions (Positions based on XP method phases in Italic)**
Requirement Management	Creating simple project repository before the *exploration phase* of XP method.
	Using project repository during *all the phases of XP method.*
Measurement And Analysis	Storing the measurement data in project repository after the *iteration to release phase.*
Process And Product Quality Assurance	Using several simple metrics for objectively verifying the products and the process during the *productionizing phase.*
	Conveying the metrics through defined channels to the affected parties and senior management during the *productionizing phase.*
Requirements Development	Storing the requirement specifications in project repository during the *exploration and planning phases.*
	Using project repository during *all the phases of XP method.*
Organizational Process Definition +IPPD	Supporting the project team with the required books of XP method and the description of Extended-XP.
	Using guidance for the development methodology *during all the phases of XP method.*
Organizational Training	Training the project team on the development methodology before the *exploration phase.*
Organizational Process Performance	Using several simple metrics to conduct the process performance during the *productionizing phase* (related to process and product quality assurance).
Organizational Innovation And Deployment	Writing proposals of improvement suggestions during *all the phases of XP method.*
	Executing the improvement process after the *productionizing phase.*
Supplier Agreement Management	Using process for supplying unavailable development tools and technologies after the *exploration phase*
Organizational Process Focus	Extracting the best practices of the current project after the *productionizing phase*

4.3.2 Extending XP method

Prior to starting to extend the XP method by adding the required software development, management, and improvement additions that are needed to fulfill the missing KPAs of CMMI-Dev1.2, there is a need to ensure that the new phases of the proposed Extended-XP method are familiar with other phases of the generic development method. Thus, it is important to take into account the main activities of the popular software development process models such as Waterfall, Spiral, Incremental, and Prototyping as a guideline to extract the new phases of the proposed Extended-XP method.

Sommerville (2011) mentions that the software process consists of four general activities which are:

- Software specification: this activity is used to establish the required services from the system, and determine the constraints of system operations and development. Software specification has two levels; level for high-end users and customer needs level for system developers;
- Software development: this activity is used to convert and translate the system specifications to the executable system;
- Software validation: this activity is used to show if the system is achieving its specifications and meeting customer needs through testing process; and
- Software evolution: this activity is used to maintain and develop the system so that the system can meet circumstantial changes such as requirement changes and customer needs.

In addition, Pressman (2009) identifies the main activities of the generic software process models (Waterfall, Spiral, incremental, and prototyping) which are: (1) Communication: project initiation, requirement gathering; (2) Planning: estimating, scheduling, tracking; (3) Modelling: analysis, design; (4) Construction: code, test; and (5) Deployment: delivery, support, feedback.

Based on the generic phases of the popular software development methodologies, there is a need to extract the new phases of the proposed Extended-XP method and harmonize these phases to be a comprehensive phases for all these methodologies. Table 4.23 presents the relations between the phases of the popular software development methodologies and the required additions that are needed to be covered by Extended-XP phases and also presents the new phases of the proposed Extended-XP method extracted from the popular methodologies. In this respect, the distributions of the new software development, management, and improvement additions were done based on the need for these additions during the software development lifecycle.

The typical XP method life cycle consists of six phases: exploration, planning, iterations to release, productionizing, maintenance, and death. However, the proposed Extended-XP method only consists of the four phases: requirement management phase, development phase, product delivery and product & process efficiency phase, system and process evolution phase. Figure 4.1 shows the phases of the proposed Extended-XP method. Sections 4.3.2.1 to 4.3.2.4 explain these phases in detail.

Table 4.23: Extracting the Phases of the Proposed Extended-XP Method

Source	**Software Process Activities**											
Waterfall, Spiral, Incremental, Prototyping (Pressman, 2009)	**Communication**		**Planning**	**Modeling**		**Construction**	**Deployment**					
General Software Development Process Activities (Sommerville, 2011)	**Specification**			**Development**			**Validation**		**Evolution**			
XP Phases (Beck, 2000)	**Exploration**		**Planning**	**Iteration to Release Phase**			**Productionizing**		**Maintenance**		**Death**	
Positions of the Required Additions to Fulfill the Partially and Not-Supported KPAs of CMMI-Dev1.2												
KPAs	**Pre-Method**											
Requirement Management	Creating project repository											
		Using project repository (during all the development methodology)										
Measurement And Analysis							Storing the measurement data in project repository					
Process And Product Quality Assurance							Using some metrics for objectively verifying the products and the process					
							Conveying the metrics through defined channels to the affected parties and senior management.					
Requirements Development		Storing the requirement specifications in project repository										
		Using project repository (during all the development methodology)										

Organizational Process Definition +IPPD	Supporting the project team with the description of Extended-XP					
		Using the description of Extended-XP method during the software development lifecycle.				
Organizational Training	Training the project team on the development methodology					
Organizational Process Performance					Using some metrics to conduct the process performance.	
Organizational Innovation And Deployment		Writing proposals of improvement suggestions				
						Executing the improvement process
Supplier Agreement Management		Using process for supplying unavailable development tools and technologies.				
Organizational Process Focus						Extracting the best practices and lessons learnt of the current project
Extracting of the Generic Phases of the Proposed Extended-XP method.						
The Proposed Extended-XP Phases	Pre-Extended-XP	**Requirement Management**		**Development**	**Product Delivery and Product & Process Efficiency**	**System and Process Evolution**

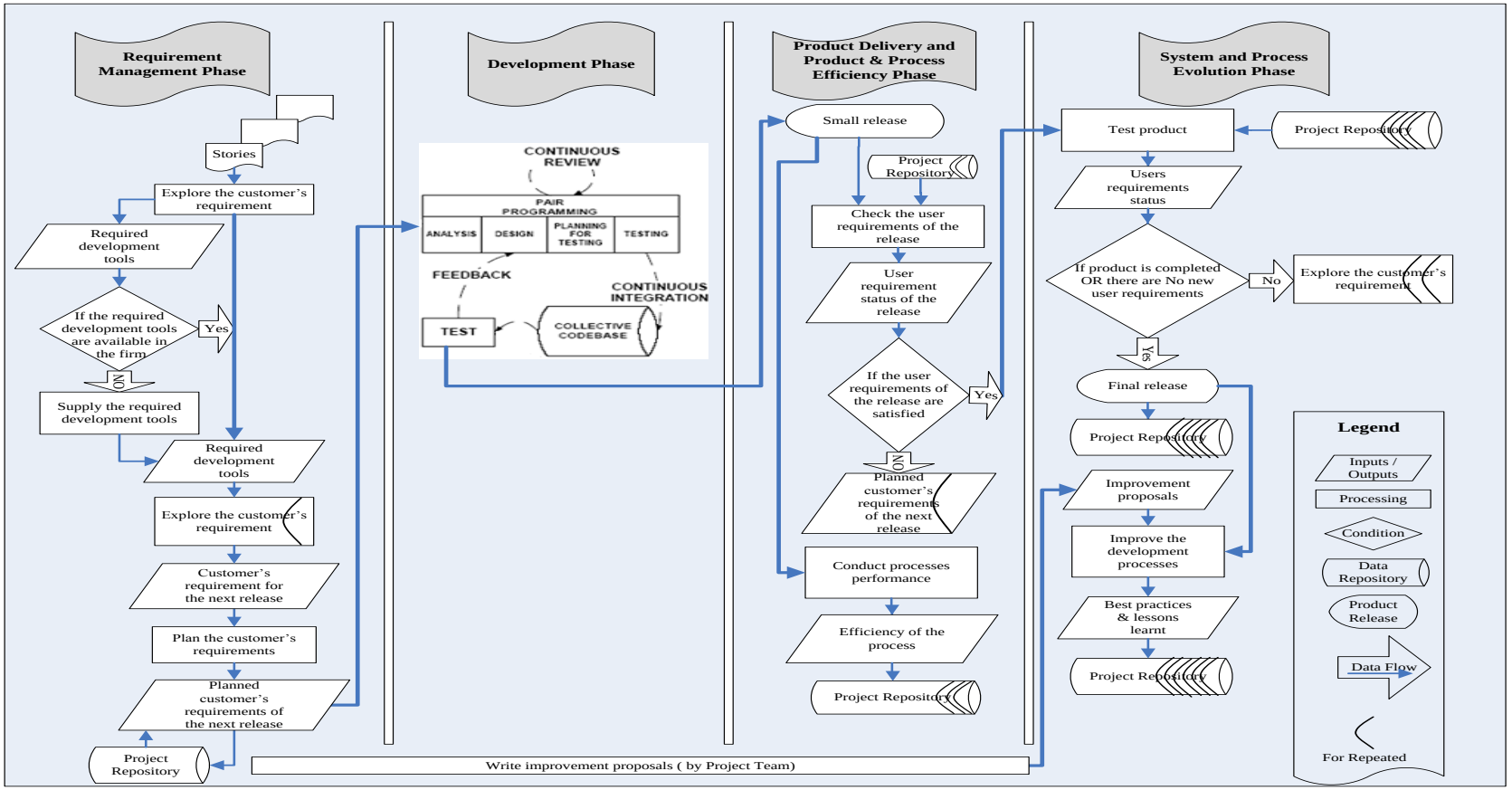

Figure 4.1: The Proposed Extended-XP Method Phases

Prior to starting the phases of the proposed Extended-XP method, there is a need to identify and create the required simple project repository which is needed to keep the training records, user requirement data, and to arrange the changes that will happen on the user requirements data during the project. Then, there is the need to train the project team to implement the proposed Extended-XP method in the right way. Therefore, training courses for the whole project team are important to ensure that they have good knowledge about their roles. Moreover, there is the need to support the project team with the required XP method books and the description of the proposed Extended-XP method.

In this aspect, SEPG members are responsible for: establishing planning for training the developers; estimating the time required for training; determining if there is need for outsourced professional team in the training process; training the developers on the proposed Extended-XP method; training the project team on writing the improvement suggestion during the project development; assessing the project teams efficiency; and recording the training efficiencies in the project repository. As a result of the training process, the teams will be ready to implement the proposed Extended-XP method in the right way.

4.3.2.1 Phase One: Requirement Management

This phase has all the development and management activities of the exploration and planning phases of XP method that are mentioned in Section 2.5.2.1, and also there are new additions that have been added to fulfill the requirement of partially and not-

supported KPAs of CMMI-Dev1.2. In this phase, there are three main processes as follows:

- **Explore Customers Requirements Process**

  Customers write story cards which contain features to be implemented in the first release (customer's requirements), where each story card contains one feature. Then, programmers analyze the user requirements and identify the required development tools that will be needed to be used in the project. Based on these required development tools, programmers need to select the available development tools that are already available in the firm, and which of the unavailable development tools need to be acquired from suppliers.

- **Supply the Required Development Tools Process**

  This process is used to support the project with the unavailable required development tools or services. In this aspect, programmers are responsible to identify the unavailable required development tools and services. Then, the SEPG members are responsible for implementing the supporting process. During this process, there are several activities that need to be done if there is a need to supply the project with new development tools or services. In implementing the supplying process, there is a need for the following:

  - Determining Acquisition Type: determining the type of acquisition that will be used for the products to be acquired such as: (1) purchasing commercial off-the-shelf (COTS) products; (2) obtaining products through a contractual agreement; (3) obtaining products from an in-house vendor; and (4) obtaining products from the customer.

- Selecting Suppliers: selecting suppliers based on an evaluation of their ability to meet the specified requirements and established criteria.
- Establishing Supplier Agreements: establishing and maintaining formal agreements with the supplier.
- Executing the Supplier Agreement: performing activities with the supplier as specified in the supplier agreement.
- Monitoring Selected Supplier Processes: selecting, monitoring, and analyzing processes used by the supplier.
- Executing the Supplier Agreement: selecting and evaluating work products from the supplier of custom-made products.
- Accepting the Acquired Product: ensuring that the supplier agreement is satisfactory before accepting the acquired product.
- Transition Products: transiting the acquired products from the supplier to the project.

Based on the supplying process; the required development tools or services will be known by the project team. Then, the project team needs to familiarize themselves with the required development tools, services, practices, and technologies. Furthermore, based on the customers' requirements, the developers need to develop a prototype to explore the architecture possibilities.

- **Plan the Customer's Requirements Process**

This process is used to estimate each story card on how long it would take to implement this card and based on these estimations, customers and programmers decide together about the prioritization of each card and agree about the contents of the first release. Then, the release plan/schedule is finally set up which highlights the features that will be implemented in each release. Then, the user requirements data

and the required development tools will be kept in the project repository. At the end of this stage; the customer's requirements will be used as inputs in the development phase.

4.3.2.2 Phase Two: Development

This phase has the same activities of iteration to release phase in the XP method, where it includes several iterations before the first release. The schedule set in the first release is divided into a number of iterations, where these iterations create one or more functions of the system in each one of them. The design as well as the coding is done, but before any line of code is written, firstly a unit test to test these lines has to be developed by the programmers. In the first iteration, the system with the architecture of the whole system is created. Functional tests are conducted at the end of each iteration by customer. Finally, as soon as the developed features are tested by the developers (probably by automated unit tests), these are given to the customer. After the last iteration, the system is ready to deliver the first release. Then, it goes to the next phase.

4.3.2.3 Phase Three: Product Delivery and Product & Process Efficiency

This phase consists of the same activities of productionizing phase in XP method and other additions that are needed to support the quality assurance of process and product and to achieve the organizational process performance. However, this phase consists of extra testing and performance checks, where the customer performs functional tests and validates if the product works as intended. If new requirements are elicited, they are either included directly or a new story card is created which will be considered in the following release exploration. Furthermore, new changes may be found and they might

still be included in the current release, and the postponed ideas and suggestions are documented for the later implementation.

In addition, there is the need to implement several metrics that could be appropriate for objectively verifying the products and the process such as: (1) release plan adherence; (2) percentage of test cases that are running successfully (number of successful test cases/ numbers of total test cases); (3) percentage of acceptance tests that are running successfully (number of successful acceptance tests/ number of total acceptance tests); (4) length of pair programming sessions; and (5) project velocity (actual time of the implemented user stories of all iteration / estimated time of all user stories of all iterations).

Furthermore, it is important to convey the metrics through defined channels to the affected parties and senior management. The metrics are most conveniently posted to the team using a white-board or an exposed wall in a central location. When metrics out of the ordinary occur it is important for the coach to communicate these findings, either directly to the affected party or during the daily stand-up meeting, and these issues will be resolved. If no satisfactory solution is found, the issue is brought to the customer or project manager, depending on the nature of the issue. If no solution can be found at this level, senior management will be presented with the issue. At the end of this phase, there is a need to keep the metrics results in the project repository to help with the measurement of the same user requirements for incoming projects.

4.3.2.4 Phase Four: System and Process Evolution

This phase consists of the same activities of maintenance and death phase of XP method with several additions. Therefore, after the first release is delivered and taken into use, the XP project has to keep the system running whilst implementing new features. Then, these features will commence from the exploration process in the first phase of the proposed Extended-XP method. This requires an effort for the customer support tasks too, which may decelerate the implementation pace of the new features. The customer is supported by (probably new) team members whose task is to ensure that certain customer requests, i.e. improvement suggestions are considered. This phase may require incorporating new people into the project team and changing the team structure. Based on this, the system will undergo the final release, or the system will be divided for some reasons such as when the costumers does not have new stories to implement, or when the system can not satisfy the customers' needs, as well as when the system is too expensive for modification.

At end of this phase, there is the need to review the organizational process (improve the development processes) to understand the current strengths and weaknesses of the organization's processes. Besides, this review can help to know the required development activities that are important for improving the overall process capability and can be used to satisfy the customer's needs within the time constraints, while maintaining high quality products. This can be done by having the project team writing proposals during the project, and suggesting the required improvement activities. Based on these proposals, SEPG members discuss and analyze these proposals to determine the

required modification on the software development processes. Furthermore, SEPG members are responsible for meeting the project team to discuss the best practices for implementing the proposed Extended-XP method by using the specific practices of CMMI-Dev1.2 as mean items in this discussion. Accordingly, they can identify best practices of implementing the proposed Extended-XP method and keeping these best practices in the project repository to be taken into account for the incoming projects.

4.4 Establishing the Proposed Software Development Process Improvement Framework

The main aim of this research is to construct a suitable software development process improvement framework for SSDFs. Sections 4.4.1 to 4.4.3 will explain the foundation of the proposed framework, the stages of the framework, and the roles of the framework members.

4.4.1 Framework Foundation

The SPI framework consists of four generic elements, which are software process, assessment, capability determination, and improvement strategy as discussed in Section 2.2. This study aims to construct a suitable software development process improvement framework for SSDFs. Accordingly, the generic elements of the SPI framework must be used as a baseline to develop the proposed framework. Thus, there is need to rearrange these elements to be suitable for software development and improvement issues by integrating the CMMI-Dev1.2 model and the proposed Extended-XP method as a generic elements in the proposed framework.

Figure 4.2 shows the foundation elements of the proposed framework. The generic elements of SPI framework were used as a baseline structure to develop the desired framework. Nevertheless, several changes were done to the contents of the generic elements of the SPI framework because the proposed software development process improvement framework focuses on development and improvement issues. In this foundation, the proposed Extended-XP was used as a software development method instead of the improvement strategy, while the CMMI-Dev1.2 model was used as an assessment model in the proposed framework.

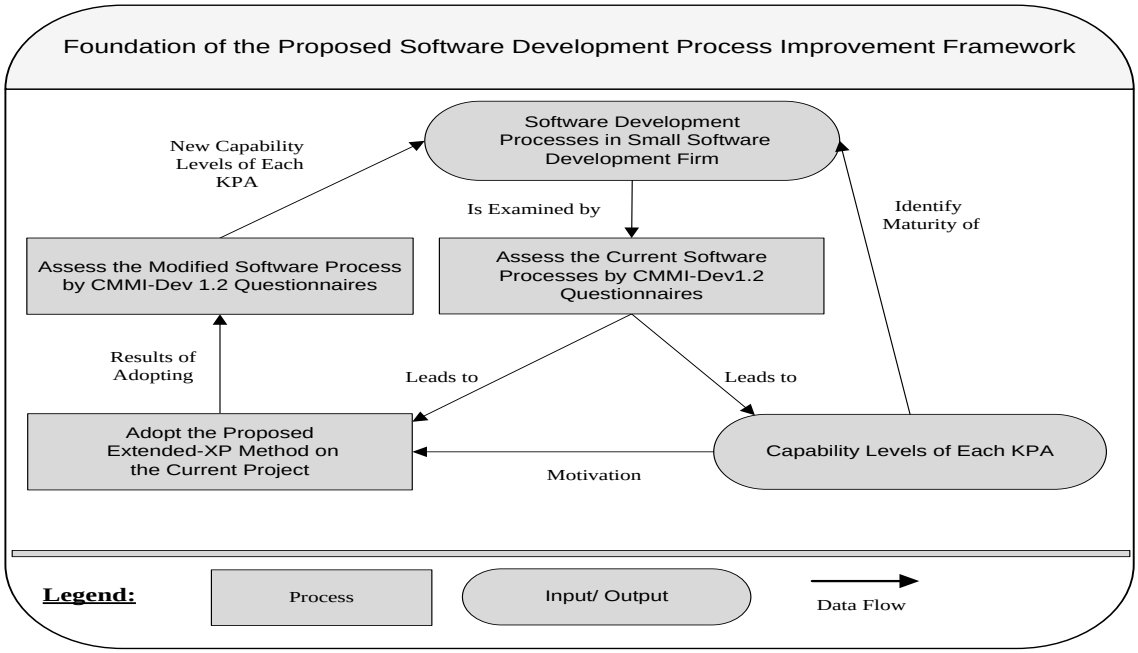

Figure 4.2: Foundation of the Proposed Software Development Process Improvement Framework

4.4.2 The Proposed Software Development Process Improvement Framework

The proposed software development process improvement framework consists of three stages which aim to improve the software development processes, which are: using the CMMI-Dev1.2 KPAs questionnaires to assess the current software development

processes, adopting the proposed Extended-XP method to improve these possesses, and assessing the modified software development process. Figure 4.3 shows the stages of the proposed software development process improvement framework.

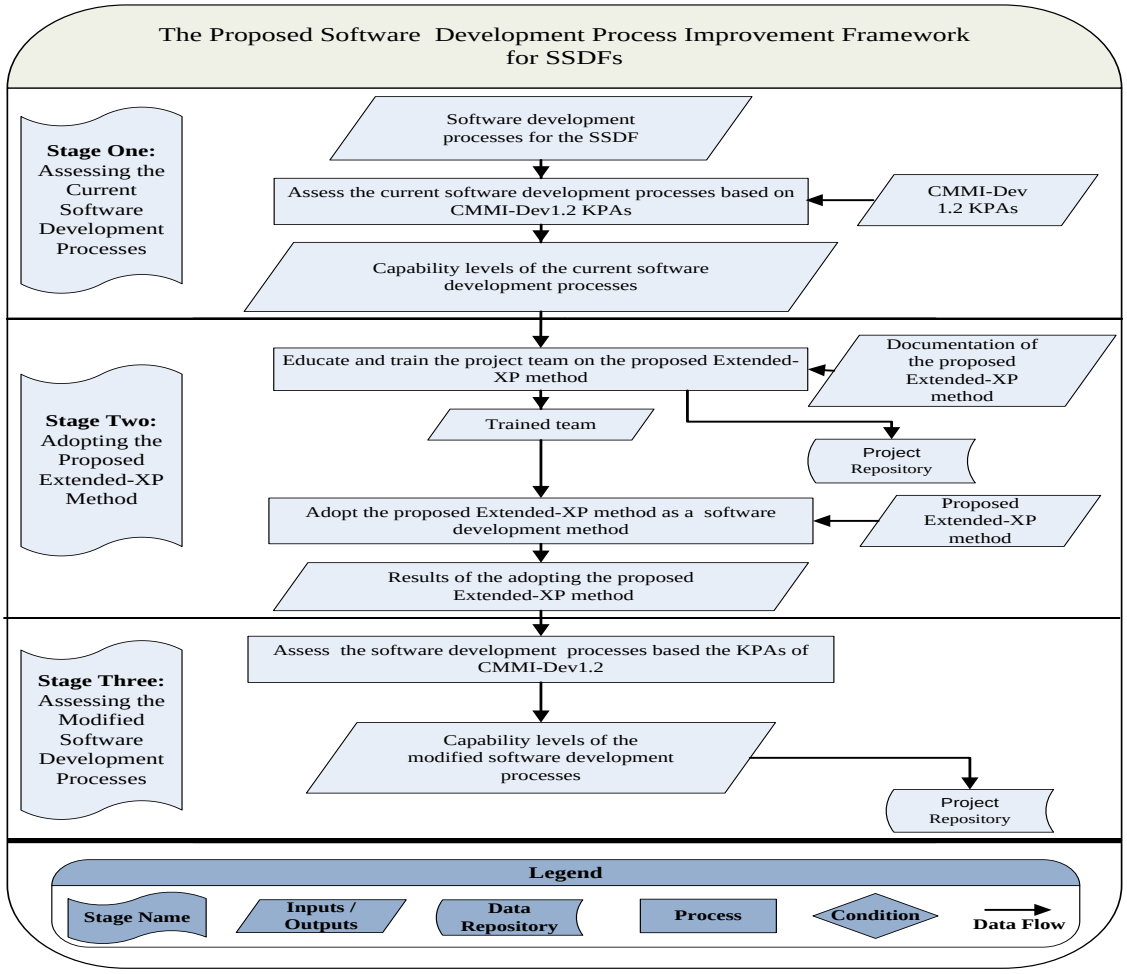

Figure 4.3: The Proposed Software Development Process Improvement Framework for SSDFs

The proposed software development process improvement framework (shown in Figure 4.3) consists of three stages as follows:

- **Stage One: Assessing the Current Software Development Processes**

Prior to starting the assessing of the current software development processes, it is important to create a suitable simple project repository by SEPG members to keep the important date during the implementation of this framework, and during proposed Extended-XP phases.

As discussed in Section 2.3.5, Self-Assessment is suitable for SSDFs, because the cost is lower and mini-assessments are effective for SPI in these firms. Therefore, the SEPG members start to assess the current software development processes in the SSDFs by using the CMMI-Dev1.2 KPAs as main items to identify the capability levels of each area for these processes (refer to Appendix F, questionnaire of assessment the current software development processes). In this aspect, three scales can be used to identify the capability levels of the current development processes, which are:

- Largely Supported: the current software development processes largely support the specific goals of the KPA.
- Partially Supported: the current software development processes partially supports the specific goals of the KPA.
- Not-Supported: the current software development processes do not support the specific goals of the KPA.

At this stage, the firms will know the weaknesses of the current software development processes. Thus, these weaknesses will give the firms motivation to improve their processes based on the proposed Extended-XP method, while the next stage illustrates the required processes of adopting this method by the firms.

- **Stage Two: Adopting the Proposed Extended-XP Method**

  To adopt the proposed Extended-XP method by the firms, there are two processes that must be followed, which are:

  - **Educate and Train the Project Team on the Proposed Extended- XP Method**

    In order to implement the proposed Extended-XP method in the right way; all of the involved team members in software development processes must have a good knowledge of their roles, and they must be trained. The best way to learn the proposed Extended-XP method is by educating and training courses. Furthermore, there is need to support the team with the required XP books and with the documentation of the proposed Extended-XP method during the training and the development lifecycle.

    In this case, SEPG members are responsible for carrying out the training process before starting the implementation of the proposed Extended-XP method by the firms, whereas these members are responsible for: establishing plans for training for developers, estimating the time required for the training, determining if there is need for outsourcing the professional team in training process, training the developers on how they can implement the activities of the proposed Extended-XP method, training the project team on writing the improvement suggestion during the project development, assessing the project teams' efficiency to know if they are ready to implement the proposed Extended-XP method or there is need for

more training, and recording the training efficiencies in the project repository.

As a result of the educating and training process and especially the results of assessing the team's efficiency; it will be known if there are needs for more training or not. Accordingly, the teams will be ready to adopt the proposed Extended-XP method in the right way.

- **Adopt the Proposed Extended-XP Method**

  Based on the descriptions of the proposed Extended-XP method mentioned earlier in Section 4.3.2.1 to 4.3.2.4 and based on the trained team from the previous step, the project team is ready to adopt this method in developing the software project. At the end of this phase, the results of the implementing of the proposed Extended-XP method will be known and it will be used in Stage Three.

- **Stage Three: Assessing the Modified Software Development Processes**

  Referring to the results of adopting the proposed Extended-XP method, SEPG members are responsible for discussing these results by meeting the project team, whereas CMMI-Dev1.2 KPAs will be used as the mean to identify the new capability levels of the modified software development processes.

4.4.3 Roles of the Proposed Software Development Process Improvement Framework

The proposed Extended-XP method is a generic element in the proposed framework, whereas this method is used as a software development method to improve the current software development processes in the firm in the Stage Two of the proposed framework. Moreover, as mentioned earlier in Section 4.4.2; there are several activities that need to be applied before and after the adopting of the Extended-XP method during the stages of the proposed framework. Therefore, it can be concluded that there are two groups of roles in the proposed framework as follows:

- **Framework Roles (Before and after adopting the proposed Extended-XP method):**

  Referring to the stages of the proposed framework; the SEPG members are responsible for several roles before and after the adoption of the proposed Extended-XP method, therefore they are responsible for:

  - In Stage One of the proposed framework, SEPG members are responsible for creating the suitable project repository to keep the important data during the implementation of the proposed framework (including the proposed Extended-XP method), and assessing the current software development processes to determine the capability level of these processes as a self-assessment.
  - In Stage Two of the proposed framework, SEPG members are responsible for arranging the required organizational training before starting to adopt the proposed Extended-XP method, where they are

responsible for: establishing plans for training the developers; estimating the time required for training; determining if there is need for outsourcing the team in training process; training the developers on how they can implement the activities of the proposed Extended-XP, training the project on writing the improvement suggestion during the project development; and recording the training results and assessing the training efficiencies.

- In Stage Three of the proposed framework, SEPG members are responsible for discussing the results of implementing the proposed Extended-XP method to identify the new capability levels of the firm.

- **Extended-XP Method Roles (During the proposed Extended-XP lifecycle):**

  The Extended-XP method has the same roles of XP method that are mentioned in Section 2.5.1.3 such as programmer, customer, tester, coach, tracker, consultant, and big boss. Nevertheless, there are several practices that have been added to the coach, tracker and programmer roles. There are also new roles that have been added to the roles of the proposed Extended-XP method, which are SEPG members. Next points discuss the additional practices that are added to the roles of coach, tracker, and programmer, and also the practices of the SEPG role during the proposed Extended-XP method:

  - **SEPG Members**

    SEPG members are responsible for the improvement strategy at the last phase of proposed Extended-XP method by collecting and analyzing the improvement proposals to determine the required modification. They are

also responsible for extracting the strong best practices and writing lessons learnt from the development processes and keeping these practices and lessons in the project repository to help other projects in the same firm.

- **Coach and Tracker**

  Coach and tracker together are responsible for implementing the required metrics to achieve the objective of process and product quality assurance, and the process performance at Phase Three of the proposed Extended-XP method as follows:

  - Calculating the difference between estimates and actual time spent on user stories or tasks.
  - Calculating the velocities of the projects and the length of pair programming sessions and keeping it into the project repository.
  - Calculating the number of failed acceptance tests, and number of severity defects after release.

- **Programmers and SEPG Members**

  Programmers and SEPG members together are responsible for implementing the supplying process at the first phase of Extended-XP method, where programmers are responsible for extracting the required unavailable development tools or services; while SEPG members are responsible for executing the supplying process with the external suppliers.

4.5 Conclusion

This chapter has described in detail the steps of developing the proposed software development process improvement framework for SSDFs. The development of the proposed framework was started by aligning XP practices to the KPAs of CMMI-Dev1.2. Based on the results of this alignment, EBA was adapted to extend XP method. The chapter illustrated in detail the integration of the Extended-XP method and CMMI-Dev1.2 that has been performed based on the generic elements of SPI framework to develop the proposed framework.

In this chapter, the phases of the Extended-XP method are discussed, which are: stages of the proposed framework are explained. These stages are: requirement management phase, development phase, product delivery and product & process efficiency phase, system and process evolution phase. In addition, the stages of the proposed framework are presented, which are: assessing the current software development processes, adopting the proposed Extended-XP method, and assessing the modified software development processes. Chapter 5 will discuss the verification process of the proposed framework using focus group method.

CHAPTER FIVE
VERIFYING THE PROPOSED SOFTWARE DEVELOPMENT PROCESS IMPROVEMENT FRAMEWORK

This chapter presents the verification process of the proposed framework. It presents the three rounds which were conducted to verify the proposed framework through focus group method coupled with Delphi technique. At the end of this chapter, the modified software development process improvement framework and the modified Extended-XP method are presented.

5.1 Introduction

As highlighted in Section 4.3.1, previous literatures were used to cover the missing KPAs of CMMI-Dev1.2 by adding new development, management, and improvement additions to extend XP method. Furthermore, Section 4.4.2 also illustrated the stages of the proposed framework, which was developed based on CMMI-Dev1.2, the proposed Extended-XP method, and the generic elements of SPI framework. Therefore, to ensure that the proposed framework is compatible with CMMI-Dev1.2 KPAs and suitable for SSDFs, there is need to: (1) verify the compatibility of the proposed framework to the specific goals of CMMI-Dev1.2 KPAs; (2) verify the commitment of the proposed Extended-XP method to XP values; (3) verify the suitability of the proposed framework and proposed Extended-XP roles for their related practices and for SSDFs; and (4) verify the suitability of the proposed framework and the proposed Extended-XP structures for the software development and improvement issues in the SSDFs. In this respect, focus group method coupled with Delphi technique was used to verify the

proposed framework. Sections 5.2 to 5.5 present focus group participants, verification questions, verification schedule, and the results of verification rounds.

5.2 Focus Group Participants

Smaller focus groups are becoming increasingly popular because smaller groups are easier to recruit and host, and they are more comfortable for participants. The ideal size of a focus group for most studies is six to ten participants (Krueger & Casey, 2000). This is especially true if the questions are meant to gain understanding of peoples' experiences and if the researcher wants more in-depth insights. These aims are usually best accomplished with a smaller group. According to Billinger (2005), the respondents are supposed to have a connection to the subject and an understanding of the matter. Thus, in this research, the expert researchers of software development and improvement processes fields and expert developers and managers in SSDFs were chosen to verify and amend the proposed framework to be compatible for the KPAs of CMMI-Dev1.2, taking into account the generic characteristics of SSDFs.

Three expert researchers were selected as members of the focus group sessions for this study because they have good knowledge of CMMI-Dev1.2 and XP method, and have published in the area of software development and improvement. Furthermore, it was convenient to meet them in Jordan. One of the participants is from Sudan and is working in Jordan, and the second researcher is from Syria and also working in Jordan. Another member of the focus group is a Jordanian researcher who is working as a lecturer and researcher at a Jordanian university (refer to Appendix G, researchers' profiles). Additionally, based on the Jordanian Ministry of Industry and Trade, it was convenient

to access the addresses of some SSDFs in Jordan. Using telephone calls employees of these firms were asked about their willingness to participate in the verification process of the proposed framework. As a result, three professional developers, two professional managers, and two members of the software engineering process group who are working in different SSDFs in Jordan agreed to participate in verifying the proposed framework through focus group sessions.

There are several reasons to choose Jordan as the country to verify the proposed framework by focus group method, which are:

- The researcher is from Jordan and has good connection of the native language which is Arabic.
- Most of Jordanian software development firms are small and they have the same generic problems with software development and improvement processes (El Sheikh & Tarawneh, 2007). Furthermore, it was easy to find suitable members to verify and validate the desired framework.
- In order to carry out the focus group method, there is need to meet the participants in person as a group. Therefore, it easier to contact the Jordanian members who are related to the research topic for participation.
- The Arabic language (native language) of participants helped in discussing and understanding each other during the sessions of the focus group.
- The three researchers who participated in the focus group sessions are working in Jordan, so it was easy for them to attend the sessions of the focus group.

5.3 Verification Questions

The verification questions had been identified to ensure that the proposed framework is compatible to the specific goals of CMMI-Dev1.2 KPAs and also to ensure that the proposed Extended-XP method still keep the agility values of XP method, because SSDFs need to have a lightweight method in developing their software products. Furthermore, there is a need to ensure that the structures of the proposed framework and the proposed Extended-XP method were suitable for these firms, and also to ensure that the proposed framework and proposed Extended-XP roles were suitable compared to their related practices. Therefore, the questions that were used in verifying the proposed framework consist of the following four parts:

- **Part One: Verifying the compatibility of the proposed framework to the specific goals of CMMI-Dev1.2 KPAs**

  The specific goals of the CMMI-Dev1.2 KPAs were used as the main items in developing the desired framework. It was necessary to make sure that the proposed framework contents were compatible to these specific goals. In this respect, it was important to use a suitable scale to identify the compatibility degree or ratio of the proposed framework contents to the standard questionnaire of CMMI-Dev1.2 KPAs. Vasiljevic and Skoog (2003) in their thesis of the same field of software process improvement used the five scales (1- Strongly Disagree and 5- Strongly Agree). Therefore, this study used a five scales response in this part. Since this part aims to verify the compatibility of the proposed framework to the KPAs of CMMI-Dev1.2, the scale-response is labeled from 1-Strongly Incompatible to 5-Strongly Compatible.

- **Part Two: Verifying the commitment of the proposed Extended-XP to XP Values**

  Based on the characteristics of SSDFs, lightweight software development methods are more suitable for these firms, such as agile methods that are more applicable for SSDFs because they have simple practices and lightweight values in the manner of software development (Fruhling & Vreede, 2006; Altarawneh & Shiekh, 2008; Fernandeas, 2009).

  As mentioned in Section 2.5.2, XP is a lightweight method with four key values (Beck, 2000) which are communication, simplicity, feedback, and courage. In this study, XP method was used as a software development method in developing the proposed framework. In implementing agile methods in the right way, there is need to improve communication, to seek simplicity, to get feedback on how well you are doing, and to always proceed with courage (Saarnak & Gustafsson, 2003). In addition Koch (2003) stresses the importance of keeping agile values if there is need for any extensions in these methods. Therefore, there is a need to keep these values up in the proposed Extended-XP method in order to be used by SSDFs.

  In this respect, this part consists of four Yes/No questions that were used to ensure that the proposed Extended-XP method was still keeping the values of XP method. Nevertheless, to get the feedback from the respondents about the required modification, the choices of answering these questions are the following:

- Yes without modifications: the proposed Extended-XP method is fully committed to the XP value, and there is no need for any modification.
- Yes with modifications: the Extended-XP method is partially committed to the XP value, and there are needs for modifications.
- No: the proposed Extended-XP method is not committed to the XP value.

- **Part Three: Verifying the suitability of the proposed framework and the proposed Extended-XP roles for their practices**

  The XP method is a collection of different practices and roles to achieve these practices that are inherited from some previous methodologies (Nawaz & Malik, 2008). In addition, it is important for each member in the XP team to know his/her role and the specific practices of this role (Beck, 2000). As mentioned in Section 4.4.3, the roles of the Extended-XP method have several additions compared to the roles of the standard XP method especially in the roles of coach, tracker, and programmers. In addition, there is a new role that has been added to the Extended-XP method compared to XP method which is SEPG role. Furthermore, SEPG role were responsible for several activities before and after the adoption of the Extended-XP method during the stages of the proposed framework. In this respect, there is need to ensure that these roles are distributed in a right way and do not conflict with their practices. Based on the expected answers for each question, the suitable scale for these questions were: Yes without modifications; Yes with modifications; or No.

- **Part Four: Verifying the suitability of the proposed framework and the proposed Extended-XP structures for the software development process improvement issues in SSDFs**

  SPI models are used to inform the organizations of what to do in general terms, but do not say how to do it. On the other hand, XP is a set of best practices that contains fairly specific information about how to implement the model for a particular type of environment (Paulk, 2001; Martinsson, 2002). Furthermore, it is often assumed that CMMI compliant processes need to be heavyweight, bureaucratic and slow-moving (Anderson, 2005). Nevertheless, agile practices such as XP have been said to offer a less bureaucratic way of developing quality software focusing on human centered processes (Bos & Vriens, 2004).

  In addition, XP method and SPI model together form a comprehensive framework for structuring the software development organization (Martinsson, 2002). The proposed framework in this study consists of XP method, CMMI-Dev1.2, and the generic elements of SPI framework. Therefore, it is important to ensure that the proposed framework components are integrated in a suitable structure, and achieves the needed development and improvement issues. To answer questions related to this, there are three scales that were used to reflect the feedback from the respondents, which are: Yes without modifications; Yes with modifications; or No.

5.4 Verification Schedule

As mentioned in Section 5.2, the addresses of the focus group members were known. Then, the proposed framework and the required data for verifying were posted as a hard copy to those members to give them sufficient time (two weeks before the first session) to read and understand the contents of the related data which were needed to verify the proposed framework.

Krueger and Casey (2000) and Billinger (2005) argued that the use of focus group plan helps the researcher to remember everything important during focus group sessions Accordingly, Table 5.1 presents the structured focus group plan (Delphi Rounds) that was followed in this research:

Table 5.1: Structured Focus Group Plan of Verification Process (Delphi Rounds)

Rounds	Session Date& Time	Activities
Round One	**Session One** 10- Nov -2010 9 am -11 am	• Present myself. • Thanking the focus group members for the participation. • Present the research problem. • Present the purpose of the research.
	Session Two 10- Nov -2010 12 pm- 1.5 pm	• Explaining the verification questions. • Answering the verification question individually. • Explaining if there is any inquiry about the verification questions.
	Session Three 10- Nov -2010 2 pm- 5 pm	• Discussing the answers and suggestions of each focus group member by all the members.
Round Two	12- Nov- 2010 To 15- Jan- 2011	• Modifying the proposed framework as suitable suggestions of focus group members.
Round Three	**Session One** 22- Jan – 2011 1 pm- 4 pm	• Viewing the modified framework to the members. • Asking if there is need for more new modifications.

5.5 Results of Verification Rounds

Three rounds were completed to verify the proposed framework. Sections 5.5.1 to 5.5.3 discuss the results of each round and the required modifications of the proposed framework.

5.5.1 Results of Round One

In the first session of this round, the researcher started by presenting a brief Curriculum Vitae (C.V) of himself and thanking the members for their acceptance and participation in the verification process. Then, the verification process continued with the presentation of the research problem and the main aim of this research. In the second session, the researcher explained the verification questions that are needed to verify the proposed framework. Focus group members started to answer the verification questions individually. During this session, there were several inquiries about the verification questions that were presented by the researcher. As a result of the answers provided at this session, the third session started by discussing the answers and suggestions of the focus group members. Then, the results of the third session were documented by the researcher to identify the required modifications that were needed to be done to the proposed framework. Sections 5.5.1.1 to 5.5.1.4 present the answers of the four parts of the verification questions and the related suggestions for each part.

5.5.1.1 Answers and Suggestions of Part One Questions

This part consists of 22 questions aimed to verify the compatibility of the proposed framework to the specific goals of CMMI-Dev1.2 KPAs. The following points present the answers and suggestions for the questions from this part of the process.

- **Focus Group Answers of Part One**

  The focus group members were asked to rate compatibility levels of the proposed framework to the specific goals of CMMI-Dev1.2 KPAs. The questions of this part consist of a scaled-response from 1 to 5 (1: strongly incompatible and 5: strongly compatible). Table 6.2 presents the frequencies, percentages, standard deviation (S.D), and the mean values (M.V) of each KPAs, where these values were performed using SPSS (statistics descriptive).

Table 5.2: Summary of Focus Group Answers for the First Part Questions

CMMI-Dev 1.2 KPAs	1		2		3		4		5		M.V	S. D
	Freq	%	Freq	%	Freq	%	Freq	%	Freq	%		
Requirement Management	0	0	0	0	0	0	3	30	7	70	4.70	.48
Project Planning	0	0	0	0	0	0	4	40	6	60	4.60	.51
Project Monitoring and Control	0	0	0	0	0	0	5	50	5	50	4.50	.52
Supplier Agreement Management	0	0	0	0	3	30	4	40	3	30	4.00	.81
Measurement and Analysis	0	0	0	0	3	30	2	20	5	50	4.20	.91
Process and Product Quality assurance	0	0	0	0	3	30	3	30	4	40	4.10	.87
Configuration Management	0	0	0	0	0	0	5	50	5	50	4.50	.52
Requirements Development	0	0	0	0	4	40	6	60	0	0	3.60	.51
Technical Solution	0	0	0	0	0	0	5	50	5	50	4.50	.52
Product Integration	0	0	0	0	3	30	4	40	3	30	4.00	.81
Verification	0	0	0	0	0	0	4	40	6	60	4.60	.51
Validation	0	0	0	0	0	0	3	30	7	70	4.70	.48
Organizational Process Focus	0	0	0	0	3	30	4	40	3	30	4.00	.81
Organizational Process Definition +IPPD	0	0	0	0	4	40	4	40	2	20	3.80	.78
Organizational Training	0	0	0	0	0	0	5	50	5	50	4.50	.52
Integrated Project Management +IPPD	0	0	0	0	4	40	4	40	2	20	3.80	.78
Risk Management	0	0	0	0	4	40	6	60	0	0	3.60	.51
Decision Analysis and Resolution	0	0	0	0	3	30	5	50	2	20	3.90	.73
Organizational Process Performance	0	0	0	0	3	30	7	70	0	0	3.70	.48
Quantitative Project Management	0	0	0	0	4	40	4	40	2	20	3.80	.78

Organizational Innovation and Deployment	0	0	0	0	4	40	6	60	0	0	3.60	.51
Causal Analysis and Resolution	0	0	0	0	3	30	7	70	0	0	3.70	.48

As was mentioned previously, the first part of the verification questions consisted of a scaled-response from 1 to 5. In Five-Point Scales, the interval width is calculated by (n-1)/n formula, where "n" is the number of scales (Birisci et al., 2009; Bidad & Campiseno, 2010). Based on this, the interval width of this part = (5-1) / (5) = 0.8. Table 5.3 shows the definitions of the interval scales and explains the level of compatibility for each interval scale.

Table 5.3: Interval Scale Definition of the Compatibility

Mean Interval	**Degree of Compatibility**
From 1 To 1.80	Strongly Incompatible
From 1.81 To 2.60	Incompatible
From 2.61 To 3.40	Average
From 3.41 To 4.20	Compatible
From 4.21 To 5	Strongly Compatible

As shown in Table 5.3, mean values were used to identify the compatibility degree of each KPA in part one questions, whereas the mean values between 1 to 1.80 are strongly incompatible, between 1.81 to 2.60 are incompatible, between 2.61 to 3.40 are average, between 3.41 to 4.20 are compatible, and between 4.21 to 5 are strongly compatible. Table 5.4 presents the compatibility degree for each process area.

Table 5.4: The Compatibility Degree for Part One Questions

CMMI-Dev1.2 Level 2 KPAs	Mean Value	Levels of Compatibility
Requirement Management	4.70	Strongly Compatible
Project Planning	4.60	Strongly Compatible
Project Monitoring and Control	4.50	Strongly Compatible
Configuration Management	4.50	Strongly Compatible
Supplier agreement management	4.00	Compatible
Measurement and analysis	4.20	Compatible
Process and product quality assurance	4.10	Compatible
CMMI-Dev1.2 Level 3 KPAs		
Technical Solution	4.50	Strongly Compatible
Verification	4.60	Strongly Compatible
Validation	4.70	Strongly Compatible
Organizational Training	4.50	Strongly Compatible
Requirements Development	3.60	Compatible
Product integration	4.00	Compatible
Organizational process focus	4.00	Compatible
Organizational Process Definition +IPPD	3.80	Compatible
Integrated Project Management +IPPD	3.80	Compatible
Risk management	3.60	Compatible
Decision Analysis and Resolution	3.90	Compatible
CMMI-Dev1.2 Level 4 KPAs		
Organizational Process Performance	3.70	Compatible
Quantitative Project Management	3.80	Compatible
CMMI-Dev1.2 Level 5 KPAs		
Organizational Innovation and Deployment	3.60	Compatible

Causal Analysis and Resolution	3.70	Compatible

Looking at Table 5.4, it can be concluded that the compatibility degree of the proposed framework to the specific goals of CMMI-Dev1.2 KPAs are as follows:

- CMMI-Dev1.2 Level 2: based on the results of this level, the proposed framework is strongly compatible to the specific goals of four KPAs of this level. These areas are: requirement management, project planning, project monitoring and control, and configuration management, while the proposed framework is compatible to the specific goals of the remaining three KPAs: supplier agreement management, measurement and analysis, and process and product quality assurance.
- CMMI-Dev1.2 Level 3: based on the results of this level, the proposed framework is strongly compatible to the specific goals of four KPAs of this level. These areas are: technical solution, verification, validation, and organizational training, while the proposed framework is compatible to the specific goals of the remaining seven KPAs: requirements development, product integration, organizational process focus, organizational process definition +IPPD, integrated project management +IPPD, risk management, and decision analysis and resolution.
- CMMI-Dev1.2 Level 4: based on the results of this level, the proposed framework is compatible to the specific goals of the two KPAs of this level. These areas are: organizational process performance and quantitative project management.

- CMMI-Dev1.2 Level 5: based on the results of this level, the proposed framework is compatible to the specific goals of the two KPAs of this level. These areas are: causal analysis and resolution and organizational innovation and deployment.

- **Focus Group Suggestions of Part One**

  With regards to the suggestions of the part one questions, the focus group members discussed their answers and suggestions and agreed with each other on the following suggestions:

  "*The proposed framework is compatible to the KPAs of CMMI-Dev1.2. Nevertheless, the organizational innovation and deployment KPA is not suitable to be implemented by SSDFs. Then, there is need to remove the related activities of this KPA that were added to the proposed framework".*

5.5.1.2 Answers and Suggestions of Part Two Questions

This part consists of four questions that aimed to verify the commitment of the proposed Extended-XP method to XP Values. The next points present the answers and suggestions for questions of this part:

- **Focus Group Answers of Part Two**

  The focus group members were asked to answer the questions in this part, where three choices were used to answers these questions which were: "yes without modification", "yes with modifications", and "no". In the first question about the commitment of the proposed Extended-XP to the simplicity value of XP method,

three members answered “yes without modifications”, while the remaining seven members chose “yes with modifications”. As for the remaining three questions of the commitment of the proposed Extended-XP to the courage, feedback, and communication values, all the members answered “yes without modifications”.

- **Focus Group Suggestions of Part Two**

  With regards to the suggestions provided in the part two questions, focus group members discussed their answers and suggestions and agreed with each other on the following suggestion:

  “*There is need to remove the related activities of the organizational innovation and deployment KPA from the proposed Extended-XP method, because the related activities of this process area conflict with the simplicity of XP method. In addition, it will be useful to use a free tool as a project repository such as Microsoft Office instead of developing a new repository, because Microsoft Office offers several options that will be suitable for data storing issues and it is easy to use by all the firm members”.*

5.5.1.3 Answers and Suggestions of Part Three Questions

This part consists of two questions to verify the suitability of the proposed framework and the proposed Extended-XP roles for their related practices. The following points present the answers and suggestions of this part’s questions.

- **Focus Group Answers of Part Three**

The focus group members were asked to answer the following questions for this part, with three answer choices for each question, which were: “yes without modifications”, “yes with modifications”, and “no”. For the answers of the first question about suitability of the distribution of the proposed framework and the Extended-XP roles compared to their practices, two members answered “yes without modifications”, while the remaining eight members answered “yes with modifications” in their answers. In the results of the second question about the suitability of the proposed framework and the Extended-XP roles for SSDFs, three members answered “yes without modifications”, while the remaining seven members answered “yes with modifications” in their answers.

- **Focus Group Suggestions of Part Three**

  With regards to the suggestions of the third part questions, focus group members discussed their answers and suggestions and agreed with each other on the following suggestions:

  “Firstly: there is no need to divide the roles of framework to framework roles and Extended-XP roles, because all the roles are for the framework as a whole and the Extended-XP method is included in the framework.”

  “Secondly: SEPG members have roles inside and outside the Extended-XP method. Therefore, there is a need to specify the roles of those members into two groups which are:

- *Framework-SEPG role: The members of this role are responsible for the practices of SEPG role that are used before and after the adoption of Extended-XP method.*
- *Extended-XP-SEPG role: The members of this role are responsible for the practices of SEPG role that are used during the Extended-XP method."*

5.5.1.4 Answers and Suggestions of Part Four Questions

This part consists of one question which aimed to verify the suitability of the proposed framework and the proposed Extended-XP structures for software development process improvement issues in SSDFs. The following points present the answers and suggestions of this part.

- **Focus Group Answers of Part Four**

  The focus group members were asked to answer the questions from this part of the questionnaire, and were given the following three choices: "yes without modifications", "yes with modifications", and "no". As a result of their answers, three members answered "yes without modifications", while the remaining seven members answered "yes with modifications" as their answers.

- **Focus Group Suggestions of Part Four**

  With regards to the suggestions of the fourth part of the questionnaire, focus group members discussed their answers and suggestions and agreed with each other on the following suggestions:

"Firstly: in the first stage of the proposed framework, there is need to add new process to be responsible for modifying and rearranging the current software development processes. However, this process aims to modify the roles of the current development processes to be suitable with the roles of the framework by distributing the new roles of the framework to the project team to commensurate with their experiences. In this case, framework-SEPG members are responsible for applying this process."

"Secondly: there is need to remove the 'improve development processes from the fourth phase of the proposed Extended-XP (System and Process Evolution phase). Thus, there is need to rename the fourth phase of Extended-XP method, because there is no improvement process at this phase."

"Thirdly: in the third stage of the proposed framework, there is no need to assess the CMMI levels for the firm. Nevertheless, there is need for identifying the best practices process in this stage by discussing the implementing of the framework with the project team for the current project based on the specific practices of CMMI-Dev1.2. This process is related to the framework-SEPG role; therefore framework-SEPG members are suitable to apply this process. Moreover, there is need to rename the third stage of the framework with an appropriate name for the included activities.

Based on the results of this round, the required modifications for the proposed framework and the proposed Extended-XP method were incorporated and used as inputs in the second round.

5.5.2 Results of Round Two

This round aimed to modify the proposed framework and the proposed Extended-XP method based on the required modifications that were made known from the first round. Table 5.5 summarizes the required modifications that were needed to modify the proposed framework and the proposed Extended-XP method. Thus, based on these modifications, the proposed framework was modified as shown in Section 5.6, and the modified Extended-XP method is presented in Section 5.7. Therefore, the modified framework and the modified Extended-XP method were used as inputs in the third round.

Table 5.5: Summary of the Required Modifications on the Proposed Framework and Proposed Extended-XP Method by Focus Group Members

Parts	**Required Modifications**

Part One & Part Two	• Removing the related activities of organizational innovation and deployment KPA from the proposed framework. These activities are:
	  • Training the project on writing the improvement suggestion during the project development (this activity belongs to the educating and training process in stage two of the propose framework).
	  • Writing the improvement suggestions during the implementation of the proposed Extended-XP method.
	  • Discussing the improvement suggestions that are written by the team during the project development (this activity belongs to analyzing the results of implementing the proposed Extended-XP method in stage three of the proposed framework).
	• Using Microsoft Office as a simple project repository.

Part Three & Part Four	• Using framework roles instead of the two groups of roles (framework roles and Extended-XP roles). • Using framework-SEPG role instead of roles of SEPG members that are used before and after the adopting of Extended-XP method. Moreover, using Extended-XP-SEPG role instead of the roles of SEPG members that are used during the Extended-XP method. • Adding new process in the first stage of the proposed framework to be responsible for modifying the current roles to be suitable with the framework roles, where framework-SEPG members are responsible for applying this process. • Remove the 'improve development processes' process from fourth phase of the proposed Extended-XP method. • Remove the "assess the software development processes by CMMI-Dev1.2" process from the third stage of the proposed framework. • Add new process into the third stage of the proposed framework to identify the best practices of implementing the Extended-XP method by discussing the implementing of the framework with the project team based on the specific practices of CMMI-Dev1.2, where framework-SEPG members are responsible for applying this process. • Rename the third stage of the proposed framework, and rename the fourth phase of the proposed Extended-XP method as included activities.

5.5.3 Results of Round Three

In this round, the modified framework (including the modified Extended-XP method) was shown to focus group members to make sure that all approved suggestions have been taken into account in the framework, and also to check if there is need for further modifications. As a result of this round, all members agreed on the framework and they also confirmed that all their approved suggestions had been taken into account in the framework. Therefore, there was no need for further modifications.

5.6 The Modified Software Development Process Improvement Framework

The proposed software development process improvement framework, which was developed in Section 4.4.2 consist of three stages, which are: assessing the current software processes, adopting the proposed Extended-XP method, and assessing the modified software development processes. Further, in reference to the suggestions of the focus group members, several modifications were made to the stages of the proposed framework. Therefore, the modified software development process improvement framework consisted of the following phases: assessing the current software development processes, adopting the extended-XP method, and identifying the best practices of the current project. Figure 5.1 show the generic elements of the modified framework. In addition, Figure 5.2 views all the processes, inputs, and outputs of the three phases in the modified software development process improvement framework.

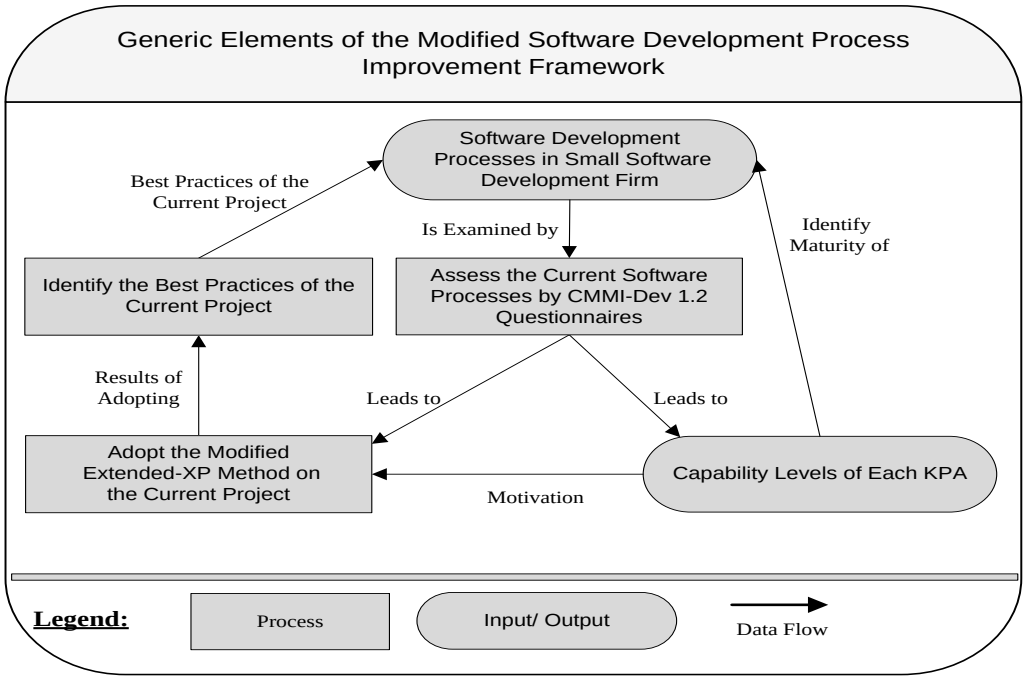

Figure 5.1: Generic Elements of the Modified Software Development Process Improvement Framework

As shown in Figure 5.1, the modified software development process improvement framework consists of five generic elements as follows:

- Software development processes in SSDFs: the current activities used to produce the software products in the firm.
- Assess the current software processes by CMMI-Dev1.2: this element is used to assess the current state of the software process and is done by using CMMI-Dev1.2 questionnaires.
- Capability Determination: this element is used to know the capability level of the software process and motivates an organization to adopt the modified Extended-XP method to improve the current processes.
- Adopt the modified Extended-XP method: this element is used to improve the current processes by adopting the modified Extended-XP method in developing the project of the firm.
- Identify the best practices of the current project: this element is used to identify the best practices of implementing the modified Extended-XP method in the current project by using CMMI-Dev1.2 best practices questionnaire to help the firm for incoming projects.

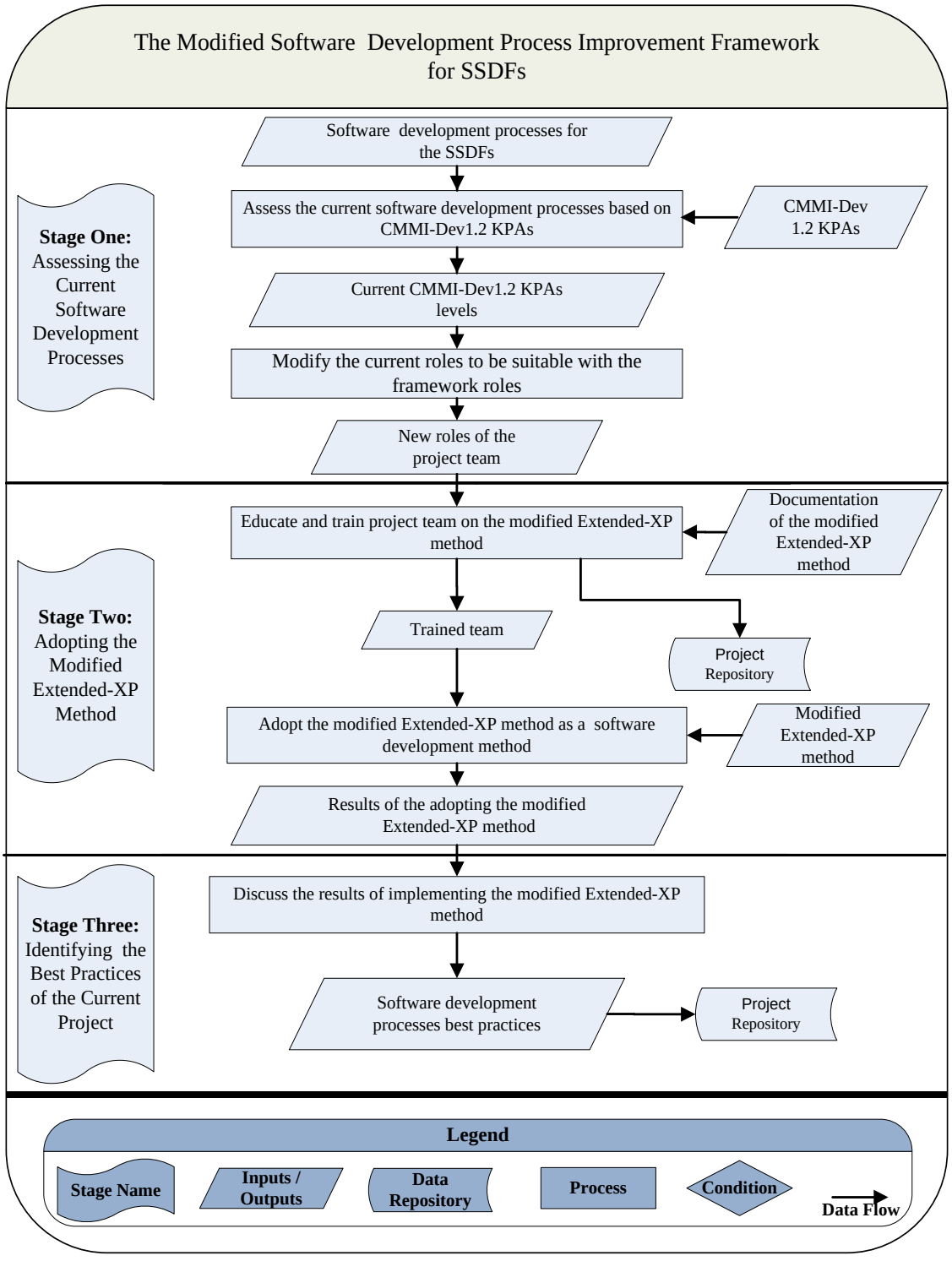

Figure 5.2: The Modified Software Development Process Improvement Framework

As shown in Figure 5.2, the modified software development process improvement framework consists of three stages as follows:

- **Stage One: Assessing the Current Software Development Processes**

  Before starting to implement the framework, it is important to determine the suitable simple project repository by framework-SEPG members to keep on schedule during the implementation of this framework, and during the modified Extended-XP phase. Microsoft Office was suggested as a free tool for data storing issues. As a result, the framework started with the current software development processes in the SSDF. Based on these processes, the framework-SEPG members are responsible for self-assessment process by assessing the current capability levels of these processes by using CMMI-Dev1.2 KPAs. In this aspect, three scales can be used to identify the capability levels of the current development processes which are:

  - Largely Supported: the current software development processes largely support the specific goals of the KPA.
  - Partially Supported: the current software development processes partially support the specific goals of the KPA.
  - Not-Supported: the current software development processes do not support the specific goals of the KPA.

  As results from the current levels indicate, the framework-SEPG members are responsible for modifying and rearranging the current software development processes to be suitable with the required roles of the framework. This can be done by distributing the new roles of the framework to the project team members

to commensurate with their experiences. At the end of this stage, the new roles are made known to each employee in the firm.

- **Stage Two: Adopting the Modified Extended-XP Method**

  This stage has the same processes as stage two in the proposed software development framework which was discussed in Section 4.4.2. There is need to use the modified Extended-XP method instead of the proposed Extended-XP method. In addition, the name of SEPG members in this method is Extended-XP-SEPG members. Section 5.7 discusses the modified Extended-XP method.

- **Stage Three: Identifying the Best Practices of the Current Project**

  In reference to the results of the second stage, the framework-SEPG members are responsible for meeting with the project team to discuss the best practices of implementing the framework by using the specific practices of CMMI-Dev1.2 KPAs as main items in this discussion (refer to Appendix H, best practices questionnaire). In this questionnaire, three choices were used to answers these questions, which are: "Yes" when the practice is well established and consistently performed, "Don't Know" when the respondents are uncertain about how to answer the question, "Does Not Apply" when respondents have the required knowledge about the project or firm and the question asked, but they feel the question does not apply to the project, and "No" when the practice is not well established or is inconsistently performed (this scale has been adopted from the European Software Institute (ESI) software best practice questionnaire (ESI,1997)). Based on the results of this meeting, the best practices for implementing the

framework for the current project could be extracted. Then, the framework-SEPG members are responsible for keeping these best practices in the project repository to be taken into account for incoming projects.

As mentioned in section 4.4.3, the proposed framework roles were divided into two groups which were proposed framework roles and proposed Extended-XP roles. Nevertheless, in the modified software development process improvement framework, there is no need to divide the framework roles into two groups. Therefore, the framework roles name is used for all roles that are used through the framework including the roles of the modified Extended-XP method. The framework roles consist of the following:

- **Programmers, Customer, Tester, Coach, Tracker, Consultant, and Big Boss**

  In this framework, these roles have the same practices as the XP method roles which are mentioned in Section 2.5.2.3. These roles are used during the implementation of the modified Extended-XP method.

- **Coach and Tracker**

  The coach and tracker together are responsible for implementing the required metrics to achieve the objective of process and product quality assurance and the process performance at the third phase of the modified Extended-XP method by the following:

  - Calculating the difference between estimates and actual time spent on user stories or tasks.

- Calculating the velocities of the projects and the length of pair programming sessions and keeping it in the project repository.
- Calculating the number of failed acceptance tests, and number of severity defects after release.

- **Programmers and Extended-XP-SEPG Members**

  Programmers and Extended-XP-SEPG members are responsible for implementation of the supplying process at the first phase of Extended-XP method, where programmers are responsible for extracting the required unavailable development tools or services, and the Extended-XP-SEPG members are responsible for executing the supplying process (in the first phase of the modified Extended-XP method) with the external suppliers.

- **Framework-SEPG Members**
  - Specifying suitable simple project repository in the first stage of the framework in order to keep important data during the implementation of this framework.
  - Assessing the current software development processes in the first stage of the framework as a self-assessment.
  - Modifying the roles of the current software development processes to be suitable with the framework roles in the first stage of the modified framework.
  - Arranging the required organizational training before starting to adopt the modified Extended-XP method in the second stage of the modified framework, where they are responsible for the following:

- Establish planning for training the programmers.
- Estimate the time required for training.
- Determine if there is need for out sourcing professional team in training process.
- Train the project team on how they can implement the activities of the modified Extended-XP.
- Record the training results and assessing the training efficiencies in the project repository.

- Meeting the project team in the third stage of the framework to extract the best practices of the current project and keeping these practices to help incoming projects in the same firm.

5.7 The Modified Extended-XP Method

The proposed Extended-XP life cycle consists of four phases: requirement management phase, development phase, product delivery and product & process efficiency phase, and system and process evolution phase. However, the modified Extended-XP method consists of the following phases: requirement management phase development phase, product delivery and product & process efficiency phase, and maintenance & death phase. The first three phases of the modified Extended-XP method have the same processes of the proposed Extended-XP method that are mentioned in Section 4.3.2.1 to 4.3.2.3. Nevertheless, several modifications were made during phase four on the modified Extended-XP method:

- Name of phase four is maintenance & death phase.

- The process of improving the development processes was removed with their related activities which are: writing improvement proposals during the project, discussing and analyzing these proposals to determine the required modification on the software development processes, and identifying the best practices and lessons learnt.

In this aspect, Figure 5.3 presents the proposed Extended-XP method which was developed in Chapter 4 with the highlighted issues that need to be modified as a result of the suggestions from the focus group members (highlighted and are in thick borders), while Figure 5.4 presents the modified Extended-XP method.

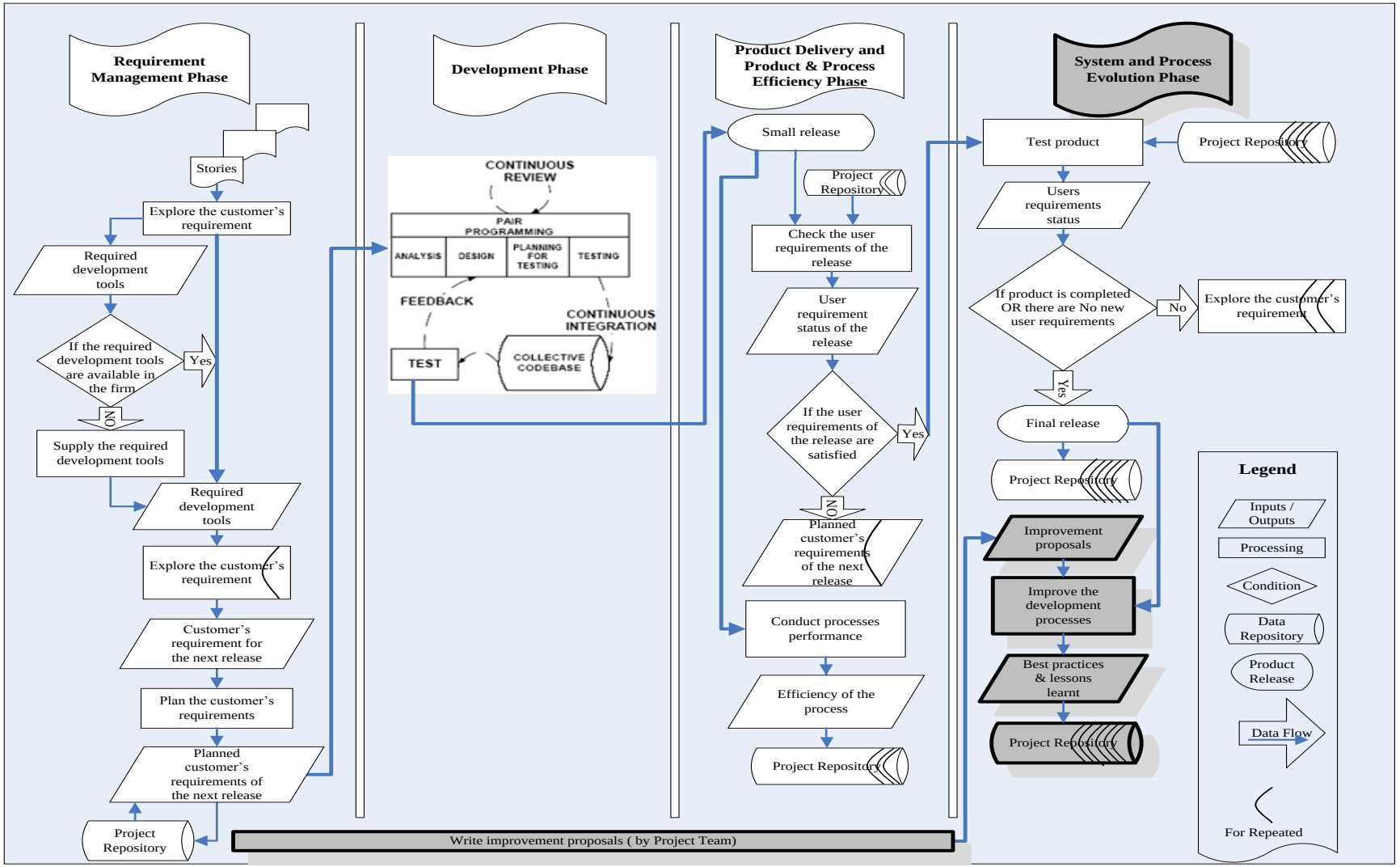

Figure 5.3: Required Modifications on the Proposed Extended-XP Method

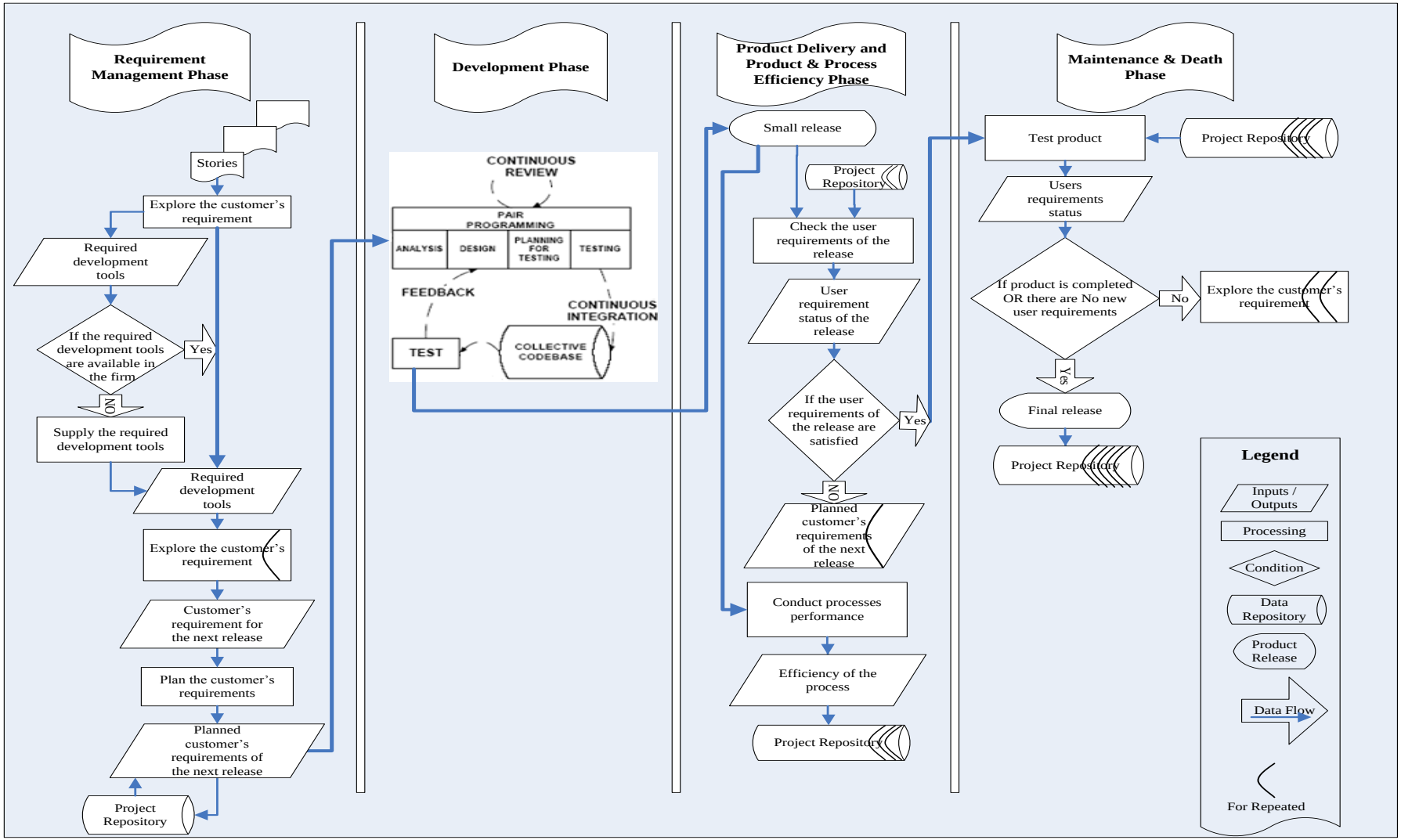

Figure 5.4: The Modified Extended-XP Method

As a result of the verification process, the proposed software development process improvement framework was modified by the suggestions of focus group members. Therefore, to make sure that the framework is suitable for SSDFs, there is need to validate the suitability of this framework by more professional managers and developers who are working in these firms (Al-Allaf, 2008). Chapter 6 discusses the two methods were used to validate the modified framework.

5.8 Conclusion

The proposed framework was developed based on CMMI-Dev1.2 as a SPI model, XP method as a software development method, and the generic elements of SPI framework. In this chapter, focus group method was used as a verification method coupled with Delphi technique to verify the proposed framework.

In this respect, three rounds were used. As a result of the first round, it can be concluded that the proposed framework is strongly compatible to the specific goals of eight KPAs, which are: requirement management, project planning, project monitoring and control, configuration management, technical solution, verification, validation, and organizational training, while the remaining twelve KPAs got the compatible level, which are: supplier agreement management, measurement and analysis, process and product quality assurance, requirements development, product integration, organizational process focus, organizational process definition +IPPD, integrated project management +IPPD, risk management, decision analysis and resolution, organizational process performance, quantitative project management, causal analysis and resolution, and organizational innovation and deployment. In

addition, the focus group members indicated that all the KPAs are suitable for SSDFs; except the organizational innovation and deployment KPA. Accordingly, several changes made to the proposed framework to make it compatible with the suitable KPAs of CMMI-Dev1.2 as showed in Table 5.5.

In the second round, the proposed framework (including the proposed Extended-XP method) was modified as the suggestions of focus group members to be suitable with the SSDFs as discussed in Sections 5.6 and 5.7. Finally, the modified framework was shown to the focus group members in the third round to make sure that the modified framework is suitable for the SSDFs. As a result of the third round, the focus group members confirmed the required modifications were done to the modified framework as discussed in Section 5.5.3. Chapter 6 will discuss about validating the modified software development process improvement framework.

CHAPTER SIX
VALIDATING THE MODIFIED SOFTWARE DEVELOPMENT PROCESS IMPROVEMENT FRAMEWORK

This chapter presents the two approaches used in the validation process to validate the suitability and applicability of the modified software development process improvement framework for the SSDFs. In addition, this chapter presents the evaluation process that was used to evaluate the effectiveness of the modified framework for SSDFs.

6.1 Introduction

The proposed software development process improvement framework was verified to be compatible with the suitable KPAs of CMMI-Dev1.2 by the verification process as discussed in Chapter 5. Therefore, there is a need to validate the suitability of the modified framework for SSDFs to ensure that this modified framework is applicable for these firms. In this aspect, this study has two validation approaches which are: quantitative research method that involved a survey to validate the suitability of this modified framework for SSDFs as discussed in Section 6.2, and a qualitative research method that involved two case studies to validate the applicability of implementing this modified framework in SSDFs as explained in Sections 6.3.1 and 6.3.2. At the end of the second approach of validation, general evaluation criteria were used to evaluate the effectiveness of implementing the modified framework by the two firms as explained in Section 6.4.

6.2 Validating the Suitability of the Modified Framework

A formal validation for the suitability of the modified framework by SSDFs was undertaken using CMMI-Dev1.2 questionnaire as the main items in this validation. The questionnaire format consists of two parts: the first part asks for general demographic information about the respondents, while the second part includes all the specific goals of the suitable CMMI-Dev1.2 KPAs. In addition, the summary of XP practices and the software development, improvement, and management additions that are used in the framework to cover the specific goals of the suitable CMMI-Dev1.2 KPAs areas was attached with these questionnaires to help the respondents in answering the validation questions.

Based on the Jordanian Ministry of Industry and Trade, it was convenient to access the addresses of some SSDFs in Jordan. Nevertheless, it was difficult to find the professional developers and managers at these firms, where most of these firms do not have professional employees. Therefore, these questionnaires were given just to 90 professional developers and managers who are working in these firms. From a total of 90 questionnaires distributed, only 37 questionnaires were returned and seven of them were returned with missing data of more than 30% for each questionnaire. Therefore, only 30 cases were used for the validation process. The problems and response rate are similar to what Al-Allaf (2008) reported.

In addition, the modified framework should be clearly read and understood by the professional developers and managers in these firms to evaluate it according to the characteristics of their firms and the requirements of the specific goals of each

CMMI-Dev1.2 KPAs. Therefore, a hard copy which included the detailed description of the modified framework (included the modified Extended-XP method) and the description of CMMI-Dev1.2 KPAs was attached with these questionnaires. Sections 6.2.1 and 6.2.2 illustrate the results of both parts of the questionnaires.

6.2.1 Part One: Respondents' Profile

This section presents the results of the part one questions which ask about the demographic information of the respondents. This part consists of four questions: current position, current work, size of firm, and software experience. As shown in Table 7.1, the majority of the respondents were members of a SEPG constituting (40%), while the rest were: managers (26.66%), technical members (20%), and project or team leaders (13.33%). Additionally, with regards to the current work activities, the highest ratio was for code and unit testing (26.66%), software design (16.66%), software quality assurance (17%), configuration management and software requirement each (13.33%), and lastly, SPI (6.66%). In term of CMMI training, 90% of the respondents did not receive any CMMI training, while 10% had received this training. With regards to the software experience term, 66.66% of respondents had 6-10 years, where the other two periods (less than five years & 11 years and above) had the same ratio (16.66%). Concerning the firm's size, 46.66% of the respondents were working in firms that had 20-31 employees while, 20% of respondents were working in firms that had 41-50 employees. The firms that had 10 – 20 employees and 31 - 40 employees had the same ratio (16.66%).

Table 6.1: Demographic Information of the Respondents

Items		Frequency	Percentage (%)
Current position	Project or team leader	4	13.33
	Manager	8	26.66
	Technical member	6	20.00
	Software engineering process group	12	40.00
Activities	Software requirement	4	13.33
	Software quality assurance	3	10.00
	Software design	5	16.66
	Configuration management	4	13.33
	Code and unit test	8	26.66
	Software process improvement	2	6.66
	Test and integration	4	13.33
Received any CMMI-related training	Yes	3	10.00
	No	27	90.00
Overall software experience	5 years &less	5	16.66
	6-10 years	20	66.66
	11years &above	5	16.66
How large is your organization	10 - 20 employees	5	16.66
	21 - 30 employees	14	46.66
	31 - 40 employees	5	16.66
	41 - 50 employees	6	20.00

6.2.2 Part Two: Suitability of the Modified Framework for SSDFs

In this part, the respondents were asked to rate the level of the suitability of the modified framework for SSDFs based on XP practices and the software development, improvement, and management additions that were used in the modified framework to cover the specific goals of the suitable CMMI-Dev1.2 KPAs. The questions in this part of the questionnaire consisted of scaled-response items from 1 to 5 such that 1= Strongly Unsuitable and 5= Strongly Suitable. Based on the results of the first part, Table 6.2 shows the frequencies, percentages, Standard Deviation (S.D), and the Mean Value (M.V) of each KPA of the thirty respondents, where these values were acquired using SPSS (statistics descriptive).

Table 6.2: The Suitability of the Modified Framework for SSDFs

CMMI-Dev 1.2	**1**		**2**		**3**		**4**		**5**		**M.V**	**S.D**
KPAs	**Freq**	**%**	**Freq**	**%**	**Freq**	**%**	**Freq**	**%**	**Freq**	**%**		
Requirement Management	0	0	0	0	0	0	10	33.3	20	66.7	4.66	.47
Project Planning	0	0	0	0	3	10.0	9	30.0	18	60.0	4.50	.68
Project Monitoring and Control	0	0	0	0	0	0	13	43.3	17	56.7	4.56	.50
Supplier agreement management	0	0	6	20.0	6	20.0	15	50.0	3	10.0	3.50	.93
Measurement and analysis	0	0	4	13.3	5	16.7	19	63.3	2	6.7	3.63	.80
Process and Product Quality Assurance	0	0	5	16.7	6	20.0	15	50.0	4	13.3	3.60	.93
Configuration Management	0	0	0	0	6	20.0	13	43.3	11	36.7	4.16	.74
Requirements Development	0	0	3	10	5	16.7	16	53.3	6	20.0	3.83	.87
Technical Solution	0	0	0	0	3	10.0	9	30.0	18	60.0	4.50	.68

Product Integration	0	0	0	0	8	26.7	10	33.3	12	40.0	4.13	.81
Verification	0	0	0	0	0	0	14	46.7	16	53.3	4.53	.50
Validation	0	0	0	0	0	0	13	43.3	17	56.7	4.56	.50
Organizational Process Focus	0	0	0	0	5	16.7	22	73.3	3	10.0	3.93	.52
Organizational Process Definition +IPPD	0	0	5	16.7	6	20.0	16	53.3	3	10.0	3.56	.89
Organizational Training	0	0	0	0	5	16.7	22	73.3	3	10.0	3.93	.52
Integrated Project Management +IPPD	0	0	7	23.3	3	10.0	18	60.0	2	6.7	3.50	.93
Risk Management	0	0	4	13.3	8	26.7	13	43.3	5	16.7	3.63	.92
Decision Analysis and Resolution	0	0	5	16.7	6	20.0	15	50.0	4	13.3	3.60	.93
Organizational Process Performance	0	0	6	20.0	4	13.3	18	60.0	2	6.7	3.53	.89
Quantitative Project Management	0	0	5	16.7	4	13.3	17	56.7	4	13.3	3.66	.92
Causal Analysis and Resolution	0	0	4	13.3	5	16.7	19	63.3	2	6.7	3.63	.80

As discussed in Section 5.5.1.1, the interval width is calculated by (n-1)/n formula, where “n” is the number of scales. Based on that, the interval width of this part = (5-1) / (5) = 0.8. Table 6.3 shows the definitions of the interval scales and explains the level of use for each interval scale. Table 6.4 presents the suitability degree for each question.

Table 6.3: Interval Scale Definition of the Suitability

Mean interval presentation	Degree of Suitability
From 1 To 1.80	Strongly Unsuitable
From 1.81 To 2.60	Unsuitable
From 2.61 To 3.40	Average
From 3.41 To 4.20	Suitable
From 4.21 To 5	Strongly Suitable

Table 6.4: The Suitability Degree for Part Two Questions

CMMI-Dev1.2 Level 2 KPAs	Mean. Val	Suitability Levels
Requirement Management	4.66	Strongly Suitable
Project Planning	4.50	Strongly Suitable
Project Monitoring and Control	4.56	Strongly Suitable
Supplier Agreement Management	3.50	Suitable
Measurement and Analysis	3.63	Suitable
Process and Product Quality Assurance	3.60	Suitable
Configuration Management	4.16	Suitable
CMMI-Dev1.2 Level 3 KPAs		
Technical Solution	4.50	Strongly Suitable
Verification	4.53	Strongly Suitable
Validation	4.56	Strongly Suitable
Requirements Development	3.83	Suitable
Product Integration	4.13	Suitable
Organizational Process Focus	3.93	Suitable
Organizational Process Definition +IPPD	3.56	Suitable
Organizational Training	3.93	Suitable
Integrated Project Management +IPPD	3.50	Suitable
Risk Management	3.63	Suitable
Decision Analysis and Resolution	3.60	Suitable
CMMI-Dev1.2 Level 4 KPAs		
Organizational Process Performance	3.53	Suitable
Quantitative Project Management	3.66	Suitable
CMMI-Dev1.2 Level 5 KPAs		
Causal Analysis and Resolution	3.63	Suitable

With regards to the suitability of the framework for SSDFs shown in Table 6.4, the following can be concluded:

- CMMI-Dev1.2 Level Two: at this level, three KPAs received the level of strongly suitable as follows: requirement management (4.66), project planning (4.50), and project monitoring and control (4.56), while the remaining four KPAs received the level of suitability as follows: supplier agreement management (3.50), measurement and analysis (3.63), process and product quality assurance (3.60), and configuration management (4.16).
- CMMI-Dev1.2 Level Three: at this level, just three KPAs received the level of strongly suitable as follows: technical solution (4.50), verification (4.53), and validation (4.56), while the remaining eight KPAs received the level of suitability as follows: requirements development (3.83), product integration (4.13), organizational process focus (3.93), organizational process definition +IPPD (3.56), organizational training (3.93) , integrated project management +IPPD (3.50), risk management (3.63), and decision analysis and resolution (3.60).
- CMMI-Dev1.2 Level Four: the two KPAs of this level received the level of suitability as follows: organizational process performance (3.53), and quantitative project management (3.66).
- CMMI-Dev1.2 Level Five: causal analysis and resolution is the only one at this level where this area obtained 3.63 at the level of suitability.

Based on the results of the suitability of the framework for SSDFs shown in Table 6.4, it can be concluded that the modified software development process

improvement framework is suitable for SSDFs, as all of the related components in the modified framework that aimed to achieve the requirements of the specific goals for the suitable CMMI-Dev1.2 KPAs are strongly suitable or suitable for these firms.

6.3 Validating the Applicability of Implementing the Modified Framework for SSDFs

In order to validate the applicability of implementing the modified framework in SSDFs, two Jordanian SSDFs used this framework to improve their software development processes. They applied the framework in developing their software projects, where the first case study project aimed to develop a computer skills online examination system, and the second case study project aimed to develop a brokerage online system. Sections 6.3.1 and 6.3.2 discuss the results of implementing this framework by the two case studies.

6.3.1 Case Study One: Developing the Computer Skills Online Examination System by "X" Firm

"X" firm was established in 2001 as a Jordanian SSDF. This firm extols itself as the best firm, integrating sophisticated technologies so as to deliver world-class solutions. This firm is committed to providing professional services in an effective, fast, user-friendly and time bound manner. The aim of this firm is to avail software programs that provide services and assist financial institutions to utilize the best technological tools for analyzing and monitoring the changes in any global market on a real-time basis. This "X" firm has 22 employees, working in managing and developing the software products.

This firm has one manager, three project or team leaders, eleven software engineering process group members, and seven technical members. In this firm, the computer skills online examination system has been developed and used as a case study for this research, where this firm had used the framework to improve their software development processes to develop this system.

This desktop application will facilitate conducting computer skills exams to students in university. This application saves time and will allow a number of students to take the exam at the same time and will display the results as soon as the test is complete. There is no need to wait for the results as they are automatically generated by the server. This application is controlled by an administrator who has the privilege to create, modify and delete exams and their contents (questions, answers, and marks). Students will be provided with a specific login id to have their exams and view the results. The following points present the development of this system by "X" firm based on the modified software development process improvement framework of this research.

Before starting the implementation of the framework stages, the researcher met the manager of "X" firm and it was agreed upon that the manager will implement this framework in this firm. Then, two of the SEPG members were asked by the manager to act as framework-SEPG members. One week was given to those members to read and understand the description of the framework. Then, the framework-SEPG members started to implement the modified framework by improving their software development processes to develop the computer skills online examination system.

Sections 6.3.1.1 to 6.3.1.3 describe in detail the stages of improving the current software development processes to develop the computer skills online examination system.

6.3.1.1 Stage One: Assessing the Current Software Development Processes

In this stage, the two framework-SEPG members chose Microsoft Office as a simple project repository to store the project data during the implementation of the framework. Then, they started to assess the current software development processes in the firm based on CMMI-Dev1.2 KPAs. This assessment was done by one meeting between the manager, two team leaders, two technical members, and framework-SEPG members who were working in the firm. Based on this, the framework-SEPG members identified the capability levels of the current software development processes of the firm. Table 6.5 shows the capability levels of the current software development processes of the firm. Three scales have been used to identify the capability levels of the current software development processes, which are:

- Largely Supported: the current software development processes largely support the specific goals of the KPA.
- Partially Supported: the current software development processes partially support the specific goals of the KPA.
- Not-Supported: the current software development processes do not support the specific goals of the KPA.

Table 6.5: Supported Levels of CMMI-Dev1.2 KPAs of the Current Software Development Processes for the First Case Study

CMMI-Dev1.2 KPAs	Largely Supported	Partially Supported	Not Supported
Level Two KPAs			
Requirement Management		X	
Project Planning		X	
Project Monitoring and Control		X	
Measurement and analysis		X	
Process and product quality assurance		X	
Configuration Management		X	
Supplier agreement management			X
Level Three KPAs			
Verification	X		
Validation	X		
Requirements Development		X	
Technical Solution		X	
Product integration		X	
Organizational Training		X	
Integrated Project Management +IPPD		X	
Risk management		X	
Decision Analysis and Resolution		X	
Organizational process focus			X
Organizational Process Definition +IPPD			X
Level Four KPAs			
Organizational Process Performance		X	
Quantitative Project Management			X
Level Five KPAs			
Causal Analysis and Resolution		X	

Based on Table 6.5, the following can be concluded:

- Two KPAs are largely supported by the current software development processes which are: verification and validation.
- Fifteen KPAs are partially supported by the current software development processes which are: causal analysis and resolution, project planning, project monitoring and control, measurement and analysis, process and product

quality assurance, configuration management, requirements development, technical solution, product integration, organizational training, integrated project management +IPPD, risk management, decision analysis and resolution, organizational process performance, and requirement management.

- Four KPAs are not supported by the current software development processes which are: supplier agreement management, organizational process definition +IPPD, organizational training, and quantitative project management.

Accordingly, these results were the main motivation for this firm to improve their software development processes by implementing the modified Extended-XP method as a general method for the software development. Therefore, based on the current software development process roles of the firm's employees, the framework-SEPG members modified these roles to be suitable with the roles of the framework which are used inside the modified Extended-XP method. Then, they distributed the new roles of the framework to the project team members commensurate with their experiences. Table 6.6 shows the new software development process roles of the project team compared to their current roles.

Table 6.6: The New Software Development Processes Roles of the Project Team Members Compared to Their Current Roles for the First Case Study

Current Roles		**New Roles**
Internal Members	Firm Manager	Project Manager
	Project OR Team Leaders	Coach, Tracker
	Technical Members	Programmers, Testers
	SEPG Members	Framework-SEPG Members, Extended-XP-SEPG Members
External Members	Customer	Customer
	Consultant (If need)	Consultant (If need)

6.3.1.2 Stage Two: Adopting the Modified Extended-XP Method

As a result of stage one, the new roles of each employee in this firm was specified. In this stage, there is need for educating and training the firm's employees on the modified Extended-XP method. This process of educating and training was prepared and executed by the framework-SEPG members, where they presented the modified Extended-XP method in five days of training courses. On the last day, the two framework-SEPG members examined the project team about their specific roles to make sure that they understood their new roles. As a result of the educating and training examination, the framework-SEGP members were sure that all the teams were ready to participate in implementing the Extended-XP method and there is no need for further educating and training. Furthermore, the documentation of the modified Extended-XP method was given to all the participants to be used as guidance during the software development processes. At the end of this stage, the

training and educating documentations and results were kept in the project repository by the framework-SEPG members. As a result of this phase, the project team was ready to implement the modified Extended-XP method. The next points illustrate the phases of the modified Extended-XP method that were used to develop the computer skills online examination system.

- **Phase One: Requirement Management**

  In this phase, the customer wrote the needed stories of the first release, where each story had one feature. All features were divided by the programmers into three modules as shown in Table 6.7.

Table 6.7: User Stories Modules for of the First Case Study

User Requirements Modules	Features
Administration Module	• The administrator has the full-fledged rights over the program. • Can create/delete an account. • Can view the accounts. • Can change the password. • Can hide any kind of features from both of users (examiner and student. Example: can specify the coordinator for insert, delete, and edit the questions and marks). • Can access all the accounts of the faculty members/students.
Examiner Module	• Insert/delete/edit the exams data (questions, and marks) by the coordinator. • Can view student answers and mark.
Student Module	• Can view their marks. • Can view the various reading material. • Can view his profile (university ID, student name). • Can reset his password.

Based on these modules, the programmers checked to see if there were needs for additional development tools or services to develop this system. As a result, there was no need for any external supplies, as all the required development tools,

services and technologies were ready and familiar with the project team. Then, the conceptual system prototype was developed by programmers to explore the architecture possibilities. Figure 6.1 presents the conceptual prototype. This prototype consists of two general components: Client Application Layers and Widows Communication Foundation (WCF) Service layers:

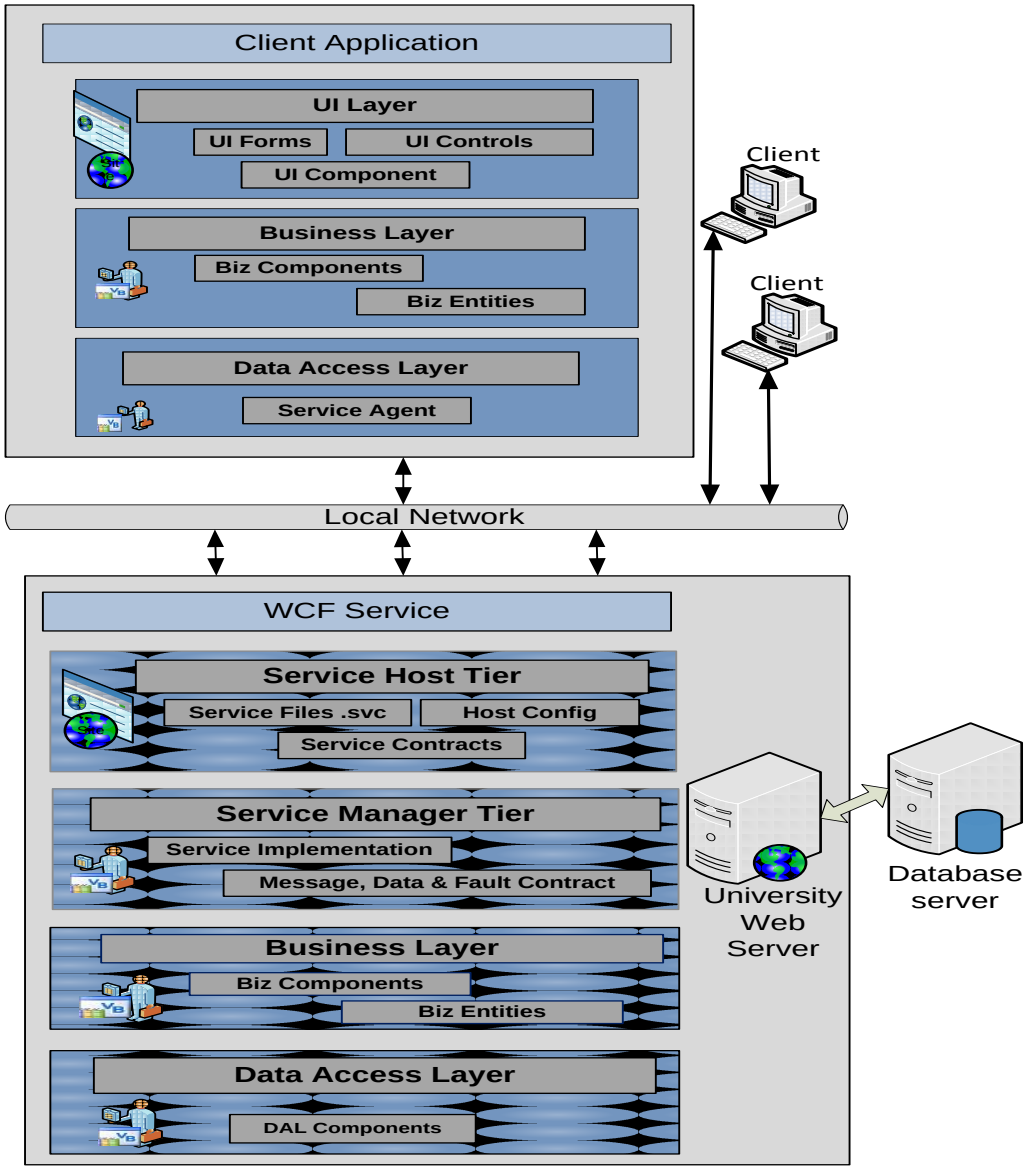

Figure 6.1: Conceptual System Prototype of Computer Skills Online Examination System

- Client Application Layers:

- The Presentation Layer (UI): responsible for presenting exams to the client side and for providing examinations entry and validation for the instructor.
- The Business Logic layer (BLL): Also known as middle layer is composed of more than one layer.
- The Business Logic Layer: responsible for handling all the examination entry logic, setting marks for questions, managing exam questions and time for students, and number of exams allowed for each student.
- The Business Entity Model: responsible for defining the entities used (i.e. exams, questions, and students) and their data types. It acts as a unified data catalog to the online examination System.
- The Data Access Layer (DAL): responsible for all of the data store and data retrieve operations from the data sources. The Data Access layer includes the Service Agents class library, which is responsible for transforming the data that came from or sent to the examination service.

- WCF Service Layers:

- Service Host Tier (UI): The service host tier will host the service contracts inside the University web server and expose them to the local network using service files (.svc). The Service Contacts are per Interfaces object implemented in the online examination service

manager tier which communicates with the system business DLLs to get the business objects. The host configuration file (Host Config) will define WCF endpoints and bindings exposed to the Local Area Network (LAN).

- Service Manager Tier: responsible for providing the implementation for the online examination system service contracts. The implementation provides an entry point to the business logic tier to transfer examination data. It also declares messages, data contracts, and fault contracts which will be used to communicate with the clients.
- The Business Logic layer (BLL): as with the client application, the business logic tier will be responsible for managing the business rules for the examination system and passing the data to the Data Access Layer(DAL) using business entities
- The Data Access Layer (DAL): at the service level, this component contains the logic which has the local database operations, CRUD operations and data retrieval operations.

Based on the conceptual system prototype, the programmers extracted the required tasks for the features that are mentioned in Table 6.7, and estimated how long it would take each task to be implemented by two programmers (pair programming). Accordingly, the customer and the programmers decided together how to prioritize each task. Table 6.8 presents the schedule for the tasks for the first release in detail.

Table 6.8 shows that the planning schedule consists of one release to develop the computer skills online examination system. This release has nineteen tasks. Furthermore, the table shows the estimated time for each task and also the level of priority for the task. At the end of this phase, the conceptual prototype description and the user stories tasks of the first release were kept in the project repository by the programmers and the tasks of the first release were used as inputs in the development phase.

Table 6.8: Planned Tasks of the First Release for the First Case Study

Tasks Name (Release One)	**Estimated Time (Day)**	**Priority**
T1: Detailing database design	2	High
T2: Building unit tests	3	
T3: Preparing entity model	1	
T4: Preparing data layer and business layer	2	
T5: Implementing interfaces	3	
T6: Securing Service	1	↓
T7: Login screen	0.5	
T8: Adding/Edit account page	1	↓
T9: Viewing accounts page	0.5	
T10: User permissions page	1	↓
T11: Adding/Modifying exams data	1.5	
T12: Setting exam date/time and duration	0.5	↓
T13: Viewing previous exams	1	
T14: Viewing students answers and marks	1	↓
T15: Student Login and setting logged in user information	0.5	
T16: Displaying exam questions and answers	2	
T17: Managing skipped question	1	
T18: Managing remaining time	0.5	
T19: Resetting password option	0.5	Low

- **Phase Two: Development**

  Based on Table 6.8, the required tasks are those entered from the requirement management phase. The schedule set in the first release was broken down into two iterations which are: first iteration, which consists of ten tasks and second iteration which consists of nine tasks. Table 6.9 shows the iterations of the first release.

  As shown in Table 6.9, the estimated time for each task was identified for the two iterations. Then, based on the tasks of this release, the programmers started to write the required code for the two iterations sequentially, where the unit test was used to test each line of coding before the writing. Moreover, functional tests were developed by programmers and used by the customer at the end of each iteration. Additionally, there were several technical tools that were used in developing the computer skills online examination system during the development phase. Table 6.10 shows these technical tools. Finally, at the end of this phase, the system was ready and entered the next phase and the development data were documented in the project repository.

Table 6.9: Iterations of the First Release for the First Case Study

Release	Iteration	Tasks	Estimated Time (Day)	Priority
1	1	T1	2	High
		T 2	3	↓
		T 3	1	↓
		T 4	2	
		T 5	3	↓
		T 6	1	↓
		T 7	0.5	
		T 8	1	
		T 9	0.5	
		T 10	1	Low
	2	T 11	1.5	High
		T 12	0.5	
		T 13	1	↓
		T 14	1	↓
		T 15	0.5	↓
		T 16	2	
		T 17	1	
		T 18	0.5	
		T 19	0.5	Low

Table 6.10: Technical Tools of the First Case Study

Items	Description
Language	Visual Basic .Net
Database	Microsoft SQL Server 2008
Development Environment	Microsoft Visual Studio Team System 2008
SCM	Microsoft Team Foundation Server (TFS)
Unit Testing	Microsoft Visual Studio
Documents	Microsoft Office 2007
Web Server	Internet Information Services 7.0

- **Phase Three: Product Delivery and Product & Process Efficiency**

As a result of the development phase, the first version of the product was developed by the first release. At the start of this phase, there was one meeting held with the project team to ensure that the customer's features had been implemented during the development phase, where project repository was used as a general guidance for these features. As a result of this meeting, the manager argued that all the customer's features had been developed in the system. Accordingly, several performance checks were done by programmers and the system was checked by customers to ensure that the system worked as intended. Based on the results of the first version of system sent to the customer, they agreed on developing the last features, and they also suggested new features to improve this system. These features are:

- Setting users' subjects in administration.
- Adding subject selection to examiner module.
- Adding subject selection to student module.
- Creating students' marks reports.

With regards to the suggested features, the second release was started again from the exploration process in the first phase of the modified Extended-XP method. Here, these features were shown to programmers to modify the conceptual system prototype. But, they argued that there was no need for any modification on this prototype. Furthermore, the programmers checked for the need for more development tools or services to develop this system. Nevertheless, all the required development tools, services, and technologies were ready and familiar to the project team. Based on this, the project team started to plan these user

stories, where programmers made an estimate for each feature and how long it would take to implement. Based on these estimations, the customer and programmers decided together to prioritize each feature.

Table 6.11 shows the tasks of the second release. Based on this, the schedule set of this release was planned to be developed in one iteration. Accordingly, the programmers started to write the code of these tasks, where unit test was used to test each line of coding before it was written. Additionally, functional tests were developed by the programmers and used by customer at the end of this iteration. Finally, at the end of this release, the system was ready to enter the next phase, and the developing data was documented in the project repository.

Table 6.11: Tasks of the Second Release for the First Case Study

Release	Iteration	Tasks	Estimated Time (Day)	Priority
2	1	T 20: Setting users subjects in administration.	1	High
		T 21: Adding subject selection to examiner module.	1	↓
				↓
		T 22: Adding subject selection to student module	1.5	↓
				↓
		T 23: Creating students marks reports.	1.5	Low

Based on developing the features of the second release, the second version of the system was developed. Consequently, the manager, programmers, coach, and tracker met to check the implementation of all required features for the two

releases that depended on the system and the project repository. As a result of this meeting, the manager confirmed that all features were already taken into account in the software system and they also checked the performance of the system. Accordingly, the system was shown to the customer and already several functional tests had been done by customer to check if the system worked as intended. As a result of these checks, the customer was satisfied with this system and there were no new features. Then, the system entered the maintenance & death phase.

Several metrics were calculated by the coach and tracker in this phase. These metrics aimed to check the processes' performance and to ensure the quality of the development processes. Table 6.12 shows the differences between the estimated and actual implementation times for all the tasks of the two releases. Furthermore, Table 6.13 shows the results of several metrics to ensure the quality of the development processes.

In this project, one pair programmers participated in developing the releases. Accordingly, Table 6.12 shows the actual and estimated times for developing the two releases as follows:

- Estimated time = 28.5 days.
- Actual time = 33.5 days.
- Daily time= 33.5 - 28.5= 5 days.

Table 6.12: The Differences between the Estimated and Actual Implementation Times for All the Tasks of the Releases for the first Case Study

Release	Iteration	Tasks	Estimated	Actual	Difference
1	1	1	2	2.5	+ 0.5
		2	3	3.5	+ 0.5
		3	1	1	0
		4	2	2.5	+ 0.5
		5	3	3	0
		6	1	1	0
		7	0.5	1	+ 0.5
		8	1	1	0
		9	0.5	0.5	0
		10	1	1	0
	2	11	1.5	2	+ 0.5
		12	0.5	0.5	0
		13	1	1	0
		14	1	1.5	+ 0.5
		15	0.5	0.5	0
		16	2	2.5	+ 0.5
		17	1	1	0
		18	0.5	1	+ 0.5
		19	0.5	0.5	0
2	1	20	1	1	0
		21	1	1.5	+ 0.5
		22	1.5	2	+ 0.5
		23	1.5	1.5	0
Total			28.5	33.5	+ 5

Table 6.13: Metrics of Processes Quality Assurance of the First Case Study

Metrics	Average
Percentage of test cases that are running successfully	87/93=93%
Percentage of acceptance test that run successfully	32/36= 88%
Length of pair programming sessions	2-2:30 hours
Project velocity compared with estimates for release 1 & 2.	28.5/33.5= 85%

At the end of this phase, the metrics of processes performance and quality assurance were put into the project repository to help with the measurement of the same user requirements for the next projects.

- **Phase Four: Maintenance & Death**

  In this phase, the computer skills online examination system was installed and tested in the university environment for two weeks and all the user requirements were tested by the end users. After implementing the system in the real environment, it showed that there was no need for further modification of the system. Accordingly, the system was ready to be used for online examinations by students.

6.3.1.3 Stage Three: Identifying the Best Practices of the Current Project

In this stage, the framework-SEPG members were responsible for discussing the implementation of the modified framework by meeting with the project team to identify the software development processes best practices. CMMI-Dev1.2 specific practices were used as main items in extracting the best practices of the current project. As a result of this meeting, the framework-SEPG members identified the best practices of implementing the modified framework in developing the online computer skills examination system, which are:

- **Best Practices of CMMI-Dev1.2 Level Two:** in this level, there are six KPAs that were achieved in developing the online computer skills examination system, and the specific practices of each KPA had been applied by using the framework. These areas are: requirement management, project planning, project monitoring

and control, measurement and analysis, process and product quality assurance, and configuration management. The supplier agreement management KPA was not applied in developing the system because there was no need for supporting development tools or services by external suppliers.

- **Best Practices of CMMI-Dev1.2 Level Three:** in this level, the specific goals of the eight KPAs were achieved in developing the online computer skills examination system, and the specific practices of each KPA were applied by using the framework. These areas are: requirements development, technical solution, product integration, verification, validation, organizational process definition + IPPD, organizational training, and integrated project management +IPPD. The specific goals of the other three KPAs were achieved in developing the system, but in a different ways compared to the specific practices of the following KPAs in CMMI-Dev1.2, which are: organizational process focus, risk management, and decision analysis and resolution.
- **Best Practices of CMMI-Dev1.2 Level Four:** in this level, the specific goals of two KPAs were achieved in developing the system in a different way compared with the specific practices of these areas in CMMI-Dev1.2. These KPAs are: organizational process performance and quantitative project management.
- **Best Practices of CMMI-Dev1.2 Level Five:** in this level, the specific goals of the causal analysis and resolution KPA were achieved by developing the system in a different ways compared to the specific practices of this area in CMMI-Dev1.2.

At the end of this stage, the framework-SEPG members kept these best practices in the project repository to be taken into account for incoming projects.

6.3.1.4 Summary of Developing the Computer Skills Online Examination System by the Modified Software Development Process Improvement Framework

Firm "X" had developed the computer skills online examination system by implementing the modified framework. After the project was concluded, the framework-SEPG members wrote a report that reflected the actual time that had been spent on each stage of this framework. Table 6.14 presents the actual time of each stage.

Table 6.14: Actual Time for Implementing the Framework in the First Case Study

Stages	Activities	Time (By Day)
Before stages	Understanding the framework by framework-SEPG members	5
Stage One	Assessing the current processes	2
	Modifying the current roles	2
Stage Two	Educating and training the project team on the Extended-XP method	5
	Adopting the Extended-XP method (all phases)	64
Stage Three	Analyze the result and extract the best practices	4
Total		82

As shown in Table 6.14, the implementation of the modified framework took 82 days, with the project team working five days a week. Accordingly, the actual time for implementing this framework was 84 days/ five days (weekly) = 16.4 weeks, where the actual time for the modified Extended-XP method was 64 days / five days (weekly) = 12.8 weeks. These periods of time will be taken into account by the

project team during their answers on the evaluation criteria to evaluate the effectiveness of the framework. Section 6.4 will discuss the evaluation criteria.

As for the framework roles that were used in this case study, ten members participated in developing the computer skills online examination system by implementing the modified framework, they are: one project manager, two framework-SEPG members, two coaches, one tracker, two programmers (pair programmers), one tester, and one on-site customer.

6.3.2 Case Study Two: Developing the Online Brokerage System by "Y" Firm

"Y" firm was founded in 2006 as a Jordanian SSDF. This firm specializes in providing great software solutions for several business sectors which are premised on the latest technological trends. "Y" firm's expertise is on software consulting, business applications & web development, outsourcing services, intelligent data analysis services and applications. This firm has 38 workers, working in managing and developing the software programs, comprising one manager, three project managers, six project or team leaders, ten software engineering process group members, and eighteen technical members. "Y" firm has developed the Online Brokerage System used as a case study for this research.

Online Brokerage System was developed to be delivered to brokerage companies that deal with the financial stock market. This system acts as the liaison between their clients and the financial stock market. The system serves two types of users: the broker and the client, each position has its own functions and privileges which are

determined by the system. The broker deals with the financial stock market and the client makes deals and orders. The brokerage system has a set of services provided, and these services are applied within the system itself such as opening accounts, placing orders, monitoring client's portfolios, tracking market prices, and generating financial reports. The following points illustrate the development of this system by "Y" firm based on the modified software development process improvement framework presented in this study.

Before starting the implementation of the modified framework, the researcher met the manager of "Y" firm to obtain permission in order to begin the implementation of the framework in this firm. Then, two SEPG members were asked by the manager to act as framework-SEPG members. These members were then given one week to read and understand the description of the framework. Then, the framework-SEPG members started to implement the framework by improving their software development processes when developing the online brokerage system. Sections 6.3.2.1 to 6.3.2.3 describe in detail the stages of improving the current software development processes to develop the online brokerage system.

6.3.2.1 Stage One: Assessing the Current Software Development Processes

In this stage, the two framework-SEPG members chose the Microsoft Office as a simple project repository to store the project data during the implementation of the framework. Then, they started to assess the current software development processes in the firm based on CMMI-Dev1.2 KPAs. This assessment was done by one meeting between the firm manager, project manager, three team leaders, four

technical members, and framework-SEPG members who were working in the firm. From the results of this meeting, the framework-SEPG members identified the capability levels of the current software development processes of the firm. In this stage, the two SEPG members assessed the current software development process in the firm based on CMMI-Dev1.2 questions. Table 6.15 shows the capability levels of the current software development processes of the firm.

Table 6.15: Supported Levels of CMMI-Dev1.2 KPAs of the Current Software Development Processes for the Second Case Study

CMMI-Dev1.2 KPAs	Largely Supported	Partially Supported	Not Supported
Level Three KPAs			
Configuration Management	X		
Requirement Management		X	
Project Planning		X	
Project Monitoring and Control		X	
Process and Product Quality Assurance		X	
Supplier Agreement Management			X
Measurement and Analysis			X
Level Three KPAs			
Verification	X		
Technical Solution		X	
Product Integration		X	
Validation		X	
Organizational Process Focus		X	
Risk Management		X	
Decision Analysis and Resolution		X	
Requirements Development			X
Organizational Process Definition +IPPD			X
Organizational Training			X
Integrated Project Management +IPPD			X
Level Four KPAs			
Quantitative Project Management		X	
Organizational Process Performance			X
Level Five KPAs			
Causal Analysis and Resolution		X	

The following can be concluded from the results presented in Table 6.15:

- Two KPAs are largely supported by the current software development processes, which are: configuration management and verification
- Twelve KPAs are partially supported by the current software development processes, which are: requirement management, project planning, project monitoring and control, process and product quality assurance, technical solution, product integration, validation, organizational process focus, risk management, decision analysis and resolution, quantitative project management, and causal analysis and resolution.
- Seven KPAs are not supported by the current software development processes, which are: supplier agreement management, measurement and analysis, requirements development, organizational process definition +IPPD, organizational training, integrated project management +IPPD, and organizational process performance.

These results were the main motivation for this firm to improve their software development processes by implementing the modified Extended-XP method as a general method for the software development. Based on the current software development processes' roles of the firm's employees, the framework-SEPG members modified these roles to be suitable with the roles of the framework. Then, they distributed the new roles of the framework to the project team members commensurate with their experiences. Table 6.16 shows the new software development processes' roles of the project team compared to their current roles.

Table 6.16: The New Software Development Processes Roles of the Project Team Members Compared to Their Current Roles for Second Case Study

Current Roles		New Roles
Internal Members	Project Manager	Project Manager
	Project OR Team Leaders	Coach, Tracker
	Technical Members	Programmers, Testers
	SEPG Members	Framework-SEPG Members, Extended-XP-SEPG Members
External Members	Customer	Customer
	Consultant (If need)	Consultant (If need)

6.3.2.2 Stage Two: Adopting the Modified Extended-XP Method

Based on the results of stage one, the new roles of each employee in this firm were specified. In this stage there was need for educating and training the firm's employees on the modified Extended-XP method. This process of educating and training was prepared and executed by the framework-SEPG members, where they presented the modified Extended-XP method in five days of educating and training courses. On the last day, the two framework-SEPG members questioned the project team about their specific roles, to ensure that they fully understood their new roles. As a result of an educating and training examination, the framework-SEGP members were confident that all the teams were ready to participate in implementing the modified Extended-XP method and that there was no need for further educating and training. Furthermore, the documentation of the Extended-XP method was given to all the participants to be used as guidance during the software development processes. At the end of this stage, the training and educating documentations and results were kept in the project repository by the framework-SEPG members. From

the results of this phase, the project team was ready to implement the Extended-XP method. The following points illustrate the phases of the Extended-XP method that was used to develop the online brokerage system.

- **Phase One: Requirement Management**

  In this phase, the customer wrote the needed stories of the system as a two module, where each module had several features as shown in Table 6.17.

Table 6.17: User Stories Modules of the Second Case Study

User Requirements Modules	Features
Front Office	• Automate client registration • Opening a demo account • Secure login to the brokerage server • Live market watch showing selected stocks • Adding removing stocks from the market watch • Displaying client account information • Placing buy or sell market orders • Placing limit orders • Placing stop orders • One cancels the other order • Cancel or replace order • Displaying current profit/loss for each stock and total profit/loss. • Probing server connection and reconnect automatically. • Displaying stocks charts and history.
Back office	• Client accounts administration • Ability to connect multiple front end clients simultaneously. • Tracking accounts positions and connectivity. • Connecting to the market data source provider and sending prices to clients. • Setting opening and closing hours of trading. • Placing order for accounts. • Adding new clients. • Editing or deleting existing clients. • Managing financial transaction for the accounts. • Generating statement report for each account at the end of day.

Based on these modules, the programmers checked to see if there were additional needs for more development tools or services to develop this system. They found that two products needed to be supported by external suppliers. Thereafter, the programmers identified the required characteristics of the required products as follows:

- **Charting Control Requirements:** charting control will be used to display financial stock charts on the system. The charting control should satisfy the following requirements to be used on the system:
    - Supporting Microsoft .Net environment.
    - Supporting all popular stock charts used on the financial markets.
    - Charting real-time stock data.
    - Loading data from textiles.
    - Embed objects used by the system like buy, sell, and orders.
    - Display financial studies.
    - Saving and printing charts.
    - Ability to zoom-in, zoom-out, scrolling to a date.

  Based on the features of the required product, Extended-XP-SEPG members sent these features to three specialist suppliers and asked them to present the details of the available products to Firm "Y" (product features, price, and time to deliver). Table 6.18 shows the suppliers' offers.

Table 6.18: Suppliers Offers of Charting Control Product

Supplier	Product	Delivery Time	Price
A	Easy Financial/Stock Chart - Windows Edition	On payment - Site download	$1499 with source code
B	Financial Charting Component	On payment - Site download	$350
C	StockChartX V.5 Professional	On payment - Site download	$689 / $1,389 with source code

The Extended-XP-SEPG members discussed these offers and the StockChartX V.5 Professional Product from "C" firm was chosen for the following reasons:

- Support of VB.net, C# and Microsoft .Net environment.
- High performance on .Net environment.
- Supporting multiple charts formats
- Support of Gregorian/Julian dates.
- Ability to embed objects within charts.
- Availability of source code if needed.

- **Stocks Market Data Feed Requirements:** data feed will provide the system with online and historical data prices for the trading stocks; it should meet the following requirements:
    - Supporting Transmission Control Protocol (TCP) to provide online prices.
    - Providing real-time prices for selected stocks.

- Providing on demand historical data for selected stocks.
- High availability and reliability of the data service.
- Data accuracy and data loss strategy.

Based on the features of the required product, the Extended-XP-SEPG members sent these features to three specialist suppliers and asked them to present the details of the available products (product features, price, and time to deliver). Table 6.19 shows the suppliers' offers. Then, Extended-XP-SEPG members discussed these offers and choose TAL Data product from "E" for several reasons as following:

- Providing real time market data, news and alerts.
- Supporting TCP/IP connectivity
- Affordable price based on selected markets.
- Customer support for free.
- High availability and reliability.

Table 6.19: Suppliers Offers of Stocks Market Data Feed Product

Company name	Product & Features	Price
D	Digital Data Feed • Broadcast, raw data, quert/response, end-of-day. • Futures, stocks, indices and forex. • Fundamental and technical data. • News, weather.	Real Time Commodities + Equities $2,500/month
E	TAL Data • Streaming real-time market data. • Stocks, futures, options, fixed-income securities, and forex. • Technical scanning formulas. • Historical data (tick, intraday, monthly, and seasonal). • Bond, forex data.	RealTickPRO $650/month
F	IQFEED • Streaming real-time and delayed data. • Equity, futures, options, index, forex. • Depth-of-market, level2. • News, fundamental data.	$220 - $800 depends on selected markets

As a result of supplying the required products, all of the required development tools, services, and technologies were ready and familiar to the project team. Then, the general conceptual system prototype (system design overview) was developed by programmers to explore the architecture possibilities as it shown in Figure 6.2.

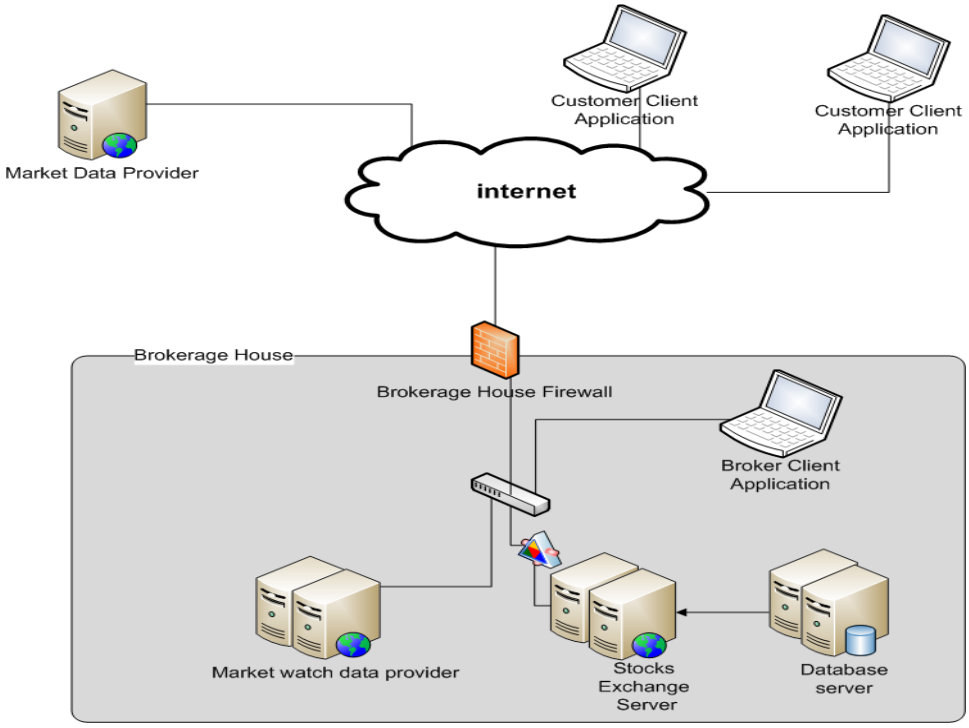

Figure 6.2: System Design Overview of the Online Brokerage System

As shown in Figure 6.2, the system design overview consists of the following components:

- **Stocks Exchange Servers:** these servers will be responsible for all the financial operations including placing orders, monitoring client's positions, managing clients, managing stock trading hours, and generating reports.
- **Market Watch Data Server:** will connect to the market data provider using TCP sockets and retrieve a real time process for the stocks. It will be responsible for providing the client applications with up to date prices based on the subscribed stocks. As part of the server function, it will cache historical data for the stocks to be provided upon clients' requests.

- **Database Servers:** two clustered database servers will maintain the customer's data.
- **Market Data Provider:** this is a third party component which provides real time stocks prices to the system; it will also provide historical data for the stocks to plot the charts.
- **Client Application:** there will be two types of client applications according to the functionality provided: customer client application and broker client application. The customer client application will be used by the brokerage house customers to monitor the market prices, issue buy/sell requests, view stocks charts, view account positions, and place orders. The broker client application will be used by brokerage house officers to open new customer's accounts, manage customer's funds, monitor customer's positions, define trading hours for stocks, place orders for customers, and generate financial reports.

The conceptual system prototype of online brokerage system consists of the following two general components: server model and client application logical model. The next points illustrate these models in detail.

- Server Model:

This model consists of three layers. Figure 7.3 illustrates the components of the server model.

- **Web Application Layer:** this layer will be responsible for managing client's connections to the server using both WCF (Windows Communication Services) and TCP sockets. This model enables

flexible deployment options, for example. Web servers do not need to have direct access to the database as the calls are invoked through the Middle Tier Business Layer.

- **Middle Tier Business Service Layer**: this layer is responsible for implementing the logic of the business services and can be changed without affecting other layers in the application, as well as it can be spanned horizontally across servers to scalability and load balancing.
- **Data Access Layer (DAL):** database specific operations are hidden from the web server or the business layer and are responsible for storing the trading data on the database.

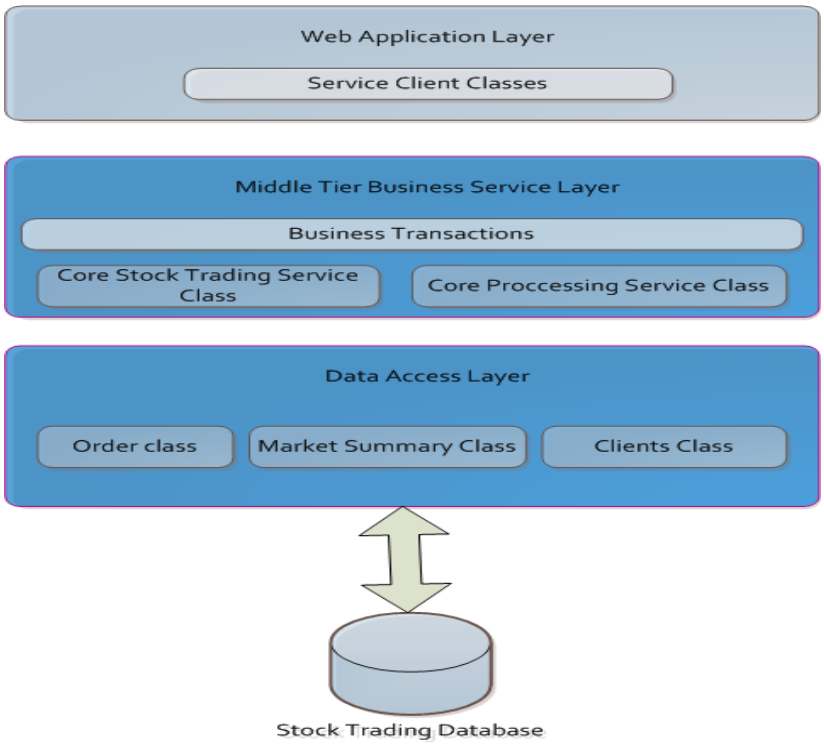

Figure 6.3: Server Model of the Online Brokerage System

- Client Application Logical Model

This model consists of two layers with six modules. Figure 6.4 shows the components of the client application logical model.

- **Presentation Layer:** the presentation layer is responsible for displaying the application layout to the user. It uses the Market Watch module to display stocks prices, Charting Module to retrieve and display stocks charts, Orders Module to display customers' positions and ability to place new transactions, and Account Management Module to calculate and display customer's financial status and customer's information.
- **Communications Layer:** will maintain connections with Stocks Trading server and Market Watch data feed server. It will send/receive new orders details using WCF to the Stocks Trading server and initiates TCP connection with the Market Watch data feed server to receive updated prices for the stocks.
- **Market Watch Module:** will manage retrieval of stocks prices that the customer is subscribed to, and track changes in their prices using the communication layer. It will provide customers the functionality to add/remove new stocks to their profiles and will provide the broker the functionality of adding/removing new stocks to the trading system.
- **Charting Module:** will create charts for selected stocks using historical data retrieved from the Market Watch data feed server.
- **Orders Module:** handles retrieving customers, opened positions and placed orders, and calculates their profit/loss of each position based on

the prices retrieved from the market data source. It will also provide buy/sell functionality to the client application.

- **Account Management Module:** will handle user login to the server and retrieval of information. It will provide the ability to open demo accounts for the customers and it will provide the broker the functionality of creating new customer accounts and updating their information.
- **Reporting Module:** will provide the functionally of generating end of day reports for the brokerage hours customers with details of their positions profit/loss, trading volume reports, and other financial reports. This functionality will be available to the broker house users only.
- **Stocks Hours Module:** will provide the officer the functionally of setting active stocks and the trading hours of each stock. This functionality will be available to the broker house users only.

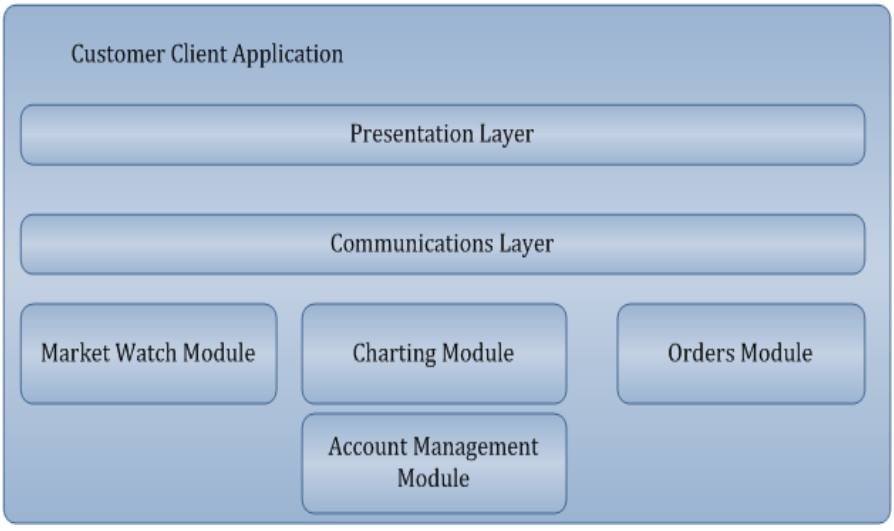

Figure 6.4: Customer Client Logical Model of the Online Brokerage System

Based on the conceptual prototype of the online brokerage system, programmers extracted the required tasks for the features that are mentioned in Table 6.20, and estimated how long it would take each task to be implemented by one pair of programmers. Accordingly, customers and programmers decided together the level of priority for each task. Table 6.20 presents the schedule planning of the required tasks in detail.

Table 6.20: Planned Tasks of all Features for the Second Case Study

Task Name	**Estimated Time (Day)**	**Priority**
T1: Preparing Data Access Layer	1	High
T2: Preparing Business Managers	2	↓
T3: Preparing Business Entities	1	↓
T4: Listing clients' accounts	2	
T5: Updating client account information	1	↓
T6: Resetting account password	1	↓
T7: Viewing account activities	2	↓
T8: enabling/disabling accounts	1	
T9: Building multithreaded TCP/IP core engine	2	↓
T10: Securing client connection	2	↓
T11: Checking account status on connection	0.5	
T12: Adding connected client to connected clients pool	0.5	↓
T13: Sending client portfolio details	2	
T14: Sending client orders upon connect	2	
T15: Connecting client to market data source	1	
T16: Building data feed connectivity engine	2	
T 17: Tracking connectivity of market data feed and reconnecting when needed	1	
T18: Reading stocks prices from data source	2	
T19: Broadcasting prices upon updates to the registered clients	1	↓
T20:Viewing clients opened positions and portfolio details	2	
T21: Adding/Closing new order for an account	2	
T 22: Adding/Removing one cancel another order to an account	2	↓
T23: Account liquidation	2	
T24: Closing an account	1	

T25: Preparing Business Managers	2	↓
T26:Preparing Business Entities	2	
T27:Creating Asynchronous TCP socket interface	1	
T28: Securing connection to server	1	↓
T29: Monitoring server connection and reconnect if disconnected	1	
T30: Open orders calculation engine	2	
T31: Retrieving stocks list	1	↓
T32: Updating stocks	0.5	
T33: Adding/Removing stock	1	
T34: Depositing money to client account	1.5	↓
T35: Withdrawing money from client account	1.5	
T36: Stocks Trading Hours	3	
T37: Retrieving a list of all active stocks	2	↓
T38: Set trading time for each stock	2	
T39: Connecting trading hours to trading server engine	2	
T40: Placing buy/sell orders	2	↓
T41: Placing limit order	1.5	↓
T42: Placing Stop orders	1.5	
T43: Placing one cancel the other operation	2	
T44: Cancel or replace stop order	1	↓
T45: Displaying available account types	0.5	
T46: Filling user information	1	
T47: Sending new account information to server	1	↓
T48: Displaying current total profit/loss	1.5	
T49: Displaying opened stocks and their profit/loss	3	
T50: Caching account information	1	↓
T51: End of day statement report of all accounts and their transactions	2	
T52: Trading volume throughout the day	2	
T53: Account statement details and opened positions	1.5	↓
T54: Account financial transactions report	1.5	
T55: Research about charting controls	2	
T56: Integrating of charting control on the client	2	↓
T57: Retrieving stock history prices	2.5	
T58: Displaying history prices on the chart	2	
T59: Caching stock prices on the client	2	Low

As shown in Table 6.20, programmers and customers estimated the time needed for each task, and also arranged these tasks from high to low priority. Depending on the high number of these tasks, the programmers and customers divided these tasks into two releases, where the first release consisted of the first thirty three tasks (T 1 to T 33), and the second release consisted of the last twenty six tasks (T 34 to T 59). At the end of this phase, the conceptual prototype description and the user stories tasks from the two releases were kept in the project repository. Afterwards, the tasks of the two releases were used as inputs in the development phase.

- **Phase Two: Development**

As a result of the previous phases, there are two releases that need to be developed in this phase. At the beginning of this phase, the schedule set for the two releases was broken down into two iterations for each release. Table 6.21 shows each release with their iterations and included tasks.

Table 6.21: Iterations of the Two Releases for the Second Case Study

Release Number	Iteration Number	Tasks	Priority
1	1	T1 to T15	High
	2	T16 to T 33	↓
2	1	T34 to T50	↓
	2	T51 to T59	Low

As shown in Table 6.21, there are two releases and two iterations for each release. Accordingly, two pair programmers started to write the required code for the iterations of the first release sequentially according to the priority of these

iterations, where unit test was used to test each line of coding before writing. Moreover, functional tests were developed by the programmers and used by the customer at the end of each iteration. At the end of developing the first release, the first version of the product (release one) was sent to the next stage to make sure that all features of the first release had been done by this version. This phase was started again in order to develop the features of the second release as the steps of the first release. At the end of developing the second release, the second version (two releases) of the product was sent to the next phase. Additionally, there were several technical tools that were used in developing the online brokerage system during this phase. Table 6.22 shows those technical tools. Finally, the system was ready and entered the next phase. Additionally, the developing data was documented in the project repository.

Table 6.22: Technical Tools of the Second Case Study

Items	Description
Language	Visual Basic .Net
Database	Microsoft SQL Server 2008
Development Environment	Microsoft Visual Studio Team System 2008
SCM	Microsoft Team Foundation Server (TFS)
Unit Testing	NUNIT
Documents	Microsoft Office 2007
Web Server	Internet Information Services 7.0

- **Phase Three: Product Delivery and Product & Process Efficiency**

  At the end of the development phase, two releases were developed; therefore two versions of the product were entered into this phase. At the beginning of this phase, there was one meeting held for each release with the project team to make

sure that the customer's features have been implemented during the development phase, where project repository had been used as a general guidance for these features. As a result of these meetings, the manager agreed that all the customer's features had been developed in the system. Accordingly, the system was shown to the customer and already several functional tests were done by customer to check if the system worked as intended. As a result of these checks, the customer agreed that he was satisfied with the system and there were no new features needed. Thereby, the system entered the maintenance & death phase.

Several metrics were calculated by coach and tracker in this phase, where these metrics aimed to check the processes' performance to ensure the quality of the development processes. Table 6.23 presents the differences between the estimated and actual implementation times for all the tasks of the two releases. Furthermore, Table 6.24 shows the results of several metrics that were calculated to ensure the quality of the development processes.

As mentioned in the development phase, there are two pair programmers who participated in developing the releases. Therefore, the estimated times for all features were ninety three days and the actual times were ninety nine days. These estimated and actual times were for one pair of programmers. Nevertheless, there are two pairs of programmers who participated in developing the system. Accordingly, the actual and estimated times for developing the two releases were divided by two pairs as follows:

- Estimated time = 93 day/ 2 pairs = 46.5 days.

- Actual time = 99 days/ 2 pairs= 49.5 days.
- Daily time= 49.5 - 46.5= 3 days.

Table 6.23: The Differences between the Estimated and Actual Implementation Times for All the Tasks for the Two Releases of the Second Case Study

Release	Iteration	Tasks	Estimated Time (Day)	Actual Time (Day)	Difference
1	1	T1	1	1	0
		T2	2	1.5	- 0.5
		T3	1	1	0
		T4	2	2	0
		T5	1	1	0
		T6	1	1.5	+ 0.5
		T7	2	3	+ 1
		T8	1	1	0
		T9	2	2	0
		T10	2	2	0
		T11	0.5	0.5	0
		T12	0.5	0.5	0
		T13	2	2	0
		T14	2	3	+ 1
		T15	1	1	0
	2	T16	2	3	+ 1
		T 17	1	1	0
		T18	2	1	-1
		T19	1	1.5	+ 0.5
		T20	2	2	0
		T21	2	2	0
		T 22	2	2.5	+ 0.5
		T23	2	2	0
		T24	1	1	0
		T25	2	2.5	+ 0.5
		T26	2	2	0
		T27	1	1.5	+ 0.5
		T28	1	1	0
		T29	1	1.5	+ 0.5
		T30	2	2	0
		T31	1	1.5	+ 0.5

		T32	0.5	0.5	0
		T33	1	1	0
2	1	T34	1.5	2	+ 0.5
		T35	1.5	1.5	0
		T36	3	2.5	-0.5
		T37	2	2	0
		T38	2	2	0
		T39	2	2	0
		T40	2	2	0
		T41	1.5	1.5	0
		T42	1.5	1.5	0
		T43	2	2.5	+ 0.5
		T44	1	1	0
		T45	0.5	0.5	0
		T46	1	1	0
		T47	1	1	0
		T48	1.5	1.5	0
		T49	3	2.5	-0.5
		T50	1	1	0
	2	T51	2	2.5	+ 0.5
		T52	2	2	0
		T53	1.5	2	+ 0.5
		T54	1.5	1.5	0
		T55	2	2.5	+ 0.5
		T56	2	2	0
		T57	2.5	2	-0.5
		T58	2	2	0
		T59	2	2	0
Total			93	99	+ 6

Table 6.24: Metrics of Processes Quality Assurance of the Second Case Study

Metrics	**Average**
Percentage of test cases that are running successfully	212/248= 85%
Percentage of acceptance test that run successfully	162/197= 82%
Length of pair programming session	2 hours
Project velocity compared with estimated for the two releases	46.5/49.5= 93%

At the end of this phase, the metrics of processes' performance and quality assurance were put into the project repository to help with the measurement of the same user requirements for the next projects.

- **Phase Four: Maintenance & Death**

  In this phase, the online brokerage system was installed and tested in the business environment for three weeks, and all the user requirements were tested by the end users. After implementing the system in the real environment, it was concluded that there was no need further modification required to the system. As such, the system was ready to be used by online brokers.

6.3.2.3 Stage Three: Identifying the Best Practices of the Current Project

In this stage, the framework-SEPG members were responsible for discussing the implementation of the framework by meeting with the project team to identify the software development processes' best practices. CMMI-Dev1.2 specific practices were used as main items in extracting the best practices of current project. As a result of this meeting, the framework-SEPG members identified the following best practices for implementing the framework in developing the online brokerage system:

- **Best Practices of CMMI-Dev1.2 Level Two:** in this level, the specific goals of all the seven KPAs were achieved in developing the online brokerage system and the specific practices of each KPA were applied by using the framework. These areas are: requirement management, project planning, supplier agreement management, project monitoring and control, measurement

and analysis, process and product quality assurance, and configuration management.

- **Best Practices of CMMI-Dev1.2 Level Three:** in this level, the specific goals of eight KPAs were achieved in developing the online brokerage system and the specific practices of each KPA had been applied by using the framework. These areas are: requirements development, technical solution, product integration, verification, validation, organizational process definition + IPPD, organizational training, and integrated project management +IPPD. Nevertheless, the specific goals of the other three KPAs were achieved by developing the system but in different way compared to the specific practices of these areas in CMMI-Dev1.2. These KPAs are: organizational process focus, risk management, and decision analysis and resolution.
- **Best Practices of CMMI-Dev1.2 Level Four:** in this level, the specific goals of the two KPAs were achieved by developing the system in different way compared to specific practices of these areas in CMMI-Dev1.2. These areas are: organizational process performance and quantitative project management.
- **Best Practices of CMMI-Dev1.2 Level Five:** in this level, the specific goals of the causal analysis and resolution KPA was achieved by developing the system in different way compared to specific practices of this area in CMMI-Dev1.2.

6.3.2.4 Summary of Developing the Online Brokerage System by the Modified Software Development Process Improvement Framework

Firm "Y" developed the online brokerage system by implementing the modified framework. Based on this, the framework-SEPG members wrote a report of the actual time that was spent on each stage of this framework. Table 6.25 presents the actual time of each stage.

Table 6.25: Actual Time for Implementing the Modified Framework in the Second Case Study

Stages	Activities	Time (By Day)
Before stages	Understanding the framework by framework-SEPG members	5
Stage One	Assessing the current processes	3
	Modifying the current roles	4
Stage Two	Educating and training the project team on the Extended-XP method	5
	Adopting the Extended-XP method (all phases)	85
Stage Three	Analyze the result and extract the best practices	3
Total		105

As shown in Table 6.25, the implementation of the modified framework took 105 days, with the project team working five days a week. Accordingly, the actual time for implementing the framework as a whole was 105 days/ five days (weekly) = 21 weeks, where the actual time for the modified Extended-XP method was 85 days / five days (weekly) = 17 weeks. These periods of time will be taken into account by

the project team during their answers on the evaluation criteria to evaluate the effectiveness of the framework. Section 6.4 will discuss the evaluation criteria.

With regards to the modified framework roles that were used in this case study; seventeen members participated in developing the online brokerage system by implementing the framework. The members included: one project manager, two framework-SEPG members, one Extended-XP-SEPG member, two coaches, two trackers, four programmers (two pairs), two testers, one on-site customer, and two suppliers.

6.4 Evaluating the Effectiveness of the Modified Software Development Process Improvement Framework

As discussed in Section 3.5, to ensure that the modified framework is effective for SSDFs, there is need to identify the required evaluation criteria that are needed to evaluate the effectiveness of the implementation of this framework by the two firms discussed above. Table 6.26 presents the resource of required criteria that used to evaluate the modified framework.

Prior to starting the evaluation process; the evaluation questionnaire was pre-tested to make sure that these questions are sufficient and suitable to be used in evaluating the effectiveness of the modified framework for SSDFs. In this regard, face-to-face interviews with two related researchers were carried out; and just slight corrections had been done on these questions to make it suitable and clear with the aim of the evaluation process.

Table 6.26: Research Variables for Evaluating the Modified Framework

Sources	Evaluation Criteria
Kitchenham (1998)	**Gain Satisfaction:** - Perceived usefulness - Decision support satisfaction - Comparison with other guidance – better - Cost – effectiveness - Clarity - clear and illuminate the process - Appropriateness for task
Garrity and Sanders (1998)	**Interface Satisfaction:** - Perceived ease of use - Appropriate for audience - Organization - well organized - Internally consistent - Presentation - readable and useful format
Kitchenham *et al.* (1997), and Garrity and Sanders (1998)	**Task Support Satisfaction:** - Ability to produce expected results - Ability to produce usable results - Completeness - adequate or sufficient - Ease of implementation - Understandability - simple to understand

The evaluation criteria that are mentioned in Table 6.26 were used as main variables in evaluating the effectiveness of the modified software development process improvement framework by interviewing the project teams who implemented the two case studies. The following points provide the results of the evaluation process:

- Gain Satisfaction Criteria

- **Perceived Usefulness:** the framework enabled the project teams in implementing their roles correctly with high effectiveness, as the practices of each role were clearly understood. Therefore, the productivity of each member of the project team was good compared to the ad-hoc manner, which had been used in the firm before implementing the Extended-XP method as a development method. In addition, the distributions of the roles in the first stage

of the framework were very familiar to the current role of each member which led them to execute their roles easily.

- **Decision Support Satisfaction:** the continuous communication during the framework (including the Extended-XP method life cycle) has the potential to reduce individual bias by involving all the members (including the customer) working as a team in the decision making process. Based on this, project managers were responsible for the confirmation of the decision making, where this is the main roles of the managers.
- **Comparison with other Guidance:** the framework was suitable for improving the software development processes compared to the traditional SPI models such as CMM, CMMI, ISO, SPICE, and BOOTSTRAP. These traditional models are used to improve the software development processes by "what to do" function. Nevertheless, this framework simplified the achieving of the specific goals of CMMI-Dev1.2 KPAs in a simple and smooth way by using Extended-XP method to know "how to do" the improvement.
- **Cost (Effectiveness):** the framework was cost-effective for several reasons such as: (1) supplying process helps in choosing the suitable products and services from the external suppliers within a set of feasible conditions and this minimizes the risk of purchasing from suppliers; (2) the coach enabled the project team to follow the right path and kept them working on the current features for the actual iteration; and (3) the tracker was careful not to interrupt the project team too many times.
- **Clarity (clear and illuminate the process):** the framework stages were very clear to the project teams, as each member had specific roles to do. Therefore,

there was no overlap between their roles. In addition, the framework was made very clear to the project teams by the educating and training that had been carried out in the first stage of this framework, and also the framework guidance helped the team during the implementation of the Extended-XP life cycle. Moreover, the roles of the coach and the tracker helped the project team in executing their roles during the development life cycle.

- **Appropriateness for Task:** the framework was appropriate for the task for which it had been developed, as it helped the firms in managing and improving their software development processes in a systemic and effective way compared to the ad-hoc manner which had been used before. In addition, the framework helped to know the best practices of developing the current project to help them in the organization and development issues for incoming projects.

- Interface Satisfaction Criteria

- **Perceived Ease of Use:** the educating and training process helped the project team in understanding the framework. Therefore, it was easy to be understood and used during the implementation.
- **Internally Consistent:** the stages of the framework and the roles of each member in the team were very clear. These roles helped in keeping the developing process consistent.
- **Organization (well organized):** the framework was organized and structured well, and the sequence of framework stages and Extended-XP phases helped to make the development activities easily understood.

- **Appropriate for Audience:** based on developing the systems by the Extended-XP method, the audiences were satisfied on product releases. This helped them add more features to the required products because the Extended-XP method is incremental and iterative software development method.
- **Presentation (readable and useful format):** the framework has a readable and useful format. The project teams argued that the phases of the Extended-XP method were very apparent and smooth, where the education and training process helped them to understand the method thoroughly.

- Task Support Satisfaction Criteria

- **Ability to Produce Expected Results:** the implementation of the framework returned high capability levels compared with the levels seen before implementing this framework, especially in terms of time and productivity.
- **Ability to Produce Usable Results:** the completed systems were usable by the end users, as the customers participated in developing the systems (On-Site Customer), so the products were very user-friendly for the systems owners.
- **Completeness (adequate or sufficient):** the framework was comprehensive for improving the software development and management processes in SSDFs. However, it would be more sufficient when all KPAs of CMMI-Dev1.2 level five are included.
- **Ease of Implementation:** the framework was very easy to implement, and the descriptions of each phase were very clear. Therefore, it was easy to know what the roles of each member in the developing process. The project teams also asserted that the Extended-XP method was easy to implement, where the

coach enabled the project team to follow the right path and kept them working in the right way. Nevertheless, the physical prototype was not suitable to be developed in the first phase of the Extended-XP method; therefore the conceptual prototype was more appropriate in this phase.

- **Understandability (simple to understand):** the framework was understandable. The project teams asserted that the activities of the Extended-XP method were easy to understand, especially after the education and training processes. In addition, the documentations of the Extended-XP method helped them in implementing this method in the right way.

As a result of the responses from the team members of the two case studies on the evolution criteria questions, it can be concluded that the framework is useful, useable, satisfied user needs and valid for use by SSDFs.

6.5 Conclusion

During the validation process, there were two approaches were used to validate the framework. The first approach used is the CMMI-Dev1.2 KPAs questionnaires as the main item to validate the suitability of the modified framework for SSDFs by 30 Jordanian professional developers and managers who were working in these firms. As a result of the first approach, the modified framework was suitable for these firms, where six KPAs were strongly suitable which are: requirement management, project planning, project monitoring and control, technical solution, verification, and validation, while the results of the last fifteen areas were suitable for these firms which are: supplier agreement management, measurement and analysis, process and

product quality assurance, configuration management, requirements development, product integration, organizational process focus, organizational process definition +IPPD, organizational training, integrated project management +IPPD, risk management, decision analysis and resolution, organizational process performance, quantitative project management, and causal analysis and resolution.

As resulted in the first approach, it can be concluded that all the components in the modified framework that aimed to achieve the requirements of the specific goals for the suitable CMMI-Dev1.2 KPAs are strongly suitable or suitable for these firms. Accordingly, this framework is suitable for the characteristics of SSDFs.

In the second validation approach, two Jordanian SSDFs implemented the modified framework as case studies as discussed in Sections 6.3.1 and 6.3.2. Then, the general evaluation criteria were used to evaluate the applicability and effectiveness of the modified framework by these firms, which are: satisfaction, task support satisfaction, and interface satisfaction. As a result of the evaluation process, it was found that the modified framework was effective when implemented in SSDFs as discussed in Section 6.4. Therefore, it can be concluded that the framework was suitable for SSDFs in improving the software development processes.

CHAPTER SEVEN
CONCLUSION AND FUTURE WORK

7.1 Introduction

This chapter concludes this thesis by presenting a brief summary of achievement of the research objectives carried out to support the thesis proposition. It discusses the contributions and the limitations of the research. The chapter ends by proposing directions for future work in the area of software development and improvement processes.

7.2 Achievement of the Research Objectives

The main aim of this research is to construct a software development process improvement framework for SSDFs, based on integrating XP as a software development method and CMMI-Dev1.2 as a SPI model. This is because the traditional SPI models and standards cannot be implemented directly by SSDFs, as these models and standards were developed for large and very large firms. The research was carried out in four stages to achieve the four objectives of the research. A summary of the research and key findings found at each stage are provided in the following sections.

7.2.1 Stage One: Aligning XP Practices to the Specific Goals of CMMI-Dev1.2 KPAs

This aim of this stage was to identify the coverage ratio of XP method to CMMI-Dev1.2. This aim was achieved by aligning XP practices to the specific goals of

CMMI-Dev1.2 KPAs, taking into account the achievement of the specific practices of each specific goal by the same or different way of CMMI-Dev1.2.

As a result of this stage, most of the specific goals of CMMI-Dev1.2 KPAs were supported by XP practices. Twelve of these KPAs were largely supported by XP practices, i.e., project planning, project monitoring and control, configuration management, technical solution, product integration, verification, validation, integrated project management +IPPD, risk management, decision analysis and resolution, quantitative project management, and causal analysis and resolution. Eight KPAs were partially supported by XP practices, i.e., requirement management; measurement and analysis, process and product quality assurance, requirements development, organizational process definition +IPPD, organizational training, organizational process performance, and organizational innovation and deployment. The last two KPAs are not-supported by XP practices and these are supplier agreement management and organizational process focus. The partially and not-supported KPAs were entered as inputs in Stage Two to develop the proposed software development process improvement framework for SSDFs.

7.2.2 Stage Two: Developing the Proposed Software Development Process Improvement Framework for SSDFs

Based on the partially and not-supported CMMI-Dev1.2 KPAs of Stage One, the EBA was adapted to extend the XP method. In this respect, the related previous literatures, and the required software development, management, and improvement

additions, were analyzed to cover these KPAs, as explained in Sections 4.3.1.1 and 4.3.1.2.

Referring to the phases of the generic and popular software development methodologies, such as Waterfall, Spiral, Incremental, Prototyping, and XP method, the required development, management, and improvement additions were distributed between the phases of these generic methods based on the suitable use of these additions during the development lifecycle. Accordingly, the comprehensive phases of the proposed Extended-XP method were extracted. Based on this, several modifications were made to the XP method phases and roles as discussed in Section 4.3.2.

Subsequently, the proposed Extended-XP method, CMMI-Dev1.2, and the generic elements of SPI framework, were merged to produce the proposed software development process improvement framework, and this was done by integrating the proposed Extended-XP method and CMMI-Dev1.2 to the generic elements of SPI framework as explained in Section 4.4.1. Then, the proposed software development process improvement framework was developed as discussed in Section 4.4.2. Then, the proposed framework was used as input in Stage Three.

7.2.3 Stage Three: Verifying the Proposed Software Development Process Improvement Framework

As a result of Stage Two, the proposed framework was developed. Accordingly, Stage Three aimed to: verify the compatibility of the proposed framework to the

suitable CMMI-Dev1.2 KPAs, verify the commitment of the proposed Extended-XP method to the agility values of XP method to ensure that the proposed method still kept lightweight values that are suitable for SSDFs, verify the suitability of the proposed framework and the proposed Extended-XP structures for the software development process improvement issues in SSDFs, and verify the suitability of the proposed framework and the proposed Extended-XP roles for their practices. In this research, the focus group coupled with Delphi technique was used as a verification method.

From the results of the verification process in this stage, this research found that the proposed framework is compatible for all the KPAs of CMMI-Dev1.2, except the organizational innovation and deployment KPA, because the related practices of this area conflicted with the agility of XP values, which made the Extended-XP method as a heavyweight, which is not suitable for SSDFs. In addition, based on the suggestions of the focus group members, several modifications were made to the proposed framework, including the proposed Extended-XP method, as presented in Sections 5.6 and 5.7.

7.2.4 Stage Four: Validating the Modified Software Development Process Improvement Framework for SSDFs

In this stage, two approaches were used to validate the suitability and applicability of the modified software development process improvement framework for SSDFs. These approaches are:

- Using CMMI-Dev1.2 questionnaires to validate the suitability of the modified framework for SSDFs.

  This validation approach aimed to validate the suitability of the modified framework for SSDFs. This was done by using CMMI-Dev1.2 KPAs as the main items of the validation questionnaires. Thirty Jordanian professional developers and managers, who were working in SSDFs, participated in achieving this process. Based on the results of this process, it was found that the framework was suitable for these SSDFs, where six KPAs were strongly suitable, i.e., requirement management, project planning, project monitoring and control, technical solution, verification, and validation. The next fifteen KPAs were suitable for these firms as follows: supplier agreement management, measurement and analysis, process and product quality assurance, configuration management, requirements development, product integration, organizational process focus, organizational process definition +IPPD, organizational training, integrated project management +IPPD, risk management, decision analysis and resolution, organizational process performance, quantitative project management, and casual analysis and resolution.

- Two case studies were conducted in order to validate the applicability and effectiveness of implementing the modified framework for SSDFs.

  This validation approach aimed to validate the applicability the modified framework for SSDFs. This was done by implementing the framework in

two Jordanian SSDFs. Then, this research used common evaluation criteria to evaluate the effectiveness of the modified framework by these firms. These criteria are gain satisfaction, interface satisfaction, and task support satisfaction.

Referring to the results of the first approach of validating the modified framework in this stage, all the software, development, and improvement practices that are used in developing the modified framework to achieve the twenty one KPAs of CMMI-Dev1.2 were suitable for SSDFs. Furthermore, the second validation approach with the results of the evaluation criteria indicates good applicability and effectiveness when using this framework in improving the software development processes by these firms.

7.3 Research Contributions

This research has demonstrated that the traditional SPI models and standards, such as CMMI-Dev1.2 can be implemented by SSDFs. This can be done by integrating this model with a suitable software development method, such as the XP method. In doing this, four major contributions are achieved. Sections 7.3.1 to 7.3.3 discuss these contributions.

7.3.1 Software Development Process Improvement Framework for SSDFs

The software development process improvement framework constructed in this study enables the SSDFs to improve their software development processes in a systematic

way. The framework was developed by integrating the XP method with CMMI-Dev1.2.

In this regard, this study provides evidence in support of a possible integration between CMMI-Dev1.2 and XP method through the usage of the XP practices as main items in achieving the specific goals of CMMI-Dev1.2 KPAs. Thus, this research promotes an understanding of how to use the SPI model (CMMI-Dev1.2) and software development method (XP method) together in order to improve the software development processes of SSDFs.

The framework is compatible and suitable for all KPAs of CMMI-Dev1.2, except the "*organization innovation and deployment"* KPA. Therefore, the framework can be used to support SSDFs in achieving high levels in CMMI-Dev1.2 certification. Additionally, the agility values of the developed framework such as simplicity, communication, feedback, and courage will increase the motivation of SSDFs to improve their software development activities.

7.3.2 Elicit Better Understanding of How to Construct the Framework

A better understanding for constructing the software development process improvement framework now exists and serves as a guideline for future development in the specific areas addressed in this study, which are: (1) the process of alignment XP method to CMMI-Dev1.2 KPAs, which is based on the XP practices and the specific goals of the CMMI-Dev1.2 KPAs. This further supports the need for increased attention to be given to the improvement of software development

processes by CMMI-Dev1.2 model and XP method. In addition, the results of this alignment have a straightforward and simple guideline to identify suitable development improvement processes for firms of all sizes; (2) the processes of adapting EBA to extend XP method, and the processes of integrating the Extended-XP method and XP method by modifying the generic elements of the SPI framework; (3) the process of using the focus group method coupled with Delphi technique to verify the proposed framework; and (3) the process of documenting the results of the case studies, and the process of conducting the evaluation criteria to evaluate the modified framework.

7.3.3 Quality Improvement of the Software Development Processes

The framework assists the process of educating and training the employees in the firm. This process contributes to increase the right understanding of the employees during the software development lifecycle by identifying employee roles specifically, and training them on the best way to achieve the goals of these roles.

In addition, the distributions of the roles in the framework are based on the experiences of each member; therefore this process enables the project teams to be very familiar with the current roles, which lead them to execute their roles easily. Furthermore, the framework documentations enable the project team to understand and implement the software development and management practices correctly and effectively during the development lifecycle without further inquiries and this will help the SSDFs to deliver the software products within limited time.

7.4 Limitations of the Research

Despite the noteworthy results obtained, this study has some limitations, as with any study. Sections 7.4.1 to 7.4.3 present these limitations.

7.4.1 Lack of the Related Researches

There is lack of researches that align the software development methods to CMMI-Dev1.2. Therefore, it was a challenge to align XP practices to the specific goal of CMMI-Dev1.2 KPAs, based on achievement of the specific practices of each KPA. In this regard, many related publications on CMMs were utilized in this study in order to carry out this alignment.

In addition, the related studies did not show how XP method can be extended and integrated to fulfill the KPAs of CMMs. Therefore, it was difficult to search for literature on extending the XP method to cover the missing KPAs of CMMI-Dev1.2 for suitable activities for SSDFs, as these firms need to have lightweight processes in their development processes. Due to these obstacles, some of the KPAs specific goals were supported by the framework without following the specific practices of each specific goal. Therefore, future research can be continued to address the missing specific practices of these KPAs.

7.4.2 The Framework is based on XP method and CMMI-Dev1.2

In this study, XP method has been used as a generic element in the software development process improvement framework, as this method is the most popular and effective lightweight development method of software development in SSDFs.

In addition, this method is more compatible to SPI models such as CMMI compared to other popular lightweight methods. Accordingly, for future research, it is advisable to examine more agile method practices to improve the software development process improvement framework in order to thoroughly address the missing specific goals of CMMI-Dev1.2.

In addition, CMMI-Dev1.2 model has been chosen as a generic element in the framework, as this model is the most comprehensive SPI and more fully complies with relevant traditional SPI models and standards. CMMI-Dev1.2 version was used as main SPI model to develop the framework, as this version was the newest version during the development of the framework and it was evaluated in other relevant studies. In this regard, the framework can be improved in future research by using the newest version of CMMI.

7.4.3 Limited Scope in the Verification and Validation Processes

During the verification process, the focus group comprising ten members was held. Three of the members had worked in CMMI or XP method fields in Jordan previously, and seven of the members were employed at a Jordanian SSDFs. Accordingly, the verification process was carried out based on the characteristics of a limited number of Jordanian SSDFs. In future research, it would be preferential to include experts from other countries in order to assess the comprehensiveness of the research results.

In addition, the modified framework was validated in two Jordanian SSDFs because time and cost were obstacles to implement the modified framework in other countries. Therefore, the implementation of this framework in other countries is important in order to assess the suitability of the framework for more SSDFs around the world. Furthermore, two case studies are too few to fully validate the effectiveness of the modified framework and therefore, a larger number of case studies are necessary to further evaluate the modified framework.

7.5 Future Work

The software development process improvement framework presented in this study is a solid starting point for working towards collaboration between the SPI models and software development methods. During the course of the research, several potential directions for future investigation were identified. Some of these are to overcome the current limitations of this study. Sections 7.5.1 to 7.5.3 highlight the potential directions for future work.

7.5.1 Fulfilling the Missing KPAs and Specific Practices of Several KPAs

The software development process improvement framework supports the specific goals of six KPAs of CMMI-Dev1.2 without following their specific practices. These KPAs are organizational process focus, risk management, decision analysis and resolution, organizational process performance, quantitative project management, and causal analysis and resolution. Therefore, future research can address the missing specific practices of these KPAs. In this respect, the software development, management, and improvement additions made during the

development of the framework can be used as a guideline and starting point for future research.

Furthermore, short term future research can continue to focus on finding a suitable method to fulfill the organizational innovation and deployment KPA by searching for suitable additions to achieve the goals of this area, as this KPA was not supported by the framework of this study.

7.5.2 Using other Agile Practices and CMMI-Dev1.3

The development of the software development process improvement framework was based on CMMI-Dev1.2 as a SPI model and XP method as a software development method. In this regard, there is possible avenue for further research to examine the agile method practices beyond the XP practices that were used in this study. Methods such as SCRUM, DSDM, LSD, and AUP (RUP) are all effective methods that could be used by SSDFs. In addition, Sidky (2011) pointed out the importance of the inclusion of agile practices as much as possible in the field of development processes, stating that this is especially important for SSDFs. Therefore, the combination of some agile methods will offer further development and improvement additions that can be used to fulfill the missing specific goals of several KPAs.

Now, CMMI-Dev1.3 is the newest version of CMMI. Even though that the KPAs of CMMI-Dev1.2 are similar to the KPAs of CMMI-Dev1.3, future research can be continued to improve the software development process improvement framework based on the newest version of CMMI, which is CMMI-Dev1.3.

7.5.3 Conducting More Case Studies

Jordanian SSDFs were the focus area for the validation of the modified framework. Therefore, future research could be continued to validate this framework by SSDFs in other countries to ensure that the developed framework is suitable to be implemented in most countries. Expanding the scope of validation will indicate subsequent need for additional modifications of the developed framework, and this will result in a more comprehensive framework that can be applied in SSDFs all over the world.

7.6 Final Remarks

This research started from the need to have a suitable software development process improvement framework for SSDFs. These firms suffer from problems during the development of software products. This is because they develop their software products in a chaotic and "ad hoc" manner since they are unaware of the basic software best practices.

In addition, all traditional SPI models were developed for large and very large firms. Therefore, SSDFs could not afford these models; these models need a lot of activities and requirements, which are not commensurate with the characteristics of SSDFs. Furthermore, most of these firms have a lack of understanding of the success factors of SPI and do not have sufficient staff to perform all the SPI activities. Therefore, SSDFs need integration between a suitable software development method and an appropriate SPI model to manage and improve their software development processes in a systemic way. This was the focus of this

research and the work reported in this thesis has successfully established the software development process improvement framework for SSDFs by integrating XP method with CMMI-Dev1.2.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. (2002). *Agile software development methods.* Espoo: VTT Publications 478, Technical Research Centre of Finland. Finland.

Agarwal, R., Umphress, D. (2008). Extreme Programming for a Single Person Team. *In Proceeding of the 46th Annual Southeast Regional Conference held on 28-29 March 2008 at Aubrun, Al, USA* (pp. 82-87). New York, USA: ACM.

Al Hussaini, A. M. (2006). *Web Engineering*. Unpublished Research Course. College of Computer and Information Sciences, King Saud University, kingdom of Saudi Arabia.

AL-Allaf, O. (2008). *A proposed hybrid web engineering process model for large-scale web-based application development in large web development enterprises methodology.* Unpublished Doctoral Thesis, Faculty of Information System and Technology, the Arab Academy of Banking and Financial Science, Amman, Jordan.

Alegra, J., & Bastarrica, M. (2006). Implementing CMMI using a Combination of Agile Methods. *Clei Electronic Journal, 9*(1), 7-22.

Alexandre, S., A. Renault, et al. (2006). OWPL: A Gradual Approach for Software Process Improvement In SMEs. *In Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA'06) held on 29 Aug. - 1 Sept. 2006 at Cavtat/Dubrovnik, Croatia* (pp. 328-335). Croatia: IEEE 2006.

Ali, R.Z.R.M., & Ibrahim, S. (2010). An iSPA model evaluation based on critical success factors and selected criteria to support Malaysia's SME environment. *In Proceeding of 2nd International Conference on Software Engineering and Data Mining (SEDM) held on 23-25 June 2010 at Chengdu, China* (pp.225-230). IEEE.

Alite, B., Spasibenko, N. (2008). *Project Suitability for Agile methodologies*. Unpublished Master Thesis, Umea University, Umeå School of Business, Sweden.

Allen, P., Ramachandran, M., & Abushama, H. (2003).PRISMS: an approach to software process improvement for small to medium enterprises. *In Proceedings of Third International Conference on Quality Software (QSIC'03) held on 6-7 November 2003 at Dallas, TX, USA* (pp. 211-214). USA: IEEE Computer Society 2003.

Alshammari, F. H., Ahmad, R. (2010). The effect of geographical region on the duration of CMMI-based software process improvement initiatives: An empirical study. *In Proceeding of the 2nd International Conference on Software Technology and Engineering (ICSTE) held on 3-5 October 2010 at San Juan, Puerto Rico, USA. Vol. 2,* (pp. V2-97-V2-100). USA: IEEE 2010.

Altarawneh, H., Amro, S. (2008). Software Process Improvement In Small Jordanian Software Development Firms. *In Proceedings of the 7th International Conference on Perspectives in Business Informatics Research (BIR'2008) held on 25-26 Sept. 2008 at the University of Gdańsk, Gdansk, Poland* (pp 175-189). Poland: University of Gdańsk.

Altarawneh, H., El Shiekh, A. (2008). A Theoretical Agile Process Framework for Web Applications Development in Small Software Firms. *In Proceeding of the Sixth International Conference on Software Engineering Research, Management and Applications held on 20-22 Aug. 2008 at the Charles University, Prague, Czech Republic* (pp125-132). Czech Republic: IEEE Computer Society**.**

Anacleto, A., Von Wangenheim, C., Salviano, C., & Savi, R. (2004). A method for process assessment in small software companies. *In Proceeding of the 4th international SPICE conference on process assessment and improvement held on 27-29 April 2004 at the Estoril Congress Centre, Lespon, Portugal* (pp. 69-76). Portugal: Estoril Congress Centre.

Anderson, D. J. (2005). Stretching Agile to fit CMMI Level 3-the story of creating MSF for CMMI Process Improvement at Microsoft Corporation. *In Proceeding of the Agile Development Conference (ADC'05) held on 24-29 July 2005 at Denver, CO, USA* (pp. 193-201). USA: IEEE Computer Society.

Baddoo, N., & Hall, T. (2002). Motivators of Software Process Improvement: an analysis of practitioners' views. *Journal of Systems and Software, 62*(2), 85-96.

BAe, D. (2007). *Panel: Software Process Improvement for Small Organizations*. Paper presented at 31st Annual International Computer Software and Applications Conference, Beijing, China.

Baharom, F., Deraman, A., & Hamdan, A. (2006). A Survey on the current practices of software development process in Malaysia. *Journal of ICT, 4*, 57-76.

Baird, S. (2002). *Sams teach yourself extreme programming in 24 hours*. USA: Sams Publishing.

Bajec, M., Vavpotia, D., & Krisper, M. (2007). Practice-driven approach for creating project-specific software development methods. *Information and Software Technology, 49*(4), 345-365.

Baker, S. (2005). Formalizing agility: an agile organization's journey toward CMMI accreditation. *In Proceeding of the Agile Development Conference (ADC'05) held on 24-29 July 2005 at Denver, CO, USA* (pp. 185-192). USA: IEEE Computer Society 2005.

Baker, S. W., & Thomas, J. C. (2007). Agile principles as a leadership value system: How agile memes survive and thrive in a corporate it culture. *In Proceeding of Agile Conference (AGILE'07) held on 13-17 Aug. 2007 at Washington D.C.* (pp. 415-420). IEEE Computer society.

Balandis, O., & Laurinskaite, L. (2005). Software Process Improvement in Lithuania-UAB Sintagma Case Study. *Information Technology and Control, 34*(2A), 195-201.

Balkanski, P. (2003). QUALITY ASSURANCE IN EXTREME PROGRAMMING. *International Journal of Information Theories & Applications. 1*(1), 113-117.

Baruah, A. (2012a). Contribution of Software Process Improvement Approaches For Small and Medium Scale Enterprises. *International Journal of Computing and Corporate Research, 2*(2), 1-10.

Baruah, N. (2012b). *Software Process Improvement (SPI) Expert In Small And Medium Scale Enterprises (SMEs).* Unpublished Master Thesis, Computer Science and Engineering Department, Thapar University, Patiala, India.

Basri, S., & O'Connor, R. V. (2011). Knowledge Management in Software Process Improvement: A case study of very small entities. In M. Ramachandran (Eds.) *Knowledge Engineering for Software Development Life Cycle: Support Technologies and Applications*. (pp. 273-288). PA, Hershey, USA: IGI Global.

Baxter, S. M., Day, S. W., Fetrow, J. S., & Reisinger, S. J. (2006). Scientific software development is not an oxymoron. *PLoS Computational Biology, 2*(9), e87, 975-978.

Beck, k. (2000). *Extreme programming explained: Embrace change* (3rd ed.). Reading, Mass. Boston: addition-Wesley.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). *Manifesto for agile software development*. Retrieved on 13 May 2009, from http://agilemanifesto.org/.

Beitz, A., El-Emam, Kh., Jarvinen, J. (1999). A Business Focus to Assessments. *In Proceeding of the European Conference on Software Process Improvement (SPI'99) held on 30 Nov. -3 Dec. 1999 at Barcelona, Spain* (pp. 1-6). ACM New York, NY, USA.

Bell, D. (2001). *Software Engineering, A programming Approach* (3rd ed.). New York: Addison Wesley.

Bidad, C. D., & Campiseno, E. R. (2010). Community Extension Services Of Sucs In Region Ix: Basis For A Sustainable Community Enhancement Program. *International Scientific Research Journal, 2*(3), 235-343.

Billinger, K. (2005). A focus group investigation of care provider perspectives in Swedish institutions for the coercive care of substance abusers. *International Journal of Social Welfare, 14*(1), 55-64.

Birisci. S, Mentin, M. and Karakas, M. (2009). Prospective Elementary Teacher's Attitudes Toward computer and internet use: A Sample from Turkey. *World Applied Sciences Journal, 6*(10), 1433-1440.

Block, E., (1986). The comprehension strategies of second language readers. *TESOL Quarterly, 20* (3), 463-494.

Boas, G. V., da Rocha, A. R. C., & Pecegueiro do Amaral, M. (2010). An Approach to Implement Software Process Improvement in Small and Mid Sized Organizations. *In Proceeding of the Seventh International Conference on the Quality of Information and Communications Technology held on 29 Sep - 2 Oct. 2010 at Porto, Portugal* (pp. 447-452). Portugal: IEEE Computer Society 2010.

Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. *IEEE Computer, 21*(5), 61-72.

Boehm, B. (2006). A view of 20th and 21st century software engineering. *In Proceeding of the 28th international conference on Software Engineering held on 20-28 May 2006 at Keynote Talks Shanghai, China* (pp. 12-29). USA: ACM.

Boehm, B., & Turner, R. (2003). *Balancing agility and discipline: A guide for the perplexed*. Boston: AddisonYWesley.

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The Unified Modeling Language User Guide.* UK: Addison-Welsley Longman Inc.

Bos, E., & Vriens, C. (2004). *An agile CMM. Extreme Programming and Agile Methods-XP/Agile Universe, 1*(1), 129-138.

Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology, 38*(4), 275-280.

Brown, N. (1999). High-Leverage Best Practices: What Hot Companies Are Doing to Stay Ahead. *Cutter IT Journal, 12*(9), 4-9.

Bucci, G., Campanai, M. and Cignoni, G. A. (2001). Rapid Assessment to Solicit Process Improvement in Small and Medium-Sized Organizations. *Software Quality Professional, 4* (1), 33-41.

Bush, M., & Dunaway, D. (2005). *CMMI (R) Assessments: Motivating Positive Change (Sei Series in Software Engineering).* Murray, KY, U.S.A: Addison-Wesley Professional.

Calvo-Manzano Villalan, J. A., Cuevas AgustÃn, G., San Feliu Gilabert, T., De Amescua Seco, A., GarcÃa SÃ¡nchez, L., & PÃ©rez Cota, M. (2002). Experiences in the application of software process improvement in SMES. *Software Quality Journal, 10*(3), 261-273.

Carter-Steel, A. (2001). Process Improvement in Four Small Companies. *In Proceeding of the 13th Australian Software Engineering Conference (ASWEC'01) held on 27-28 Aug. 2001 at Canberra, Australia* (pp. 262-272). Los Alamitos, California, Washington, Tokyo: IEEE Computer Society.

Cater-Steel, A. (2002). Process capability assessments in small development firms. *In Proceedings of IASTED 6th International Conference Software Engineering and Applications held on 4-6 Nov. 2002 at Cambridge, Massachusetts, USA* (pp. 737-42). ACTA Press: Anaheim, CA, USA.

Cater-Steel, A. (2004a). *An Evaluation Of Software Development Practice And Assessment-Based Process Improvement In Small Software Development Firms*. Unpublished doctoral thesis, School of Computing and Information Technology, Faculty of Engineering and Information Technology, Griffith University, Australia.

Cater-Steel, A. (2004b). Low-rigour, rapid software process assessments for small software development firms. *In Proceeding of the 2004 Australian Software Engineering Conference (ASWEC'04) held on 13-16 April 2004 at Melbourne, Australia* (pp. 368-377). Washington, DC, USA: IEEE Computer Society.

Cepeda, S., Garcia, S., & Langhout, J. (2008). Is CMMI Useful and Usable in Small Settings? One Example. *The Journal of Defense Software Engineering, 21*(2), 14-18.

Chrissis, M., Konrad, M., & Shrum, S. (2003). *CMMI Guidelines for Process Integration and Product Improvement*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Cignoni, G. A. (1999). Rapid Software Process Assessment to promote Innovation in SMEs. *In Proceeding of the European Software Day (EUROMICRO'99) held on 8-10, Sept. 1999 at Milan, Italy* (pp. 1-14). DC, USA: IEEE Computer Society.

Clarke, P., & O'Connor, R. (2011). The influence of SPI on business success in software SMEs: An empirical study. *Journal of Systems and Software, 1*(1).1-28.

CMMI Product Team (2002). *Capability Maturity Model® Integration (CMMI), Version 1.1: CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing, CMMI-SE/SW/IPPD/SS, V1.1,* Carnegie Mellon University Software Engineering Institute, Pittsburgh PA,USA.

CMMI Product Team (2006). *CMMI for Development, version 1.2. Preface (CMU/SEI-2006-TR-008).* Software Engineering Institute, Carnegie Mellon University, USA.

CMMI Product Team. (2010). *CMMI® for Development, Version 1.3 (CMMI-DEV, V1.3), Improving processes for developing better products and services, TECHNICAL REPORT, CMU/SEI-2010-TR-033, ESC-TR-2010-033.*Carnegie Mellon University Software Engineering Institute, Pittsburgh PA, USA.

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer, 34*(11), 131-133.

Cohen, D., Lindvall, M., & Costa, P. (2004). An introduction to agile methods. *Advances in Computers, 62*, 1-66.

Coram, M. and Bohner, S. (2005). The Impact of Agile Methods on Software Project Management. *In Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer Based Systems (ECBS'05) held on 4-7 April 2005 at Greenbelt, MD,* USA (pp. 363- 370). IEEE Computer Society.

Cruz Mendoza, R., Morales Trujillo, M., Morgado, C., Oktaba, H., Ibarguengoitia, G., Pino, F. J., et al. (2009). Supporting the software process improvement in very small entities through e-learning: the HEPALE! Project. *In Proceeding of the Mexican International Conference on Computer Science (ENC 2009) held on 21-25 Sept. 2009 at UNAM, Mexico City, Mexico* (pp. 221-231). DC, USA: IEEE Computer Society.

Da Rocha, A., Montoni, M., Weber, K., & de Araujo, E. (2007). A Nationwide Program for Software Process Improvement in Brazil. *In Proceeding of the Sixth International Conference on the Quality of Information and*

Communications Technology (QUATIC'2007) held on 12-14 Sept. 2007 at Lisbon New University, Portugal (pp. 449-460). Los Alamitos, CA, USA: IEEE Computer Society.

Dagnino, A., Cordes, A., & Smiley, K. (2009). *Adapting rapidly to change using the IDEAL improvement model.* ABB Corporate Research Raleigh, NC, USA. Retrieved on 4 May 2009, from http://library.abb.com/global/scot/scot271.nsf/0cb8394a97bc4979c1256c6b004c4f2e/4735e42889ef82bfc12575e4004bbbb0/$FILE/58-%202M974ENG72dpi.pdf.

Dalkey, N. C., & Helmer, O. (1963). An experimental application of the Delphi method to the use of experts. *Management Science, 9* (3), 458-467.

Davis, A. M. (1993). *Software Requirements: Objects, Functions, States*. New Jersey, USA: Prentice-Hall.

Deep, A. (2012). An Empirical Study of Agile Software Development. *International Journal of Latest Trends in Engineering and Technology (IJLTET), 1*(1), 35-40.

Devesh, K. S., Durg, S. C., & Raghuraj, S. (2011). Square Model-A Proposed Software Process Model for BPO based Software Applications. *International Journal of Computer Applications, 13*(7), 33-36.

Diez, D., Fernandez, C., Dodero, J., Diaz, P., & Aedo, I. (2007). Instructional Software Analysis: Lessons from Software Development Process Improvement. *In Proceeding of the Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007) held on 18-20 July 2007 at Niigata, Japan* (pp. 499-501). Los Alamitos, California, Washington, Tokyo: IEEE Computer Society.

Dyba, T., & Dingsøyr. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50*(9-10), 833-859.

Easterby-Smith M. and Thorpe R. and Lowe A. (1991). *Management Research: An Introduction,* London: *Sage* Publications Ltd.

El Emam, K. & Briand, L. (1997). *Costs and Benefits of Software Process Improvement.* Technical Report ISERN 97-12, Fraunhofer Institute for Experimental Software Engineering, Germany.

El Emam, K., Melo, W., & Drouin, J. (1999). *SPICE: The theory and practice of software process improvement and capability determination*. Press Los Alamitos, CA, USA: IEEE Computer Society.

El Sheikh, A., & Tarawneh, H. (2007). A survey of web engineering practice in small Jordanian web development firms. *In Proceeding of the seventh European software engineering conference and the ACM SIGSOFT (ESEC/FSE'07) and the ACM SIG-SOFT International Symposium on Foundations of Software Engineering (ESEC/SIGSOFT FSE) held on 3-7 Sept. 2007 at Cavtat near Dubrovnik. Dubrovnik, Croatia* (pp. 481-490). New York, USA: ACM.

Elshafey, L. A., & Galal-Edeen, G. (2008). Combining CMMI and Agile Methods. *In Proceeding of the 6th International Conference on Informatics and Systems (INFOS2008) held on 27 - 28 March 2008 at Faculty of Computers and Informatics, Cairo University, Egypt* (pp. SE-27- SE-39). Egypt: Cairo University Press.

Erharuyi, E. (2007). *Combining eXtreme Programming with ISO 9000: 2000 to Improve Nigerian Software Development Processes.* Unpublished master thesis, School of Engineering, Blekinge Institute of Technology, Sweden.

Evans, M. (2001). SPMN director identifies 16 critical software practices. *CrossTalk, The Journal of Defense Software* Engineering. *March, 2001*. 27-31.

Fayad, M., Laitinen, M., & Ward, R. (2000). Thinking objectively: software engineering in the small. *Communications of the ACM, 43*(3), 118.

Fernandeas, D. (2009). *Study on the correlation between CMMI and agile practices and their application in SMEs*. Unpublished Master Thesis, computer faculty, university Polytechnic of Madrid, Spain.

Fogle, S., Loulis, C., & Neuendorf, B. (2001). The benchmarking process: one team's experience. *Software, IEEE, 18*(5), 40-47.

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software Development, 9*(8), 28-35.

Fritzsche, M., & Keil, P. (2007). Agile Methods and CMMI: Compatibility or Conflict?. *E-Informatica Software Engineering Journal, 1*(1). 9-26.

Fruhling, A., & Vreede, G. (2006). Field experiences with eXtreme programming: Developing an emergency response system. *Journal of Management Information Systems, 22*(4), 39-68.

Galinac, T. (2008). Analysis of Quality Management In Modern European Software Development. *Electronic form only: NE Eng. Rev, 28*(2), 65-76.

Garcia, I., Pacheco, C., & Andrade, G. (2010b). Applying the Psychometric Theory to Questionnaire-Based Appraisals for Software Process Improvement. *In*

Proceeding of the Eighth ACIS International Conference on Software Engineering Research, Management and Applications Montreal held on 24-26 May 2010 at Concordia University & l'École de technologie supérieure (ETS) Montreal, Canada (pp. 198-204). Los Alamitos, CA, USA: IEEE Computer Society.

Garcia, I., Pacheco, C., & Calvo-Manzano, J. (2010a). Using a web-based tool to define and implement software process improvement initiatives in a small industrial setting. *Software, IET, 4*(4), 237-251.

Garrity, E.J., and Sanders, G.L. (1998). *Information Systems Success Measurement*. Hershey, USA: Idea Group Publishing.

Gerami, M., & Ramsin, R. (2011). A framework for extending agile methodologies with aspect-oriented features. *In Proceeding of the Fifth International Conference on the Research Challenges in Information Science (RCIS) held on 19-21 May 2011 at Guadeloupe-French West Indies, France* (pp. 1-6). IEEE.

Glass, R. (1995). *Software Creativity*. Englewood Cliffs, NJ.USA: Prentice Hall.

Goldenson, DR & Gibson, D. (2003). *Demonstrating the impact and benefits of CMMI: an update and preliminary results* (CMU/SEI-2003-SR-009). Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA.

Gruner, S., & Zyl, V. J. (2011). Software testing in small IT companies: a (not only) South African problem. *South African Computer Journal, 47*, 7-32.

Guerrero, F., & Eterovic, Y. (2004). Adopting the SW-CMM in a Small IT Organization. *IEEE software*, *21*(4), 29-35.

Guha, P., Shah, K., Shukla, S. S. P., & Singh, S. (2011). Incorporating Agile with MDA Case Study: Online Polling System. *International Journal of Software Engineering & Applications (IJSEA), 2*(4). 83-96.

Haase, V. (1996). Software process assessment concepts. *Journal of Systems Architecture*, *42*(8), 621-631.

Habib, Z. (2009*). The Critical Success Factors in implementation of Software Process Improvement Efforts*. Unpublished Master Thesis, University of Gothenburg, Department of Applied Information Technology Gothenburg, Sweden.

Habra, N., Alexandre, S., Desharnais, J. M., Laporte, C. Y., & Renault, A. (2008). Initiating software process improvement in very small enterprises:

Experience with a light assessment tool. *Information and Software Technology, 50*(7-8), 763-771.

Habra, N., Niyitugabira, E., Lamblin, A., Renault, A. (1999). Software Process Improvement in Small Organizations Using Gradual Evaluation Schema. *In Proceeding of the International Conference on Product Focused Software Process Improvement (PROFES'99) held on 22-24 June 1999 at the Universit. Oulu, Oulu, Finland* (pp. 381-396). Finland: valtion teknillinen tutkimuskeskus (VTT).

Hashmi, S., & Baik, J. (2007). Software Quality Assurance in XP and Spiral - A Comparative Study. *In Proceeding on the Fifth International Conference on Computational Science and Applications held on 26-29 Aug 2007 at University of Malaya, Kuala Lumpur, Malaysia* (PP.367-374). IEEE.

Hashmi, S., & Baik, J. (2008). Quantitative Process Improvement in XP Using Six Sigma Tools. *In Proceeding of the Seventh IEEE/ACIS International Conference on Computer and Information Science held on 14-16 May 2008 at Melbourne, Australia* (pp. 519-524). Washington, DC, USA: IEEE Computer Society.

Hauck, J. C. R., Gresse von Wangenheim, C., Souza, R. H., & Thiry, M. (2008). Process Reference Guides-Support for Improving Software Processes in Alignment with Reference Models and Standards. *In Proceeding of the 15th European Conference (EuroSPI 2008) held on 3-5 Sept. 2008 at Dublin, Ireland* (pp. 70-81). Springer 2008.

Hearty, P. (2008). *Modeling Agile Software Processes Using Bayesian Networks.* Unpublished Doctoral Thesis, Queen Mary, University of London, UK.

Highsmith, J., & Cockburn, A. (2004). Agile Software Development: The Business of Innovation. *IEEE Computer, 34*(1), 120–122.

Hightower, R. (2004). *Professional Java Tools for EXtreme Programming: Ant, XDoclet, Junit, Cactus, and Maven*. Hoboken, NJ, USA: John Wiley & Sons, Incorporated.

Hofer, C. (2002). Software development in Austria: results of an empirical study among small and very small enterprises. *In Proceeding of the 28th Euromicro Conference (EUROMICRO'02) held on 4-6 Sept. 2002 at Dortmund, Germany* (pp. 361-366). Los Alamitos, California: IEEE Computer Society.

Hneif, M., & Hock Ow, S. (2009). Review of Agile Methodologies in Software Development. *International Journal of Research and Reviews in Applied Sciences, 1*(1). 1-8.

Huang, W., Li, R., Maple, C., Yang, H., Foskett, D., & Cleaver, V. (2008). Web Application Development Lifecycle for Small Medium-Sized Enterprises (SMEs). *In Proceeding of the Eighth International Conference on the Quality Software (QSIC'08) held on 2-13 August 2008, Oxford, UK* (pp. 247-252). IEEE Computer Society.

Humphrey, W. S. (1993). *Introduction to software process impro*vement. Technical Report CMU/SEI-92-TR-7, Software Engineering Institute, Carnegie-Mellon University, USA.

Humphrey, W. S. (1998). Three Dimensions of Process Improvement. *The Journal of Defense Software Engineering, 14*, 39-72.

Humphrey, W. S. (2008). The software quality challenge. *Crosstalk The Journal of Defense Software Engineering, 21*(6), 4-10.

Humphrey, W. S., & Kellner, M. I. (1989). *Software process modeling: principles of entity process models. In Proceedings of the 11th International Conference on Software Engineering held on 15-18 May 1989 at Pittsburgh, Pennsylvania* (pp.331-342). IEEE Computer Press.

Ibrahim, S., Ali, R.Z.R.M. (2011). Study on acceptance of customised Software Process Improvement (SPI) model for Malaysia's SME. *In Proceeding of the 5th Malaysian Conference in Software Engineering (MySEC) held on13-14 Dec. 2011 at Johor Bahru, Malaysia* (pp.25-30). IEEE.

IEEE Std 730-1998. (1998). IEEE Standard for Software Quality Assurance Plans. IEEE.

Isawi, A. B. M. (2011). *Software Development Process Improvement for Small Palestinian Software Developme*nt. Unpublished Master Thesis, Faculty of Graduate Studies, An-Najah National University, Nablus, Palestine.

Jakobsen, C. R., & Johnson, K. A. (2008). Mature Agile with a Twist of CMMI. *In Proceeding of Agile 2008 Conference (AGILE'08) held on 4-8 Aug. 2008 at Toronto, Canada* (pp. 212-217). IEEE Computer Society.

Jantunen, S. (2010). Exploring software engineering practices in small and medium-sized organizations. *In Proceeding of the Cooperative and Human Aspects of Software Engineering (CHASE'10) held on 2 May 2010 at Cape Town, South Africa* (pp. 96-101). ACM.

Jeffries, R., Anderson, A., and Hendrickson, C. (2002). *Extreme Programming Installed.* Boston: Addison Wesley.

Johannesen, R. (2004). Software Engineering in the Small: Is Chaos Likely to Fall?. *Section of the September/October 2000 issue of IEEE Software,* Retrieved 20, June, 2009, from http://toalango.com/msc/in-the-small.pdf.

Jones, C. (1996). *Patterns of Software Systems Failure and Success*. London: International Thompson Computer Press.

Kähkönen, T. (2005). *Framework for Agile Software Development in Embedded Systems.* Agile Deliverable D.2.1. Version 1.0.Information Technology for European Advancement. ITEA.

Kähkönen, T., & Abrahamsson, P. (2004). Achieving CMMI level 2 with enhanced extreme programming approach. *In Proceeding of the 5th International Conference of Product Focused Software Process Improvement held on 5-8 April 2004 at Kansai Science City, Japan* (pp. 378-392). Berlin*:* Springer Berlin Heidelberg.

Kalpana, A., & Jeyakumar, A. E. (2011).Software Process Improvisation Framework Based On Fuzzy Logic Approach For Optimizing Indian Small Scale Software Organizations. *International Journal of Multimedia and Ubiquitous Engineering*, *6*(1), 29-42.

Karlstrom, D., & Runeson, P. (2006). Integrating agile software development into stage-gate managed product development. *Empirical Software Engineering, 11*(2), 203-225.

Kitchenham, B. (1998). "Evaluating software engineering methods and tool," *ACM SIGSOFT software engineering Notes, 23*(5), pp. 21-24.

Khalaf, S. & Al-Jedaiah, M. (2008). *Software Quality and Assurance in Waterfall Model and XP - A Comparative Study*. Retrieved 7 April 2011 http://www.wseas.us/e-library/transactions/computers/2008/31-097.pdf.

Koch, A. S. (2003). CMM-compliant XP. Retrieved on 20 Aug. 2009, from http://www.askprocess.com/Articles/CMM-XP.pdf.

Kontio, J., Lehtola, L., & Bragge, J. (2004). Using the focus group method in software engineering: obtaining practitioner and user experiences. *In Proceeding of the International Symposium on Empirical Software Engineering (ISESE'04) held on 19-20 August 2004, Redondo Beach, California, USA* (pp. 271-280). Los Alamitos, CA, USA. IEEE Computer Society.

Kothari, C. (1985). *Research Methodology: Methods and Techniques.* New Delhi: Wiley Eastern.

Koznov, D. (2011). Process Model of DSM Solution Development and Evolution for Small and Medium-Sized Software Companies. *In Proceeding of the 15th IEEE International of Enterprise Distributed Object Computing Conference Workshops (EDOCW) held on 29 Aug.-2 Sept., Helsinki, Finland* (pp.85-92). IEEE Computer Society.

Kroeger, T. (2005). *CMMI – Strengths, Weaknesses, and Guidelines for Use.* Presentation in EDS Australia South ADU Australian Organization for Quality (SA) – Software SIG. Australia.

Krueger, R, A., & Casey, M, A. (2000). *Focus Groups: A Practical Guide for Applied Research* (3rd ed.). Thousand Oaks, CA: Sage Publications.

Kuan, S. T., Wu, B. Y., & Lee, W. J. (2008). Finding friend groups in blogosphere. *In Proceeding of the 22nd International Conference on the Advanced Information Networking and Applications-Workshops (AINAW 2008) held on 25-28 March 2008 at GinoWan, Okinawa, Japan* (pp. 1046-1050). IEEE Computer Society, TCDP.

Kuhlmann, U. (2003). *Maintenance Activities in Software Process Models: Theory and Case Study Practice.* Unpublished *m*aster thesis. Faculty of Computer Sciences, Koblenz Landau Campus Koblenz Uni, Germany.

Kunda, S. (2001), *A social-technical approach to selecting software supporting COTS-Based systems*. Unpublished Doctoral Thesis, University of York, UK.

Kuvaja, P. (1995). BOOTSTRAP: A software process assessment and improvement methodology. *Objective Software Quality, 926*, 31-48.

Laporte, C., Desharnais, J., Abouelfattah, M., Bamba, J., Renault, A., & Habra, N. (2005). Initiating Software Process Improvement in Small Enterprises: Experiments with Micro-Evaluation Framework. *In Proceeding of the SWDC-REK International Conference on Software Development held on 27 May- 1June 2005 at University of Iceland, Reykjavik, Iceland* (pp. 153-163). Iceland: University of Iceland.

Larman, C. (2003). *Agile & Iterative Development: A Manager's Guide*. Boston: Addison Wesley.

Laudon, KC., & Laudon, JP. (2004*). Management Information Systems: Managing the Digital Firm*. Upper Saddle River, New Jersey: Prentice Hall.

Laugen, B.T., Acur, N., Boer, H., Frick, J. (2005). Best manufacturing practices. What do best-performing companies do?. *International Journal of Operations and Production Management, 25* (2), 131-150.

Lee, M., Lee, Y., Yoon, H., Song, S., & Cheong, S. (2008). Issues and Architecture for Supporting Data Warehouse Queries in Web Portals. *International Journal of Computer Science and Engineering, 1*(2). 110-115.

Lina, Z., & Dan, S. (2012). Research on Combining Scrum with CMMI in Small and Medium Organizations. *In Proceeding of the International Conference on Computer Science and Electronics Engineering (ICCSEE) held on 23-25 March 2012 at Hangzhou, China*. IEEE.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., et al. (2004). *Agile software development in large organizations. Computer, 37*(12), 26-34.

Loftus, C., &Ratcliffe, M. (2005). Extreme programming promotes extreme learning?". *In Proceeding of the 10th Annual Joint Conference Integrating Technology into Computer Science Education held on 27-29 June 2005 at Lisbon, Portugal* (pp. 311-315). New York, USA: ACM.

Ludwig, B. (1997). Predicting the future: Have you considered using the Delphi methodology?. *Journal of Extension, 35* (5), 1-4.

Makitalo-Keinonen, T., Virolainen, H., Laurell, J., Varkoi, T., & Makinen, T. (2011). Critical incidents in a growth path of a small software company. *In Proceeding of the Technology Management in the Energy Smart World (PICMET' 11) held on 31 July - 4 Aug. 2011 at Hilton Portland and Executive Tower Portland, Oregon, USA* (pp. 1-10). IEEE.

Martinsson, J. (2002). *Maturing Extreme Programming Through the CMM*. Unpublished Master Thesis, Department of Computer Science, Lund University, Lund, Sweden.

Mathiassen, L., Ngwenyama, O., & Aaen, I. (2005). Managing change in software process improvement. *IEEE software, 22*(6), 84-91.

Mazza, R., & Berre, A. (2007). Focus group methodology for evaluating information visualization techniques and tools. *In Proceeding of the 11th International Conference Information Visualization (IV'07) held on 4-6 July 2007 at Zurich, Switzerland* (pp. 74-80). Los Alamitos, CA, USA: IEEE Computer Society.

McDonald, A., & Welland, R. (2001). *A survey of web engineering in practice.* Technical Report: R-2001-79, Department of Computing Science, University of Glasgow, Scotland.

McFarlane, R., & Biktasheva, I. V. (2008). High Performance Computing for the Simulation of Cardiac Electrophysiology. *In Proceeding of the Third International Conference on the Software Engineering Advances (ICSEA'08)*

held on 26-31 October 2008 at Sliema, Malta (pp. 13-18). IEEE Computer Society.

Mehrfard, H., Pirzadeh, H., & Hamou-Lhadj, A. (2010). Investigating the Capability of Agile Processes to Support Life-Science Regulations: The Case of XP and FDA Regulations with a Focus on Human Factor Requirements. *Software Engineering Research, Management and Applications, Volume 296*, 2010, 241-255.

Mishra, D., & Mishra, A. (2009). Software process improvement in SMEs: A comparative view. *Computer Science and Information Systems, 6*(1), 111-140.

Mongkolnam, P., Silparcha, U., Waraporn, N., & Vanijja, V. (2009). A Push for Software Process Improvement in Thailand. *In Proceeding of the 16th Asia-Pacific Software Engineering Conference held on 1-3 Dec. 2009 at Penang, Malaysia*. (pp. 475-481). Los Alamitos, CA, USA: IEEE Computer Society.

Morgan D.L. (1997), *Focus groups as qualitative research* (2nd ed.). London: Sage publication.

Mnkandla, N. (2008). *A Selection Framework For Agile Methodology Practices: A Family of Methodologies Approach*. Doctoral thesis, Faculty of Engineering and the Built Environment, University of Witwatersrand, Johannesburg, South Africa.

Mushtaq, Z., & Qureshi, M. R. J. (2012). Novel Hybrid Model: Integrating Scrum and XP. *International Journal of Information Technology and Computer Science (IJITCS), 4*(6), 39.

Munassar, N. M. A., & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. *IJCSI International Journal of Computer Science Issues, 7*(5), 94-101.

Nawaz, A., & Malik, K. (2008). *Software Testing Process In Agile Development*. Unpublished Master Thesis, Comp Science Dept. School of Engineering, Blekinge Institute of Technology, Sweden.

Nawazish Khokhar, M., Zeshan, K., & Aamir, J. (2010). Literature review on the software process improvement factors in the small organizations. *In Proceeding of the 4th International Conference on New Trends in Information Science and Service Science (NISS) held on 11-13 May 2010 at Gyeongju, Korea* (pp. 592 – 598). Los Alamitos, CA, USA: IEEE Computer Society.

Nisa, S. U., & Qureshi, M. R. J. (2012) .Empirical Estimation of Hybrid Model: A Controlled Case Study. *International Journal of Information Technology and Computer Science (IJITCS), 4*(8), 43.

Oktaba, H., GarcÃa, F., Piattini, M., Ruiz, F., Pino, F. J., & Alquicira, C. (2007). Software process improvement: The Competisoft project. *Computer, 40*(10), 21-28.

Oktaba, H., Piattini.M. (2008). *Software Process Improvement for Small and Medium Enterprises: Techniques and Case Studies*, Illustrated Edition, ISBN 978-1-59904-906-9. New York: Idea Group Inc (IGI).

Omran, A. (2008). AGILE CMMI from SMEs perspective. *In Proceeding of the 3rd International Conference on Information & Communication Technologies: from Theory to Applications (ICTTA 2008) held on 7-11 April 2008 at Damascus, Syria* (pp. 1-8). Los Alamitos, CA, USA: IEEE Computer Society.

Palani, A., & Mohideen. P. (2012). Trends In Working Capital Management And Its Impact On SME With Reference To Manufacturing Firms. *South Asian Academic Research Journals, 2*(2), 123-141.

Paulk, M. (2001). Extreme Programming from a CMM Perspective. *IEEE Software, 18*(6), 19-26.

Paulk, M. (2002). Agile Methodologies and Process Discipline. *CrossTalk: The Journal of Defense Software Engineering, 15*(10), 15-18.

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). Capability maturity model, version 1.1. *Software, IEEE, 10*(4), 18-27.

Pettersson, F., Ivarsson, M., Gorschek, T., & Ohman, P. (2008). A practitioner's guide to light weight software process assessment and improvement planning. *Journal of Systems and Software, 81*(6), 972-995.

Phillips, M. (2003). *CMMI appraisal tutorial*. Paper presented to Australian Software Engineering Process Group (SEPG), Surfers Paradise, Carnegie Mellon University, USA.

Pikkarainen, M. (2008). *Towards a framework for improving software development process mediated with CMMI goals and agile practices.* Unpublished Academic Dissertation, Faculty of Science, Department of Information Processing Science. University of Oulu, Finland.

Poole, C., & Huisman, J. (2001). Using Extreme Programming in a Maintenance Environment. *IEEE Software 18*(6), 42–50.

Pourkomeylian, P. (2002). *Software Practice Improvement.* Unpublished doctoral dissertation, Department of Informatics Göteborg University Viktoriagatan, Göteborg. Sweden.

Powell, R, A., and Single, H, M. (1996). Focus groups. *International Journal of Quality in Health Care, 8* (5), 499-504.

Pressman, R. (2005). *Software Engineering: A Practitioner's Approach*, (6th ed.). New York, USA: McGraw-Hill Education.

Pressman, R. (2009). *Software Engineering: A Practitioner's Approach*. (7th ed.). New York, USA: McGraw-Hill Education.

Preuninger, R, D. (2006). *The advantages of implementing software engineering process models.* Unpublished master thesis, Faculty of the Graduate School, Texas At Arlington Uni, USA.

Pusatli, O. T., & Misra, S. (2011). A Discussion On Assuring Software Quality In Small And Medium Software Enterprises: An Empirical Investigation. *Technical Gazette, 18*(3), 447-452.

Qasaimeh, M., & Abran, A. (2010). Extending Extreme Programming User Stories to Meet ISO 9001 Formality Requirements. *Journal of Software Engineering and Applications, 4*(11), 626-638.

Qureshi, M. (2011). Empirical Evaluation of the Proposed eXSCRUM Model: Results of a Case Study. *International Journal of Computer Science Issues (IJCSI), 8*(3). 150-157.

Rainer, A., & Hall, T. (2002). A quantitative and qualitative analysis of factors affecting software processes. *Journal of Systems and Software, 66*(1), 7-21.

Ralyté, J., Deneckère, R., Rolland, C. (2003). Towards a Generic Method for Situational Method Engineering. *In Proceeding the 15th International Conference Advanced on Information Systems Engineering (CAiSE2003) held on 16-18 June 2003 at Klagenfurt, Austria* (pp. 95-110). Springer-Verlag, LNCS 2681.

Ralyté, J., Rolland, C., and Deneckère, R. (2004). Towards a Meta-tool for Change-Centric Method Engineering: A Typology of Generic Operators. *In Proceeding of the 16th International Conference on Advanced Information Systems Engineering (CAiSE2004) held on 7-4 June at Riga, Latvia* (pp. 202-218). Springer.

Ramsin, R. (2006). *The engineering of an object-oriented software development methodology.* Unpublished Doctoral Thesis, Department of Computer Science, university of York, UK.

Richardson, I. (2001). Software process matrix: a small company SPI model. *Software Process: Improvement and Practice, 6*(3), 157-165.

Richardson, I., & Von Wangenheim, C. (2007). Guest Editors' Introduction: Why are Small Software Organizations Different?. *IEEE software, 24(*1), 18-22.

Rout, T (project manager). (2002*). SPICE: Software Process Assessment-Part 1: Concepts and Introductory Guides*. Retrieved on 10 May 2009, from http://www.uio.no/studier/emner/matnat/ifi/INF5180/v10/undervisningsmaterial e/reading-materials/p10/spice/part1 100.pdf.

Rout, T., Tuffley, A., Cahill, B., and Hodgen, B.(2000). The Rapid Assessment of Software Process Capability. *In Proceedings of SPICE 2000 - the First International Conference on Software Process Improvement and Capability dEtermination, held on 1-3 June at Limerick, Ireland* (pp. 47-56). USA: Wiley-IEEE Computer Society Press.

Royce, W.W. (1987) .Managing the Development of Large Software Systems: Concepts and Techniques. *In Proceedings of the Ninth International Conference on Software Engineering held on 30 March - 2April 1987 at California, USA* (pp. 328–338). California, USA: ACM Press.

Saarnak, S., & Gustafsson, B. (2003). *A comparison of lifecycles*. Unpublished Master Thesis, Department of Software Engineering and Computer Science, Blekinge Institute of Technology, Sweden.

Saiedian, H., & Carr, N. (1997). Characterizing a software process maturity model for small organizations. *ACM SIGICE Bulletin, 23*(1), 2-11.

Salo, O. (2006). *Enabling software process improvement in agile software development teams and organisations.* Unpublished academic dissertation, Faculty of Science, University of Oulu, Linnanmaa, Finland.

Santos, G., Montoni, M., Vasconcellos, J., Figueiredo, S., Cabral, R., Cerdeiral, C., et al. (2007). Implementing software process improvement initiatives in small and medium-size enterprises in Brazil. *In Proceeding of the Sixth International Conference on the Quality of Information and Communications Technology held on 12-14 Sept. 2007 at Lisbon, Portugal* (pp.187-196). IEEE.

Savolainen, P., Sihvonen, H., & Ahonen, J. (2007). SPI with lightweight software process modeling in a small software company. *Lecture Notes in Computer Science, 4764*, 71-81.

Sengodan, B. (2003). *Integrating Automated Testing Into Object-Oriented Development Using Junit.* Unpublished Master Thesis, School of School of

Graduate Studies, College of Arts and Sciences, Florida Agricultural and Mechanical Engineering University, USA.

Shackel, B. (1991). Usability-context, framework, definition, design and evaluation. *Human factors for informatics usability*, 21-37.

Sharma, B., Sharma, N., Sharma, N. (2009). Software Process Improvement: A Comparative Analysis of SPI models. *In Proceeding of the Second International Conference on Emerging Trends in Engineering and Technology held on 16-18 Dec. 2009 at Nagpur* (pp. 1019-1024). Los Alamitos, CA, USA: IEEE Computer Society.

Sharma, L., & Sharma, N. (2012). Software Process Improvement in Small Scale Organizations: An Empirical Study. *In Proceeding of the International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT2012) held on 21- 23 March 2012 at Patiala, Punjab, India* (pp. 18-21). India: International Journal of Computer Applications® (IJCA).

Sidky, A. (2007). *A Structured Approach to Adopting Agile Practices: The Agile Adoption Framework.* Doctoral thesis, Virginia Polytechnic Institute and State University, USA.

Simila, S., Kuvaja, P.; Krzanik, L. (1997). BOOTSTRAP: a software process assessment and improvement methodology. *In Proceeding of the First Asia-Pacific Software Engineering Conference, held on 7-9 Dec. 1997 at Tokyo, Japan* (pp.183-196). IEEE.

Singh, R. (1996). International Standard ISO/IEC 12207 software life cycle processes. *Software Process: Improvement and Practice, 2*(1), 35-50.

Sison, R., Jarzabek, S., Hock, O., Rivepiboon, W., & Hai, N. (2006). Software practices in five ASEAN countries: an exploratory study. *In Proceedings of the 28th international conference on Software engineering (ICSE'06) held on 20-28 May 2006 at Shanghai, China* (pp. 628-631). New York, NY, USA: ACM.

Sommerville, I. (2007).*Software Engineering* (8th ed.). UK: Addison-Wesley.

Sommerville, I. (2011).*Software Engineering* (9th ed.). UK: Addison-Wesley.

Stambollian, A., Habra, N., Laporte, C., Desharnais, J., & Renault, A. (2006). OWPL: A Light Model & Methodology for Initiating Software Process Improvement. *In Proceedings of the 6th International SPICE conference on Process Assessment & Improvement (ISO/IEC 15504) held on 4-5 May 2006 at Luxembourg* (pp. 97-106). Luxembourg: Centre de Recherche Public Henri Tudor.

Stephens, M. (2001). The case against extreme programming. *On Software Reality*. Retrieved on 2 June 2009, from http://www. softwarereality. com/lifecycle/xp/case_against_xp. jsp.

Stitt-Gohdes, W. L., & Crews, T. B. (2004). The Delphi technique: A research strategy for career and technical education. *Journal of Career and Technical Education, 20*(2), 55-67.

Stojanovic, Z., Dahanayake, A., & Sol, H. (2003). *Modeling and Architectural Design in Agile Development Methodologies.* Paper presented at the third International Workshop on Evaluation of Modeling Methods in System Analysis and Design (EMMSAD'03), Velden, Austria.

Talbot, A., & Connor, A. (2011). Requirements Engineering Current Practice and Capability in Small and Medium Software Development Enterprises in New Zealand. *In Proceeding of the Ninth International Conference on Software Engineering Research, Management and Applications held on 10-12 Aug. 2011 at Maryland, U.S.A* (pp. 17-25).IEEE.

Thapa, V., Song, E., & Kim, H. An Approach to Verifying Security and Timing Properties in UML Models. *In Proceeding of the 15th IEEE International Conference on the Engineering of Complex Computer Systems (ICECCS) held on 22-26 March 2010 St. at Anne's College, University of Oxford, UK* (pp.193-202). IEEE Computer Society.

Thorn, C. (2009).Current state and potential of variability management practices in software-intensive SMEs: Results from a regional industrial survey. *Information and Software Technology, 52*(4), 411-421.

Tosun, A., Bener, A., & Turhan, B. (2009). Implementation of a Software Quality Improvement Project in an SME: A Before and After Comparison. *In Proceeding of the 35th Euromicro Conference on Software Engineering and Advanced Applications held on 27-29 Aug. 2009 at Patras, Greece* (pp. 203-209). Los Alamitos, CA, USA: IEEE Computer Society.

Turk, D., France, R., Rumpe, B. (2002). Limitations of Agile Software Processes. *In Proceeding of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering held on 26-30 May 2002 at Alghero, Sardinia, Italy* (pp. 43-46). New York: ACM.

Unterkalmsteiner, M., Gorschek, T., Islam, A. K. M. M., Cheng, C. K., Permadi, R. B., & Feldt, R. (2011). Evaluation and Measurement of Software Process Improvement-A Systematic Literature Review. *IEEE Transactions on Software Engineering*. *PP* (99). 1-29.

Vahaniitty, J., & Rautiainen, K. (2005). Towards an approach for managing the development portfolio in small product-oriented software companies. *In*

Proceedings of the 38th Annual Hawaii International Conference on *System Sciences (HICSS'05) held on 3-6 Jan. 2005 at Helsinki University of Technology, Finland* (pp. 314c). Big Island, HI, USA: IEEE Computer Society.

Valdes, G., Visconti, M., & Astudillo, H. (2011). The Tutelkan Reference Process: A Reusable Process Model for Enabling SPI in Small Settings. *In proceeding of the Systems, Software and Service Process Improvement: 18th European conference (EuroSPI 2011) held on 27-29 June 2011 at Roskilde, Denmark* (pp. 179-190). Heidelberg: Springer, 2011.

Vasiljevic, D. & Skoog, S. (2003).*A Software Process Improvement Framework for Small Organizations.* Unpublished Master Thesis. Department of Software Engineering and Computer Science Blekinge Institute of Technology, Sweden.

Vitoria, D. (2004). *Aligning XP with ISO 9001: 2000 TickIT Guide 5.0 "A case study in two academic software projects.* Unpublished master thesis, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden.

Von Wangenheim, C. G., Anacleto, A., & Salviano, C. F. (2004). *Mares-a methodology for software process assessment in small software companies*. Technical Report LPQS001. 04E, Universidade do Vale do Itajai-UNIVALI, Brazil.

Von Wangenheim, C. G., Weber, S., Hauck, J. C. R., & Trentin, G. (2006). Experiences on establishing software processes in small companies. *Information and Software Technology, 48*(9), pp. 890-900.

Vriens, C. 2003. Certifying for CMM Level 2 and ISO9001 with XP@Scrum. *In Proceeding of the Agile Development Conference (ADC'03) held on 25-28 June 2003 at Salt Lake City, Utah, USA* (pp. 120-124). Los Alamitos, CA, USA: IEEE Computer Society.

Walker, A., & Selfe, J. (1996). The Delphi technique: a useful tool for the allied health researcher. *British Journal of Therapy and Rehabilitation, 3*(12), 677–681.

Wang, Y. & King, G. (2000). *Software Engineering Processes: Principles and Applications.* CRC series in software engineering (Vol. I). Boca Raton, FL: CRC Press.

Weerd, G. (2009). *Advancing in software product management: An incremental method engineering approach*. Unpublished Doctoral Thesis, Dutch Research School for Information and Knowledge Systems. Utrecht University, Netherlands.

Wilkie, F., Mc Caffery, F., McFall, D., Lester, N., & Wilkinson, E. (2007). A Low-overhead Method for Software Process Appraisal. *Software Process - Improvement and Practice 12*(4), 339-349.

Withers, D.H. (2000). Software engineering best practices applied to the modeling process. *In Proceedings of the 2000 Winter Simulation Conference (WSC 2000) held on 10-13 Dec. 2000 at Wyndham Palace Resort & Spa, Orlando, FL, USA* (pp. 432-439). ACM.

Wong, B., & Hasan, S. (2007). Software Process Improvement in Bangladesh. *Software Engineering Research and Practice, ed. Arabnia, HR and Reza, H., SERP*, *1*(1), 26-29.

Xie, Y. (2011).The design of software process management and evaluation system in small and medium software enterprises. *In Proceeding of the International Conference on Computer Science and Service System (CSSS) held on 27-29 June 2011 at Nanjing, China* (pp. 2699-2701). IEEE.

Xie, Z., Li, T., Dai, F., Yu, Q., Yu, Y., Zhao, N., et al. (2010). A formal meta-model of software process. *In proceeding of the 2nd International Conference on information Science and Engineering (ICISE) held on 4-6 Dec. 2010 at Hangzhou, China* (pp.4245-4248). DC, USA: IEEE Computer Society.

Xu, B. (2009). Towards high quality software development with extreme programming methodology: practices from real software projects. *In Proceeding of the International Conference on Management and Service Science (MASS '09) held on 20-22 Sept. 2009 Wuhan, China* (pp.1-4). IEEE.

Xu, Y., Lin, Z., & Foster, W. (2003). *Agile Methodology in CMM Framework: an Approach to Success for Software Companies in China*. Paper presented at the Global Information Technology Management (GITM) Conference, Calgary, Canada.

Yin, R. K. (1984). *Case Study Research: Design and Methods.* Beverly Hills, Calif: Sage Publications.

Yourdon, E. (1997). Death March: Managing "Mission Impossible" Projects. Upper Saddle River, NJ: Prentice Hall.

Zahran, S. (1998). *Software Process Improvement: Practical Guidelines for Business Success*. Harlow, England: Addison-Wesley.

Zainal, Z. (2007). Case study as a research method. *Jurnal Kemanusiaan, Vol.*9. 1-6.

Zarour, M. (2009). *Methods to evaluate lightweight software process assessment methods based on evaluation theory and engineering design principles*.

Unpublished Doctoral Thesis. Computer Science Department, Du Quebec Uni, Canada.

Zhang, L., & Shao, D. (2011). Software process improvement for small and medium organizations based on CMMI. *In Proceeding of the 2nd International Conference on the Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC) held on 8-10 Aug. 2011 Deng Feng, China* (pp. 2402-2405). IEEE.

Zoysa, L. (2011). *Software Quality Assurance in Agile and Waterfall Software Development Methodologies: A Gap Analysis*. Unpublished Master Thesis, School of Computing, University of Colombo, Sri Lanka.

Appendix A

CMMI-Dev1.2 KPAs

KPA	Specific Goal	Specific Practices
Requirement Management The purpose of Requirements Management (REQM) is to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products.	SG 1 Manage Requirements: Requirements are managed and inconsistencies with project plans and work products are identified.	SP 1.1 Obtain an Understanding of Requirements: Develop an understanding with the requirements providers on the meaning of the requirements.
		SP 1.2 Obtain Commitment to Requirements: Obtain commitment to the requirements from the project participants.
		SP 1.3 Manage Requirements Changes: Manage changes to the requirements as they evolve during the project.
		SP 1.4 Maintain Bidirectional Traceability of Requirements: Maintain bidirectional traceability among the requirements and work products.
		SP 1.5 Identify Inconsistencies Between Project Work and Requirements: Identify inconsistencies between the project plans and work products and the requirements.
Project Planning The purpose of Project Planning (PP) is to establish and maintain plans that define project activities.	SG 1 Establish Estimates: Estimates of project planning parameters are established and maintained.	SP 1.1 Estimate the Scope of the Project: Establish a top-level work breakdown structure (WBS) to estimate the scope of the project
		SP 1.2 Establish Estimates of Work Product and Task Attributes: Establish and maintain estimates of the attributes of the work products and tasks.
		SP 1.3 Define Project Lifecycle: Define the project lifecycle phases on which to scope the planning effort.
		SP 1.4 Determine Estimates of Effort and Cost: Estimate the project effort and cost for the work products and tasks based on estimation rationale.
	SG 2 Develop a Project Plan: A project plan is established and maintained as the basis for managing the project.	SP 2.1 Establish the Budget and Schedule: Establish and maintain the project's budget and schedule.
		SP 2.2 Identify Project Risks: Identify and analyze project risks
		SP 2.3 Plan for Data Management: Plan for the management of project data
		SP 2.4 Plan for Project Resources: Plan for necessary resources to perform the project.
		SP 2.5 Plan for Needed Knowledge and Skills: Plan for knowledge and skills needed to perform the project.
		SP 2.6 Plan Stakeholder Involvement: Plan the involvement of identified stakeholders.
		SP 2.7 Establish the Project Plan: Establish and maintain the overall project plan content.
	SG 3 Obtain Commitment to the Plan: Commitments to the project plan are established and maintained.	SP 3.1 Review Plans That Affect the Project: Review all plans that affect the project to understand project commitments.
		SP 3.2 Reconcile Work and Resource Levels: Reconcile the project plan to reflect available and estimated resources.
		SP 3.3 Obtain Plan Commitment: Obtain commitment from relevant stakeholders responsible for performing and supporting plan execution.
Project Monitoring and Control	SG 1 Monitor Project Against Plan:	SP 1.1 Monitor Project Planning Parameters: Monitor the actual values of the project planning parameters

The purpose of Project Monitoring and Control (PMC) is to provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance deviates significantly from the plan.	Actual performance and progress of the project are monitored against the project plan.	against the project plan.
		SP 1.2 Monitor Commitments: Monitor commitments against those identified in the project plan.
		SP 1.3 Monitor Project Risks: Monitor risks against those identified in the project plan.
		SP 1.4 Monitor Data Management: Monitor the management of project data against the project plan.
		SP 1.5 Monitor Stakeholder Involvement: Monitor stakeholder involvement against the project plan.
		SP 1.6 Conduct Progress Reviews: Periodically review the project's progress, performance, and issues.
		SP 1.7 Conduct Milestone Reviews: Review the accomplishments and results of the project at selected project milestones.
	SG 2 Manage Corrective Action to Closure: Corrective actions are managed to closure when the project's performance or results deviate significantly from the plan.	SP 2.1 Analyze Issues: Collect and analyze the issues and determine the corrective actions necessary to address the issues
		SP 2.2 Take Corrective Action: Take corrective action on identified issues.
		SP 2.3 Manage Corrective Action: Manage corrective actions to closure.
Supplier Agreement Management The purpose of Supplier Agreement Management (SAM) is to manage the acquisition of products from suppliers.	SG 1 Establish Supplier Agreements: Agreements with the suppliers are established and maintained.	SP 1.1 Determine Acquisition Type: Determine the type of acquisition for each product or product component to be acquired.
		SP 1.2 Select Suppliers: Select suppliers based on an evaluation of their ability to meet the specified requirements and established criteria.
		SP 1.3 Establish Supplier Agreements: Establish and maintain formal agreements with the supplier.
	SG 2 Satisfy Supplier Agreements: Agreements with the suppliers are satisfied by both the project and the supplier.	SP 2.1 Execute the Supplier Agreement: Perform activities with the supplier as specified in the supplier agreement.
		SP 2.2 Monitor Selected Supplier Processes: Select, monitor, and analyze processes used by the supplier.
		SP 2.3 Evaluate Selected Supplier Work Products: Select and evaluate work products from the supplier of custom-made products.
		SP 2.4 Accept the Acquired Product: Ensure that the supplier agreement is satisfied before accepting the acquired product.
		SP 2.5 Transition Products: Transition the acquired products from the supplier to the project.
Measurement and Analysis The purpose of Measurement and Analysis (MA) is to develop and sustain a measurement capability that is used to support management information needs.	SG 1 Align Measurement and Analysis Activities: Measurement objectives and activities are aligned with identified information needs and objectives.	SP 1.1 Establish Measurement Objectives: Establish and maintain measurement objectives that are derived from identified information needs and objectives.
		SP 1.2 Specify Measures: Specify measures to address the measurement objectives.
		SP 1.3 Specify Data Collection and Storage Procedures: Specify how measurement data will be obtained and stored.
		SP 1.4 Specify Analysis Procedures: Specify how measurement data will be analyzed and reported.

	SG 2 Provide Measurement Results: Measurement results, which address identified information needs and objectives, are provided.	SP 2.1 Collect Measurement Data: Obtain specified measurement data.
		SP 2.2 Analyze Measurement Data: Analyze and interpret measurement data.
		SP 2.3 Store Data and Results: Manage and store measurement data, measurement specifications, and analysis results.
		SP 2.4 Communicate Results: Report results of measurement and analysis activities to all relevant stakeholders.
Process and Product Quality Assurance The purpose of Process and Product Quality Assurance (PPQA) is to provide staff and management with objective insight into processes and associated work products.	SG 1 Objectively Evaluate Processes and Work Products: Adherence of the performed process and associated work products and services to applicable process descriptions, standards, and procedures is objectively evaluated.	SP 1.1 Objectively Evaluate Processes: Objectively evaluate the designated performed processes against the applicable process descriptions, standards, and procedures.
		SP 1.2 Objectively Evaluate Work Products and Services: Objectively evaluate the designated work products and services against the applicable process descriptions, standards, and procedures.
	SG 2 Provide Objective Insight: Noncompliance issues are objectively tracked and communicated, and resolution is ensured.	SP 2.1 Communicate and Ensure Resolution of Noncompliance Issues: Communicate quality issues and ensure resolution of noncompliance issues with the staff and managers.
		SP 2.2 Establish Records: Establish and maintain records of the quality assurance activities.
Configuration Management The purpose of Configuration Management (CM) is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits.	SG 1 Establish Baselines: Baselines of identified work products are established. Specific practices to establish baselines are covered by this specific goal.	SP 1.1 Identify Configuration Items: Identify the configuration items, components, and related work products that will be placed under configuration management.
		SP 1.2 Establish a Configuration Management System: Establish and maintain a configuration management and change management system for controlling work products.
		SP 1.3 Create or Release Baselines: Create or release baselines for internal use and for delivery to the customer.
	SG 2 Track and Control Changes: Changes to the work products under configuration management are tracked and controlled.	SP 2.1 Track Change Requests: Track change requests for the configuration items.
		SP 2.2 Control Configuration Items: changes to the configuration items.
	SG 3 Establish Integrity: Integrity of baselines is established and maintained.	SP 3.1 Establish Configuration Management Records: Establish and maintain records describing configuration items.
		SP 3.2 Perform Configuration Audits: Perform configuration audits to maintain integrity of the configuration baselines.
Requirements Development The purpose of Requirements Development (RD) is to produce and analyze customer, product, and product component requirements.	SG 1 Develop Customer Requirements: Stakeholder needs, expectations, constraints, and interfaces are collected and translated into customer requirements.	SP 1.1 Elicit Needs: Elicit stakeholder needs, expectations, constraints, and interfaces for all phases of the product lifecycle.
		SP 1.2 Develop the Customer Requirements: Transform stakeholder needs, expectations, constraints, and interfaces into customer requirements.
	SG 2 Develop Product Requirements: Customer requirements are refined and elaborated to develop product and product component requirements.	SP 2.1 Establish Product and Product Component Requirements: Establish and maintain product and product component requirements, which are based on the customer requirements.
		SP 2.2 Allocate Product Component Requirements: Allocate the requirements for each product component.

		SP 2.3 Identify Interface Requirements: Identify interface requirements.
	SG 3 Analyze and Validate Requirements: The requirements are analyzed and validated, and a definition of required functionality is developed.	SP 3.1 Establish Operational Concepts and Scenarios: Establish and maintain operational concepts and associated scenarios.
		SP 3.2 Establish a Definition of Required Functionality: Establish and maintain a definition of required functionality.
		SP 3.3 Analyze Requirements: Analyze requirements to ensure that they are necessary and sufficient.
		SP 3.4 Analyze Requirements to Achieve Balance: Analyze requirements to balance stakeholder needs and constraints.
		SP 3.5 Validate Requirements: Validate requirements to ensure the resulting product will perform as intended in the user's environment.
Technical Solution The purpose of Technical Solution (TS) is to design, develop, and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product-related lifecycle processes either singly or in combination as appropriate.	SG 1 Select Product Component Solutions: Product or product component solutions are selected from alternative solutions.	SP 1.1 Develop Alternative Solutions and Selection Criteria: Develop alternative solutions and selection criteria.
		SP 1.2 Select Product Component Solutions: Select the product component solutions that best satisfy the criteria established.
	SG 2 Develop the Design: Product or product component designs are developed.	SP 2.1 Design the Product or Product Component: Develop a design for the product or product component.
		SP 2.2 Establish a Technical Data Package: Establish and maintain a technical data package.
		SP 2.3 Design Interfaces Using Criteria: Design product component interfaces using established criteria.
		SP 2.4 Perform Make, Buy, or Reuse Analyses: Evaluate whether the product components should be developed, purchased, or reused based on established criteria.
	SG 3 Implement the Product Design: Product components, and associated support documentation, are implemented from their designs.	SP 3.1 Implement the Design: Implement the designs of the product components.
		SP 3.2 Develop Product Support Documentation: Develop and maintain the end-use documentation.
Product Integration The purpose of Product Integration (PI) is to assemble the product from the product components, ensure that the product, as integrated, functions properly, and deliver the product.	SG 1 Prepare for Product Integration: Preparation for product integration is conducted.	SP 1.1 Determine Integration Sequence: Determine the product component integration sequence.
		SP 1.2 Establish the Product Integration Environment: Establish and maintain the environment needed to support the integration of the product components.
		SP 1.3 Establish Product Integration Procedures and Criteria: Establish and maintain procedures and criteria for integration of the product components.
	SG 2 Ensure Interface Compatibility: The product component interfaces, both internal and external, are compatible.	SP 2.1 Review Interface Descriptions for Completeness: Review interface descriptions for coverage and completeness.
		SP 2.2 Manage Interfaces: Manage internal and external interface definitions, designs, and changes for products and product components.
	SG 3 Assemble Product Components and Deliver the Product: Verified product components are assembled and the	SP 3.1 Confirm Readiness of Product Components for Integration: Confirm, prior to assembly, that each product component required to assemble the product has been properly identified, functions according to its description, and that the product component interfaces comply with

	integrated, verified, and validated product is delivered.	the interface descriptions.
		SP 3.2 Assemble Product Components: Assemble product components according to the product integration sequence and available procedures.
		SP 3.3 Evaluate Assembled Product Components: Evaluate assembled product components for interface compatibility.
		SP 3.4 Package and Deliver the Product or Product Component: Package the assembled product or product component and deliver it to the appropriate customer.
Verification The purpose of Verification (VER) is to ensure that selected work products meet their specified requirements.	SG 1 Prepare for Verification: Preparation for verification is conducted.	SP 1.1 Select Work Products for Verification: Select the work products to be verified and the verification methods that will be used for each.
		SP 1.2 Establish the Verification Environment: Establish and maintain the environment needed to support verification.
		SP 1.3 Establish Verification Procedures and Criteria: Establish and maintain verification procedures and criteria for the selected work products.
	SG 2 Perform Peer Reviews: Peer reviews are performed on selected work products.	SP 2.1 Prepare for Peer Reviews: Prepare for peer reviews of selected work products.
		SP 2.2 Conduct Peer Reviews: Conduct peer reviews on selected work products and identify issues resulting from the peer review.
		SP 2.3 Analyze Peer Review Data: Analyze data about preparation, conduct, and results of the peer reviews.
	SG 3 Verify Selected Work Products: Selected work products are verified against their specified requirements.	.SP 3.1 Perform Verification: Perform verification on the selected work products.
		SP 3.2 Analyze Verification Results: Analyze the results of all verification activities
Validation The purpose of Validation (VAL) is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment.	SG 1 Prepare for Validation: Preparation for validation is conducted.	SP 1.1 Select Products for Validation: Select products and product components to be validated and the validation methods that will be used for each.
		SP 1.2 Establish the Validation Environment: Establish and maintain the environment needed to support validation.
		SP 1.3 Establish Validation Procedures and Criteria: Establish and maintain procedures and criteria for validation.
	SG 2 Validate Product or Product Components: Establish and maintain procedures and criteria for validation.	SP 2.1 Perform Validation: Perform validation on the selected products and product components.
		SP 2.2 Analyze Validation Results: Analyze the results of the validation activities.
Organizational Process Focus The purpose of Organizational Process Focus (OPF) is to plan, implement, and deploy organizational process improvements based on a thorough understanding of the current strengths and weaknesses of the organization's processes and process assets.	SG 1 Determine Process Improvement Opportunities: Strengths, weaknesses, and improvement opportunities for the organization's processes are identified periodically and as needed.	SP 1.1 Establish Organizational Process Needs: Establish and maintain the description of the process needs and objectives for the organization.
		SP 1.2 Appraise the Organization's Processes: Appraise the organization's processes periodically and as needed to maintain an understanding of their strengths and weaknesses.
		SP 1.3 Identify the Organization's Process Improvements: Identify improvements to the organization's processes and process assets.
	SG 2 Plan and Implement Process Improvements: Process actions that address improvements to the	SP 2.1 Establish Process Action Plans: Establish and maintain process action plans to address improvements to the organization's processes and process assets.

	organization's processes and process assets are planned and implemented.	SP 2.2 Implement Process Action Plans: Implement process action plans.
	SG 3 Deploy Organizational Process Assets and Incorporate Lessons Learned: The organizational process assets are deployed across the organization and process-related experiences are incorporated into the organizational process assets.	SP 3.1 Deploy Organizational Process Assets: Deploy organizational process assets across the organization.
		SP 3.2 Deploy Standard Processes: Deploy the organization's set of standard processes to projects at their startup and deploy changes to them as appropriate throughout the life of each project.
		SP 3.3 Monitor Implementation: Monitor the implementation of the organization's set of standard processes and use of process assets on all projects.
		SP 3.4 Incorporate Process-Related Experiences into the Organizational Process Assets: Incorporate process-related work products, measures, and improvement information derived from planning and performing the process into the organizational process assets.
Organizational Process Definition +IPPD The purpose of Organizational Process Definition (OPD) is to establish and maintain a usable set of organizational process assets and work environment standards.	SG 1 Establish Organizational Process Assets: A set of organizational process assets is established and maintained.	SP 1.1 Establish Standard Processes: Establish and maintain the organization's set of standard processes.
		SP 1.2 Establish Lifecycle Model Descriptions: Establish and maintain descriptions of the lifecycle models approved for use in the organization.
		SP 1.3 Establish Tailoring Criteria and Guidelines: Establish and maintain the tailoring criteria and guidelines for the organization's set of standard processes.
		SP 1.4 Establish the Organization's Measurement Repository: Establish and maintain the organization's measurement repository.
		SP 1.5 Establish the Organization's Process Asset Library: Establish and maintain the organization's process asset library.
		SP 1.6 Establish Work Environment Standards: Establish and maintain work environment standards.
	SG 2 Enable IPPD Management: Organizational rules and guidelines, which govern the operation of integrated teams, are provided.	SP 2.1 Establish Empowerment Mechanisms: Establish and maintain empowerment mechanisms to enable timely decision making.
		SP 2.2 Establish Rules and Guidelines for Integrated Teams: Establish and maintain organizational rules and guidelines for structuring and forming integrated teams.
		SP 2.3 Balance Team and Home Organization Responsibilities: Establish and maintain organizational guidelines to help team members' balance their team and home organization responsibilities.
Organizational Training The purpose of Organizational Training (OT) is to develop the skills and knowledge of people so they can perform their roles effectively and efficiently.	SG 1 Establish an Organizational Training Capability: A training capability, which supports the organization's management and technical roles, is established and maintained.	SP 1.1 Establish the Strategic Training Needs: Establish and maintain the strategic training needs of the organization.
		SP 1.2 Determine Which Training Needs Are the Responsibility of the Organization: Determine which training needs is the responsibility of the organization and which will be left to the individual project or support group.
		SP 1.3 Establish an Organizational Training Tactical Plan: Establish and maintain an organizational training tactical plan.
		SP 1.4 Establish Training Capability: Establish and maintain training capability to address organizational training needs.

	SG 2 Provide Necessary Training: Training necessary for individuals to perform their roles effectively is provided.	SP 2.1 Deliver Training: Deliver the training following the organizational training tactical plan.
		SP 2.2 Establish Training Records: Establish and maintain records of the organizational training.
		SP 2.3 Assess Training Effectiveness: Assess the effectiveness of the organization's training program.
Integrated Project Management +IPPD The purpose of Integrated Project Management (IPM) is to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process that is tailored from the organization's set of standard processes.	SG 1 Use the Project's Defined Process: The project is conducted using a defined process that is tailored from the organization's set of standard processes.	SP 1.1 Establish the Project's Defined Process: Establish and maintain the project's defined process from project startup through the life of the project.
		SP 1.2 Use Organizational Process Assets for Planning Project Activities: Use the organizational process assets and measurement repository for estimating and planning the project's activities.
		SP 1.3 Establish the Project's Work Environment: Establish and maintain the project's work environment based on the organization's work environment standards.
		SP 1.4 Integrate Plans: Integrate the project plan and the other plans that affect the project to describe the project's defined process.
		SP 1.5 Manage the Project Using the Integrated Plans: Manage the project using the project plan, the other plans that affect the project, and the project's defined process.
		SP 1.6 Contribute to the Organizational Process Assets: Contribute work products, measures, and documented experiences to the organizational process assets.
	SG 2 Coordinate and Collaborate with Relevant Stakeholders: Coordination and collaboration of the project with relevant stakeholders is conducted.	SP 2.1 Manage Stakeholder Involvement: Manage the involvement of the relevant stakeholders in the project.
		SP 2.2 Manage Dependencies: Participate with relevant stakeholders to identify, negotiate, and track critical dependencies.
		SP 2.3 Resolve Coordination Issues: Resolve issues with relevant stakeholders.
	SG 3 Apply IPPD Principles: The project is managed using IPPD principles.	SP 3.1 Establish the Project's Shared Vision: Establish and maintain a shared vision for the project.
		SP 3.2 Establish the Integrated Team Structure: Establish and maintain the integrated team structure for the project.
		SP 3.2 Allocate Requirements to Integrated Teams: Allocate requirements, responsibilities, tasks, and interfaces to teams in the integrated team structure.
Risk Management The purpose of Risk Management (RSKM) is to identify potential problems before they occur so that risk-handling activities can be planned and invoked as needed across the life of the product or project to mitigate adverse impacts on achieving objectives.	SG 1 Prepare for Risk Management: Preparation for risk management is conducted. Preparation is conducted by establishing and maintaining a strategy for identifying, analyzing, and mitigating risks.	SP 1.1 Determine Risk Sources and Categories: Determine risk sources and categories.
		SP 1.2 Define Risk Parameters: Define the parameters used to analyze and categorize risks and the parameters used to control the risk management effort.
		SP 1.3 Establish a Risk Management Strategy: Establish and maintain the strategy to be used for risk management.
	SG 2 Identify and Analyze Risks: Risks are identified and analyzed to determine their relative importance.	SP 2.1 Identify Risks: Identify and document the risks.
		SP 2.2 Evaluate, Categorize, and Prioritize Risks: Evaluate and categorize each identified risk using the defined risk categories and parameters, and determine its relative priority.
	SG 3 Mitigate Risks: Risks are handled and mitigated, where appropriate,	SP 3.1 Develop Risk Mitigation Plans: Develop a risk mitigation plan for the most important risks to the project as defined by the risk management strategy.

	to reduce adverse impacts on achieving objectives.	SP 3.2 Implement Risk Mitigation Plans: Monitor the status of each risk periodically and implement the risk mitigation plan as appropriate.
Decision Analysis and Resolution The purpose of Decision Analysis and Resolution (DAR) is to analyze possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria.	SG 1Evaluate Alternatives: Decisions are based on an evaluation of alternatives using established criteria. Issues requiring a formal evaluation process may be identified at any time.	SP 1.1 Establish Guidelines for Decision Analysis: Establish and maintain guidelines to determine which issues are subject to a formal evaluation process.
		SP 1.2 Establish Evaluation Criteria: Establish and maintain the criteria for evaluating alternatives, and the relative ranking of these criteria.
		SP 1.3 Identify Alternative Solutions: Identify alternative solutions to address issues.
		SP 1.4 Select Evaluation Methods: Select the evaluation methods.
		SP 1.5 Evaluate Alternatives: Evaluate alternative solutions using the established criteria and methods.
		SP 1.6 Select Solutions: Select solutions from the alternatives based on the evaluation criteria.
Organizational Process Performance The purpose of Organizational Process Performance (OPP) is to establish and maintain a quantitative understanding of the performance of the organization's set of standard processes in support of quality and process-performance objectives, and to provide the process-performance data, baselines, and models to quantitatively manage the organization's projects.	SG 1 Establish Performance Baselines and Models: Baselines and models, which characterize the expected process performance of the organization's set of standard processes, are established and maintained.	SP 1.1 Select Processes: Select the processes or sub processes in the organization's set of standard processes that are to be included in the organization's process-performance analyses.
		SP 1.2 Establish Process-Performance Measures: Establish and maintain definitions of the measures that are to be included in the organization's process-performance analyses.
		SP 1.3 Establish Quality and Process-Performance Objectives: Establish and maintain quantitative objectives for quality and process performance for the organization.
		SP 1.4 Establish Process-Performance Baselines: Establish and maintain the organization's process-performance baselines.
		SP 1.5 Establish Process-Performance Models: Establish and maintain the process-performance models for the organization's set of standard processes.
Quantitative Project Management The purpose of Quantitative Project Management (QPM) is to quantitatively manage the project's defined process to achieve the project's established quality and process-performance objectives.	SG 1 Quantitatively Manage the Project: The project is quantitatively managed using quality and process-performance objectives.	SP 1.1 Establish the Project's Objectives: Establish and maintain the project's quality and process-performance objectives.
		SP 1.2 Compose the Defined Process: Select the sub processes that compose the project's defined process based on historical stability and capability data.
		SP 1.3 Select the Sub processes that Will Be Statistically Managed: Select the sub processes of the project's defined process that will be statistically
		SP 1.4 Manage Project Performance: Monitor the project to determine whether the project's objectives for quality and process performance will be satisfied, and identify corrective action as appropriate.
	SG 2 Statistically Manage Sub process Performance The performance of selected sub processes within the project's defined process is statistically managed.	SP 2.1 Select Measures and Analytic Techniques: Select the measures and analytic techniques to be used in statistically managing the selected sub processes.
		SP 2.2 Apply Statistical Methods to Understand Variation: Establish and maintain an understanding of the variation of the selected sub processes using the selected measures and analytic techniques.

		SP 2.3 Monitor Performance of the Selected Sub processes: Monitor the performance of the selected sub processes to determine their capability to satisfy their quality and process-performance objectives, and identify corrective action as necessary.
		SP 2.4 Record Statistical Management Data: Record statistical and quality management data in the organization's measurement repository.
Organizational Innovation and Deployment The purpose of Organizational Innovation and Deployment (OID) is to select and deploy incremental and innovative improvements that measurably improve the organization's processes and technologies. The improvements support the organization's quality and process-performance objectives as derived from the organization's business objectives.	SG 1 Select Improvements: Process and technology improvements, which contribute to meeting quality and process-performance objectives, are selected.	SP 1.1 Collect and Analyze Improvement Proposals: Collect and analyze process- and technology-improvement proposals.
		SP 1.2 Identify and Analyze Innovations: Identify and analyze innovative improvements that could increase the organization's quality and process performance.
		SP 1.3 Pilot Improvements: Pilot process and technology improvements to select which ones to implement.
		SP 1.4 Select Improvements for Deployment: Select process and technology improvements for deployment across the organization.
	SG 2 Deploy Improvements: Measurable improvements to the organization's processes and technologies are continually and systematically deployed.	SP 2.1 Plan the Deployment: Establish and maintain the plans for deploying the selected process and technology improvements.
		SP 2.2 Manage the Deployment: Manage the deployment of the selected process and technology improvements.
		SP 2.3 Measure Improvement Effects: Measure the effects of the deployed process and technology improvements.
Causal Analysis and Resolution The purpose of Causal Analysis and Resolution (CAR) is to identify causes of defects and other problems and take action to prevent them from occurring in the future.	SG 1 Determine Causes of Defects: Root causes of defects and other problems are systematically determined.	SP 1.1 Select Defect Data for Analysis: Select the defects and other problems for analysis.
		SP 1.2 Analyze Causes: Perform causal analysis of selected defects and other problems and propose actions to address them.
	SG 2 Address Causes of Defects: Root causes of defects and other problems are systematically addressed to prevent their future occurrence.	SP 2.1 Implement the Action Proposals: Implement the selected action proposals that were developed in causal analysis.
		SP 2.2 Evaluate the Effect of Changes: Evaluate the effect of changes on process performance.
		SP 2.3 Record Data: Record causal analysis and resolution data for use across the project and organization.

Appendix B

Detailed Comparisons of XP Practices to CMMI-Dev1.2 KPAs.

Key Process Area	**Coverage the CMM Key Process Areas by XP Practices**		
	Fritzsche & Keil (2007)	**Omran (2008)**	**Elshafey & Galal-Edeen (2008)**
Requirement Management	The intensive communication of XP method helps in understanding the requirements by integrating the customer into the development team. Therefore, the requirements of the project can be quickly exchanged and discussed. However, XP practices do not directly support the traceability of requirement, while the stories, task, stories, functional test, and unit test help in detecting the inconsistencies between the project work and the requirements. (L.S)	This process area is largely supported by some of XP practices, which are: user-stories, on-site customer, and continuous integration. In addition, short release and continues integration help in integrating the feedback on customer expectations and needs. Furthermore, on-site customer and intensive communications support the establishment and maintenance the common understanding of building stories and selecting them for the next release. (L.S)	Intensive communication and on-site customer support the understanding of the requirements and provide the commitment to these requirements. In addition, iteration to release practices helps in conducting the change or requirements. XP does not care so much for keeping or tracking requirements for future changes. Nevertheless, on-site customer and test-driven development help in detecting the inconsistencies between project work and requirements. (L.S)
Project Planning	Planning game practice is responsible for establishing the project plan for the project such as: estimation of stories and tasks. In addition, the iteration to release practice helps in increasing the estimation precision. Therefore, the risk can be identified during the short iterations. Furthermore, the high involvement and responsibility of the team members increase the commitment to the release and iteration plans. (L.S)	This process area is largely supported by two XP practices, which are: planning game and small releases practices, where planning game is responsible for establish the project plan during the small releases. (L.S)	Planning phase of XP is responsible for establishing the project schedule, budget, and plan for each iteration. In addition, iterative to release practice increase the estimates precision, helps to identify the risks, better resources allocation, and better identification for needed skills for a certain phase. Furthermore, collective ownership practice encourages the involvement of all relevant stakeholders in the planning phase, where this practice increases the commitment to the iteration plans. (L.S)
Project Monitoring and Control	Tracker is responsible for mentoring the schedule and estimates, where the information of the project's progress is gathered by the use of measures. In addition, the required information can be	This process area is largely supported XP method. Big visual chart for the project is always developed by both XP team to state the velocity of the project and commitments (stories) for small releases. (L.S)	Iteration to release and intensive communication practices help to monitor the project against the plan. These practices also help to measure the progress of the project and support to have good opportunity to

	communicated between the project team by the intensive communication practice. Furthermore, short iteration and regular commitments helps in monitoring the project against the baseline, and also offer opportunities to make adjustments. (L.S)		make corrective actions to issues in previous iterations. In addition, tracker is responsible for informing the results of daily meetings to check the status of each iteration against the plan. (L.S)
Supplier Agreement Management	This process area is not supported by XP practices. (N.S)	This process area is not supported by XP practices. (N.S)	This process area is not supported by XP method. (N.S)
Measurement and Analysis	Tracker is responsible for the measurement and analysis procedures, but there is no specific guideline for the measurement process. In addition, intensive communication helps in obtaining the measurement data within the team, and also the tracker is responsible for analyzing the measurement data and pass on the results to the team using wall charts. However, there is no specific storage to keep the results of the measurement. (P.S)	This process area is largely supported by re-factoring practice, where this practice helps in altering internal structure of the system without changing its external behavior. (L.S)	Intensive communication helps in obtaining the measurement data that measured by the tracking to keep track of the project progress. However, there is no specific guideline for the measurement and analysis. (P.S)
Process and Product Quality assurance	There is no direct practice to evaluate the processes and services aligned with the applicable process descriptions. Nevertheless, the coach is responsible for controlling that the method is applied in the right way to guide the project team in the use of XP. In addition, coach role provides the objective insight. However, there is problem in the noncompliance issues and the establishing of records of quality assurance activities. (P.S)	This process area is partially supported by pair programming practice, where peer pressure helps in assuring the conformance of the standards. (P.S)	This process area is not supported by XP method. (N.S)
Configuration Management	Code, design, test, and requirements are the items of configuration; which used by functional tests to establish the baselines at the end of each iteration. In addition, pair programming and on-site customer practices are responsible for control and track the changes or requirements. Furthermore, coding standard helps to read the code easily, while the	This process area is partially supported by some XP practices, which are planning game, collective ownership, small releases, and continuous integration. These practices lead to details of this process area. (P.S)	Pair programming, test-driven development, and re-factoring are responsible for controlling and tracking the requirement changes. However, there is no directly establishment for the configuration management baselines in XP. (P.S)

	continuous iterations help to establish the baselines. Moreover, on-site customer, pair programming, and test-driven development practices perform the audits. (L.S)		
Requirements Development	Story cards and functional tests are used to elicit and specified the requirements by customers, where the developers often support him in these tasks. In addition, iterative to release practice enable the project team to discuss the requirement details with the customer during the development. This will enable to refine the customer requirement into product requirements, and allow the validation of the requirement. However, there XP dose not support deep analyses for the requirements. (P.S)	This process area is largely supported by some of XP practices, which are: on-site customer, user stories, and iterative development. These practices help to manage the requirement development. (L.S)	Re-factoring, iteration to release, and test driven development practices support the constancy analysis and validation of the requirements. However, in XP method; there is no documentation for all the actions on the requirements, while just story cards and on-site customer support some of these documentations. This is because XP focuses on the issues which effect on the deliver value of the product, therefore they dose not care about the heavy documentations during the development. (P.S)
Technical Solution	At the beginning of the development, prototypes provide the alternative solutions, while the re-factoring and iterative development supports these alternatives during the development. In addition, coding standard, re-factoring, and pair programming support the implementation of the product. (L.S)	This process area is largely supported by some of XP practices, which are: metaphor, iterative solutions, and test-driven development. These practices lead to a high quality of technical solutions. (L.S)	In the beginning of the project development, prototype supports the alternative solutions; while iteration to release and re-factoring help in find out the alternative during the development. In addition, simple design practice provides the flexibility of the system, helps to accept the changes easily, and minimize the changes cost during the short iterative. Furthermore, re-factoring, coding standards, pair programming, test driven development, and continuous improvement practices helps to enhance the implementation of the product design to ensure better quality. (L.S)
Product Integration	Continuous integration is a main practice in XP, where this practice and on-site customer help in assembling the product components and deliver the product. In addition, the tests used in each of the integration steps help to ensure the interface compatibility. (L.S)	This process area is largely supported by XP practices which are: planning game, iteration to release, and test-driven development. (L.S)	Just continuous integration practice supports the product integration. This practice contains multiple iterations, therefore the integration steps are performed very often, a thorough preparation is critical, and then integration takes place. (P.S)
Verification	Test-driven development practice supports the verification. In addition, pair programming, collective code ownership, and re-factoring support	This process area is largely supported by some of XP practices, which are: on-site customer, user stories and iteration to release. (L.S)	Test driven development supports the enhancement of the verification process by increasing the probability of meeting the verified work to the specified

	the peer reviews. Therefore, peer reviews and test driven development are considered the main methods for verification. (L.S)		requirements. In addition, pair programming and collective ownership are suitable practices to enhance peer review process. (L.S)
Validation	Iteration to release and on-site customer support the validation by the customer acceptance, where the customer is responsible for validating the product with the product team consistently at the end of each iteration. Therefore, it will be suitable for the team to know the required changes of the requirement before to start in the subsequence iteration. (L.S)	This process area is largely supported by some of XP practices, which are: iterations to release, test-driven development, and on-site customer. In addition, pair programming addresses peer reviews. (L.S)	Re-factoring, iteration to release and on-site customer practices are responsible to ensure the validation of the product. This can be done by demonstrating that the product fulfills its intended use as required by the customer. (L.S)
Organizational Process Focus	This process area is not supported by XP practices. (N.S)	This process area is partially supported by XP method, because XP addresses organization process focus at the team level rather than organizational level. In addition, XP focuses on the software engineering process rather than organizational infrastructure issues. (P.S)	This process area is not supported by XP method. (N.S)
Organizational Process Definition +IPPD	This process area is not supported by XP practices. (N.S)	This process area is partially supported by XP method, because XP addresses the team process definition without the organizational assets. (P.S)	This process area is not supported by XP method. (N.S)
Organizational Training	At the exploration phase, the training is already supported. In addition, pair programming and coach are also improving the organizational training during the development. However, XP dose not support the assessment of the training effectiveness and dose not record the results of assessment. (P.S)	This process area is largely supported by collective ownership practice, where no one can complete his tasks without organizational training as individual development. (L.S)	There is no specific process for training in XP method; while pair programming practice supports some of the required training to develop the skills and knowledge of people. (P.S)
Integrated Project Management +IPPD	Developers, customers, testers, and management are already coordinated and collaborated by XP practices. In addition, the shard vision can be established by the intensive communication between the team members. Furthermore, the intensive communication and cooperation XP supports the baselines for the IPPD, where the skills of each	This process area is partially supported by planning game and iterations to release practices. In addition, visual charts support this process area. However, it is difficult to mange stakeholders outside the firm. (P.S)	There is no specific process for the project as a whole in XP method, where only define practices for the development project. In addition, developers, customers, and testers are coordinated and integrated by collective ownership practice. Furthermore, the intensive communication helps to establish a shared vision. (P.S)

	member can be promoted to the other team members by the intensive. However, the project's defined process is not addressed. (P.S)		
Risk Management	There is no directly support for conducting the risk management in XP method. Nevertheless, the planning phase of XP method supports the identification and analysis of the risk management. In addition, the iterative to release practice helps to mitigate the risks. (L.S)	This process area is partially supported by on-site customer and test-driven development practices, because these practices help the developers to work on the project with covered scenarios. In addition, this practices such analysis but when defining the user stories and the acceptance-test. (P.S)	Iteration to release practice and intensive communications practices helps identifying and mitigating the risks. (L.S)
Decision Analysis and Resolution	This process area is not supported by XP practices. (N.S)	This process area is partially supported by some of XP practices, which are: planning game, simple design, test-driven development, and user stories. However, these practices in XP depend on the tacit knowledge of the project team. Therefore, XP method dose not fully address this process area. (P.S)	This process area is not supported by XP method. (N.S)
Organizational Process Performance	This process area is not supported by XP practices. (N.S)	This process area is partially supported by planning game and iteration to release practices. (P.S)	This process area is not supported by XP method. (N.S)
Quantitative Project Management	This process area is not supported by XP practices. (N.S)	This process area is not supported by XP practices. (N.S)	This process area is not supported by XP method. (N.S)
Organizational Innovation and Deployment	This process area is not supported by XP practices. (N.S)	This process area is partially supported by XP values such as simplicity and feedback. In addition, these values and re-factoring practice supports this process area by improve the processes at the team level. (P.S)	This process area is not supported by XP method. (N.S)
Causal Analysis and Resolution	This process is not supported by XP practices. (N.S)	This process area is partially supported by some of XP practices, which are: planning game, pair programming, on-site customer, and re-factoring during the iterative development. (P.S)	This process area is not supported by XP method. (N.S)

Appendix C

Verification Questionnaire

Questionnaires Related to the Software Process Improvement in Small Software Development Firms

PHD Student: Mejhem Yousef AL-Tarawneh
College of Arts and Sciences, Universiti Utara Malaysia
Sintok, Kedah, MALAYSIA
Mejhem1981@yahoo.com

This questionnaire is part of Ph.D research and it is designed to ask the focus group members to verify the proposed framework. This study aims to help small software development firms to improve and manage their software development processes by using capability maturity model integration model (CMMI-Dev1.2) as a software process improvement model, and Extreme Programming (XP) as a software development method. So we need your help to clearly read the proposed framework and the related attachment files (CMMI-Dev1.2 key process areas description; XP method description; the comparison between CMMI-Dev1.2 and XP method; and the proposed framework description) to fill out the following questionnaires. This questionnaire has four parts:

- **Part One:** To verify the compatibility of the proposed framework to the specific goals of CMMI-Dev1.2 key process areas.
- **Part Two**: To verify the commitment of the proposed Extended-XP method to XP values.
- **Part Three:** To verify the suitability of the proposed framework and Extended-XP roles for their practices and for small software development firms.
- **Part Four:** To verify the suitability of the proposed framework and the proposed Extended-XP structures for the software development process improvement issues in small software development firms.

(This Questionnaire is for Studying Objectives Only)

Part One

This part aims to verify the compatibility of the proposed framework to the specific goals of CMMI-Dev1.2 key process areas.

For each process area; the question is **"Is the proposed framework compatible to the specific goals of this area".** To answer this questions; you can use (**x**) to choose your rating as follows:

- Strongly Incompatible: the proposed framework does not achieve all specific goals of the key process area.
- Strongly Compatible: the proposed framework achieves all specific goals of the key process area.

Furthermore, if you have any suggestions; you can use the specified space.

CMMI-Dev1.2 Key Process Areas	**Strongly Incompatible** ⟷ **Strongly Compatible**					**Suggestions**
	1	2	3	4	5	
Requirement Management - SG 1 Manage Requirements						
Project Planning - SG 1 Establish Estimates - SG 2 Develop a Project Plan - SG 3 Obtain Commitment to the Plan						
Project Monitoring and Control - SG 1 Monitor Project Against Plan - SG 2 Manage Corrective Action to Closure						
Supplier Agreement Management - SG 1 Establish Supplier Agreements - SG 2 Satisfy Supplier Agreements						
Measurement and Analysis - SG 1 Align Measurement and Analysis Activities - SG 2 Provide Measurement Results						
Process and Product Quality Assurance - SG 1 Objectively Evaluate Processes and Work Products - SG 2 Provide Objective Insight						
Configuration Management - SG 1 Establish Baselines - SG 2 Track and Control Changes - SG 3 Establish Integrity						
Requirements Development - SG 1 Develop Customer Requirements - SG 2 Develop Product Requirements - SG 3 Analyze and Validate Requirements						
Technical Solution - SG 1 Select Product Component Solutions - SG 2 Develop the Design - SG 3 Implement the Product Design						
Product Integration - SG 1 Prepare for Product Integration - SG 2 Ensure Interface Compatibility - SG 3 Assemble Product Components						
Verification - SG 1 Prepare for Verification - SG 2 Perform Peer Reviews - SG 3 Verify Selected Work Products						
Validation - SG 1 Prepare for Validation - SG 2 Validate Product or Product Components						

Organizational Process Focus - SG 1 Determine Process Improvement Opportunities - SG 2 Plan and Implement Process Improvements - SG 3 Deploy Organizational Process Assets and Incorporate Lessons Learned						
Organizational Process Definition +IPPD - SG 1 Establish Organizational Process Assets - SG 2 Enable IPPD Management						
Organizational Training - SG 1 Establish an Organizational Training Capability - SG 2 Provide Necessary Training						
Integrated Project Management +IPPD - SG 1 Use the Project's Defined Process - SG 2 Coordinate and Collaborate with Relevant Stakeholders - SG 3 Apply IPPD Principles						
Risk Management - SG 1 Prepare for Risk Management - SG 2 Identify and Analyze Risks - SG 3 Mitigate Risks						
Decision Analysis and Resolution - SG 1Evaluate Alternatives						
Organizational Process Performance - SG 1 Establish Performance Baselines and Models						
Quantitative Project Management - SG 1 Quantitatively Manage the Project - SG 2 Statistically Manage Sub process Performance						
Organizational Innovation and Deployment - SG 1 Select Improvements - SG 2 Deploy Improvements						
Causal Analysis and Resolution - SG 1 Determine Causes of Defects - SG 2 Address Causes of Defects						

==========

Part Two

This part aims to verify the commitment of the proposed Extended-XP method to XP values. To answer the questions of this part; please write your answer in the specific space as follows:

- YES without modifications.
- YES with modifications.
- NO.

If your answer is "YES with modifications" or "NO"; please write your suggestions in the specific space.

Values	**Questions**	**Answers**
Simplicity	Does the proposed Extended-XP achieve the simplicity value?	
Communication	Does the proposed Extended-XP achieve the communication value?	
Feedback	Does the proposed Extended-XP achieve the feedback value?	
Courage	Does the proposed Extended-XP achieve the courage value?	
Suggestions		

==========

Part Three

This part aims to verify suitability of the distribution of the proposed framework and Extended-XP roles compared to their practices. To answer the questions of this part; please write your answer in the specific space as follows:

- YES without modifications.
- YES with modifications.
- NO.

If your answer is "YES with modifications" or "NO"; please write your suggestions in the specific space.

Questions	**Answers**
Are the distribution of the proposed framework and Extended-XP roles suitable compared to their practices?	
Are the roles of the proposed framework and Extended-XP suitable for small software development firms?	
Suggestions	

==========

Part Four

This part aims to verify suitability of the proposed framework and the proposed Extended-XP structures for software development process improvement issues in small software development firms. To answer this question; please write your answer in the specific space as follows:

- YES without modifications.
- YES with modifications.
- NO.

If your answer is "YES with modifications" or "NO"; please write your suggestions in the specific space.

Question	**Answers**
Are the structures of the proposed framework and the proposed Extended-XP suitable for the software development process improvement issues in small software development firms?	
Suggestions	

Appendix D

Validation Questionnaire

Questionnaires Related to the Software Process Improvement in Small Software Development Firms

PHD Student: Mejhem Yousef AL-Tarawneh
College of Arts and Sciences
Universiti Utara Malaysia
Sintok, Kedah, MALAYSIA
Mejhem1981@yahoo.com

This questionnaire is part of Ph.D research. It is designed to ask the professional developers and managers of small software development firms about their software development background and to validate the suitability of the framework for small software development firms based on the related XP practices and additions that are used in this framework to achieve the specific goals of the suitable key process areas of CMMI-Dev1.2. So we need your help to clearly read the framework and the specifications of specific goals of each key process areas of CMMI-Dev1.2 which are attached with questionnaire to the following questions:

This questionnaire has two parts:

- Part One: To know the respondents' background.
- Part Two: To validate the suitability of the framework for small software development firms.

(This Questionnaire is for Studying Objectives Only)

Part One

Respondent Background

1- Which Best Describes Your Current Position? (Please mark as many boxes as apply)

□ Project OR Team Leader □ Manager
□ Technical Member □ Software Engineering Process group (SEPG) Member
□ Other (please specify): …….

2- On What Activities Do You Currently Work?

□ Software Requirements □ Software Quality Assurance
□ Software Design □ Configuration Management
□ Code And Unit Test □ Software Process Improvement
□ Test And Integration □ Other (please specify) (……)

3- Have You Received Any CMMI Training?

□ Yes □ No

4- How long is your software experience?

□ 5 years & less □ 6-10 years □ More than 11 years

5- How large is your firm?

□ 10 - 20 employees □ 21- 30 employees
□ 31 - 40 employees □ 41-50 employees

==========

Part Two

This part aims to find the suitability of framework for small software development firms.

For each process area; the question is **"are the related practices and additions that used to achieve the specific goals of each key process area in the framework suitable for small software development firms?".**

To answer this questions; you can use (**x**) mark to choose your rating as following:

- Strongly Unsuitable: all the related practices and additions are strongly unsuitable for small software development firms.
- Strongly Suitable: all the related practices and additions are strongly suitable for small software development firms.

CMMI-DEV 1.2 Key Process Areas	**Summary of XP practices and the suggested additions that are used in the framework to cover the specific goals of CMMI-Dev1.2 key process areas.**	**Strongly Unsuitable** ⟷ **Strongly Suitable**				
		1	2	3	4	5
Requirement Management SG 1 Manage Requirements	On-Site Customer, Planning Game, Continuous Integration, Small Release. Creating project repository at the first stage of the framework to keep the important data during the implementing of these stages of this framework.					
Project Planning SG 1 Establish Estimates SG 2 Develop a Project Plan SG 3 Obtain Commitment to the Plan	Planning Game, Small Releases, On-Site Customer					
Project Monitoring and Control SG 1 Monitor Project Against Plan SG 2 Manage Corrective Action to Closure	Small Releases, On-Site Customer, Test-driven development, Design Improvement.					
Supplier Agreement Management SG 1 Establish Supplier Agreements SG 2 Satisfy Supplier Agreements	Using process for supplying unavailable development tools, services and technologies in the first phase of the Extended-XP method.					
Measurement and Analysis SG 1 Align Measurement and Analysis Activities SG 2 Provide Measurement Results	On-Site Customer, Design Improvement. Using the project repository for storing the measurement data in the third phase of the Extended-XP method.					
Process and Product Quality Assurance SG 1 Objectively Evaluate Processes and Work Products SG 2 Provide Objective Insight	Pair programming, Continuous integration, Test-driven development, Metaphor. Using some metrics for objectively verifying the products and the process in the third phase of the Extended-XP method. Conveying the metrics through defined channels to the affected parties and senior management in the third phase of the Extended-XP method.					
Configuration Management SG 1 Establish Baselines SG 2 Track and Control Changes SG 3 Establish Integrity	Planning Game, Continuous Integration, Collective Ownership, Design Improvement, Small Releases.					
Requirements Development SG 1 Develop Customer Requirements SG 2 Develop Product Requirements SG 3 Analyze and Validate Requirements	Planning Game, On-Site Customer, Small Releases. Storing the requirement specifications in the project repository in the first phase of the Extended-XP method.					
Technical Solution SG 1 Select Product Component Solutions SG 2 Develop the Design SG 3 Implement the Product Design	Simple Design, Coding Standards, Design Improvement, Metaphor, On-Site Customer.					
Product integration SG 1 Prepare for Product Integration SG 2 Ensure Interface Compatibility SG 3 Assemble Product Components	Continuous integration, Simple Design, Coding Standards, Design Improvement, Metaphor, On-Site Customer.					

Verification SG 1 Prepare for Verification SG 2 Perform Peer Reviews SG 3 Verify Selected Work Products	Small Releases, On-Site Customer, User Stories, Design Improvement, Pair Programming.					
Validation SG 1 Prepare for Validation SG 2 Validate Product or Product Components	Small Releases, Pair Programming, On-Site Customer.					
Organizational Process Focus SG 1 Determine Process Improvement Opportunities SG 2 Plan and Implement Process Improvements SG 3 Deploy Organizational Process Assets and Incorporate Lessons Learned	Extracting the best practices of the current project at stage three of the framework.					
Organizational Process Definition +IPPD SG 1 Establish Organizational Process Assets. SG 2 Enable IPPD Management	Metaphor. Supporting the project team with the required XP books and the description of the Extended-XP method and using this description as guidance during the software development lifecycle.					
Organizational Training SG 1 Establish an Organizational Training Capability. SG 2 Provide Necessary Training	Pair Programming, Collective Ownership. Training the project team on the Extended-XP in the beginning of the second stage of the framework.					
Integrated Project Management +IPPD SG 1 Use the Project's Defined Process SG 2 Coordinate and Collaborate with Relevant Stakeholders SG 3 Apply IPPD Principles	Pair Programming, Metaphor, On-Site Customer, Collective ownership.					
Risk Management SG 1 Prepare for Risk Management SG 2 Identify and Analyze Risks SG 3 Mitigate Risks	Small Releases, Pair Programming, On-Site Customer, Simple Design.					
Decision Analysis and Resolution SG 1Evaluate Alternatives	Simple Design, Pair Programming.					
Organizational Process Performance SG 1 Establish Performance Baselines and Models	Planning Game, Small Releases, Design Improvement. Using some metrics to conduct the process performance at the third phase of the Extended-XP method.					
Quantitative Project Management SG 1 Quantitatively Manage the Project SG 2 Statistically Manage Sub process Performance	On Site Customer, Planning Game, Continuous Integration, Collective Code Ownership, Design Improvement.					
Causal Analysis and Resolution SG 1 Determine Causes of Defects SG 2 Address Causes of Defects	Test-driven development, Continuous Integration, Pair Programming, On-Site Customer					

Appendix E

Evaluation Criteria Questionnaire

Criteria	Questions
Gain Satisfaction	How did you find the framework with regard to its perceived usefulness?
	How do you rate the framework support for decision-making or decision-making satisfaction?
	How would you rate the framework with regard to its appropriateness for task, comparison with alternatives (other available guidance), cost-effectiveness, and clarity (clear and illuminate the process)?
Interface Satisfaction	How did you find framework regarding perceived ease of use?
	How do you rate the framework with regard to its presentation (readable and useful format), internal consistency, organization (well organized), and appropriateness for audience?
Task Support Satisfaction	How did you find the framework with regard to ease of implementation?
	How do you rate the framework with regard to its understandability (simple to understand)?
	How would rate the framework with regard to the completeness of its features and procedures were they adequate and sufficient? How about completeness of output information and was it self-contained?
	To what extent does the framework produce results that you expected?
	How did you find the results of using the framework; is it enable to produce the desired products (provide reports or outputs to you that seem to be just about what you need)?

Appendix F
Assessing the Current Software Development Processes

Instructions

To the right of each question, there are boxes for the three possible responses: Largely Supported, Partially Supported, and Not Supported.

Check "Largely Supported" when:

- The current software development processes achieve the majority specific goals of the key process area.

Check "Partially Supported" when:

- The current software development processes achieve some of the specific goals of the key process area.

Check "Not Supported" when:

- The current software development processes can-not achieve the specific goals of the key process area.

Key Process Areas	Largely Supported	Partially Supported	Not Supported
Requirement Management: The purpose of Requirements Management (REQM) is to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products. This process involves of: SG 1 Manage Requirements: Requirements are managed and inconsistencies with project plans and work products are identified.			
Project Planning: The purpose of Project Planning (PP) is to establish and maintain plans that define project activities. This process involves of: - SG 1 Establish Estimates: Estimates of project planning parameters are established and maintained. - SG 2 Develop a Project Plan: A project plan is established and maintained as the basis for managing the project. - SG 3 Obtain Commitment to the Plan: Commitments to the project plan are established and maintained.			
Project Monitoring and Control: The purpose of Project Monitoring and Control (PMC) is to provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance deviates significantly from the plan. This process involves of: - SG 1 Monitor Project Against Plan: Actual performance and progress of the project are monitored against the project plan. - SG 2 Manage Corrective Action to Closure: Corrective actions are managed to closure when the project's performance or results deviate significantly from the plan.			

Supplier Agreement Management: The purpose of Supplier Agreement Management (SAM) is to manage the acquisition of products from suppliers. This process involves of: - SG 1 Establish Supplier Agreements: Agreements with the suppliers are established and maintained. - SG 2 Satisfy Supplier Agreements: Agreements with the suppliers are satisfied by both the project and the supplier.			
Measurement and Analysis: The purpose of Measurement and Analysis (MA) is to develop and sustain a measurement capability that is used to support management information needs. This process involves of: - SG 1 Align Measurement and Analysis Activities: Measurement objectives and activities are aligned with identified information needs and objectives. - SG 2 Provide Measurement Results: Measurement results, which address identified information needs and objectives, are provided.			
Process and Product Quality Assurance: The purpose of Process and Product Quality Assurance (PPQA) is to provide staff and management with objective insight into processes and associated work products. This process involves of: - SG 1 Objectively Evaluate Processes and Work Products: Adherence of the performed process and associated work products and services to applicable process descriptions, standards, and procedures is objectively evaluated. - SG 2 Provide Objective Insight: Noncompliance issues are objectively tracked and communicated, and resolution is ensured.			
Configuration Management: The purpose of Configuration Management (CM) is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits. This process involves of: - SG 1 Establish Baselines: Baselines of identified work products are established. Specific practices to establish baselines are covered by this specific goal. - SG 2 Track and Control Changes: Changes to the work products under configuration management are tracked and controlled. - SG 3 Establish Integrity: Integrity of baselines is established and maintained.			
Requirements Development: The purpose of Requirements Development (RD) is to produce and analyze customer, product, and product component requirements. This process involves of: - SG 1 Develop Customer Requirements: Stakeholder needs, expectations, constraints, and interfaces are collected and translated into customer requirements. - SG 2 Develop Product Requirements: Customer requirements are refined and elaborated to develop product and product component requirements. - SG 3 Analyze and Validate Requirements: The requirements are analyzed and validated, and a definition of required functionality is developed.			

Technical Solution: The purpose of Technical Solution (TS) is to design, develop, and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product-related lifecycle processes either singly or in combination as appropriate. This process involves of: - SG 1 Select Product Component Solutions: Product or product component solutions are selected from alternative solutions. - SG 2 Develop the Design: Product or product component designs are developed. - SG 3 Implement the Product Design: Product components, and associated support documentation, are implemented from their designs.			
Product Integration: The purpose of Product Integration (PI) is to assemble the product from the product components, ensure that the product, as integrated, functions properly, and deliver the product. This process involves of: - SG 1 Prepare for Product Integration: Preparation for product integration is conducted. - SG 2 Ensure Interface Compatibility: The product component interfaces, both internal and external, are compatible. - SG 3 Assemble Product Components and Deliver the Product: Verified product components are assembled and the integrated, verified, and validated product is delivered.			
Verification: The purpose of Verification (VER) is to ensure that selected work products meet their specified requirements. This process involves of: - SG 1 Prepare for Verification: Preparation for verification is conducted. - SG 2 Perform Peer Reviews: Peer reviews are performed on selected work products. - SG 3 Verify Selected Work Products: Selected work products are verified against their specified requirements.			
Validation: The purpose of Validation (VAL) is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment. This process involves of: - SG 1 Prepare for Validation: Preparation for validation is conducted. - SG 2 Validate Product or Product Components: Establish and maintain procedures and criteria for validation.			
Organizational Process Focus: The purpose of Organizational Process Focus (OPF) is to plan, implement, and deploy organizational process improvements based on a thorough understanding of the current strengths and weaknesses of the organization's processes and process assets. This process involves of: - SG 1 Determine Process Improvement Opportunities: Strengths, weaknesses, and improvement opportunities for the organization's processes are identified periodically and as needed. - SG 2 Plan and Implement Process Improvements: Process actions that address improvements to the organization's processes and process assets are planned and implemented. - SG 3 Deploy Organizational Process Assets and Incorporate Lessons Learned: The organizational process assets are deployed across the organization and process-related experiences are incorporated into the organizational process assets.			

Organizational Process Definition +IPPD: The purpose of Organizational Process Definition (OPD) is to establish and maintain a usable set of organizational process assets and work environment standards. This process involves of: - SG 1 Establish Organizational Process Assets: A set of organizational process assets is established and maintained. - SG 2 Enable IPPD Management: Organizational rules and guidelines, which govern the operation of integrated teams, are provided.			
Organizational Training: The purpose of Organizational Training (OT) is to develop the skills and knowledge of people so they can perform their roles effectively and efficiently. This process involves of: - SG 1 Establish an Organizational Training Capability: A training capability, which supports the organization's management and technical roles, is established and maintained. - SG 2 Provide Necessary Training: Training necessary for individuals to perform their roles effectively is provided.			
Integrated Project Management +IPPD: The purpose of Integrated Project Management (IPM) is to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process that is tailored from the organization's set of standard processes. This process involves of: - SG 1 Use the Project's Defined Process: The project is conducted using a defined process that is tailored from the organization's set of standard processes. - SG 2 Coordinate and Collaborate with Relevant Stakeholders: Coordination and collaboration of the project with relevant stakeholders is conducted. - SG 3 Apply IPPD Principles: The project is managed using IPPD principles.			
Risk Management: The purpose of Risk Management (RSKM) is to identify potential problems before they occur so that risk-handling activities can be planned and invoked as needed across the life of the product or project to mitigate adverse impacts on achieving objectives. This process involves of: - SG 1 Prepare for Risk Management: Preparation for risk management is conducted. Preparation is conducted by establishing and maintaining a strategy for identifying, analyzing, and mitigating risks. - SG 2 Identify and Analyze Risks: Risks are identified and analyzed to determine their relative importance. - SG 3 Mitigate Risks: Risks are handled and mitigated, where appropriate, to reduce adverse impacts on achieving objectives.			
Decision Analysis and Resolution: The purpose of Decision Analysis and Resolution (DAR) is to analyze possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria. This process involves of: - SG 1Evaluate Alternatives: Decisions are based on an evaluation of alternatives using established criteria. Issues requiring a formal evaluation process may be identified at any time.			

Organizational Process Performance: The purpose of Organizational Process Performance (OPP) is to establish and maintain a quantitative understanding of the performance of the organization's set of standard processes in support of quality and process-performance objectives, and to provide the process-performance data, baselines, and models to quantitatively manage the organization's projects. This process involves of: - SG 1 Establish Performance Baselines and Models: Baselines and models, which characterize the expected process performance of the organization's set of standard processes, are established and maintained.			
Quantitative Project Management: The purpose of Quantitative Project Management (QPM) is to quantitatively manage the project's defined process to achieve the project's established quality and process-performance objectives. This process involves of: - SG 1 Quantitatively Manage the Project: The project is quantitatively managed using quality and process-performance objectives. - SG 2 Statistically Manage Sub process Performance: The performance of selected sub processes within the project's defined process is statistically managed.			
Organizational Innovation and Deployment: The purpose of Organizational Innovation and Deployment (OID) is to select and deploy incremental and innovative improvements that measurably improve the organization's processes and technologies. The improvements support the organization's quality and process-performance objectives as derived from the organization's business objectives. This process involves of: - SG 1 Select Improvements: Process and technology improvements, which contribute to meeting quality and process-performance objectives, are selected. - SG 2 Deploy Improvements: Measurable improvements to the organization's processes and technologies are continually and systematically deployed.			
Causal Analysis and Resolution: The purpose of Causal Analysis and Resolution (CAR) is to identify causes of defects and other problems and take action to prevent them from occurring in the future. This process involves of: - SG 1 Determine Causes of Defects: Root causes of defects and other problems are systematically determined. - SG 2 Address Causes of Defects: Root causes of defects and other problems are systematically addressed to prevent their future occurrence.			

Appendix G

Focus Group Researchers' Profiles

Expert Researchers of Focus Group	Academic Background	Related Experiences of this Research
Professor. Asim Abdel Rahman El Sheikh	BSc (Comp. Science, University of Khartoum , Sudan, 1979). MSc (Operational Research, University of London, England, 1983). Ph.D (Computer Simulation, University of London, England, 1987).	Supervision of doctoral students in the following areas (2005-2011): -Extreme Programming - CMMI - SPI for large software firms. **Several publications in software process fields such as:** - Asim El-Sheikh, Evon Abu Taieh & Jeihan M. Abu-Tayeh, "Information Technology Projects System Development Life Cycles: Comparative Study", in the book "Handbook of Research on Technology Management's Planning and Operations, edited by Dr. Kidd, Idea Group Inc. 2009, USA. - Asim El Sheikh, Mouhib Alnoukari, and Faek Diko, "Introducing Discipline to XP: Introducing PRINCE2 on XP Projetcs", in the Proc. Informatics 2009, IADIS Multi Conference on Computer Science and Information Systems (MCCMIS 2009), Hans Weghorn, Jorg Roth, and Pedro Isaias (eds), ISBN: 978-972-8924-86-7, pp. 51-58, Algarve, Portugal, 18-20 June 2009. - Omaima Al-Allaf, Asim El-Sheikh and Ghassan Al-Utaibi, An Analytical Survey of Large Web Applications Development in Large Jordanian Web Development Enterprises, 10th International Conference on Information Integration & Web-based Applications & services (iiWAS 2008), Linz, Austria, 24-26 November 2008. - Asim El Sheikh, Haroon Tarawneh, "Web Engineering in Small Jordanian Web Development Firms: An XP Based Process Model", in the book "Utilizing Information Technology Systems across Disciplines: Advancements in the Application of Computer Science", edited by Asim El-Sheikh, Evon Abu Taieh & Jeihan M. Abu-Tayeh, Idea Group Inc. (IGI), 2009, USA.

Dr. Mouhib Alnoukari	BSc (Computer Engineering, Damascus University Syria, 1990). MSc (Computer Engineering, Montpellier University, France, 1993). Ph.D (Management Information Systems (MIS), Arabic Academy for Banking and Financial Sciences, Damascus, Syria, 2009).	**Thesis Title:** A Business Intelligence Modeling and Integration Framework Based on Agile Methodologies. Project Director & CMMI Consultant in CMMI-Syria (2006- present): The goal is to prepare 10 Syrian software companies to obtain CMMI L2&L3. Project is cooperation between the Syrian Ministry of Telecommunication and Technology, Egyptian Ministry of Telecommunication and Technology, and Syrian Computer Society. **Several publications in software process fields such as:** - Asim El Sheikh, and Mouhib Alnoukari: "Business Intelligence and Agile Methodologies for Knowledge-Based Organizations : Cross-Disciplinary Applications". IGI Global. This publication is anticipated to be released in 2011. - Mouhib Alnoukari, Asim El Sheikh, and Zaidoun Alzoabi, "Applying ASD-DM Methodology on Business Intelligence Solutions: A Case Study on Building Customer Care Data Mart", in the Proc. Data Mining 2009, IADIS Multi Conference on Computer Science and Information Systems (MCCMIS 2009), Ajith P. Abraham (ed.), ISBN: 978-972-8924-88-1, pp. 153-157, Algarve, Portugal, 18-20 June 2009.
Dr.Haroon Salem AL-Tarawneh	BSc (Comp. Science, University of Mu'tah, Jordan, 1997). MSc (Computer Information System (CIS), Arabic Academy for Banking and Financial Sciences, Amman, Jordan, 2003). Ph.D (Computer Information System (CIS), Arabic Academy for Banking and Financial Sciences, Amman, Jordan, 2007).	**Thesis Title:** A Theoretical Software Process Framework for Web Applications Development in Small Software Firms. Head of computer department in Al-balqa Applied University (Jordan) and lecturer for several subjects such as:- Systems Analysis, Software Engineering, Management Information Systems, - Information Systems Management, - Decision Support Systems & Expert Systems, and Database Systems. **Several publications in software process fields such as:** -Altarawneh, H., Amro, S. (2008), Software Process Improvement In Small Jordanian Software Development Firms. Paper presented at the 7th International Conference on Perspectives in Business Informatics Research (BIR'2008), Gdansk, Poland. PP 175-189. - Asim El Sheikh, Haroon Tarawneh, A Theoretical Agile Process Framework for Web Applications Development in Small Software Firms, The 6th ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2008, 20-22 August 2008, Prague, Czech Republic, SERA 2008: 125-132. -Asim El Sheikh, Haroon Tarawneh, A Survey of Web Engineering Practice in Small Jordanian Web Development Firms, ESEC/FSE'07, Cavtat near Dubrovnik, Croatia , September 3-7, 2007.

Appendix H

Best Practices Questionnaire

Instructions

To conduct the best practices of the current project, please response by selecting one of these options: Yes, No, Does Not Apply, and Don't Know.

"Yes": when the practice is well established and consistently performed. -
The practice should be performed nearly always in order to be considered well-established and consistently performed as a standard operating procedure.

"No": when the practice is not well established or is inconsistently performed.
- The practice may be performed sometimes, or even frequently, but it is omitted under difficult circumstances.

"Does Not Apply": when you have the required knowledge about the project or organization and the question asked, but you feel the question does not apply to the project. For example, the entire section on "**Supplier agreement management**" may not apply to the project if you do not need any external development tools or services.

"Don't Know": when you are uncertain about how to answer the question.

CMMI-Dev1.2 Level 2		
Process Area, Specific Goal and Practices		**Answers**
Requirement Management		
SG1	SP 1.1 Obtain an Understanding of Requirements	
	SP 1.2 Obtain Commitment to Requirements	
	SP 1.3 Manage Requirements Changes	
	SP 1.4 Maintain Bidirectional Traceability of Requirements	
	SP 1.5 Identify Inconsistencies Between Project Work and Requirements	
Project Planning		
SG 1	SP 1.1 Estimate the Scope of the Project	
	SP 1.2 Establish Estimates of Work Product and Task Attributes	
	SP 1.3 Define Project Lifecycle	
	SP 1.4 Determine Estimates of Effort and Cost	
SG 2	SP 2.1 Establish the Budget and Schedule	
	SP 2.2 Identify Project Risks	
	SP 2.3 Plan for Data Management	
	SP 2.4 Plan for Project Resources	
	SP 2.5 Plan for Needed Knowledge and Skills	

	SP 2.6 Plan Stakeholder Involvement	
	SP 2.7 Establish the Project Plan	
SG 3	SP 3.1 Review Plans That Affect the Project	
	SP 3.2 Reconcile Work and Resource Levels	
	SP 3.3 Obtain Plan Commitment	
Project Monitoring and Control		
SG 1	SP 1.1 Monitor Project Planning Parameters	
	SP 1.2 Monitor Commitments	
	SP 1.3 Monitor Project Risks	
	SP 1.4 Monitor Data Management	
	SP 1.5 Monitor Stakeholder Involvement	
	SP 1.6 Conduct Progress Reviews	
	SP 1.7 Conduct Milestone Reviews	
SG 2	SP 2.1 Analyze Issues	
	SP 2.2 Take Corrective Action	
	SP 2.3 Manage Corrective Action	
Supplier Agreement Management		
SG 1	SP 1.1 Determine Acquisition Type	
	SP 1.2 Select Suppliers	
	SP 1.3 Establish Supplier Agreements	
SG 2	SP 2.2 Monitor Selected Supplier Processes	
	SP 2.3 Evaluate Selected Supplier Work Products	
	SP 2.4 Accept the Acquired Product	
	SP 2.5 Transition Products	
Measurement and Analysis		
SG 1	SP 1.1 Establish Measurement Objectives	
	SP 1.2 Specify Measures	
	SP 1.3 Specify Data Collection and Storage Procedures	
	SP 1.4 Specify Analysis Procedures	
SG 2	SP 2.1 Collect Measurement Data	
	SP 2.2 Analyze Measurement Data	
	SP 2.3 Store Data and Results	
	SP 2.4 Communicate Results	
Process and Product Quality Assurance		
SG 1	SP 1.1 Objectively Evaluate Processes	
	SP 1.2 Objectively Evaluate Work Products and Services	
SG 2	SP 2.1 Communicate and Ensure Resolution of Noncompliance Issues	
	SP 2.2 Establish Records	
Configuration Management		
SG 1	SP 1.1 Identify Configuration Items	
	SP 1.2 Establish a Configuration Management System	
	SP 1.3 Create or Release Baselines	
SG 2	SP 2.1 Track Change Requests	
	SP 2.2 Control Configuration Items	
SG 3	SP 3.1 Establish Configuration Management Records	

	SP 3.2 Perform Configuration Audits	
CMMI-Dev1.2 Level 3		
Process Area, Specific Goal and Practices		**Answers**
Requirements Development		
SG1	SP 1.1 Elicit Needs	
	SP 1.2 Develop the Customer Requirements	
SG 2	SP 2.1 Establish Product and Product Component Requirements	
	SP 2.2 Allocate Product Component Requirements	
	SP 2.3 Identify Interface Requirements	
SG 3	SP 3.1 Establish Operational Concepts and Scenarios	
	SP 3.2 Establish a Definition of Required Functionality	
	SP 3.3 Analyze Requirements	
	SP 3.4 Analyze Requirements to Achieve Balance	
	SP 3.5 Validate Requirements	
Technical Solution		
SG 1	SP 1.1 Develop Alternative Solutions and Selection Criteria	
	SP 1.2 Select Product Component Solutions	
SG 2	SP 2.1 Design the Product or Product Component	
	SP 2.2 Establish a Technical Data Package	
	SP 2.3 Design Interfaces Using Criteria	
	SP 2.4 Perform Make, Buy, or Reuse Analyses	
SG 3	SP 3.1 Implement the Design	
	SP 3.2 Develop Product Support Documentation	
Product Integration		
SG 1	SP 1.1 Determine Integration Sequence	
	SP 1.2 Establish the Product Integration Environment	
	SP 1.3 Establish Product Integration Procedures and Criteria	
SG 2	SP 2.1 Review Interface Descriptions for Completeness	
	SP 2.2 Manage Interfaces	
SG 3	SP 3.1 Confirm Readiness of Product Components for Integration	
	SP 3.2 Assemble Product Components	
	SP 3.3 Evaluate Assembled Product Components	
	SP 3.4 Package and Deliver the Product or Product Component	
Verification		
SG 1	SP 1.1 Select Work Products for Verification	
	SP 1.2 Establish the Verification Environment	
	SP 1.3 Establish Verification Procedures and Criteria	
SG 2	SP 2.1 Prepare for Peer Reviews	
	SP 2.2 Conduct Peer Reviews	
	SP 2.3 Analyze Peer Review Data	
SG 3	SP 3.1 Perform Verification	
	SP 3.2 Analyze Validation Results	
Validation		
SG 1	SP 1.1 Select Products for Validation	
	SP 1.2 Establish the Validation Environment	

	SP 1.3 Establish Validation Procedures and Criteria	
SG 2	SP 2.1 Perform Validation	
	SP 2.2 Analyze Validation Results	
Organizational Process Focus		
SG 1	SP 1.1 Establish Organizational Process Needs	
	SP 1.2 Appraise the Organization’s Processes	
	SP 1.3 Identify the Organization's Process Improvements	
SG 2	SP 2.1 Establish Process Action Plans	
	SP 2.2 Implement Process Action Plans	
SG 3	SP 3.1 Deploy Organizational Process Assets	
	SP 3.2 Deploy Standard Processes	
	SP 3.3 Monitor Implementation	
	SP 3.4 Incorporate Process-Related Experiences into the Organizational Process	
Organizational Process Definition +IPPD		
SG 1	SP 1.1 Establish Standard Processes	
	SP 1.2 Establish Lifecycle Model Descriptions	
	SP 1.3 Establish Tailoring Criteria and Guidelines	
	SP 1.4 Establish the Organization’s Measurement Repository	
	SP 1.5 Establish the Organization’s Process Asset Library	
	SP 1.6 Establish Work Environment Standards	
SG 2	SP 2.1 Establish Empowerment Mechanisms	
	SP 2.2 Establish Rules and Guidelines for Integrated Teams	
	SP 2.3 Balance Team and Home Organization Responsibilities	
Organizational Training		
SG 1	SP 1.1 Establish the Strategic Training Needs	
	SP 1.2 Determine Which Training Needs Are the Responsibility of the Organization	
	SP 1.3 Establish an Organizational Training Tactical Plan	
	SP 1.4 Establish Training Capability	
SG 2	SP 2.1 Deliver Training	
	SP 2.2 Establish Training Records	
	SP 2.3 Assess Training Effectiveness	
Integrated Project Management +IPPD		
SG 1	SP 1.1 Establish the Project’s Defined Process	
	SP 1.2 Use Organizational Process Assets for Planning Project Activities	
	SP 1.3 Establish the Project's Work Environment	
	SP 1.4 Integrate Plans	
	SP 1.5 Manage the Project Using the Integrated Plans	
	SP 1.6 Contribute to the Organizational Process Assets	
SG 2	SP 2.1 Manage Stakeholder Involvement	
	SP 2.2 Manage Dependencies	
	SP 2.3 Resolve Coordination Issues	
SG 3	SP 3.1 Establish the Project’s Shared Vision	
	SP 3.2 Establish the Integrated Team Structure	
	SP 3.3 Allocate Requirements to Integrated Teams	
	SP 3.4 Establish Integrated Teams	

	SP 3.5 Ensure Collaboration among Interfacing Teams	
Risk Management		
SG 1	SP 1.1 Determine Risk Sources and Categories	
	SP 1.2 Define Risk Parameters	
	SP 1.3 Establish a Risk Management Strategy	
SG 2	SP 2.1 Identify Risks	
	SP 2.2 Evaluate, Categorize, and Prioritize Risks	
SG 3	SP 3.1 Develop Risk Mitigation Plans	
	SP 3.2 Implement Risk Mitigation Plans	
Decision Analysis and Resolution		
SG 1	SP 1.1 Establish Guidelines for Decision Analysis	
	SP 1.2 Establish Evaluation Criteria	
	SP 1.3 Identify Alternative Solutions	
	SP 1.4 Select Evaluation Methods	
	SP 1.5 Evaluate Alternatives	
	SP 1.6 Select Solutions	
CMMI-Dev1.2 Level 4		
Process Area, Specific Goal and Practices		**Answers**
Organizational Process Performance		
SG 1	SP 1.1 Select Processes	
	SP 1.2 Establish Process-Performance Measures	
	SP 1.3 Establish Quality and Process-Performance Objectives	
	SP 1.4 Establish Process-Performance Baselines	
	SP 1.5 Establish Process-Performance Models	
Quantitative Project Management		
SG 1	SP 1.1 Establish the Project's Objectives	
	SP 1.2 Compose the Defined Process	
	SP 1.3 Select the Sub processes that Will Be Statistically Managed	
	SP 1.4 Manage Project Performance	
SG 2	SP 2.1 Select Measures and Analytic Techniques	
	SP 2.2 Apply Statistical Methods to Understand Variation	
	SP 2.3 Monitor Performance of the Selected Sub processes	
	SP 2.4 Record Statistical Management Data	
CMMI-Dev 1.2 Level 5		
Process Area, Specific Goal and Practices		**Answers**
Causal Analysis and Resolution		
SG 1	SP 1.1 Select Defect Data for Analysis	
	SP 1.2 Analyze Causes	
SG 2	SP 2.1 Implement the Action Proposals	
	SP 2.2 Evaluate the Effect of Changes	
	SP 2.3 Record Data	