universität wien

DISSERTATION

Titel der Dissertation

„A Framework for Transferring Software Project Management Approaches into the Thai Telecommunications Industry“

Verfasserin

Nalinpat Porrawatpreyakorn

angestrebter akademischer Grad

Doktorin der technischen Wissenschaften (Dr. techn.)

Wien, 2012

Studienkennzahl lt. Studienblatt:	A 786 175
Dissertationsgebiet lt. Studienblatt:	Informatik
Betreuer:	Univ.-Prof. Dipl.-Ing. DDr. Gerald Quirchmayr

Abstract

This dissertation focuses on the Thai telecommunications industry, which still is one of the fastest-evolving and most competitive markets and also one of the fastest-growing technology areas, as a case study. Looking at the current situation of software development in this industry, we use the findings of interviews with in-house and outsourcing software development teams working for two of the largest broadband Internet Service Providers (ISPs) in Bangkok, Thailand (named ISP1 and ISP2) during March and April 2009. The findings reveal that many of the typical problems (e.g., a lack of good user participation, a lack of teamwork, a lack of training support, a lack of management commitment, a lack of project management competence, a lack of knowledge transfer, and so forth) are still arising throughout the software development lifecycle. These problems result in a significant level of unsatisfactory quality results. This software development situation emphasizes that there is a need for more efficient and effective software development processes and a supporting knowledge transfer process. This dissertation consequently aims at providing a methodologically sound approach that leads to a practically feasible solution resulting in improved software development performance.

Focusing on project management and software development processes, agile methods (e.g., Adaptive Software Development, eXtreme Programming, and Scrum) are widely used in many business environments, as they provide an effective software development process to tackle many of the typical problems. Nevertheless, they offer limited support for project management (e.g., for outsourcing and high quality assurance) which is the backbone for cost-efficient software development. Furthermore, they generally deal with "how", but not much with "what" software development processes should be implemented. Concentrating on only "how" cannot guarantee that software quality will be delivered. Therefore, this dissertation proposes a software process maintenance framework which in this context means a framework for software process development and improvement to overcome these shortcomings. The framework consists of two core components: a software development maturity model providing the "what" to improve with a software process assessment mechanism and an integrated PMBOK-Scrum model providing the "how" to implement with a comprehensive set of project management and software development processes. To support the application of the framework, a prototype tool is then introduced. It was created as a Web-based application, using the Java programming language and a MySQL database. It is important to perform a feasibility check on whether the framework and the tool are practical in real-life software projects. Hence, this dissertation demonstrates their implementation and results through two case studies in the Thai telecommunications industry (i.e., CAT Telecom Public Company Limited and TOT Public Company Limited) from November 2010 to February 2011. The data collection was carried out through on-site observations, individual interviews, and questionnaires. The findings indicate the generation of positive effects by (i) increasing software development performance in terms of efficiency (e.g., increasing work completeness and work productivity) and effectiveness (e.g., reducing defects and increasing customer and team satisfaction); and (ii) cultivating teamwork, collaboration, informal and frequent communications, and a knowledge sharing culture.

Focusing on a knowledge transfer process, a software project consists of knowledge-intensive activities and its implementation requires stakeholders' expertise and experience, transferability, and the absorptive capacity to learn and apply knowledge to solve problems occurring during software development. The knowledge transfer itself has its components and

can be viewed in different ways (e.g., process base, antecedent base, and component base). Although many knowledge transfer models and studies in software development have been proposed, and are available to learn from; they neither put an emphasis on a knowledge transfer's common components, nor do they clearly provide comprehensive descriptions or relationships between those components in a knowledge transfer process. The ones offering guidance on how to drive knowledge transfer into action are also scarce. Consequently, this dissertation proposes a knowledge transfer framework. It aims at covering common components (i.e., problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes); providing guidance for planning knowledge transfer activities; and contributing to an effective knowledge transfer amongst software development team members. Unfortunately, owing to time limitations of this study, the usability and practicality of the knowledge transfer framework could not be tested in real-life practice. However, this dissertation demonstrates the application of the framework, using our two prior case studies as a base. The demonstration descriptions are categorized into two parts. The first part describes how the author transferred knowledge to the case study teams; therefore, its focus is on the actual transfer results. The second part describes how the case study team members can transfer knowledge within their teams; hence, its focus is on how the framework can be applied in real-life software development practice. Although the framework cannot yet promise to contribute to knowledge transfer effectiveness in software development, the results of the demonstration show a high degree of compatibility with Scrum-oriented software development. Moreover, the framework was designed and constructed based on the positive results of the case studies in Chapter 5. This implies that there is a great likelihood that the framework is practical in real-life software projects.

As efficient and effective software development processes and a knowledge transfer process are required for quality software development, this dissertation incorporates the software process maintenance framework and the knowledge transfer framework into an umbrella framework. This is a framework for transferring novel software project management concepts into the Thai telecommunications industry. Owing to our time limitations as aforementioned, this dissertation demonstrates how to apply the umbrella framework in software projects; using our prior two case studies as a base again. The results of the demonstration show a great probability that the framework is practical in real-life software projects. At this stage, the umbrella framework partly promises an improvement of software development performance, as a result of the software process maintenance framework. In the future, we hope to carry out more case studies in order to raise more confidence in the usability and practicality of the umbrella framework. At the end of this dissertation, theoretical contributions, implications for future research design, implications for practice, limitations of this study, and recommendations for future work are described. Additional practical tests of the developed frameworks will be carried out by the author after returning to Thailand, before finally handing them over to industry partners.

Zusammenfassung

Heute gehört die Telekommunikation noch immer zu den sich am schnellsten entwickelnden und am härtesten umkämpften Märkten sowie zu einer der weltweit am schnellsten wachsenden Technologiegebiete. Diese Dissertation konzentriert sich auf die thailändische Telekommuniktionsindustrie als Studienobjekt. Bei der Betrachtung der gegenwärtigen Situation der Software Entwicklung in diesem Industriezweig, stützen wir uns auf Interviews mit Hauseigenen und externen Software Entwicklungsteams von zwei der größten Breitband Internet Service Anbietern (Internet Service Providers (ISPs)) in Bangkok, Thailand (ISP1 and ISP2), aus den Monaten März und April 2009. Die Ergebnisse zeigen, daß viele typische Probleme (z.b. das Fehlen einer guten Nutzereinbindung, das Fehlen von Teamarbeit, fehlende Ausbildung, fehlendes Engagement des Managements, fehlende Kompetenz des Projektmanagements, fehlender Wissensaustausch usw.) während des Lebenszyklus des Software Entwicklungsprozesses noch ansteigen. Diese Probleme führen zu deutlich unbefriedigenden Ergebnissen in der Qualität. Diese Situation der Software Entwicklung zeigt, daß es einen Bedarf an effizienten und effektiven Entwicklungsprozessen gibt, sowie Bedarf an unterstützenden Wissenstransfer. Das Ziel dieser Dissertation war es daher, nach praktikablen Lösungen zu suchen, um die Leistung der Software Entwicklung zu verbessern.

Mit Schwerpunkt auf Projektmanagement und Software Entwicklungsprozesse sind geschickte Methoden (z.b. adaptive Software Entwicklung, extreme programming und Scrum) in vielen Geschäftsfeldern weit verbreitet, da sie einen effektiven Software Entwicklungsprozess bieten um diese typischen Probleme zu überwinden. Dennoch bieten sie nur begrenzte Unterstützung für das Projektmanagement (z.b. für Outsourcing und Sicherstellung hoher Qualität) welches das Rückgrat effizienter Software Entwicklung darstellt. Darüber hinaus beschäftigen sie sich damit „wie“ aber nicht „welche“ Software Entwicklungsprozesse implementiert werden sollten. Nur das „wie“ kann nicht garantieren, dass Software-Qualität geliefert wird. Zur Überwindung dieser Probleme schlägt diese Dissertation ein System zur Entwicklung und ständigen Verbesserung des Softwareprozesses vor. Dieses System besteht aus zwei Kernkomponenten. Einem Modell zur Ausreifung der Software Entwicklung um das „was“ zu klären, zur Verbesserung mit einem Software Process Assessment-Mechanismus und einem integrierten PMBOK-Scrum Model zur Klärung des „wie“, eine umfassenden Reihe von Projekt-Management und Software-Entwicklungsprozessen zu implementieren. Um die Anwendung dieses Systems zu unterstützen, wird ein Prototyp-Tool eingeführt. Es wurde als web-basierte Anwendung entwickelt unter Ausnutzung von Java und einer MySQL Datenbank. Es ist wichtig, zu überprüfen, ob das Systems und das Tool in realen Software Projekten praktikabel sind. Daher zeigt diese Dissertation die Implementierung und Ergebnisse im Verlauf von zwei Studien der thailändischen Telekommunikations Industrie (der CAT Telecom Public Company Limited und der TOT Public Company Limited) von November 2010 bis Februar 2011. Die Datenerhebung erfolgte durch Vor-Ort-Beobachtungen, Einzelinterviews und Fragebögen. Die Ergebnisse zeigen das generieren positiver Effekte durch (i) Steigerung der Software-Entwicklung in Bezug auf Effizienz (z. B. Erhöhung der Arbeitsproduktivität) und Effektivität (z. B. getane Arbeit, deren Überprüfung und Bewertung, Verringerung der Fehlerquote und Steigerung der Kundenzufriedenheit und Team-Zufriedenheit) und (ii) Förderung einer Kultur von Teamwork, Zusammenarbeit, regelmäßiger informeller Kommunikation und Wissensaustausch.

Wenn man sich auf den Prozess des Wissensaustausch konzentriert, besteht ein Software Projekt aus wissensintensiven Aktivitäten deren Implementierung Stakeholder Kenntnisse und Erfahrung erfordert, sowie die Lernfähigkeit und die Fähigkeit Wissen anzuwenden um die Probleme zu lösen, die während der Software Entwicklung entstehen. Wissenstransfer selbst hat seine Komponenten und kann auf verschiedene Weisen betrachtet werden (z. B. Prozess basiert, auf die Vorgeschichte basierend und auf die Komponenten basierend). Zwar wurden viele Wissenstransfer Modelle und Studien im Bereich der Softwareentwicklung vorgeschlagen und stehen zur Verfügung um zu lernen; aber sie haben weder einen Schwerpunkt auf die gemeinsamen Komponenten des Wissenstransfer noch liefern sie eine eindeutige und umfassende Beschreibungen oder Darstellung der Beziehungen zwischen diesen Komponenten in einem Wissenstransfer Prozess. Diejenigen, die dazu Anleiten, wie ein Wissenstransfer zu realisieren ist, sind ebenfalls rar. Daher schlägt diese Dissertation ein System zum Wissenstransfer vor (Probleme, Faktoren, Wissen, Mechanismen, Anwendung von Wissen und Ergebnisse). Sie bietet Orientierungshilfen für die Planung von Wissenstransfer Aktivitäten, und den effektiven Wissenstransfer zwischen den Mitgliedern des Software Entwicklungsteams. Aus Zeitgründen konnte die Benutzerfreundlichkeit und Funktionalität des Wissenstransfer Systems leider nicht in der realen Praxis getestet werden. Allerdings zeigt diese Dissertation die Anwendung des Systems mit unserern vorherigen zwei Fallstudien als Basis. Die Demonstrationsbeschreibung ist in zwei Teile unterteilt. Der erste Teil beschreibt, wie der Autor Wissen auf das Teams der Fallstudie übertragt, daher ist der Fokus auf die eigentliche übertragenen Ergebnisse gerichtet. Der zweite Teil beschreibt, wie die Team-Mitglieder der Fallstudie Wissen innerhalb des Teams übertragen. Daher ist der Fokus darauf gerichtet, wie das System an die reale Software-Entwicklung der Praxis angepasst werden kann. Wenn gleich dieses System noch nicht versprechen kann zur Effektivität des Wissenstransfers in der Software Entwicklung beizutragen, so zeigen die Ergebnisse der Demonstration ein hohes Maß an Kompatibilität mit Scrum-oriented software development. Dies impliziert, dass es eine hohe Wahrscheinlichkeit gibt, dass das System in realen Software Projekten also praktikabel erweist.

Da ein effizienter und effektiver Software-Entwicklungsprozesse und ein Wissenstransfer-Prozess für qualitative Software Entwicklung nötigt sind, enthält diese Dissertation das software process maintenance framework und knowledge transfer framwork in einem übergeordneten System, ein System zur Übertragung von Software Projektmanagement in die thailändische Telekommunikationsindustrie. Aufgrund unserer zeitlichen Beschränkungen wie oben erwähnt, zeigt diese Dissertation, wie man das übergeordnete System in Software Projekten anwendet, wieder mit unseren vorherigen zwei Fallstudien als Basis. Die Ergebnisse der Demonstration zeigen eine große Wahrscheinlichkeit, dass das System in realen Software Projekten anwendbar ist. In diesem Stadium verspricht das übergeordnete System eine teilweise Verbesserung der Software-Entwicklungsleistung, als Ergebnis des software process maintenance framework. Wir hoffen in Zukunft mehr Fallstudien durchführen zu können, um mehr Sicherheit beim Nutzen und der Funktionalität des übergeordneten Systems und seiner Komponenten zu gewinnen. Am Ende dieser Dissertation sind theoretische Beiträge, Implikationen für die zukünftige Forschung, Implikationen für die Praxis, die Begrenzungen dieser Studie und Empfehlungen für die künftige Arbeit beschrieben. Weitere Praxistests des hier entwickelten Systems werden von der Autorin nach der Rückkehr nach Thailand durchgeführt, bevor die Arbeit schließlich an Partnern aus der Industrie übergeben wird.

Acknowledgements

My dissertation would not have been possible without the help of several individuals who in one way or another contributed their valuable support in the preparation and completion of this work. First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. DDr. Gerald Quirchmayr, for his excellent guidance, patience, understanding, caring, and providing me with a very encouraging atmosphere for doing research in Vienna, Austria. My utmost gratitude also goes to my Thai expert advisor, Assoc. Prof. Dr. Wichian Chutimaskul, for the time and effort he has invested in supporting my research throughout the past several years, especially with respect to local and industry-specific information and access in Bangkok. My thanks also go to Prof. Dr. Uwe Zdun for his role as second examiner of this dissertation, in which he gave me very valuable methodological and structural feedback. Moreover, I would like to especially thank ÖAD, the Austrian Agency for International Cooperation in Education and Research, and the Higher Education Commission of Thailand for supporting this work in the form of a scholarship for me. Without this support, doing my education abroad would not have been possible.

I would like to thank my colleagues at King Mongkut's University of Technology Thonburi and King Mongkut's University of Technology North Bangkok for their support and comments on my work. Without the substantial support from CAT Telecom Public Company Limited and TOT Public Company Limited it would have been difficult to carry out case studies for evaluating my work. Special thanks go to Bhuchai Mungsommai, Aree Teeraworakul, Jaroon Pakpong, and Chalermphan Lohjarassuriya who gave considerable cooperation and support, which I never had from anywhere before. I would like to thank all individuals who gave me good cooperation to collect data from the field, feedback, and help on my work. I would especially like to thank Amorn Suwantraiamorn who was always willing to help me on programming. Without his support, a prototype tool would have not been done easily.

I would like to thank my good friends, Pornthip Akaranijjirachat, Maiyasit Karyanyam, Kraiwan Punyain, Tharaporn Premjairuthaitawee, Worarat Kruthu, Rangsan Sarikabhuti, and Wuttichai Kwangnakorn. They were always willing to help and give best suggestions, cheer me up, and stand by me through good and bad time.

Last but not least, I would like to thank my family to bring up my inspiration and encouragement with their best wishes at all time. All of my heart, I would also like to express my profound gratitude to the August Virtue of the Buddhist Triple Gems and Universal Sacredness for always giving me the strength and guiding me proper ways to move forward through the sometimes very challenging and tough time with consciousness, wisdom, and good deeds. For this work, I must clarify that the blame for possible errors in this work lies with me alone.

Table of Contents

List of Figures

Figures are numbered by repeating the chapter number followed by a dash and the sequential number of the figures in that chapter. Hence, a figure numbered 2-1 is the first figure in Chapter 2.

List of Tables

Tables are numbered by repeating the chapter number followed by a dash and the sequential number of the tables in that chapter. Hence, a table numbered 4-2 is the second table in Chapter 4.

List of Abbreviations

AIS	Advanced Info Service Public Company Limited
AM	Agile Modeling
APMM	Agile Process Maturity Model
ASD	Adaptive Software Development
CAT	CAT Telecom Public Company Limited
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CMMI-DEV	Capability Maturity Model Integration for Development
COBIT	Control Objectives for Information and Related Technology
CPM	Critical Path Method
CSF	Critical Success Factor
DSDM	Dynamic Systems Development Method
DSS	Decision Support Systems
DTAC	Total Access Communication Public Company Limited
EIS	Executive Information System
ERP	Enterprise Resource Planning
eTOM	enhanced Telecom Operations Map
FAB	Fulfillment, Assurance and Billing
FDD	Feature Driven Development
ICT	Information and Communication Technology
IDEAL	Initiating, Diagnosing, Establishing, Acting, and Leveraging
IS	Information System
ISO/IEC 15504	International Organization for Standardization and the International Electrotechnical Commission 15504 Standard
ISP	Internet Service Provider
IT	Information Technology
IU	Intention to Use
KPA	Key Process Area
MIS	Management Information System
MPCU	Model of Personal Computer Utilization
MSF	Microsoft Solutions Framework
OPEN	Object-oriented Process, Environment and Notation

OSR	Operations Support & Readiness
PEOU	Perceived Ease of Use
PERT	Program Evaluation and Review Technique
PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
PRINCE2	Projects in Controlled Environments
PSP	Personal Software Process
PU	Perceived Usefulness
QIP	Quality Improvement Paradigm
RUP	Rational Unified Process
SCAMPI	Standard CMMI Appraisal Method for Process Improvement
SDM	Software Development Maturity
SEI	Software Engineering Institute
SIP	Strategy, Infrastructure & Product
SME	Small-to-Medium-sized Enterprise
SPAD	Software Process Assessment and Development
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability Determination
TAM	Technology Acceptance Model
TDD	Test Driven Development
TFS	Team Foundation Server
TOT	TOT Public Company Limited
TPB	Theory of Planned Behavior
TPS	Transaction Processing Systems
TRA	Theory of Reasoned Action
TRUE	True Corporation Public Company Limited
UML	Unified Modeling Language
UTAUT	Unified Theory of Acceptance and Use of Technology
WBS	Work Breakdown Structure
XML	eXtensible Markup Language
XP	eXtreme Programming

Chapter 1

Introduction

The focus of this dissertation is to develop an overarching framework for transferring software project management approaches into the Thai telecommunications industry, with the aim of contributing to the improvement of software development performance. The framework does itself consist of two components which are frameworks themselves. They are a software process maintenance framework providing guidance for assessing, planning, and improving project management and software development processes, and a knowledge transfer framework providing guidance for planning knowledge transfer activities. The introductory chapter gives the background to the research, explaining why the study is important. This chapter then provides an organization of the dissertation, explaining the logical structure and layout used to develop the research from the literature review towards the conclusions of this study.

1.1 Introduction

Telecommunications is still one of the most rapidly evolving competitive markets and one of the fastest-growing areas of technology in the world. This study focuses on the Thai telecommunications industry as a case study. Thailand's telecommunications industry is worth mentioning, as it has continued to experience stable growth. As recently reported, Thailand became the second fastest growing broadband market in the world and led all Asian countries surveyed with a 67% annual growth rate from the first quarter of 2010 to the first quarter of 2011 [1]. Considering software development situations in this industry, software development teams are facing very typical problems, e.g. a lack of agile logistical arrangement, a lack of good user participation, a lack of management commitment, a lack of project management competence, a lack of teamwork, a lack of training support, a lack of knowledge transfer, and etc [2]. These problems significantly result in unsatisfactory quality results. This emphasizes that there is a lack of efficiency and effectiveness of software development processes (hereafter referred to as "software process") and knowledge transfer.

The best known traditional software development method is still the waterfall method, which in fact is the oldest original method. It is a systematic and sequential pattern reaching from an initial feasibility study to the maintenance of the developed information systems. However, there are several limitations, e.g. the necessity of having well-defined requirements, being time-consuming, needing too much documentation and resulting in a high cost [3]. Agile software development methods, such as Adaptive Software Development (ASD), Agile Modeling (AM), Crystal family, Dynamic Systems Development Method (DSDM), eXtreme programming (XP), Feature Driven Development (FDD), and Scrum were thus developed to overcome those limitations. They have gained recognition in the software development community due to their response to market expectation, i.e., innovative and high quality software [4]. In an increasingly competitive world, software development methods should be efficient [5]. Efficiency requires project management activities to enable the proper execution of software development tasks [6]. Project management thus provides the backbone

for efficient software development [7]. From this view, some agile methods (e.g., ADS, Crystal, DSDM, FDD, and Scrum) are supplemented with guidelines on project management that allow for the rapid delivery of quality software products. Nevertheless, in the general sense, there is no comprehensive project management support [8]. In other words, they offer limited project management support, e.g., for outsourcing, developing with large teams, developing software that demands high quality control, and distributed development environments [4, 9-11]. Regarding the benefits of standards, using existing standards to develop the new framework instead of creating a new one can save a lot of time and effort [12]. Although researchers such as Turk et al. [11] suggest that traditional project management practices are an applicable way. So far, little attention has been paid to such integrated traditional project management into agile software development methods to overcome inadequate project management support.

Besides, a software development method generally deals with how it can be implemented, but not so much with which software processes should be implemented. Thus, only the "how" cannot guarantee that software quality will be delivered. The quality of software depends on the quality of a software process [13, 14] that result from Software Process Improvement (SPI) [15]. Albeit approaches to SPI such as the Capability Maturity Model Integration (CMMI) and ISO/IEC 15504 (the International Organization for Standardization and the International Electrotechnical Commission 15504 standard, which is also known as SPICE: Software Process Improvement and Capability Determination) are a challenge to organizations trying to improve the quality of software processes and software products, there has been only limited success in many SPI programs with a 70% failure rate [16, 17]. This is because these approaches just explain critical attributes that would be expected to characterize an organization at a particular maturity level [18]. They have not suggested how to improve a given status to get to a particular maturity level [18-20]. Nor have they tackled the issues on how to elicit and model processes in order for software projects to follow specific development processes, or how to gather project practices and knowledge for SPI, and how to deal with existing problems [18].

Focusing on SPI in agile software development, much attention has been paid to how SPI and agile methods can be applied together by mapping SPI methods' Key Process Areas (e.g. CMMI KPAs) and agile practices [21-25], and how to assess agile software development by mapping CMMI goals and agile practices [26, 27]. Even though there is an agile process maturity model by IBM [28], it is different from traditional SPI approaches as its objective is mainly to improve process visibility and adaptability and fit them to the surrounding organizational objectives. Traditional SPI approaches mostly aim at improving process repeatability and predictability. Moreover, although a vast body of literature cites factors that have an impact on successful agile software development, little attention has been paid to how to deal with those influential factors. This suggests that the current problems with SPI in agile software development still result from a lack of mechanisms to overcome the above limitations.

Furthermore, knowledge transfer and its application can significantly contribute to software project success. A software project is characterized by frequent changes and its implementation requires effective activities, stakeholders' expertise and experience, and the ability to transfer, acquire, and apply knowledge to problems occurring during software development [29]. Without using the existing knowledge, team members have to create new solutions to every occurring problem. The existing knowledge includes implemented software processes, experience, and knowledge gained during prior software development. It becomes apparent that transferring and applying new knowledge is crucial for creating

innovative software development and competitive software products. This supports the fact that software development is a knowledge-intensive activity [30]. Moreover, success in producing quality software demands the presence of sufficient knowledge on software development teams [31]. Therefore, a software project requires knowledge transfer to ensure that the software project will not get a hard landing.

These points of views have led us to a set of research questions together with research approaches organized by Chapters and summarized in Table 1-1. In this table, research questions are numbered by repeating the chapter number followed by a dash and the sequential number of the research questions in that chapter. Hence, a Research Question (RQ) numbered 2-1 is the first research question in Chapter 2.

Table 1-1. A summary of research questions and research approaches

Chapter	Research Question	Research Approach
Chapter 2	RQ2-1: Do the problems identified in prior research on executive information system development in Thailand currently still exist?	1. A literature review on the Thai executive information systems development in Thailand was performed to consider its problems. 2. Semi-structured interviews were carried out with two Thai software development teams: in-house and outsourcing teams working for two of the largest broadband Internet Service Providers (ISPs) in Bangkok, Thailand (named ISP1 and ISP2) during March and April 2009.
	RQ2-2: Do the problems in the current EIS development in Thailand involve project management, software development, and knowledge transfer aspects?	
	RQ2-3: What are the factors affecting the successful agile software development?	A literature review on influential factors affecting the successful agile software development was performed.
	RQ2-4: What are the factors affecting the successful knowledge transfer in software development?	A literature review on influential factors affecting the successful knowledge transfer was performed.
	RQ2-5: What could our conceptual software process maintenance framework based on its requirements look like?	The principles of the Project Management Body of Knowledge (PMBOK) and two models of Scrum and CMMI are used as a basis.
	RQ2-6: What could our conceptual knowledge transfer framework based on its requirements look like?	Szulanski's knowledge transfer model is used as a basis.
Chapter 3	RQ3-1: Which existing research results on agile software development integration with software process improvement and with traditional project management are available that we can build on?	A systematic literature review on agile software development integration with software process improvement and with traditional project management was performed.
	RQ3-2: What are some interesting aspects that existing research results on agile software development integration with software process improvement and with traditional project management do not yet cover?	

Chapter	Research Question	Research Approach
	RQ3-3: How should a software process maintenance framework be constructed? Is a software process maintenance framework workable? What does the test of a software process maintenance framework in a real-life situation contribute?	
Chapter 4	RQ4-1: Are CSFs in software development, as identified in Table 2-4 in Chapter 2, similar to CSFs in SPI identified in the literature?	1. A literature review on 24 sources including reports, case studies, and software process articles was performed to investigate influential factors in software process improvement practices that are recognized internationally. 2. A frequency analysis was used to extract quantitative data from the collected qualitative data
	RQ4-2: What agile practices, as identified in the literature on agile software development and data quality, should be implemented for successful software development?	1. A literature review on 31 sources including reports, case studies, and software process articles was performed to investigate influential factors in agile practices that are recognized internationally. 2. A frequency analysis was used to extract quantitative data from the collected qualitative data
	RQ4-3: What agile practices, as identified in our questionnaire-style information collection, should be implemented for successful software development?	1. A questionnaire-style information collection was carried out in three companies in Thailand, including telecommunications player and co-players in order to investigate their agile practices. 2. The median values were used to analyze data.
	RQ4-4: How should a software development maturity model (as one of two core components of a software process maintenance framework) be constructed?	CMMI and CSFs are used as a basis.
	RQ4-5: How should an integrated PMBOK-Scrum model (as one of two core components of a software process maintenance framework) be constructed?	PMBOK and Scrum are used as a basis.
Chapter 5	RQ5-1: How can the developed software process maintenance framework be executed efficiently and effectively in the given context?	Basic ideas of field case study (e.g., direct observations, interviews, and questionnaires to discuss the challenges within the context, the results achieved, and the lessons learned) were used to test a software process maintenance framework in CAT Telecom Public Company Limited during November - December 2010 and TOT Public Company Limited during December 2010 - February 2011.
	RQ5-2: What are the challenges that impact software development, using the developed software process maintenance framework?	
	RQ5-3: What changes are necessary to adapt the developed software process maintenance framework?	
	RQ5-4: How do practitioners transfer new knowledge into their existing software processes?	

Chapter	Research Question	Research Approach
	RQ5-5: What is the developed software process maintenance frameworks perceived usefulness and ease of use?	
	RQ5-6: What are the requirements for successful adaptation of the developed software process maintenance framework?	
Chapter 6	RQ6-1: What are the differences in how knowledge transfer is defined in the prior literature and what can we learn from these differences?	A literature review on knowledge transfer in software development and other contexts (e.g., intra-firm knowledge transfer) was performed.
	RQ6-2: How does each individual knowledge transfer component interact with others?	
	RQ6-3: What are the missing points in the prior literature on knowledge transfer in software development?	A literature review on the 27 highly visible knowledge transfer studies in software development was performed.
Chapter 7	RQ7-1: How should a knowledge transfer framework be constructed?	Szulanski's knowledge transfer model and TAM are used as a basis.
	RQ7-2: What knowledge transfer activities under each of the six knowledge transfer components (i.e., problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes) should be implemented?	Literature review on knowledge transfer and relevant studies (e.g., knowledge management and knowledge acquisition) were performed.
	RQ7-3: How do knowledge transfer activities play an important role in each of the four knowledge transfer stages (i.e., Initiation, Implementation, Ramp-up, and Integration)?	
	RQ7-4: How can the developed knowledge transfer framework be performed?	The findings of two case studies in Chapter 5 were used to demonstrate the application of a knowledge transfer framework.
Chapter 8	RQ8-1: How should a framework for transferring software project management approaches into the Thai telecommunications industry be constructed?	A software process maintenance framework presented in Chapter 4 and a knowledge transfer framework presented in Chapter 7, are used as a basis.
	RQ8-2: How can the developed framework for transferring software project management approaches into the Thai telecommunications industry be performed?	The findings of two case studies in Chapter 5 were used to demonstrate the application of the framework for transferring software project management approaches into the Thai telecommunications industry.

In relation to the above set of research questions, research approaches, and their research results, a set of main theoretical and empirical Contributions (C) to the research field of software development can be summarized as follows.

C1: The identified Critical Success Factors (CSFs) affecting the successful software development and knowledge transfer were used as requirements for constructing a software process maintenance framework (presented as C3) and a knowledge transfer framework (presented as C6).

C2: Overviews of the literature on agile software development integration with software process improvement and with traditional project management, and knowledge transfer were created. This enables us to identify what had been investigated and to extract some interesting aspects by which these research results should be extended or conducted for a software process maintenance framework (presented as C3) and a knowledge transfer framework (presented as C6).

C3: The software process maintenance framework was developed by paying attention to the "what" to improve through a software development maturity model and the and "how" to implement software processes through an integrated PMBOK-Scrum model. The software development maturity model was created based on CMMI and the CSFs affecting the successful agile software development identified in C2. It has a threefold objective: to appraise an organization's current software process through the identified CSFs, to get the current maturity level rating from the model, and to identify which software processes demand immediate and sustainable improvement. The integrated PMBOK-Scrum model was created based on PMBOK and Scrum approaches. It assists practitioners in implementing integrated project management and software development processes.

C4: The prototype tool was developed as a Web-based application, using the Java language and a MySQL database, to support the use of the software process maintenance framework. It helps an end user (e.g., a project manager and a team leader) to get insight into the organization's current maturity by assessing the identified CSFs through the list of practices required by the software development maturity model. Weak practices as a part of assessment results will be used to plan the project together with the defined information (e.g., project, phase, and activity) required by the integrated PMBOK-Scrum model. The defined software process is then validated to ensure its appropriateness and prepared in an eXtensible Markup Language (XML) file format for export to the organization's project planning tools.

C5: The software development performance in terms of efficiency and effectiveness, using the software process maintenance framework, was improved in two major telecommunications players in Thailand. This is based on the overall evaluation results of the software process maintenance framework through two case studies at CAT Telecom Public Company Limited and TOT Public Company Limited in Thailand from November 2010 to February 2011. The data collection was carried out through on-site observations, individual interviews, and questionnaires.

C6: The knowledge transfer framework was designed and developed based on the knowledge transfer CSFs identified in C2, the findings of C5, and Szulanski's model. It is drawn on the connectionistic perspective and communication-based research on knowledge transfer. It aims at providing guidance for planning knowledge transfer activities. In the framework, the six components of knowledge transfer (i.e., problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes), the four stages of knowledge transfer (i.e., Initiation, Implementation, Ramp-up, and Integration), and the relationships between the components and the stages are elaborately described. Owing to

time limitations of this study, we could not carry out additional empirical case studies in the Thai telecommunications industry to evaluate the framework. Consequently, the findings of C5 were used to demonstrate the application of the framework.

C7: The framework for transferring software project management approaches into the Thai telecommunications industry was developed, called the umbrella framework. It aims at contributing to the improvement of software development performance in terms of efficiency and effectiveness. It consists of two core components which are frameworks themselves: the developed software process maintenance framework (presented as C3) and the developed knowledge transfer framework (presented as C6). Owing to time limitations of this study as mentioned, the findings of C5 were used to demonstrate the application of the umbrella framework.

Furthermore, some of our main contributions were published as follows.

P1: Porrawatpreyakorn, N., Quirchmayr, G., and Chutimaskul, W. 2009, 'Requirements for a Knowledge Transfer Framework in the Field of Software Development Process Management for Executive Information Systems in the Telecommunications Industry', in Papasratorn, B., Chutimaskul, W., Porkaew, K., and Vanijja, V. (eds), Proceedings of the 3rd International Conference on Advances in Information Technology, Springer Berlin Heidelberg, Bangkok, Thailand, vol. 55, pp. 110-122.

P2: Porrawatpreyakorn, N., Quirchmayr, G., and Chutimaskul, W. 2010, 'Requirements for a Software Process Maintenance Framework for Executive Information Systems in the Telecommunications Industry', Journal of Global Management Research, vol. 6. No. 1, pp. 7-18.

P3: Porrawatpreyakorn, N., Quirchmayr, G., and Chutimaskul, W. 2010, 'A Prototype for the Support of Integrated Software Process Development and Improvement', in Papasratorn, B., Chutimaskul, W., Porkaew, K., and Vanijja, V. (eds), Proceedings of the 4th International Conference on Advances in Information Technology, Springer Berlin Heidelberg, Bangkok, Thailand, vol. 114, pp. 94-105.

Providing a clearer view, the connection between chapters, research questions, contributions, and publications is presented in Table 1-2 and Figure 1-1.

Table 1-2. A relation between chapters, research questions, contributions, and publications

Chapter	**Research Question**	**Contribution**	**Publication**
Chapter 2	RQ2-1 to RQ2-4	C1	P1, P2
	RQ2-5	C3	P1, P2
	RQ2-6	C6	P1
Chapter 3	RQ3-1 to RQ3-3	C2	-
Chapter 4	RQ4-1 to RQ4-5	C3, C4	P3
Chapter 5	RQ5-1 to RQ5-6	C5	-
Chapter 6	RQ6-1 to RQ6-3	C2	-
Chapter 7	RQ7-1 to RQ7-4	C6	-
Chapter 8	RQ8-1 and RQ8-2	C7	-

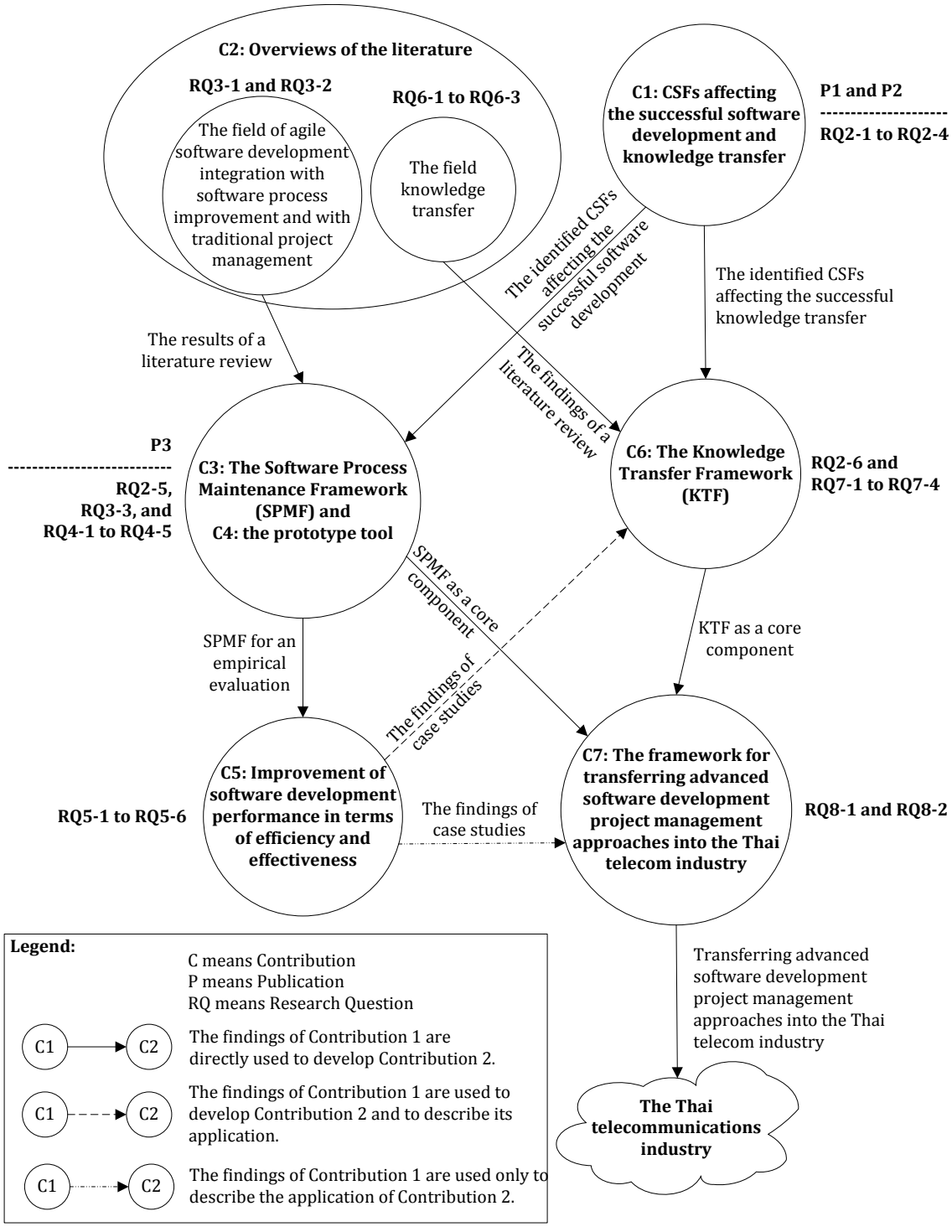


Figure 1-1. Main contributions of this study

1.2 Organization of the Dissertation

This dissertation is structured into nine chapters and two additional sections as follows.

Chapter 1 presents the background to the research, covering our motivation and the research questions. This chapter also establishes why this research is important, states the goals and objectives of the research, and closes with the organization of the dissertation.

Chapter 2 presents an idea of the current situation in the software development process management for Executive information Systems (EISs) in the Thai telecommunications industry by using findings of interviews with in-house and outsourcing software development teams working for two Internet Service Providers in Thailand. This chapter also describes influential factors affecting successful software development addressed in the literature by focusing on two main parts, i.e., software management and development and knowledge transfer. This is because both parts significant contribute to the improvement of software development performance. Based on the findings, this chapter identifies two sets of requirements, i.e., one for the proposed software process maintenance framework and one for the proposed knowledge transfer framework. Both frameworks are core components of the proposed framework for transferring software project management approaches into the Thai telecommunications industry.

Chapter 3 delves into the prior literature that forms the foundation of the proposed software process maintenance framework. The literature review in the fields of agile software development integration with software process improvement and with traditional project management presents what research results are available to build on, some interesting aspects that those research results do not yet cover, and how to construct the workable software process maintenance framework.

Chapter 4 presents a software process maintenance framework, advocating software process improvement through a software development maturity model and providing a comprehensive set of project management and software development processes through an integrated PMBOK-Scrum model. The framework consists of two main components. The first component is the proposed software development maturity model which is based on CMMI, the CSFs affecting the successful agile software development identified in Chapter 2, the findings of the literature review in Chapter 3, and the findings of (i) a literature survey on worldwide agile software projects (ii) a questionnaire-style information collection on agile practices in three companies in Thailand including a telecommunications player and two co-players. The questionnaire-style information collection was conducted in June 2010. The data was collected from respondents who had been doing agile software development on a daily basis. The second component is the proposed integrated PMBOK-Scrum model. It is derived from the PMBOK principles which are then merged with ideas borrowed from the Scrum model. Supporting the application of the framework, a prototype tool has been introduced.

Chapter 5 presents the usability and practicality of the developed software process maintenance framework through case studies in the Thai telecommunications industry. To get accurate results, our case studies are focused on direct players in the industry. One of the original target companies was the direct player where we were focused on in Chapter 4. Unfortunately, obtaining its permission to test the developed software process maintenance framework was not achieved. Instead, two case studies in CAT Telecom Public Company Limited (CAT) and TOT Public Company Limited in Thailand (TOT) were carried out from November 2010 to February 2011. The findings present how to execute software

development efficiently and effectively, challenges impacting the project results, changes necessary to adapt the framework, how to integrate new knowledge into the existing software processes, and requirements for successful adaptation of the framework.

Chapter 6 delves into the prior literature that forms the foundation of a knowledge transfer framework. The literature review is presented in three sections. The first section examines what the differences are in how knowledge transfer is defined in the knowledge transfer literature and what we can learn from those differences. As knowledge transfer has its components, the second section thus scrutinizes how its individual components interact amongst them. The third section highlights what the differences are in the 27-highly-visible literature on knowledge transfer in software development.

Chapter 7 presents a knowledge transfer framework, based on the knowledge transfer CSFs identified in Chapter 2, the findings of the case studies in Chapter 5, the findings of the literature review in Chapter 6, and Szulanski's model. It aims at providing guidance on planning knowledge transfer activities. This chapter describes knowledge transfer into three main sections. The first section elaborates six components of a knowledge transfer process which are problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes. A set of activities under each component and a set of questions under each of those activities have been designed for suggestions in planning. The second section elaborates four stages of the knowledge transfer process which are Initiation, Implementation, Ramp-up, and Integration stages. An activity flow has been designed under each stage, based on its functionality. For a better understanding, this chapter demonstrates the application of the framework, based on the findings of the case studies in Chapter 5.

Chapter 8 presents the developed framework for transferring software project management approaches into the Thai telecommunications industry, by pulling the developed software process maintenance framework presented in Chapter 4 and the developed knowledge transfer framework presented in Chapter 7 together. For a better understanding, this chapter demonstrates the application of the framework, based on the findings of the case studies in Chapters 5.

Chapter 9 summarizes the findings of this study. Drawing on the whole research project, this chapter summarizes theoretical contributions, theoretical implications, and empirical implications in relation to the problem areas. The concluding chapter also provides suggestions for further research.

The last chapter is followed by two sections which contain supporting information for this dissertation. This supporting information includes a complete reference list of all sources cited in the dissertation and a set of appendices facilitating the more detailed understanding of this dissertation.

Chapter 2

Requirements for a Framework for Transferring Software Project Management Approaches into the Thai Telecommunications Industry

Related Publications:

P1: Porrawatpreyakorn, N., Quirchmayr, G., and Chutimaskul, W. 2009, 'Requirements for a Knowledge Transfer Framework in the Field of Software Development Process Management for Executive Information Systems in the Telecommunications Industry', in Papasratorn, B., Chutimaskul, W., Porkaew, K., and Vanijja, V. (eds), Proceedings of the 3rd International Conference on Advances in Information Technology, Springer Berlin/Heidelberg, Bangkok, Thailand, vol. 55, pp. 110-122.

P2: Porrawatpreyakorn, N., Quirchmayr, G., and Chutimaskul, W. 2010, 'Requirements for a Software Process Maintenance Framework for Executive Information Systems in the Telecommunications Industry', Journal of Global Management Research, vol. 6, no. 1, pp. 7-18.

This chapter presents the interview findings of the current situation in software development for Executive Information Systems (EISs) in two of the largest broadband Internet Service Providers (ISPs) in Bangkok, Thailand, named ISP1 and ISP2. This chapter then identifies requisite requirements for a framework for transferring software project management into the Thai telecommunications industry which consists of two core components that are frameworks themselves. They are the proposed software process maintenance framework and the proposed knowledge transfer framework. The findings reveal that software development teams do not perceive formal routines as an efficient and effective way to manage software development processes, to deliver quality results, and to transfer knowledge. However, the quality of software depends not only on an efficient and effective project management and software development process, but also on an effective knowledge transfer amongst software development team members. Efficiency requires project management activities to enable the proper task execution. Existing agile methods (e.g., Scrum and eXtreme Programming) offer effective software development processes, but inadequate support for project management (e.g., limited support for subcontracting and developing software that demands high level of quality control). Hence, this study proposes a software process maintenance framework, covering adequate project management and software development perspectives. As the development of a software project requires the presence of sufficient expertise and experience of stakeholders, this study also proposes a knowledge transfer framework providing guidance for driving knowledge transfer into action. Therefore, the overall goal of the resulting frameworks is to contribute to the improvement of software development performance in terms of efficiency and effectiveness.

2.1 Introduction

The best known traditional software development method still is the waterfall method, which in fact is the oldest original method. It is a systematic and sequential pattern reaching from an initial feasibility study to the maintenance of the developed Information Systems (ISs). Nevertheless, there are several limitations (e.g., the necessity of having well-defined requirements, being time-consuming, needing too much documentation and resulting in a high cost [3]). Agile software development methods (e.g., eXtreme programming (XP), Dynamic Systems Development Method (DSDM), Feature Driven Development (FDD), and Scrum) were thus developed to overcome these limitations. They are gaining recognition in the software development community due to their response to market expectation, i.e., innovative and high quality software [4]. Moreover, Critical Success Factors (CSFs) for successful agile software development are identified by a multitude of studies [32-35]. However, software development methods should be efficient [5]. Efficiency requires project management activities to enable the proper execution of software development tasks. Project management thus provides the backbone for efficient software development [7]. From this view, some agile methods (e.g., DSDM, FDD, and Scrum) are supplemented with guidelines on project management that allow for the rapid delivery of quality products. Nevertheless, in the general sense, there is no comprehensive project management support [6]. Scrum, which is definitely the most popular [36], offers limited project management support (e.g., for subcontracting, developing with large teams, developing software that demands high quality control, and distributed development environments [4, 9-11]). Although researchers such as Turk et al. [11] suggest that traditional project management practices are an applicable way, so far no integrated method offering adequate project management support to overcome these Scrum's limitations has been identified. Project Management Body of Knowledge (PMBOK) is the broadest and most widely used standard reference of industry best project management practices [37], and definitely compatible with agile ways [38]. Hence, there is a need to build an integrated PMBOK-Scrum approach.

Besides, the quality of the software development process (hereafter referred to as "software process") results in the quality of software [13]. A software process generally deals with how it can be implemented, but not so much with what software processes should be implemented. Thus, only the "how" cannot guarantee that software quality will be delivered. Software Process Improvement (SPI) can produce the quality of the software process [15] that results in software quality [14]. Capability Maturity Model Integration (CMMI) is a well-accepted model for improving the performance of software development processes and software quality, and referring to what software processes should be implemented to achieve successful software development [39]. Consequently, CMMI is considered an efficient way to maintain the quality of software processes. Furthermore, knowledge transfer is crucial since a software project typically consists of multiple stakeholders with diverse backgrounds and skill sets. Talents in software development teams (hereafter referred to as "teams") should continuously complement each other for better work efficiency and effectiveness [40]. Besides, a knowledge transfer amongst team members means that software can be optimized for improved efficiency and effectiveness above or beyond what any individual can achieve [41]. Consequently, it can be concluded that quality software depends upon quality software processes and knowledge transfer.

Arnott et al. [42] said that many developing countries are investing in Information Technology (IT), especially the newly industrialized countries (e.g., Thailand, India, China, Turkey, and South Africa, according to the International Monetary Fund's World Economic

Outlook Report, April 2011 [43]) to support their businesses in highly competitive markets. It follows that many large organizations (e.g., Transport Company, Banks, and Energy Company in Thailand [42], Major Railway Corporation, International Airline and Large University in China [44], and Food Company, Soft Drinks Company, and Consumer Packaged Goods Company in Turkey [45]) have developed or tend to consider developing Executive Information Systems (EISs) to support their senior management. This chapter therefore focuses on EIS development. Recently, the use of EISs has significantly increased since the success of EIS in developed countries stimulates a number of executives to adapt this IS into their organizations in order to compete in an increasingly competitive environment. EISs are different from Transaction Processing Systems (TPSs), Management Information Systems (MISs), and Decision Support Systems (DSSs) in terms of problems addressed, users, and data used. TPSs serve operational management by performing and recording the daily routine transactions necessary to conduct the business and solve structured problems which have standard solutions. MISs and DSSs serve middle management. However, there are different characteristics for the way in which MISs deal with summarized and compressed data from TPSs and sometimes perform an analysis of that summarized data to solve structured problems. On the other hand, DSSs use data from TPSs, MISs, and external sources to solve semi-structured problems whereby only part of the problem has a structured quality. EISs provide information for top management to solve unstructured problems which have no standard solutions for resolving the situation, so that they can identify problems and opportunities by combining internal and external information that is relevant to decision making [3, 46]. EISs can directly aid and support communications, coordination, planning and control functions of managers and executives in an organization. Supporting this, Nord and Nord [47] argue that utilizing EIS software can provide valuable benefits (i.e., better communication, increased confidence in decision making, and eventually increased profits). In addition, Telecommunications is still one of the most rapidly evolving competitive markets and one of the fastest-growing areas of technology in the world. Thailand's telecommunications industry is worth mentioning that it has continued to experience stable growth. As reported, Thailand became the second fastest growing broadband market in the world and led all Asian countries surveyed with a 67% annual growth rate from the first quarter of 2010 to the first quarter of 2011 [1].

EISs require fundamental revision and software development methods that must be able to deal with rapid evolution. Unfortunately, EIS development in Thailand is likely to be more difficult due to difficult software development environments, e.g., economic and volatile political environments, organizational cultures, a lack of user participation, and inappropriate software development methods [42, 48]. For understanding and dealing with the problems, six fundamental research questions of this chapter are listed as follows.

RQ2-1: Do the problems identified in prior research on executive information system development in Thailand currently still exist?

RQ2-2: Do the problems in the current EIS development in Thailand involve project management, software development, and knowledge transfer aspects?

RQ2-3: What are the factors affecting the successful agile software development?

RQ2-4: What are the factors affecting the successful knowledge transfer in software development?

RQ2-5: What could our conceptual software process maintenance framework based on its requirements look like?

RQ2-6: What could our conceptual knowledge transfer framework based on its requirements look like?

This chapter is organized by starting with the descriptions of the current situation in EIS development of two broadband Internet Service Providers (ISPs) in Thailand, using a qualitative analysis via interviews. The following sections describe two primary focuses of this study, theoretical foundations of this study, and the identified influential factors as requirements for a framework for transferring software project management approaches into the Thai telecommunications industry. A proposed conceptual framework based on the identified requirements is then introduced.

2.2 Look at the Current Situation in Executive Information Systems Development in the Thai Telecommunications Industry

For getting an idea of the current situation in the EIS development in the Thai telecommunications (by focusing on Internet services), we use findings of interviews with in-house and outsourcing teams working for two of the largest broadband ISPs in Bangkok, Thailand. To preserve their anonymity, we refer to them here as ISP1 and ISP2. However, the size of companies does not affect the model of EIS development. The data was collected during March and April 2009. There are two main reasons to choose these two companies. First, they have experience in EIS development. Second, they are the two largest broadband ISPs in the Bangkok region and have their own optical fiber cable networks in Bangkok and in the vicinity. Even though there are many ISPs in Thailand, most of them still lease bandwidth from one of these two companies. Owing to a very small sample size, we do not claim that it is a representative sample. In other words, this threatens the generalizability of the results. To reduce this treat to some extent, we interviewed (1) practitioners who had experience in EIS development and dealt with software development on a daily basis; (2) practitioners who had different roles (e.g., including project manager, developer, and coordinators); (3) practitioners who used different software development methods (i.e., waterfall and outsourcing methods); and (4) practitioners who worked with different team sizes (i.e., small and large teams). In order to reduce treats to reliability and validity, main questions about the implemented software process, the environments of EIS development, the problems occurring during EIS development, and the solutions to deal with those problems (presented in Appendix A) were answered in semi-structured interviews. This is also in order to increase comparability of responses and facilitate an analysis of the data. Additionally, the practitioners' experiences and perceptions were explored independently and without any suggestion from the authors. The interviews ranged in length from one to two hours.

In the organizational context of EIS development, the findings reveal that the executives could sometimes not provide adequate participation in the projects. Subordinates did not have full authorities when it came to making decisions. Communication processes during EIS in organizations were also quite complicated (e.g., executives or users do neither have good cooperation nor do they participate well). These limitations resulted in development teams sometimes not being able to identify the information requirements from executives effectively, often having to wait for Steering Committee decisions, and resulted in an extensive organizational process. As was to be expected, the projects were delayed.

Given the underlying EIS development strategies of prototyping and outsourcing, the EIS development project with a small team in ISP1 had a short duration. ISP1 used a prototyping model. The software processes involved requirements analysis, preliminary

design, prototype design, construction and testing, implementation and maintenance. In the other case, the EIS development project with a large team in ISP2 had an initial period of two years. Outsourcing usually covers a wide range of contractual arrangements ranging from contract programmers to third party facilities management [49]. The EIS development in ISP2 was to some extent outsourced. One of the reasons for employing the consultant was that the internal staff lacked knowledge and experience in EIS development. Although the development methodology used terms like prototyping and module delivery, it can best be characterized as a variant of the waterfall approach. The development process involved a large execution of requirements analysis, system development, user acceptance, system installation, and maintenance. During EIS development, the teams face similar problems. For example, users between business units neither have good cooperation nor do they participate well; users provide inadequate requirement specifications and quite frequently change their requirements; users have only limited IT/IS skills; and so on. This situation is quite typical, and not limited to the Thai telecommunications industry. Knowledge transfer practices were performed in a very similar way, e.g., by discussing and sharing ideas in regular meetings; transferring theoretical knowledge by self-learning that is based on the existing internal documents; providing practical training case by case during EIS development; and finally supporting theoretical training prior to project for specialists.

Additionally, data quality has a great impact on the overall quality of software. Data quality is defined as "data that are fit for use by data consumers" [50]. A basic set of data quality dimensions includes accuracy, completeness, timeliness, and consistency [51]. According to Wang and Strong [50], accuracy is defined as the degree to which data is correct, reliable, and certified free of error. Completeness is defined as the degree to which data is of sufficient breadth, depth, and scope for the task at hand. Timeliness can be defined as the degree to which the age of data is appropriate for the task at hand. Consistency is defined as the degree to which the representation of the data value is the same in all cases [52]. Therefore, data quality can determine success or failure of software development. Moreover, successful software development or software projects should consider internal features (i.e., stakeholders and policy, and development methodology) and external features (i.e., Information and Communication Technology (ICT), and the environment) [53]. The stakeholders and policy feature is referred to the quality of organization and people. The development methodology feature is referred to the quality of software processes. The quality of ICT feature must consider ICT competency, vendor support, and ICT personal, whilst the quality of environment feature must consider external factors (i.e., requirement volatility). Concentrating on investigating current problems in EIS development, the problems found and the data quality aspect can be summarized in Table 2-1 as failure factors.

Table 2-1. The failure factors in EIS development

Dimension	**Failure Factor**
Organization	Lack of management commitment, organizational culture too traditional, lack of agile logistical arrangement
People	Lack of necessary skill-set, lack of project management competence, lack of good user participation and cooperation, lack of teamwork
Process	Ill-defined project scope, requirements, and planning, user team having no full authority
Technology	Lack of provision and support of training to teams, inappropriateness of methods and tools
Project	Unsuitable team size
Data	Lack of data quality (e.g., inconsistent data, contradictory data, redundant data, missing data,

Dimension	Failure Factor
	and out of date data)

Prior research (i.e., [42, 48]) identified the following problems in EIS development in Thailand: the low level of user participation in design and development, a lack of EIS development knowledge and experience, a lack of knowledge transfer, inappropriate software development methods, political and economic pressures, and organizational cultures. This supports the findings to answer the RQ2-1 that almost all of those problems currently still exist in EIS development in the Thai telecommunications context. The findings also help answer the RQ2-2 that the problems identified in Table 2-1 involve project management, software development, and knowledge transfer perspectives.

2.3 Two Primary Focuses of this Study

As the current problems in EIS development involve project management, software development, and knowledge transfer aspects, the primary focuses of this study are agile software development and knowledge transfer. The main reason we concentrate on agile software development is that agile methods (e.g., XP, DSDM, FDD, and Scrum) are drastically gaining recognition in the software development community due to their quick response to rapid changes in user requirements, often volatile business environments, and market expectation (i.e., innovative and high quality software [4]). As defined CFSs should be oriented towards completing software development efficiently and effectively, we therefore perform literature review to investigate the identified influential factors affecting the successful agile software development and the successful knowledge transfer.

For the first focus, 20 highly visible studies on successful agile software development published between 2001 and 2011 have been chosen. The reviewed literature is mostly based on survey studies, experiences, case studies of agile software projects. In particular, Calo et al. [54] present an approach assisting in the way agile methods satisfy Agile Manifesto postulates. Ceschi et al. [33] have investigated whether agile methods improve project management practices. Chow and Cao [34], França et al. [55], Livermore [56], Misra et al. [57], Othman et al. [58], and Tong et al. [59] have explored influential factors of agile software projects. Cockburn and Highsmith [60] and Turner and Boehm [61] present the people factor in agile software development; whilst Iivari and Iivari [62] and Strode et al. [63] present the impact of organizational culture on agile software development. Others who have discovered several approaches or suggestions for successfully introducing or migrating agile software processes to organizations, include Cohn and Ford [64] and Nerur et al. [65]; whilst McMahon [66] suggests the means to bridge agile and traditional development methods. The others present results from empirical studies (e.g., Dybå and Dingsøyr [67], Hoda et al. [68], Korkala et al. [69], Lindvall et al. [70] and Schatz and Abdelshafi [71]). We assume that the above-mentioned studies are a representative, not an exhaustive list. Based on these studies, the data were collected in multiple settings (e.g., developed nations and developing nations). The results shows that the identified factors impacting on software projects in those areas are similar, as summarized in Table 2-2.

Table 2-2. A summary of the identified influential factors of agile software projects

Source	Research Setting	Factor
Calo et al. [54]	Not specified	Correct delivery strategy, proper practice of agile software engineering techniques, team capacity, style of team work, good management of the agile development process, and active participation of the users in the project
Ceschi et al. [33]	Developed nation(s)	Individual competence, teamwork, motivation
Chow and Cao [34]	Developed and developing nation(s)	Team environment (including team size), team capability, user involvement, project management process, agile software development techniques, delivery strategy
Cockburn and Highsmith [60]	Developed and developing nation(s)	Individual competence, team competence, organizational culture, management support, communication, project type, team size, software process, appropriate methodologies
Cohn and Ford [64]	Not specified	Individual competence, leadership, frequent communication, organizational culture, user commitment
Dybå and Dingsøyr [67]	Not specified	Continuous feedback, organizational culture, collaborative work, team characteristics, a high degree of knowledge creation, team member competence, team size, user and team member satisfaction, and appropriate techniques, tools and methods
França et al. [55]	Developing nation(s)	Project management process, agile software development techniques, delivery strategy, team capability, team environment, customer involvement
Hoda et al. [68]	Developed and developing nation(s)	Customer involvement
Iivari and Iivari [62]	Not specified	Organizational culture, cultural compatibility
Korkala et al. [69]	Developed and developing nation(s)	Frequency of communication, the content of that communication and engagement with the customer along with a support for rapid decision making amongst the development teams and the customer groups, and customer involvement
Lindvall et al. [70]	Not specified	Project size and characteristics (e.g., criticality, reliability, and safety), corporate culture, team member competency, project management, agile software development techniques, team size, training and learning, user involvement, communication, physically co-located teams
Livermore [56]	Not specified	Training, management involvement and support, access to external resources (e.g., books, journals, consultants, and attendance at methodology user groups), and company size
McMahon [66]	Not specified	frequent feedback, communication and collaboration, user involvement, and project management (i.e., planning and control)
Misra et al. [57]	Many continents around the world	Customer satisfaction, customer collaboration, customer commitment, decision time, corporate culture, planning and control, personal characteristics, societal culture, and training and learning
Nerur et al. [65]	Not specified	Individual competence, teamwork, organizational culture, management style, management of software development knowledge, reward systems, user relationships, management and software development processes, appropriate methods
Othman et al. [58]	Developed and developing	Management commitment, organizational environment, team environment, team capability, user involvement, project

Source	Research Setting	Factor
	nation(s)	management process, agile software development process, appropriate techniques, training support, project type, project nature, team size
Schatz and Abdelshafi [71]	Developed nation(s)	Teamwork, organizational culture, management support, negotiation skills
Strode et al. [63]	Developed nation(s)	Leadership-and-collaborative management style, feedback and learning environment, teamwork, empowerment of people, results-oriented organization, leadership, loyalty, mutual trust, and commitment
Tong et al. [59]	Not specified	Enterprise stratagem, mature information techniques, the resource of capital and time, and the ability of study, and communication
Turner and Boehm [61]	Not specified	Organizational culture, people competency, customer involvement, communication, training, user and team member satisfaction (i.e., values), and expectations management

For the second focus, we have chosen 24 papers which are highly visible, relevant to knowledge transfer in IT/IS related areas, and published between 2002 and 2011. We assume that they are a representative, not an exhaustive list. The reviewed literature is all based on empirical studies. In particular, those studies empirically examined the factors influencing knowledge transfer and/or knowledge acquisition in software process improvement [41], software development [72-78], IT/IS outsourcing [79-84], the software industry [85], IT consulting and/or client firms [86, 87], IT/IS usage [88-90], Enterprise Resource Planning (ERP) implementation [91, 92], IT-related small and medium enterprises [93], project management [94]. Based on the above literature, data was collected in multiple settings (e.g., developed nations and developing nations). The results shows that the identified factors affecting knowledge transfer in those areas are similar, as summarized in Table 2-3.

Table 2-3. A summary of the identified influential factors of knowledge transfer

Source	Research Setting	Factor
Al-Salti [79]	Not specified	Capability, credibility, nature of knowledge (i.e., complexity and tacitness), transfer mechanism, absorptive capacity, organizational culture, motivation, cultural distance, communication quality, and use of collaborative techniques
Arshad et al. [80]	Developing nation(s)	Distribution capacity, perceived benefit, quality content and accuracy of knowledge, and knowledge infrastructures (i.e., sharing culture, ICT infrastructure, staff posting, trust, and job satisfaction)
Cantú [93]	Developed nation(s)	Source's reliability, source's resistance, recipient's absorptive capability, recipient's receptiveness, knowledge causal ambiguity, knowledge complexity, ease of teaching, organizational culture, physical distance, and time available
Chen et al. [90]	Developed nation(s)	Organizational capital (i.e., information system and organizational structure), human capital, and relational capital (i.e., credibility, interpersonal relationship, and commitment)
Dayasindhu [85]	Developing nation(s)	Recipient's experience, relationship between knowledge source and recipient, uncertainty over future knowledge needs, not congruent in product and knowledge domains, and culture (e.g., power distance, individualism vs. collectivism, uncertainty avoidance, and

Source	Research Setting	Factor
		masculinity vs. femininity)
Gregory et al. [81]	Developed and developing nations	Motivation, clear roles and responsibilities, cultural competence (i.e., cross-cultural learning and adaptation), continuous working relationships
Hongli and Lei [87]	Not specified	Knowledge sharing culture, source's knowledge possession, source's and recipient's capacity, and willingness to transfer
Hsu and Lin [89]	Developed nation(s)	Motivation factors (i.e., altruism and reputation)
Joshi et al. [72]	Developed nation(s)	Great motivation, source's capability, source's credibility, knowledge type, good relationship, and extensive communication
Ko et al. [86]	Not specified	Source's credibility, absorptive capacity, shared understanding, motivation, and communication encoding and decoding competence
Kotlarsky and Oshri [75]	Developed and developing nations	Relationship and trust
Malhotra and Galletta [88]	Not specified	Motivation, commitment
Mohamed et al. [83]	Not specified	Source's sharing motivation, recipient's absorptive capacity, knowledge's quality, communication flow, training, and ICT infrastructure
Park et al. [82]	Developed nation(s)	Capability, trust, source's and recipient's character (e.g., skills, competencies, and integrity including commitment), cooperative learning, project complexity, and organizational support
Sarker [73]	Developed and developing nations	Source's capability, source's credibility, communication, and culture
Sarker et al. [78]	Developed nation(s)	Communication, capability, credibility, and culture
Slaughter and Kirsch [41]	Not specified	Nature of relationship, proximity, and work units between a source and a recipient
Tiexin et al. [94]	Developing nation(s)	Source's transferability, source's transfer willingness, communication attitude (e.g. good relationships), friendly exchanges, recipient's absorptive ability, recipient's learning motivation, and knowledge characteristics (i.e., systemic)
Upadhyaya and Krishna [74]	Developed and developing nations	Absorptive capacity, good relationship, task inter-dependence, task complexity, and communication frequency
Wang et al. [91]	Developed nation(s)	Absorptive capacity, and consultant competence (i.e., capability)
Xu and Ma [92]	Developing nation(s)	Transfer willingness, acquirement willingness, absorptive capacity, project priority, and transfer activity
Yuan et al. [76]	Developing nation(s)	Project commitment and mutual trust
Yun [84]	Developing nation(s)	Project character (e.g., novelty, customization, complex), firm size, interaction participation, process maturity, communication quality, knowledge overlap, prior cooperation experiences, culture fit, work dispersion, and absorptive capability
Zhang et al. [77]	Developing nation(s)	Organizational and technology factors (i.e., trust, leadership, issues and incentives, number and variety of groups, technology, implementation strategy, and interactions) and knowledge factors (i.e., explicitness, context-embeddedness, and practice-embeddedness)

It is important to discuss potential limitations. Exhaustive reviews to investigate the influential factors affecting the success in agile software development and knowledge transfer were not performed in this chapter. This limits the generalizability of the results, as we have not captured all the relevant papers in the boundaries. However, we reduce this threat to some extent by reviewing papers principally based on empirical studies in multiple settings (e.g., developed nations, developing nations, and specific worldwide environments) and gaining the current influential factors in the boundaries. Our focus primarily is on the papers providing the stories of either successful agile software development or successful knowledge transfer in practice, published in the last decade ranging between 2001 and 2011. The field of Software Engineering has changed dramatically over the decades. Hence, the influential factors have to be updated to fit the contemporary era as well. This also helps improve the quality of the results. In consequence, we believe that the review results can be used as the representative results.

There is an amount of IS research that focuses on the improvement of software development success in aspects of speed, effectiveness, efficacy, and low cost, to only name the most important ones. It is commonly accepted that no single method can serve for all types of software projects and all types of project objectives. To develop a framework for transferring software project management approaches into the Thai telecommunications industry, we consequently consider many related theoretical models as foundations of this study as described in the next section.

2.4 Foundations of this Study - Where We Can Start From

In this study, it is important to clearly define that we have decided to limit the analysis to the Thai telecommunications industry for the following five main reasons. First, the telecommunications industry was chosen as the research domain since it is a significant and highly developed area of the Thai economy. Moreover, implementing and deploying its elements (e.g., advanced mobile networks) is likely to stimulate innovation in the development of the mobile content and software industry [95]. Hence, focusing on the telecommunications industry may also benefit the software industry. Second, as everywhere else, the telecommunications industry is characterized by free competition. Companies in this domain do consequently depend upon quickly rolling out higher quality of services and products and innovation through efficient and effective software development and knowledge transfer mechanisms. Third, the perspectives and results of this research are presumably easier to transfer into an already developed industry. Fourth, the setting of this study was determined by ÖAD (the Austrian Agency for International Cooperation in Education and Research) and the Higher Education Commission of Thailand who support this study in the form of a scholarship. Hence, the economically most beneficial contribution of this study is knowledge that can be transferred into the Thai telecommunications companies. Lastly, as the sample of the participating companies was limited to the Thai telecommunications industry, it would be too risky to draw more general conclusions. This is because they cannot be substantiated by data from our case studies. Therefore, we at this stage limit our proposed frameworks (i.e., a software process maintenance framework and a knowledge transfer framework) and conclusions to software development in the Thai telecommunication industry. Nevertheless, we hope to further investigate, modify, and test our framework in other industries in order to prove its general applicability.

Concerning efficient and effective software development, currently both traditional project management and agile software development methods are gaining great popularity in

the software development sector [96]. PMBOK and Scrum are definitely the most popular for project management and agile software projects, respectively [36, 37]. Nonetheless, PMBOK has been influenced by agile tendency, as we can see that the latest PMBOK edition promotes its practices in an agile way [38]. Scrum is management-oriented and has more advantages in facets of responsiveness to environment, team flexibility and creativity, knowledge transfer during software development, and high probability of success [97]. However, it offers limited support for project management (i.e., limited support for scope, time, cost, risk, quality, procurement and documentation management [9, 11]). An efficient software process needs to be able to cope with project and process management activities. To provide adequate support for these two aspects, an integrated PMBOK-Scrum approach is thus taken into account. Supporting this, Fitsilis [98] suggests that connecting Scrum with PMBOK can benefit the teams since software processes in Scrum and PMBOK, are addressed in a compatible way. Moreover, this study intends to maintain the software process being continuously efficient by assessment and improvement. Within this area, CMMI is widely adopted appraisal approach that helps improve software processes, produce quality and project reliability, and eliminate problems and defect causes [99]. Currently, many organizations are increasingly interested in adopting CMMI with agile methods together [24]. There are evidences that CMMI and agile, especially Scrum, can considerably coexist [22, 24, 25, 100]. CMMI is thus deemed for this study. During software development, knowledge transfer is crucial. This is because a software project typically consists of multiple stakeholders with diverse backgrounds and skill sets. Talents in teams should continuously complement each other for better work efficiency and effectiveness [40]. Therefore, it can be concluded that the quality of software depends upon the quality of software processes and knowledge transfer.

In this study, a framework for transferring software development project management approaches into the Thai telecommunications industry is proposed. It consists of two core components. First, a software process maintenance framework aims at providing the “what” and the “how” to improve and implement software processes. The framework consists of two main parts. For the first part providing the “what” software processes to improve, CMMI is used as a base. For the second part providing the “how” to plan and implement software processes, the authors merge the principles derived from the core of the PMBOK into the Scrum model. Second, a knowledge transfer framework aims at providing guidance for planning knowledge transfer activities, based on Szulanski’s model. The descriptions of the above models and principles are presented as follows.

2.4.1 Capability Maturity Model Integration (CMMI)

CMMI is a widely known appraisal approach for continuous SPI [39]. It is a process improvement capability maturity model for the processes controlling development, implementation, acquisition and maintenance of software products and services. It strives to achieve process consistency, predictability and reliability. CMMI consists of best practices that address development and maintenance activities that cover the software life cycle from conception through delivery and maintenance. CMMI itself has two representations: staged and continuous.

The staged representation is most suitable for an organization that does not know which processes need to be improved first since the staged representation provides a systematic structured way to improve and offers process areas applicable to each maturity level [101]. The staged representation focuses on process areas organized by five maturity levels, numbered 1 through 5 and dubbed initial, managed, defined, quantitatively managed,

and optimizing, respectively. Each maturity level comprises a predefined set of process areas, indicating which areas need to be implemented in order to reach a certain maturity level.

The continuous representation provides flexibility for selecting processes and maturity levels fit for achieving business goal of the organization [101]. In the continuous representation, each process area is rated in terms of capability level. There are six capacity levels, numbered 0 through 5 and dubbed incomplete, performed, managed, defined, quantitatively managed, and optimizing, respectively. Each capability level corresponds to a generic goal and a set of generic and specific practices. Moreover, the continuous representation has more specific practices than the staged representation since the continuous representation has two types of specific practices (i.e., base and advanced), whilst the staged representation has only one type of specific practice.

This study aims at providing a systematic structured way to improve rather than a flexible way for selecting processes and maturity levels to improve. Accordingly, the staged representation is used as the basis of this study.

2.4.2 Project Management Body of Knowledge Guide (PMBOK)

The PMBOK guide developed by the Project Management Institute (PMI) [38] is the standard that describes the project management processes, tools, and techniques used to manage a wide range of projects in many types of industries. Project management is the application of knowledge, skills, tools, and techniques to project activities in order to meet the project requirements. It is accomplished through the appropriate application and integration of the 42 processes which fall into five process groups and nine knowledge areas that are typical of almost all projects. The five basic process groups consist of (1) Initiating, which processes performed to define a new project or phase by obtaining authorization to start the project or phase; (2) Planning, which processes needed to establish the scope of the project, refine the objectives, and define what actions needed to attain the objectives; (3) Executing, which processes performed to complete the work defined in the project plan; (4) Monitoring and Controlling, which processes needed to track, review, control the progress and performance of the project; and (5) Closing, which processes performed to finalize all activities across all process groups to formally close the project or phase.

Those processes overlap and interact throughout a project or phase. Each process is described in terms of its inputs, outputs, and tools and techniques. Inputs and outputs are documents (e.g., a scope statement and user requirements) or documentable items (e.g., activity dependencies). Tools and techniques are mechanisms applied to inputs to create outputs. Those processes are also organized into the nine knowledge areas which are (1) Integration Management, which includes the processes and activities needed to identify, define, combine, unify, and coordinate the various processes and activities within the process groups; (2) Scope Management, which ensures that all the required work and only the required work is planned, defined, documented, and delivered to the user's satisfaction; (3) Time Management, which includes the processes needed to manage timely completion of the project; (4) Project Cost Management, which includes the processes involved cost estimation and expense monitoring, and intended to ensure that the project is delivered within its approved budget; (5) Quality Management, which encompasses quality definition, assurance, and control; (6) Human Resource Management, which includes the processes that organize, manage, and lead the project team; (7) Communication Management, which includes the processes for information dissemination and collection; (8) Risk Management, which

includes the processes for risk identification, quantification, avoidance, and mitigation; and (9) Procurement Management, which includes the processes necessary to purchase or acquire products or services needed from outside the project team. Figure 2-1 presents a graphical representation of all 42 project management processes, falling into process group and knowledge area dimensions.

Knowledge Areas	Project Management Process Groups				
	Initiating Process Group	Planning Process Group	Executing Process Group	Monitoring & Controlling Process Group	Closing Process Group
4. Project Integration Management	4.1 Develop Project Charter	4.2 Develop Project Management Plan	4.3 Direct and Manage Project Execution	4.4 Monitor and Control Project Work 4.5 Perform Integrated Change Control	4.6 Close Project or Phase
5. Project Scope Management		5.1 Collect Requirements 5.2 Define Scope 5.3 Create WBS		5.4 Verify Scope 5.5 Control Scope	
6. Project Time Management		6.1 Define Activities 6.2 Sequence Activities 6.3 Estimate Activity Resources 6.4 Estimate Activity Durations 6.5 Develop Schedule		6.6 Control Schedule	
7. Project Cost Management		7.1 Estimate Costs 7.2 Determine Budget		7.3 Control Costs	
8. Project Quality Management		8.1 Plan Quality	8.2 Perform Quality Assurance	8.3 Perform Quality Control	
9. Project Human Resource Management		9.1 Develop Human Resource Plan	9.2 Acquire Project Team 9.3 Develop Project Team 9.4 Manage Project Team		
10. Project Communications Management	10.1 Identify Stakeholders	10.2 Plan Communications	10.3 Distribute Information 10.4 Manage Stakeholder Expectations	10.5 Report Performance	
11. Project Risk Management		11.1 Plan Risk Management 11.2 Identify Risks 11.3 Perform Qualitative Risk Analysis 11.4 Perform Quantitative Risk Analysis 11.5 Plan Risk Responses		11.6 Monitor and Control Risks	
12. Project Procurement Management		12.1 Plan Procurements	12.2 Conduct Procurements	12.3 Administer Procurements	12.4 Close Procurements

Figure 2-1. A graphical representation of all 42 processes [38]

PMBOK is a general guide used by professional project managers to achieve long-term goals and is applied in many software development projects. It is also viewed as quasi standard by several leading software development companies. Therefore, PMBOK can be used as input for developing the proposed software process maintenance framework.

2.4.3 Scrum

Scrum was developed by Ken Schwaber and Jeff Sutherland. It is an iterative and incremental software development process commonly used in the context of agile software development [102]. Scrum focuses on project management in situations where it is difficult to plan ahead, with mechanisms for ‘‘empirical process control” and where feedback loops constitute the core element. Software is developed by a self-managing team in iterations (called ‘‘sprints”), starting with planning and ending with a review. Scrum has three primary roles, three primary artefacts, and four primary ceremonies designed to deliver work products in sprints.

The three primary roles consist of product owner, Scrum master, and team. Product Owner is a person responsible for creating and prioritizing the features of the product, deciding on release date and content, adjusting features and priority, and accepting or rejecting work results. Scrum Master is a facilitative team leader working closely with the product owner and responsible for ensuring that the team is fully functional and productive, removing impediments, shielding the team from external interference, and making certain that the process is followed. Team typically consists of seven plus or minus two members. The team is committed to achieving a sprint goal and has the right to do whatever it takes to achieve the goal. The team organizes itself and its work and demos results to the product owner.

The three primary artefacts consist of product backlog, sprint backlog, and Burndown chart, which are all openly accessible and visible to the team. Product Backlog is a list of all prioritized business and technical requirements that need to be developed and defects that need to be fixed. Each requirement contains a description such as category (e.g., feature, enhancement, and defect), status, priority, and estimated effort. Sprint Backlog is a list of all requirements in the current sprint that are broken down into tasks. Each task contains a short task description (e.g., owner, status, and effort). The sprint backlog is daily updated to obtain the latest effort of the work remaining to complete the task. Efforts can increase when the team member realizes that the work was underestimated. Burndown chart shows the hours remaining to complete sprint tasks. It is primarily displayed for the team.

The four primary ceremonies consist of sprint planning, daily Scrum meeting, sprint review, and sprint retrospective. Sprint Planning is held in 4-to-8-hour length at the beginning of each sprint. The product owner prioritizes over the product backlog and the team defines tasks that they can complete during the coming sprint. Once this set of features has been identified, no re-prioritization takes place during the ensuing sprint in which features are designed, implemented and tested. Daily Scrum Meeting is held daily in 15-minute length. Stakeholders may attend the meeting, but only the team and the Scrum master can speak. Each team member answers the questions of “What did you do yesterday? What will you do today? What impediments are in your way?” Sprint Review takes place at the end of the sprint for the team to review progress, demonstrate what they have built during the sprint to the stakeholders and the product owner, and get feedback. Sprint Retrospective is a place for

the team to discuss what is working and what is not working, and agree on changes to try for software process improvement.

Moreover, Scrum processes are grouped in three stages (i.e., pre-game, game, and post-game) [97]. Pre-game includes two processes (i.e., planning and architecture development). The planning includes the definition of a new release based on currently known product backlog, along with an estimate of its schedule and cost. If the software product under development is new, planning includes both conceptualization and analysis, but only limited analysis for an existing software product. The architecture development includes system and/or software architecture development and high level design. Game includes the process of sprint execution. This stage consists of a collection of development sprints to produce new release functionality, with constant respect to the variables of time, requirements, quality, cost, and competition. Post-game is the closure of the project, which includes preparing the releases, producing the final documentation, executing the site acceptance testing and the final product release. The overall Scrum process is illustrated in Figure 2-2.

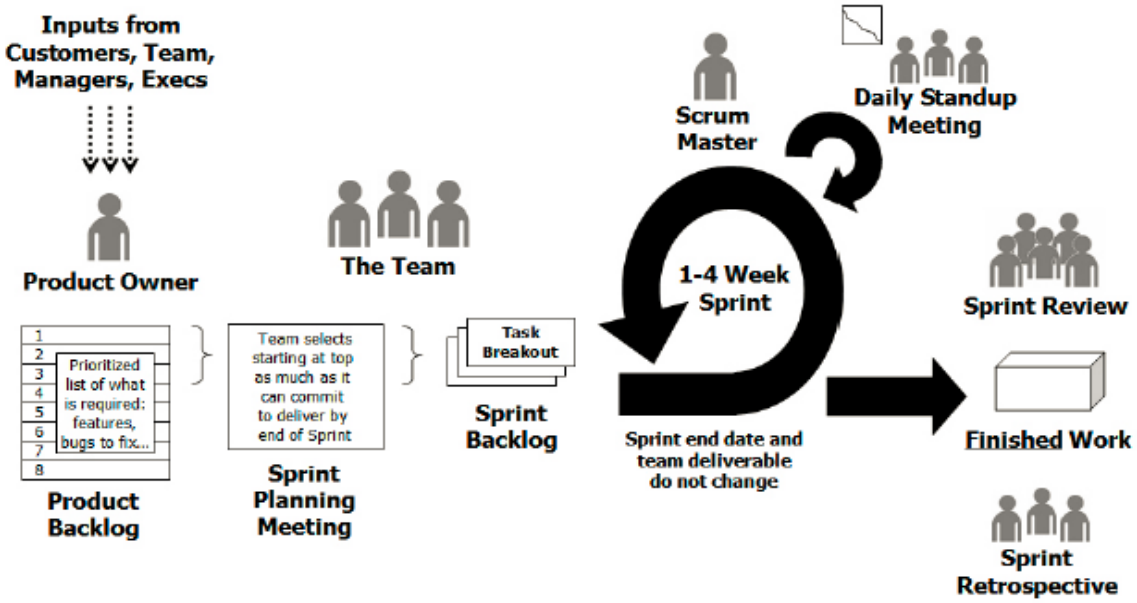


Figure 2-2. The Scrum process [102]

2.4.4 Szulanski's Knowledge Transfer Model

Szulanski's (1996) theory of a communication-based knowledge transfer model describes an intra-firm knowledge transfer process [103]. The process is viewed as a message transmission from a source to a recipient in a given context. The process evolves in the following four stages. First, Initiation comprises all events that lead to the decision to transfer. A transfer begins when both a need and the knowledge to meet that need coexist within the organization. Once the need and a solution to that need are identified, the feasibility of the transfer is explored. Second, Implementation begins with the decision to proceed in which resources flow between the recipient and the source. Transfer-specific social ties between the source and the recipient are established and the transferred practice is

often adapted to suit the anticipated needs of the recipient, to prevent problems experienced in a previous transfer of the same practice, or to make the introduction of new knowledge less threatening to the recipient. Implementation related activities diminish after the recipient begins using the transferred knowledge. Third, Ramp-up begins when the recipient starts to use the transferred knowledge; that is after the first day of use. During this stage, the recipient will be primarily concerned with identifying and resolving unexpected problems that impede its ability to match or exceed post-transfer performance expectations. The recipient is likely to use the new knowledge ineffectively at first, but gradually improves performance, ramping up toward a satisfactory level. Last, Integration begins after satisfactory result is achieved by the recipient from the transferred knowledge and the transferred knowledge is converted into the organization's routine. The four stages are presented in Figure 2-3.

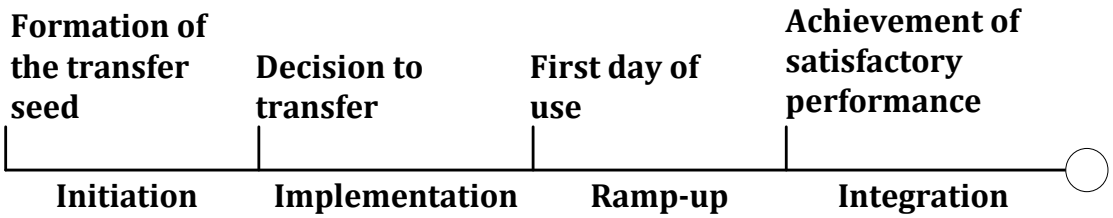


Figure 2-3. Knowledge transfer stages and milestones [104]

This model has also explored the origin of internal stickiness. Stickiness is a difficulty encountered within the knowledge transfer process. It can be predicted by examining a number of conditions relating to characteristics of the knowledge transferred (i.e., causal ambiguity and unprovenness), characteristics of the knowledge source (i.e., a lack of motivation and not perceived as reliable), characteristics of the knowledge recipient (i.e., a lack of motivation, a lack of absorptive capacity, and a lack of retentive capacity), and characteristics of the context in which the transfer takes place (i.e., barren organizational context and arduous relationship). Appropriate frameworks of software process maintenance and knowledge transfer are vital to achieving the improvement of software development performance in terms of efficiency and effectiveness. To build such frameworks, the requirements which are specific to the EIS development in the Thai telecommunications industry thus need to be identified.

2.5 Influential Factors in the Areas of Software Development and Knowledge Transfer as Requirements for a Framework for Transferring Software Project Management Approaches into the Thai Telecommunications Industry

Stating requirements is very important for the design of all mechanisms. Requirements for the proposed software process maintenance framework are summarized into Table 2-4, which are also compared to the influential factors identified in the reviewed literature presented in Table 2-2. This is based on the consolidation of a number of failure/success factors listed in Tables 2-1 and 2-2 which share similar characteristics. The results in Table 2-4 help answers the RQ2-3 "What are the factors affecting the successful agile software development?".

Table 2-4. Requirements for the proposed software process maintenance framework

Study	Organization		People		Process		Technology		Project			Data
	Organizational environment	Management commitment	Team capability	User involvement	Project management process	Agile software development process	Appropriate methods, techniques, and tools	Training support	Project type	Team size	Team environment	Data quality (i.e., accurate, complete, and up to date)
Calo et al. [54]			X	X	X	X	X				X	
Ceschi et al. [33]			X								X	
Chow and Cao [34]			X	X	X		X			X	X	
Cockburn and Highsmith [60]	X	X	X			X	X		X	X	X	
Cohn and Ford [64]	X	X	X								X	
Dybå and Dingsøyr [67]	X		X				X			X	X	
França et al. [55]			X	X	X		X				X	
Hoda et al. [68]				X								
Iivari and Iivari [62]	X											
Korkala et al. [69]				X							X	
Lindvall et al. [70]	X		X	X	X		X	X	X	X	X	
Livermore [56]		X						X				
McMahon [66]				X	X						X	
Misra et al. [57]	X	X	X	X	X			X			X	
Nerur et al. [65]	X		X	X	X	X	X				X	
Othman et al. [58]	X	X	X	X	X	X	X	X	X	X	X	
Schatz and Abdelshafi [71]	X	X	X								X	
Strode et al. [63]	X	X	X								X	
Tong et al. [59]	X	X					X					
Turner and Boehm [61]	X		X	X	X			X			X	
Total	**12**	**8**	**14**	**11**	**9**	**4**	**9**	**5**	**3**	**5**	**16**	**0**

Furthermore, Table 2-5 presents the summary of the influential factors as requirements for the proposed knowledge transfer framework, which are compared to the influential factors identified in the reviewed literature presented in Table 2-3. This is based on the consolidation of the problems of the knowledge transfer process in the current situation in EIS development and the influential factors identified in the reviewed literature. This help answers the RQ2-4 "What are the factors affecting the successful knowledge transfer in software development?". In this table, we categories the influential factors into five contexts [105] which are source, recipient, knowledge, relational, and situational contexts. The source and recipient contexts refer to the attributes of the source and the

recipient which can facilitate or impede the knowledge transfer process. The knowledge context refers to the nature and characterization of the type of knowledge being transferred. The relational context refers to the attributes that characterize the relationship between the source and the recipient. The situational context refers to the environmental characteristics surrounding the knowledge transfer process.

Concerning the meanings of each influential factor, source's motivation refers to motivation to transfer knowledge. Source's capability refers to the source's greater reservoir of knowledge that has a potential to transfer knowledge. Source's credibility refers to the degree in which the source is perceived as trustworthy and reputable by the recipient. Recipient's motivation refers to motivation to absorb knowledge. Recipient's absorptive capacity refers to the ability of the recipient to recognize the value of new knowledge, assimilate it, and apply it. Knowledge's usefulness refers to the degree to which the source and the recipient believe that using knowledge would enhance their job performance. The greater the knowledge is valuable; the greater would be its attractiveness for the recipient and the knowledge application by the recipient. Knowledge's ease of use refers to the degree to which the source and the recipient believe that using knowledge would be free of effort. The easier the recipient can use the knowledge, the greater the recipient's effort to obtain the knowledge. Good relationship refers to the intimacy of a relationship between the source and the recipient. Commitment refers to the source's and the recipient's commitment in terms of time, effort, and attention. Extensive communication refers to frequent communication between the source and the recipient; while organizational culture refers to the values, practices, and assumptions that influence the organization's members to act and behave in a particular manner [79].

Table 2-5. Requirements for the proposed knowledge transfer framework

Study	Source Context			Recipient Context		Knowledge Context		Relational Context		Situational Context	
	Motivation	Capability	Credibility	Motivation	Absorptive capacity	Usefulness	Ease of use	Good relationship	Commitment	Extensive communication	Organizational Culture
Al-Salti [79]	X	X	X	X	X		X			X	X
Arshad et al. [80]			X			X					X
Cantú [93]			X		X		X				X
Chen et al. [90]			X					X	X		
Dayasindhu [85]					X			X			X
Gregory et al. [81]	X			X				X			X
Hongli and Lei [87]	X	X			X						X
Hsu and Lin [89]	X			X							
Joshi et al. [72]	X	X	X	X				X		X	
Ko et al. [86]	X		X	X	X						
Kotlarsky and Oshri [75]			X					X			

Study	Source Context			Recipient Context		Knowledge Context		Relational Context		Situational Context	
	Motivation	Capability	Credibility	Motivation	Absorptive capacity	Usefulness	Ease of use	Good relationship	Commitment	Extensive communication	Organizational Culture
Malhotra and Galletta [88]	X			X					X		
Mohamed et al. [83]	X				X	X				X	
Park et al. [82]		X	X		X				X		X
Sarker [73]		X	X							X	X
Sarker et al. [78]		X	X							X	X
Slaughter and Kirsch [41]								X			
Tiexin et al. [94]	X			X	X			X			
Upadhyaya and Krishna [74]					X			X		X	
Wang et al. [91]		X			X						
Xu and Ma [92]	X			X	X						
Yuan et al. [76]			X						X		
Yun [84]					X					X	X
Zhang et al. [77]			X				X			X	
Total	**10**	**7**	**12**	**8**	**12**	**2**	**3**	**8**	**4**	**8**	**10**

According to Tables 2-4 and 2-5, there are indications that some of the major findings of this research might actually be of a more general nature and hence of a wider applicability. Hence, our frameworks proposed in the next section are not limited to EIS development.

2.6 Towards a Conceptual Framework for Transferring Software Project Management Approaches into the Thai Telecommunications Industry

In general, according to Wallin and Crnkovic [106], each software development project is run through a platform deployment lifecycle of four stages (i.e., ideas, feasibility study, software development, and rollout). First, the ideas stage starts with a collection of ideas for end user solutions that can be enable through the new software platform. Second, the feasibility study stage is to compile the information needed for the responsible management to make a decision whether to start a pilot development project. Third, the software development stage is where the approved pilot development project is run based on the feasibility study results. Last, the rollout stage runs when developed software is ready to be employed. For this platform deployment lifecycle, most development theories have similar methods for the stages of ideas, feasibility study, and rollout. Except for the utilization of software development methodologies, it depends on the type, nature, and characteristics of

each project. To be clear, this study mainly focuses on the software development stage as presented in Figure 2-4.

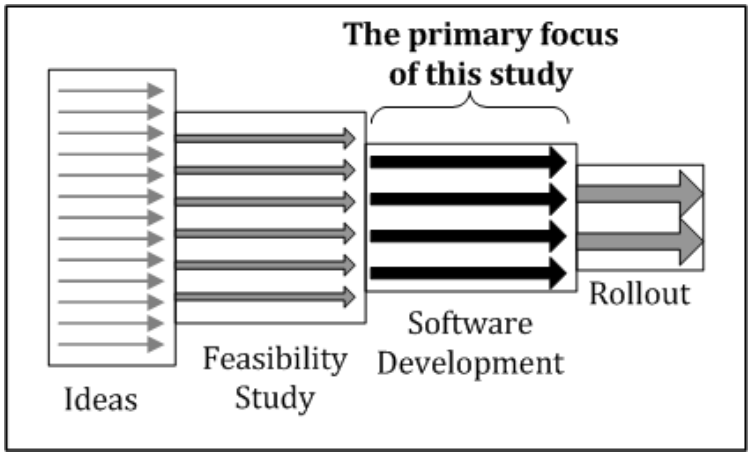


Figure 2-4. The primary focus of this study

To answer the RQ2-5 "What could our conceptual software process maintenance framework based on its requirements look like?", the CMMI, PMBOK, and Scrum all have strong benefits for the proposed software process maintenance framework. In the framework, there are two main parts. First, a CMMI-based software process improvement component aims at providing a systematic structured way to guide practitioners "what" software processes needs to be improved. Non-identified influential factors or other influential factors unidentified in the Section 2.5 might be discovered in the future. To be clear, at this stage the identified influential factors of this work are used as the identified CSFs. Success in a software process can be viewed in terms of Key Process Areas (KPAs) and CSFs. A number of studies argue that KPA approaches should improve the organization's capabilities to manage, develop, and deliver quality software products [107-111]. On the contrary, a multitude of studies concur that a successful software process should be viewed in terms of CSFs rather than KPAs. These studies emphasize the importance of the CSF approach in SPI and the use of the CSF approach rather than the KPA approach [20, 112-115]. They have also confirmed the value of the CSF approach in the field of information technology [20, 113-119]. Thus, the staged representation of CMMI and CSF approaches are employed. Second, an integrated PMBOK-Scrum process development component aims at providing a mechanism for establishing efficient and effective software processes. Thus, the PMBOK and Scrum processes are mapped together. Figure 2-5 shows our conceptual software process maintenance framework.

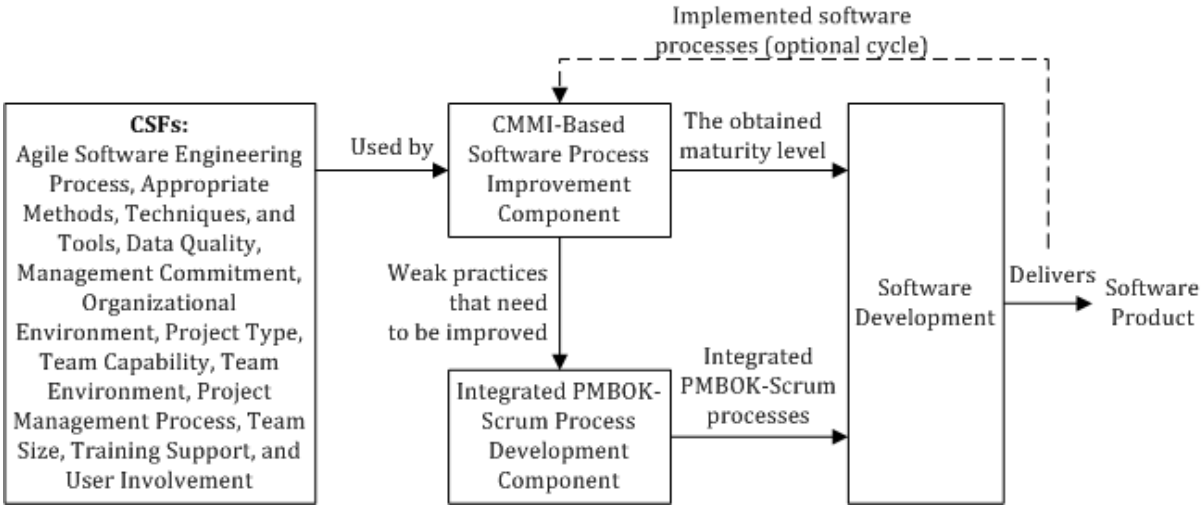


Figure 2-5. The proposed conceptual software process maintenance framework

To answer the RQ2-6 "What could our conceptual knowledge transfer based on its requirements look like?", our solution is based on Szulanski's model. Knowledge transfer can be viewed as a communication process between the source and the recipient engaged in teams. The process flows through four distinct stages which are Initiation, beginning with all events that lead to the decision to transfer; Implementation, beginning with the decision to transfer; Ramp-up, beginning when the recipient starts using the transferred knowledge; and Integration, beginning after the recipient achieves satisfactory outcomes. As the identified CSFs affect effective knowledge transfer, Figure 2-6 presents our conceptual proposed knowledge transfer framework.

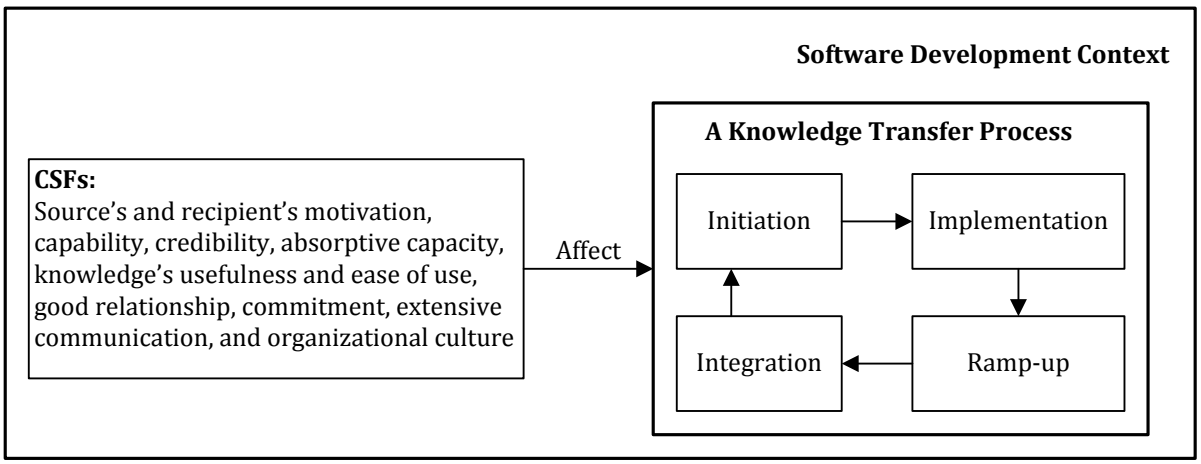


Figure 2-6. The proposed conceptual knowledge transfer framework

The requisite requirements identified for the proposed software process framework and the proposed knowledge transfer framework serve as basis for designing an abstract level models of the proposed conceptual frameworks. For the next steps, we separate work into three sections. The first section involves the sound development of the proposed software

process maintenance framework, which is presented in Chapters 3, 4, and 5. The second section involves the sound development of the proposed knowledge transfer framework, which is presented in Chapters 6 and 7. The last section involves the sound development of the proposed framework for transferring software project management approaches into the Thai telecommunications industry, which is presented in Chapter 8.

2.7 Summary

As this study focuses on the Thai telecommunications industry as a case study, we performed the interviews with in-house and outsourcing teams in the two ISPs in Thailand. This was in order to look into the current software development situation by focusing on the EIS development. The findings reveal that there are many software development problems (e.g., a lack of agile logistical arrangement, a lack of good user participation, a lack of management commitment, a lack of project management competence, a lack of teamwork, inappropriateness of methods and tools, a lack of training support, a lack of knowledge transfer, and so on). This emphasizes that software development teams do not perceive formal routines as an efficient and effective way not only to manage software processes but also to transfer knowledge. This also supports that the problems defined in prior research on EIS development in Thailand still exist today. These are challenging problems since a software process and a knowledge transfer process play a central role in successful software development.

Although agile software development methods (e.g., Scrum, XP, and FDD) offering effective software processes are available, they provide limited management support (e.g., for subcontracting, developing with large teams, developing software that demands high quality control, and distributed development environments). Since efficiency requires project management activities to enable the proper execution of software development tasks; therefore, project management provides the backbone for efficient software development. This shows that an approach offering an adequate set of project management and software development is required for software development efficiency and effectiveness. To sustain software development efficiency and effectiveness by maintaining the quality of software processes, SPI is thus needed. Moreover, knowledge transfer is crucial to a software project due to multiple stakeholders with diverse backgrounds and skill sets. Talents in teams should continuously complement each other for better work performance. From this perspective, it can be concluded that quality software depends upon quality software processes and knowledge transfer. As a result, a framework for transferring software development project management approaches into the Thai telecommunications industry has been proposed in this study. It consists of two core components which are frameworks themselves. First, a software process maintenance framework which in this context means a framework for software process development and improvement, provides the “what” to improve through a CMMI-based SPI component and the “how” to implement software processes through an integrated PMBOK-Scrum process development component. The framework is based on a principle set of the PMBOK and two models of CMMI and Scrum. CMMI is an SPI approach that provides organizations with the essential elements of effective software processes that help improve their performance. PMBOK provides general guidance covering all facets of project management in the traditional sense. Scrum is commonly used in the agile software development context. Second, a knowledge transfer framework provides guidance for planning knowledge transfer activities. As communication is at heart of knowledge transfer, Szulanski’s model serves as a basis for the framework. The resulting frameworks at this stage

aim at contributing to the improvement of software development performance in terms of efficiency and effectiveness in the Thai telecommunications industry.

The starting point to construct the frameworks is to identify two sets of the influential factors as requisite requirements, based on the interview findings and the literature review findings. The first set for efficient and effective software development consists of 12 influential factors. They are agile software development process, appropriate methods, techniques, and tools, data quality, management commitment, organizational environment, project management process, project type, team capability, team environment, team size, training support, and user involvement. A second set for successful knowledge transfer consists of 11 influential factors. They are a source's motivation, a source's capability, a source's credibility, a recipient's motivation, a recipient's absorptive capacity, usefulness of knowledge and its ease of use, good relationship, commitment, extensive communication, and organizational culture. Nevertheless, there are indications that some of the major findings of this research might actually be of a more general nature and hence of a wider applicability. Therefore, our proposed frameworks are not limited to EIS development. Non-identified influential factors unidentified in this chapter that affect the successful software development and knowledge transfer might be discovered in the future.

For the next steps, we separate work into three sections. The first section involves the sound development of the proposed software process maintenance framework, which is presented in Chapters 3, 4, and 5. The second section involves the sound development of the proposed knowledge transfer framework, which is presented in Chapters 6 and 7. The last section involves the sound development of the proposed framework for transferring software project management approaches into the Thai telecommunications industry, which is presented in Chapter 8.

Chapter 3

Gap Analysis in the Field of Agile Software Development Integration with Software Process Improvement and with Traditional Project Management

A large part of the business world applies agile methods for effectively responding to often unexpected and unpredictable changes in customer requirements and delivering quality software. The quality of software depends on the quality of software development processes. Hence, this study argues for a software process maintenance framework aiming at contributing to the improvement of software development performance in terms of efficiency and effectiveness. As a starting point for planned further research, this chapter presents a gap analysis regarding the prerequisites for the sound development of the proposed software process maintenance framework. This gap analysis is performed through a systematic literature review in the field of agile software development integration with software process improvement and with traditional project management. Based on the findings, solutions bridging gaps in this field are then presented.

3.1 Introduction

An efficient and effective software development process (hereafter referred to as "software process") significantly influences successful software development. Efficiency requires project management activities to enable the proper execution of software development tasks. Project management thus provides the backbone for efficient software development [7]. Currently, agile methods offering effective software development (e.g., Adaptive Software Development (ASD), Dynamic System Development Method (DSDM), eXtreme Programming (XP), Test Driven Development (TDD), Feature Driven Development (FDD), Lean Software Development, Rational Unified Process (RUP), and Scrum) are widely used. Unfortunately, they provide limited project management support, e.g., for subcontracting, developing with large teams, developing software that demands high quality control, and distributed development environments [4, 9-11]. From this view, integration of agile with project management processes could overcome this limitation and result in software development performance in terms of efficiency and effectiveness. However, software process development cannot guarantee sustainable software development. It also needs continuous Software Process Improvement (SPI). Agile methods typically have iterative SPI during a software project; whilst traditional SPI approaches, e.g., Capability Maturity Model (CMM), Capability Maturity Model Integration (CMMI), and International Organization for Standardization and the International Electrotechnical Commission 15504 Standard (ISO/IEC 15504) (also known as SPICE: Software Process Improvement and Capability Determination), typically use retrospective reports of the previous software projects for future software projects. It would be better to integrate these two concepts

together for being more efficient. Consequently, there is a need to build a software process maintenance framework which in this context means a framework for software process development and improvement.

With respect to the improvement of a software process, different well-known maturity models, e.g., CMMI, ISO/IEC 15504, Six Sigma, and Control Objectives for Information and Related Technology (COBIT), are supporting SPI. However, there are different aspects of these models. CMMI is a well-accepted model to optimize the development activity in every stage for improving software process performance and software quality [120]. It is proposed to help organizations to establish a mature software development process with high predictability and low risk [99]. ISO/IEC 15504 provides a structured approach and represents a continuous conception [121], similarly to CMMI. Nonetheless, its goal is to improve the capability in each process in many areas such as a custom service process, not specific to only software development. Six Sigma is a manufacturing-oriented model for measuring and improving a company's operational performance through rigorous use of data and statistical analysis by identifying and eliminating defects in manufacturing and service-related processes [122]. It typically addresses quality and customer satisfaction issues by focusing on process problems and production of a measurable return on investment [123, 124]. COBIT is business-oriented. The keys to successful implementations are concentrating on business drivers and results the organization is seeking [125]. A COBIT maturity model is at a strategic level and focuses on high-level Information Technology (IT) management processes. Its main purpose is to give management a tool to help them better understand the current capability of IT management processes, and do benchmarking, gap analysis and improvement planning. This research aims to improve software process performance and software quality by optimizing the development activities. Compared to the SPI models, CMMI is the most appropriate for constructing an SDM model as an SPI component of our software process maintenance framework.

With respect to the development of an efficient and effective software process, agile methods are most currently adopted and have generated lots of interest in the software development sector [96] due to their high probability of success and effectiveness. A survey by Ambysoft in 2008 showed that 70% of agile software projects were successful compared to 66% of software projects based on the waterfall method. The results of effectiveness of agile methods compared with traditional methods also showed that 82% of productivity, 77% of quality, and 78% of business stakeholder satisfaction were higher; and 72% of the system development costs were lower [126]. However, these methods offer limited project management support [6, 8]. Albeit integration of agile software development and traditional project management methods has been suggested to overcome these limitations (e.g., [8, 96, 98]), little attention has been paid to it. Hence, there is a great need to develop an integrated project management and software development approach. In the domain of agile methods, Scrum is definitely the most popular [36]. It has emerged as the most successful agile development process for organizations and developers [127]. An agile development survey in 2008 by VersionOne provides one key trend that almost 50% of the responses indicated they were using Scrum [128]. In 2008, Digital Onion stated that its engagements succeeded 80% of the time, an 8.5% margin over average agile software projects, and a 17.2% margin over waterfall software projects was also identified [129]. Considering traditional project management, PMBOK is the broadest and most widely used standard reference of industry best practices for project management [37]. It identifies generally accepted and fundamental practices and guidelines that are applicable to a wide range of markets. Moreover, the use of PMBOK is still increasing. Project Management Institute (PMI) membership statistics show an overall increase of 14.3% for the year 2009 [130]. Supporting this, a process framework

survey in 2008 by Ambysoft showed that compared with other project management frameworks, PMBOK is for example recognized for being used more than Projects in Controlled Environments (PRINCE2). There was 22% of primary audiences for a framework who had not heard about PMBOK, but 69.5% for PRINCE2 [126]. Moreover, the results of Fitsilis' study [98] reveal the appropriateness of PMBOK and agile integration that amongst XP, Scrum, and FDD - Scrum is the most compatible with PMBOK. Hence, two outstanding methods (i.e., PMBOK and Scrum) are used for constructing an integrated PMBOK-Scrum model as an integrated software process development component of our software process maintenance framework.

For the sound development of our software process maintenance framework, a gap analysis in the field of agile software development integration with software process improvement and with traditional project management has been presented. This is in order to answer the following research questions.

RQ3-1: Which existing research results on agile software development integration with software process improvement and with traditional project management are available that we can build on?

RQ3-2: What are some interesting aspects that existing research results on agile software development integration with software process improvement and with traditional project management do not yet cover?

RQ3-3: How should a software process maintenance framework be constructed? Is a software process maintenance framework workable? What does the test of a software process maintenance framework in a real-life situation contribute?

This chapter is organized as follows. The following sections describe research results on agile software development integration with software process improvement and with traditional project management that we can build on, and some interesting aspects that those research results do not yet cover. The description of how to construct a workable software process maintenance framework is then presented.

3.2 Review Approach

In this work, a systematic review has been performed according to the guidance proposed by Kitchenham and Charters [131]. Our goal is to provide theoretical and empirical support for a proposal of a software process maintenance framework, regarding agile software development integration with software process improvement and with traditional project management. The research questions guiding this review are as follows.

RQ3-1: Which existing research results on agile software development integration with software process improvement and with traditional project management are available that we can build on?

RQ3-2: What are some interesting aspects that existing research results on agile software development integration with software process improvement and with traditional project management do not yet cover?

RQ3-3: How should a software process maintenance framework be constructed? Is a software process maintenance framework workable? What does the test of a software process maintenance framework in a real-life situation contribute?

This set of research questions guided the selection of the search keywords for this review. The search keywords were categorized into three categories (i.e., Agile, SPI, and Traditional Project Management) covering the areas of agile software development, SPI, and traditional project management, respectively. The combination of Agile and SPI categories was used to search related literature on agile software development integration with software process improvement; whilst the combination of Agile and Traditional Project Management categories was used to search related literature on agile software development integration with traditional project management. In each category, both generic and specific terms were used to help improving the search results. For instance, generic terms in the Agile category (i.e., agile, agile method, agile development, agile project, agile process, and agile practice) were used to retrieve literature mentioning agile software development; whereas a specific term (i.e., Scrum), which is the foundation of this study, was used to retrieve literature particularly mentioning Scrum software development. The search keywords used in this review are presented in Table 3-1.

Table 3-1. Keywords used in the review process

Category	Keyword
Agile	agile, agile method, agile development, agile project, agile process, agile practice, Scrum
SPI	software project improvement, CMMI
Traditional Project Management	traditional project management, plan-based, plan-driven, disciplined, PMBOK

3.2.1 Data Sources and Search Strategy

The search strategy included five electronic databases as follows.

- ACM Digital Library Database
- IEEE Xplore Database
- Elsevier ScienceDirect Database
- Springer Link Database
- Wiley InterScience Database

This review was conducted in four stages. This review focuses on two main parts: agile software development integration with software process improvement and with traditional project management. In stage 1, the title, abstracts, and keywords of the papers published between 2001 and 2011 in the included data sources were searched using two combinations of categories: Agile and SPI, and Agile and Traditional Project Management. Therefore, we have two search strings as follows.

- (agile OR “agile method” OR “agile development” OR “agile project” OR “agile process” OR “agile practice” OR Scrum) AND (“software process improvement” OR CMMI)
- (agile OR “agile method” OR “agile development” OR “agile project” OR “agile process” OR “agile practice” OR Scrum) AND (“traditional project management” OR “plan-based” OR “plan-driven” OR disciplined OR PMBOK)

3.2.2 Inclusion and Exclusion Decisions

We searched for experience reports, theoretical papers, and empirical papers. To include a paper in this review, the paper must have been peer reviewed, must have been available online, must have been given permissions to access, must have been written in English, and must have reported on either the integration of Agile and SPI or the integration of Agile and Traditional Project Management. Excluded from the search were editorials, prefaces, news, summaries of tutorials, panels, comments, and poster sessions. Moreover, we include only one of the papers that have the same or continuous stories but appeared in different publications. All the papers that clearly did match the inclusion criteria were excluded. This search strategy resulted in a total of 3,730 papers in the first stage. Figure 3-1 presents the systematic review process and the number of papers identified at each stage.

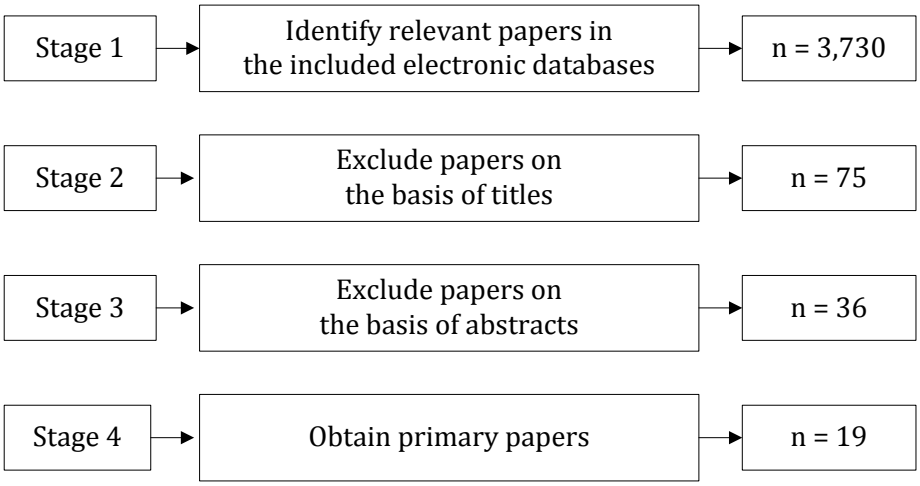


Figure 3-1. Stages of the primary paper selection process

In the second stage, the authors went through the titles of all obtained papers from the first stage to determine their relevance to the systematic review. At this stage, papers with titles that clearly indicated that the papers were outside the scope of this systematic review were excluded. Nevertheless, titles may not always represent what a paper is about. At this stage, there are 75 relevant papers. At stage 3, papers were excluded if their focus was clearly not on either agile software development integration with software process improvement or with traditional project management. At this stage, 19 papers were left for stage 4, enlisted in the next section.

3.2.3 Final Selection

There are three main screening criteria used to ensure the papers address our research topic as follows.

1. Does the paper address software process improvement or project management in agile software development?
2. Is there a clear statement of the research aims?
3. Is there a clear statement of contributions or findings?

These three criteria provided a measure of the extent to which we could be confident that a selected paper could make a valuable contribution to the review. Each criterion was graded on a dichotomous ("yes" or "no") scale. We finally selected 19 papers out of the 36 papers by performing the quality assessment based on the three screening criteria. We accepted a paper graded "yes" on all the three criteria.

3.2.4 Data Extraction and Synthesis

During this stage, a predefined data extraction form was used to extract data from each of the papers in this review. The following data was extracted from each paper: publication identification, electronic database, paper type and name (i.e., journal or conference), title, authors, year, abstract, type of study (i.e., theoretical papers, empirical papers, and experience reports), research aims, research methodology, and main research results (i.e., proposed approaches, methods used, lessons learned, recommendations, and limitations). This form helped to extract all needed details. Moreover, we synthesized the data by identifying themes emanating from the contributions or findings of each of the reviewed papers. There are two main themes: (i) agile software development integration with SPI and (ii) agile software development integration with traditional project management. The descriptions of this synthesis are presented in the next section.

3.2.5 Threats to Validity

The main threats to validity of this study are publication bias, selection bias, and possible inaccuracy in data extraction which are described as follows.

Publication Bias: Only five electronic databases were selected for this review. This limits the possibility to generalize the results. However, the electronic databases we selected contain peer reviewed publications in the field of information technology, including information systems, software engineering, and software process improvement. Some also include the most highly cited publications in the field. As major publications in the field are included in the review, this threat should be limited.

Selection Bias: The selection of papers from the five electronic databases is also a treat to the validity. First, the paper included in this review must have been peer reviewed, must have been available online, must have been given permissions to access, must have been written in English, and must have reported on either the combination of Agile and SPI or the combination of Agile and Traditional Project Management, based on our identified search keywords. Therefore, some relevant papers within this review boundary, published or stored outside our selected sources may be missed. As the papers included in this paper must have been peer reviewed, this makes the selection suitable for answering our research questions due to the obtained quality papers in the field. Both general search keywords (e.g., agile, plan-based, and disciplined) and specific search keywords (e.g., Scrum, CMMI, and PMBOK) were also used to search in both metadata (i.e., titles, abstracts, and indexing terms) and full text. This helps improving the search results. Second, the inclusion criteria used to include papers in this review are based on a reading of the titles and abstracts in the first three steps of our primary paper selection process. This introduces a threat, as the titles and abstracts may not reflect the actual contents of the papers. This treat was investigated, as described in the sub-section "Inclusion and Exclusion Decision", and found to be limited.

Possible Inaccuracy in Data Extraction: A potential threat to validity is the subjective judgment used to include or exclude papers and extract data from the selected papers. To limit this treat, we used three main criteria providing a measure of the extent to which we could be confident that a selected paper could make a valuable contribution to the review. We also used a predefined data extraction form to extract all needed details of each selected paper, based on the research questions. Besides, the selected papers were classified giving authors the benefit of the doubt. For instance, the selected papers were classified in accordance with what is addressed in the selected papers. This is beneficial for the subsequent analysis. The majority of the selected papers provide either theoretical contributions with positive results of empirical evaluations or empirical contributions that are highly relevant to the objectives of this review. Therefore, the accuracy bias should be limited.

3.3 Results

3.3.1 Overview of the Reviewed Papers

Considering publication years in Table 3-2, the results show that the trend of integrating agile with disciplined methods is increasing since 2008. It can be argued that the publication trend may be an indicator of researchers' and practitioners' growing interest in this matter.

Table 3-2. Reviewed papers by year interval

Year	2004	2005	2006	2007	2008	2009	2010	Total
No. of papers	1	1	1	2	5	3	6	19
Percent	5.26	5.26	5.26	10.53	26.32	15.79	31.58	100

Considering publication types in Table 3-3, the results show a similar degree of theoretical and empirical (including empirical papers and experience reports) evidence. It can be argued that there is a high likelihood to build a workable theoretical solution of agile-and-disciplined-method integration for real-life practice.

Table 3-3. Types of the reviewed papers

Type of Papers	Theoretical Paper	Empirical Paper	Experience Report	Total
No. of Papers	10	3	6	19
Percent	52.63	15.79	31.58	100

Considering research purposes of the reviewed papers, Table 3-4 summaries standard methods used in those papers which can be grouped into three categories: software process improvement methods, agile software development methods, and traditional project management methods. Almost all of those papers have used more than one standard method. In each category, the results emphasize that CMMI, Scrum, and PMBOK have gained the

most recognition in the area of agility and discipline integration. However, almost all disciplined methods (i.e., Automotive SPICE, CMM/CMMI, IDEAL (Initiating, Diagnosing, Establishing, Acting, and Leveraging), ISO/IEC 15504, MSF (Microsoft Solutions Framework) for CMMI Process Improvement, PSP (Personal Software Process), and QIP (Quality Improvement Paradigm)) have been used to integrate with agile processes for an SPI purpose more than to enhance project management in agile software development. This implies that more theoretical and empirical evidence on the latter purpose is still required.

Table 3-4. Standard methods used in the reviewed papers

Software Process Improvement			**Agile Software Development**			**Traditional Project Management**		
Methods	**No. of Papers**		**Methods**	**No. of Papers**		**Methods**	**No. of Papers**	
Automotive SPICE	1	5.56%	Lean Software Development	1	5.88%	PMBOK	1	100%
CMM/CMMI	11	61.11%	MSF for Agile Software Development	2	11.76%	-	-	-
IDEAL	1	5.56%	RUP	1	5.88%	-	-	-
ISO/IEC 15504	1	5.56%	Scrum	7	41.18%			-
MSF for CMMI Process Improvement	2	11.11%	TDD	1	5.88%	-	-	-
PSP	1	5.56%	XP	5	29.41%	-	-	-
QIP	1	5.56%	-	-	-	-	-	-
Total	**18**	**100%**	**Total**	**17**	**100%**	**Total**	**2**	**100%**

3.3.2 Findings about Research Questions

The descriptions of the reviewed papers can be summarized into Table 3-5.

Table 3-5. Descriptions of the reviewed papers

Id	**Paper**	**Type**	**Description**
RP1	Anderson [132]	Experience	As agile developers often sceptically perceive formal process improvement initiatives as management generated inefficiency, this paper has thus adopted teachings of W. Edwards Deming and stretched the Microsoft Solutions Framework (MSF) for agile methods to fit the requirements for CMMI level 3. The result is a process template which is larger than a typical agile process with slightly more formality; however, it is lightweight comparing to CMMI. This paper also shows that it is possible to develop a truly agile full life cycle process which meets the requirements for all 5 CMMI maturity levels.
RP2	Baker [133]	Experience	This paper reports experience on DTE Energy's agile IT organization's journey passing two SCAMPI (Standard CMMI

Id	Paper	Type	Description
			Appraisal method for Process Improvement) appraisals towards formal CMMI Level 3 accreditation. This paper also offers three suggestions on embracing a formal process framework that are applicable to any organization. First, it is important to clarify why to seek to improve software processes at all, and why to leverage a framework like the CMMI. The case for change is vital and a shared vision of the future will clarify each decision. Second, it is to clarify who is doing the process improvements. Third, it is to clarify how to go about developing, deploying, and maintaining process improvements.
RP3	Callegari and Bastos [8]	Theoretical	This paper presents a model for software project management based on PMBOK and its integration with RUP. This is in order to provide an adequate combination of project management and software development processes for developing a software product with quality.
RP4	Cohan and Glazer [134]	Experience	This experience report describes steps to improve agile development disciplines, which are deemed to lead to being appraised at CMMI maturity level 5, using an SCAMPI method for measurement.
RP5	Diaz et al. [22]	Experience	This paper reports empirical results that confirm the theoretical comparisons between Scrum and CMMI. The results show that process areas related to CMMI-DEV (CMMI for Development) level 2 were largely covered. In other words, Scrum provides criteria to identify a minimum set of good practices to achieve CMMI maturity level 2.
RP6	Jakobsen and Johnson [23]	Experience	This paper reports experience on how the generic practices from CMMI can be used to institutionalize Scrum. This paper also recommends twelve activities to extend agile methods inspired from an understanding of the mandatory goals and expected practices for CMMI levels 2 and 3.
RP7	Jakobsen and Sutherland [100]	Experience	This paper evidences that projects integrating Scrum with CMMI can bring a more powerful combination of adaptability with predictability than either one alone.
RP8	Kähkönen and Abrahamsson [135]	Empirical	This paper explores an empirical case where a project using XP was assessed using CMMI. The analysis covers specific goals and practices corresponding to CMMI maturity level 2 process areas (excluding Supplier Agreement Management) and use the Nokia CMMI-Based Process Assessment method (based on SCAMPI) to assess software processes. The results reveal that it is possible to use CMMI for assessing and improving agile processes to achieve maturity level 2. However, CMMI does not always support interpretations in an agile context.
RP9	Khan et al. [136]	Theoretical	This paper present mapping between XP practices and CMM key process practices. Although the results show that XP is partially compatible with CMM, the paper suggests that small-and-mid-sized companies should go for adaptation of agile methods for excellent performance on the footsteps of CMMI. Those companies can move towards CMM after agile maturity.
RP10	Leithiser and Hamilton [137]	Theoretical	This paper proposes a tool-based approach, using the Team Foundation Server (TFS) product to leverage the Microsoft Solutions Framework (MSF), support both CMMI and agile as process templates, and providing tools around those templates to automate the use of either approaches. The paper also presents a flowchart to assist in formulating the decision to utilize the CMMI template and a matrix that outlines the key factors involved in the decision (i.e., traceability, quick return of investment, auditing, resource shortage,

Id	Paper	Type	Description
			time shortage, requirements known in advance). Similarly to many researchers' viewpoints, CMMI provides "what" while agile methods provide "how".
RP11	Marçal et al. [24]	Theoretical	This paper presents mapping between CMMI and Scrum, showing major gaps between them and identifying how to adopt complementary practices to make these two approaches more compliant. The results reveal that few adaptations on Scrum mainly related to agile risk management, issues management and estimate methods make it much more compliant with CMMI project management process areas. The results are useful for organizations that have CMMI plan-driven processes and are planning to improve the agility of processes.
RP12	McCaffery et al. [138]	Theoretical	This paper describes a new low-overhead assessment method designed specifically for Small-to-Medium-sized Enterprises (SMEs) wishing to be automotive software suppliers. This assessment method integrates CMMI process areas, Automotive SPICE processes, and several agile practices. The assessment method consists of eight stages: developing assessment schedules and receiving site briefing, conducting overview briefing, analyzing key documents, interviewing key staff members, generating assessment results and creating the findings report, delivering the findings report, developing an SPI path, and re-assessing the SPI path and produce a final report.
RP13	Omran [139]	Theoretical	This paper presents mapping between XP practices and CMMI key process areas. Albeit the results show that XP is partially compatible with CMMI, this paper suggests that small-and-mid-sized companies can adopt agile methods whilst following the CMMI standard to gain new additional competence values in their environments.
RP14	Petersen and Wohlin [140]	Theoretical	This paper proposes a Software Process Improvement through lean Measurement (SPI-LEAM) method aiming at enabling continuous software process improvement leading to a lean software process; and avoiding problems related to resistance of change by improving in a continuous manner. The method combines the Quality Improvement Paradigm (QIP) with lean measurements and consists of six steps: characterizing the current project, setting quantifiable goals and measurements, choosing process models and measurements, executing processes and collecting and validating data, analyzing data and recommending improvements, and packaging and storing experiences made. However, main contribution of this paper is to present a solution to the second step for lean software development.
RP15	Pino et al. [141]	Theoretical	This paper proposes a "Lightweight process to incorporate improvements", using Scrum and aiming to give guidelines for supporting the management and performance of the incorporation of improvement opportunities within software processes. The proposed process aims to improve the organization's processes as appropriate to its particular business objectives and to assist it in carrying out its SPI by focusing on small companies. It is constructed to cover four main principles: early and ongoing achievement of improvements, continuous and rapid process diagnosis, elemental process measurement, and effective group collaboration and communication continuous learning. It provides the relationship between its entities and Scrum elements and defines a guide in many steps using Scrum, i.e., planning the iteration, designing the improvement case, executing the improvement sprint, presenting improved process, and presenting next improvement iteration.

Id	Paper	Type	Description
RP16	Rong et al. [142]	Theoretical	This paper proposes an integrated process model, Scrum-PSP, combining the strengths of each. The results of its verification in a real project environment where typical agile processes are favored (e.g., rapid development and fast delivery) show that manageability and predictability which traditional plan-driven processes usually benefit can also be achieved. Scrum-PSP is designed as two layers. First, the lifecycle layer describes the main process framework consisting of several iterations that turn customer requirements into final products. Second, the iteration layer describes five steps within an iteration. These are launch/re-launch, plan, requirement and design, construction, and iteration post-mortem.
RP17	Salo and Abrahamsson [143]	Empirical	This paper proposes an iterative improvement process for conducting SPI within agile teams for increasing the ability of software developers to improve the development process bases on their experiences and context knowledge. The iterative improvement process consists of six steps: preparation, experience collection, planning of improvement actions, piloting, follow-up and validation, and storing.
RP18	Williams et al. [144]	Empirical	This paper proposes a survey-based assessment tool, Comparative Agility (CA), to assist organizations in determining their relative agility compared with other teams who respond to CA. The CA approach assesses agility on seven dimensions: teamwork; requirements; planning; technical practices; quality; culture; and knowledge creating. Each dimension is made up of several characteristics and each characteristic has approximately four statements that are assessed by the respondents. From the results, industry trends indicate the highest adoption of agile practices occur in the areas of embracing emergent requirements and creating knowledge throughout the iteration and release; whilst the lowest industry adoption is relative to using technical practices and focusing on quality throughout all iterations. The results show that work progress of teams that use only Scrum practices eventually slows because the team has not paid enough attention to the quality of the code produced during each iteration.
RP19	Zaki and Moawad [145]	Theoretical	This paper proposes a new hybrid model that merges agile with traditional methods to overcome their shortcomings and make use of their strengths. The model consists of six phases: (1) Inception, to set up a project with five main activities which are start-up activities, aspects evaluation activities, gathering requirements and building backlogs, architectural activities, and conducting a prototype; (2) Planning, to set up the project boundaries; (3) Iterative Assessment, to customize agile and traditional processes; (4) Iterative Building, to build the product; (5) Production, to deliver the product; and (6) Closure, to close the project when there are no longer new requirements for implementation, when the product is not delivering the desired outcomes, or when the product is too expensive for further development.

A. Which existing research results on agile software development integration with software process improvement and with traditional project management are available that we can build on?

Answering the RQ3-1 "Which existing research results on agile software development integration with software process improvement and with traditional project management are available that we can build on?", we separate the answer into two themes: (i) agile software development integration with SPI and (ii) agile software development integration with traditional project management. Focusing on the first theme of agile software development integration with SPI, based on the review results there are three proposed directions to improve software processes in agile software development as presented in Table 3-6.

Table 3-6. Proposed directions to improve software processes in agile software development

Research Direction	No. of Papers	Percent
1. Providing a possible way to combine SPI standard processes with agile processes	13	72.22
2. Proposing a new software improvement process in agile software development	2	11.11
3. Proposing a software process assessment approach	3	16.67
Total	**18**	**100**

Table 3-6 shows that a number of the reviewed papers provide guidelines to combine SPI standard processes with agile processes, especially CMMI with Scrum. Focusing on compatibility of CMMI and Scrum, the results on a consolidated view of the coverage of CMMI project management process areas by Scrum practices reported in Marçal et al. [24] reveal that 32.8% of specific practices of CMMI project management process areas are satisfied, 16.4% are partially satisfied, and 50.8% are unsatisfied. This shows that Scrum is not fully compliant with CMMI project management process areas, mainly related to supplier agreement management, risk management, and quantitative project management process areas. Besides, in relation to risk management experience, 23.2% of organizations have little or no experience. From the ones that have it, there are 34% based their processes on PMBOK, 21% on CMMI, 24% do not use any model, and 21% do not manage risks. This emphasizes that most of them follow a plan-based method (i.e. PMBOK or CMMI) for risk management. More specifically, the results show that CMMI project management process areas corresponding to maturity level 2 have 43.8% of its specific practices satisfied by Scrum, 21.9% are partially satisfied, and 34.4% are unsatisfied. Those corresponding to maturity level 3 have 28.6% of its specific practices satisfied by Scrum, 14.3% are partially satisfied, and 57.1% are unsatisfied. Finally, those corresponding to maturity level 4 are 100% unsatisfied by Scrum. Supporting these results, Diaz et al. [22] and Jakobsen and Sutherland [100] show positive results of blending CMMI levels 2 and 5 to Scrum practices, respectively. This helps us to design our CMMI-based model emphasizing on four maturity levels (i.e., Level 1 "Initial", Level 2 "Managed", Level 3 "Defined", and Level 5 "Optimizing").

Moreover, McCaffery et al. [138] state that SPI provides that first step to move towards software quality and assessments are a critical part of this process, whilst Khan et al. [136] suggest that an agile method should be adopted as prerequisite to CMM/CMMI. In other words, an organization wishing to adopt CMM should start agile software development

at the beginning. This is because CMM is expensive to implement due to documentation, training, and so forth; an organization should thus commence with agile software development for cost reduction and rapid software project completion. From these points of views, it leads us to conduct an assessment approach to guide practitioners to improve their software process and prepare for achieving CMMI-based process improvements.

Considering three assessment methods of the reviewed papers, they have different purposes. One has been proposed by McCaffery et al. [138] to provide what process areas are most applicable. One has been proposed by Petersen and Wohlin [140] to assess the performance of the development process and take continuous actions. Their lean measurement method consists of two parts. The first part is concerned with setting quantifiable goals and measuring and analyzing individual inventories (i.e., requirements, test cases, change requests, faults and failures, and fault-slip-through). The second part is concerned with the analysis of the situation aiming at determining the causes for high inventory level and quality problems. Another one has been proposed by Williams et al. [144] as a tool used by individuals and organizations needing to compare their own agility to that of others. This tool assesses agility on seven dimensions: Teamwork, Requirements, Planning, Technical Practices, Quality, Culture, and Knowledge Creating. With our aim of improving agile processes, our assessment approach is thus intended to provide what agile processes need improvement.

Like software, software processes need to be evolved lest they become inefficient or obsolete. It is essential to maintain software processes so that their maturity can be improved. The results in Table 3-6 show that there are two possible ways to maintain and improve software processes. First, like a traditional software project, project retrospectives have been one way to yield process knowledge from the finished projects, so that SPI can be performed on future projects. Second, SPI can be conducted iteratively throughout an agile software project. From this view, it would be better to raise a better degree of software process and product quality by combining agile and traditional SPI ways together.

Focusing on the second theme of agile software development integration with traditional project management, Callegari and Bastos [8] propose an integrated project management and software development model based on PMBOK and RUP. This integrated model shows a great level of compatibility and provides us with a direction to combine PMBOK with Scrum. Zaki and Moawad [145] propose a new hybrid agile-disciplined software process model aiming to overcome agile and traditional methods' shortcomings and make use of their strengths. The model consists of six phases starting from inception, planning, iterative assessment, iterative building, production, and closure. Based on the above models, the results reveal that traditional project management processes is highly compatible with agile processes and important in the planning stage of software development. Practitioners can select and customize project management processes (especially in the areas of cost, risk, and procurement management) to fulfill agile weaknesses, when applicable. For a software process customization, practitioners should iteratively evaluate and establish feasible solutions to deal with any occurring issues.

B. What are some interesting aspects that existing research results on agile software development integration with software process improvement and with traditional project management do not yet cover?

Answering the RQ3-2 "What are some interesting aspects that existing research results on agile software development integration with software process improvement and with traditional project management do not yet cover?", the following three interesting aspects are found. First, most of the reviewed papers propose SPI approaches by mapping software processes related to CMMI Key Process Areas (KPAs) with agile practices, especially Scrum. Albeit there is a consensus on a high compatibility to customize plan-driven processes into agile software development and a number of studies pay attention to investigating influential factors impacting successful agile software development (as presented in Chapter 2), the findings of this review show that no study emphasizes on dealing with those influential factors in order to get agile software processes continuously improved and become more mature.

Possible Solution: In the CMMI-based SPI area and found that success in a software process can be viewed in terms of KPAs and CSFs. A number of studies argue that KPA approaches should improve the organization's capabilities to manage, develop, and deliver quality software products [107-111]. On the contrary, a multitude of studies concur that a successful software process should be viewed in terms of CSFs rather than KPAs. These studies emphasize the importance of the CSF approach in SPI and the use of the CSF approach rather than the KPA approach [20, 112-115]. They have also confirmed the value of the CSF approach in the field of information technology [20, 113-119]. Considering SPI CSFs identified in existing studies, albeit a deep comparison analysis of existing SPI CSFs has not been performed in this chapter, the findings reveal that there is a similarity between SPI CSFs identified in existing studies and our CSFs identified in Table 2-4 in Chapter 2, e.g., management commitment, training, staff involvement, and experienced staff (team members' capability) [114, 146-149]. Moreover, many studies cite the "reviews" factor that has a major impact on successful SPI [114, 146-148] and some of these studies also cite that it is only one factors corresponding to the top CMMI-based maturity level [20, 114, 115, 150-152]. The "reviews" factor is hence considered as an additional CSF of this study. From this view, it emphasizes that there is a need to develop a mechanism that could assist practitioners in dealing with agile software development problems and simultaneously get higher maturity with a set of specific agile practices, by extension the application of the CSF concept for SPI.

Second, in relation to agile software development integration with traditional project management, the main objective of the relevant, reviewed papers is to overcome weaknesses of both agile and traditional project management methods. However, the weaknesses stated in the reviewed papers are addressed as generic weaknesses.

Possible Solution: Particularly to Scrum and PMBOK integration, it is necessary to understand Scrum weaknesses. We performed an additional review on this matter and found as follows.

1. Scrum neither specifies configuration management which is crucial for correct individual work and continuous integration [153-155] nor supports procurement management [98].
2. Companies with a large customer base, especially in the e-service sector, need both rapid value and high quality assurance. Using either agile or traditional methods cannot meet these needs [156]. This implies that Scrum does not highly support high quality assurance software.

3. Customer needs to have a clear sense of the product's direction; if not, the final product can significantly differ from what is expected [9].
4. Unless there is a definite end date, Scrum is one of the leading causes of scope creep [157].
5. Scrum has relatively low visibility over the project outside sprints which means that it is very difficult to estimate how long a project will take or how much it will cost [9]. In other words, Scrum mentions cost estimation during an iterative planning. It might be a problem if cost is a constraint for the whole project [157].
6. The risk involved in the software project is significant if customers cannot intervene in the software project in a relatively long period [9]. On the other hand, the software project may be never complete or fail if team members' commitment does not exist [157].
7. Scrum requires experienced team members. Hence, the software project may not be completed in time if team members are novices [157].
8. With agile, there is no project milestone. This may somehow lead to a problem for large agile projects [154]. McMahon [154] suggests that a milestone event should be established as a checkpoint to ensure collaboration is really happening.
9. Early and continuous delivery of valuable software can benefit small software projects [9, 156, 157], but over-focus on early results especially in large projects can lead to big trouble when the architecture does not scale up [156].
10. Scrum neither clearly details unit and acceptance tests nor explicitly discusses code styles (e.g., clean and simple), technology environments (e.g., quick feedback required), physical environments (e.g., co-located and distributed teams), and business cultures (e.g., collaborative and cooperative) [6, 155].

Coping with Scrum weaknesses in managerial aspects, PMBOK is more likely to be of assistance. However, technical software processes in Scrum are still missing. We thus need to search for possible agile practices to fulfill and overcome technical problems. This gap is also related to the first gap in which a set of specific agile practices is required to define and design for success in software development.

Third, although the findings reveal that there is a great possibility to apply PMBOK for risk management in agile software development, and a great compatibility between PMBOK and Scrum methods; all of the reviewed papers neither specifically offer an integrated PMBOK-Scrum approach nor apply it in real-life software projects.

Possible Solutions: An integrated PMBOK and Scrum model needs to be developed and tested to ensure its usability and practicality. In the model, integrated PMBOK-Scrum processes are performed in the Scrum way.

These three interesting aspects serve as the identified gaps that are worth addressing from a research point of view for this study. These help provide more details on our conceptual software process maintenance framework.

C. How should a software process maintenance framework be constructed? Is a software process maintenance framework workable? What does the test of a software process maintenance framework in a real-life situation contribute?

The above answers to the RQ3-1 and RQ3-2 help answer how to construct software process maintenance framework which in this context means a framework for software process development and improvement. In the framework, there are two core components: an SDM model as a CMMI-based software process improvement component and an integrated PMBOK-Scrum model as an integrated software process development component.

For the SDM model, in order to provide practitioners a systematic structured mechanism to assess and improve software processes to get into a particular maturity level, the staged representation of CMMI and CSF approaches are employed. At this stage of this study, four CMMI maturity levels of "Initial", "Managed", "Defined", and "Optimizing" are adopted for a software development maturity model of this study. The main reason not to replicate the CMMI maturity level-4 "Quantitatively Managed" is that the two key CMMI practices of (1) establishment and maintenance of quantitative objectives for the process and (2) stabilization of the performance of one or more sub-processes to determine its ability to achieve are not compatible with agile best practices [158]. Additionally, there is no success factor cited in literature and empirical studies that directly relates to this level [20, 114, 115]. Consequently, the maturity levels 1 through 4 of the model are "Initial", "Managed", "Defined", and "Optimizing", respectively.

The 13 identified CSFs (including 12 CSF affecting the successful agile software development identified in Chapter 2 and the additional factor of "reviews" described above) are categorized into their corresponding maturity levels. There is no category for the level-1 "Initial" since this level does not have to be achieved due to its chaotic characteristic. Similarly, CMMI does not have process areas for this level. The level-2 "Managed" contains the identified CSFs supporting project management activities which are the foundation for all subsequent levels. This level contains management commitment, project management process, project type, training support, and user involvement. The level-3 "Defined" contains the identified CSFs that support to design systematic structures for SPI implementation (i.e., agile software development process, appropriate methods, techniques, and tools, data quality, organizational environment, team capability, team environment, and team size). The level-4 "optimizing" contains the "reviews" factor to support continuous SPI activities. We considered CSFs' categories regarding SPI implementation process of prior empirical studies [20, 114, 115] in order to have more confidence in the CSF categorization. In order to guide practitioners how to implement or improve the identified CSFs, agile-oriented practices need to be designed and mapped with their related influential CSFs. Figure 3-2 presents the proposed SDM model by giving details on the CMMI-based software process improvement component.

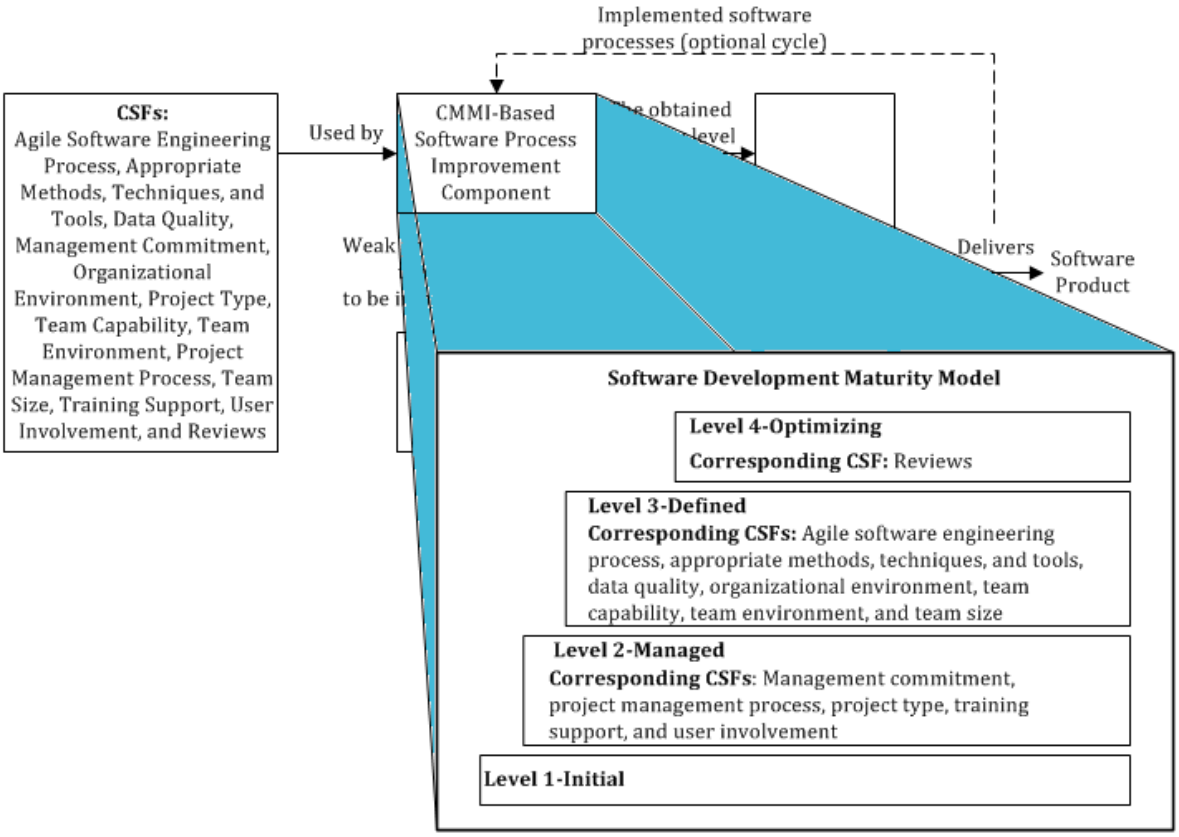


Figure 3-2. The proposed software development maturity model

For the integrated PMBOK-Scrum model, the Scrum and PMBOK processes are mapped together for iterative initiating, planning, executing, controlling, and closing. Figure 3-3 presents the proposed integrated PMBOK-Scrum model by giving details on the integrated PMBOK-Scrum process development component.

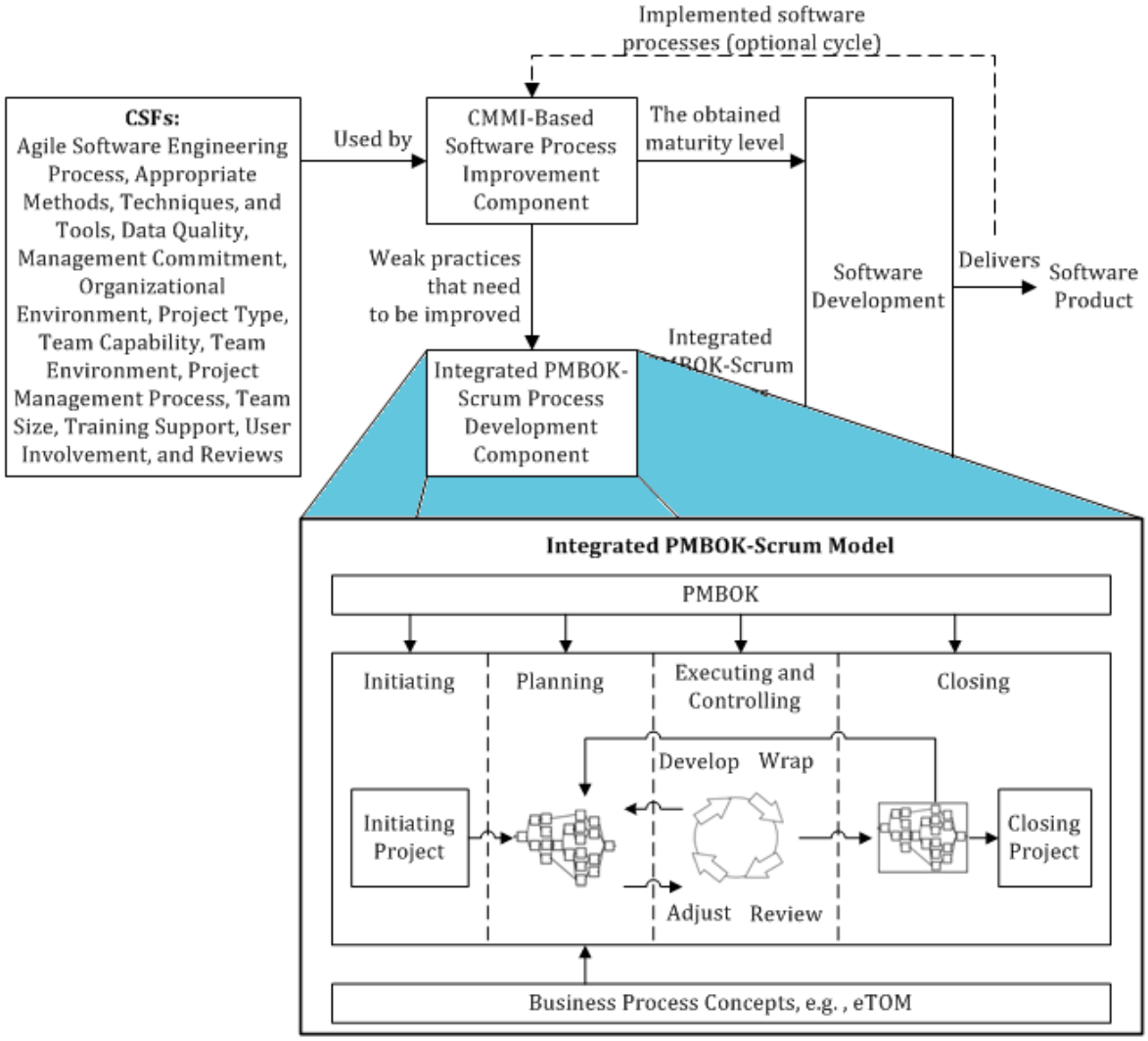


Figure 3-3. The proposed integrated PMBOK-Scrum model

Is a software process maintenance framework workable? What does the test of a software process maintenance framework in a real-life situation contribute? There are a number of models and theories of intentions to use an Information Technology/Information System (IT/IS) or an innovation that can be structured into three categories: adoption, acceptance, and innovation [159]. Adoption models and theories have been used to predict the decision to adopt an IT/IS or an innovation. Those deemed to be relevant to adoption are, e.g., Technology Organization Environment Framework by Tornatzky and Fleischer [160], IS Success Model by DeLone and McLean [161], and Fit-Viability Model by Liang et al. [162]. Acceptance models and theories have been used to measure the decision to accept and make use of an IT/IS or an innovation. Those deemed relevant to acceptance are, e.g., Theory of Reasoned Action (TRA) by Fishbein and Ajzen [163], Theory of Planned Behavior (TPB) by Ajzen [164], Technology Acceptance Model (TAM) by Davis [165], Model of Personal Computer Utilization (MPCU) by Thompson et al. [166], and Unified Theory of Acceptance and Use of Technology (UTAUT) by Venkatesh et al. [167]. Innovation models and theories have been used to explain the diffusion of an IT/IS or an innovation. Those deemed relevant to innovation are, e.g., Diffusion of Innovations by Rogers [168] and Perceived Characteristics of Innovating by Moore and Benbasat [169].

To measure the success and acceptance of the software process maintenance framework, acceptance models are deemed more appropriate than adoption models. As claimed by many researchers that the introduction of an IT/IS or an innovation is influenced by factors both controllable and uncontrollable by the organization, acceptance is thus deemed to be a prerequisite to diffusion [170]. Therefore, acceptance models are deemed for this study. Brief descriptions of the above acceptance models are presented as follows.

TRA, introduced by Fishbein and Ajzen [163], is one of the most fundamental and influential theories and used to predict behavioral intention and also explain technology acceptance [159].

TPB, introduced by Ajzen [164, 171], extends TRA and posits three conceptually independent determinants of intention which are (1) attitude, referring to an individual's degree of favorableness or unfavorableness towards the behavior in question; (2) subjective norm, referring to the perceived social pressure to perform or not to perform the behavior; and (3) perceived behavioral control, referring to the perceived ease or difficulty of performing the behavior.

TAM, introduced by Davis [165], is an extension of TRA and posits that success, acceptance, and usage of an IT/IS or an innovation are jointly determined by two factors. First, Perceived Usefulness (PU) refers to the degree to which a person believes that using a particular IT/IS would enhance his or her job performance. Second, Perceived Ease of Use (PEOU) refers the degree to which a person believes that using a particular IT/IS would be free of effort. However, PU was considered to significantly influence usage and user acceptance more than ease of use. TAM has been applied in abundant studies testing user acceptance of IT/IS, e.g., word processors, spreadsheet applications, e-mail, web browser, telemedicine, and websites [172]. Later, an extended version of TAM, called TAM2 [173], was introduced by subjective norm as an additional factor exerting a significant direct effect on usage intentions over and above PU and PEOU in mandatory settings. It refers the degree to which people think that others who are important to them think they should or should not perform the behavior in question.

MPCU, introduced by Thompson et al. [166], is based on the theory of interpersonal behavior and used to predict the way of usage rather than the decision to use [159]. It comprises social factors, affect towards use, complexity, job-fit, long-term consequences, and facilitating conditions. However, those factors are deemed suitable to predict individual acceptance [167].

UTAUT, introduced by Venkatesh et al. [167], integrate elements across the eight user acceptance models including TRA, TPB, TAM, the motivational model, the model combining TAM and TPB, MPCU, the diffusion of innovations theory, and the social cognitive theory. Those elements were formulated with four core determinants of intention and usage (e.g., performance expectancy, effort expectancy, social influence, and facilitating conditions), and up to four moderators of key relationships (e.g., gender, age, experience, and voluntariness of use). It is used to assess the likelihood of success for new IT introductions and help understand the drivers of acceptance in order to proactively design interventions, e.g., training and marketing.

Compared to the above models, TAM is a simple and very successful IT/IS acceptance model in terms of studying IT/IS success, acceptance, and usage intention [174]. We therefore consider original TAM for this study to measure whether or not our software process maintenance framework is perceived as workable and acceptable. TAM's PU and PEOU are used as key measurement variables of this study. To do so, it is therefore important

to carry out the empirical test of the proposed software process maintenance framework in real-life software development. Based on the empirical test, it should contribute to what parts or elements of the framework are working and are not working, what changes are necessary for adaptation of the framework, what challenges occurring during the application of the framework that should be concerned, how practitioners transfer new knowledge into existing software processes, and what requirements are necessary for successful adaptation of the framework. For the next steps, the proposed software process maintenance framework is constructed in Chapter 4. Its empirical test is performed in Chapter 5.

3.4 Summary

In this chapter, a gap analysis for the sound development of the proposed software process maintenance framework was performed through a systematic literature review in the field of agile software development integration with SPI and with traditional project management.

Focusing on the first theme of agile software development integration with SPI, the findings reveal two main interesting research results that we can build on. First, most of the reviewed papers propose SPI mechanisms by mapping CMMI key processes with agile practices, especially Scrum. According to the CMMI and Scrum, the findings show the positive theoretical and empirical results of blending CMMI levels 2, 3, and 5 to Scrum practices. This helps us to design the SDM model, emphasizing on these four maturity levels. Second, some researchers suggest that an agile method should be adopted as prerequisite to CMM/CMMI. As the first step to move towards SPI is software process assessment, this leads us to conduct an assessment approach to guide practitioners to improve their agile software process and prepare for achieving CMMI-based process improvements in future. From this point of view, we found two interesting aspects that those research results do not cover yet. First, albeit there is a consensus on a high compatibility to blend plan-driven Key Process Areas (KPA) into agile software development and a number of studies pay attention to investigating influential factors impacting the successful agile software development, the findings show that no study emphasizes on dealing with those influential factors in order to get agile software processes continuously improved and become more mature. Second, no study provides guidance to improve Scrum processes by coping with Scrum weaknesses in both managerial and technical aspects. Bridging these gaps, we thus need to search for agile-oriented practices to fulfill managerial and technical weaknesses. Those practices should also be mapped with the related influential factors in order to guide practitioners on the “what” to improve.

Focusing on the second theme of agile software development integration with traditional project management, the results reveal that traditional project management processes is highly compatible with agile processes and important in the planning stage of software development. Practitioners can select and customize project management processes (especially in the areas of cost, risk, and procurement management) to fulfill agile weaknesses, when applicable. For a software process customization, practitioners should iteratively evaluate and establish feasible solutions to deal with any occurring issues. Although the findings reveal that there is a great possibility to apply PMBOK in agile software development (i.e., Scrum software development in particular), all of the reviewed papers neither specifically offer a theoretical integrated PMBOK-Scrum model nor apply it in real-life software projects. Bridging this gap, we thus need to develop a theoretical integrated PMBOK-Scrum model.

All of the findings above help suggest on how to construct a software process maintenance framework, meaning a framework for software process development and improvement. The framework consists of two core components: the SDM model and the integrated PMBOK-Scrum model. First, the SDM model provides practitioners a systematic structured mechanism to assess and improve software processes to get into a particular maturity level, based on the staged representation of CMMI and CSF approaches. In the model, four CMMI maturity levels of "Initial", "Managed", "Defined", and "Optimizing" are this stage adopted. The main reason not to replicate the CMMI maturity level-4 "Quantitatively Managed" is that the two key CMMI practices of (1) establishment and maintenance of quantitative objectives for the process and (2) stabilization of the performance of one or more sub-processes to determine its ability to achieve are not compatible with agile practices. Consequently, the maturity levels 1 though 4 of the model are "Initial", "Managed", "Defined", and "Optimizing", respectively. To guide how to achieve a certain maturity level, we have categorized the 13 identified CSFs (including the 12 CSFs affecting the successful agile software development identified in Chapter 2 and the additional CSF of "reviews" identified in this chapter) into their corresponding maturity levels. This CSF categorization is based on CMMI objectives at each maturity level. To guide how to implement and improve the 13 identified CSFs, we have to design a list of agile practices under each CSF. Therefore, the results of the SDM model should guide practitioners on their current software development maturity level and weak practices that demand immediate improvement. Second, the integrated PMBOK-Scrum model provides guidance on how to implement with an adequate set of project management and software development processes. In the model, the PMBOK and Scrum processes are mapped together for iterative initiating, planning, executing, controlling, and closing. Therefore, the resulting framework is expected to provide guidance on the "what" and the "how" to improve and implement.

It is important to perform an empirical test to check whether or not the software process maintenance framework is perceived as workable and acceptable in real-life practice. As TAM is a very successful IT/IS acceptance model, TAM's PU and PEOU are used as key measurement variables to measure the usability and practicality of the framework. The empirical test is expected to contribute to what parts or elements of the framework are working and are not working, what changes are necessary for adaptation of the framework, what challenges occurring during the application of the framework should be addressed, how practitioners transfer new knowledge into their existing software processes, and what requirements are necessary for successful adaptation of the framework. For the next steps, the proposed software process maintenance framework is constructed in Chapter 4. Its empirical test is performed in Chapter 5.

Chapter 4

The Software Process Maintenance Framework

Related Publication:

P3: Porrawatpreyakorn, N., Quirchmayr, G., and Chutimaskul, W. 2010, 'A Prototype for the Support of Integrated Software Process Development and Improvement', in Papasratorn, B., Chutimaskul, W., Porkaew, K., and Vanijja, V. (eds), Proceedings of the 4th International Conference on Advances in Information Technology, Springer Berlin Heidelberg, Bangkok, Thailand, vol. 144, pp. 94-105.

An efficient and effective software development process is one of key success factors for quality software development processes and products. Not only can the appropriate development but also the continuous improvement of integrated project management and of the agile software development process result in software development efficiency and effectiveness. This chapter proposes a software process maintenance framework which consists of two core components. They are a software development maturity model advocating software process improvement and an integrated PMBOK-Scrum model offering a comprehensive set of project management and software development processes. A prototype tool supporting the use of the framework is also introduced.

4.1 Introduction

A software development process (hereafter referred to as "software process") consists of a set of practices in software development, together with management and organization needed for building a software product [175]. It is viewed as a vehicle to deliver the quality of software [14]. The software process should thus be efficient and effective. The software development efficiency and effectiveness requires both the development and the continuous improvement of integrated project management and of the software process [6, 176]. Capability Maturity Model Integration (CMMI) as a Software Process Improvement (SPI) approach has proved that the effort put into this method can assist in producing high quality software, reducing cost and time, and increasing productivity [114, 177-179]. Many studies also emphasize the importance and the use of Critical Success Factors (CSFs) in SPI rather than CMMI key process areas (e.g., CSFs help to extend the boundaries) [112-114, 180] and have confirmed the values of the CSFs approach in the area of Information Technology (IT) [113, 114]. Consequently, CMMI and CSFs are used as the basis for our proposed Software Development Maturity (SDM) model. As little attention has been paid to how to assess and implement CSFs in order to gain success in agile software projects, a literature survey and a questionnaire-style information collection on SPI, agile practices, and data quality are performed to investigate the following research questions.

RQ4-1: Are CSFs in software development, as identified in Table2-4 in Chapter 2, similar to CSFs in SPI identified in the literature?

RQ4-2: What agile practices, as identified in the literature on agile software development and data quality, should be implemented for successful software development?

RQ4-3: What agile practices, as identified in our questionnaire-style information collection, should be implemented for successful software development?

RQ4-4: How should a software development maturity model (as one of two core components of a software process maintenance framework) be constructed?

The answers to the RQ4-1 to the RQ4-4 provide advice in developing the SDM model. Additionally, agile methods that combine teamwork with an intense focus on effectiveness and maneuverability are ubiquitously applied for the rapid delivery of quality software [4]. However, they are not efficient enough in the managerial sense. Issues such as limited support for outsourcing, distributed development environments, developing with large teams, and developing software demanding high quality control still remain uncovered [9, 11]. As this study aims at minimizing changes of the software process which software development teams are already familiar with, the Project Management Body of Knowledge (PMBOK) and Scrum are accordingly used as the basis for our proposed integrated PMBOK-Scrum model. PMBOK is the broadest and most widely used standard reference of industry best practices for project management [38]. Scrum is one of the most popular agile methods [181]. However, Scrum itself has shortcomings. For example, Scrum neither specifies configuration management which is crucial for correct individual work and continuous integration [153-155] nor supports high assurance software and procurement management. Early and continuous delivery of valuable software can benefit small projects, but over-focus on early results especially in large projects can lead to big trouble when the architecture does not scale up [156]. Scrum mentions cost estimation during an iterative planning. It might be a problem if cost is a constraint for the whole project. Considering the software process lifecycle perspective, Scrum neither details an acceptance test nor explicitly discusses a code style (e.g., clean and simple), a physical environment (e.g., co-located and distributed teams), and a business culture (e.g., collaborative and cooperative) [6, 155]. These shortcomings could be reduced by plan-based methods (e.g., PMBOK and CMMI). This point of view leads to the following research question.

RQ4-5: How should an integrated PMBOK-Scrum model (as one of two core components of a software process maintenance framework) be constructed?

The SDM model and the integrated PMBOK-Scrum model are two core components of the software process maintenance framework, which in this context means a framework for software process development and improvement. Many organizations face either unfulfilled promises about software quality gained from applying software development methods, or the inability to manage the software process realized as their fundamental problem [182]. The search for solutions to this barrier has continued for decades. Dealing with this barrier, the software process maintenance framework could help by acting as guidance for quality software development. Nonetheless, the framework might be too complex without the right tools. Hence, we have created a prototype tool to support the use of framework called SPAD (Software Process Assessment and Development). It helps an end user (e.g., a project manager and a team leader) to get insight into the organization's current maturity by assessing the identified CSFs through the list of agile-oriented practices required by the SDM model. Weak practices as a part of the obtained assessment results will be used to plan the project together with the defined information (e.g., project, phase, and activity) required by the integrated PMBOK-Scrum model. The defined process is then validated and prepared in an eXtensible Markup Language (XML) file format for export to the organization's project

planning tools. This chapter is organized as follows. The following section describes the software process maintenance framework comprising the SDM model as an assessment component and the integrated PMBOK-Scrum model as an integrated software process planning component. The prototype tool supporting the framework is then presented.

4.2 The Software Process Maintenance Framework

In order to consistently deliver quality results, an efficient and effective software process requires both the development and the continuous improvement of integrated project management and of software development processes. This study hence proposes a software process maintenance framework as depicted in Figure 4-1.

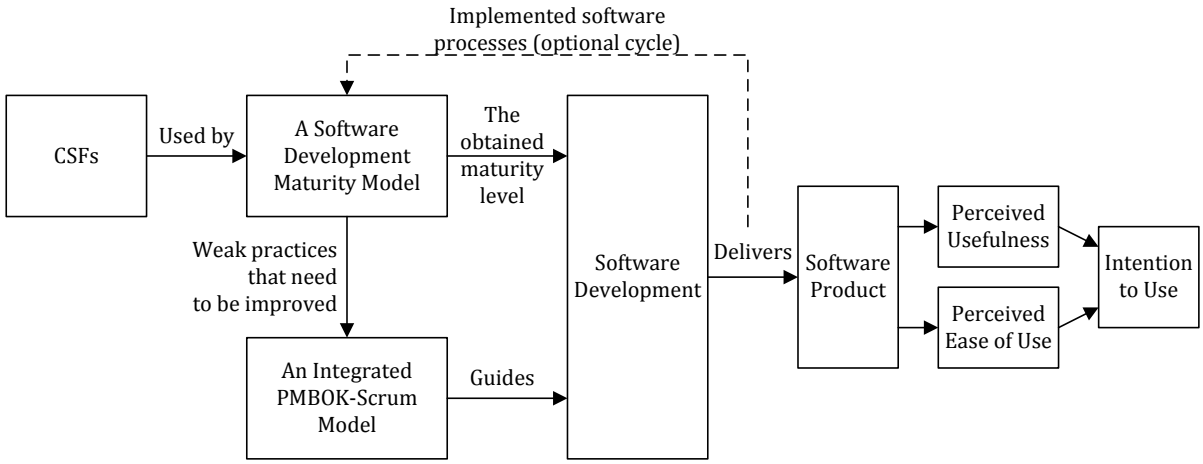


Figure 4-1. The proposed software process maintenance framework

The framework has paid attention to the "what", "how", and "how good" to improve and implement the software process through an SDM model and an integrated PMBOK-Scrum model which are described in the following sub-sections.

4.2.1 The Software Development Maturity Model

At the starting point of this study, the goal was to identify CSFs in software development [183]. In order to provide guidance in assessing and improving agile practices through the identified CSFs, we have performed a literature survey and a questionnaire-style information collection regarding SPI, agile software projects, and data quality. The descriptions of these surveys are presented as follows.

4.2.1.1 Research Design

A literature survey and a questionnaire-style information collection of our pilot study were performed in order to compare similarities and differences (i) between CSFs in SPI identified in the literature and CSFs in software development in the Thai Internet Service Providers identified in Table 2-4, Chapter 2 [2, 183] and (ii) between agile practices in worldwide software projects identified in the literature and those in software projects in three companies in Thailand, responding to the questionnaire-style information collection.

In relation to a literature survey, 55 sources including reports, case studies, and software process articles have been reviewed to investigate with respect to either CSFs in SPI or agile software projects that are recognized globally. References of the sources are shown in the next sub-section. As an exhaustive survey was not performed in this study, we do not claim to have captured all the relevant papers in the boundaries. This introduces the limited generalizability of the results. To reduce this threat to some extent, the primary focus is on empirical studies in multiple settings. In the reviewed papers, many companies are renowned for success in either SPI or agile software development (e.g., Boeing [179], Hughes [184], Motorola [112], Nokia, DaimlerChrysler, and ABB [185]) and had been surveyed from small to large organizations (e.g., [19, 186]) and worldwide (e.g., [34, 177, 187]) to investigate factors that influence the success in either SPI or agile software development. Besides, we have chosen the papers published from the late 1990's onwards. This is because we tried to eliminate the impractical CSFs in the field. Consequently, the accuracy of the resulted should be somewhat gained. Another limitation is a publication bias. To ameliorate it to some extent, we include peer reviewed literature and grey literature (e.g., working papers and technical papers) in this survey. As all of the reviewed papers are highly relevant to our survey objectives, we believe that they can be used as a representative sample of this study. To analyze, frequency analysis is used to extract quantitative data from the collected qualitative data.

As part of our pilot study, a questionnaire-style information collection concerning agile practices was also conducted in June 2010. The data was collected from seven respondents in three companies in Thailand including a Telecommunications player and two co-players. To preserve their anonymity, we refer to them here as Telecom Player1, Co-Player1, and Co-Player2. There are two main reasons to choose these three companies. First, the respondents have been doing agile software development on a daily basis. Second, they were all willing or voluntarily agreed to participate in this survey. Their profiles are described in Table 4-1. It is difficult to determine the exact number of Thai telecommunications players and co-players adopting agile methods. However, the current trend towards adopting agile methods in Thailand is just at the initial stages [188, 189], as supported by some respondent companies not adopting it as a whole and the majority of respondents indicating only a few years of agile experiences. As the sample size is very small, we do not claim that it is a representative sample and the results of this questionnaire-style information collection cannot prove anything. However, the main aim to carry out this questionnaire-style information collection is to find some identification by looking into a set of agile practices that are recognized as important for implementation in these three companies. This questionnaire-style information collection therefore has a bias towards co-play in the Thai telecommunications industry. By volunteering to participate, they have become a self-selecting sample that often leads to bias [114]. Albeit a perfect representative sample is unattainable, the authors should reduce as much of the sample bias as possible [190]. In order to limit the sample bias, the variety in company sizes (approximately 100 employees to more than 8,000 employees), years of agile experience (e.g., 1-3 years), and roles in agile software

development (e.g., project manager, Scrum master, hyper-productivity seeker, developer, and programmer analyst) have therefore been used. Similar approaches have been used by other researchers (e.g., Baddoo and Hall [191] and Niazi et al. [114]). Another limitation is that the practitioners' experiences have not been verified directly. Thus, their perceptions may not be accurate. However, the results give an interesting picture as described in the next sub-section.

In the questionnaire, 67 items were designed to determine which agile practices found in the reviewed literature were recognized as important for success in agile software development in the three companies in Thailand. Those items were assessed on a five-point Likert scale of importance, ranging from 1 "not at all important or not implemented" to 5 "extremely important and need to be implemented". To calculate the value of the importance of each item, the mode and the median can be used. However, the mode may return several values that have the same frequency. Therefore, the median is more appropriate and used for this calculation. Moreover, it is important to evaluate the instrument reliability and validity. Reliability is referred as the degree to which the scale is free from measurement error [192]. Hence, Cronbach's alpha was used to evaluate the reliabilities of the entire scale and each item. The evaluation results reveal that the Cronbach's alpha of the entire scale is 0.878 and the Cronbach's alpha of the items range between 0.864 and 0.888. All items having the coefficient of above 0.7 demonstrate acceptable reliability [192]. Albeit the content validity of this instrument was not formally evaluated, content relevance and completeness of this instrument was guided by using the results of the reviewed literature survey on agile practices and semi-open-ended questions. This instrument and its reliability test results are presented in Appendix B.

Table 4-1. Profiles of three respondent companies

	Telecom Player1	**Co-Player1**	**Co-Player2**
Area	Telecommunications	Software Company	Software Company
Primary Function	Services	Software/ Services	Software/ Services
Application Type	Telecommunications	Workshops, consulting, speaking, and software development	Software development and IT solutions addressing key business problems such as CRM, Customer Services, and E-Commerce in many industries such as banking and telecommunications
Applying Agile Methods	Some units	Whole company	Some units
No. of Respondents	4	2	1

4.2.1.2 Analysis and Results

In this sub-section, the results of all research questions are discussed. In order to answer the RQ4-1, Table 4-2 shows a list of CSFs that are identified in 24 SPI publications [19, 119, 121, 153-156, 181, 183, 188, 190, 191, 197-208] and listed in alphabetic order, using our 13 identified CSFs in agile software development [183] as a base.

Table 4-2. CSFs identified through the SPI literature

CSF	Occurrence in SPI literature (n=24)	Frequency	%	Rank
(Agile) software development process		14	58	5
Appropriate methods, techniques, and tools		13	54	6
Data quality		5	21	8
Management commitment		21	88	1
Organizational environment		16	67	3
Project management process		15	63	4
Project type		2	8	9
Reviews		10	42	7
Team capability		18	75	2
Team environment		14	58	5
Team size		-	-	-
Training support		16	67	3
User (staff) involvement		16	67	3

From the table, nine out of 13 CSFs had occurred more than 50%, three CSFs had not occurred very often, and only one CSF had not occurred at all. Albeit the majority of the CSFs did not have a very high number of occurrences in the literature, the results reveal that there are similarities between the two sets of the identified CSFs. This is the answer to the RQ4-1. The results also imply that the better the organization can implement these CSFs, the better the organization can achieve successful software development and higher maturity levels.

In order to answer the RQ4-2 and the RQ4-3, Table 4-3 summarizes agile practices under each identified CSF which are scrutinized and/or recommended for successful software development globally in the literature and locally in the questionnaire. In the literature on agile software development, most of them rarely address data quality aspects. To get more precise on the percentage of frequency occurrence, two sets of 23 publications on organization, people, process, project, and technology aspects in agile software development [4, 34, 57, 58, 60, 64, 65, 71, 185, 193-206] and 8 publications on the data quality aspect in software development [52, 207-213] are reviewed separately.

Table 4-3. Agile practices identified through the literature and questionnaire

List of Agile Practices	Occurrence in	Literature (n=23) Frequency	%	Questionnaire (n=7) Median
Agile Software Development Process (SD)				
SD1. A project has been established with well-defined coding standards up front.		5	22	4
SD2. A project has been established by pursuing simple design.		7	30	4
SD3. A project has been established with rigorous refactoring activities.		8	35	4
SD4. A project has been established with right amount of documentation.		14	61	4
SD5. A project has been established with correct integration testing.		18	78	5
SD6. A project has been established with short increments.		18	78	4
SD7. Most important features have been first delivered.		10	43	5

List of Agile Practices Occurrence in	Literature (n=23) Frequency	%	Questionnaire (n=7) Median
Appropriate Methods, Techniques, and Tools (MT)			
MT1. Appropriate methods, techniques and tools have been assessed and performed.	9	39	4
Management Commitment (MC)			
MC1. Management provides strong commitment and presence.	11	48	4
MC2. Management supports the software development.	13	57	5
MC3. Management is willing to participate in assessment and development activities.	6	26	4
MC4. Management is committed to provide training and resources.	2	9	4
Organizational Environment (OE)			
OE1. Cooperative organizational culture has been established instead of hierarchical culture.	13	57	5
OE2. Oral culture placing high value on face-on-face communication has been established.	11	48	4
OE3. Agile has been promoted and accepted throughout the organization.	5	22	4
OE4. All the key stakeholders are involved in development and improvement activities.	5	22	5
OE5. Management has provided strong leadership-collaboration; meaning management understands that collaboration on information to make informed decisions and trusting individuals to apply their competency in effective ways is important.	13	57	5
OE6. Facility with proper agile-style work environment has been established.	13	57	4
OE7. Reward system appropriate for agile software development has been promoted amongst the management and team members.	3	13	4
Project Management Process (PM)			
PM1. Agile-oriented project management process has been followed.	9	39	4
PM2. Cost evaluation has been done up front.	2	9	4
PM3. Risk analysis has been done up front.	2	9	4
PM4. A process has been established to monitor and track the progress of the project.	9	39	5
PM5. Strong face-to-face communication has been established as a primary communication method.	16	70	4
PM6. Teams have honored their regular working schedule.	5	22	5
PM7. Work has been done to continuously improve a project management process.	9	39	4
Project Type (PT)			
PT1. Project characteristics (e.g., extreme, complex, or high-change) have been assessed for the suitability of software process development.	13	57	4
PT2. Project criticality (e.g., life-critical and non-life-critical) has been assessed for the suitability of software process development.	11	48	4
Reviews (RE)			
RE1. Organization has developed a review process for development and improvement requirements.	7	30	4
RE2. Work has been done to continuously monitor existing software development processes.	8	35	4
RE3. Organization has developed a process in order to review each influential factors of software development.	1	4	4
RE4. Responsibilities have been assigned to conduct continuous	1	4	4

List of Agile Practices Occurrence in	Literature (n=23) Frequency	%	Questionnaire (n=7) Median
software process development and improvement reviews within organization.			
RE5. All the key stakeholders are involved in software process development and improvement reviews.	3	13	4
Team Capability (TC)			
TC1. People have been selected as team members who have high competence and expertise.	18	78	5
TC2. People have been selected as team members who have great motivation.	21	91	4
TC3. People have been selected as project managers or team leaders who have an adaptive management style.	20	87	4
TC4. People have been selected as project managers or team leaders who are knowledgeable in an agile process.	18	78	4
TC5. People who have track record of different successful projects have been selected for development activities.	-	-	4
TC6. Role and responsibilities have clearly been assigned to each team member.	3	13	5
TC7. A process has been established to monitor the progress of each team member.	6	26	5
TC8. A process has been established to collect and analyze the feedback data from each team member and to extract the main lessons learned.	13	57	5
Team Environment (TE)			
TE1. Collocation of the whole team has been established.	22	96	4
TE2. Coherent and self-organizing teamwork has been established.	23	100	5
TE3. A project has been established with no multiple independent teams.	8	35	4
TE4. A process has been established to monitor the progress of each team.	8	35	4
TE5. A process has been established to collect and analyze the feedback data from each team and to extract the main lessons learned.	13	57	5
TE6. A process has been established to distribute the lessons learned to the relevant stakeholders and team members.	12	52	4
TE7. Team members are aware of their roles and responsibilities during software development and improvement.	8	35	5
Team Size (TS)			
TS1. Project team size has been assessed the suitability of the project.	8	78	4
Training Support (TR)			
TR1. Appropriate training has been provided to team members for developing the skills and knowledge needed to perform the project.	10	43	4
TR2. Sufficient resources and additional time to participate in training will be provided to team members.	3	13	4
TR3. Training program activities are reviewed on a periodic basis.	-	-	4
TR4. All future group or individual trainings of software development are planned.	-	-	4
User (Staff) Involvement (UI)			
UI1. The software development effort has been staffed by people who indicated interest and commitment in the effort.	9	39	4
UI2. A project has been established with a good user relationship.	12	52	5

List of Agile Practices Occurrence in	Literature (n=23) Frequency	%	Questionnaire (n=7) Median
UI3. A project has been established with user commitments, collaborations, and participation.	17	74	5
UI4. Users directly involving the project have had full authority.	4	17	4
UI5. Work has been done to facilitate team members during software development.	12	52	4
UI6. Work has been done to allocate the time necessary to make user participation successful.	9	39	4
List of Agile Practices Occurrence in	**Literature (n=8) Frequency**	**%**	**Questionnaire (n=7) Median**
Data Quality (DQ)			
DQ1. Plans or strategies to address data quality problems have been performed.	7	88	4
DQ 2. Common data standards or guidelines have been conducted.	8	100	4
DQ 3. Software development teams have their own working environments.	1	13	4
DQ 4. Basic skills have been trained to people relevant to data quality.	6	75	5
DQ 5. Data governance to ensure the quality, availability, integrity, security, and usability has been performed.	6	75	4
DQ 6. Database regression testing has been performed.	1	13	4
DQ 7. Many types of database testing (e.g., database input, database output, stored procedures, column constraints, default column values) have been performed.	3	38	4
DQ 8. The data aspects of software have been modeled iteratively.	4	50	4

Answering the RQ4-2, the results of the 31-literature survey show that 64 agile practices (all agile practices presented in Table 4-3 except TC4, TR3, and TR4 due to no data found in the literature) were globally recognized as influential in achieving successful software development. Considering the number of occurrences; however, only 13 agile practices were occurred at a rate of 75% or higher. Fourteen agile practices occurred at rates between 50% and 74%, whilst 37 agile practices occurred at a rate of less than 50%. Answering the RQ4-3, the results of the questionnaire show that 18 out of the 67 agile practices in Table 4-3 were recognized locally as the most important for successful software development and as little less significant for the rest (49 agile practices). The results noticeably reveal that all of these agile practices play a vital role in software development.

Figure 4-2 shows two domains of agile practices categorized by the median values on the questionnaire-style information collection responses. In the domain A, 18 agile practices occurred at the median value of 5 or recognized as the most important in the local realm. Compared to the results from the literature survey, it shows that only 4 agile practices (i.e., SD5, TC1, TE2, and DQ4 occurred at a rate of 75% or higher) were emphasized as the most important in both global and local realms. Albeit the other 14 agile practices were recognized as the most important in the local realm, 7 of them (i.e., MC2, OE1, OE5, TC8, TE5, UI2, and UI3 occurred at a rate between 50%-74%) were slightly less significantly emphasized, whilst the other 7 (i.e., SD7, OE4, PM4, PM6, TC6, TC7, and TE7 occurred at a rate of less than 50%) were insignificantly emphasized in the global realm.

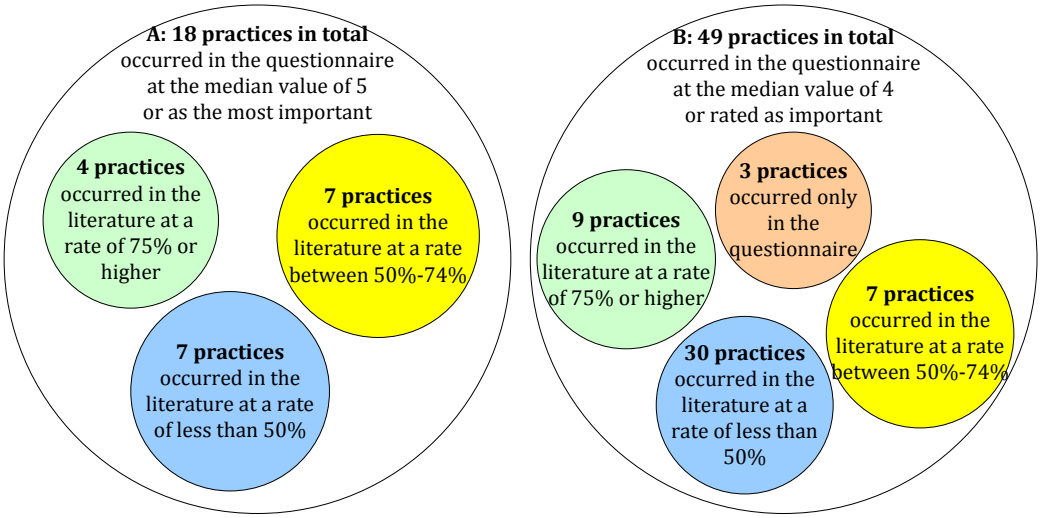


Figure 4-2. Comparison between the results of a literature survey and a questionnaire-style information collection

In the domain B, 49 agile practices occurred at the median value of 4 or recognized as important in the local realm. Comparing to the results of the literature survey, it shows that 9 of them (i.e., SD6, TC2, TC3, TC4, TE1, TS1, DQ1, DQ2, and DQ5 occurred at the rate of 75% or higher) are significantly emphasized in the global realm. The 7 of them (i.e., SD4, OE6, PM5, PT1, TE6, UI5, and DQ8 occurred at the rate between 50%-74%) were slightly less significantly emphasized. Surprisingly, the majority of this domain (i.e., the 30 of them: SD1, SD2, SD3, MT1, MC1, MC3, MC4, OE2, OE3, OE7, PM1, PM2, PM3, PM7, PT2, RE1, RE2, RE3, RE4, RE5, TE3, TE4, TR1, TR2, UI1, UI4, UI6, DQ3, DQ6, and DQ7 occurred at a rate at the rate of less than 50%) were insignificantly emphasized in the global realm. The other 3 (i.e., TC5, TR3, and TR4) were not emphasized in the global realm at all.

In the open-ended part of the questionnaire, some respondents suggest their additional agile practices, e.g., employing user stories, daily meetings, sprint review meetings, and sprint retrospective meetings. Those agile practices can be viewed as common Scrum practices. Therefore, we do not consider them as critical practices. However, an interesting point we found is that the respondents were aware of the importance of knowledge transfer during software development. This raises us more inspiration to develop a knowledge transfer framework (see Chapters 6 and 7 for comprehensive details). In summary, all of 67 agile practices in Table 4-3 were recognized as critical for successful software development in the local realm but only 16 agile practices were significantly emphasized in the global realm. However, our main focus is on the local realm. Therefore, all of 67 agile practices are directly used for designing our SDM model which is described in the next sub-section.

4.2.1.3 Structure of the Software Development Maturity Model

This sub-section answers the RQ4-4 "How should a software development maturity model (as one of two core components of a software process maintenance framework) be constructed?" A software development maturity model is created with a threefold objective: to appraise an organization's current software process through the identified CSFs, to get the current maturity level rating from the model, and to identify what processes demand immediate and sustainable improvement in agile software development. The CMMI staged

representation is an approach using predefined sets of process areas to define an SPI path and also providing a proven sequence of SPI [39]. CSF is "the limited number of areas in which satisfactory results will ensure successful competitive performance for the individual, department, or organization" as defined by Bullen and Rockhart [214]. It is a highly effective approach to define significant information needs. Based on the CMMI and CSFs approaches, Niazi et al. [114, 115] propose an SPI implementation maturity model providing a very practical structure with which to assess and improve SPI implementation processes. The model has empirically been proven to have potential to assist practitioners in assessing and improving SPI implementation processes and maturity in practice. Moreover, there is a high possibility to adapt the model to fit the SDM model's objectives. Therefore, the SPI implementation maturity model, CMMI, and CSFs are adapted for developing the SDM model. In the model, there are three dimensions (i.e., maturity stage, CSFs, and assessment) as illustrated in Figure 4-3.

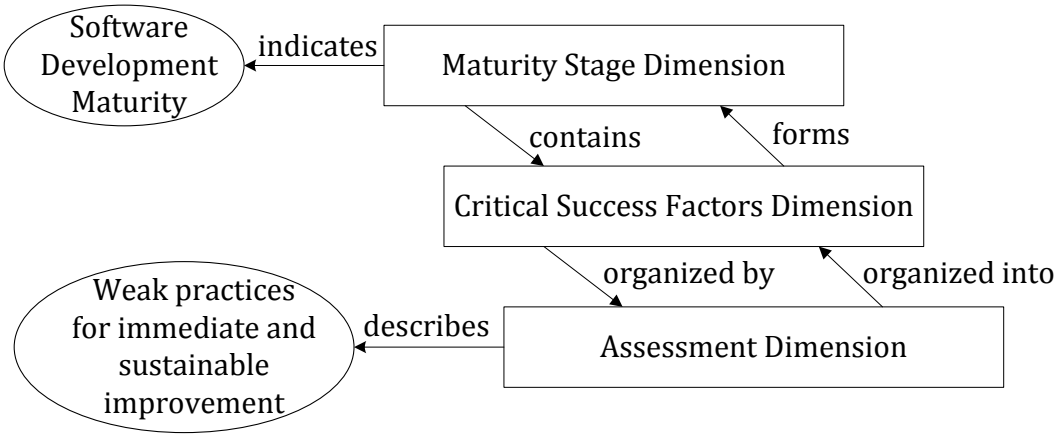


Figure 4-3. An SDM model structure (adapted from Niazi et al. [114])

In the maturity stage dimension, there at this stage are four maturity levels presented in Table 4-4. The maturity levels 1 through 4 have been adopted from four CMMI maturity levels: "Initial", "Managed", "Defined", and "Optimizing", respectively. The main reason not to replicate the CMMI maturity level-4 "Quantitatively Managed" is that the two key CMMI practices of establishment and maintenance of quantitative objectives for the process, and stabilization of the performance of one or more sub-processes to determine its ability to achieve are not compatible with agile practices [158]. There is no CSF cited in the literature that directly relates to this level [114].

At the starting point of this study, the goal was to identify CSFs of software development [183], which are directly used in the CSFs dimension. Based on the perception of CMMI process area division amongst different CMMI maturity levels [39] and the prevailing opinion in the SPI literature, we have categorized the identified CSFs into three categories (i.e., foundation, standardization, and support).

At the maturity level-1 "Initial", there is no category since this level does not have to be achieved due to its chaotic characteristic, similarly to CMMI.

At the maturity level-2 "Managed", basic project management processes, necessary process discipline, and commitments amongst key stakeholders are established. Hence, the foundation category containing CSFs that are the foundation for all subsequent levels can be

linked to this level. These CSFs include management commitment, project management process, project type, training support, and user involvement.

At the maturity level-3 "Defined", the project management and software development processes are standardized and integrated into a standard software process. Accordingly, the standardization category containing CSFs that support the design of systematic structures can be linked to this level. These CSFs include agile software development process, appropriate methods, techniques and tools, data quality, organizational environment, team capability, team environment, and team size.

Table 4-4. Four CMMI-based maturity levels

Maturity Level	Description
Level 1-Initial	This is the level where processes are usually chaotic and few processes are defined. The organization usually does not provide a stable environment. Its software development success thus depends on the competence and heroics of the people and not on the use of proven processes.
Level 2-Managed	At this level, processes are characterized for projects. The projects have ensured that requirements are managed and that processes are planned, performed and controlled. The work products satisfy their specified requirements, standards and objectives. The processes in this level are the foundation for all subsequent levels.
Level 3-Defined	At the level, processes are documented, standardized, and integrated into a standard software development process.
Level 4-Optimizing	This is the level for establishing structures for continuous software process improvement. At this level, processes depend on the participation of an empowered workforce aligned with the business values and objectives of the organization. The ability to rapidly respond to changes is enhanced by finding ways to accelerate and share learning. The software process performance is then continually improved.

Moreover, continuous SPI must be enabled. Therefore, the support category, which contains the reviews factor to support continuous SPI activities, can be linked to the maturity level-4 "Optimizing". Table 4-5 summarizes three categories and their belonging CSFs.

Table 4-5. Three CSF categories

Maturity Level	Category	CSF
Level 1-Initial	-	-
Level 2-Managed	Foundation	Management commitment, project management process, project type, training support, and user involvement
Level 3-Defined	Standardization	Agile software development process, appropriate methods, techniques and tools, data quality, organizational environment, team capability, team environment, and team size
Level 4-Optimizing	Support	Reviews

In order to guide how to assess and implement the identified CSFs, a list of agile practices has been designed under each CSF as presented in Table 4-4. We have high confidence in the CSF categorization and the design of agile practice lists. This is because there is a similarity between CSFs identified in the SPI literature and CSFs identified in this

study and great compatibility between CMMI SPI objectives and our identified CSFs at each maturity level.

Last, in the assessment dimension, an assessment instrument successfully developed and tested at Motorola [215] has been adapted to measure software development maturity. As illustrated in Table 4-6, this instrument has three evaluation dimensions:

- Approach, key criteria here are the organization commitment to and management support for the practice, and the organization's ability to implement the practice;
- Deployment, key criteria here are the breadth and consistency of practice implementation across project area;
- Results, key criteria here are the breadth and consistency of positive results across project areas.

In this instrument, "Commitment" means commitment to perform. "Ability" means ability to perform. "Activities" means activities performed. "Monitoring" means monitoring implementation. "Verification" means verifying implementation. It is important that practitioners must ensure that they use these terms in the same direction as described when determining a practice's score. This instrument can be adapted at many levels, e.g., organization, division, and project levels. When applied at the division level, the guideline "parts of the organization" should translate to "projects" or "project areas". When applied at the project level, the guideline "parts of the organization" should translate to "sub-projects" or "sub-systems".

Table 4-6. An assessment instrument [215]

Score	Key Activity Evaluation Dimensions		
	Approach	**Deployment**	**Results**
Poor (0)	• No management recognition of need • No organizational ability • No organizational commitment • Practice not evident	• No part of the organization uses the practice • No part of the organization shows interest	• Ineffective
Weak (2)	• Management begins to recognize • Support items for the practice be created • A few parts of organization to implement the practice	• Fragmented use • Inconsistent use • Deployed in some parts of the organization • Limited to monitoring/ verification of use	• Spotty results • Inconsistent results • Some evidence of effectiveness for some parts of the organization
Fair (4)	• Wide but not complete commitment by management • Road map for practice implementation defined • Several supporting items for the practice in place	• Less fragmented use • Some consistency in use • Deployed in some major parts of the organization • Monitoring/verification of use for several parts of the organization	• Consistent and positive results for several parts of the organization • Inconsistent results for other parts of the organization
Marginally qualified (6)	• Some management commitment; some management becomes	• Deployed in some parts of the organization • Mostly consistent use	• Positive measurable results in most parts of the organization

Score	Key Activity Evaluation Dimensions		
	Approach	**Deployment**	**Results**
	proactive • Practice implementation well under way across parts of the organization • Supporting items in place	across many parts of the organization • Monitoring/verification of use for many parts of the organization	• Consistently positive results over time across many parts of the organization
Qualified (8)	• Total management commitment • Majority of management is proactive • Practice established as an integral part of the process • Supporting items encourage and facilitate the use of practice	• Deployed in almost all parts of the organization • Consistent use across almost all parts of the organization • Monitoring/verification of use for almost all parts of the organization	• Positive measurable results in almost all parts of the organization • Consistently positive results over time across almost all parts of the organization
Outstanding (10)	• Management provides zealous leadership and commitment • Organizational excellence in the practice recognized even outside the company	• Pervasive and consistent deployed across all parts of the organization • Consistent use over time across all parts of the organization • Monitoring/verification for all parts of the organization	• Requirements exceeded • Consistently world-class results • Counsel sought by others

To calculate, each practice is weighted by three-dimensional scores in integer between 0 and 10. The three-dimensional scores of each practice are added, divided by 3, and rounded up. All obtained practice scores are then rolled over into an average score for each CSF. Any CSF with an average score falling below the threshold is deemed a weakness. The threshold is initially set to 7 as guided by Motorola [215]. However, it can be reset to better fit an organization's current situation. To achieve a certain maturity level, all CSFs belonging to that maturity level should have an average score of the threshold or higher. Table 4-7 shows an example of a CSF evaluation.

Table 4-7. A CSF evaluation (average score = 6+8+7+7/no. of practices = 28/4 = 7)

Management Commitment	Scores (0-10)			Avg. Score
	Approach	**Deployment**	**Results**	
MC1. Management provides strong commitment and presence	6	6	6	6
MC2. Management supports the software development	8	7	8	8
MC3. Management is willing to participate in assessment and development activities	7	7	6	7
MC4. Management is committed to provide training and resources	8	7	6	7
Average score for management commitment				7

The obtained maturity levels and weak practices demanding immediate and sustainable improvement are an output of the model. It is also be used as an input for an integrated PMBOK-Scrum model for planning integrated project management and software development processes. The integrated PMBOK-Scrum model is described in the next section.

4.2.2 The Integrated PMBOK-Scrum Model

Two of the most adopted methods are PMBOK for project management processes and Scrum for agile software development processes, serving as the basis of this study. In this section, the proposed base meta-models of PMBOK and Scrum are introduced. An integrated PMBOK-Scrum model is then presented.

4.2.2.1 A Project Management Body of Knowledge Meta-model

According to the PMBOK guide [38] and its definition presented in Section 2.4, Chapter 2, a PMBOK meta-model has been constructed. Figure 4-4 is a graphical representation of the PMBOK meta-model, using Unified Modeling Language (UML) notation. In the meta-model, an *Organization* is an official group of people making any contribution to achieving its organizational goals. The *Organization* may or may not have *Programs*. Each *Program* contains its related *Project*. Those related *Projects* are managed in a coordinated way to obtain benefit and control not available from managing them independently. A *Project* is a temporary endeavor undertaken to produce a unique (software) product, service, or result. Each *Program* is directed by one or more *Stakeholders*; while each *Project* is managed by a *Stakeholder*. This means that not every *Stakeholder* has to direct a program or manage a project. *Stakeholders* and *Physical Resources* are *Resources*. In other words, both *Stakeholders* and *Physical Resources* inherit from a *Resource.* A *Stakeholder* is a human resource or a person (e.g., sponsor, director, manager, developer, and user) who is directly involved in the project or whose interests may be positively or negatively affected by execution or completion of the project. A *Stakeholder* has *Team Members* and *Third Party Members* inheriting from it. A *Team Member* typically is a *Stakeholder* from inside the organization; whilst a *Third Party Member* is a *Stakeholder* from outside the organization. On the other hand, a *Physical Resource* has *Material*, *Equipment*, and *Facilities* inheriting from it. They are utilized to support work in the project.

Each *Project* has one or more *Phases*. Those *Phases* are generally sequential and sometimes overlapping. The number of *Phases* depends on the size, complexity, and potential impact of the *Project*. Each *Phase* has one or more locally related *Activities.* The *Activities* may or may not have *Dependencies* between them that can affect the use of *Stakeholders* or *Physical Resources*. There are four possible types of *Activity Dependencies* or logical relationships which are Finish-to-Finish where completion of the successor activity cannot finish until the completion of the predecessor activity; Finish-to-Start where initiation of the successor activity depends on the completion of the predecessor activity; Start-to-Finish where completion of the successor schedule activity depends on the initiation of the predecessor schedule activity; Start-to-Start where initiation of the successor schedule activity depends on the initiation of the predecessor schedule activity.

Each *Activity* has at least one *Stakeholder* performing it, but may or may not have *Physical Resources* supporting it. However, not every *Stakeholder* and *Physical Resources*

has to perform or support an *Activity*. Consequently, a *Stakeholder* who performs any *Activity* will have his/her workload, represented as a *Stakeholder Workload*. Likewise, a *Physical Resource* used to support any *Activity* will have its workload, represented as a *Physical Resource Workload.* Besides, a *Stakeholder* typically has one *Role* describing the responsibilities, competencies and related information of *Stakeholders* taking that role. Nevertheless, each *Role* may or may not have *Stakeholders* and may or may not be responsible for *Deliverables.* However, each *Deliverable* must have one *Role* responsible for it. A *Deliverable* is a unique and verifiable work product, e.g., project charter, project plan, work result, and software product. Moreover, each *Role* may or may not perform *Activities*; however, each *Activity* must have at least one *Role* performing it. Each *Activity* may or may not have its decomposed Tasks and may or may not have *Techniques* or *Tools* supporting it. Both *Techniques* and *Tools* inherit from *Guidance.* Each A*ctivity* may or may not be performed (i) to deliver a *Deliverable* as an output or (ii) using Deliverables as its inputs. In other words, a *Deliverable* must be produced as an output from at least one *Activity*, and may or may not be input to a successor activity.

One or more *Activities* are composed of a *Management Processes.* Each *Management Process* is organized into one *Process Group* and one *Knowledge Area*; while each *Process Group* and each *Knowledge Area* contain one or more *Management Processes.* According to PMBOK, there are five basic *Process Groups* and nine *Knowledge Areas*. The five *Process Groups* consist of (1) Initiating, which processes performed to define a new project or phase by obtaining authorization to start the project or phase; (2) Planning, which processes needed to establish the scope of the project, refine the objectives, and define what actions needed to attain the objectives; (3) Executing, which processes performed to complete the work defined in the project plan; (4) Monitoring and Controlling, which processes needed to track, review, control the progress and performance of the project; and (5) Closing, which processes performed to finalize all activities across all process groups to formally close the project or phase. The nine *Knowledge Areas* consist of (1) Integration Management, which includes the processes and activities needed to identify, define, combine, unify, and coordinate the various processes and activities within the process groups; (2) Scope Management, which ensures that all the required work, and only the required work, is planned, defined, documented, and delivered to the user's satisfaction; (3) Time Management, which includes the processes needed to manage timely completion of the project; (4) Project Cost Management, which includes the processes involved cost estimation and expense monitoring, and intended to ensure that the project is delivered within its approved budget; (5) Quality Management, which encompasses quality definition, assurance, and control; (6) Human Resource Management, which includes the processes that organize, manage, and lead the project team; (7) Communication Management, which includes the processes for information dissemination and collection; (8) Risk Management, which includes the processes of risk identification, quantification, avoidance, and mitigation; and (9) Procurement Management, which includes the processes necessary to purchase or acquire products or services needed from outside the project team.

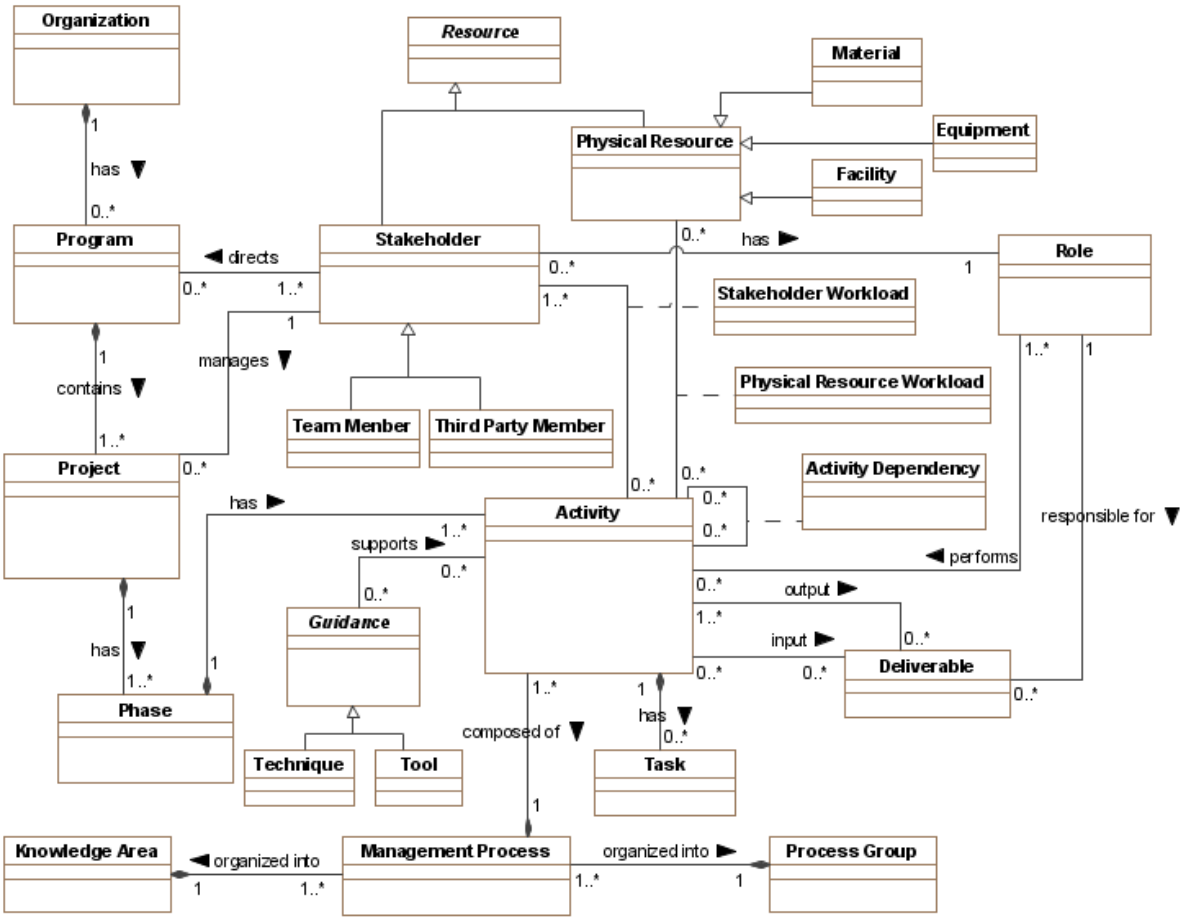


Figure 4-4. A PMBOK meta-model

4.2.2.2 A Scrum Meta-model

According to the Scrum definition presented in Section 2.4 in Chapter 2, a Scrum meta-model has been constructed. Figure 4-5 is a graphical representation of the Scrum meta-model, using UML notation. In the meta-model, a software *Project* has one or more *Phases* (called iterations or sprints). Each *Phase* has one or more *Activities*. Related *Activities* are composed of a *Process*. Likewise, a *Stage* has one or more related *Processes.* In Scrum, there are three *Stages*, i.e., pre-game, game, or post-game. Pre-game contains two *Processes* (i.e., planning and architecture development). The planning includes the definition of a new release based on currently known product backlog, along with an estimate of its schedule and cost. If the software product under development is new, planning includes both conceptualization and analysis, but only limited analysis for an existing software product. The architecture development includes system and/or software architecture development and high level design. Game includes the *Process* of sprint execution. This stage consists of a collection of development sprints to produce new release functionalities, with constant respect to the variables of time, requirements, quality, cost, and competition. Post-game is the closure of the project, which includes preparing the releases, producing the final documentation, executing the site acceptance testing and the final product release.

The *Activities* may or may not have *Dependencies* between them. Albeit Scrum does not explicitly address activity dependency, in general there are four possible types of *Activity*

Dependencies or logical relationships which are Finish-to-Finish where completion of the successor activity cannot finish until the completion of the predecessor activity; Finish-to-Start where initiation of the successor activity depends on the completion of the predecessor activity; Start-to-Finish where completion of the successor schedule activity depends on the initiation of the predecessor schedule activity; Start-to-Start where initiation of the successor schedule activity depends on the initiation of the predecessor schedule activity.

Each *Activity* is performed by one or more *Roles* describing the responsibilities, competencies, and related information of *Stakeholders* who perform it. However, not every *Role* has to perform *Activities*. There are three primary *Role*s consisting of product owner, Scrum master, and team. Product owner is a person responsible for creating and prioritizing the features of the software product, deciding on release date and content, adjusting features and priority, and accepting or rejecting work results. Scrum master is a facilitative team leader working closely with the product owner and responsible for ensuring that the team is fully functional and productive, removing impediments, shielding the team from external interference, and making certain that the process is followed. Team typically consists of seven plus or minus two members. The team is committed to achieving a sprint goal and has the right to do whatever it takes to achieve the goal.

Each *Role* may or may not associate with *Stakeholders* and may or may not be responsible for any *Artefact.* However, each *Stakeholder* must have a *Role*. There are three primary *Artefact*s consisting of product backlog, sprint backlog, and Burndown chart which all are openly accessible and visible to the team. Product backlog is a list of all prioritized business and technical requirements that need to be developed and defects that need to be fixed. Sprint backlog is a list of all requirements in the current sprint that are broken down into tasks. Each task contains a short task description (e.g., owner, status, and effort). The Sprint Backlog is daily updated to obtain the latest effort of the work remaining to complete the task. A Burndown chart shows the hours remaining to complete sprint tasks A *Stakeholder* is a person (e.g., sponsor, product owner, Scrum master, team member, and user) who is directly involved in the project or whose interests may be positively or negatively affected by execution or completion of the project.

Each *Activity* has at least one *Stakeholder* performing it, but not every *Stakeholder* has to perform an *Activity*. Hence, a *Stakeholder* who performs any *Activity* will have his/her workload, represented as a *Stakeholder Workload*. Each *Activity* may or may not have its decomposed *Tasks*, and may or may not use *Tools* or *Techniques* to support it. Both *Tools* and *Techniques* inherit from *Guidance*. Similarly to *Stakeholder,* not every *Tool* and *Technique* has to support an *Activity.* Moreover, each *Activity* may or may not (i) produce or update *Artefact*s as outputs, or (ii) use *Artefact*s as its inputs. However, each *Artefact* must be produced from one *Activity*.

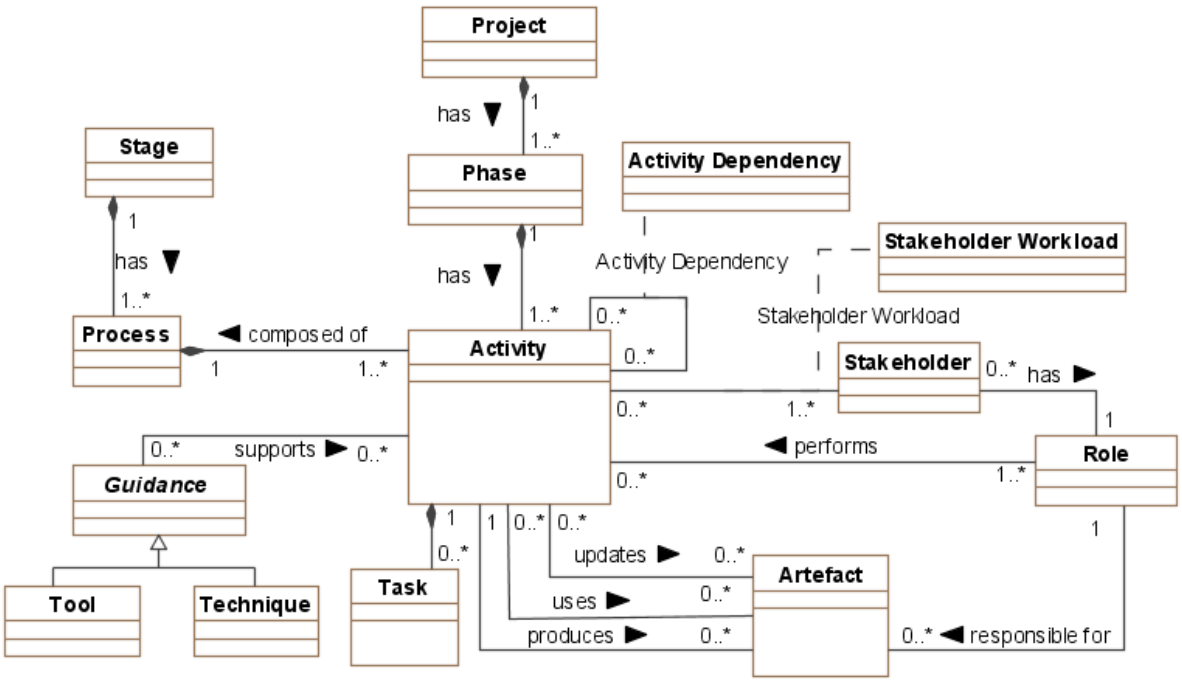


Figure 4-5. A Scrum meta-model

4.2.2.3 An Integrated PMBOK-Scrum Model

This sub-section answers the RQ4-5 "How should an integrated PMBOK-Scrum model (as one of two core components of a software process maintenance framework) be constructed?". The integrated PMBOK-Scrum model is composed by three layers: a managerial layer, a production layer, and an integration layer. The layers of managerial and production are derived from the distinction of concepts, meaning each distinct class on PMBOK and Scrum meta-models can be left in managerial and production layers, respectively. The integration layer is derived from an overlapping of concepts. This means two classes on each of the meta-models have the same concept that can be transformed into a single concept inside the integration layer. Moreover, a relationship of concepts is introduced for creating an association between related classes. Figure 4-6 illustrates an integrated PMBOK-Scrum model. The model is developed based on the same kind of approach on it is made for the integration of, e.g., PMBOK and RUP (Rational Unified Process) [8] and RUP-OPEN (Object-oriented Process, Environment and Notation) [216], and explained similarly to Callegari and Bastos [8]. We begin with the overlapping concepts and the relations to the two original models.

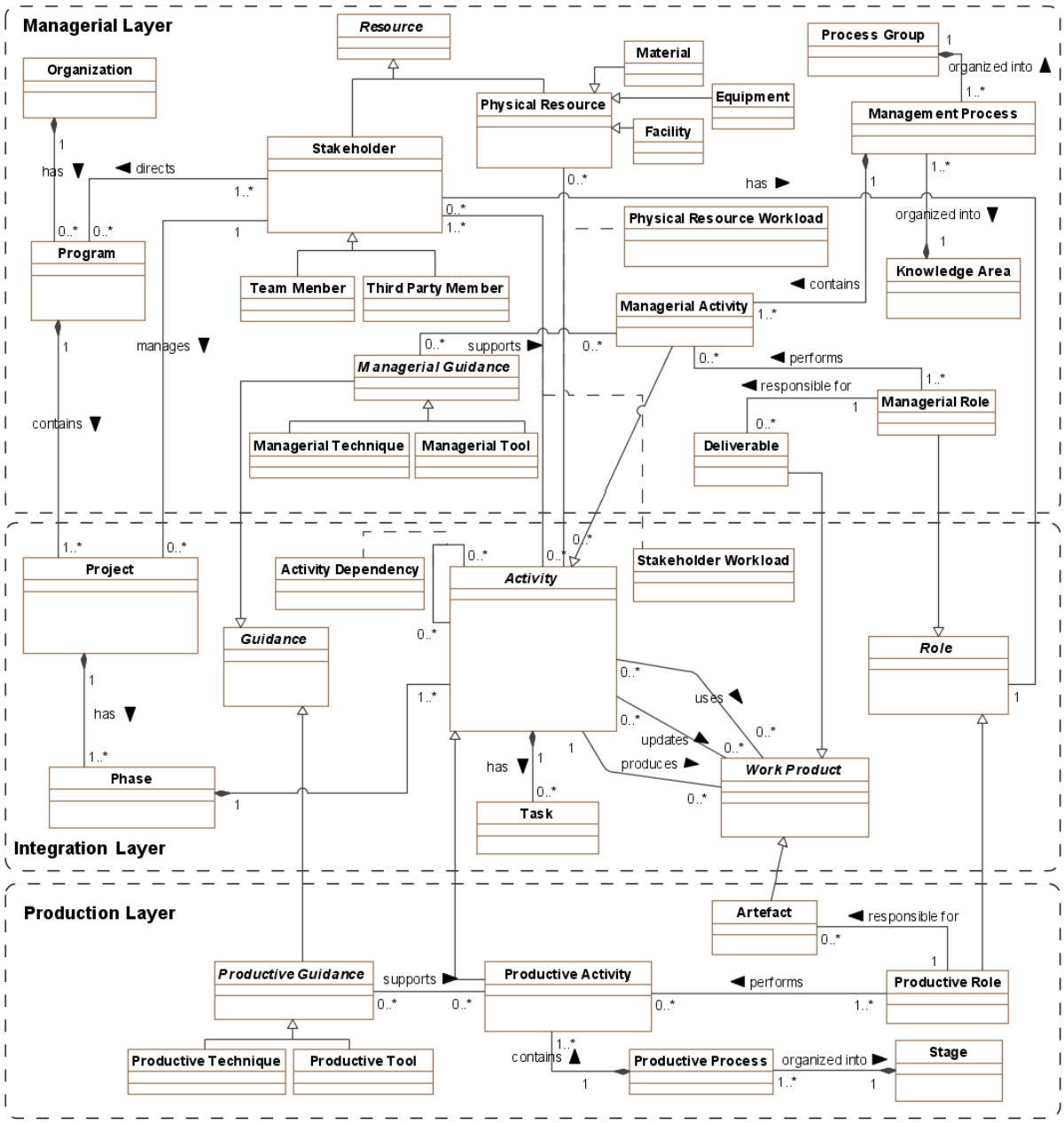


Figure 4-6. An integrated PMBOK-Scrum model

In the integration layer, both models have a *Project* concept which has one or more *Phases* or iterations. Each *Phase* has one or more *Activities*. A given *Activity* must be defined as *Managerial Activity* or *Productive Activity*. Each *Activity* may or may not have its decomposed *Tasks* and may or may not have *Dependencies* between *Activities*, which allow the definition of the order they can occur inside the *Project*. For instance, PMBOK's activities are broken down into tasks, called Work Breakdown Structure (WBS), and Scrum's backlog items are broken down into tasks for developing a sprint backlog. Each *Activity may or may not* be supported by *Tools or Techniques* in order to produce, use or update a *Work Product*. Both *Managerial Guidance* and *Productive Guidance* inherit from *Guidance.* With respect to types of guidance*, Managerial Tools* and *Managerial Techniques are derived from Managerial Guidance*; whilst *Productive Tools* and *Productive Techniques* are derived from

Productive Guidance. Likewise, a managerial work product (represented as a *Deliverable*) and a productive work product (represented as an *Artefact*) inherit from a *Work Product*. Each *Work Product* may or may not be used, or updated by an *Activity;* however, it must be produced by one *Activity*.

Any given *Activity* has one or more *Roles* performing it. Like *Activities*, a *Role* must be defined as a *Managerial Role* or a *Productive Role*. Each *Role* describes the responsibilities, competencies, and related information of *Stakeholders* who performs an *Activity*. In other words, the association between a *Role* and its *Activities* a *Stakeholder* must be present, including the stakeholder's workload which is an attribute in the *Stakeholder Workload* class in that relation. Similarly, for a *Physical Resource* used for any Activity, its workload which is an attribute in the *Physical Resource Workload* class must be present. Since in the Scrum meta-model the *Updates* association occurs between the *Activity* and the *Artefact*, while in the PMBOK meta-model the *Output* association meaning producing and updating an output occurs between the *Activity* and the *Deliverable*. Clearly specifying the associations, in the integrated model there are three associations (i.e., *Produces*, *Uses*, and *Updates*) connecting between the *Work Product* and the *Activity*.

Although Scrum has a project management perspective, in the integrated model its project management concepts are moved into the managerial layer. In the managerial layer, an *Organization* may or may not have *Programs*. Each *Program* contains it related *Projects* and is directed by one or more *Stakeholders*; while each *Project* is managed by a *Stakeholder*. *Stakeholders* and *Physical Resources* are derived from *Resource.* A *Stakeholder* has *Team Members* and *Third Party Members* inheriting from it. A *Team Member* typically is a *Stakeholder* from inside the organization; while a *Third Party Member* is a *Stakeholder* from outside the organization. On the other hand, a *Physical Resource* has *Material*, *Equipment*, and *Facilities* inheriting from it. They are utilized to support work in the project.

Related *Activities* are composed of a *Process*, depending upon activity type. This means a *Managerial Process* contains a set of related *Managerial Activities*; whilst in the productive layer, a *Productive Process* contains a set of related *Productive Activities*. For management aspects, each *Managerial Process* is organized into one *Process Group* and one *Knowledge Area*; while each *Process Group* and each *Knowledge Area* contains one or more *Managerial Process.* For productive aspects, each *Productive Activity* is organized into one *Stage*.

Increased software complexity, shortened development cycles, and higher quality expectations have placed a major responsibility to software development teams [217]. A possible way for overcoming this matter is a good software project model. Thus, the following set of constraints is provided to support practitioners who are responsible for planning a software project with a comprehensive set of project management and software development processes and to ensure the consistency of the integrated PMBOK-Scrum model.

1. A program must have a director; therefore, a stakeholder who is a program director must have a managerial role.
2. A project must have only one project manager; therefore, a stakeholder who is a project manager must have a managerial role.
3. An activity flow must not result in a cycle (for example, activity A is a prerequisite for activity B and activity B is also a prerequisite for activity A)
4. The same activity can only either produce, update, or use the same work product; they must be performed in different activities.

5. Each activity must be performed by at least one role and have only one stakeholder responsible for it; the stakeholder must also be compatible with that role.
6. A managerial activity cannot produce or update a productive work product, except only a managerial work product; however, this activity can use a productive work product.
7. A productive activity cannot produce or update a managerial work product, except only a productive work product; however, this activity can use a managerial work product.
8. An activity can update or use a work product only when it has already been created by a predecessor activity; otherwise it first needs to produce that work product.

The SDM model and the integrated PMBOK-Scrum model have then been incorporated into the software process maintenance framework. Consequently, the framework provides the "what" to improve with a software process assessment mechanism and the "how" to implement with and a comprehensive set of project management and software development processes that could lead to the improvement of software development performance in terms of efficiency and effectiveness. Using the framework may be too complex without the right tools. Therefore, a prototype tool supporting the use of the framework is produced and described in the next section.

4.3 A Prototype Tool Supporting the Software Process Maintenance Framework

Under the framework foundation, we designed a prototype tool called SPAD (Software Process Assessment and Development) to assist in the assessment, improvement and definition of project management and software development processes. The prototype is being developed as a Web-based application, using the Java language and a MySQL database. Figure 4-7 illustrates the SPAD functionality using a Use-case diagram.

Through a Use-case diagram, an end user (i.e., project sponsor, product owner, project manager, team leader, team member, project end user, and project stakeholder) is a person primarily or actively involved in the project effort and responsible for assessing and planning integrated project management and software development processes. A team member is a person having various possible roles (e.g., business analyst, system analyst, data analyst, database designer, developer, and tester), depending upon a software project's strategies. Except those persons, other project stakeholders (e.g., project consultant) can also be involved. For each project, there are two possible cases to use our prototype tool. First, a person as a software project representative (typically a project manager) having authority and responsibility can individually assess, define, and plan a project. Second, all persons are responsible for assessing and planning a project, but there is only one data set of the project logged into the prototype tool.

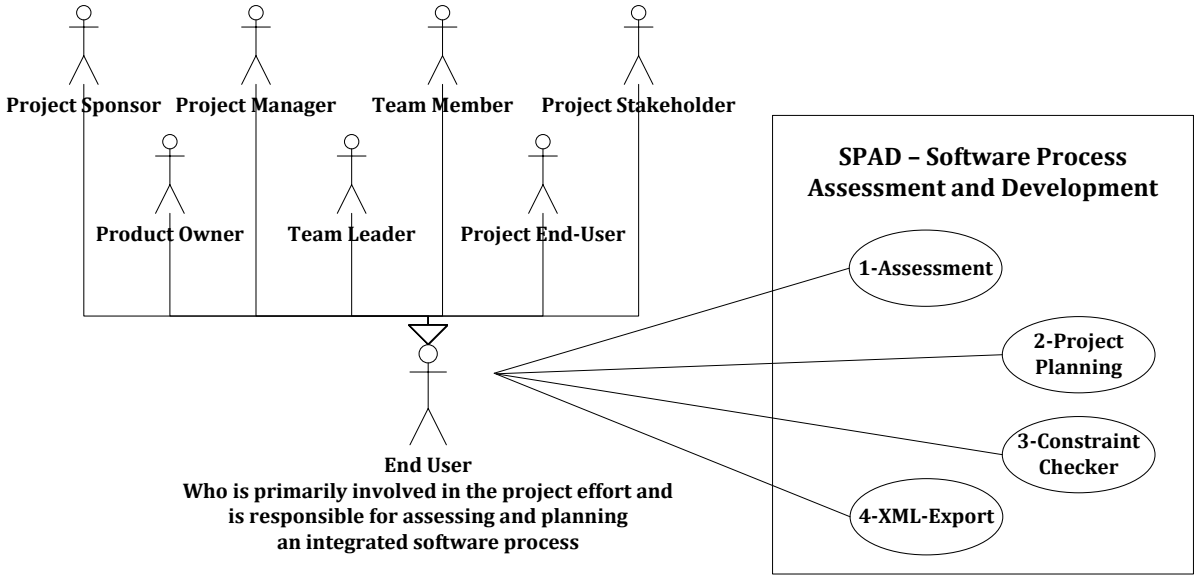


Figure 4-7. A high-level Use-case diagram showing the main SPAD functionality

Practitioners can gain considerable insight into the organization's current maturity by evaluating it in the project management and software development environments as a prerequisite to defining and planning the software process required. To do so, in the "Assessment", the user can first get the assessment details to understand the SDM concepts, as depicted in the sample screenshots in Figure 4-8 to Figure 4-11. The user then needs to set a threshold to support an assessment calculation, as depicted in the sample screenshot in Figure 4-12. After the setting, the user can then assess the identified CSFs through their lists of practices required by the SDM model, as depicted in the sample screenshot in Figure 4-13. SPAD then calculates with the calculation logic mentioned in Section 4.2.1.3 and then summarizes (i) the obtained maturity level and scores in scoring worksheets as depicted in the sample screenshot in Figure 4-14, (ii) weak practices in tables as depicted in the sample screenshot in Figure 4-15, and (iii) the overall status of the CSFs in bar charts as depicted in the sample screenshot in Figure 4-16. The obtained results should assist the user in understanding the organization's current situation, e.g., its strong practices and weak practices. For facilitating the user to plan project management and software development processes, the user can use the application feature to automatically feed the weak practices into the project definition and planning module as the required practices for improvement.

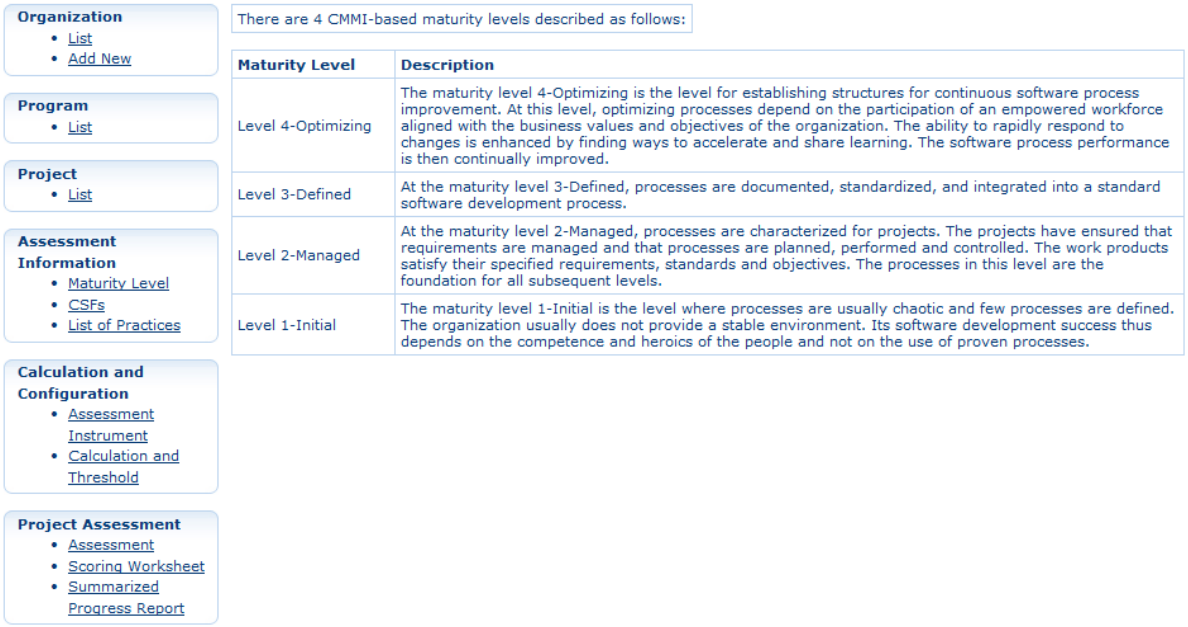


Figure 4-8. A sample screenshot of maturity level details

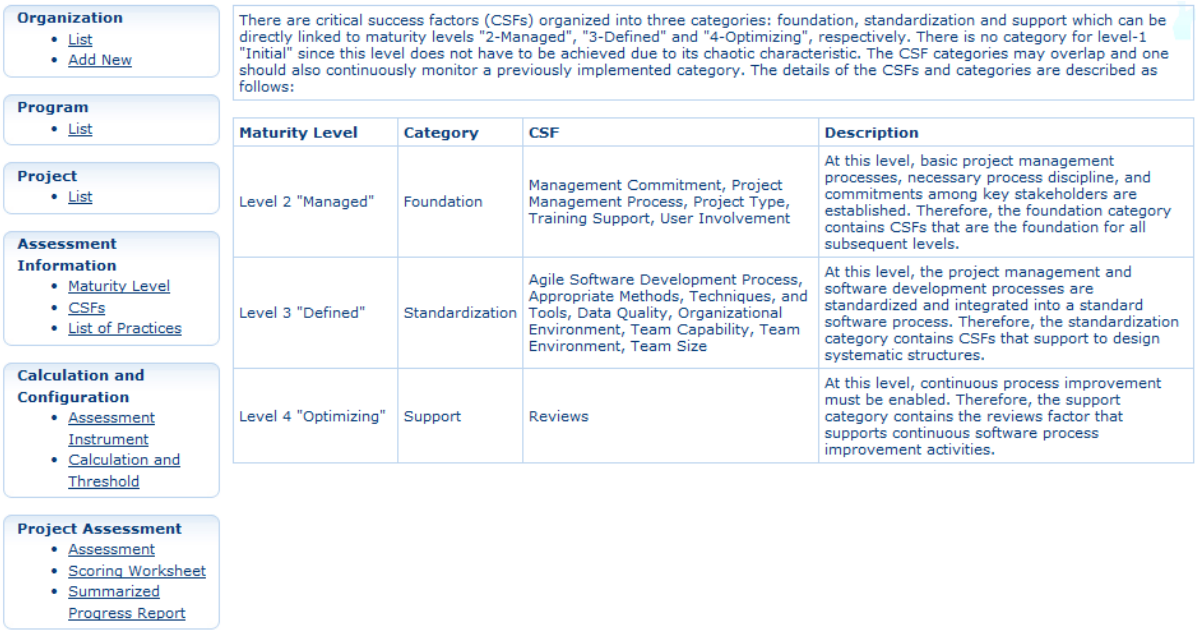


Figure 4-9. A sample screenshot of critical success factor details

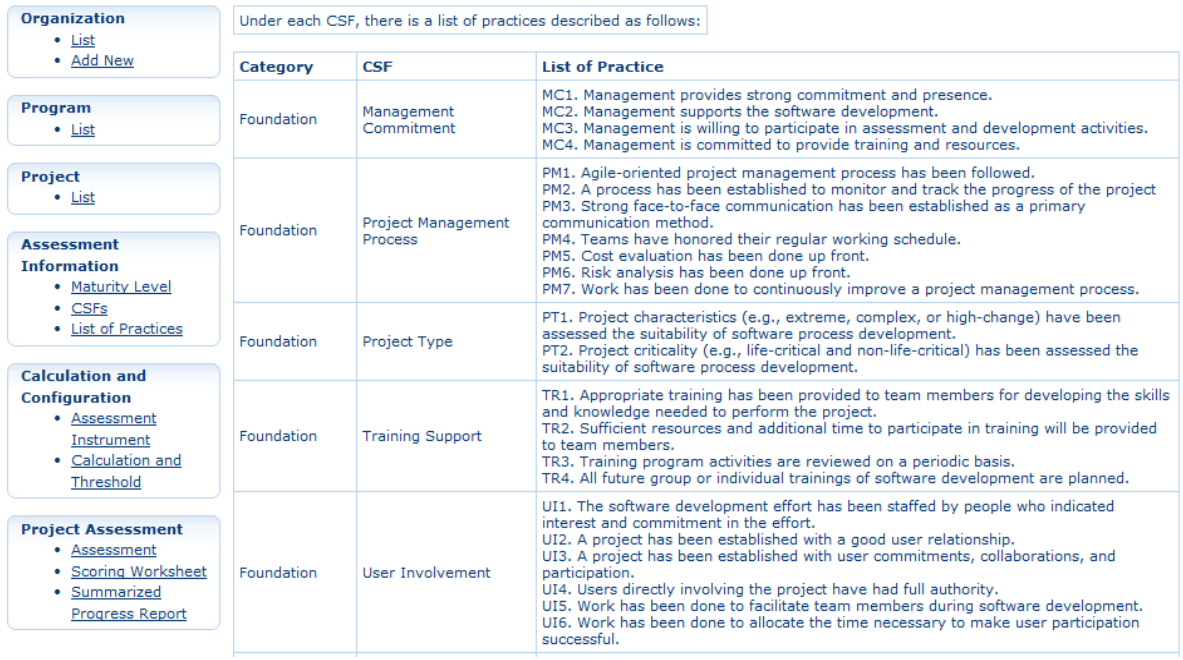


Figure 4-10. A sample screenshot of a list of practices

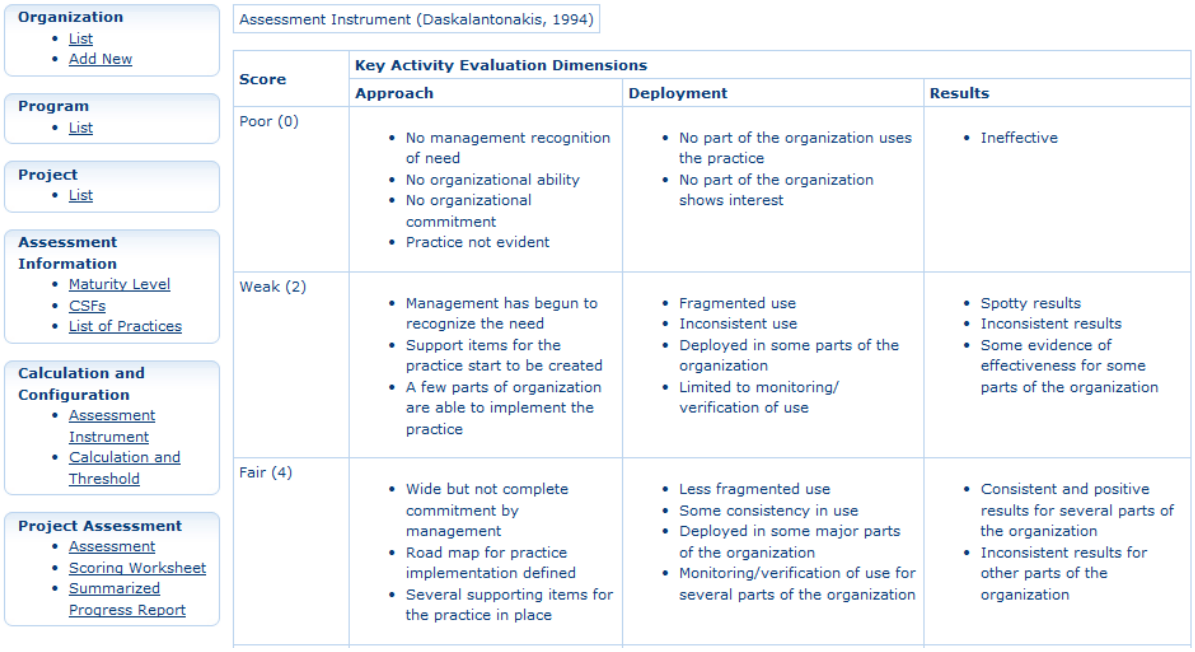


Figure 4-11. A sample screenshot of assessment instrument details

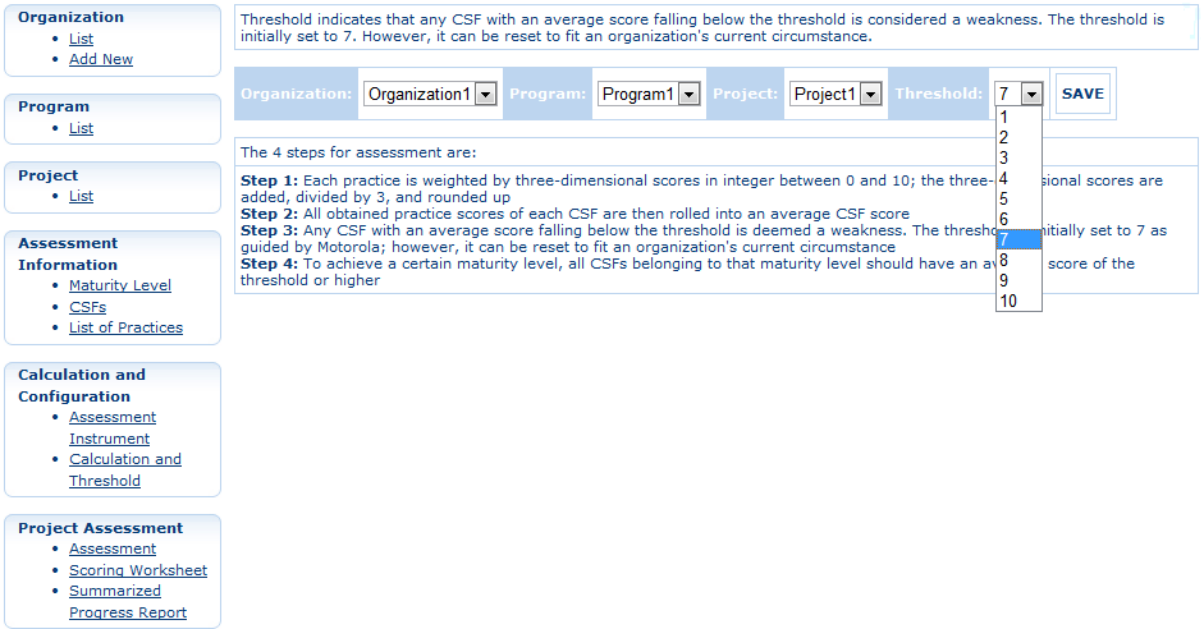
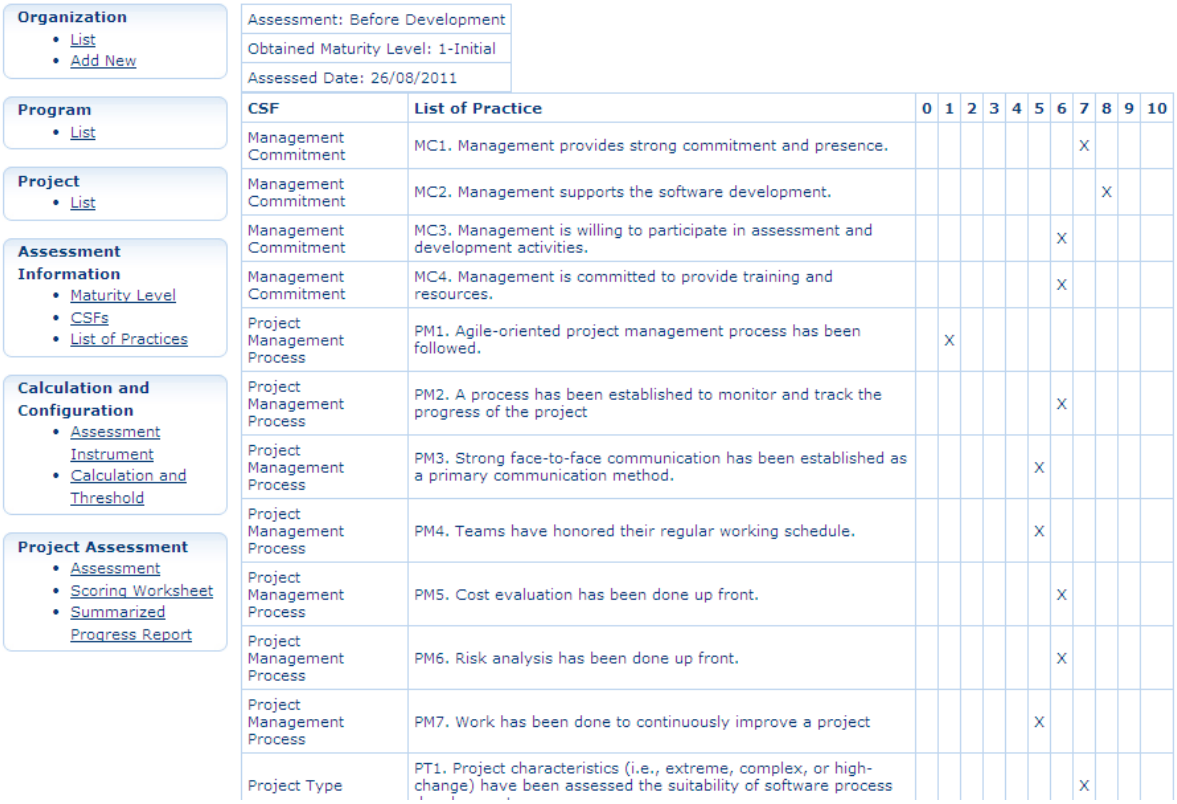


Figure 4-12. A sample screenshot of setting a threshold to support an assessment calculation

Figure 4-13. A sample screenshot of measuring implemented software processes



Organization
- List
- Add New

Program
- List

Project
- List

Assessment Information
- Maturity Level
- CSFs
- List of Practices

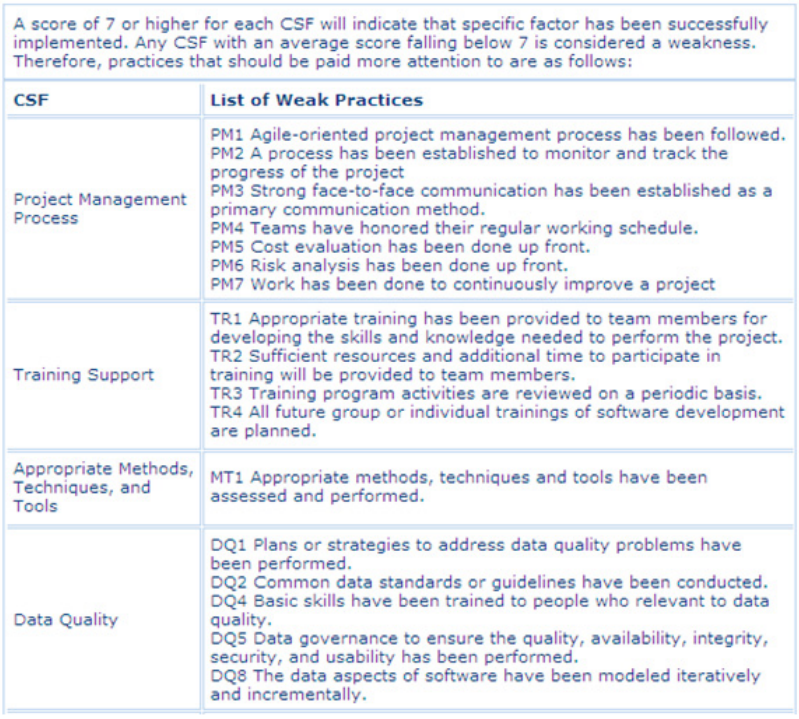
Calculation and Configuration
- Assessment Instrument
- Calculation and Threshold

Project Assessment
- Assessment
- Scoring Worksheet
- Summarized Progress Report

Assessment: Before Development
Obtained Maturity Level: 1-Initial
Assessed Date: 26/08/2011

CSF	List of Practice	0	1	2	3	4	5	6	7	8	9	10
Management Commitment	MC1. Management provides strong commitment and presence.								X			
Management Commitment	MC2. Management supports the software development.									X		
Management Commitment	MC3. Management is willing to participate in assessment and development activities.							X				
Management Commitment	MC4. Management is committed to provide training and resources.							X				
Project Management Process	PM1. Agile-oriented project management process has been followed.		X									
Project Management Process	PM2. A process has been established to monitor and track the progress of the project							X				
Project Management Process	PM3. Strong face-to-face communication has been established as a primary communication method.						X					
Project Management Process	PM4. Teams have honored their regular working schedule.						X					
Project Management Process	PM5. Cost evaluation has been done up front.							X				
Project Management Process	PM6. Risk analysis has been done up front.							X				
Project Management Process	PM7. Work has been done to continuously improve a project						X					
Project Type	PT1. Project characteristics (i.e., extreme, complex, or high-change) have been assessed the suitability of software process								X			

Figure 4-14. A sample screenshot of displaying the assessment results in a scoring worksheet



A score of 7 or higher for each CSF will indicate that specific factor has been successfully implemented. Any CSF with an average score falling below 7 is considered a weakness. Therefore, practices that should be paid more attention to are as follows:

CSF	List of Weak Practices
Project Management Process	PM1 Agile-oriented project management process has been followed. PM2 A process has been established to monitor and track the progress of the project PM3 Strong face-to-face communication has been established as a primary communication method. PM4 Teams have honored their regular working schedule. PM5 Cost evaluation has been done up front. PM6 Risk analysis has been done up front. PM7 Work has been done to continuously improve a project
Training Support	TR1 Appropriate training has been provided to team members for developing the skills and knowledge needed to perform the project. TR2 Sufficient resources and additional time to participate in training will be provided to team members. TR3 Training program activities are reviewed on a periodic basis. TR4 All future group or individual trainings of software development are planned.
Appropriate Methods, Techniques, and Tools	MT1 Appropriate methods, techniques and tools have been assessed and performed.
Data Quality	DQ1 Plans or strategies to address data quality problems have been performed. DQ2 Common data standards or guidelines have been conducted. DQ4 Basic skills have been trained to people who relevant to data quality. DQ5 Data governance to ensure the quality, availability, integrity, security, and usability has been performed. DQ8 The data aspects of software have been modeled iteratively and incrementally.

Figure 4-15. A sample screenshot of displaying the obtained weak processes in a table

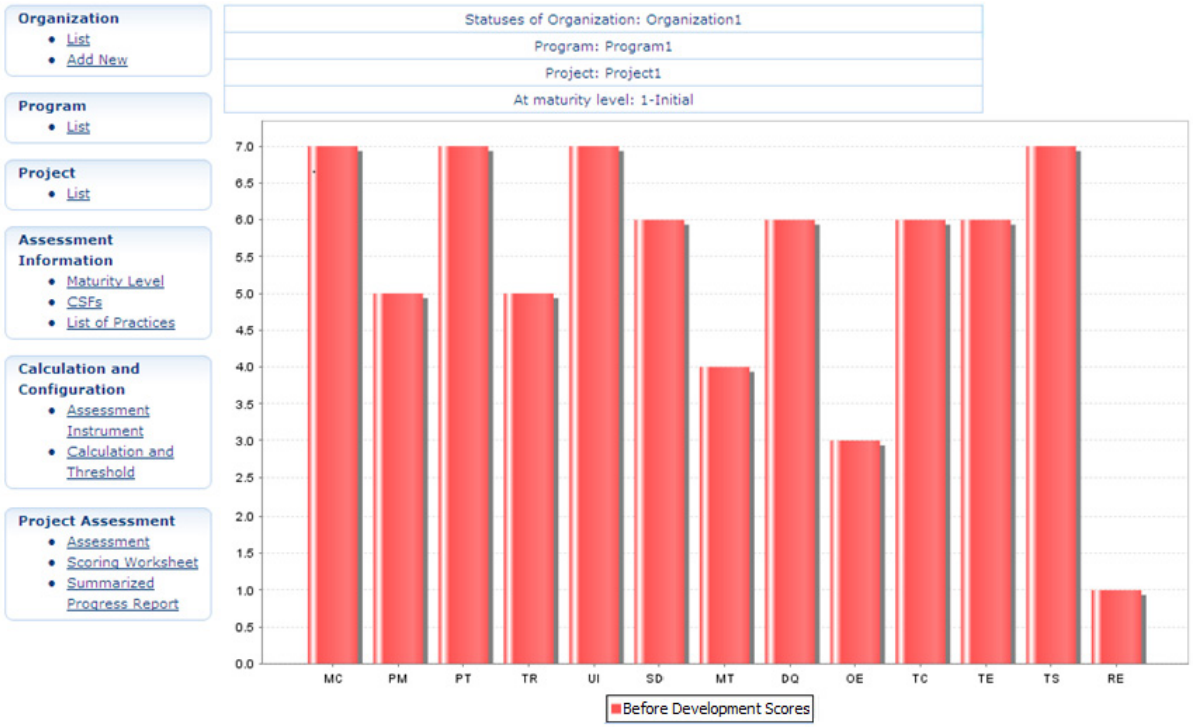


Figure 4-16. A sample screenshot of displaying the assessment results in a bar chart

Second, in the "Project Planning", the user can define the information required by the integrated PMBOK-Scrum model. This information contains the details of organizations, programs, projects, phases, stages, work products, roles, activities, guidance, stakeholders, physical resources, managerial knowledge areas, managerial process groups, managerial processes, working times, and work breakdown structure codes, as depicted in the sample screenshot in Figure 4-17. This information and the imported weak practices are used to plan the project. The module has been at this stage designed to assist the user in developing software project plans, assigning resources to tasks, and analyzing workloads. After planning, the user can preview the activity usage, the resource usage, and the software project plan, as depicted in the sample screenshot in Figure 4-18.

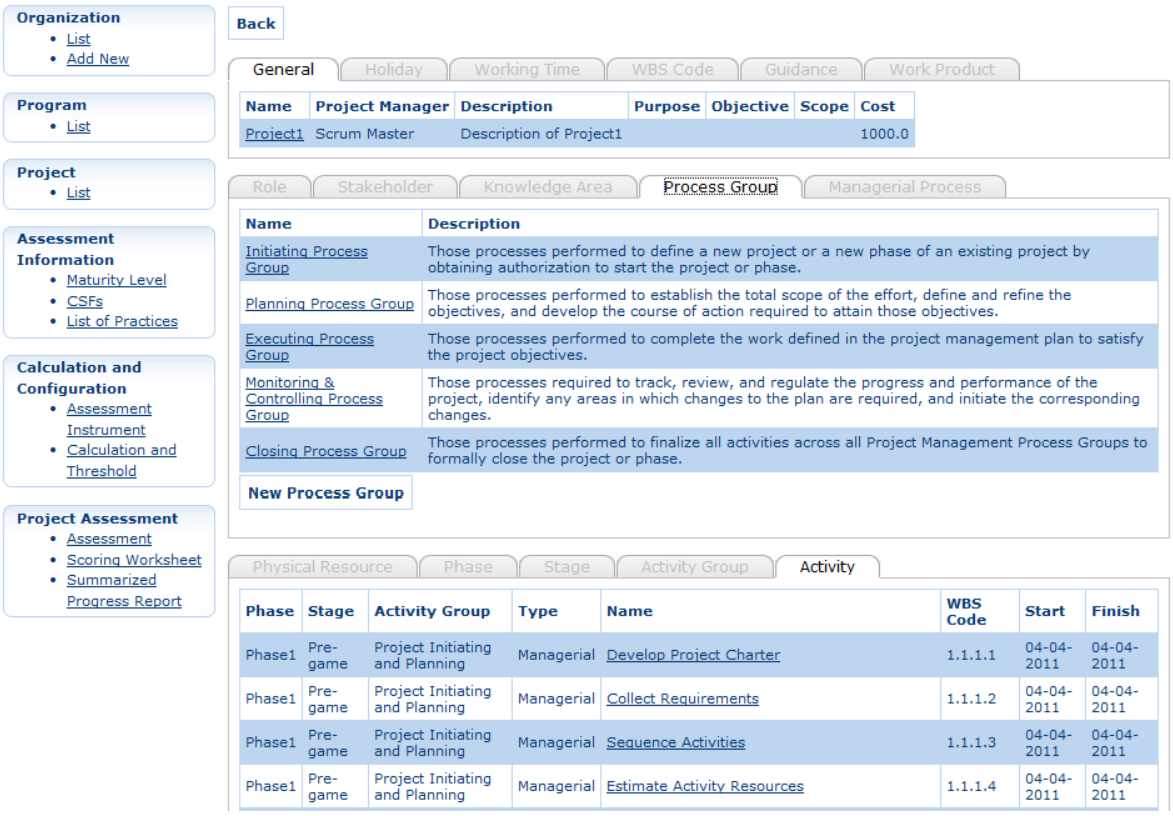


Figure 4-17. A sample screenshot of defining project information and planning

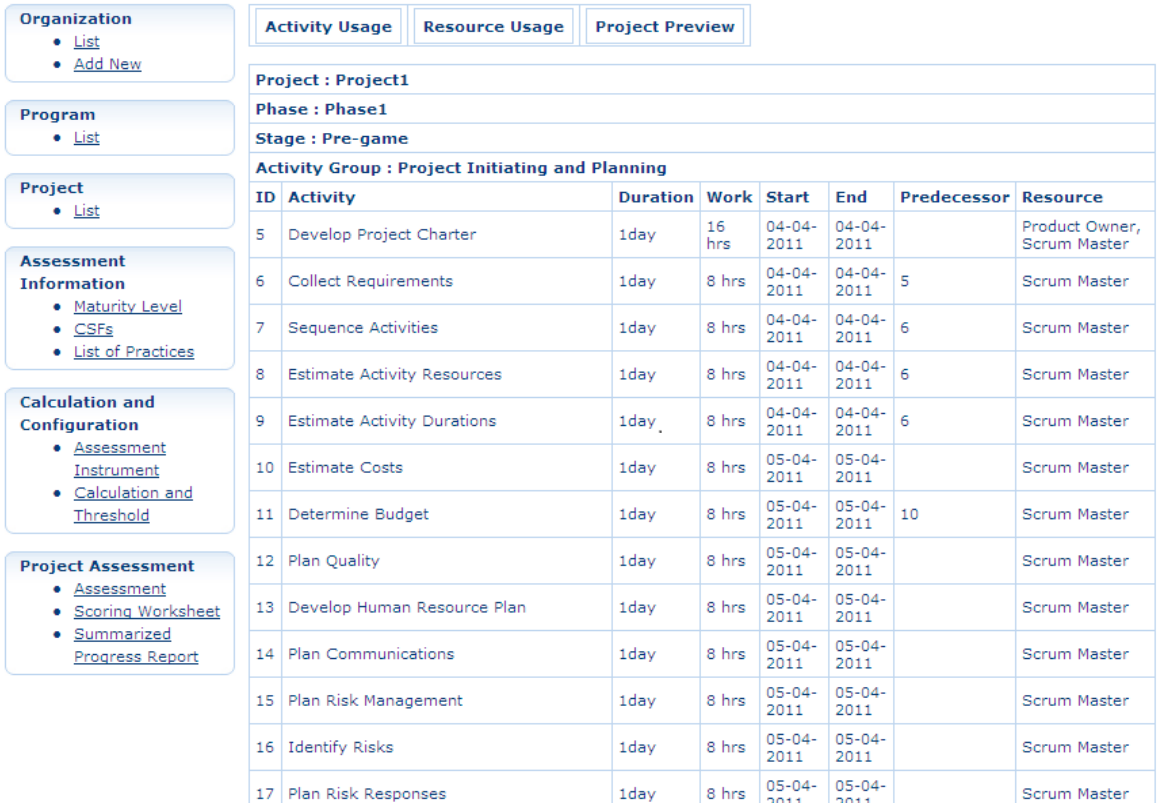


Figure 4-18. A sample screenshot of previewing a project plan

In the "Constraint Checker", the defined processes are then validated by the mentioned constraints proposed by the integrated PMBOK-Scrum model. This is to assure their appropriateness and consistency. The validation results will be shown for tracking an inappropriate process (if any). However, all of the constraints (except the two constraints of (i) a program must have a director; therefore, a stakeholder who is a program director must have a managerial role; and (ii) a project must have only one project manager; therefore, a stakeholder who is a project manager must have a managerial role) are also checked during planning the project management and software development processes in the "Project Planning" module. This is in order to help the user to plan the project properly. After correcting all inappropriate processes, the validated the "XML-Export" is then executed.

Fourth, in the "XML Export", the validated project is then prepared in a form of a standardized MS Project 2003 eXtensible Markup Language (XML) file format for export to the organization's project planning tools. The main reason for the need of other project planning tools is that SPAD at this stage provides the limited functionality. For instance, in case that there is a need of the application features to track project progress, manage project budgets, or visualize project schedules in Gantt charts, the user should export the project to their suitable tools having such required features. Figure 4-19 depicts a sample screenshot of the "Constraint Checker" and the "XML Export".

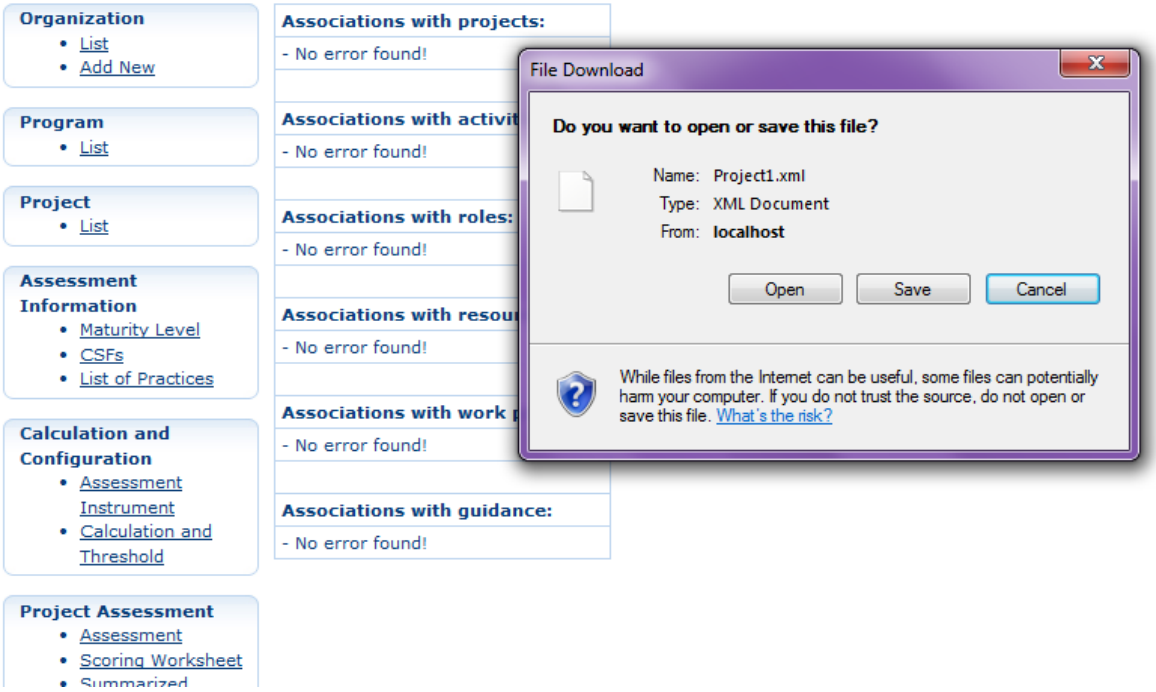


Figure 4-19. A sample screenshot of the "Constraint Checker" and the "XML-Export"

Moreover, the user can again perform the implemented process appraisal in the "Assessment" to compare the performance of the before- and after- software process development. As stated by Sommerville [218], the process improvement cycle involves three stages: process measurement, attributes of the current process are measured; process analysis, the current process is assessed and weaknesses are identified; and process change, changes to the process are introduced. After software development, these three stages should be completed. One of criteria that can be used for evaluating the performance of the process

improvement is the obtained higher scores of CSFs or maturity levels. This module should therefore assist the user in considering the performance of the process improvement. Figure 4-20 depicts a sample screenshot of the before- and after- implemented CSFs comparison.

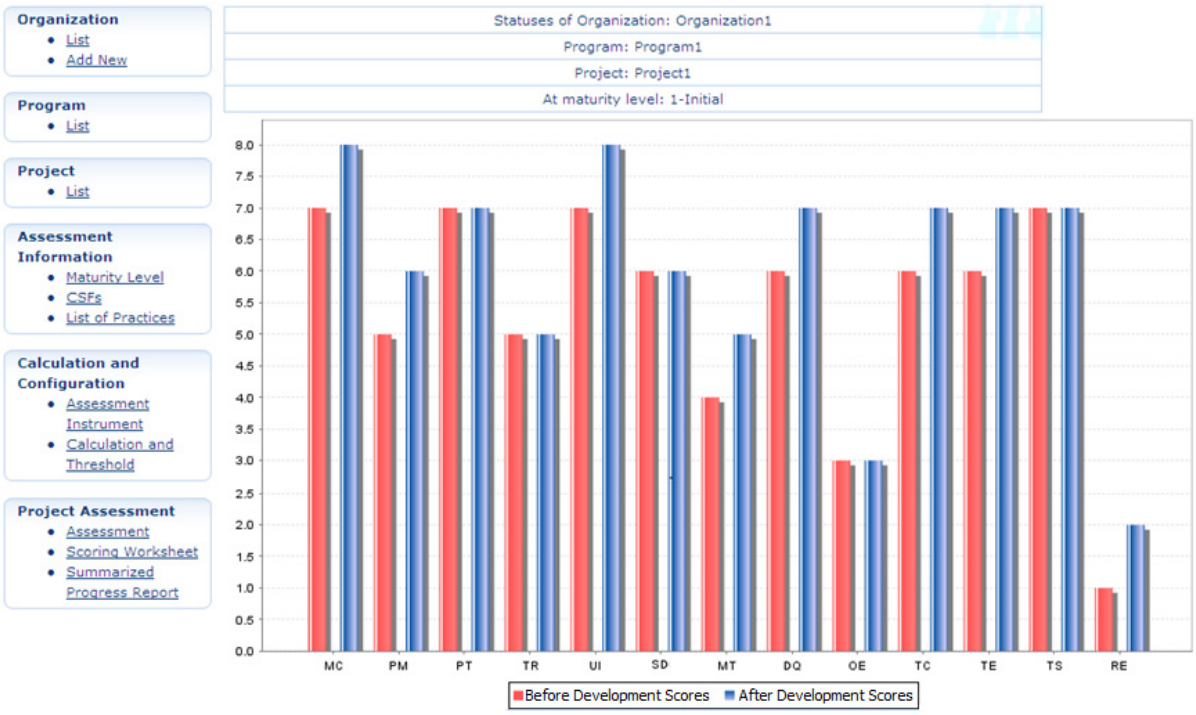


Figure 4-20. A sample screenshot of the before- and after- implemented CSFs comparison

This is the first prototype and therefore needs further evaluation and improvement. Consequently, practical tests of the framework and the prototype tool are carried out through case studies and presented in Chapter 5.

4.4 Summary

One of the factors that play a central role in quality software development is an efficient and effective software process, which can be derived from software process development and continuous improvement. In this chapter, we have presented the developed software process maintenance framework which in this context means a framework for software process development and improvement. It consists of two core components: an SDM model and an integrated PMBOK-Scrum model.

First, the SDM model has been developed based on CMMI and CSF approaches. To design the SDM model, we have performed literature reviews (1) to compare similarities and differences between CSFs in SPI identified in the literature and the 13 CSFs affecting the successful agile software development identified in Chapter 2 [2, 183] and (2) to explore important agile practices (including the data quality aspect) in worldwide software projects. The results of the latter literature review were then used to compare similarities and differences with the results of a questionnaire-style information collection. The frequency analysis was used to extract quantitative data from the collected qualitative data in the

literature. The data of the questionnaire-style information collection was collected from seven respondents in three companies in Thailand in June 2010 and analyzed using the median values. The main aim to carry out this questionnaire-style information collection was to find some identification by looking into a set of agile practices that are recognized as important for implementation in these three companies.

Concerning the first literature review to compare similarities and differences between CSFs in SPI identified in the literature and our identified 13 CSFs, the findings reveal that the CSFs in SPI identified in the literature are similar to our identified CSFs in software development. This implies that the better the organization can implement these CSFs, the better the organization can also achieve successful software development and higher maturity levels.

Concerning the second literature review to explore important agile practices in worldwide software projects, the findings reveal that there were 64 agile practices globally recognized as important for successful agile software development, as presented in Table 4-3. However, all of them have different levels of occurrences in the literature. Thirteen agile practices occurred at a rate of 75% or higher. Fourteen agile practices occurred at rates between 50% and 74%; whilst 37 agile practices occurred at a rate of less than 50%.

Concerning the questionnaires-style information collection, the findings reveal that there were 67 agile practices locally recognized for successful agile software development, including all of the 64 agile practices found in the literature and three additional agile practices. Within these agile practices, 18 agile practices were recognized as the most important, whilst the rest of 49 agile practices were recognized as important. As our focus of this study is on software development in Thailand, we have decided to use all of the 67 agile practices for designing the SDM model.

The SDM model has been developed to assist practitioners in assessing and improving their implemented software processes. It consists of three dimensions: maturity stage, CSFs, and assessment. First, the maturity stage dimension contains four CMMI-based maturity levels: “Level 1-Initial”, “Level 2-Managed”, “Level 3-Defined”, and “Level 4-Optimizing”. Second, the CSF dimension contains our 13 identified CSFs. Based on the perception of CMMI process area division amongst different CMMI maturity levels; the identified CSFs were categorized into three categories: foundation, standardization, and support. The foundation category contains the CFSs that support to establish project management processes, necessary process discipline, and commitments amongst key stakeholders. It can be linked to the maturity level-2 “Managed”. The standardization category containing the CSFs that support the design of systematic structures can be linked to the maturity level-3 “Defined”. The support category containing CSFs to support continuous SPI activities can be linked to the maturity level-4 “Optimizing”. As a guide on how to implement the CSFs, a list of agile practices has been designed under each CSF. Third, in the assessment dimension, an assessment instrument successfully developed and tested at Motorola has been adapted to assess agile practices. This instrument can be applied at many levels, e.g., organization, department, and project levels. The results of the SDM model can be used to guide practitioners on their current software development maturity and weak practices that demand immediate and sustainable improvement.

Second, the integrated PMBOK-Scrum model aims to assist practitioners in developing and implementing integrated project management and software development processes. It has been developed by merging the core entities of the PMBOK meta-model with the core entities of the Scrum meta-model into three layers (i.e., a managerial layer, a

production layer, and an integration layer) using Unified Modeling Language (UML). The layers of managerial and production are derived from the distinction of concepts, meaning each distinct class on PMBOK and Scrum meta-models can be left in managerial and production layers, respectively. The integration layer is derived from an overlapping of concepts. This means two classes on each of the meta-models have the same concept that can be transformed into a single concept inside the integration layer. Moreover, a set of eight constraints is provided in order to support practitioners who are responsible for planning a software project with a comprehensive set of project management and software development processes and to ensure the consistency of the integrated PMBOK-Scrum model.

The SDM model and the integrated PMBOK-Scrum model have been incorporated into the software process maintenance framework in order to provide the "what" to improve with a software process assessment mechanism and the "how" to implement with and a comprehensive set of project management and software development processes that could lead to the improvement of software development performance in terms of efficiency and effectiveness. However, using the framework might be too complex without the right tools. Therefore, a prototype tool supporting the use of the framework was created as a Web-based application, using the Java Language and a MySQL database. It helps an end user (e.g., a project manager and a team leader) to get insight into the current software development maturity and the health of the software practices by assessing the identified CSFs through a list of agile practices required by the SDM model. The obtained weak practices will be used to plan an integrated project management and software development processes together with the defined information (e.g., project, phase, and activity) as required by the integrated PMBOK-Scrum model. The defined processes are then validated and prepared in an XML file format for export to the organization's project planning tools. Nevertheless, the framework and the prototype tool need further evaluation and improvement. Consequently, practical tests of the framework and the prototype tool are carried out through case studies and presented in Chapter 5.

Chapter 5

Two Case Studies of the Software Process Maintenance Framework

To ensure whether software development approaches fit into a particular context, it generally requires a practical test. We have developed a software process maintenance framework which in this context means a framework for software process development and improvement. The framework provides the "what" to improve with a software process assessment mechanism and the "how" to implement with and a comprehensive set of project management and software development processes. This chapter aims to perform a feasibility check on whether the framework is practicable in real-life software projects. This has been accomplished through two case studies in the state-owned Thai telecommunications industry. Data was collected via on-site observations, semi-structured interviews, and questionnaires. The findings indicate the generation of positive effects by (i) increasing software development performance in terms of efficiency (e.g., increasing work completeness and team productivity) and effectiveness (e.g., reducing defects and increasing customer and team satisfaction); and (ii) cultivating collaborative teamwork, informal frequent communications, and knowledge sharing culture. Based on these advantages, the framework is perceived by its usability and the acceptance in terms of usefulness and ease of use. This chapter also presents the challenges that impact on the project results; the changes necessary to adapt the framework; and the mechanisms that the participants transferred, learned, applied, and integrated new knowledge into their existing software processes. The requirements for successful adaptation of the framework are then discussed.

5.1 Introduction

Many organizations face either unfulfilled promises about software quality gained from applying software development approaches, or the inability to manage the software process realized as their fundamental problem [194]. The search for solutions to this barrier has continued for decades. Consequently, we have developed a software process maintenance framework which in this context means a framework for software process development and improvement. The framework aims at providing a comprehensive set of project management and software development processes with a mechanism for assessing and improving software development performance in terms of efficiency and effectiveness. The framework consists of two core components. First, a Software Development Maturity (SDM) model is based on Capacity Maturity Model Integration (CMMI) and Critical Success Factors (CSFs) approaches. Second, an integrated PMBOK-Scrum model is based on Project Management Body of Knowledge (PMBOK) and Scrum approaches.

The framework can be used as a general framework for improving software processes in many management areas required to deal with volatile requirements, e.g., configuration management for assuring software quality, key stakeholder management for reducing risks of project failure, quality management in which various kinds of software testing (e.g., unit,

integration, system, and user acceptance tests) should be performed, and subcontracting management. As the framework provides all aspects of traditional project management and software development processes in software development, it partially conforms to approaches offering features similar to the framework, e.g., CMM (Capability Maturity Model) and CMMI. Moreover, the framework suggests some software development techniques, e.g., coding standards and continuous integration. Thus, it could somewhat be applied to approaches offering similar software development techniques, e.g., eXtreme Programming (XP) [219] as also suggested by [220] that XP can be used to complement a Scrum-based software development approach.

A prototype tool to support the use of framework called SPAD (Software Process Assessment and Development) has been developed. It helps an end user (e.g., a project manager and a team leader) to get insight into the current software development maturity and the quality of the software practices by assessing the identified CSFs through the list of agile practices required by the SDM model. The obtained weak practices will be used to plan integrated project management and software development processes together with the defined information (e.g., project, phase, and activity) required by the integrated PMBOK-Scrum model. The defined process is then validated and prepared in an eXtensible Markup Language (XML) file format for export to the organization's project planning tools.

Our main objective of this chapter is to perform a feasibility check on whether or not the developed software process maintenance framework is practical in real-life software projects through two case studies in the Thai telecommunications industry. To get high accurate and precise results, our case studies are hence focused on direct players in the industry. The original target company was Telecom Player1 who has been surveyed about its agile practices affecting successful software development, presented in Section 4.2.1 in Chapter 4. It is the only one out of the three companies being a key player in the industry and responding to our questionnaire-style information collection in Chapter 4. Unfortunately, obtaining its permission to test the developed software process maintenance was not achieved. Instead, two case studies in CAT Telecom Public Company Limited (CAT) and TOT Public Company Limited in Thailand (TOT) were carried out from November 2010 to February 2011. The data collection was carried out through on-site observations, individual interviews, and questionnaires in order to provide the descriptions of what has been done, what is working, what is not working, what still needs improvement, and guidance on how to make that improvement, through the following set of research questions.

RQ5-1: How can the developed software process maintenance framework be executed efficiently and effectively in the given context?

RQ5-2: What are the challenges that impact software development, using the developed software process maintenance framework?

RQ5-3: What changes are necessary to adapt the developed software process maintenance framework?

RQ5-4: How do practitioners transfer new knowledge into their existing software processes?

RQ5-5: What is the developed software process maintenance framework perceived usefulness and ease of use?

RQ5-6: What are the requirements for successful adaptation of the developed software process maintenance framework?

This chapter is organized as follows. The following section describes the research approach. The descriptions of analysis and results as well as summary of the findings are then discussed.

5.2 Research Approach

This section describes how the research design for this chapter is arranged.

5.2.1 Data Collection

The case study methodology is well suited for software development research due to contemporary phenomena in its natural context [221]. We have borrowed basic ideas of field case study (e.g., direct observations, interviews, and questionnaires to discuss the challenges within the context, the results achieved, and the lessons learned) for testing the software process maintenance framework in the Thai telecommunications industry. The test was split into two phases: the first phase performed at CAT Telecom Public Company Limited (CAT) from November to December 2010 and the second phase performed at Public Company Limited (TOT) from December 2010 to February 2011. The main goal of the first phase is to provide an analysis of the application of the framework and the practitioners' process to learn, use, and integrate new knowledge (e.g., the framework and software-development-related knowledge) into their existing software process; whilst the second phase involves collecting only interesting data which offers our double check on certain factors and issues in the case studies. There are three reasons to choose these two companies. First, from their beginning until today TOT and CAT are still two of the major telecommunications operators in Thailand [222]. TOT, the monopoly provider of fixed-line services, remains the largest player. It had market share of national fixed-line services at 58.95% at the end of the second quarter in 2011 [223], whilst CAT had the largest share of the international call service in 2010 [224]. Second, most of the participants had strong experience in software development. The CAT team consists of four members: a product manager who was the product expert and had 14-year experiences in PMBOK, a technical manager who had 15-year experience in software development, a developer and a tester who had 7- and 2-year experience in software development, respectively. On the other hand, the TOT team consists of a project manager and a developer. They had 10-20 years in the telecommunications industry and 3-6 years in software development. Third, both companies showed great interest and desire to participate in testing the software process maintenance framework with the prototype tool.

Initially, the authors explained what the case study was about. The participants then made the authors verbal and email queries to solicit more information about the use of the framework and the tool. During the case projects, the participants used the framework and the tool to assess their software development maturity without any suggestion from the authors, but to define integrated PMBOK-Scrum processes and to implement the software processes in the Scrum way with some suggestions from the authors. Quantitative and qualitative data were collected through on-site observations, individual semi-structured interviews, and questionnaires. We interviewed three team members in CAT face-to-face (e.g., a product owner, a Scrum master, and a developer); and a Scrum master who also acted as a product owner and a developer via an international call for the case project in TOT due to the limited available time of the team. Each interview lasted 30-90 minutes and was recorded to allow for subsequent accurate analysis of the data. During the observations and interviews, we

collected the data pertaining to how the framework was being executed in the teams, what challenges occurring during software development, how new knowledge was transferred into their existing software processes, what changes were necessary to adapt the framework, and how satisfaction of the framework was perceived. A TAM-based (Technology Acceptance Model-based) questionnaire for investigating the acceptance of the framework and the tool was also developed based on the related TAM literature and circulated to all interviewees. The questionnaire had two versions. The original one was in English and then translated into Thai.

5.2.2 Threats to Validity

There are four types of threats that have an impact on the validity of the outcomes. These are threats to internal validity concerning the ability to isolate and identify factors influencing the studied variables, construct validity concerning the ability to measure the construct under study, threats to conclusion validity concerning the ability to draw the valid conclusion about relationships based on statistical inference, and threats to external validity concerning the ability to generalize the findings [225]. A threat to internal validity is the objectivity of measurements affected by the interpretation of the authors. To reduce this risk, the quality of the data obtained during the case projects was monitored by the participants and the results were discussed with the participants. This data quality check also helped reduce threats to construct validity. The result comparisons between the inference statistics from the cases and qualitative data from the interviews help to diminish a threat to conclusion validity. Furthermore, threats to external validity are, firstly, we did not have history documents of the participating companies' existing software projects. Secondly, we collected only interesting data in the second phase performed at the TOT team. The main reason was to double-check certain factors and issues in the case studies. Thirdly, the project scale of the cases was relatively small in terms of the team size and the project duration. Fourthly, the participants were previously inexperienced in agile software development. Lastly, the focus was on only state-owned enterprises, not private companies who are leaders in the overall Thai telecommunications market [226]. Different software development environments may give different results. These limited the generalizability of this study. To reduce this risk, we used two different cases with different types of information systems, organizational cultures, etc. The contexts of the cases were described in order to make explicitly clear to what degree the results are generalizable. The authors also tried to analyze to what extent the findings are relevant or similar to the findings of other cases. As the result of the continuation of using the framework on other software projects in the participating organizations, this implies that generalizability should more or less be increased, albeit a case study naturally limits generalizability due to its specific context. Moreover, the current trend towards adopting agile methods in Thailand is just at the initial stages [188, 189]. This implies that a majority of companies in the Thai telecommunications industry may probably still currently either use traditional software development methods or have traditional software development environments. Hence, this study may provide generable results to companies or software projects having contexts similar to the cases. However, additional case studies are needed to increase the generalizability of this study. Other two limitations are as follows. First, due to the narrow focus of our samples of TAM-based questionnaires, it is recommended that the interpretation of results remains limited to the chosen context. Second, the case projects were small and non-complex. This limits the ability to evaluate the framework on whether it can efficiently and effectively overcome major shortcomings of agile methods (particularly the

Scrum method) in some management aspects, e.g., limited support for developing with large teams, high quality assurance, and procurement management.

5.3 Analysis and Results

From the case projects, the overall reaction of all team members was positive. While they seem to embrace integrated PMBOK-Scrum processes, there was little resistance as to whether some software processes would not fit for their organizational and software development environments. However, they expressed the desire to adapt such software processes when they felt comfortable, to learn and understand things that went well and what may need improvement, and to incorporate new knowledge into their software processes. Facilitating discussions in this section, the interview questions matched with the relevant research questions (i.e., RQ5-1 to RQ5-5) are presented in Appendix C. The findings are described into three main parts as illustrated in Figure 5-1.

Part I: Software Process Assessment illustrated in the blue area in Figure 5-1 presents the teams' current software development maturity and weak practices that demand immediate and sustainable improvement.

Part II: Software Planning, Development, and Outcomes illustrated in the light green area in Figure 5-1 presents how practitioners set up, planned, and executed their software projects under four sub-sections: Scrum Planning Meetings, Sprint Planning Meetings, Sprint Executions, and Sprint Review and Retrospective Meetings. This part then presents whether or not the framework was perceived as satisfactory and could contribute to the improvement of software development performance in terms of efficiency and effectiveness, under two sub-sections: Customer and Team Satisfaction and Software Process Performance.

Part III: Acceptance of the Framework illustrated in the yellow area in Figure 5-1 presents whether or not the framework was perceived as acceptable in terms of usefulness, ease of use, and intention to use.

Under each sub-section, we also describe the answers to the RQ5-1 "How can the developed software process maintenance framework be executed efficiently and effectively in the given context?" under the item "How to execute the framework"; the RQ5-2 "What are the challenges that impact software development, using the developed software process maintenance framework?" under the item "Challenges"; the RQ5-3 "What changes are necessary to adapt the developed software process maintenance framework?" under the item "Necessary Changes"; and the RQ5-4 "How do practitioners transfer new knowledge into their existing software processes?" under the item "Knowledge Transfer". These answers are also summarized in the next section.

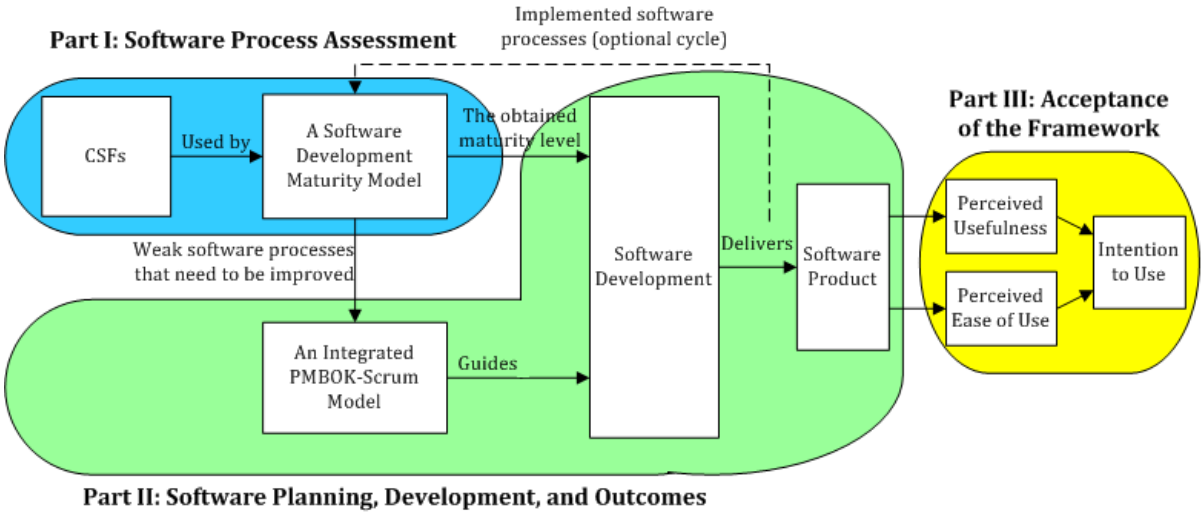


Figure 5-1. Three parts of our software process maintenance framework

Part I: Software Process Assessment

How to execute the framework: At the starting point, all team members carried out an assessment of their implemented software process, using our tool. At the threshold of 7 for an assessment calculation as guided by Motorola [215], the results in Table 5-1 present a clear explanation that CAT stands at the maturity level 2-"Managed" of the SDM model due to the weak factor "Team Size" belonging to the maturity level 3. At this stage, they did not focus on the factor "Reviews" at all. Therefore, the CAT team should improve the "Team Size" in order to get the higher maturity level and meanwhile maintain all factors belonging to the maturity levels 2 and 3 in order to sustain their quality software processes. All team members agreed with the results, except one developer as pointed out *"(Our existing software process) doesn't really conform to any standards (e.g., PMBOK and CMMI) yet. I think we should get the lower maturity level instead."* This implies that their assessment may somehow be overrated. On the other hand, TOT stands at the maturity level 1-"Initial" of the SDM model. TOT usually uses an outsourcing method for their existing software projects. Owing to less experience in internal software development, they decided to start from scratch on the case project. Hence, the TOT team should improve all factors, starting to focus on the factors belonging to the maturity level 1.

Table 5-1. A summary of the assessment results

Maturity Level	Category	CSF	Score (0-10)	
			CAT	TOT
2	Foundation	Management Commitment	9 (Strong)	-
		Project Management Process	9 (Strong)	-
		Project Type	7 (Strong)	-
		Training Support	9 (Strong)	-
		User Involvement	9 (Strong)	-
3	Standardization	Agile Software Development Process	8 (Strong)	-
		Appropriate Methods, Techniques, and Tools	9 (Strong)	-
		Data Quality	10 (Strong)	-
		Organizational Environment	10 (Strong)	-
		Team Capability	10 (Strong)	-
		Team Environment	10 (Strong)	-
		Team Size	4 (Weak)	-
4	Support	Reviews	-	-

Similarly to CMMI, Software Process Improvement (SPI) is a long-term approach. The recent report of Software Engineering Institute (SEI) shows the median time for moving from one maturity level to the higher one: 4.5 months for moving from maturity levels 1 to 2, 19 months for moving from maturity levels 2 to 3, 24 months for moving from maturity levels 3 to 4, and 19 months for moving from maturity levels 4 to 5 [227]. It is rarely plausible to get a higher level within 1-2 months. Owing to this time constraint, both CAT and TOT teams decided not to perform an assessment after completing the case projects.

Part II: Software Planning, Development, and Outcomes

As illustrated in the light green area in Figure 5-1, this part presents how practitioners set up, planned, and executed their software projects under four sub-sections: Scrum Planning Meetings, Sprint Planning Meetings, Sprint Executions, and Sprint Review and Retrospective Meetings. This part then presents whether or not the developed software process maintenance framework is perceived as satisfactory and could contribute to the improvement of software development performance in terms of efficiency and effectiveness, under two sub-sections of Customer and Team Satisfaction and Software Process Performance.

A. Scrum Planning Meetings

How to execute the framework: After the software development maturity assessment, the teams set up and planned the software projects. The case project in CAT developed additional Web-based functionalities bundled into the ongoing software project, whilst the case project in TOT developed a small decision support application. Both software projects are non-life critical. They could be categorized as a classic software project which means it requires the creation of a project plan for a significantly new body of work [228]. It has a high level of unknowns at the start but these are mostly resolved early, and few new unknown

arise during execution. However, requirements in real-life practice were constantly unstable as stated by team members in CAT. As CAT had being used PMBOK for their existing software projects, all related project documents (e.g., the project charter, the program/project roadmap, the project management plans, and the software design) were thus used in the case project. On the other hand, at the case study time TOT usually used an outsourcing method for their existing software projects and the case project was a new project. All required documentation was therefore created. Consequently, the TOT team began the case project with two PMBOK processes, i.e., conducting a project charter and performing a stakeholder analysis. Once the case projects were formally authorized the case projects then flowed through the sprint planning.

B. Sprint Planning Meetings

How to perform the framework: Albeit the CAT team had been developing an ongoing software project for almost two years, a project roadmap was not enlightened to the team. At the beginning of both case projects, the product owners explained the project roadmap in order to draw the teams a big picture. It was recognized as an important practice that can drive the teams into the right direction. When observed and asked what project management aspects were considered, the TOT team followed PMBOK guidance by planning integration and configuration management in a simple way; while the CAT team used their existing integration management plan.

The CAT team was not concerned about whether the scope would become broader since they had continued enhancing the application's functionality. However, in order to prevent any risk from enlarging the project's scope, the CAT team considered the capacity of network and application architecture. On the other hand, the TOT team developed a simple scope management plan to prevent scope creep, used together with a product backlog. Concerning the gathered user and technical requirements, they were described and ranked using a relative weighting approach. Work estimation which is a collaborative effort amongst team members was also used. This data (i.e., prioritized requirements and estimated effort) was then logged into product backlogs. After getting the first set of requirements to develop, the teams then broke down the work into tasks, estimated task efforts, assigned responsible persons, and logged these data into sprint backlogs. As observed, this practice seemed to provide smaller and manageable tasks to the teams.

After the verification of the scope, the identified requirements were then scheduled into iterations. Time was also not a major constraint in both teams. Rather than attempt to build the entire application from ground zero or through long-term iterations, both teams used small iterations lengthening between 1-4 weeks. The CAT team initially started using 2-week iterations due to time given for learning agile during the development, but later preferred one-week iterations to deliver features. On the other hand, the TOT team initially used 2-week iterations. However, it was not enough to produce a meaningful functionality due to two main reasons as claimed by the Scrum master. First, they did not well assess the appropriate techniques and tools before using them, nor did they analyze the data quality from the source systems before development. Second, the Scrum master needed to transfer programming techniques to the developer in the team. Hence, they needed more time to create meaningful and valid features. As knowledge training and coaching requires time, adequate independence from the software development tasks needed to be provided to team members involved in the knowledge transfer. This shows that the amount of user and technical requirements that needed to be implemented in each iteration was sometimes reduced.

Owing to internal development, both teams did not emphasize cost management. To guarantee software quality, the CAT team followed the Scrum validation and verification ways; whilst the TOT team used a simple PMBOK quality management plan and managed through sprint reviews. Concerning human resource aspects, the CAT team was formed with the same team responsible for the existing ongoing software project, composing of a product owner, a Scrum master (also acted as a developer), a developer and a tester. This helped to reduce time to learn business logic, programming languages, and development tools. On the other hand, the TOT team was formed with only two members, i.e., a Scrum master who had multi-projects and multi-roles (i.e., product owner, developer, and tester) and a developer, due to the small size of the software project and their available resources at that time. To accelerate software development, the product owner in CAT had full authority to make decisions but not in TOT. Team members in TOT had much experience in telecommunications but not software development; whilst team members in CAT had 7-15 years of experience in software development.

Both teams followed the Scrum communication mechanism through sprint planning meetings, daily meetings, and sprint review meetings; except through sprint retrospective meetings only in the CAT team. Considering development environments, the CAT team worked approximately 60% in co-location and approximately 40% in distributed sites; whilst the CAT team worked fully in co-location. Both teams cultivated informal communication for collaborating on work and transferring knowledge. Creating more chances of communication, the CAT team established several communication channels, e.g., face-to-face communications, mobiles, emails, instant messaging, and e-conferencing. Since management in TOT had a heavy workload due to multi-projects, the team was not fully approachable. Dealing with this situation, they used approximately 70% for mobiles and only 30% for face-to-face communications. Both teams used non face-to-face media for necessary explanation and feedback when the product owners were remote and for technical knowledge exchange when team members worked in different sites. This enabled them to obtain quick feedback.

Moreover, the CAT team planned risk management with short-term and long-term solutions using risk and impediment backlogs; whilst the TOT team created a simple PMBOK risk management plan for the overall project and used risk and impediment backlogs for iterations. The main reason for this, as observed in the TOT team, was that they preferred to get familiar with PMBOK risk management for further complex software projects. Those plans were continuously reviewed and adjusted during the case projects. Concerning procurement management, it was not performed in both case projects. However, both teams were planning to acquire outsourcing teams for future software projects.

When asked about documentation of their previous software projects, the product owner in CAT said *"We don't place an emphasis on heavy documentation, (just only necessary documentation)."* This is in line with agile philosophy, although they applied PMBOK for their previous software projects. On the other hand, although TOT usually uses an outsourcing method; surprisingly, the product owner said *"We didn't get any documentation including source codes from the outsourcing teams. It's now causing huge trouble for maintaining the applications… and for recovering the knowledge lost… Now, we must do documentation."* Consequently, they started to change, starting from the case project. As stated by the product owner (also acted as a Scrum master) during developing software, they preferred to conduct all related documents in order to get familiar with the tools (e.g., project management templates, product backlogs, and sprint backlogs) and to use the developed documents as templates for further software projects.

Challenges: Several challenges related backlog administration, human resource management, communication, and documentation are described as follows.

Concerning backlog administration, there were four challenges. First, goals and requirements described to team members sometimes remained unclear. This in turn resulted in misunderstandings and some rejected work. Second, although work could proceed in priority order, there was a conflict between business value criteria and technical criteria in requirement prioritization. However, in dealing with this matter, they sought to set ground rules for score rating and used CPM (Critical Path Method) or PERT (Program Evaluation and Review Technique) together with business value conditions. Third, there was a conflict in dividing the work into manageable tasks, due to different opinions between software developers in CAT. Owing to the authority of the team leader, developers sometimes felt the need to follow his solution without collaborative decision-making. This indicates that they were still familiar with the strong traditional manner. Last, both teams felt they spent more time on backlogs. A developer in CAT said *"Our weak skills in the work breakdown structure slowed us down."* Due to the beginning of their journey, this led to more non-task effort on each sprint. This effort can be minimized when they gain more experience on these practices. Otherwise, it might cause a problem if this feeling was not reduced. Although the team faced many challenges, they were strongly satisfied with sprint planning meetings. This was because this kind of meetings gave the team a better understanding of requirements and better scope management.

Concerning human resource management, there were three challenges. First, due to the multi-roles of the Scrum master in CAT and the multi-projects of the Scrum master in TOT, the Scrum masters did not fully act like a Scrum facilitator. Thus, software developers in both teams sometimes needed to solve an impediment by themselves that could lead to an obstacle to rapid development. The possible means to deal with this challenge is to clearly clarify roles and responsibilities to all team members and users and make them aware of their roles and responsibilities. Second, some of management people in CAT and TOT teams were strongly familiar with the traditional software development manner. To execute software projects in a hybrid agile-traditional way, management is required to transition and balance their managerial styles between command-and-control and leadership-and-collaboration management [65, 229-231]; whilst team members need to learn how to be collaborative leadership (or self-managing). Third, albeit a self-managing team is one of the unique aspects of Scrum, there is a need of team leaders who can make a decision and guide them in the right direction. This is supported by the expressions of the Scrum master in TOT that *"Owing to the nature of our culture, we need to have a team leader."* and a team member in CAT that *"I still prefer to have (the Scrum master) as a team leader."* This may result from having a long journey of traditional software development and traditional organizational cultures.

Concerning communication, there were two challenges. First, management in TOT had a heavy workload due to multi-projects; hence, its team was not fully approachable. This caused software development to be slow. Second, although face-to-face communication is strongly promoted throughout agile projects [232] and practitioners expressed that *"It is an efficient method."*, they used it only on demand. Instead, they mostly used other established communication media (e.g., instant messaging and phones). However, non- or virtual face-to-face communication must be used carefully as the product owner in CAT stated that *"Using Skype or phones, the team sometimes got information lose and mutated… We solved this problem by using the writing or whiteboard features in Skype…and it worked."*

Concerning documentation, most documents were informal and less-detailed. This shows that the teams primarily relied on the knowledge residing in the individual team

members more than the explicit knowledge in documentation. Even though a lack of detailed software design (e.g., flowcharts) did not cause a problem in the case projects, it may take time for newcomers to recover the knowledge lost when they leave the projects.

Necessary Changes: There were four main changes. First, the project goals, objectives, and roadmaps must be clearly explained to all team members to ensure that all team members are going in the same direction.

Second, the teams had to change from using only the ongoing project management plans throughout the software projects to using both iterative and ongoing project management plans.

Third, unlike their traditional software development practices where communications between users and software developers were heavy at the beginning and the end of software projects and where lots of volatile requirements were difficult to deal with, users needed to be involved in almost all stages of software development (e.g., from iterative planning through iterative reviews, retrospectives, and closures). Not only did management have to plan a project, but users and all team members were required to take part in the planning. Moreover, they needed to rely on continuous communications and lots of meetings.

Last, as mentioned, the TOT team had to perform necessary documentation in order to retain knowledge within the organization.

Knowledge Transfer: In the planning stage team members who acted as knowledge sources mostly prepared materials for describing program and project roadmaps, goals and objectives, and user and technical requirements mainly through face-to-face communications. Documentation was also performed to ensure that knowledge of the software projects exists in the organizations. Concerning how to use and integrate new knowledge into their existing software processes, both teams considered new knowledge whether or not it is useful, suitable for their organizational and team cultures, and compatible with their existing software processes. For instance, although the Scrum master in the TOT team agreed that key team members (e.g., a product owner and a Scrum master) directly involved in the software project should have full authority for rapidly making decisions as suggested by the SDM model, this process was not approved by top management (or a project sponsor), due to the unsuitability with the TOT organizational culture. In contrast, there was no significant incompatibility amongst the CAT organizational culture, their existing software processes, and all software processes suggested by the SDM model.

However, there was a conflict between business value criteria and technical criteria in requirement prioritization. To deal with this problem, the CAT team set ground rules for rating scores and planned to use CPM (Critical Path Method) together. Once satisfied with the outcomes, new knowledge was then integrated into their existing software processes. Moreover, we found that perceived ease of use of the transferred knowledge affected knowledge transfer effectiveness. For instance, weak experience in the work breakdown structure led the CAT team to more non-task efforts and slight deceleration in the planning stage. Owing to the short period of the case project, the improvement of this practice could not be evidently observed. However, this effort may be minimized when they gain more experience on this practice. Otherwise, this practice is unlikely to be integrated into their existing software processes.

C. Sprint Executions

How to execute the framework: During sprints, daily meetings were established to coordinate work, synchronize efforts, and tackle anticipated problems. The meetings took place around 5-15 minutes with non-fixed place and time in both teams. When asked how to perform the three Scrum daily-meeting questions, the daily-meeting questions of "What did you do yesterday?" and "What will you do today?" were not asked every day. One issue we found is that the Scrum master in CAT felt *"Asking these two questions every day seemed like micromanaging or not having confidence in the team."* Hence, he asked these two questions approximately few times a week. This implies that management did not perform strong micromanagement in their existing software projects. In contrast, the developer in CAT *said "It's a normal thing to do."* However, both CAT and TOT teams emphasized the occurring impediments which were related to the third daily-meeting question of "What impediments are in your way?". In the view of product owners, these question discussions were recognized as important. Noticeably, the product owner in CAT encouraged and facilitated support to the team and its performance. These meetings provide management early visibility to tackle risks and impediments.

During the coding stage, the CAT team followed their coding standard for having easily maintainable and expandable code, pursued simple design, and used code refactoring to allow for improving existing code to support new functionalities of the software application, as suggested by the SDM model. These practices were not only used for this case study, but also implemented into their existing software projects. They employed a configuration management system for controlling individual check-in, check-out, and continuous integration of their source code and applications. It was also used for supporting quality assurance. Their development environment closely mirrors the production environment to guarantee quality and minimize unexpected risks. Additionally, unit and integration tests were performed against test cases to ensure work completeness. On the other hand, the TOT team had less experience in internal software development. They thus faced many technical difficulties as the Scrum master said "*During the development, most occurring problems were about technical problems such as no available feature of the programming language supporting the available data types, insufficient data available in the source systems, the differences of data types in the source systems, and the appropriate tools and databases using in the case project.*" This shows that they did not well assess the appropriate techniques and tools before using them, nor did they analyze data quality from the source systems before the development. There was no configuration management system used in the case project. However, they had their own development environment and performed unit tests to ensure software quality.

Challenges: We found two challenges in the TOT team. The first challenge was about multi-projects of key team members (especially a Scrum master). As the Scrum master having experience in the programming language had to transfer the programming techniques to another developer, his multi-projects led to insufficient time for the case project and resulted in a late project completion. Second, a lack of well preparation of appropriate technical environments (e.g., assessing appropriate techniques and tools and analyzing data quality from the source systems before the development) at the early stage of software development significantly impedes rapid software development.

Necessary Changes: The teams had to change from responding changes immediately to freezing requirements during sprints (or iterations). It is a common unpleasant situation in the traditional software projects, as a developer in CAT said *"Changes that occurred during*

the existing development always have top priority. When those changes interrupted my current work, it resulted in an immediate feeling of discouragement within me." When asked about their feelings on the freezing requirements during sprints, the developer replied *"I really liked it."* As observed, following this rule resulted not only in the team satisfaction and stronger work commitment without interruption, but also better relationship between the product owner and the team members.

Knowledge Transfer: Developers especially in the CAT team heavily exchanged technical knowledge during sprints. This paved the way for improving their development techniques and preparing system infrastructure for better software maintenance in the future through face-to-face communications and mobiles. On the other hand, the Scrum master in the TOT team transferred programming techniques to the team and supported the use of those programming techniques mostly though phones, not face-to-face communications. However, these activities were not ineffective due to the Scrum master's heavy workload in multi-projects and a lack of commitment in terms of time. In contrast, the team lacked absorptive capacity due to no prior experience in those programming techniques. This situation was thus likely to decrease motivation to use the knowledge.

Once new knowledge or the transferred knowledge (e.g., freezing requirements during the sprint and the daily question of "What impediments are in your way?") was perceived as useful or being able to solve their existing problems, the transferred knowledge was continually used and then integrated into their existing software process. Nevertheless, some knowledge (i.e., two Scrum daily questions of "What did you do yesterday?" and "What will you do today?") had to be adapted to fit their team cultures. For instance, developers in the CAT team usually reported their work progress at the scheduled time and spoke of any problems, as they occurred. This existing routine was somehow fossilized into their work behaviors. Therefore, those two daily questions needed adaptation before being integrated into their existing software processes.

D. Sprint Review and Retrospective Meetings

How to execute the framework: In sprint review meetings which are places for showing the team's accomplishment during sprint executions, both teams held approximately 30-90 minutes. The software products were verified and validated against the sprint backlogs. The product owners then determined which requirements had been completed against acceptance criteria, clarified to the teams the reasons for work acceptance and rejection, and discussed until all team members accepted the results and/or product modification solutions. When asked for opinions on the review meetings, the positive findings we found were cultivating teamwork and better software quality, especially in the CAT team. The product owner in the CAT team said *"It's like we commit to work together… take responsibility for failure together."* The team said *"It's like we reiterate requirements together again, to check on whether we are going in the same direction."* When asked about shortcomings of their existing software processes, the developer mentioned that *"There is code redundancy. We don't have time to review codes. By using sprints, we must review codes and test it in order to get the work completed."* Owing to the visibility of the actual project status and the validated product, management in CAT said they could gain better project control and product commercialization planning. On the other hand, the TOT team was satisfied with this kind of meetings but still needed time to gain more experience in software development for a better understanding and improvement of the software processes.

Sprint retrospective meetings are places for lessons learned by discussing on what went well, what did not, what could be improved in the next iterations. The TOT team did not yet concentrate on this kind of meetings due to insufficient time available on the part of the Scrum master. Only the CAT team performed these meetings holding approximately 20-30 minutes. Concerning software process improvement, the CAT team was just beginning their journey in gaining a deeper understanding of the software process and taking ownership of the software process. However, their solutions and software processes focused retrospectives were likely aimed to bring more effective control and diminish returns. Furthermore, as the CAT team was also planning to go for CMMI, they intended to use the SDM model as guidance for their preparation. During these two kinds of meetings, all related plans (e.g., project management plans, sprint backlogs, risk backlogs, impediment backlogs, lessons learned, and Burndown charts) were reviewed and adjusted. This iteration was formally closed. The software development then flowed into the next iterations. Before releasing products or closing the projects, the teams performed many levels of software testing (i.e., integration, system, and acceptance tests) against test cases and acceptance criteria to guarantee software quality.

Challenges: One challenge we found in the CAT team, the product owner was the only person who had extensive business knowledge in the organization. If the product owner leaves the project or the organization during the software project for any reason, it would be a catastrophe. The software project might be either too late in delivering the right product or rejected outright. Indeed, it is not easy to keep all team members equal, by sharing knowledge and skills on the software project. It emphasizes that sufficient knowledge transfer within teams is necessary.

Necessary Changes: From the disciplined to hybrid agile-disciplined manners, both teams had to change from validating and verifying work at the end of either long release cycles or the software projects to iteratively test, review work, and/or collect lessons learned in short-time iterations.

Knowledge Transfer: The observations show that the product owners in both teams were expert in their product areas and had business, managerial, and/or technical skills and expertise. As observed they were willing to share such knowledge with their teams. Team members were thus able to increase their software development performance. Software developers in both teams largely shared technical knowledge to each other. As observed, software developers in the CAT team enjoyed sharing technical knowledge more than other knowledge types. This shows that knowledge transfer effectiveness relies considerably on knowledge interest and communication density as the software developer in CAT explained *"When we work co-located, we usually enjoy exchanging technical knowledge."* On the other hand, the software developer in the TOT team had a lack of absorptive capacity to learn and apply the programming techniques due to no prior experience of such knowledge, as the Scrum master said "*(The software developer) often faces the technical problems and keeps fixing bugs she found."* This significantly affected rapid software development.

Concerning how to use and integrate into their existing software processes, we found that when the transferred knowledge (e.g., sprint reviews in the CAT team) was perceived as useful and satisfactory, team members were motivated to continue to use the transferred knowledge. Meanwhile, the transferred knowledge was integrated into their existing software processes. We also found that commitment in terms of time and effort and knowledge awareness greatly affected learning and transferring knowledge. For instance, insufficient time on the part of the Scrum master on the case project had led to non-focus on or non-application of a sprint retrospective in the TOT team. As this software process is considered

as new knowledge for them, a lack of commitment and practicing can result in knowledge transfer failure. On the other hand, even though the CAT team performed a sprint retrospective, they just began to learn and apply it. As observed, when they faced difficulties to perform this process (e.g., inexperience on recognizing which software process is working or is not working), their motivation to perform this practice was likely to decrease.

E. Customer and Team Satisfaction

The overall customer and team satisfaction was positive. When asked how satisfied they were with the implemented software process and the developed product before the framework was introduced, the product owner in CAT expressed her worry about a lack of team commitment *"The team always postpones product delivery, until it became very late… They don't often communicate with me for work discussion."* On the other hand, the team said *"The main problem was regarding unstable requirements (from the product owner) ... Today she wants these things, but tomorrow it changes to another thing."* The Scrum master also stated *"Work that cannot be completed gives me lower motivation to continue working."* Obviously, this led to frustrating situations, unhealthy relationships, very late delivery, and eventually less software quality. On the other hands, the product owner (also acted as the Scrum master) in TOT said *"We have no internal standard software development framework, principally using an outsourcing method"* and for the existing software projects "*We didn't get any documentation, including source codes from the outsourcing teams. It's now causing a huge problem for maintaining the applications… and for recovering the knowledge lost…"* This observably shows the weaknesses of their software development.

When asked how satisfied they were with the software process and the software product developed after the framework had been introduced, due to being able to deal with existing problems, the Scrum masters in both teams were strongly satisfied, whilst the others were satisfied. Interestingly, the CAT team expressed *"We now feel we have a standard for improving ourselves to do the work.",* whilst the Scrum master in the TOT team said *"We are using the framework on another software project. At this stage, we are educating the framework to the team and in the meantime analyzing the problems that affected the project (feasibility study). Therefore, we cannot yet say about the project performance. However, (the case project) went well with good team collaboration."* However, the Scrum master in the TOT team pointed out two weak points of the framework that *"The framework requires lots of meetings. Since we primarily use our e-meeting planning system, it's not too comfortable to schedule lots of meetings via the system. Therefore, we need to have a person who is mainly responsible for the meeting arrangements. Another weak point is that we have to do documentation since we never do it before."* From this point of view, we however consider the second weak point as a necessary change to use the framework instead. When asked how work is improved, the participants were in consensus that work performance increased. The details of software development performance are described in the next sub-section.

F. Software Development Performance

According to Scrum's weaknesses as presented in Chapter 3, the findings reveal that the framework can to some extent overcome some of Scrum's weaknesses in the following knowledge areas: (1) integration management (i.e., providing configuration management and details of many types of testing), scope management (i.e., a clearer sense of product's direction), time management (i.e., improving the predictability of time estimate for the whole project according to the scope management plan), and technical aspects (i.e., suggesting generic agile practices such as data quality technique, simple design, and code standard as suggested by the SDM model). Some of Scrum's weaknesses require more cultivation to overcome, e.g., commitment, collaboration, intensive communications, and knowledge sharing to build up team members' experience in real-life software projects. This emphasizes that there is a need for an effective knowledge transfer mechanism. Owing to small software projects of our case studies, this limits our ability to argue whether or not some Scrum's weaknesses (i.e., limited support for high quality assurance, large teams, outsourcing, and accurate cost estimate for the whole project) can effectively be overcome by the software process maintenance framework.

As the software process maintenance framework aims at contributing to the improvement of software development performance, there are two dimensions of performance appearing essential for software development: efficiency and effectiveness. Efficiency can be measured by software quality. Two key variables used to represent work efficiency in this study are team productivity [233, 234] and achieved doneness. Productivity can be considered by using velocity metrics. Velocity is the amount of requirements (or backlog items) successfully delivered in an iteration. Achieved doneness is a ratio of the amount of the tasks that the product owner accepts over the amount of the tasks that the team said was done at the sprint review. Effectiveness is often associated with doing the right things; therefore, two key variables used to represent software development effectiveness in this study are defect reduction and customer/team satisfaction [233, 234].

In the CAT team, the velocity was increased from 14 in the first iteration to 30 in the last iteration. The achieved doneness increased from 64.29% in the first iteration to 100% in the last iteration. Defects were reduced from 5 in the first iteration to zero in the last iteration. Based on the questionnaire findings, the average rated scores of (1) the increased work productivity, (2) the increased work effectiveness, (3) the increase work performance, and (4) the improved quality of software process and product were 4.33, 4.67, 4.67, and 4.33 out of 5 points. In the TOT team, we used only the questionnaire findings to analyze their work performance using the same set of variables tested in the CAT team. Their rated scores were 4, 5, 5, and 5 out of 5 points, respectively. According to work satisfaction in terms of perceived usefulness and perceived ease of use, the interview findings reveal that both CAT and TOT teams were strongly satisfied with their work and the software process maintenance framework. Consequently, they continued using the software process maintenance framework on their further software projects. Based on the questionnaire findings, the mean values of perceived usefulness and perceived ease of use rated by all participants were 4.357and 4.2 out of 5 points, respectively (see more details in Part III: Acceptance of the Framework). Based on the findings, we consequently conclude that our software process maintenance framework promises to provide the improvement of software development performance in terms of efficiency and effectiveness.

Part III: Acceptance of the Framework

As illustrated in the yellow area in Figure 5-1, this part presents whether or not the developed software process maintenance framework is perceived as useful, easy to use, and acceptable. The part also answers the RQ5-4 "How do practitioners transfer new knowledge into their existing software processes?" In this part, we also evaluate the perception of the respondents of the tool. Two important factors (i.e., perceive usefulness and perceived ease of use) of Davis's TAM [165] were used to evaluate the acceptance and the usability of the framework and also the tool. Usability is "capability in human functional terms to be used easily and effectively by the specified range of users, given specified training and support, to fulfill a specified range of tasks, within the specified range of environmental scenarios" [235]. This describes two key usability characteristics: effectiveness and ease of use [236]. Effectiveness can be referred as usefulness [237]. Hence, the usability of the framework and the tool can be associated with usefulness and ease of use. In the interviews, all respondents expressed that the framework and the tool was useful and easy to use. They would also continue using the framework and the tool to the gained positive results. This shows that the framework and the tool were perceived as usable and acceptable. However, usefulness leads to greater acceptance than ease of use. To get more supportive results, a TAM-based questionnaire was used at the conclusion of the case projects. As mentioned, due to the narrow focus of our samples, it is recommended that the interpretation of results remains limited to the chosen context.

A. Research Instrument

A TAM-based questionnaire was created to obtain the respondents' opinions on the framework and the tool. The three constructs (i.e., Perceived Usefulness (PU), Perceived Ease of Use (PEOU), and Intention to Use (IU)) were used to assess, using multi-item scales. PU is defined as the degree to which users believe that using the framework and the tool would enhance their job performance. PEOU is defined as the degree to which users believe that using the framework and the tool would be free of effort. IU is defined as the degree to which users intend to continue using the framework and the tool. The items were generated based on the existing studies [165, 238] and assessed on a five-point Likert scale, ranging from 1 "strongly disagree" to 5 "strongly agree". The items are presented in Appendix D.

B. Instrument Reliability and Validity

Reliability refers to the degree to which the scale is free from measurement error [192]. To evaluate the instrument reliability, Cronbach's alpha was used to measure the reliabilities of the entire scale and each of the constructs. Regarding the framework, the Cronbach's alpha values of the entire scale, the PU, the PEOU, and the IU are 0.840, 0.817, 0.847, and 0.819, respectively as presented in Table 5-2. Regarding the tool, the Cronbach's alpha values of the entire scale, the PU, the PEOU, and the IU are 0.934, 0.926, 0.931, and 0.923, respectively as presented in Table 5-2. All constructs having the coefficient of above 0.7 demonstrate acceptable reliability [192]. Besides, factorial validity refers to whether an instrument item really belongs to a particular concept or must be assigned to another [238]. Owing to a very small sample size, factorial validity of our instrument cannot be statistically verified using factor analysis and correlation analysis. In other words, a very small sample

size limits the ability to perform statistic analysis. However, our instrument was guided by the accepted or proven instruments [165, 238].

Table 5-2. Analysis of the reliability of the framework and the tool

Items	Framework (Cronbach's Alpha = 0.840)			Tool (Cronbach's Alpha = 0.934)		
	Mean	Standard Deviation	Cronbach's Alpha	Mean	Standard Deviation	Cronbach's Alpha
PU	**4.357**	**.378**	**.817**	**4.191**	**.436**	**.926**
PU1	3.500	.578	.806	3.33	.577	.934
PU2	4.750	.500	.819	4.667	.577	.923
PU3	4.500	.578	.806	4.667	.577	.923
PU4	4.250	.500	.819	3.667	.577	.923
PU5	4.750	.500	.842	4.333	.577	.934
PU6	4.000	.817	.857	4.333	.577	.934
PU7	4.750	.500	.842	4.333	.577	.934
PEOU	**4.200**	**.163**	**.847**	**3.867**	**.306**	**.931**
PEOU1	3.750	.500	.866	4.000	.000	.937
PEOU2	4.500	.578	.806	4.333	.577	.934
PEOU3	4.000	.817	.889	3.667	.577	.923
PEOU4	4.000	.000	.842	3.667	1.155	.946
PEOU5	4.250	.500	.819	3.667	.577	.951
IU	**4.750**	**.500**	**.819**	**4.667**	**.577**	**.923**
IU1	4.750	.500	.819	4.667	.577	.923
IU2	4.750	.500	.819	4.667	.577	.923
IU3	4.750	.500	.819	4.667	.577	.923
IU4	4.750	.500	.819	4.667	.577	.923

C. Perceived Usefulness

The mean values were used to analyze the perceived usefulness of the framework and the tool. The statistical results in Table 5-2 reveal that the mean values of the framework's PU and the tool's PU are 4.357 and 4.191, respectively. Considering the maximum scale of 5, we conclude that the respondents consider the framework and the tool useful. Considering on the framework's PU details, the respondents significantly gained the increased work performance, work effectiveness, and work benefits from using the framework (i.e., the PU2, PU5, and PU7 means of 4.750). The increased software process quality, software product quality, team productivity, and easiness to work were also perceived. However, the framework's ability to accomplish work more quickly was slightly affirmed. Considering the tool's PU details, the respondents significantly gained the increased work performance, software process quality, and software product quality (i.e., the PU2 and PU3 means of 4.667); whilst the increased work effectiveness, easiness to work, and work benefits were slightly less significantly perceived. The tool's ability to accomplish the job more quickly and increase work productivity was rarely affirmed.

D. Perceived Ease of Use

The mean values were also used to analyze perceived ease of use of the framework and the tool. The statistical results in Table 5-2 reveal that the mean values of the framework's PEOU and the tool's PEOU are 4.200 and 3.867, respectively. Considering the maximum scale of 5, we conclude that only the framework was perceived as easy to use. Considering on the framework's PU details, the respondents significantly perceived that the framework was clear and understandable (i.e., the PEOU2 mean of 4.500). They recognized that it was easy to become skilful, to be easy to use, and to remember how to perform tasks. However, easy learning to use the framework was slightly affirmed. Considering the tool's PEOU details, the respondents significantly perceived that the tool was clear, understandable, and easy to learn (i.e., the PEOU1 and PEOU2 means of 4.000 and 4.333); whereas recognizing that it was easy to become skilful, to be easy to use, and to remember how to perform tasks was slightly affirmed.

E. Intention to Use

According to statistical results in Table 5-2, the mean ratings of the framework's and the tool's IU are 4.750 and 4.667, respectively. Considering the maximum scale of 5, we conclude that the respondents appreciably portend to continue to use the framework and the tool. This supports that the framework and the tool were perceived as usable and acceptable. Owing to a very small sample size, the statistical results confirming that the PU and the PEOU have a significant positive effect on intention to use remain very limited. Instead, considering the "mean" which is the average of a set of values, the results reveal that the respondents put an emphasis on the PU greater than the PEOU of both the framework and the tool. This implies that the PU leads to acceptance more than PEOU.

5.4 Summary of the Findings

A. The answer to RQ5-1 "How can the developed software process development framework be executed efficiently and effectively in the given context?"

We summarize the following lessons learned that can be used as guidance on which opportunities need to be addressed for further increased software development performance. In order to maximize generalizability, we also provide what extent our findings are relevant or similar to the findings of other cases. The above descriptions and the following lessons learned help answer the RQ5-1 "How can the developed software process development framework be executed efficiently and effectively in the given context?".

Lesson 1: The teams executed the case projects in a way which is in line with Griffiths' recommendations on integrating agile and traditional project management [96]. Griffiths' recommendations and our findings suggest as follows.

- The PMBOK processes were used for project initiation (e.g., conducting a project charter and stakeholder analysis) and project closure (e.g., obtaining user acceptance and formally closing the software project).
- The modifications of the PMBOK and Scrum processes were used for project planning (e.g., using iterative and ongoing project management plans in various

management aspects throughout the software project guided by feedback from actual project performance as well as business and technical changes). This is also similar to the key findings of Karlström and Runeson's case studies [239] on integrating XP into traditional Coopers' Stage-gate models. They suggest adapting the project planning to accommodate for agile micro planning in combination with macro project planning.

- Scrum techniques were primarily used for project execution and controlling (e.g., daily meetings, iterative sprints, sprint reviews, sprint retrospectives, empowering the teams, and using Burndown charts).

Lesson 2: Several communication channels should be established when physical face-to-face communications cannot be fully implemented in order to increase the chance of responses. For instance, mobiles can be of good use for quick feedback, while mailing lists as a synchronous communication can increase the active and constant participation in software development [240]. Besides, e-conferencing with a whiteboard function helps reduce information mutation. Nevertheless, face-to-face communication that is recognized as the most effective method should be developed as a primary method.

Lesson 3: During software development, team building is important. The observations suggest that an effective transfer of knowledge and skills significantly impact on ongoing development activities and software project success. Management should thus provide adequate time and be able to train team members on relevant knowledge (e.g., business and managerial knowledge), whilst team members themselves should have motivation and willingness to share technical skills to others. This helps to cultivate an interactive work environment with shared values.

Lesson 4: A tester and a configuration management system should be integrated into a software development team, similarly to McMahon's lessons learned [154]. In a term of shippable work, it is used to describe the quality that incremental work must have [181]. This means that work must be fully tested and ready to ship before demonstrated to a product owner. Therefore, a tester should work closely with software developers to ensure complete test coverage. Moreover, a configuration management system can support the proper control of work integration to ensure quality shippable work.

Lesson 5: Teams should strictly focus on (1) monitoring and minimizing impediments and risks through daily meetings, (2) iterative validation and verification through sprint reviews, and (3) iterative inspection and adaptation of the software process through sprint retrospectives when applicable. These meetings are beneficial, e.g., to increase (1) internal communications (similarly to the Karlström and Runeson's findings [239]), (2) the quality of software processes and products and (3) early visibility of project progress and incremental products that in turn significantly values management on better project control and commercial planning. This also raises more opportunities for product management to bring potential incremental products to the market.

Lesson 6: Management providing strong commitment and facilitating the team with supportive environments is a strong enabler to enhance work motivation of team members. This in turn cultivates commitment between management (or users) and team members and a collaborative environment.

Based on the above lessons learned, the practices that were efficiently and effectively executed in the CAT and TOT teams are mapped with their related CSFs in the SDM model and summarized into Table 5-4. According to the assessment results, CAT stood at the maturity level 2-"Managed" of the SDM model. On the other hand, the findings reveal that

the TOT team might stand at maturity level 1-"Initial" of the SDM model, as they rated themselves to start from the beginning. This was due to many weak software practices under CSFs corresponding to the maturity level 1, found in the TOT team during the case. For instance, under the "Management Commitment" factor, there was a lack of management commitment in terms of time and effort to effectively train the programming language and support its use. Under the "Project Management Process" factor, there was a lack of agile-oriented project management process, a lack of intensive face-to-face communications, and a lack of continuous software process improvement. Therefore, Table 5-3 clearly shows that the better the CSFs are implemented, the better the increased software development performance can be achieved. However, there are at least five certain CSFs required to be implemented, including project management process; user involvement; appropriate methods, techniques, and tools; team capability; and team environment.

Table 5-3. A summary of the practices efficiently and effectively executed in the case studies

Level	CSF	Lessons Learned	CAT	TOT
2	Management Commitment	Provide commitment and support software development by management	X	
	Project Management Process	Establish several communication channels	X	X
		Primarily focus on the occurring impediments during daily meetings	X	X
		Strictly perform iterative validation and verification through sprint reviews; this helps increase early visibility of project progress and incremental products to value management on better project control and commercial planning	X	X
	User Involvement	Collaboration between users and team members	X	X
3	Agile Software Engineering Process	Integrate a tester and a configuration management system into teams	X	
	Appropriate Methods, Techniques, and Tools	Use PMBOK for project initiation and project closure; use the modification of PMBOK and Scrum for project planning; and primarily use Scrum for project execution and controlling	X	X
	Organizational Environment	Facilitate the team with supportive environments	X	
	Team Capability and Team Environment	Build teams through knowledge transfer and shared-value environments	X	X
4	Reviews	Iteratively inspect and adapt the software processes	X	

C. The answer to the RQ5-2 "What are the challenges that impact software development, using the developed software process maintenance framework?"

Answering the RQ5-2 "What are the challenges that impact software development, using the developed software process maintenance framework?", we summarize the following lessons learned that can be used as guidance on which challenges need to be addressed for further improvement. In order to maximize generalizability, we also provide what extent our findings are relevant or similar to the findings of other cases.

Lesson 7: The observation results of the CAT team and the assessment results of software development maturity they gained remain in conflict. For instance, they rated the factor of team environment at the greatest score of 10, but the actual software process implementation show a weakness of their implemented software processes (e.g., lacking self-managing teamwork). This may result from either the assessment being somehow overrated or a lack of the maintenance of their strong software practices. The assessment results are an indicator to guide where they are, but the software development results and the implemented software processes are even more important to express the actual software development maturity and the quality of the implemented software processes. In order to get the right suggestion from the SDM model, the teams should perform the assessment with honesty or the minimum bias.

Lesson 8: Clear statements of goals, objectives, and requirements need to be ensured at any time to avoid work rejected.

Lesson 9: During planning, requirement prioritization plays a key role in agile-oriented software development. This is because agile excels at the delivery of the most important and valuable feature to users. Work (or feature) breakdown structure is also important for estimating efforts and staffing. When conflicts related to these occurred, as we found in the CAT team, it is more likely to diminish motivation to work collaboratively and in turn work satisfaction. Therefore, conflict solutions should be established at the early stages, based on the negotiation between the users and the team. Those conflict solutions should be inspected and adapted to fit the team's circumstances.

Lesson 10: Backlog administration requires self-discipline. Although backlogs are useful planning, controlling, and tracking artefacts, the motivation to keep them up-to-date is still lacking in both CAT and TOT teams. This seems a common lesson as it can be found in other agile software projects [241]. Management should hence get the teams to have work motivation and self-discipline.

Lesson 11: Management should deem an appropriate workload allocation for each team member who is assigned to have multi-roles and/or multi-projects. Meanwhile, the team member who has multi-roles and/or multi-projects should carefully prioritize his/her own responsible work to execute since it significantly affects rapid software development. It is also important that roles and responsibilities should be clearly clarified to all team members and users, similarly to Karlström and Runeson's findings [239].

Lesson 12: Teams are required to have the ability to self-manage. The need of a team leader can be found in many agile software projects (e.g., [154, 202]) and also in both CAT and TOT teams. Since the teams are strongly familiar with traditional software development culture that easily impedes the ability of the team to self-manage; therefore, team leaders should coach and allow their teams to collaboratively self-manage and meantime balance the collaborative self-managing and leader-guided atmospheres.

Lesson 13: Teams must get out of the strong traditional sequential mentality. It does not work with the agile-oriented mentality [154]. When some traditional project management processes are required, management should balance discipline and agility to fit their organizational and team cultures and any particular circumstances by continuous inspection and adaption of the software processes.

Lesson 14: Even though face-to-face conversations are recognized as the most efficient and effective method by Agile Manifesto and especially the CAT team, they were not used intensively in both teams. Face-to-face communications significantly affect not only software development but also knowledge transfer. To reach a higher opportunity to achieve successful software projects, face-to-face communications should be established as intensively as possible.

Lesson 15: Teams should decide what and when documentation is needed. Although agile values working software over comprehensive documentation, technical documents (especially software design such as flowcharts) should not be neglected. It is not necessary to elaborate all details, just make documents simple enough to understand [242]. Agile recommends using communication and collaboration amongst team members as a means of maintaining knowledge rather than using documentation [243]. However, at the beginning of the agile-oriented or hybrid agile-disciplined journey it is not easy to get the right people having the ability to effectively communicate and collaborate into the software projects. Hence, performing necessary documentation together with enhancing the teams' communication and collaboration skills should help to augment the higher levels of knowledge maintenance.

Lesson 16: Well preparation of appropriate technical environments (e.g., evaluating appropriate techniques and tools and conducting an up-front assessment of data quality in the source systems) at the early stage of software development can minimize the chances of software project failures. This is more or less similar to Karlström and Runeson's key findings [239]. They suggest involving developers early in the software development to quickly identify and eliminate technical issues and clearly outline possible solutions.

Lesson 17: Software development is knowledge-intensive activity [30]. Hence, the teams should assess, implement, and improve the factors that affect knowledge transfer effectiveness (e.g., motivation, sufficient communications, commitment, the ability to share, learn, and apply knowledge, and the usefulness of knowledge). The observations suggest that the more the transferred knowledge is perceived as useful, the more likely the transferred knowledge is continued to be performed and integrated into the existing software processes. Nevertheless, the transferred knowledge must be compatible with the organizational culture, e.g., standards, policies, and practices. Otherwise, it is greatly likely to be rejected.

Based on the above lessons learned, the challenges can be summarized into Table 5-4. In this table, there are eight certain challenges that need to be addressed for further improvement of software development performance. These include a lack of consistent self-discipline on backlog administration, a lack of appropriate workload allocation and awareness of their roles and responsibilities, a lack of team self-management, the need of team leaders who can make a decision and guide teams in the right direction, a lack of balanced agile and disciplined environments, a lack of intensive face-to-face communications, less-detailed documentation, and a lack of sufficient knowledge transfer.

Table 5-4. A summary of the challenges found in the case studies

Dimension	Challenge	CAT	TOT
Assessment	Software practices should be assessed with the minimum bias.	X	
Backlog administration	There were unclear statements of goals and requirements in some sprints (or iterations).	X	
	There was a conflict between business value criteria and technical criteria in requirement prioritization.	X	
	There was a conflict for breaking down work into tasks due to the different opinions between software developers.	X	
	The participants felt they had to spend more time on backlogs due to less experience.	X	X
Human resource management	Multi-roles and/or multi-projects of key team members affect effective software development. In other words, a heavy workload, a lack of commitment in terms of time and effort, or unawareness of their roles and responsibilities affect effective software development.	X	X
	There was a lack of team self-management due to long journey of traditional software development and organizational environments.	X	X
	There was the need of team leaders who can make a decision and guide them in the right direction.	X	X
	Management should transition and balance command-and-control to leadership-and-collaboration management [229-231]. In other words, management should balance agility and discipline; whilst team members should learn how to be collaborative leadership (or self-managing).	X	X
Communication management	There was a lack of intensive face-to-face communications.	X	X
Documentation	Less-detailed software design may cause problems in the future.	X	X
Development technique	Ill-preparation of appropriate technical environments (e.g., assessing appropriate techniques and tools and analyzing data quality from the source systems before the development) significantly impedes rapid software development.		X
Knowledge transfer	Sufficient knowledge transfer within teams is required for software development performance and knowledge retention.	X	X

B. The answer to the RQ5-3 "What changes are necessary to adapt the developed software process maintenance framework?"

The answer to the RQ5-3 "What changes are necessary to adapt the developed software process maintenance framework?" is summarized in Table 5-5. Owing to agile-inexperienced software development teams, the findings provide a clear picture that both teams need to make the following changes to adapt hybrid agile-disciplined processes.

Table 5-5. A summary of the changes necessary to adapt the developed software process maintenance framework

Dimension	Necessary Changes	CAT	TOT
Software process	The project goals, objectives, and roadmap must be clearly explained to all team members to ensure that all team members are going in the same direction.	X	X
	The teams had to change from using only the ongoing project management plans throughout the software projects to using both iterative and ongoing project management plans.	X	X
	Users and all team members had to work together from the planning to iteration/project closure through continuous communications and lots of meetings.	X	X
	The teams had to change from responding changes immediately to freezing requirements during iterations.	X	X
	The teams had to change from validating and verifying work at the end of either long release cycles or the software projects to iteratively test, review work, and/or collect lessons learned in short-time iterations.	X	X
Knowledge transfer	Not only relying on explicit knowledge in the project documents, but also shared-values environments need to be cultivated.	X	X

D. The answer to the RQ5-4 "How do practitioners transfer new knowledge into their existing software processes?"

The answer to the RQ5-4 "How do practitioners transfer new knowledge into their existing software processes?" is summarized as follows. A knowledge transfer process began with either (1) the authors' decision to transfer new knowledge (i.e., the framework) to the teams and the transfer plans or (2) any events leading to the decision to transfer amongst team members (e.g., the need to describe user requirements). During transferring new knowledge (e.g., the framework and software-development-related knowledge), the teams considered the following four factors to make a decision to use new knowledge. They were the knowledge's usefulness and ease of use, suitability with the organizational or team cultures, and compatibility with the existing software processes. Once all of these factors were satisfied by all team members, the transferred knowledge was used. Otherwise, the transferred knowledge was more likely to be rejected. Some transferred knowledge was directly used; whilst some was tailored to fit into their software development environments. For instance, the process of discussing three questions (i.e., "What did you do yesterday?", "What will you do today?", and "What impediments are in your way?") in daily meetings was tailored in both CAT and TOT teams. Both teams performed the three questions uncomfortably at first and also felt incompatible with their team cultures. Hence, they decided to apply the first two questions sometimes but only the last question for every daily meeting. In contrast, although the Scrum

master in the TOT team agreed that key team members (e.g., a product owner and a Scrum master) directly involved in the software project should have full authority for rapidly making decisions as suggested by the SDM model, this process was not approved by top management (or a project sponsor) due to the unsuitability with the organizational culture. This process was thus rejected. Once the expected outcomes (e.g., work performance and work satisfaction) from using the transferred knowledge were satisfied, the transferred knowledge was integrated into their standard practices. Otherwise, it was either re-tailored until being to solve their occurring problems or meeting their objectives, or continuously used until achieving the expected outcomes. For instance, the process of freezing requirements during the sprint was perceived as satisfactory and being able to solve the existing problems in the CAT team, this process was integrated into their existing practices.

During the transfer process, the findings reveal that team members' motivation, absorptive capacity, credibility (i.e. trust and reputation), capability (i.e., the knowledge source's reservoir of knowledge), communication frequency, good relationships between team members, and key stakeholder commitment affects the knowledge transfer success. For instance, the CAT team faced difficulties to perform retrospective meetings, e.g., a difficulty to recognize which software process is working or is not working due to less experience or a lack of absorptive capacity. Their motivation to perform this software process was likely to be decreased. The product owners in both teams were expert in their product areas and had business, managerial, and/or technical skills and expertise. As observed they took the knowledge the teams should have into account and were willing to share such knowledge with their teams. Team members were thus able to increase their software development performance. Besides, software developers in both teams largely shared technical knowledge to each other, especially via face-to-face conversations. They also established other communication channels (e.g., mobiles, emails, instant messaging, and e-conferencing) to exchange knowledge or get feedback when the team members were remote or worked in different sites. This shows that source's credibility and capacity, good relationships, and extensive communications impact the successful knowledge transfer. Furthermore, a lack of commitment in terms of time of the Scrum master led to non-application of sprint retrospectives in the TOT team. The more these factors exist in the teams; it is more likely to gain knowledge transfer effectiveness. This suggests that the participants should continuously assess, implement, and improve these factors in order to achieve successful knowledge transfer.

Considering the knowledge transfer process of the teams, it is very similar to Szulanski's knowledge transfer mechanism [103]. In Szulanski's model, a knowledge transfer process flows through four stages. First, Initiation begins with all events leading to the decision to transfer. Second, Implementation begins with the decision to transfer. Third, Ramp-up begins when the recipient starts using the transferred knowledge. Fourth, Integration begins after the recipient achieves satisfactory outcomes. Giving a clearer picture, the knowledge transfer process of the teams can be mapped with the four stages in Szulanski's model and summarized in Figure 5-2.

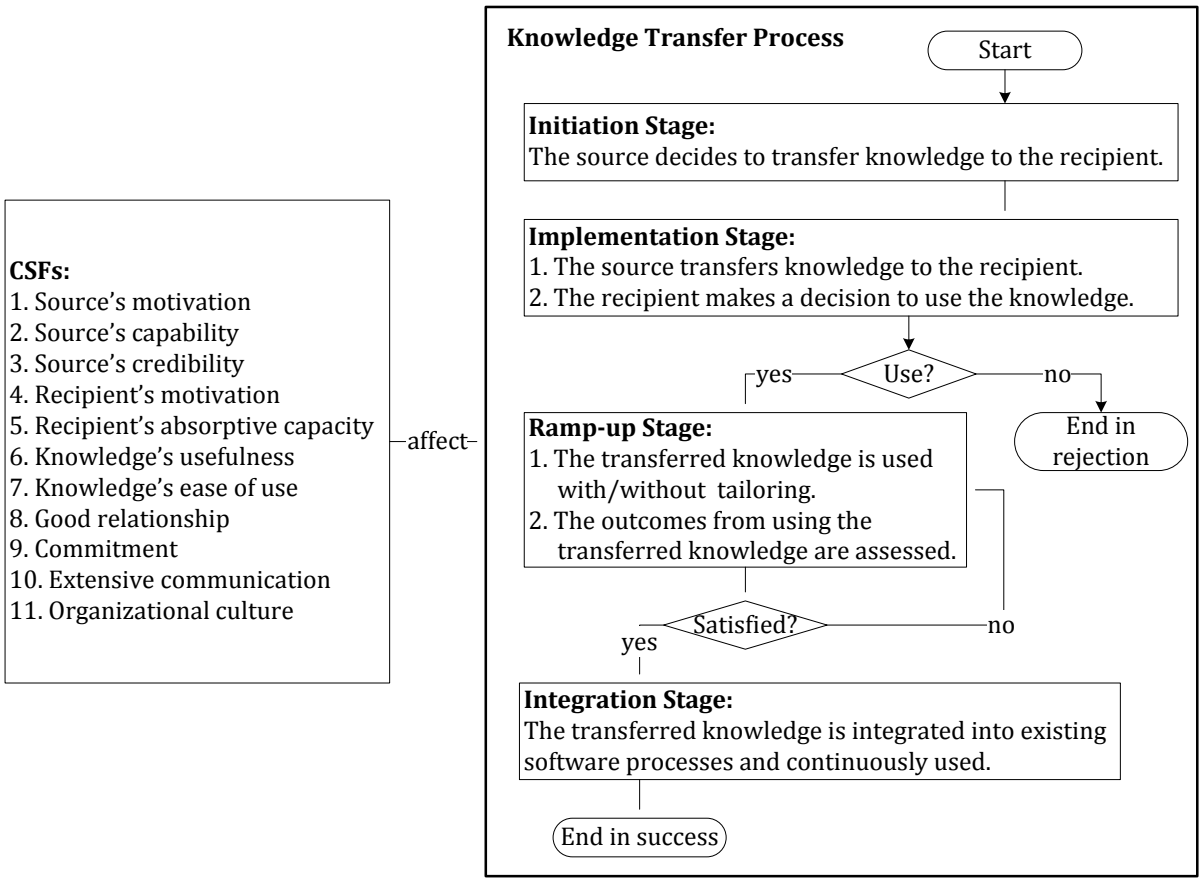


Figure 5-2. The participants' knowledge transfer process

Nevertheless, how to successfully organize knowledge transfer still remains a challenge for the organizations. Hence, these findings (i.e., the participants' knowledge transfer process and the identified factors) are used to design and develop the proposed knowledge transfer framework.

E. The answer to the RQ5-5 "What is the developed software process maintenance framework perceived usefulness and ease of use?"

The answer to the RQ5-5 "What is the developed software process maintenance framework perceived usefulness and ease of use?" is summarized as follows. Based on the questionnaire findings, the framework is perceived as useful and easy to use. Regarding the perceived usefulness, the respondents significantly perceived the improvement of work performance, work effectiveness, team productivity, software process quality, software product quality, and easiness to work. Regarding the perceived ease of use, the framework considerably perceived as clear and understandable, easy to become skilful, easy to use, and easy to remember how to perform tasks. Moreover, the respondents substantially portend to continue to use the framework. This supports that the framework were perceived as usable and acceptable.

F. The answer to the RQ5-6 "What are the requirements for successful adaptation of the software process maintenance framework?"

This section presents the requirements for successful adaptation of the developed software process maintenance framework that answer the RQ5-6 "What are the requirements for successful adaptation of the software process maintenance framework?". The requirements are summarized into three categories as follows.

The first category contains the requirements concerning the organizational context. First of all, management should define and communicate the needs for the change and how to accomplish the change successfully with people who are involved [244]. Those people should be dedicated to training of the framework and related knowledge and should have motivation to make positive changes in their behaviors and incorporate new skills and knowledge into their own knowledge packages [245]. Second, management should balance discipline and agility to fit their organizational and team cultures by continuous inspection and adaption of the software processes. Third, management should get team members to self-manage and self-discipline with collaboration [232]. Self-management is key agile characteristic that helps self-managed teams make important decisions, deal with various situations, and overcome challenges that arose [246, 247]. Collaboration is the working together to make a decision or deliver a work product. Imposing software development on non-collaboration is likely to fail [246]. Management should hence cultivate all key stakeholders (e.g., users and team members) into the intense collaborative culture for rapid decision making and improved ability to cope with ambiguity.

The second category contains the requirements concerning the software process context. Although the framework can act as guidance on the "what" and "how" to improve and implement software processes; it is crucial that teams must also iteratively inspect and adapt the integrated project management and software development processes to fit into any circumstances. For instance, to comfort their team cultures, the teams tailored daily meetings by adapting two daily questions of "What did you do yesterday?" and "What will you do today?" approximately few days a week. This requirement is similar to Schatz and Abdelshafi's suggestion [71] as they said "building software is a continuous learning process". However, in order to appropriately adapt the integrated project management and software development processes, it requires adaptive people who understand both traditional and agile software development approaches to be taken place in teams.

The third category contains the requirements concerning the knowledge transfer context. As mentioned, an effective transfer of knowledge and skills significantly impact ongoing development activities. Management should provide adequate time and effort and be able to train team members on relevant knowledge (e.g., business, managerial, and technical), while software developers should have motivation and willingness to especially share technical skills to others. Knowledge sources need credibility in terms of trust and reputation; whereas knowledge recipients need the ability to learn and apply the transferred knowledge. Besides, knowledge can be transferred effectively when it is perceived as useful and not too complicated by the recipients and when a good relationship between team members has taken place [72]. An interactive work environment with frequent informal communications should also be established to facilitate an effective knowledge transfer. The organizational culture should also be taken into account when exchanging or tailoring knowledge. Based on Joshi et al. [72], Table 5-6 summarizes the identified requirements into five categories (i.e., the contexts of source, recipient, knowledge, relational, and situational). The source and recipient context refer to the attributes of the knowledge source and recipients which can facilitate or

impede the process of knowledge transfer. The relational context refers to the attributes that characterize the relationship between a knowledge source and a recipient. The knowledge context refers to the nature and characterization of the type of knowledge that is being transferred. The situational context refers to the environmental characteristics surrounding the knowledge transfer process. All of the identified requirements should regularly be assessed, implemented, and improved for successful knowledge transfer.

Table 5-6. Requirements concerning the knowledge transfer context

Context	Requirement
Source	Great motivation, capability, credibility
Recipient	Great motivation, absorptive capacity
Knowledge	Usefulness, ease of use
Relational	Good relationship, commitment
Situational	Extensive communication, organizational culture

Two sets of the knowledge transfer requirements identified in Table 2-5 in Chapter 2 and those identified in Table 5-3 in this chapter are the same. This emphasizes that these identified requirements strongly need to be implemented for successful knowledge transfer. For the sound development of the proposed knowledge transfer framework, a gap analysis in the field of knowledge transfer is performed in Chapter 6. The findings of the gap analysis and the findings (i.e., the participants' knowledge transfer process and the knowledge transfer factors (or requirements) from this chapter are used to design and construct the knowledge transfer framework. The descriptions of the framework are presented in Chapter 7.

5.5 Summary

Two case studies in state-owned telecommunications companies in Thailand were carried out to check on whether the developed software process maintenance framework is practical and usable. It must be emphasized that the framework was tested without any controlled settings. Its application to real-life software development case studies indicates that the framework promises to provide the improvement of software development performance in terms of efficiency and effectiveness. Hence, it can be used as a feasible alternative to manage and develop software projects. The test was split into two phases: the first phase performed at CAT Telecom Public Company Limited (CAT) and the second phase performed at Public Company Limited (TOT). The main goal of the first phase is to provide an analysis of the application of the framework and the practitioners' process to learn, use, and integrate new knowledge (e.g., the framework and software-development-related knowledge) into their existing software process; whilst the second phase involves collecting only interesting data which offers our double check on certain factors and issues in the case studies. The findings are summarized into six perspectives as follows.

The first perspective describes how the developed software process maintenance framework can be executed efficiently and effectively in the given context. Beginning the cases with the assessment of software development maturity using the SDM model, the maturity level of the CAT team was higher than that of the TOT team. Based on the findings, we identify certain software practices under five CSFs that were efficiently and effectively

implemented in both cases (i.e., project management process; user involvement; appropriate methods, techniques, and tools; team capability; and team environment). There also were four additional CSFs that were efficiently and effectively implemented in the CAT team (i.e., management commitment, agile software engineering process, organizational environment, and reviews). Based on those practices, the findings also indicate that the framework partially conforms to approaches offering similar features (e.g., project management and software development processes, continuous software process improvement, coding standards, simple design, refactoring, and continuous integration), e.g., CMMI and XP.

The second perspective describes the challenges that impact software development, using the framework. Based on the findings, we identify eight certain challenges that need to be addressed for further improvement. These include a lack of consistent self-discipline on backlog administration, a lack of appropriate workload allocation and awareness of their roles and responsibilities, a lack of team self-management, the need of team leaders who can make a decision and guide teams in the right direction, a lack of balanced agile and disciplined environments, a lack of intensive face-to-face communications, less-detailed documentation, and a lack of sufficient knowledge transfer.

The third perspective describes the changes necessary to adapt the framework. Both teams were at the beginning of hybrid agile-disciplined journey. It particularly requires six certain practices that both teams needed to make changes for adapting the framework. These include clearly explaining project goals, objectives, and roadmaps to all team members; using both iterative and ongoing project management plans throughout the software projects; working together between users and team members from iterative planning to closure through continuous communications; freezing requirements during iterations; testing, reviewing work, and collecting lessons learned in short-time iterations; and cultivating shared-value environments through sufficient knowledge transfer.

The fourth perspective describes how practitioners transferred new knowledge into their existing processes. A knowledge transfer process began with either (1) the authors' decision to transfer new knowledge (i.e., the framework) to the teams and the transfer plans or (2) any events leading to the decision to transfer amongst team members (e.g., the need to describe user requirements). During transferring new knowledge (e.g., the framework and software-development-related knowledge) from the authors to the teams or amongst team members, the participants considered four factors (i.e., the knowledge's usefulness and ease of use, suitability with the organizational or team cultures, and compatibility with the existing software processes) to decide whether or not to use new knowledge. Once all of these factors were satisfied by all team members, the transferred knowledge was used. Otherwise, the transferred knowledge was more likely to be rejected. Some transferred knowledge was directly used; whilst some was tailored to fit into their software development environments. Once the expected outcomes (e.g., work performance and work satisfaction) from using the transferred knowledge were satisfied, the transferred knowledge was integrated into their standard practices. Otherwise, it was either re-tailored until being to solve their occurring problems or meeting their objectives, or continuously used until achieving the expected outcomes. During the transfer process, the findings reveal that team members' motivation, absorptive capacity, credibility, capability or the knowledge source's reservoir of knowledge, communication frequency, good relationships between team members, and key stakeholder commitment significantly affects the knowledge transfer success. The more the quality of these factors exists in the teams; it is more likely to gain knowledge transfer effectiveness. This suggests that the participants should continuously assess, implement, and improve these factors in order to achieve successful knowledge transfer. Nevertheless, how to successfully

organize knowledge transfer still remains a challenge for the organizations. Based on these findings, the participants' knowledge transfer mechanism is very similar to that of Szulanski's model. Hence, we used these findings (i.e., the practitioners' knowledge transfer mechanism and the identified factors) to design and develop the proposed knowledge transfer framework which is described in Chapter 7.

The fifth perspective describes what the framework was perceived usefulness and ease of use. The teams were satisfied with the framework. It enables them to deliver frequent, tangible, and right results that significantly lead to the increased team and customer satisfaction. Regarding the framework's perceived usefulness, the teams significantly perceived the improvement of work performance, work effectiveness, team productivity, software process quality, software product quality, and easiness to work. Regarding the framework's perceived ease of use, the framework considerably perceived as clear and understandable, easy to become skilful, easy to become easy to use, and easy to remember how to perform tasks. Moreover, the respondents substantially portend to continue to use the framework. This supports that the framework were perceived as usable and acceptable.

The sixth perspective describes the requirements for successful adaptation of the framework. We have identified requirements for successful adaptation of the framework. In the organization context, the framework requires management to motivate changes, support hybrid agile and disciplined environments, and cultivate collaborative self-management. In the software process context, teams must iteratively inspect and adapt the integrated project management and software development processes to fit into any circumstances. To do so, it is important to have adaptive people who understand both traditional and agile software development approaches on teams. In the knowledge transfer context, the following factors are required to be existed in teams. Those factors includes knowledge's source motivation, capability and credibility; knowledge recipient's motivation and absorptive capacity; knowledge usefulness and ease of use; good relationships between team members, commitment; frequent communications; and (supportive) organization culture. Practitioners should continuously assess and improve these factors for successful knowledge transfer.

In summary, the findings reveal that the framework can to some extent overcome some of Scrum's weaknesses in the following knowledge areas: (1) integration management (i.e., providing configuration management and details of many types of testing), scope management (i.e., a clearer sense of product's direction), time management (i.e., improving the predictability of time estimate for the whole project, using the scope management plan and backlogs together), and technical aspects (i.e., using generic agile practices such as data quality technique, simple design, and code standard as suggested by the SDM model). Some of Scrum's weaknesses require more cultivation to overcome, e.g., commitment, collaboration, intensive communications, and knowledge sharing to build up team members' experience in real-life software projects. This emphasizes that there is a need for an effective knowledge transfer mechanism. Owing to small software projects of our case studies, this limits our ability to argue whether or not some Scrum's weaknesses (i.e., limited support for high quality assurance, large teams, outsourcing, and accurate cost estimate for the whole project) can efficiently and effectively be overcome by the framework. However, the findings reveal that the framework presents the promise to provide the improvement of software development performance in teams of efficiency (i.e., reducing rework and increasing team productivity) and effectiveness (i.e., reducing defects and increasing customer/team satisfaction).

It is necessary to state the limitations of this study. First, we did not have history documents of the participating companies' existing software projects. Secondly, we collected

only interesting data in the second phase performed at the TOT team. The main reason was to double-check certain factors and issues in the case studies. Third, the project scale of the cases was relatively small in terms of the team size and the project duration. Fourth, the participants were previously inexperienced in agile software development. Last, the focus was on only state-owned enterprises, not private companies who are leaders in the overall Thai telecommunications market. Different software development environments may give different results. These limited the generalizability of this study. However, we described the contexts of the cases and analyzed to what extent the findings are relevant or similar to the findings of other cases. This is in order to make clear to what degree the results are generalizable. As the result of the continuation of using the framework on other software projects in the participating organizations, this implies that generalizability should more or less be increased, albeit a case study naturally limits generalizability due to its specific context. Moreover, the current trend towards adopting agile methods in Thailand is just at the initial stages. This implies that a majority of companies in the Thai telecommunications industry may probably still currently either use traditional software development methods or have traditional software development environments and cultures. Hence, this study may provide generable results to companies or software projects having contexts similar to the cases. Nevertheless, additional case studies are needed to increase the generalizability of this study.

For the next steps, a gap analysis in the field of knowledge transfer is performed in Chapter 6. For the sound development of the proposed knowledge transfer framework, the findings of the gap analysis and the findings (i.e., the participants' knowledge transfer process and the identified knowledge transfer factors from this chapter are used to design and construct the framework. The descriptions of the framework are presented in Chapter 7.

Chapter 6

Gap Analysis in the Field of Knowledge Transfer in Software Development

Knowledge transfer is critical for software development success and performance. It can be conceptualized in many different ways, e.g., as a communication process and a diffusion process. This chapter reviews literature on knowledge transfer in order to compare the similarities and differences. The findings reveal three interesting aspects. First, a connectionistic perspective of knowledge transfer would be the most suitable for software development. Based on communication-based knowledge transfer models, this study has adapted Szulanski's model due to four reasons: (i) it puts forward more complicated approach specific to knowledge transfer and describes the notion of internal stickiness to explore the difficulties of knowledge transfer that leads to the discovery for potential means to overcome those difficulties; (ii) it can be employed at many levels, e.g., organizational, team, and individual levels; (iii) there are a myriad of studies using Szulanski's model in terms of the transfer process or transfer stickiness as a base, which can imply that knowledge transfer effectiveness can gain from this model; and (iv) the knowledge transfer mechanism of our case study participants presented in Chapter 5 is very similar to that of Szulanski's model. Knowledge transfer can thus be viewed as a dyadic communication process between the source and the recipient engaged in software development teams through communication channels for their learning and transferring knowledge. The transfer process flows through four distinct stages which are Initiation, Implementation, Ramp-up, and Integration. Second, knowledge transfer consists of six components which are problems, antecedents (i.e., determining factors of the ease or difficulty of knowledge transfer), knowledge, mechanisms, knowledge application, and outcomes. During transferring knowledge, individual components can occur at the same or different times and more than once. In other words, they interact with others as multi-directional. Third, all reviewed studies neither put an emphasis on all of the six components nor do they clearly offer comprehensive descriptions of and relationships between those components. The ones providing guidance on how to drive knowledge transfer into action are sparse. These findings provide advice on how to design and construct the proposed knowledge transfer framework aiming at covering the six components, providing guidance for planning knowledge transfer activities, and contributing to an effective knowledge transfer amongst software development team members.

6.1 Introduction

Knowledge is information possessed by individuals through a process of reflection, enlightenment, or learning until it becomes a basis for action [41, 248]. A knowledge transfer amongst software development team members (hereafter referred to as "team members") is crucial since a software project typically consists of multiple stakeholders with diverse backgrounds and skill sets. Talents in software development teams (hereafter referred to as "teams") should continuously complement each other for better work efficiency [40]. Besides, a knowledge transfer amongst team members means that software processes (e.g.,

project management and software development processes) can be optimized for improved efficiency and effectiveness above or beyond what any individual can achieve [41]. Research on knowledge transfer in software development has been conducted within (at least) two main settings: collocated and distributed teams [40, 73, 75, 249]. However, the findings of Sapsed et al. [250] reveal that there is a very high similarity in how teams transfer knowledge, just difference in communication channels used. Consequently, this study puts an emphasis on an efficient and effective knowledge transfer amongst team members.

Knowledge transfer can be defined in many different ways. For instance, Szulanski [103] views knowledge transfer as a dyadic process in which a complex, causally ambiguous set of routines is recreated and maintained in a new setting. Argote and Ingram [251] consider knowledge transfer as "the process through which one unit (e.g., group, department, or division) is affected by the experience of another"; whereas Darr and Kurtzber [252] argue that knowledge transfer occurs "when a contributor shares knowledge that is used by an adopter". Ko et al. [86] define knowledge transfer as "the communication of knowledge from a source so that it is learned and applied by a recipient". While different definitions of knowledge transfer are being used, different solutions for effective knowledge transfer have been proposed. Consequently, it is important to understand terms of knowledge transfer, which leads to the following research question.

RQ6-1: What are the differences in how knowledge transfer is defined in the literature and what can we learn from these differences?

Moreover, knowledge transfer itself has several components. Becker and Knudsen [253] view knowledge transfer comprising three components (i.e., antecedents, mechanisms, and outcomes), whereas Ward et al. [254] view it as consisting of five components (i.e., problem identification and communication, knowledge development and selection, analysis of context barriers and supports, knowledge transfer activities, and knowledge utilization). Martinkenaite [255] views knowledge transfer composing of three main components which are antecedents, knowledge acquisition, and outcomes. Besides, researchers argue that knowledge transfer takes place where the transfer effectiveness depends upon antecedents in the surrounding contexts of knowledge transfer. An antecedent in this study is meant to be a determining factor of the ease or difficulty of knowledge transfer [255]. For instance, antecedents in the knowledge context include causal ambiguity and unprovenness [256]. Antecedents in the source context include shortage of motivation and reliability [78]. Antecedents in the recipient context include a lack of absorptive and retentive abilities [86], and antecedents in the relational context include arduous relationship between the source and the recipient and barren organizational context [81, 84]. Knowledge transfer should be based on existing needs and problems of the recipient. Identifying a problem can lead to knowledge transfer with possible mechanisms [257]. Mechanisms can focus on knowledge transfer activities and communication channels. There are main types of knowledge transfer activities, e.g., ones focused on assessing the knowledge embeddedness, ones focused on managing the transfer process, and ones focused on transferring knowledge [258]. Besides, communication channels play as a key enable in facilitating transfer effectiveness and achieving satisfactory outcomes. Satisfactory outcomes are the result of using useful knowledge, until one is able to make decisions and solve problems effectively [259]. As how each individual component interacts with others significantly affects successful knowledge transfer as well as knowledge transfer components must be provided to ensure a clear understanding of the transfer process [253], this leads to the following research question.

RQ6-2: How does each individual knowledge transfer component interact with others?

Based on the findings on the above research questions and our review of 27 highly visible knowledge transfer studies in the field of software development (2000-2011), the answer to the following research question is then highlighted and used for designing our knowledge transfer framework which aims at providing guidance for planning knowledge transfer activities.

RQ6-3: What are the missing points in the literature on knowledge transfer in software development?

This chapter is organized as follows. The following section presents literature review on knowledge transfer epistemologies, knowledge transfer definitions, knowledge transfer models, and knowledge transfer components. This is followed by the descriptions of common knowledge transfer components and the missing points in the reviewed literature.

6.2 Literature Review

During software development, team members can learn from their experiences in order to find an effective way to create, share, apply, and retain their relevant knowledge. Knowledge transfer is an important step towards higher competencies of team members, successful software development, and eventually sustainable competitive advantages. To understand the definitions of knowledge transfer, a literature review is performed and described as follows.

6.2.1 Epistemologies of Knowledge Transfer

Epistemology is a branch of philosophy concerned with the study of knowledge [260]. Venzin et al. [261] claim that before researching any knowledge concepts, it is important to explore its epistemological roots. This is because "concepts take different forms depending on the epistemology they are based on". They distinguish three epistemologies (i.e., cognitivistic, connectionistic, and autopoietic) that lead to an amount of research on knowledge transfer. Those epistemologies have their meanings as follows:

A cognitivistic epistemology is based in the western management tradition where an organization is viewed as an information processing machine and knowledge is referred to as explicit [260, 262]. This knowledge is a fixed and representable entity (or data) that can be universally stored in databases, computers, Information Technology/Information System (IT/IS), Information and Communications Technologies (ICTs) and physical documentation [260, 261]. This in turn allows it to be easily shared within and across an organization. The cognitivistic perspective considers knowledge like data that can be "unproblematically shared from one entity to another" [40, 261]. Moreover, this knowledge is developed and managed in accordance with universal and standardized rules [260], which are not viewed as a critical factor affecting knowledge transfer under this perspective. As this knowledge has a universal characteristic of the source, the recipient, or the knowledge itself, it thus plays no role in the transfer [40].

A connectionistic epistemology considers the rules governing knowledge transfer and acquisition not being universal, but varying locally [261]. Organizations are viewed as self organized networks driven by communication. However, Senge [263] defines a team as a fundamental learning group within an organization. Hence, the rules governing the transfer are team-based and dependent on the conditions of social interactions, ties, or networks [260].

In the connectionistic perspective, knowledge is problem-solution orientated and contextual [40, 261], which leads knowledge transfer being inherently difficult due to different factors, e.g., the contextualized nature of knowledge, the need for shared understanding, and the nature of communication [40].

An autopoietic epistemology refers to tacit knowledge residing in mind, body, and social systems [261]. This knowledge is viewed as observer- and history- dependent and context sensitive. It is not directly shared, but only indirectly through individual discussions and socialization. In other words, it is developed in an autonomous manner [40, 261]. This provides belief that the central concepts of this autopoietic perspective are "the concepts of autonomy, unity, and co-evolution" [40]. Moreover, as an organization is open to the data influx and closed to the knowledge exodus, this knowledge is thus not seen as abstract and sharable [40, 260, 264]. The autopoietic perspective is hence referred to knowledge conversion or knowledge creation rather than knowledge transfer [40]. This knowledge can be converted through many strategies, e.g., socialization, externalization, combination, and internalization [262].

From these epistemological perspectives, the connectionistic perspective would be the most suitable for the area of software development. This is because software development is a sense-making process that fundamentally involves human connections (e.g., social interaction, collaboration, negotiation, and learning) [40, 265]. We consequently believe that the connectionistic epistemology serves as a basis for designing a framework for transferring knowledge in a software development setting.

6.2.2 Definitions of Knowledge Transfer

In literature, the phrases of, e.g., knowledge sharing, knowledge exchange, knowledge flow, knowledge dissemination, knowledge distribution, and organizational learning are often used as synonyms of knowledge transfer. From those phrases, many definitions of knowledge transfer are given. For instance, Szulanski [103] views knowledge transfer as a dyadic process in which a complex, causally ambiguous set of routines is recreated and maintained in a new setting. Argote and Ingram [251] consider knowledge transfer as "the process through which one unit (e.g., group, department, or division) is affected by the experience of another"; whereas Darr and Kurtzber [252] argue that knowledge transfer occurs "when a contributor shares knowledge that is used by an adopter". Ko et al. [86] define knowledge transfer as "the communication of knowledge from a source so that it is learned and applied by a recipient". While different definitions of knowledge transfer are being used, different ways for successful knowledge transfer have been proposed. As the connectionistic perspective is used for knowledge transfer in a software development setting, this study consequently defines knowledge transfer as a dyadic process sharing software-development-related knowledge from a source to a designated recipient within teams engaged in software development through various communication channels for their learning and applying knowledge.

6.2.3 Models of Knowledge Transfer

Many knowledge transfer models and frameworks have been conducted to show transfer processes, influential antecedents, roles of knowledge sources and recipients, and knowledge transfer channels. Those models and frameworks can be classified into many types, e.g., process base, antecedent base, and component base [266], which are described as follows.

6.2.3.1 Process-based Knowledge Transfer

The foundation of the transfer process can be traced back to the first communication model of Shannon and Weaver [267] in 1949 [268]. As the signaling metaphor, a message is sent from a source through a signal towards a recipient. The message is relayed through an encoder and then through noise before reaching a decoder; after that the decoder must convey the message to the recipient. This model is recognized as the mother of all communication models [269]. Later on, most communication-based models begin with a source, who then passes a message to a recipient through a linear communication channel [268, 270]. Nevertheless, knowledge transfer is complex and requires a great deal of communication and collaboration [74]. The higher the complexity of knowledge takes place, the higher the transfer is inert. Knowledge transfer should thus be deemed as a process of reconstruction rather than an action of transmission and reception [271]. Based on this view, many knowledge transfer models have been proposed. In 1995, Nevis et al. [272] proposed an organizational learning model consisting of three stages which are knowledge acquisition, knowledge sharing, and knowledge utilization. In 1996, Szulanski [103] proposed a knowledge transfer model. In the model, a transfer process follows four distinct stages, i.e., (i) Initiation, where the source distinguishes the knowledge which can meet the recipient's need; (ii) Implementation, where the source and the recipient establish their transfer-specific channel and meanwhile the source adapts the knowledge to suit the recipient's need; (iii) Ramp-up, where the recipient continually adjusts the transferred knowledge towards a satisfactory level; and (iv) Integration, where the recipient achieves satisfactory results with the transferred knowledge and gradually routinizes the knowledge as part of his/her own knowledge packages. Later, Inkpen and Dinur [273] performed a more extensive analysis based on Szulanski's model and found that a knowledge transfer process follows four stages, i.e., (i) Initiation, where knowledge to be transferred is recognized; (ii) Adaptation, where knowledge is changed at the source location to the recipient's perceived needs; (iii) Translation, where knowledge alterations occur at the recipient unit as part of the general problem-solving process of adaptation to new context; and (iv) Implementation, where knowledge is institutionalized into the recipient's knowledge package. However, this study considers Szulanski's model due to four reasons. First, it puts forward more complicated approach specific to knowledge transfer. It also describes the notion of internal stickiness to explore the difficulties of knowledge transfer that leads to the discovery for potential means to overcome those difficulties. Second, even though it is originally employed at an organizational level, it can be adapted at team or individual levels [274]. Third, there are a myriad of studies using Szulanski's model in terms of the transfer process or transfer stickiness (i.e., impediments to the transfer of best practices), e.g., [86, 92, 256, 273, 275-279] (1998-2010). This implies that knowledge transfer efficiency and effectiveness can gain from this model. Forth, it is even more important that the knowledge transfer mechanism of our case study participants presented in Chapter 5 is very similar to that of Szulanski's model.

6.2.3.2 Antecedent-based Knowledge Transfer

Antecedents surrounding the knowledge transfer process notably impact the degree of knowledge transfer efficiency and effectiveness. A significant number of studies have been conducted for a deeper understanding of the antecedents that enable or impede the ability of either the source or the recipient to share and learn from knowledge transfer interactions within their software development surroundings [73, 78, 81, 84, 86, 92, 256]. According to Joshi et al. [72], antecedents can be classified into many contexts, e.g., source, recipient, knowledge, relational, and situational contexts. The source and recipient contexts refer to the attributes of the source and the recipient which can facilitate or hinder the transfer process, e.g., motivation, capability, credibility, retentive ability, and absorptive capacity. The knowledge context refers to the nature and characterization of the knowledge being transferred, e.g., causal ambiguity and knowledge unprovenness. The relational context refers to the attributes that characterize the relationship between the source and the recipient, e.g., arduous relationship, team culture, and commitment. The situational context refers to the environmental characteristics surrounding the knowledge transfer process, e.g., extent of communication and organizational culture. This shows that measuring antecedents surrounding the transfer contexts is important to point out strengths and weaknesses in the transfer process that needs reinforcements or improvements for successful knowledge transfer.

6.2.3.3 Component-based Knowledge Transfer

Knowledge transfer itself has several components that lead to various aspects in designing component-based knowledge transfer models and frameworks. Albino et al. [280] describe four components influencing knowledge transfer, which is similar to Duan et al.'s framework [266]. These components are actors involved in the knowledge transfer process, context where the interaction takes place, knowledge content transferred between actors, and media by which the transfer is carried out. Becker and Knudsen [253] argue that a definition must include antecedents, mechanisms, and outcomes of a particular thing. Hence, they view a knowledge transfer process composing of those three components that should be provided to ensure a comprehensive understanding of the transfer process. Especially, the outcomes should be explicitly stated. Martinkenaite [255] proposes an integrative framework by giving a mediating role of knowledge acquisition in the relationship between antecedents and outcomes of knowledge transfer. Antecedents including knowledge attributes, organizational attributes, and inter-organizational dynamics are considered as transfer inputs. Knowledge acquisition including type, extent, and nature of new knowledge learned is considered as transfer outputs. Last, performance results in terms of financial, product, market, and strategic performance are considered as transfer outcomes. Ward et al. [254] propose a knowledge transfer framework consisting of five components (i.e., problem identification and communication, knowledge development and selection, analysis of context barriers and supports, knowledge transfer activities, and knowledge utilization). Identifying a problem leads to knowledge transfer with possible transfer solutions and activities. An evaluation of context barriers and enablers might lead to the selection of appropriate knowledge. The utilization of knowledge transfer activities might lead to a new consideration of the underlying problem or the identification of new problems. These components are connected through a multi-directional set of interactions. Similar to these multi-directional relationships, many studies describe how problems are associated with the transfer process [31, 84, 91, 103, 265, 274, 281, 282], how different kinds of knowledge play a role in a project life cycle [29,

84, 274], how knowledge-related activities or mechanisms influence knowledge outcomes [41, 81, 92, 249, 265, 274], and how important to use the transferred knowledge [29, 103, 249, 274, 275, 282, 283]. This supports the statement of D.L. Chu, 1995 "Knowledge without action is useless; action without knowledge is dangerous." Consequently, we consider that the transfer process should consist of six components which are problems, antecedents, knowledge, mechanisms, knowledge application, and outcome.

6.2.4 Lessons Learned

Answering the RQ6-1 "What are the differences in how knowledge transfer is defined in the literature and what can we learn from these differences?", knowledge transfer definitions and concepts depend upon the epistemologies they are based on. There are three epistemologies. A cognitivistic epistemology refers to explicit knowledge stored in IT/IS systems; whilst an autopoietic epistemology refers to knowledge conversion rather than knowledge transfer. It refers to tacit knowledge residing in the mind and social systems. A connectionistic epistemology refers to knowledge residing in human connections, which would be the most suitable for knowledge transfer in software development. Based on the connectionistic perspective, different models have been proposed which can be categorized into many types, e.g., process base, antecedent base, and component base. However, most of those models describe relationships amongst the knowledge transfer process, antecedents, and components. Therefore, this study takes the three aspects of the process, antecedents, and components into account. There are two main lessons considered as suitable for successful knowledge transfer amongst team members. First, knowledge transfer should be viewed as a communication process between the source and the recipient engaged in teams through communication channels for their learning and applying software-development-related knowledge. This knowledge transfer process flows through four distinct stages, i.e., Initiation, beginning with all events leading to the decision to transfer; Implementation, beginning with the decision to transfer; Ramp-up, beginning when the recipient starts using the transferred knowledge; and Integration, beginning after the recipient achieves satisfactory results. Second, knowledge transfer has six common components which are problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes. Identifying a problem can suggest teams to identify knowledge needed and define knowledge transfer activities with an appropriate mechanism. Analyzing antecedents surrounding the transfer contexts indicates teams' health in terms of knowledge transfer efficiency and effectiveness. According to Pérez-Nordtvedt et al. [284], knowledge transfer efficiency can be viewed as the amount of resources used to produce a unit of output within reasonable time and cost. To achieve efficiency, the transferred knowledge transfer should be used speedily and economically. Knowledge transfer effectiveness can be viewed as the degree to which goals of the knowledge transfer are attained. To achieve effectiveness, the transferred knowledge transfer should be perceived as useful and comprehensible. Iteratively applying knowledge may lead to knowledge embeddedness in their workspace and the identification of new problems. Besides, frequently evaluating transfer outcomes may bring about continuous improvement of knowledge transfer, team members' competencies, successful software development, and eventually sustainable competitive advantages. For more understanding, the next section presents how each of the six components of the transfer process interact with others.

6.3 Interactions of Knowledge Transfer Components

Before designing a knowledge transfer concept, it is important to understand the interactions amongst the six common components within the transfer process. The descriptions of each individual component and its interactions are presented as follows.

6.3.1 Problems

Knowledge transfer facilitates innovation which is a function of knowledge acquisition and application [285] through problem identification, generation, evaluation, and ultimate choice of the knowledge transferred [277, 286]. This shows that knowledge transfer should be based on problems or existing needs of the recipient. Identifying and formulating a problem into a clear question can form initial part of the knowledge transfer process [257]. However, potential problems can be identified properly when business goals are clearly defined [287]. Moreover, Duan et al. [266] found that knowledge transfer is only possible to occur when all partners aim at the same objectives. Hence, the identified problem should be based on business/software project goals and objectives and clarified to ensure team understandings. Identifying a problem can lead to knowledge transfer with possible solutions and transfer activities [257]. During transferring knowledge, many activities associated with the identified problem are involved over time. However, unexpected problems may also occur due to antecedents negatively influencing knowledge transfer, e.g., difficulty or complexity of knowledge being transferred, less motivation, and ineffective communications. From this point of view, we conclude that problems associates with the components of knowledge, antecedents, mechanisms, and knowledge application.

6.3.2 Antecedents

Many studies have proved that there are crucial antecedents in the contexts of source, recipient, knowledge, relational, and situational that affect knowledge transfer effectiveness [40, 72, 288]. For instance, source's and recipient's great motivation is recognized as an significant trigger for knowledge transfer and acquisition [79]. Lacking motivation, the source may be disinclined to share knowledge due to additional effort and time associated with knowledge transfer, whilst the recipient may be reluctant to acquire knowledge or may reject new knowledge due to various reasons (e.g., perceived less value of knowledge being transferred and knowledge complexity) [72, 103]. The degree of source's capability affects the degree of knowledge transfer [289]. This is because developing software requires a large amount of transferring several types of relevant knowledge [40]. Recipient's absorptive capacity is the ability to recognize the value of new knowledge, assimilate it and apply it [290]. It also is a function of the recipient's prior related knowledge, experience, and abilities. Learning new knowledge can be achieved when the knowledge is associated with what the recipient already knows. Usefulness of knowledge has been proved to be most important during the first stages (i.e., Initiation and Implementation) of the transfer process [291]. Knowledge with perceived usefulness from prior experience is less difficult to transfer and more likely to be selected to transfer [103]. Good relationship can facilitate knowledge transfer by decreasing the competitive and motivational impediments [292]. The level of emotional commitment to the personal tie affects the motivation to provide support. In other words, the stronger the personal ties, the more likely the source is willing to devote effort and time for knowledge transfer and the more easily the transfer is taken place [105, 292].

Extensive communication is critical for effective knowledge transfer [103] and knowledge creativity amongst team members [293]. Team members who communicate with each other frequently are more likely to share knowledge [292]. Moreover, frequent communication can in turn facilitate more effective communication through the development of relationship-specific heuristics. Organizational culture refers to the values, practices, and assumptions that influence the organization's members to act and behave in a particular manner. Therefore, it significantly facilitates or impedes knowledge transfer and learning [79]. This shows that unexpected problems may occur if there is a lack of supportive antecedents. Therefore, we conclude that antecedents are associated with all other components (i.e., problems, knowledge, mechanisms, knowledge application, and outcomes).

6.3.3 Knowledge

Success in producing quality software needs the presence of sufficient specialized skills and knowledge (called expertise) on teams [31]. Teams are thus demanded to know what knowledge is necessitated, how much knowledge is required, where knowledge is located, where knowledge is needed, and how much knowledge is useful or complex[257]. Recognizing when and where knowledge is required is at the heart of knowledge communication [31]. If team members cannot recognize the need and the value of the knowledge for a given software process; it may not be successfully transferred, although it may be available in teams. However, the need for certain knowledge varies as a software project progresses through its life cycle. Knowledge transfer thus demands team members to localize the knowledge around different problems [294] and customize it to fit into a given software practice [257, 294]. When knowledge proves successful, team members are likely to apply that knowledge to solve problems in the future. From this point of view, we conclude that knowledge is associated with the components of problems, antecedents, mechanisms, and knowledge application.

6.3.4 Mechanisms

Mechanisms can focus on knowledge transfer activities and communication channels (or technologies). Concerning transfer activities, there are three main types: ones focused on assessing the knowledge embeddedness, ones focused on establishing and managing the transfer process (e.g., managing influential antecedents, reducing conflict, and supporting knowledge transfer environments), and ones focused on transferring knowledge [258]. Those activities notably affect successful knowledge transfer outcomes. Concerning communication channels, the appropriateness of transfer media or communication channels depends upon many antecedents, e.g., personal relationships, knowledge types, and distance between the source and the recipient [295]. For instance, when the source and the recipient have a strong relationship and work collocated, they may mainly employ face-to-face interactions. On the other hand, when they are geographically dispersed, they may use computer mediated channels instead, e.g., videoconferencing, instant messaging, email, and knowledge management systems which themselves are based on the integration of technology and a transfer mechanism [282, 295]. Technology, particularly ICT, is also considered as a key enabler in facilitating and achieving successful knowledge transfer. However, the degree of knowledge transfer performance depends upon both adequate know-how on and extensive use of ICTs [296]. Different mechanisms fit into different situations [253], depending upon either the defined or unexpected problems as well as communication and collaboration plays

a crucial role in both knowledge transfer and software development. Consequently, suitable collaborative communication channels (or technologies) should be employed to facilitate team members to transfer, acquire, and use knowledge [297]. From this point of view, we conclude that mechanisms are associated with all other components (i.e., problems, antecedents, knowledge, knowledge application, and outcomes).

6.3.5 Knowledge Application

Useful knowledge significantly leads to its application [298]. Many researchers state that during a knowledge transfer process knowledge application is the most important activity in which the transferred knowledge is brought to bear on any problem at hand [248, 290, 298, 299]. Therefore, knowledge application can be referred to as the degree to which team members can apply knowledge to make decisions and solve problems effectively [259]. Other knowledge activities (e.g., acquisition and transformation) do not significantly lead to better work performance or any value. This is because value is created only when transferred knowledge is successfully applied when it is needed [248]. Moreover, knowledge application can be achieved through appropriate mechanisms and supportive antecedents (e.g., extensive communication, collaboration, great motivation, and absorptive capacity) [283, 298]. Many studies also suggest that while team members access and read about new knowledge (e.g., new technology, specific market conditions, or competitive developments) in order to localize and apply the knowledge, they need the context of the information or knowledge which can be learned through communications with others [40, 300, 301]. When team members access the knowledge for use on a software project, they may be able to save effort and time by continued use of the knowledge [302]. Knowledge application that enables team members to learn can result in the knowledge retention [254] and may lead to a new consideration of the underlying problem or the identification of new problems, which in turn leads to the creation of new knowledge transfer [303]. From this point of view, we conclude that knowledge application is associated with all other components (i.e., problems, antecedents, knowledge, mechanisms, and outcomes).

6.3.6 Outcomes

There are various aspects considered as outcomes of sharing, transferring, and learning knowledge. For instance, knowledge transfer performance in terms of satisfaction [103, 277] and frequency [304] is considered as outcomes of transferring knowledge. Hult et al. [305] consider cycle time as an outcome of learning in global purchasing. Slater and Narver [306] consider customer satisfaction, new product success, sales growth, and profitability as outcomes of learning in the context of marketing. The greater the benefit received from sharing knowledge, the greater the knowledge exchange [302]. However, in the area of software development, work satisfaction and work performance would be more appropriate for considering as knowledge transfer outcomes [40, 307]. The outcomes may be iteratively measured for further improvement. This improvement can be performed in many ways, e.g., establishing a reasonable incentive mechanism to enhance the source's willingness to transfer and the recipient's consciousness to acquire and use knowledge, and providing essential trainings to increase absorptive capacity [304]. From this point of view, we conclude that outcomes are associated with the components of antecedents, mechanisms, and knowledge application.

Answering the RQ6-2 "How does each individual knowledge transfer component interact with others?", the descriptions above reveal that each individual component interacts with others as multi-directional. Based on the defined problems, teams can define what knowledge is required and what mechanisms fit their software development contexts. Weak antecedents may lead to new occurring problems, whilst supportive antecedents affect transferability, the ability to use knowledge, and satisfactory outcomes. Moreover, designing and selecting transfer mechanisms depends upon required knowledge and software development environments. Suitable mechanisms lead to transfer effectiveness. Otherwise, unexpected problems may occur and expected outcomes are unlikely to be achieved. Using the knowledge can bring about knowledge retention. It may also lead to a new consideration of the underlying problem or the identification of new problems. When satisfactory outcomes are achieved, sustaining knowledge use is more likely to occur. This shows that during the process, individual components can occur at the same or different times and more than once. The component interactions are similar to those reported in Ward et al. [254].

6.4 Knowledge Transfer in Software Development

Table 6-1 presents the review of 27 highly visible studies on knowledge transfer in the software-development-related area (2000-2011) by focusing on the six components (i.e., problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes) and the purpose of each study.

Table 6-1. Previous studies on knowledge transfer in software development

Study	Problem	Antecedent	Knowledge	Mechanism	Knowledge Application	Outcome	Description
Al-Salti [79]		X					This study examines the factors cited as significant influences on the ability to transfer knowledge from the vendor to the client organizations in the information system outsourcing context. This study also discusses how those factors can encourage and improve knowledge transfer and acquisition.
Betz et al. [281]	X		X	X			This study recommends best practices of knowledge transfer in information technology offshore outsourcing projects by linking proven solutions to identified problem areas, based on a literature review and expert interviews.
Chen [308]	X	X	X	X		X	This paper examines how a task partitioning in the software project influences learning and knowledge development within the firm. This paper also suggests that internal development projects encourage synthetic learning and development of architectural and tacit knowledge. In contrast, outsourcing and joint ventures encourage analytic learning and development of component and explicit knowledge.
Chua and Pan	X		X	X	X	X	This study examines how knowledge is transferred

Study	Problem	Antecedent	Knowledge	Mechanism	Knowledge Application	Outcome	Description
[274]							for the five Information System Body of Knowledge (ISBOK) areas (i.e., technology, application domain, IS application, organizational, and IS development process knowledge). The findings show that while some areas of the ISBOK are easily grafted, some require intense vicarious and experiential learning using rich media, while others are more difficult to transfer.
Dayasindhu [85]		X		X		X	The framework is used to assess global competitiveness of organizations in the Indian software industry. In the framework, knowledge embeddedness and knowledge transfer are as key determinants of industry clusters leading to competitiveness. Industry clusters are characterized by external economies, generalized reciprocity, and flexible specialization.
Faraj and Sproull [31]	X		X			X	This study investigates the importance of expertise coordination through a cross-sectional investigation of 69 software development teams. The findings reveal that expertise coordination shows a strong relationship with team performance that remains significant over and above team input characteristics, presence of expertise, and administrative coordination.
García et al. [309]			X	X	X		This study provides a set of guidelines to develop knowledge-based Process Asset Libraries (PAL) to store software development best practices, implemented as a wiki. It shows that the learning process can be facilitated using PAL to transfer software process knowledge, while products were developed by junior software engineers with a greater degree of independence.
Gregory et al. [81]		X	X	X		X	This study analyzes managerial mechanisms and techniques to make knowledge transfer from client to vendor in IT offshore outsourcing relationships more effective. The findings reveal that facilitating motivation for knowledge transfer at the individual level is an important prerequisite for effective knowledge transfer. Once a positive attitude is present, formal management mechanisms (e.g., project reviews and communication counterparts) and informal management mechanisms (e.g., cultural competence and face-to-face meetings) can further facilitate the transfer processes. These mechanisms reinforce each other and the adequate use of both types of mechanisms in combination leads to the greatest outcomes.
Jackson and Klobas [310]		X	X			X	This study describes the development of a knowledge creation and sharing process model based upon the social constructivist theory and the integration into the model of heuristics for effective knowledge

Study	Problem	Antecedent	Knowledge	Mechanism	Knowledge Application	Outcome	Description
							construction. This intent is to help project managers create an optimal environment for the creation and maintenance of shared knowledge.
Janz and Prasarnphanich [249]	X	X	X		X	X	The study empirically examines the pattern of relationships amongst software development team contexts, knowledge-related activities, and outcomes in terms of work performance and satisfaction. The findings reveal that team contexts positively influenced knowledge-related activities which in turn positively influenced their outcomes.
Joshi et al.[40]		X					Drawing on the connectionistic epistemology and the communications-based resource on knowledge transfer, the model suggests that source's capability, credibility, and communication plays a vital role in determining the extent of knowledge transferred to recipients.
Ko et al. [86]		X					The model posits that knowledge transfer is influenced by knowledge-related factors (i.e., absorptive capacity, shared understanding, and arduous relationship), motivational factors (i.e., source's and recipient's intrinsic and extrinsic motivation), and communication-related factors (i.e., communication encoding and decoding competence and source credibility).
Kotlarsky and Oshri [75]		X		X		X	This study shows that human-related issues in the form of social ties (e.g., rapport and trust) and knowledge sharing significantly contribute to successful collaboration in globally distributed information system development teams. Transactive memory is defined as the set of knowledge possessed by group members coupled with an awareness of who knows what.
Oshri et al. [311]	X		X	X	X		This paper explores the role of transactive memory in enabling knowledge transfer between globally distributed teams. This paper also describes the knowledge transfer between on-site and offshore teams through encoding, storing, and retrieving processes.
Roberts et al. [312]				X			This paper examines the prescribed versus actual use of external consultants, universities, and vendors as knowledge links during the implementation of systems development methodologies. Knowledge links are valued for their expertise and experience in systems development methodologies. However, while knowledge-related activities can provide an organization with faster diffusion of the new methodology through organizational learning, using knowledge links does not guarantee successful systems development methodology implementation. Barriers to knowledge transfer must be recognized

Study	Problem	Antecedent	Knowledge	Mechanism	Knowledge Application	Outcome	Description
							and overcome for enhancing cooperation.
Sandhawalia and Dalcher [29]	X		X	X	X		The framework mobilizes and integrates both tacit and explicit knowledge, and facilitates the flow of common knowledge to address unstructured situations in software projects and ensure that that the right knowledge is available to the right person at the right time during the software development effort. The framework also provides a better understanding of the interactions and relationship between software development, project management, and knowledge management processes.
Sarker [73]		X					The framework consists of source's capability, credibility, communication, and culture that significantly affect knowledge transfer. The findings of its examination in the context of both cross-cultural distributed and local teams support the role of credibility and communication on knowledge transfer. Besides, culture of the source did affect knowledge transfer in the distributed teams.
Scott and Sarker [283]		X		X	X		This study shows that a channel characteristic (i.e., symbol sets) and motivation to learn have a positive effect on knowledge possessed and knowledge applied, whilst absorptive capacity influences only knowledge possessed. The findings also reveal that knowledge application is much different from knowledge possession. It is important for an individual to possess relevant knowledge and to apply it seamlessly in other contexts for complete knowledge internalization.
Slaughter and Kirsch [41]		X		X		X	The study posits how the composition and intensity of knowledge transfer mechanism portfolios affect performance improvement. The findings reveal that a more intense portfolio of knowledge transfer mechanisms is utilized when the source and the recipient are proximate, are in a hierarchical relationship, or work in different units.
Soini [282]	X			X	X		A case study in this paper deals with software development measurement and related knowledge collection, distribution and utilization in practice, using the developed information system which enables organizations to control and improve their software development process and product quality.
Steen [313]	X		X		X	X	Software product quality is related to interpretations and understanding in practice and on practical knowledge. Based on a qualitative study of practicing software developers' understanding of the concept of quality and quality assessment, the results show why quality resists definition and why experience-based practical knowledge is important.
Timbrell et al.		X					This study discusses impediments to knowledge

Study	Problem	Antecedent	Knowledge	Mechanism	Knowledge Application	Outcome	Description
[256]							transfer within enterprise system contexts, compared to Szulanski's ranked determinants in each stage of a transfer process.
Upadhyaya and Krishna [74]		X		X			The proposed model is used to identify team level antecedents of knowledge sharing and how effective of transfer mechanisms in a distributed work context. The key contribution of this study is to view the knowledge sharing process in teams with respect to different dimensions of distribution index (i.e., time zone, site, isolation, and imbalance) and relational attributes of the team.
Volkoff et al. [314]		X		X			This study identifies critical knowledge transfer barriers and empirically uncovers two complementary knowledge transfer mechanisms (i.e., an intermediate community of practice and a bridge structure) that are effective for addressing the knowledge transfer barriers related to a lack of common practices and purposes in the enterprise systems context.
Wang et al. [91]	X	X				X	The model explains the roles played by the client through absorptive capacity and the consultant through competence. The findings confirm that transfer is improved with higher levels of capacity and competence, while the transfer process leads to a better fit between enterprise resource planning systems and organizational processes.
Xu and Ma [92]		X		X		X	This model posits that knowledge transfer is significantly influenced by the knowledge-, source-, recipient-, and transfer context- related aspects. The influence on knowledge transfer from the source's willingness to transfer and the recipient's willingness to accept knowledge was fully mediated by transfer activities, while the influence on knowledge transfer from the recipient's ability to absorb knowledge was only partially mediated by transfer activities. The influence on knowledge transfer from the communication capability was fully mediated by arduous relationship.
Yun [84]	X	X	X			X	This study presents the body of knowledge (i.e., domain, technical, process, and culture knowledge) transferred between clients and vendors in information system development projects. The results reveal that in different stage of project life cycle, the transfer intensity of every kind of knowledge is different. This study also proposes key factors (e.g., firm size, process maturity, knowledge overlap, absorptive capacity, and culture fit) that impact the efficiency and effectiveness of knowledge transfer in software projects.

Answering the RQ6-3 "What are the missing points in the literature on knowledge transfer in software development?", the results in Table 6-1 reveals that all of the reviewed studies explaining all or part of the transfer process neither put an emphasis on all of the six components nor do they offer comprehensive descriptions of and relationships between those components. The ones providing guidance on how to put knowledge transfer into action are also sparse. Consequently, there is a need to build a knowledge transfer framework aiming at (1) covering the six components, (2) providing guidance for planning knowledge transfer activities, and (3) contributing to effective knowledge transfer amongst team members. For the next steps, the knowledge transfer framework is constructed based on the findings of the case studies in Chapter 5 and the findings of the literature review in this chapter. The descriptions of the framework are presented in Chapter 7.

6.5 Limitations

It is important to highlight potential limitations of this review. First, not all antecedents and components of knowledge transfer are discussed. Instead, the focus is primarily on the antecedents identified in Chapter 5 and the components which are commonly addressed in the majority of the reviewed literature and also compatible with our findings in Chapter 5, regarding the knowledge transfer mechanism of our case study participants. Second, we have conducted a review of the literature eliciting work from 69 different authors (including grey literature, e.g., working papers and technical papers, and some secondary studies where we used the reference in the primary study to lead to another study) in total and 27 different authors in the particular boundary of knowledge transfer in the software-development-related field, published in the last decade ranging between 2002 to 2012). The overall objective of this review is to capture the similarities in the field and the current gaps in the particular boundary, and to identify needs and opportunities for design and develop our knowledge transfer framework. As this review includes grey literature, the publication bias can be ameliorated to some extent. Whilst all of the selected papers are relevant to the objective of this review, the accuracy bias should somewhat be reduced. However, we note that with the increasing number of works in this field we cannot guarantee to have captured all the material in this field. This limits generalizability of the results, as some relevant papers within this review boundary may be missed.

6.6 Summary

Sustainable team competencies, successful software development, and competitive advantages require a high degree of knowledge transfer. However, how to achieve effective knowledge transfer still remains a challenge. The starting point of this study to design a knowledge transfer framework which aims at providing guidance for knowledge transfer activities is to understand terms of knowledge transfer, capture its components and its component interactions, and fulfill gaps in the literature on knowledge transfer in software development. The findings reveal that the connectionistic epistemology which refers to knowledge residing in human connections is in this study considered the most suitable for software development. Based on the connectionistic perspective, knowledge transfer should be viewed as a communication process between the source and the recipient engaged in teams through communication channels for their learning and applying software-development-related knowledge. Concerning the transfer process, we deem Szulanski's model due to four

reasons. First, it puts forward more complicated approach to knowledge transfer and describes the concept of stickiness to explore the difficulty of knowledge transfer. Second, it can be employed at many levels, e.g., organizational level, team, and individual levels. Third, there are many studies using Szulanski's model as a base. This implies that knowledge transfer effectiveness can gain from this model. Forth, it is even more important that the knowledge transfer mechanism of our case study participants presented in Chapter 5 is very similar to that of Szulanski's model. In this model, the transfer process flows through four distinct stages. They are Initiation, beginning with all events leading to the decision to transfer; Implementation, beginning with the decision to transfer; Ramp-up, beginning when the recipient starts using the transferred knowledge; and Integration, beginning after the recipient achieves satisfactory results. The transfer process should consist of six common components which are problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes. Problem identification should be based on business/software project goals and objectives and recipient's needs. After defining potential problems, teams can define what knowledge is required and what mechanisms fit their software development contexts. Weak antecedents lead to new occurring problems, whereas supportive antecedents affect transferability, the ability to use knowledge, and satisfactory outcomes. Moreover, designing and selecting transfer mechanisms depends upon required knowledge and software development environments. Suitable mechanisms lead to transfer effectiveness; otherwise, unexpected problems may occur and expected outcomes are unlikely to be achieved. In addition, using knowledge can bring about knowledge retention. It may also lead to a new consideration of the underlying problem or the identification of new problems. When satisfactory outcomes are achieved, sustaining knowledge use is more likely to occur. This shows that these components are connected with others through a multi-directional set of interactions. During the transfer process, individual components can occur at the same or different times and more than once. Furthermore, the review of 27 highly visible studies on knowledge transfer in software development by focusing on the six common components and the studies' objectives reveals that all of these studies neither put an emphasis on all of the six components nor do they clearly proffer comprehensive descriptions and relationships between those components. The ones providing guidance on how to drive knowledge transfer into action are also sparse. Consequently, there is a need to build a knowledge transfer framework aiming at covering the six components, providing guidance for planning knowledge transfer activities, and contributing to knowledge transfer effectiveness. For the next steps, the knowledge transfer framework is constructed based on the findings of our case studies presented in Chapter 5 and the findings of the literature review in this chapter. The descriptions of the framework are presented in Chapter 7.

Chapter 7

The Knowledge Transfer Framework

Software project success particularly requires efficient and effective software development and knowledge transfer processes, stakeholders' expertise and experience, and the ability to transfer, acquire, and apply knowledge to solve any development problems. Although many approaches to knowledge transfer in software development have been proposed, how to achieve software process and product quality enhancement through knowledge transfer still remains a challenge. Besides, guidance on how to drive knowledge transfer into action is also scarce. Hence, this chapter proposes a knowledge transfer framework providing guidance for planning knowledge transfer activities. The framework is based on Szulanski's model. In the framework, a knowledge transfer process has six components (i.e., problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes) and flows through four distinct stages (i.e., Initiation, Implementation, Ramp-up, and Integration). In each stage, a set of components interact with others as multi-directional and play an important role depending on each stage's functionality. For a deeper understanding of the transfer process, the comprehensive descriptions of the six components are presented. Under each component, a list of activities is designed. Under each activity, a list of key questions that should be considered is suggested. Under each stage, a flow of relevant activities is also illustrated. For a better understanding of the proposed knowledge transfer framework, a demonstration on how to apply it in real-life software projects is presented.

7.1 Introduction

Knowledge transfer and its application can significantly contribute to software project success. A software project is characterized by frequent changes and its implementation requires effective activities, stakeholders' expertise and experience, and the ability to transfer, acquire, and apply knowledge to problems occurring during software development [29]. Without using the existing knowledge (i.e., implemented software processes, experience, and knowledge gained during prior software development), software development team members (hereafter referred to as "team members") have to create new solutions to every occurring problem. Transferring and applying new knowledge is crucial for creating innovative software development and competitive software products. This supports the fact that software development is a knowledge-intensive activity [30]. Moreover, success in producing quality software demands the presence of sufficient knowledge on teams [31]. A software project therefore requires effective knowledge transfer to ensure that the software project will not get a hard landing.

Knowledge transfer concepts take different forms depending on the epistemology they are based on. The epistemological roots hence need to be explored. There are three distinct epistemologies: cognitivistic, connectionistic, and autopoietic [261]. A cognitivistic perspective views knowledge as explicit universally stored in databases, computers, and physical documentations [260, 261]. It deems knowledge like data that is unproblematically

shared between entities [40, 261] with universal rules [260]. This knowledge thus plays no role in the transfer [40]. A connectionistic perspective views knowledge as governed by local roles [261]. The roles are team-based and reliant on the conditions of social interactions, ties, and networks [260]. This knowledge is problem-solution oriented, which leads knowledge transfer being complex due to different antecedents, e.g., the nature of knowledge and communication [40, 261]. An autopoietic perspective views knowledge as tacit which is only indirectly shared through individual discussions and socialization [261]. As this knowledge is not seen as abstract and sharable, the autopoietic perspective is thus referred to knowledge conversion rather than knowledge transfer [40, 260, 264]. Software development is a sense-making process that basically involves human communications [40, 265]. Consequently, the connectionistic perspective would be the most suitable for knowledge transfer in software development.

Based on the connectionistic perspective, knowledge transfer in software development can be viewed as a communication process between the source and the recipient engaged in software development teams (hereafter referred to as "teams") through communication channels for their learning and applying software-development-related knowledge. There are several models of knowledge transfer in the area of software development. However, most reviewed studies place an emphasis on investigating influential antecedents that affect knowledge transfer efficiency and effectiveness [40, 73, 79, 84, 86, 92] and software quality and productivity [265]. Some explore knowledge transfer mechanisms that either facilitate the flow of common knowledge to illustrate unstructured situations, address the knowledge transfer barriers, or affect performance improvement in software projects [29, 41, 314]. Although the reviewed studies explain all or part of the transfer process, ones focusing on how to transfer knowledge into action are scarce. Transferring knowledge into action appears to be a complex process which involves intricate interactions between the source and the recipient [257]. To overcome this complexity, this study needs to comprehensively understand the transfer process and then produce a knowledge transfer framework providing guidance for planning knowledge transfer activities and transferring knowledge.

Knowledge transfer itself has several components that must be provided to ensure an understanding of the transfer process [253]. The findings from the reviewed models reveal that there are six components crucial to the transfer process. They are problems, antecedents, knowledge, mechanisms (i.e., Information and Communication Technologies (ICTs) and transfer activities), knowledge application, and outcomes. Identifying a problem leads to knowledge transfer with appropriate knowledge, possible solutions, and transfer activities. The results from analyzing antecedents surrounding the transfer process contexts indicate teams' health in terms of knowledge transfer effectiveness. Iterative application of knowledge may lead to a new consideration of the underlying problems, the identification of new problems, increased absorptive capacity, and eventually knowledge embeddedness in their workspace. Frequent evaluation of the transfer outcomes may bring about transfer improvement and effectiveness. This shows that each individual component interacts with others through a multi-directional set of interactions. Moreover, knowledge transfer is complex and requires a great deal of communication and collaboration [74]. The higher the complexity of knowledge takes place, the higher the transfer is inert. This study deems Szulanski's model [103]. In this model, a transfer process flows through four distinct stages which are Initiation, Implementation, Ramp-up, and Integration.

The Initiation stage is the starting point of the transfer process. It is triggered by all events leading to the decision to transfer, e.g., the discovery of problems, valuable knowledge, and possible solutions. However, there is stickiness that makes difficulties to

initiate this stage, e.g., difficulties to recognize the opportunity for transferring knowledge, no data measurement to support planning, no adequate resources, and no inclusion of targets in the plan. To pre-empt those shortcomings, knowledge transfer activities should be strategically planned. When the planning is done, the actual activities can be then executed directly through to the second stage of the Implementation [315].

The Implementation is the subsequent stage commencing with the decision to transfer. When the Implementation takes place, the plan should be followed [315]. During this stage, resources flow between the source and the recipient. The knowledge being transferred is often tailored to suit the expected needs and to pre-empt problems experienced in the past [103]. During the transfer, there are many antecedents affecting the Implementation success, e.g., capability, absorptive capacity, and motivation. The degree of knowledge transfer significantly depends upon the source's wealth of experience, knowledge, and transferability [289]. The source' capability should therefore be enhanced for higher inclination to share knowledge. The recipient's deficiencies in absorptive capacity can lead the recipient to experience many difficulties in the transfer process, e.g., large knowledge gaps, communication difficulties, and weak relationships [316]. A lack of the recipient's motivation to adopt new knowledge and no tangible reward systems and emotional support to encourage team members can hamper the transfer process. Hence, supportive antecedents should be developed to facilitate the transfer process. Transfer activities of this stage cease after the recipient begins using the transferred knowledge.

The Ramp-up begins when the recipient starts using the transferred knowledge. The overall objective of this stage is to ramp up to work performance and satisfaction by using the transferred knowledge to solve the problem and meet the defined objectives [276, 315]. During the use, the recipient may abandon the transfer process if there are too many difficulties to use the transferred knowledge or it is unlikely to solve the problem or achieve a satisfactory outcome [276]. Those difficulties can be resulted from many circumstances, e.g., weak personal ties, insufficient support from the source during the Ramp-up stage, strong embeddedness of the old routine that leads the recipient to take time to familiarly use the transferred knowledge and later abandon its use. To minimize failure, the source should monitor the use of the transferred knowledge, e.g., by obtaining feedback and using such feedback for further improvement. After achieving satisfactory outcomes, the transfer process then flows through the Integration stage.

Last, the Integration stage begins after the recipient achieves satisfactory outcomes. Knowledge application and its integration with existing routines gradually becomes routinized [317]. This stage primarily looks at the efforts required to minimize obstacles and deal with challenges to the routinization of the transferred knowledge [316]. At this stage, the Integration activities are carried out to ensure that the recipient can use the transferred knowledge without any support from the source and can take any remedial action to improve the understanding of the transferred knowledge and integrate it into his/her practices [274]; knowledge transfer is then recognized as successful.

As the proposed knowledge transfer framework aims at providing guidance for planning knowledge transfer activities, these points of view lead us to the following research questions.

RQ7-1: How should a knowledge transfer framework be constructed?

RQ7-2: What knowledge transfer activities under each of the six knowledge transfer components (i.e., problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes) should be implemented?

RQ7-3: How do knowledge transfer activities and components play an important role in each of the four knowledge transfer stages (i.e., Initiation, Implementation, Ramp-up, and Integration)?

In the framework, the comprehensive descriptions of the six core components are presented. Under each component, a list of activities has been designed. Under each activity, a list of key questions that should be considered has also been suggested. It is also important to demonstrate the application of the framework, which leads to the following research question.

RQ7-4: How can the developed knowledge transfer framework be performed?

Owing to time limitations of this study, we could not carry out an empirical case study in real-life software projects. However, we use our previous case studies in the Thai telecommunications companies (i.e., CAT Telecom Public Company Limited and TOT Public Company Limited presented in Chapter 5) as a base for describing the application of the knowledge transfer framework. This chapter is organized as follows. The following section presents the knowledge transfer framework in two core sub-sections: six components and four stages of knowledge transfer. This is then followed by the descriptions of the application of the knowledge transfer framework.

7.2 The Knowledge Transfer Framework

Knowledge transfer in this study is viewed as a communication process between the source and the recipient engaged in teams through their communication channels for their learning and applying software-development-related knowledge. Ward et al. [257] propose a knowledge transfer model. They consider knowledge transfer as a process consisting of five crucial elements: the problem which the knowledge needs to address, the context which surrounds the knowledge sources and recipients, the knowledge to be transferred, interventions (or knowledge transfer activities), and use of the knowledge in practice. These elements have dynamic multi-directional interactions with others. This means they can occur in simultaneous or different sequences. In line with this research direction, we have extended their knowledge transfer model by adding one more component (i.e., outcomes) and modified it to fit our research purposes, based on Szulanski's model. Our knowledge transfer framework hence consists of six components which are problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes. Like Ward et al. [257], individual components have multi-directional interactions between them. Thus, we propose a knowledge transfer framework depicted in Figure 7-1. As aiming at providing guidance for planning knowledge transfer activities; in the framework, activities under each knowledge transfer components must therefore be defined. Relationships between components and flows of those activities under each knowledge transfer stages must also be introduced. The details in this section help answer the RQ7-1 "How should a knowledge transfer be constructed?".

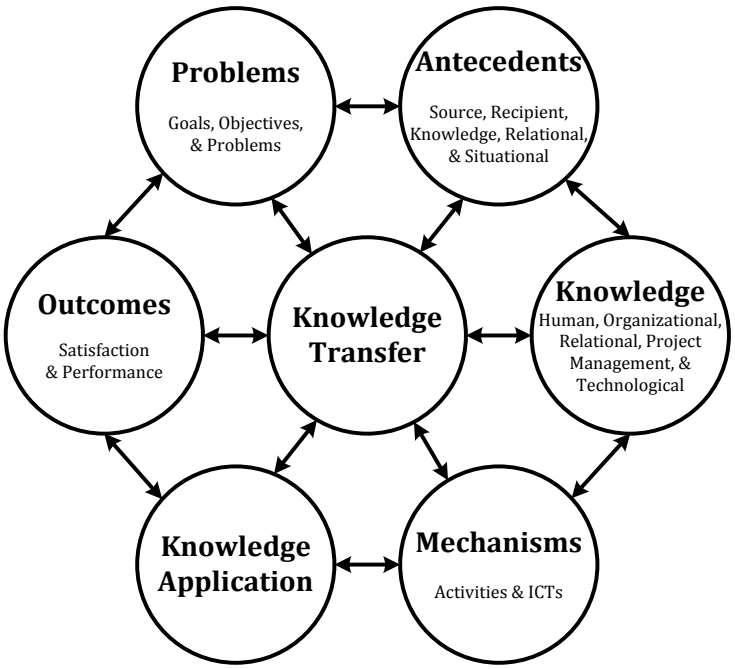


Figure 7-1. The proposed knowledge transfer framework (extended from Ward et al. [257])

7.2.1 Components of Knowledge Transfer

The details of the six knowledge transfer components (i.e., problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes) are described as follows. This helps answer the RQ7-2 "What knowledge transfer activities under each of six knowledge transfer components (i.e., problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes) should be implemented?".

7.2.1.1 Problems

Team members generally use knowledge gained from experience in previous software development domains to solve problems in the current domain. Potential problems can be identified properly when business/software project goals are clearly defined [287]. Moreover, Duan et al. [266] found that knowledge transfer is only possible when all partners aim at the same objectives. This supports that a knowledge transfer process should begin with a clear statement of business goals and objectives and a set of defined problems. However, teams should take valuable problems which could be successfully solved and yield desirable knowledge as the first priority to solve [318]. The value of a given problem depends upon potential solutions. Supporting this, Ward et al. [257] claim that identifying a problem can lead to knowledge transfer with possible means. During transferring knowledge, many activities associated with the problem are involved over time. Therefore, the problem should be clearly clarified to all team members to ensure team understandings. Focusing and reflecting on the problem can be taken place by scoping information searches and presenting relevant information to team members. The problem is then evolved. However, there is a limit for the problem to evolve when team members face a practical difficulty. The problem, especially in a complex software project, may tend to be cumulative and become serious if

not regularly reviewed [319]. From this view, Table 7-1 summarizes a set of activities and suggested questions that should be taken into consideration within the knowledge component.

Table 7-1. Activities within the problem component

Activity	Description
Defining a clear statement of goals and objectives	- *What are the business or project goals and objectives?* Teams should define business/project goals and objectives of either the overall software project or each iteration, and then clearly clarify those expectations to ensure that all team members thoroughly understand them.
Identifying a clear statement of prioritized problems	- *What are the problems that need to be solved?* Based on the defined goals and objectives, the team identifies a set of potential problems. Those problems are then clarified into clear questions to ensure that all team members thoroughly understand them. Teams may prioritize those problems by focusing on valuable ones which could be successfully solved and yield desirable knowledge.
Focusing on problems	- *How should the problems be discussed with relevant team members?* The problems must be focused and discussed with relevant team members through proper mechanisms. During knowledge transfer, more relevant knowledge may be requested to fulfill. However, focusing on the problems can scope information needed. - *What are suitable mechanisms to deal with the problems?* Teams must assess and employ potential solutions to cope with the problems. This activity should be considered with activities within the mechanism component.
Reviewing problems	- *When should the problem be reviewed?* Teams should continuously obtain feedback from team members and review the problems. - *Are there any new occurring problems?* During focusing on the existing problems, new related problems may occur. Teams thus need to iteratively inspect them in order to minimize chances of knowledge transfer failure. - *Are there any other plans to deal with the problem?* The problem can be recognized as successfully solved when team members are satisfied with the knowledge transferred. If not, they need to evaluate the problems and find new proper solutions to solve.

7.2.1.2 Antecedents

Many studies have proved that there are crucial antecedents in the contexts of source, recipient, knowledge, relational, and situational that affect knowledge transfer effectiveness [40, 72, 288]. It is important to investigate influential antecedents in our focused industry, the Thai telecommunications. According to the first investigation performed in Chapter 2, 11 influential factors affecting the successful knowledge transfer (or influential antecedents) were found. In order to find the certain antecedents, the second investigation was performed through two case studies in Chapter 5. The findings reveal the same set of those influential antecedents. Therefore, all of the 11 influential antecedents are used for this study, which are summarized in Table 7-2.

Table 7-2. Influential antecedents surrounding the knowledge transfer process

Context	Antecedents
Source	great motivation, capability, creditability
Recipient	great motivation, absorptive capacity
Knowledge	ease of knowledge access and use, usefulness of knowledge
Relational	good relationship, commitment
Situational	extensive communication, organizational culture

Source's and recipient's great motivation: Motivation is recognized as an significant trigger for knowledge transfer and acquisition [79]. Lacking motivation, the source may be disinclined to share knowledge due to additional effort and time associated with knowledge transfer, while the recipient may be reluctant to acquire knowledge or may reject new knowledge due to various reasons (e.g., perceived less value of knowledge being transferred and knowledge complexity) [72, 103]. Gold et al. [320] claim that motivation, reward, or incentive systems should be established to encourage individuals to take time to transfer, acquire, and use knowledge. With high motivation, the source will attempt to share knowledge, whereas the recipient will attempt to master and use new knowledge [92]. The greater the source and the recipient have motivation the more beneficial it will be for knowledge transfer.

Source's capability: A software development team generally consists of multiple members having different levels of skills, knowledge, and backgrounds. Developing software requires a large amount of transferring several types of relevant knowledge [40]. The degree of knowledge transfer significantly depends on the source's wealth of experience, knowledge, and transferability [289]. Accumulating experience facilitates understandings of relevant knowledge and extent communications which in turn leads to more effective knowledge transfer [40]. The source with more relevant experience will easily initiate a transfer of knowledge from itself to the recipient [103]. Besides, the degree of the source' capability affects the degree of good relationship between the source and the recipient [40, 104]. This shows that the source that has greater relevant knowledge and is perceived as capable has higher inclination to share knowledge and build positive relationships.

Source's credibility: Credibility refers to the degree in which the source is perceived as trustworthy and reputable by the recipient [40]. Many knowledge transfer studies indicate that the presence of source credibility (i.e., trust and reputation) critically influences the recipient's behavior in the knowledge transfer process [40, 103]. This is because source credibility is often used by the recipient to screen and appraise the value of the source's knowledge [40, 104]. In other words, the greater the source credibility is perceived the more likely the knowledge is perceived as valuable. Besides, the source who has high credibility will be able to transfer more knowledge and the recipient is more likely to expend efforts at assimilating and integrating knowledge transferred into his/her own knowledge package [72, 103].

Recipient's absorptive capacity: Absorptive capacity is the ability of the recipient to recognize the value of new knowledge, assimilate it and apply it [290]. It also is a function of the recipient's prior related knowledge, experience, and abilities. Learning new knowledge can be achieved when the knowledge is associated with what the recipient already knows. Consequently, the more experience, skills, and knowledge the recipient has in a given

expertise, more effectively and easily the recipient can acquire, assimilate, and apply new knowledge in that field [105].

Ease of access and use of knowledge: When the source and the recipient find the knowledge too difficult to transfer, acquire, or use, this will lead to lower knowledge transfer and decreased performance [277]. Based on Davis's study [165], the positive link between ease of use and intention to use the knowledge may lead to continuous knowledge life cycle through knowledge creation, knowledge transfer, and knowledge application. The recipient's effort expended in attempting to access and apply useful knowledge will result in obtaining relevant knowledge, when the recipient receives ease of knowledge access and use [302]. The easier it is to access or use the knowledge, the greater the recipient's effort to obtain the knowledge. To allow for ease in the way the knowledge is used, knowledge transfer should be performed via appropriate mechanisms and tools.

Usefulness of knowledge: Knowledge is transferred effectively when the source and the recipient perceive value of the knowledge. Szulanski [291] found that proof of the usefulness of knowledge is most important during the first stages (i.e., Initiation and Implementation) of the knowledge transfer process. Knowledge with perceived usefulness from prior experience is less difficult to transfer and more likely to be selected to transfer [103]. The greater the knowledge is valuable, the greater its attractiveness to the recipient and the knowledge application by the recipient [289]. However, the value of the knowledge to the recipient may depend upon the degree of the need and interest of the recipient.

Good relationship: Success in knowledge exchanges depends somewhat on the ease of communication and the intimacy of a relationship between the source and the recipient [103]. Cohesion around a relationship can facilitate knowledge transfer by decreasing the competitive and motivational impediments [292]. The level of emotional commitment to the personal tie affects the motivation to provide support. In other words, the stronger the personal ties, the more likely the source is willing to devote effort and time for knowledge transfer and the more easily the transfer is taken place [105, 292]. In contrast, arduous relationships (e.g., laborious, weak, and distant) between the source and the recipient may create additional difficulty in the transfer [103].

Commitment: Knowledge transfer is a process that requires all relevant members' commitment in terms of time, effort, and attention [90]. Commitment plays a crucial role in enabling knowledge sharing, especially in the Implementation stage of the transfer process [88, 315, 321-325]. It is also an important indicator to guarantee that the acceptance of the transferred knowledge and involvement which are key issues at the subsequent Ramp-up stage of the transfer process will be taken place [315]. Commitment should be obtained at many levels, e.g., top-management commitment to support important issues (e.g., time, effort, and resources) and team member commitment to transfer, acquire, and use knowledge as key players in the transfer process. Moreover, higher commitment indicating the feelings of attachment to relationships can establish positive ties amongst team members and greater motivation [90, 325]. This emphasizes that the higher level of commitment can exhibit better knowledge transfer performance.

Extensive communication: Knowledge transfer success increases as the number of transfer activities increases [258]. Success in performing those activities requires team collaboration and communication. Besides, software development success obliges team members to continuously communicate and learn from each other [40]. This shows the importance of communication. Supporting this, many knowledge transfer studies found that frequent communication between the source and the recipient is critical for effective

knowledge transfer [103] and knowledge creativity amongst team members [293]. Brown and Eisenhardt [326] said that learning in the contexts of complex products or tasks is increased by extensive communications amongst team members. In other words, team members who communicate with each other frequently are more likely to share knowledge [292]. Frequent communication in turn facilitates more effective communication through the development of relationship-specific heuristics. Supporting this, Joshi et al. [40] said that frequent communication leads to ardent relationships between the source and the recipient. This shows that extensive communication plays a crucial role in an effective knowledge transfer process.

Organizational culture: Organizational culture refers to the values, practices, and assumptions that significantly influence team members to perform the organization's standard practices and to act and behave in a particular manners [79]. Knowledge in the organization is created from sharing and learning in embedded routines, e.g., business process and domain-specific problem-solving activities [87]. Hence, organizational culture can facilitate and impede transferring and learning knowledge. For instance, sharing culture and management support that encourage interaction for creating, sharing, and learning knowledge can ensure successful knowledge transfer in the organization [80]. On the other hands, existing old routines of an old organization may become part of the organizational members' work behaviors that are difficult to get rid of. Although new knowledge is operational for a considerable time, the organizational members may somehow revert to the old routines [279]. Even though new knowledge is perceived as useful by team members, it is likely to be rejected if the knowledge is incompatible with the organization's standard practices as found in our case studies. This shows that the success of knowledge transfer highly depends upon the organization culture.

When problems occur and questions arise, knowledge transfer is then initiated. Based on the importance of antecedents presented above, achieving successful knowledge transfer requires teams to (1) assess influential antecedents in order to understand the strengths and weaknesses affecting knowledge transfer effectiveness and outcomes, (2) develop supportive antecedents, and (3) manage and improve the antecedents to fit into the current circumstances. These activities and their suggested questions are described in Table 7-3. However, activities for dealing with antecedents that strongly relate to other knowledge transfer components are demonstrated in the related component sections. For instance, source's credibility, usefulness of the knowledge, and ease of access and use of the knowledge strongly associates with the knowledge component, so that activities for coping with those antecedents will be illustrated in the knowledge component section.

Table 7-3. Activities within the antecedent component

Activity	Description
Assessing antecedents	- *What are antecedents that enable and impede the transfer process?* At the onset of the software project, teams may assess what antecedents facilitate and hamper the transfer process. The results will guide teams to develop a plan with proper mechanisms. During the project, teams may frequently inspect and improve those antecedents for knowledge transfer effectiveness.
Developing supportive antecedents	- *Do team members have great motivation to transfer, acquire, and use the knowledge?* Management may establish reward or incentive systems to encourage team members to take time to transfer, acquire, and use knowledge. However, Goh [327] suggests that those reward or incentive systems should be based on, e.g., successful knowledge transfer, collaboration, and teamwork, but not financial results or outcomes that are based on team competition.

Activity	Description
	- *Are team members aware of knowledge needed?* At the start of a software project, all team members should be aware what knowledge is required to get transferred in the software project. Management needs to understand what fears exist amongst team members, so that solutions to get rid of those fears could be defined properly [281]. - *Does the ability to transfer, acquire, integrate, or use the knowledge need to be increased?* Management should observe and develop the necessary abilities of both the source and the recipient that needs to be improved. - *Is there any conflict between the source and the recipient or amongst team members?* Albeit the source and the recipient are available, they have not established a positive relationship to communicate knowledge between them or vice versa. This makes the transfer effort more difficult and takes time. The transfer may have become a burden that to impede the work performance and the progress of the project [328]. In order to build good relationships, management must minimize conflict between the source and the recipient. In order to increase stronger relationships, face-to-face interaction can somewhat help [295]. - *Are all relevant members committed enough to enable knowledge transfer?* Commitment of all relevant members (including management, team members, and key stakeholders) in terms of time, effort, and attention must be obtained to enable knowledge transfer and guarantee that the acceptance of the transferred knowledge and involvement will be taken place. - *Do team members interact to each other enough to support the transfer process?* Management must cultivate communicative and collaborative environments, especially face-to-face interactions, to support the transfer process. In fact, agile software development values face-to-face conversation as the most efficient and effective method of conveying information to and within teams [232].
Managing antecedents	- *Do any antecedents need to be managed in a particular way?* Activities to develop and improve the antecedents surrounding the transfer process may be included in the project plan. Management needs to manage and adapt those activities to fit into the current situation.

7.2.1.3 Knowledge

For understanding knowledge transfer in the telecommunications industry where this study focuses on, the details of knowledge, business environments in the telecommunications industry, and knowledge in the areas of (1) Strategy, Infrastructure & Product (SIP), (2) Operations, and (3) Enterprise Management are described as follows.

Knowledge as Intangible Resources

Knowledge is basically recognized as intangible resources which can be broken into two dimensions: people dependent and people independent [329]. The people dependent resource is human knowledge as it is inseparable from its possessor. It typically refers to the knowledge required by a person that can increase productivity and the value of contribution to the organization and the software project. It also includes personal contacts, relations, and individual qualities (e.g., characteristics, experiences, and reputation). This human knowledge can be derived from team members concerning software development aspects and other stakeholders concerning business aspects. The people independent resource related to software development can be distinguished into main four categories: organizational, relational, project management, and technological [329, 330]. Organizational knowledge provides a context for team members to work in and communicate to each other. It includes its norms, guidelines, business processes, databases, organizational routines, corporate

culture, and co-operation agreements [329, 331]. Relational knowledge consists of the potential derived from the intangible resources associated with the market place. This includes reputation, brands, loyalty, long-term relationships, and distribute channels. Project management knowledge can be traditionally divided into nine knowledge bodies which include integration, scope, time, cost, quality, human resource, communication, risk, and procurement. However, implementation styles of the project management depend on software development methods used (e.g., tradition, agile, and hybrid tradition and agile). For instance, if a software project uses an agile software development method, those knowledge bodies in the agile style are required for successful adaptation. Technological knowledge is involved in two perspectives: business and software development. In the business view, this knowledge is related to the access, use, and innovation of production techniques and product technology, e.g., industrial models and drawings, trade secrets, copyrights, and patents. In the software development view, successful software development requires knowledge about software process development, programming, system and database administration, and hardware/network. All of these knowledge dimensions and categories can be summarized in the Table 7-4.

Table 7-4. Knowledge dimensions and categories

Intangible Resource	Category	Knowledge
People dependent	Human knowledge	Knowledge resided in a person, personal contacts, relation, and individual qualities (e.g., characteristics, experiences in both the business and software development views, and reputation)
People independent	Organizational knowledge	Norms, guidelines, databases, organizational routines, corporate culture, co-operation agreements
	Relational knowledge	Reputation, brands, commercial name, loyalty, long-term relationships, and distribution channels
	Project management knowledge	Integration, scope, time, cost, quality, human resource, communication, risk, and procurement
	Technological knowledge	*Business view:* Industrial models and drawings, copyrights, and patents *Software development view*: software process development, programming, system and database administration, and hardware/network

Business Environments in Telecommunications Industry

Patel [332] states that "the telecommunications surroundings can be characterized by its inherent distributive, continuous expansion in the size of network, and the particular importance of fault-tolerance requirement". These characteristics are reflected in the design of software systems and architectures [333]. Therefore, both organizations and teams have to deal with the universe of telecommunications protocols, numerous hardware platforms, and network architectures. As today's telecommunication market reaches high levels of competitive rivalry, they thus also need to keep up with the velocity at which new services, software applications are introduced while maintaining the quality and reliability levels in order to be a competitive player in the market [332]. This shows that the success of a telecommunication operator depends upon its ability to develop and deliver quality services and software applications. Increasing this ability can be achieved by effective knowledge

transfer, which in turn can help the organization to keep sustainable competitiveness and competency [333].

Understanding knowledge in telecommunications business processes is also vital to effective knowledge transfer. The enhanced Telecom Operations Map (eTOM) is a framework that focuses on the business process, the linkages between those processes, the identification of interfaces, and the use of customer, service, resource, supplier, partner and other information by multiple processes [334]. It uses hierarchical decomposition to structure the business processes and represent the whole of the enterprise environment. At the overall conceptual level, eTOM can be viewed as having three major process areas: (1) Strategy, Infrastructure & Product (SIP) covering planning and life cycle management which is associated with development and delivery, (2) Operations covering the core of operational management, and (3) Enterprise Management covering corporate or business support management. In the areas of SIP and Operations, there are seven vertical process groupings that are the end-to-end processes required to support customers and manage the business. Those groupings are Strategy & Commit, Infrastructure Lifecycle Management, and Product Lifecycle Management in the SIP area and Operations Support & Readiness, Fulfillment, Assurance, and Billing in the Operations area. There are horizontal functional process groupings that differentiate functional operations process and other types of business functional processes, e.g., service development vs. service configuration. The horizontal functional process groupings in the SIP area facilitate support and direct the work in the Operations area.

Knowledge in the Strategy, Infrastructure & Product Process Area

The Strategy, Infrastructure, & Product (SIP) process area includes processes that (1) develop strategies and commitment to them within the enterprise, (2) plan, develop, and manage the delivery and enhancement of infrastructures, products, and services, and (3) develop and manage the supply chain. Infrastructures in the eTOM framework refer to IT, application, computing, and network infrastructures required to support products and services. It also includes the operational and organizational infrastructure required to support marketing, sales, services, and supply chain processes [334]. These processes direct and enable processes within the Operations process area. From this definition, it shows that the SIP process area greatly involves in software development. Hence, human, organizational, relational, project management, and technological knowledge are all involved in the SIP process area. Human knowledge entails experiences in developing telecommunications software, services, and products. This human knowledge can be derived from team members regarding software development aspects and other stakeholders regarding various aspects of business. How to comply with organizational standards, laws, regulations, and partnership agreements is associated with organizational knowledge. Relational knowledge involves, e.g., reputations, customer loyalty, and relationships between partners that are used for planning strategies. Specialized expertise in managing a particular project in the areas of, e.g., scope, time, cost, quality, and communication is considered as project management knowledge. Many techniques and skills used to develop software, services, and products are considered as technological knowledge concerning both aspects of product and software development technologies. In the SIP process area, all intangible resources will unavoidably be used for success in designing and developing marketing strategies, software, services, and products. A successful strategy and development will lead to increasing new knowledge [333].

Knowledge in the Operations Process Area

The Operation process area is the heart of eTOM. It includes all operation processes that (1) support the customer operations and management and (2) enable direct customer operations with the customer [334]. The vertical processes of Fulfillment, Assurance, and Billing (FAB) which are recognized as front-office real-time operations provide the core of this area; whereas Operations Support & Readiness (OSR) relates to the border of back-office near real-time or offline support processes. Besides, the horizontal processes (e.g., customer relationship management, service operations, resource operations, and supplier or partner relationship management) represent functionally-related activities. As these processes are principally involved front-office operations; hence, human, organizational, relational, and technological knowledge are main knowledge in the Operation process area [333]. There is an abundance of human knowledge embedded in various aspects of the business, e.g., sales staff's experience and customer service staff's experience. How to organize operational processes, cultures, and partnerships are considered as organizational knowledge. Relational knowledge involves, e.g., relationships between customers and partners. Many innovative techniques and skills for performing and learning routinized day-to-day tasks are associated with technological knowledge in business aspects.

Knowledge in the Enterprise Management Process Area

The Enterprise Management process area includes processes required to manage enterprise-wide activities and needs [334]. These processes interface with all business management processes that (1) are necessary to support the whole of the enterprise, including those for financial management, process management, quality management, and regulatory management; (2) are responsible for setting corporate policies and strategies and providing guideline and targets, including those for strategy development; and (3) occur throughout the enterprise, including those for project management and performance assessment. Hence, human, organizational, relational, project management, and technological knowledge are all involved in the Enterprise Management process area. How to run businesses effectively is significantly associated with human knowledge in business aspects, e.g., management experience, negotiation and communication skills, and staff's credibility. A variety of organizational knowledge is inevitably used in all business process branches of this area, e.g., policies and practices, organization development, corporate management, and group enterprise management. Relational knowledge is especially required for stakeholder, employee, and external relations management. A high degree of project management knowledge is needed in almost all of business process branches of this area, e.g., financial and asset management, risk management, enterprise effectiveness management, and human resource management. Technological knowledge is essentially involved in business aspects, i.e., evaluation of potential technology or technique acquisitions.

The details of knowledge in the three business process areas emphasize that eTOM can be viewed to have two dimensions: one oriented towards the business, customers, services, and products (called the business dimension in this context) and one towards solutions, systems, software, and implementations supporting the business (called the software development dimension in this context) [334]. As this study primarily focuses on knowledge transfer in software development, the SIP process area is the most related. However, software development can be associated with other areas, e.g., in business aspects. Therefore, understandings of knowledge in all business process areas should benefit both

knowledge transfer and software development. Table 7-5 summarizes main knowledge in three business process areas. In this table, "S/W Dev." stands for software development.

Table 7-5. Knowledge in the telecommunications industry

Knowledge Type / Process Area	Human		Organizational	Relational	Project Management	Technological	
	Business	S/W Dev.				Business	S/W Dev.
Strategy, Infrastructure and Product	X	X	X	X	X	X	X
Operations	X		X	X		X	
Enterprise Management	X		X	X	X	X	

Activities Associated with the Knowledge Component

Success in producing quality software needs the presence of sufficient specialized skills and knowledge (called expertise) [31]. Teams must be able to transfer, manage, and coordinate inter-dependencies of team members' expertise effectively. From this point of view, teams are demanded (i) to know what knowledge is necessitated, how much knowledge is required, where knowledge is located, where knowledge is needed, how much knowledge is useful and complex, (ii) to tailor the required valuable knowledge to fit into a particular software practice, and to reside the transferred knowledge in team members. Based on defined problems, relevant knowledge required to manage and develop a software project can be properly defined. This knowledge can be classified into five categories (i.e., human, organizational, relational, project management, and technological). Moreover, knowing the business area of the software being developed may guide teams towards what knowledge types are needed. Teams as a knowledge-based community use the variety of expertise and competencies of team members to create a match between the defined problem and the required knowledge [319]. Knowing the location of team members' knowledge serves an important integrative and coordinative activity [31]. This activity requires knowing a variety of useful sources which include specialized documents and people who have what skills and knowledge. It is even more important that management should match required knowledge with potential sources and then locate such knowledge in teams. Targeted recipients of each of required knowledge must also be defined. For effective transfer, teams need to develop a common language for describing related work, knowledge contents, and knowledge locations. Recognizing when and where knowledge is required is at the heart of knowledge communication. If team members cannot recognize the need and the value of the knowledge for a given software process, it may not be successfully transferred although it may be available in teams. However, the need for certain knowledge varies as a software project progresses through its life cycle. The effective development of knowledge also demands team members to localize the knowledge around different problems [294]. Consequently, knowledge to be transferred must be useful and tailored to fit into a given software practice [257, 294]. Knowledge complexity is one of the significant determinants that affect the volume of knowledge dissemination and acquisition [335]. In other words, the more knowledge is perceived as easy to use, access, and learn; the more likely that knowledge

transfer will be achieved. When knowledge proves successful, team members are likely to apply that knowledge to solve problems in the future. This could in turn lead to increasing work performance, work satisfaction, and knowledge resided in team members. However, the amount of required knowledge depends upon required outcomes, based on defined goals and objectives [29]. Hence, teams should frequently measure knowledge sufficiency to ensure the presence of adequate knowledge in teams. From this view, Table 7-6 summarizes a set of activities and suggested questions that should be taken into consideration within the knowledge component.

Table 7-6. Activities within the knowledge component

Activity	**Description**
Defining knowledge	- *What knowledge is needed?* Based on the chosen problems to solve, teams need to define a list of knowledge required to manage and develop a software project.
Classifying knowledge	- *What type of knowledge is needed?* The required knowledge should be classified into main categories (e.g., human, organizational, relational, project management, and technological). This is in order to help find proper sources and mechanisms.
Assessing knowledge	- *Is the knowledge valuable and easy to be used (or accessed)?* Teams should assess the value and complexity of the knowledge. It is more likely to be acquired when the knowledge are perceived as useful and easy to use (or access). - *What is the most suitable knowledge?* Based on a knowledge assessment, teams select appropriate knowledge which is likely to be transferred successfully. - *Is the amount of knowledge being transferred enough for solving the focused problem?* Teams should frequently measure the enough amount of required knowledge to ensure being able to solve the focused problem and achieve the required outcomes.
Locating knowledge	- *Where and how can knowledge be found and located?* Management needs to assess skills and knowledge residing in each team member and then match the selected knowledge with potential internal sources. If there is a need for external human sources, management and teams need to find creditable sources that meet required competence and time for transfer and then locate such sources in teams. - *What knowledge sources are the most creditable?* Based on the required knowledge, management should seek either human sources having specialized skills and knowledge, trust, or reputation, or non-human sources being reliable and up-to-date. - *Who are the target recipients?* For given knowledge and practices, target recipients, required competence and time must be assessed and defined - *What are roles and responsibilities needed for knowledge transfer?* A clear division of roles and responsibilities to implement practices that is well defined and clarified to all team members is more likely to lead to higher degree of knowledge transfer richness and effectiveness [103, 104]. Hence, management needs to define and clarify necessary roles and responsibilities to all relevant team members, and also make awareness of their roles and responsibilities during a knowledge transfer process. This is because teams who are made aware how roles of the source are circulated amongst team members tend to include more unshared information in the discussion and in turn increase their team performance [31]. - *Where is an appropriate chain linking the source to the recipient?* Management must link the trusted source to the target recipient together with the required specific knowledge and its goals and objectives. During creating this chain, management should consider their personal relationships for analyzing how effective transfer would be and finding appropriate mechanisms. The stronger the personal relationship, the more likely the source is willing to transfer [336]. - *Is the linkage of the source and the recipient appropriate?* During the transfer

Activity	Description
	process, management should continuously observe the chain linking the source to the recipient and solve any occurring impediments. Those impediments can be caused by inappropriate ICTs employed or unsupported contexts surrounding the transfer process.
Tailoring knowledge	- *Does the knowledge need to be adapted?* Knowledge can be used directly or with modification. If there is a need for adaptation, knowledge must be tailored to fit into a particular practice.
Integrating knowledge	- *Can new knowledge be integrated into existing practices?* New knowledge may not be compatible with other existing practices. Although new knowledge is considered as valuable, it may not be integrated. - *How can new knowledge be integrated into existing practices?* If new knowledge can be integrated, management then considers how to effectively integrate it into their *existing or standard practices*.

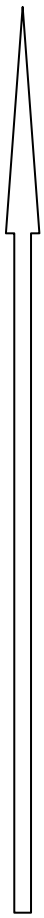
7.2.1.4 Mechanisms

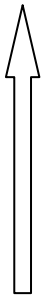
Knowledge is transferred through a variety of mechanisms [41, 337]. Mechanisms in this study focus on communication channels (or ICTs) and knowledge transfer activities. Concerning ICTs, appropriateness of ICTs is considered as a key enabler in facilitating and achieving knowledge transfer effectiveness. ICTs help increase the velocity of knowledge transfer, while reduce costs [41, 338]. The selection and use of ICTs is affected by a range of factors, e.g., personal ties, location, the volume of information richness, and knowledge types [295, 339, 340]. The stronger the relationship between the source and the recipient, the more likely the source is willing to transfer [336]. While having a strong personal tie, there is indifference in employing a mechanism [295]. On the other hand, ICTs are robustly focused to be employed when having a weak personal tie. Distance between source and recipient locations also impacts transfer success. While team members are collocated in the same location, they may primarily employ informal face-to-face interaction [282, 295]. When they are geographically dispersed, they may largely use computer mediated channels, e.g., videoconferencing, instant messaging, email, and knowledge management systems which itself is based on the integration of a technology and a transfer mechanism. This is because those media make a knowledge transfer process easier and less expensive than using face-to-face interaction. However, dispersed teams are more demanded on communication and collaboration systems to support knowledge transfer environments [338].

Effective knowledge transfer requires suitability between the ICTs and the formation processing requirements of a given task which vary with ambiguity of the task [339, 341]. Gorovaia and Windsperger [339] said that richness consists of four attributes of the ICTs: feedback capability, availability of multi-cues (e.g., voice, body, and word), a range of language, and personal focus (e.g., emotions). The higher ICTs have these attributes, the greater the volume of information richness, its capacity to handle ambiguity, and its knowledge transfer capacity. Moreover, ICTs with a relatively high volume of information richness refer to face-to-face interactions and team-based mechanisms which are recognized as the highest information richness (e.g., trainings, meetings, and workshops), whilst those with a relatively low volume of information richness include, e.g., telephone, email, written personal media (e.g., letters and fax), written formal media (e.g., documents and manuals), and numeric formal media (e.g., accounting data). Moreover, different software processes involve different knowledge types (e.g., management, development, and business) [340]. Besides, different knowledge types require different ICTs. As suitable ICTs play a key role in

how knowledge is managed; hence, teams employing suitable ICTs should encourage and facilitate team members to acquire and transfer knowledge [297]. However, the degree of knowledge transfer performance depends upon both adequate know-how on ICTs and extensive use of ICTs [296]. During transferring knowledge, management should thus take this aspect into consideration. Consequently, we summarize ICTs for knowledge transfer in a software project and examples of their purpose of use in Table 7-7.

Table 7-7. ICTs used for knowledge transfer

Media Type [339]	**Media**	**Purpose of Use**	**The degree of Information Richness [339]**
Face-to-face interactions	Informal conversation	Searching and scanning for new ideas and design concepts, consulting team members on solutions to technical problems, resolving managerial problems on the project, verifying and validating information, concepts, and ideas, and routinizing knowledge exchange [250]	Richest  Lowest
	Meeting	Consulting team members on solutions to technical problems, resolving managerial problems on the project, verifying and validating information, concepts, and ideas, and routinizing knowledge exchange [250]	
	Training	Typically transferring technical knowledge [342]	
	Sense-making and sense-giving	Transferring management and business knowledge [342]	
Telephone	Telephone and mobile	For synchronous group conference [343], consulting team members on solutions to technical problems, and verifying and validating information, concepts, and ideas [250]	
Computer mediated channels	Videoconferencing	For synchronous video conference [343], consulting team members on solutions to technical problems, and verifying and validating information, concepts, and ideas [250]	
	Instant messaging	Demonstration, speech, whiteboard, synchronous text chat [343]	
	Email	For asynchronous message [343], consulting team members on solutions to technical problems, verifying and validating information, concepts, and ideas, routinizing knowledge exchange [250]	
	Teleconferencing	Routinizing knowledge exchange [250]	
	Use of groupware	Consulting team members on solutions to technical problems, verifying and validating information, concepts, and ideas, and routinizing knowledge exchange [250]	
	Staff Intranet	Sharing database of project management, software development, standard procedures, and so on [343]	
	Project Website	Routinizing knowledge exchange [250]	
	Knowledge management	Creating, managing, and transferring knowledge	

Media Type [339]	Media	Purpose of Use	The degree of Information Richness [339]
	systems		 Lowest
Written formal media	Standard document template	Using standardized templates (e.g., for software deliverables and process phases) to capture customer knowledge and reside the knowledge in the software project [311].	
	Standard process procedure	Making things in a software project become easy to understand by following standard process procedures, (e.g., for defining user requirements) [311]	
	Formal documentation	Performing main documentation (e.g., programming codes, technical issues and business procedures) for residing project's information and maintaining the project in the future [344]	
	Informal documentation	Performing informal documentation to create learning and sharing processes, which in turn lead to new ideas and discovered weaknesses of the areas that need to be improved or changed [344]	

Regarding knowledge transfer activities, the literature identifies three types of activities: ones focused on assessing the knowledge embeddedness, ones focused on establishing and managing the knowledge transfer process, and ones focused on transferring knowledge [258]. Those activities drastically affect desired outcomes. For instance, while teams carry out managerial initiatives designed to solve any problem, a lack of assessments of embeddedness of the knowledge required to be transferred can easily result in less desired outcomes [104, 258]. Therefore, to achieve satisfactory outcomes team members have to perform a variety of activities, e.g., managing the influent antecedents, establishing the knowledge transfer process, reducing conflict, transferring knowledge, and supporting knowledge transfer environments. The greater the volume of various effective activities especially in the managerial sense the greater the recipient is more likely to assimilate and integrate the knowledge into his/her own knowledge package. From this view, Table 7-8 summarizes a set of activities and suggested questions that should be taken into consideration within the mechanism component.

Table 7-8. Activities within the mechanism components

Activity	Description
Developing a plan	- *What activities are needed to enable knowledge transfer?* Teams develop a plan for what specific activities are engaged to enable knowledge transfer, how time will be spent, what ICTs are needed, and the knowledge area. - *What suitable ICTs are needed to facilitate knowledge transfer?* Teams should assess and identify suitable ICTs specific for particular practices or activities. The selected ICTs must be clearly described to all related team member on how to use.
Managing the plan	- *How the transferred knowledge or the selected ICTs should to be managed?* Management should manage related activities as planned. However, the plan should be iteratively reviewed and re-planned to fit into the current circumstance.
Reviewing the plan	- *Are there any activities or ICTs needed to be changed?* Management should regularly inspect and adapt the activities and the ICTs to fit into the current situation.

7.2.1.5 Knowledge Application

Useful knowledge extensively leads to its application [298]. Many researchers state that during a knowledge transfer process, knowledge application is the most important activity in which the transferred knowledge is brought to bear on any problem at hand [248, 290, 298, 299]. Therefore, knowledge application can be referred to the degree to which team members can apply knowledge to make decisions and solve problems effectively [259]. Some argue that other knowledge activities, e.g., knowledge acquisition and knowledge transformation, do not significantly lead to better work performance or any value [248]. Value is created only when transferred knowledge is successfully applied when it is needed. Consequently, managing and implementing an activity of knowledge application must be carefully concentrated. Knowledge application can be achieved through extensive communication and collaboration [298]. However, teams must also explore and establish influential antecedents, especially absorptive capacity, that facilitate the recipients' knowledge possession and application [283]. This is because absorptive capacity is the ability to recognize the knowledge value, assimilate it, and apply it [290]. Many studies also suggest that while team members access and read about new knowledge (e.g., new technology, specific market conditions, or competitive developments) in order to localize and apply the knowledge, they need the context of the information or knowledge which can be learned through communications with others [40, 300, 301]. From this perspective, it shows that the greater team members receive valuable knowledge and understand its context, the more likely they will apply that knowledge for solving problems and achieving desired outcomes. When team members access the knowledge for use on a software project, they may be able to save effort and time by continued use of the knowledge [302]. Without knowledge application, they have to create solutions to and spend effort and time on every problem encounter [29]. Knowledge application that enables team members to learn can result in the knowledge retention [254] and may lead to a new consideration of the underlying problem or the identification of new problems, which in turn leads to the creation of new knowledge transfer [303]. Knowledge transfer should thus lead to changes in behaviors, practices, and policies which help secure the efficient application and retention of the knowledge transferred [345]. From this view, Table 7-9 summarizes a set of activities and suggested questions that should be taken into consideration within the knowledge application component.

Table 7-9. Activities within the knowledge application component

Activity	Description
Making use of the knowledge	- *Can the transferred knowledge be used directly in practices?* Some knowledge (e.g., standard process procedure) can be used directly, whilst some knowledge needs modification to fit into a given problem or practice. - *Will the transferred knowledge likely change team members' opinions or behaviors?* Management should observe whether the *transferred knowledge* is likely to change team members' opinions or behaviors that positively lead to the efficient use of the knowledge. If yes, the knowledge may improve team members' managerial sense, e.g., decision making. - *Will the transferred knowledge likely support or challenge practices or policies?* Management should observe whether the *transferred* knowledge is likely to support or challenge practices or policies. If challenge, solutions to deal with those challenges should be established.
Supporting use of the knowledge	- *Is there any need of support for making decisions about using the knowledge?* The source should facilitate the recipient the decision support about using the

Activity	Description
	knowledge (e.g., offering advice and opinions) when needed.
Monitoring use of the knowledge	-*Who is responsible for monitoring activities?* Persons responsible for monitoring activities can be the source or other team members. However, when responsible persons are defined, role and responsibilities must be clearly clarified to those persons. -*When and how long should monitoring activities be taken place?* The source or persons who are responsible for monitoring should allow the recipient to make a mistake and correct the mistake by himself/herself, supervises the recipient, and provides support when the recipient encounters a tough problem. This activity requires time. Consequently, teams should carefully plan activities and time to ensure that the knowledge is indeed internalized and integrated as part of the recipient's knowledge package. -*What mechanisms should be used for monitoring?* There are many mechanisms for monitoring knowledge application. However, mechanisms intended to employ should fit software development environments. For instance, observations are suitable when team members are collocated.
Auditing use of the knowledge	- *Who is responsible for auditing activities?* Persons responsible for quality auditing can be the source or the quality assurance persons. However, when responsible persons are defined, role and responsibilities must be clearly clarified to those persons. - *What practical limitations or impediments may affect knowledge application?* Once the transferred knowledge is put into use, teams should review by getting feedbacks on using the transferred knowledge (e.g., any practical limitations and impediments affecting knowledge application). If any, teams must minimize those obstacles for better and further use. - *What assessment mechanisms are suitable to gauge whether the recipient indeed absorbed the knowledge?* There are many assessment mechanisms, e.g., an oral test and a written test [274]. The assessment may focus on the recipient's deeper understanding, the absorption of the transferred knowledge, and the ability to analyze and solve problems. Meanwhile, the assessment may rate on the recipient's soft skills (e.g., eagerness to learn, willingness to share, and teamwork), so that the source can be then discussed to make the recipient aware of his/her strengths and weaknesses for further improvement.
Sustaining use of the knowledge	- *Can the* transferred *knowledge be used incrementally?* Teams should consider whether the transferred knowledge is for specific or general purposes and how it can be used incrementally. If the knowledge is very specific, there may be a limit for further use. - *Are the team members' capacities needed to build up more for making the continuous knowledge application?* Management should assess and strengthen team members' skills, competencies, or abilities that enable *the continuous knowledge application*, e.g., supporting team members to engage in the process of learning and adapting to knowledge application. - *How can the knowledge application be sustained?* Teams need to ensure that the transferred knowledge is resided in team members and can be efficiently used in the future. Sustaining the transferred knowledge can be performed in many ways, e.g., increasing access to the transferred knowledge, providing collaborative and communicative environments to facilitate knowledge application, and supporting teams on how to use the transferred knowledge by, e.g., training.

7.2.1.6 Outcomes

There are various aspects deemed as an outcome of sharing, transferring, and learning knowledge. For instance, knowledge transfer performance in terms of satisfaction [103, 277] and frequency [304] is deemed as an outcome of transferring knowledge. Hult et al. [305] deem cycle time as an outcome of learning in global purchasing, whereas Slater and Narver [306] deem customer satisfaction, new product success, sales growth, and profitability as an outcome of learning in the context of marketing. The greater the benefit received from the transferred knowledge, the greater the knowledge exchange [302]. In the area of software development, work satisfaction and work performance would be more appropriate for considering as a knowledge transfer outcome [40, 307]. Many studies show that perceived usefulness and perceived ease of use are important factors when measuring work satisfaction [346-348]. Based on Technology Acceptance Model (TAM), perceived usefulness refers to the degree to which users/team members believe that using the knowledge and the software products/services would enhance their performance. Perceived ease of use refers to the degree to which users/team members believe that using the knowledge and the software products/services would be free of effort. These two factors are also appreciably associated with work performance [165]. Work performance can be measured in terms of efficiency and effectiveness as evidence that knowledge is gained [307, 349]. Efficiency is the provision of software products/services via the most suitable use of resources. It can be measured by software quality. Effectiveness is the extent to which the software products/services are delivered in a timely, correct, and consistent manner. It is often associated with doing the right things. When work performance and satisfaction is perceived, continuous applying the transferred knowledge is more likely to occur. Otherwise, it may lead to new problems for knowledge transfer.

Improving the outcomes in terms of work performance can be managed in many ways. For instance, Wan et al. [304] suggest that concerning a knowledge transfer process, management may establish a reasonable incentive mechanism to enhance the source's willingness to transfer and the recipient's consciousness to acquire and use knowledge. Empirical studies show that the higher team members have absorptive capacity, the greater the effectiveness and performance of new product development [350-352]. Essential training may therefore be carried out for team members to increase absorptive capacity and the ability to convey knowledge, which in turn can gain work improvement. Regarding a software process, an assessment of software process improvement (e.g., Capability Maturity Model Integration (CMMI)) may be used to find out strengths and weaknesses of the implemented software process [304], so that management can identify required knowledge with suitable transfer mechanisms to remedy those weaknesses. Success in knowledge transfer should consider value gained at both sides of the source and the recipient. This is because knowledge transfer can occur in the cycle loop. This means new knowledge can be created by both sides: the source to the recipient and vice versa. The more value both sides gain, the more likely collaboration and good relationship is enhanced [298], which in turn the desired outcome could be effectively achieved. From this view, Table 7-10 summarizes a set of activities and suggested questions that should be taken into consideration within the outcome component.

Table 7-10. Activities within the outcome component

Activity	Description
Measuring outcomes	- *Is the outcome satisfied?* In order to get evidence on how effective knowledge is gained, teams need to measure an outcome on work products in terms of work satisfaction (i.e., perceived usefulness and perceived ease of use) and work performance (i.e., efficiency and effectiveness). Teams may iteratively perform an assessment and then use the results as inputs for improvement in the next iterations.
Improving outcomes	- *How can the outcome be improved?* To achieve satisfactory outcomes, work can be improved by many ways, e.g., establishing incentives to promote knowledge transfer, supporting cooperative environments, and encouraging teams to consider what they doing right and wrong [249].

7.2.2 Stages of Knowledge Transfer

As the knowledge transfer mechanism of our case study participants (presented in Chapter 5) is very similar to that of Szulanski's model, our knowledge transfer framework is thus principally based on Szulanski's model [103]. It is also based on our understandings of literature and prior empirical studies on knowledge transfer and related aspects (e.g., knowledge management and knowledge acquisition), which can serve as a guideline for planning knowledge transfer activities. Knowledge transfer is considered as a communication process that is divided into four stages (i.e., Initiation, Implementation, Ramp-up, and Integration). In the process, its components (i.e., problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes) play an important role in different stages. This section describes relationships between those components and an activity flow in each stage. This details of this section help answer the RQ7-3 "How do knowledge transfer activities play an important role in each of the four knowledge transfer stages (i.e., Initiation, Implementation, Ramp-up, and Integration)?".

7.2.2.1 Initiation

Initiation is the starting point of a knowledge transfer process. Szulanski [103] states that the Initiation is triggered by all events leading to the decision to transfer knowledge. A transfer begins when the required knowledge meets a need. The discovery of the need leads to the search for valuable knowledge, which in turn may trigger the desire to seek a potential solution. However, there is stickiness that makes difficulties to initiate this stage. This stickiness is related to the difficulties to recognize the opportunity for transferring knowledge and act on them. Recognizing this opportunity requires lots of time and effort in defining and selecting knowledge to be transferred, and then taking the initiative to decide when and how to begin the transfer process with the selected knowledge area [103, 278]. Moreover, Leseure et al. [315] found that the adoption of knowledge is not managed strategically due to various typical weaknesses, e.g., no clear goals and objectives, no data collection or measurement to support planning, no analysis of knowledge integration constraints, no adequate resources, and no inclusion of targets in the strategic plan. This shows that most activities are about management. To pre-empt those shortcomings, this study thus mainly focuses on managerial activities.

Knowledge to be transferred should be based on recipients' needs and problems [266]. Potential problems can be identified suitably when goals and objectives are clearly

defined and clarified to all team members [287]. Based on those problems, the required knowledge can be then defined and selected properly. Considering selecting knowledge to be transferred, Davies and Kochar [353] propose a framework for the selection of best practices for the improvement of manufacturing planning and control systems. It is a linear sequence of five activities commencing with (1) the objective or identification of the need to improve performance, (2) identification of best practices, (3) prioritization of identified best practices, (4) assessment of required predecessor practices, and (5) implementation of those practices. These activities are somewhat similar to those within the problem and the knowledge components. Therefore, this study suggests that this stage should be started with identifying a clear statement of goals and objectives, defining a clear set of potential problems, prioritizing those problems by focusing on their values. The prioritized problems help teams to understand what knowledge is needed to transfer in order. Moreover, the required knowledge should be classified in order to help properly define suitable sources and mechanisms. Knowing the location of team members' knowledge serves an important integrative and coordinative activity [31]. This activity requires knowing a variety of useful sources including specialized non-human sources (e.g., documents, policies, standard, source codes, and information systems) and people who have what skills and knowledge. It is important that the suitable knowledge needs to be assessed, matched it with potential sources and target recipients, and located in teams. Moreover, roles and responsibilities of the source and the recipient should be clearly defined and clarified to all related team members. In case the source is human, the source that is assigned to transfer the knowledge may put together a relevant knowledge package for his/her area of specialization, e.g., bringing all related documents up-to-date, and preparing training materials and quiz questions to be distributed at the end of the training in order to assess whether the recipient can grasp the important points of the learning session [274, 316]. However, if the selected knowledge cannot be located, the transfer may be either re-initiated or abandoned.

At this stage, many studies mainly focus on identifying negative antecedents or difficulties affecting the decision to transfer. Most of those influential difficulties are about a lack of source's credibility, a lack of recipient's absorptive capacity, perceived less-valuable knowledge, arduous relationships, a lack of ability to recognize needs and opportunities to transfer, and a lack of ability to identify potential mechanisms for transfer [103, 256, 278, 317]. Thus, an antecedent assessment needs to be performed to understand what antecedents enable and impede the transfer process and then plan strategies for developing supportive contexts, minimizing obstacles to knowledge transfer, and defining all activities required for knowledge transfer to take place. Once the strategic planning and resource preparation is done, the actual activities can then be executed directly through to the second stage of Implementation [315]. From this view, it shows that the Initiation stage mainly associates with the components of problems, antecedents, knowledge, and mechanisms. Consequently, a flow of main knowledge transfer activities of this stage can be illustrated in Figure 7-2.

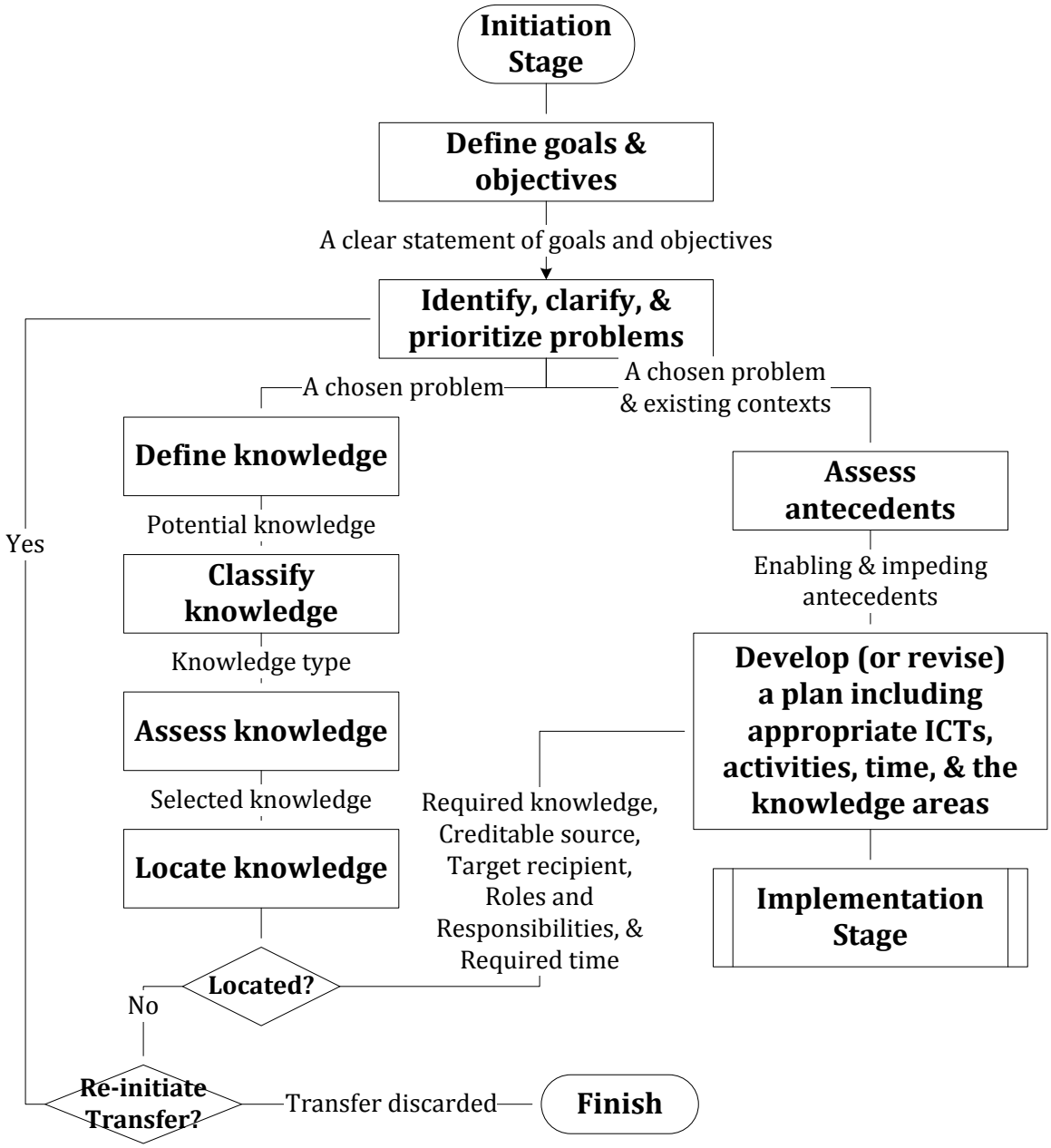


Figure 7-2. A flow of main knowledge transfer activities of the Initiation stage

7.2.2.2 Implementation

The Implementation is the subsequent stage commencing with the decision to transfer the knowledge [103]. When the Implementation takes place, the plan should be followed [315]. During this stage, resources flow between the source and the recipient. If the source stores in non-human (e.g., information systems and previous software project documents), knowledge may flow directly from the source to the recipient [276]. If the source is human, the updated or prepared materials may be distributed to the recipient. The knowledge is often tailored to meet the expected needs and pre-empt problems experienced in the past [103]. This is much more about communication between the source and the recipient [275]. Hence, positive personal ties between them need to be established for effective transfer [103].

Moreover, it is beneficial to engage in parallel activities that aim to prepare the subsequent Ramp-up stage, e.g., developing supportive contexts [103, 315]. This activity indeed benefits all stages of the transfer process and should be carried out at the early stages. There are many antecedents that affect the Implementation success and need to be prepared for the Ramp-up stage as following described.

First, teams should secure commitment for the Implementation success [315, 321-325]. While acceptance of the transferred knowledge and involvement are key issues at the Ramp-up stage, it is thus vital to get commitment from all key stakeholders (e.g., management and team members) in order to guarantee that the Ramp-up stage can be taken place. This includes early building on a general sense of commitment to the knowledge application, getting top-management commitment to support important issues (e.g., time, effort, resources, and management) that lead to successful Ramp-up and Integration, and securing commitment of team members since they are key players in the transfer process. However, it is important to provide training about the required knowledge to team members prior to securing their commitment.

Second, the recipient's deficiencies in absorptive capacity can lead the recipient to experience many hindrances in the transfer process, e.g., communication difficulties, large knowledge gaps, weak relationships, and perceived difficulties in learning the knowledge being transferred [316]. Therefore, efforts to improve the recipient's absorptive capacity should be undertaken [315]. Moreover, many studies state that the provision of training is an effective means to enhance absorptive capacity and knowledge application success [322, 354-356].

Third, the degree of knowledge transfer significantly depends on the source's wealth of experience, knowledge, and transferability [289]. The source with more relevant experience will easily initiate a transfer of knowledge from itself to the recipient [103]. Hence, the source' capability should be enhanced for higher inclination to share knowledge.

Fourth, technical and communication gaps between the source and the recipient can occur at this stage [278]. Bridging this gap successfully is related to careful planning. However, the depth of the planning itself depends on the understanding of the software project objectives and the knowledge being transferred. The degree of these effects significantly depends on the ability of the source and the recipient to work together to resolve conflicts between them as well as between the knowledge being transferred and their operating culture [278, 315]. This emphasizes that role and responsibilities of the source and the recipient need to be clearly clarified [317] and possible conflicts should be detected and pre-empted for effective implementation [315].

Fifth, Hendricson et al. [317] found that a lack of the recipient's motivation to adopt new knowledge, no tangible reward systems, and emotional support to encourage team members hamper the transfer process. In Chua and Pan's study [274], written tests were used to check whether the recipient absorbs the important things and to motivate the recipient to enable greatest absorption. Sundaresan and Zhang [357] designed the incentives that induce team members to share and learn knowledge and exert best efforts that are aligned with objectives. Besides, Duan et al. [266] said that organizations need to offer motivational incentives to team members. This shows that motivation of both the source and the recipient need to be maximized. However, in order to prevent a negative role of rewarding behaviors, those systems should be based on, e.g., successful knowledge transfer, collaboration, and teamwork, but not financial results or outcomes that are based on team competition [327].

Last, when the source is not perceived as credible or the knowledge being transferred is perceived as useful, the transfer process tends to be discarded [103, 256]. Since problems associated with antecedents can occur at any time, teams need to monitor and manage those antecedents and the plan to minimize failure probabilities of the transfer process. At this stage, the source customizes the knowledge being transferred until suitable for an absorbed unit. However, if the knowledge being transferred is unsuitable, teams may either (1) re-initiate the transfer process, (2) abandon the transfer process, or (3) re-assess the knowledge and antecedents again, review related occurring problems, use the assessment results to adjust the plan for improvement and then re-circle the Implementation stage until the knowledge being transferred is compatible for solving a particular problem and fitting into their software development contexts. Activities of this stage cease after the recipient begins using the transferred knowledge [103]. From this view, it shows that the Implementation stage mainly associates with the components of problems, antecedents, knowledge, and mechanisms. Consequently, a flow of main knowledge transfer activities of this stage can be illustrated in Figure 7-3.

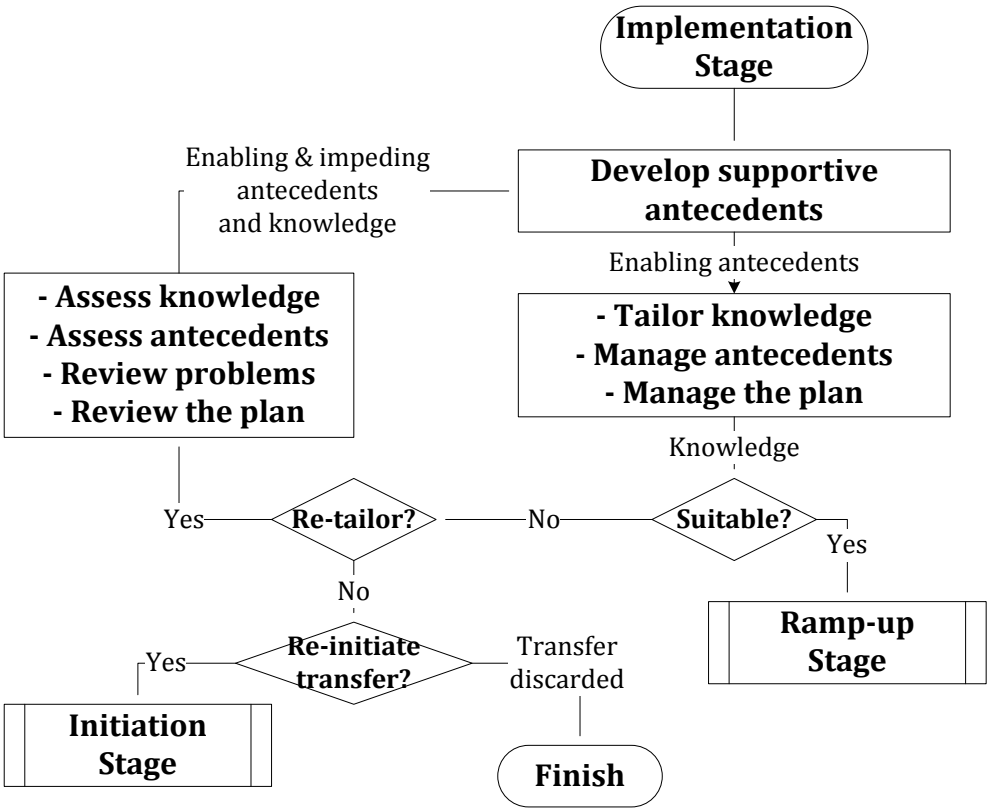


Figure 7-3. A flow of main knowledge transfer activities of the Implementation stage

7.2.2.3 Ramp-up

The Ramp-up starts when the recipient begins using the transferred knowledge [103]. The overall objective of this stage is to ramp up to work performance and satisfaction by using the transferred knowledge to solve the problem and meet the defined objectives [276, 315]. Typically, the recipient uses the knowledge ineffectively at first, but gradually identifies and rectifies unexpected problems until being able to achieve satisfactory outcomes [274, 276, 358]. During the use of the transferred knowledge, the recipient may request additional support from the source in solving both expected and unexpected problems. However, the recipient may abandon or re-initiate the transfer process if there are too many difficulties to use the transferred knowledge or it is unlikely to solve the problem or achieve a satisfactory outcome [276]. For instance, Tsang [279] found an important unexpected problem which is that the recipient sometimes continues to enact the old routine, albeit it has been replaced by the new knowledge. This has resulted from three main circumstances: insufficient support from the source during the Ramp-up stage that may lead to perceived difficulties in using the transferred knowledge, weak personal ties, and strong embeddedness of the old routine that leads the recipient to take time to use the transferred knowledge and later abandon its use. This implies that during practical use of the transferred knowledge, its incompatibility with either teams' cultures or existing practices may occur. Consequently, the transfer process may flow back to the Implementation stage again in order to alter the transferred knowledge being well-suited. The earlier the compatibility between the transferred knowledge and the transfer context is taken place; the likelihood of further abandonment of the transfer process is diminished. As routinization often requires cooperation and communication, weak relationships between the source and the recipient can thus be a barrier to knowledge application. To minimize failure, the source should continuously monitor and audit the use of the transferred knowledge, e.g., by obtaining feedback from the recipient and subsequently consolidating and using such feedback for further improvement.

Moreover, knowledge transfer leads to changes in behaviors, practices, and policies [345], successful Ramp-up thus requires acceptance of the transferred knowledge [103, 315]. For instance, the recipient may resist using the transferred knowledge that increases his/her workload or reduces his/her authority [359]. Hence, building acceptance of the transferred knowledge needs to secure motivation and involvement [315]. Since involvement is associated with commitment, personal ties, coordination and communication [360], securing involvement thus needs to augment these antecedents. From this view, Leseure et al. [315] also suggest many key activities that management should perform at this stage. First, management should pay attention to individual-level issues (e.g., credibility, transferability, absorptive capacity, awareness, motivation, behaviors, mindsets, and personal accountability) since those significantly affect transfer success or failure. In case of having no incentives or rewards yet, management may establish them to encourage team members to take time to use the transferred knowledge. Second, extensive communication is critically demanded to maintain for using the transferred knowledge and solving the problem. Third, maintaining focus while being flexibility is important for transfer success. When focus is absent, the adoption efforts are likely to decline. When the plan is idealistic, the transfer success is unlikely to be achieved. However, teams may suffer from either too much focus or too much flexibility. During this stage, management should therefore regularly maintain and review, and revise the plan to suit any particular situation. Meanwhile, antecedents affecting successful Ramp-up (e.g., extensive communication, great motivation of the source and the recipient, positive personal ties, and the organizational culture) also need to be maintained,

re-assessed and, improved. The problem must be focused to ensure successful solving and reviewed to early tackle any unexpectedness. Activities related to the use of the transferred knowledge (i.e., making, monitoring, supporting, and auditing the used of the transferred knowledge) should be carried out until being able to solve the focused problem and achieving satisfactory outcomes. After achieving satisfactory outcomes, the transfer process then flows through the Integration stage. From this view, it shows that the Ramp-up stage associates with the components of problems, antecedents, mechanisms, knowledge application, and outcomes. Consequently, a flow of main knowledge transfer activities of this stage can be illustrated in Figure 7-4.

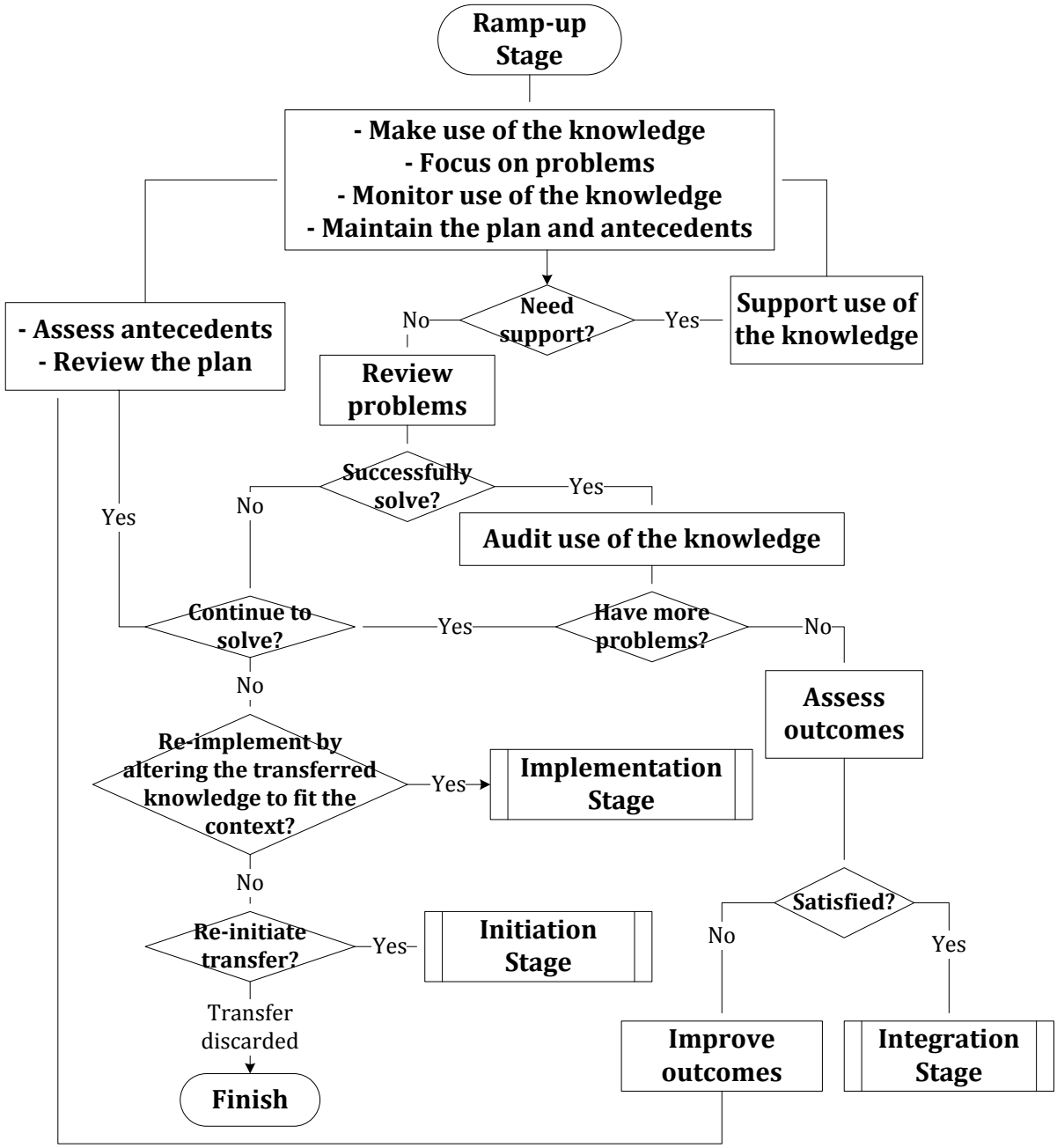


Figure 7-4. A flow of main knowledge transfer activities of the Ramp-up stage

7.2.2.4 Integration

The Integration stage begins after the recipient gained desired outcomes with the transferred knowledge [103]. Knowledge application and its integration with existing routines (or existing practices) gradually become routinized (called institutionalization) [317]. A routine may blend into standard practices, so that the transferred knowledge can be then embedded within day-to-day tasks. A routine is often not a stand-alone item [279]. Routines are interconnected in the sense that the performance of one may involve that of another. This shows that it is unlikely to change all the routines at the same time. Teams may hence bring in new routines in a sequential manner. At this stage, the Integration activities are carried out to ensure that the recipient can use the transferred knowledge without any support from the source and can take any remedial action to improve the understanding of the transferred knowledge and integrate it into his/her practices [274]. Three main activities of this stage consist of sustaining, monitoring and quality auditing the use of the transferred knowledge [316]. There are many ways to sustain the knowledge, e.g., providing environments to facilitate and increase knowledge use. during monitoring, the source should allow the recipient to make a mistake and correct the mistake by himself/herself, provide support when the recipient encounters a tough problem, and supervise the recipient until the knowledge is internalized and then integrated as part of his/her own knowledge package. Quality auditing is an ongoing assessment activity. The source may give feedback to the recipient and vice versa, provide coaching, conduct an remedial action plan to help the recipient to overcome his/her weaknesses, and eventually check with all team members whether the outcomes are satisfied. In order to gauge whether the recipient indeed absorbs the knowledge, there are many assessment mechanisms, e.g., an oral test and a written test [274]. The assessment may focus on the recipient's deeper understanding and the absorption of the transferred knowledge and the ability to analyze and solve problems. Meanwhile, the assessment may also rate on the recipient's soft skills (e.g., eagerness to learn, willingness to share, and teamwork) and the source is then discussed to make the recipient aware of his/her strengths and weaknesses for further improvement. This can serve as an indication of the desired characteristics of team members.

This stage primarily looks at the efforts required to minimize obstacles and deal with challenges to the routinization of the transferred knowledge [316]. When the transferred knowledge presents too many difficulties, it is unlikely to become part of routines and therefore sustained in a practice [278]. In other words, the transfer process may be abandoned or re-initiated if there is no suitable remedial plan to deal with encountered difficulties. This emphasizes that successful knowledge transfer depends upon the ability to remove barriers and cope with how to make the transferred knowledge or the new practice more routine. Apart from recipient's absorptive capacity and motivation described above, there are various antecedents affecting the institutionalization of new routines or leading the transfer process back to the former stages.

For instance, first, as routines can be stored in both the non-human memory (e.g., project documents, standard practices, organizational policies, and information systems) and the collective human memory (e.g., team members); however, the latter usually forms the greatest barrier to change [279]. The older the organization, the more fossilized the collective human memory becomes. Existing old routines of an old organization may become part of the members' work behaviors that are difficult to get rid of. Albeit the transferred knowledge is operational for a considerable time, team members may somehow revert to the old routines [279].

Second, time facilitates unlearning since human memory fades with time [279]. When the transferred knowledge is not enacted regularly, tendency to revert to the old routine makes institutionalization of the transferred knowledge difficult. Two of possible means for dealing with this matter are either motivating team members the current stage or turning back to the Ramp-up stage by continuously using the transferred knowledge until getting satisfactory familiarity with it.

Third, the transferred knowledge may neither be compatible with other existing old routines nor easily accessible. Although the transferred knowledge or the new practice is considered as valuable, it may not take root in teams. For instance, if the existing storing information system is awkward and the transferred knowledge cannot be accessed efficiently, team members is not motivated to collect it [279]. In this case, teams may either fix the incompatibility at the current stage or back to the Implementation to alter the transferred knowledge being able to mingle with the old routines.

Fourth, the complexity of the knowledge and its limitations to use the knowledge may take the recipient more time and effort to fully complete this stage [317]. Another factor is that communication difficulties may bring on misunderstanding and distrust between the source and the recipient, which in turn results in a weak relationship and eventually hampers the successful knowledge transfer [316]. However, those communication difficulties can be overcome by team building and personal ties between the source and the recipient can be strengthened by social entertainment activities. Hence, teams should maintain, re-assess, and improve those antecedents to ensure successful knowledge transfer. When the recipient can integrate the transferred knowledge into his/her knowledge package and use it without any support, the knowledge transfer is then recognized as successful. From this view, it shows that the Integration stage primarily associates with the components of problems, antecedents, knowledge, mechanisms, and knowledge application. Consequently, a flow of main knowledge transfer activities of this stage can be illustrated in Figure 7-5.

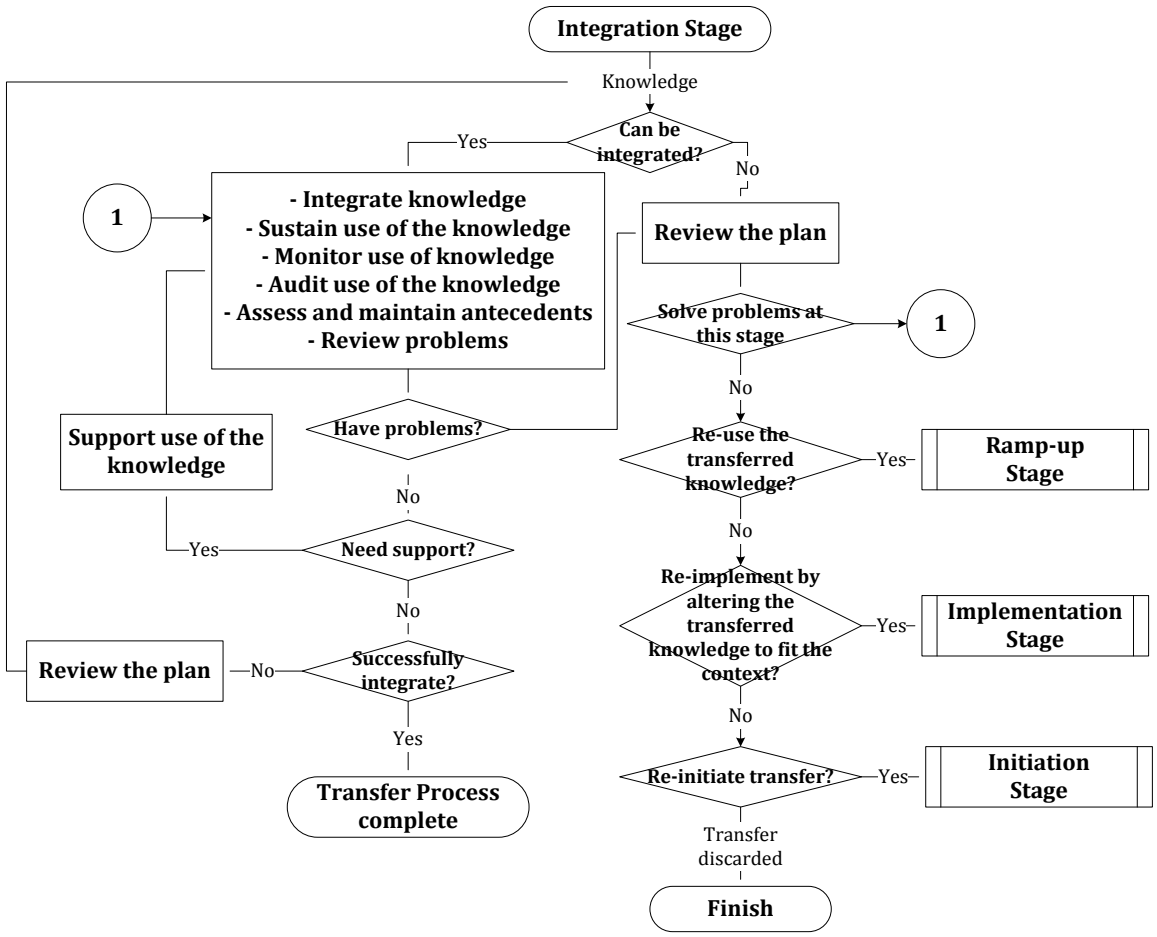


Figure 7-5. A flow of main knowledge transfer activities of the Integration stage

According to Figure 7-1, it shows multi-directional interactions between components in the transfer process. Giving a clearer picture, Table 7-11 summarizes sets of components playing a vital role in each stage of the transfer process.

Table 7-11. Core components in each stage of the transfer process

Component \ Stage	Initiation	Implementation	Ramp-up	Integration
Problems	X	X	X	X
Antecedents	X	X	X	X
Knowledge	X	X		X
Mechanisms	X	X	X	X
Knowledge Application			X	X
Outcomes			X	

There are many difficulties leading to ineffective knowledge transfer or backward stages as described above. Providing a clearer view, Table 7-12 summarizes those difficulties

and influential antecedents in each stage of the transfer process that should be taken into consideration in practice.

Table 7-12. The difficulties and influential antecedents in each stage of the transfer process

Context	Stage / Difficulty	Initiation	Implementation	Ramp-up	Integration
Source	(Lack of) motivation		X	X	
	(Lack of) capability		X		
	(Lack of) credibility	X	X	X	
	(Lack of) sufficient support from the source			X	
	(Lack of) ability to recognize needs and opportunities to transfer	X			
	(Lack of) ability to identify potential transfer mechanisms	X			
Recipient	(Lack of) motivation		X	X	X
	(Lack of) absorptive capacity	X	X	X	X
	(Lack of) regularly practicing the transferred knowledge				X
Knowledge	(Lack of) knowledge perceived as easy to use/access		X	X	X
	(Lack of) knowledge perceived as useful	X	X		
	Large knowledge gaps between the source and the recipient		X		
Relational	(Lack of) good relationship	X	X	X	X
	(Lack of) commitment		X	X	
Situational	(Lack of) communications		X	X	X
	Strong embeddedness of old routines (or old practices)			X	X
	(Lack of) compatibility between the transferred knowledge and old routines (or old practices)			X	X

For more understandings, the descriptions of our knowledge transfer framework's application in practice are presented in the next section.

7.3 Application of the Knowledge Transfer Framework

Owing to time limitations of this study, we could not evaluate our knowledge transfer framework in terms of usability and practicality in real-life software projects. Nevertheless, we hope to further improve the framework by carrying out empirical case studies in the Thai telecommunications industry in the near future. Since the framework provides a general conceptual lens of knowledge transfer in software development; in case positive results are arisen, empirical case studies with an improvement of the framework may be performed in other industries. However, our previous software development case studies in two Thai telecommunications companies (CAT Telecom Public Company Limited and TOT Public Company Limited) (presented in Chapter 5) are used to demonstrate the application of the

framework in practice in this section. This helps answer the RQ7-4 “How can the developed knowledge transfer framework be performed?”.

7.3.1 Data Collection

We use the collected data and findings of our previous case studies as a base. The case studies were performed to test our software process maintenance framework at CAT Telecom Public Company Limited (CAT) during November - December 2010 and TOT Public Company Limited (TOT) during December 2010 - February 2011. Quantitative and qualitative data were collected through on-site observations, individual semi-structured interviews, and questionnaires. We interviewed three team members in CAT face-to-face (e.g., a product owner, a Scrum master, and a developer); and a Scrum master who also acted as a product owner and a developer via an international call for the case project in TOT due to limited available time of the team. A Technology Acceptance Model (TAM)-based questionnaire for investigating the acceptance of our software process maintenance framework was also developed based on the related literature and circulated to all interviewees (see Chapter 5 for more details).

7.3.2 Analysis and Results

In this section, the Scrum principles are borrowed for describing the way to perform software development with our knowledge transfer framework. The mapping between the three stages of Scrum (i.e., pre-game, game, and post-game) and the four stages of a knowledge transfer process (i.e., Initiation, Implementation, Ramp-up, and Integration) is depicted in Figure 7-6. The pre-game stage of Scrum and the Initiation of the transfer process mainly involve planning, whilst the game stage of Scrum and the Implementation stage of the transfer process are the places to execute actual planned activities. Therefore, two pairs of these stages can be mapped together. Typically, knowledge is used to solve business problems during the game stage of Scrum. The software product, work performance, and user satisfaction is evaluated through validation and verification at the post-game stage of Scrum. As the Ramp-up stage of the transfer process begins when the recipient starts using the knowledge and then ramp up it to work performance and satisfaction, activities of the Ramp-up stage can thus be performed through both the game and post-game stages of Scrum. As the transferred knowledge needs to be routinized, the Integration stage of the transfer process can thus be executed in the next iterations together with the creation of new knowledge transfer. It is important to note that the proposed mapping is based on theoretical aspects. In practice, all stages of the transfer process can be executed at any stages of Scrum (or software development). For more understandings, the descriptions of each stage of the transfer process are presented in the following sub-sections. The descriptions are categorized into two parts, i.e., the descriptions on the author side transferring knowledge to the CAT and TOT teams and the descriptions on the case study side transferring knowledge within their teams. Consequently, the focus of the first part is on the results of transferring our software process maintenance framework; whilst the focus of the second part is on how our knowledge transfer framework can be applied in real-life software development practice.

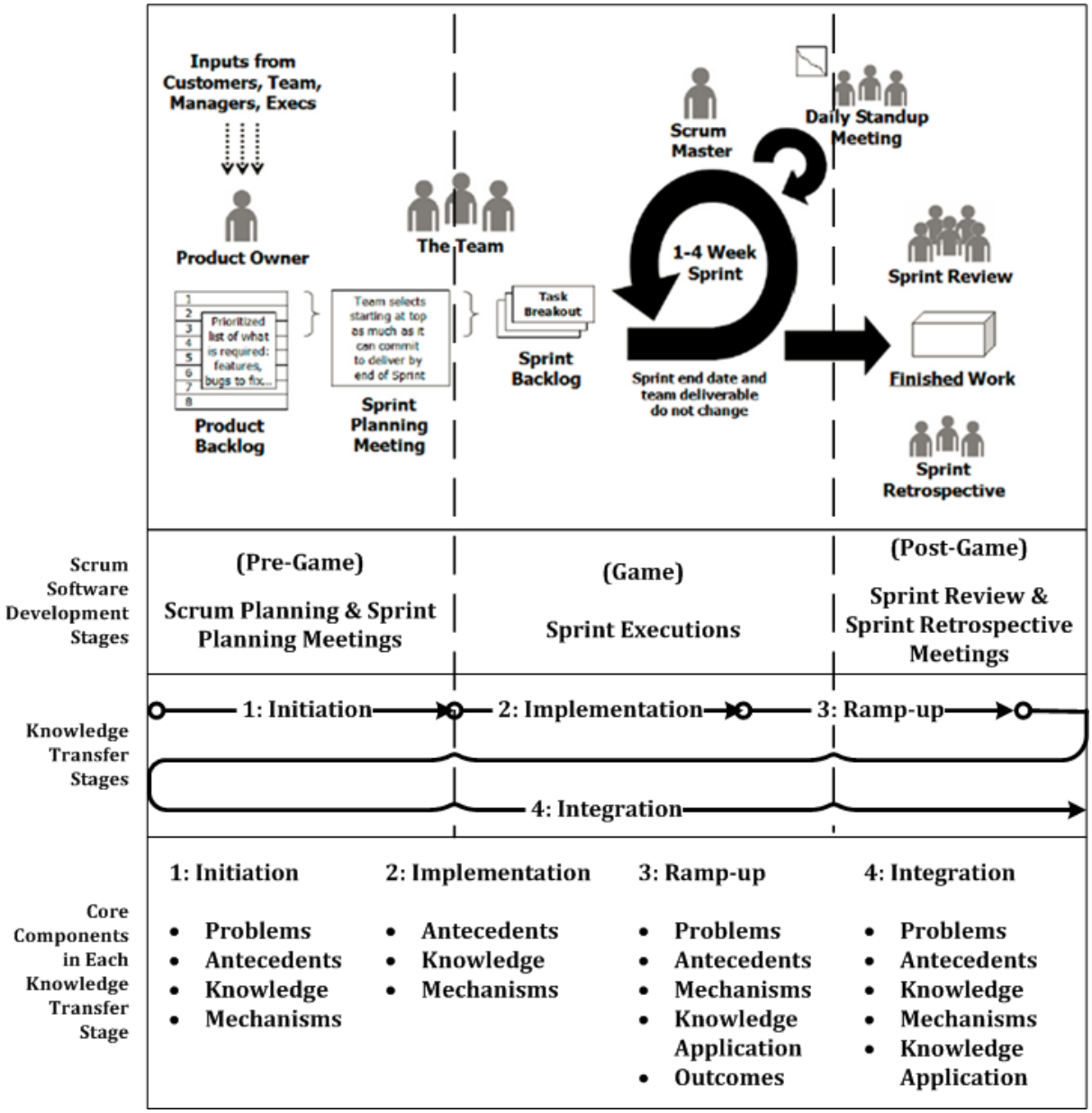


Figure 7-6. A mapping between Scrum stages and knowledge transfer stages (Scrum's source: [102])

7.3.2.1 Initiation

On the author side transferring knowledge to the teams: Initiation is triggered by all events leading to the decisions to transfer, e.g., the search for problems, necessary knowledge, and potential solutions. As presented in Chapter 2, the potential problems we found in software development in the Thai telecommunications industry is that software developers (hereafter referred to as "developers") did not perceive formal routines as an efficient and effective way to develop and manage software development and deliver quality results. Hence, the question of "how to improve software development performance" has been brought up. A software process maintenance framework has been proposed and constructed, based on suitable necessary knowledge. For evaluating the usability and practicality of the framework, the framework's knowledge was required to transfer to our

case studies. For targeting recipients, CAT and TOT were chosen due to their great interest in the framework and their experience in the telecommunications and software development. However, one of challenges was that a developer in the TOT team found the framework difficult since TOT usually used an outsourcing method for their existing software projects. On the other hand, the CAT team had experience with PMBOK. Therefore, an amount of knowledge required to transfer to the TOT team was heavier than that required to transfer to the CAT team. Without related experience, it was difficult and required more time to transfer the knowledge to the recipient. For planning transfer mechanisms, the authors used face-to-face communications as a primary method, emails when either the authors or the teams were remote, and interviews and questionnaires at the end of their software development to measure their learning, understandings, satisfaction, and acceptance of the framework. Once the transfer schedules were done, the authors needed to prepare documentations and presentation slides, pull together all related documents and tools that the recipients should have, set up our software application supporting the use of the framework, and conduct interview and questionnaire questions.

On the case study side transferring knowledge within teams: In software development, transfer activities of the Initiation stage can be performed at an early stage of software development (i.e., initiating and planning a software project). During software project initiation, management conducted a project charter and a stakeholder analysis. Apart from software development aspects, the teams should have additionally analyzed what skills, experience, and knowledge each stakeholder had; the degree to which each stakeholder had that knowledge; and each stakeholder's characteristics (antecedents) that affected knowledge transfer (especially in terms of motivation, credibility, capability, absorptive capacity, commitment, and relationship amongst team members).

During sprint planning meetings, management developed project management plans and formed teams. The case project in CAT was to develop additional Web-based functionalities bundled into their ongoing software project. The CAT team was formed with the same team responsible for the ongoing software project in order to reduce time to learn business logic, programming languages, and development tools. The TOT team developed a small decision support application and was formed with people having business experience in the telecommunications industry but not much in software development. This supports that the degree of required knowledge transfer in the TOT team was higher than that in the CAT team. Product owners clarified to the teams the software project roadmaps, goals, and objectives by using up-to-date related presentation slides and documents. As both teams carried out the software process assessment before software development in order to understand the strengths and weaknesses of their existing software processes, the assessment results and the defined objectives may have helped teams being able to identify potential problems (as knowledge transfer requirements) that needed knowledge transfer for software process and product improvement. After gathering requirements (i.e., user, technical, and knowledge transfer), they would be then prioritized by considering their values and logged into a product backlog together with estimated effort in terms of time.

As knowledge training and coaching requires time, adequate independence from the software development tasks should be provided to team members involved in the transfer. It means that an amount of user and technical requirements needed to be implemented in each iteration may be reduced. Moreover, iterative transfer is recognized as more effective since it is easier to transfer clearly defined chunks with predictable schedules [361]. Therefore, all prioritized requirements should be scheduled into iterations. Once getting the first chunk of the prioritized requirements to implement, the teams broke down those requirements into

tasks and logged them into a sprint backlog. Similar to the process of breaking down user and technical requirements, for a given knowledge transfer requirement, the teams should have defined and classified what knowledge and what types of the knowledge were needed for transfer.

The CAT team developed a Web-based application as a commercial product, whilst the TOT team developed a decision support application for planning business strategies. According to eTOM business process areas, both applications can be classified into the Strategy, Infrastructure and Product (SIP) business process area. It means that all of the five knowledge types (i.e., human, organizational, relational, project management, and technological knowledge) may be required for transfer. The teams should then have assessed the required knowledge by considering its values, complexity, and accessibility and select the most suitable one. For instance, the CAT team planned to apply a standard software process improvement method (e.g., Capability Maturity Model-CMM, CMMI, Software Process Improvement and Capability Determination-SPICE, and Six Sigma). The CAT team decided to be certified in CMMI due to being very suitable for their business purposes and having its knowledge source in their team. After getting suitable knowledge, the teams should have found a credible source having such knowledge and located it in the teams. Sources can be both human (e.g., team members and consultants) and non-human (e.g. project documents, organizational policies, and information systems). However, the transfer process may be abandoned or re-initiated if the knowledge cannot be located. It will not be plausible to teach everybody for every task [361]. Therefore, target recipients need to be defined.

Agile software development methods typically advocate face-to-face communications as the most efficient and effective method of conveying information [232]. The CAT team approximately 60% in co-location and approximately 40% in distributed sites. They planned to employ face-to-face interactions and mobiles as main communication media and used instant messaging, e-conferencing, and emails when team members were remote. This enabled them to obtain quick feedback and required information. On the other hand, The TOT team worked in 100% co-location, but planned to employ phones as a primary communication method for both developing software and transferring knowledge. This was due to multi-roles and multi-projects of the Scrum master. The details of the source, the recipient, their roles and responsibilities, the knowledge area, and communication channels for a particular knowledge transfer requirement should have been logged into the sprint backlog and clearly clarified to all related team members.

Besides, during identifying and planning risk or impediment management, influential antecedents should have been assessed to understand what facilitated and hindered the transfer process. The teams may have used the results of the stakeholder analysis as an input for measuring influential antecedents in the source (i.e., motivation, capability, and credibility), recipient (i.e., motivation and absorptive capacity), and relational (i.e., relationship between the source and the recipient and commitment) contexts. For the hindering antecedents, they may have been logged into an impediment backlog with strategic solutions.

It is essential to prepare materials for transfer. Sources may have put together a package for their areas of expertise into various forms of documentation and presentation slides. Once the planning and resources were done, the actual activities could be directly executed in the Implementation stage of the transfer process. However, it is important to note that some of actual knowledge transfer was performed during planning the software project, e.g., describing user requirements to team members.

7.3.2.2 Implementation

On the author side transferring knowledge to the teams: The Implementation commences with the decision to transfer. Before the scheduled transfer time, preliminary explicit knowledge in the form of documentation were handed over to the recipient teams. This allowed them to go through the documentation to understand the overall concepts of the framework. At the scheduled transfer time, the actual transfer activities were executed. Face-to-face communications (including presentations and conversations) were used as a key transferring and learning mechanism at the beginning of the transfer. The training period for the Implementation stage was 1-2 weeks. Not all recipients who attended the training were able to absorb the knowledge due to less experience in either PMBOK or agile-oriented software development. However, the recipients were not expected to understand all the knowledge immediately. The intention was first to provide them the concepts of the framework (including related tools), and then to allow them to get better understandings by doing.

During the Implementation stage, the authors faced three main circumstances leading the transfer process re-transferred or discarded. First, although training by face-to-face conversations was in Thai, all of the original prepared materials (i.e., related documents, presentation slides, and tools) were in English. As recognized during the transfer, when the authors transferred knowledge without translation from English into Thai, some recipients required much time and effort to learn. The authors sometimes had to re-transfer with the translated version to ensure that the recipient clearly understood that knowledge. Moreover, albeit the TOT team was interested in using those materials (e.g., project management and backlog templates) in real-life practice, they needed to convert almost all of them into Thai according to their organizational policies.

Second, after the authors consulted a set of software development practices that should be performed for quality software development and products, there was one practice unsuitable for the TOT culture. The practice was that key team members (e.g., a product owner and a Scrum master) directly involving the software project should have full authority for rapid making decisions. Top management in TOT refused this practice with the reason of no such practice performed in TOT before. Nevertheless, the main reason behind it may be the threat to lose their authority as is what usually happens in traditional organizations.

Last, the TOT team did not apply a retrospective meeting (which is one of main meetings suggested by the framework) for the case project due to no sufficient time of the Scrum master. The transfer process of this knowledge could be considered as failure. These circumstances support that antecedents should be regularly reviewed and managed in order to reduce transfer ineffectiveness and failure. Once the recipients were ready to use the transferred knowledge, the transfer process then flowed through the Ramp-up stage.

On the case study side transferring knowledge within teams: Transfer activities of the Implementation stage can be performed during sprint executions. During this Implementation stage, resources flow between the source and the recipient. Therefore, the first step should be the development of supportive antecedents in accordance with the impediment backlog and the strategic plans. As observed, a developer in the TOT team lacked absorptive capacity due to less prior software-development-related knowledge, whilst developers in the CAT team somewhat lacked motivation or interest in project management knowledge that may in turn have led to a lack of absorptive capacity to learn and use that knowledge. Moreover, management (especially in the TOT team) as a main source had multi-roles and multi-projects. This led to inadequate communications with the recipient and a lack

of commitment in terms of time. This supports that the teams needed to build up absorptive capacity, motivation, the extent of communication, and commitment.

Consequently, the absorptive ability may have been enhanced by allowing team members to learn by doing and making mistakes. Learning by doing can strengthen the understanding of tasks through gaining accumulative practical experience [361]. Besides, the source who is threatened to lose his/her importance or authority is likely to demonstrate non-cooperative behaviors with the recipient. A clear vision of his/her future may be communicated early for motivation. Nevertheless, the sources in both teams as observed were willing to transfer their knowledge to their team members. To motivate the recipient, management in CAT motivated team members through conversations and planned to establish a reward system on future software projects; whilst management in TOT considered the case project results as one of key performance indicators for the annual staff performance appraisal. For communication aspects, both teams realized that only face-to-face conversations could not be held for all software processes. They thus increased the volume of communications by using other media, e.g., phones and instant messaging. However, using phones in TOT in the Implementation stage was not highly efficient. In addition, it is important to make awareness and get commitment for knowledge transfer from all key stakeholders and team members. Management in both teams should have secured both key stakeholders' and team members' commitment especially through the Implementation and Ramp-up stages of the transfer process.

Meantime, the source should have tailored and transferred the required knowledge until suitable for the current software development context. As observed during the coding stage of software development, the CAT team largely exchanged technological knowledge and discussed on how to adapt it for improving their development techniques and software maintainability. During conducting project documents, organizational knowledge (e.g., organizational templates, standards, and policies) was engaged. Most of those documents were informal and less-detailed. This implies that most transferred knowledge was more likely to become human knowledge residing in individual team members. On the other hand, the source in the TOT team used on-the-job training to transfer programming techniques to the recipient. Since the recipient had no experience with the programming language before, time and effort for learning and coding was highly required. However, if the required knowledge was not suitable or effectively tailored, the transfer process was likely discarded or re-initiated.

During sprint executions, daily meetings took place to coordinate work, synchronize efforts, and tackle anticipated problems. Similar to software processes, the knowledge transfer requirements, impediments, and all related plans (e.g., sprint backlogs, impediment backlogs, and risk management plans) should have been managed, reviewed, and adjusted to fit into the current circumstances. During these meetings, influential antecedents should also have been reviewed in order to minimize failure chances. For instance, although good motivation and relationship was taken place in both teams, a conflict between the source and the recipient sometimes occurred. When this occurred, it was likely to decrease the extent of communication and in turn motivation to transfer and acquire the knowledge. This emphasized that a lack of supportive antecedents was more likely to result in transfer ineffectiveness or abandonment. It is also important that the teams should have measured whether or not an amount of knowledge being transferred was sufficient for achieving satisfactory outcomes through these meetings. The transfer activities of the Implementation ceased when the recipient started using the transferred knowledge. The transfer process then flowed through the Ramp-up stage.

7.3.2.3 Ramp-up

On the author side transferring knowledge to the teams: During this stage, the recipient gradually ramps up to work performance and satisfaction by using the transferred knowledge to solve anticipated problems. On-site face-to-face communications were mainly used for observing, supporting, monitoring, and auditing the use of the transferred knowledge; whilst emails were used to provide support requested when either the authors or the recipients were remote. Transfer activities of this stage ceased after the recipients achieved satisfactory outcomes. Otherwise, the transfer process needed either continuing until getting satisfaction, reverting back to earlier stages, or unfortunately discarding.

In case of needing to re-tailor the transferred knowledge, both CAT and TOT teams adapted the Scrum practice to fit their software development environments. This practice was that in daily meetings developers should provide answer to three daily questions (i.e., "What did you do yesterday?", "What will you do today?", and "What impediments are in your way?"). Both teams performed the three questions uncomfortably at first and also felt incompatible with their team cultures. Hence, they decided to apply the first two questions sometimes but only the last question for every daily meeting. This is because on the prior software projects, the developers usually reported their work progress at the scheduled time and spoke of any problems, as they occurred. This existing routine was somehow fossilized into their work behaviors. In this situation, if they could not overcome this fossilized practice, the first two questions were unlikely to be routinized into their standard practices. However, they recognized that the last question was crucial for early visibility to tackle risks and problems. Hence, this knowledge was more likely to be integrated and gradually become their standard practices.

In case of requiring more knowledge to transfer, although work could proceed in priority order, there was a conflict between business value criteria and technical criteria in requirement prioritization. In dealing with this matter, they sought to set ground rules for score rating and used CPM (Critical Path Method) together. As observed, the CAT team had less experience in using it; so that, such knowledge was required. On the other hand, the TOT team may have been used PERT (Program Evaluation and Review Technique).

In case of requiring more time until achieving work satisfaction, especially the CAT team felt that they spent much time on backlogs. The developer pointed out "*Our weak skills in the work breakdown structure slowed us down.*" Owing to the beginning of their agile-oriented journey, this led to more non-task effort on each sprint. This effort can be minimized when they gain more experience on these practices. Otherwise, it might cause a problem if this feeling was not reduced. Another situation was pertinent to sprint retrospective meetings. Only the CAT team performed the meetings. However, the team was just beginning their journey in gaining a deeper understanding of the software process and taking ownership of the software process. This knowledge required time to ramp up the effective use of transferred knowledge. Albeit this knowledge is very useful, it is not easy. It is likely to be discarded in the future if they encounter too many difficulties to use it. During, both teams requested the authors' support or suggestions to solve their problems some times.

In case of satisfaction with the outcomes, the recipients especially in the CAT team were satisfied with sprint review meetings. The product owner in CAT pointed out *"It's like we commit to work together… take responsibility for failure together."* The team said *"It's like we reiterate requirements together again and again, to check on whether we are going in the same direction."* This supports that better software quality was gained as direct results of software development; whilst cultivating teamwork was gained as indirect results. Another

situation was pertinent to the overall satisfaction and usability of the framework. The recipients perceived the framework as useful and easy to deal with their existing problems and increase team performance and productivity. When the transferred knowledge was satisfied, they considered to integrate it into their existing practices. The transfer process was recognized to flow through the Integration stage.

On the case study side transferring knowledge within teams: Transfer activities of the Ramp-up stage can be executed through sprint executions and sprint reviews. In order to accomplish tasks as committed, the use of the transferred knowledge is likely to occur during sprint executions. As observed, at the first use of the transferred knowledge, a developer in the TOT team who had been transferred the programming techniques requested lots of decision support from the source to solve any occurring problem. Owing to no experience with this knowledge before, the developer took more time to use the transferred knowledge properly. However, the volume of support was decreased when getting deeper understandings of the transferred knowledge through gaining practical experience. Moreover, the developer encountered many difficulties during the use of the transferred knowledge. Even though the developer was allowed to phone the source for any support at any time, the source might have not supported or helped to solve the problem immediately every time due to his multi-projects. Typically, when the problems cannot be resolved within a reasonable time, recipient's motivation to use the transferred knowledge may easily decline. This supports that ineffective communication channels and a lack of source's monitoring significantly lead to transfer ineffectiveness or unfortunately the transfer process discarded. To reduce unsatisfactory results, the source should have regularly supported and monitored the recipient's use of the transferred knowledge. As daily meetings are the place to monitor work progress and tackle problems; all requirements, problems, antecedents, and related plans should have been maintained, reviewed and adjusted.

In sprint review meetings which are places for demonstrating the team's accomplishment during sprint executions, the use of the transferred knowledge can be audited. The software products and knowledge transfer outcomes (i.e., work performance in terms of efficiency and effectiveness as well as work satisfaction in terms of perceived usefulness and perceived ease of use) can also be verified and validated against the sprint backlog. In case of team members unsatisfied with the outcomes, the transfer process may be either continued at the Ramp-up stage or reverted back to the earlier stages (i.e., Implementation and Initiation stages). For instance, developers in the CAT team misunderstood user requirements and in turn delivered the wrong work. In this case, the transfer process needed to be reverted back to the Implementation stage. The project owner as the source had to re-transfer the required knowledge and ensure the developers' understandings by informal oral questions and answers. As observed, if the rejected work could be fixed within approximately 15 minutes, both the Implementation and Ramp-up stages of the transfer process were re-circled during sprint review meetings. Otherwise, the rejected work and the Implementation and Ramp-up stages of the transfer process need to be re-executed in the next iteration. In case of team members satisfied with the outcomes, the transfer activities of the Ramp-up stage will stop. The transfer process then flows through the Integration stage. The transfer activities of the Integration stage may be discussed in either sprint review meetings or sprint retrospective meetings and executed in the next iterations. In this study, these activities are discussed in sprint retrospective meetings.

7.3.2.4 Integration

On the author side transferring knowledge to the teams: the first activity of this stage should begin with the consideration to integrate the transferred knowledge into the existing practices. As observed, there was no the satisfied, transferred knowledge that could not be integrated into both teams' existing practices. In other words, the knowledge was not contrary to their organizational standards and policies. After the first iteration, both teams found the transferred knowledge could be integrated into their existing practices. Since they were just beginning their journey in agile-oriented software development, they needed time to continuously learn, adapt, and use the transferred knowledge in next iterations until assimilating the transferred knowledge into their knowledge packages and completing their case projects. From the second iteration onwards, on-site observations were used to monitor, audit, and support the use of the transferred knowledge when needed. Interviews and questionnaires were also used at the end of the case projects to measure whether the learning was really taken place and whether the transferred knowledge was likely to be sustained in their further software projects.

Considering the outcomes in terms of work performance and work satisfaction at the end of the cases, based on the questionnaire findings, the average scores of (1) the increased work productivity, (2) the increased work effectiveness, (3) the increase work performance, and (4) the improved quality of software process and product, rated by the CAT team were 4.33, 4.67, 4.67, and 4.33 out of 5 points. In the TOT team, the average rated scores of those variables were 4, 5, 5, and 5 out of 5 points, respectively. Work satisfaction in terms of perceived usefulness and perceived ease of use was perceived in both teams. The interview findings reveal that both CAT and TOT teams were strongly satisfied with their work and the software process maintenance framework. Based on the questionnaire findings, the mean values of perceived usefulness and perceived ease of use rated by the CAT team were 4.33 and 4.2 out of 5 points; whilst those rated by the TOT team were 4.43 and 4.2 out of 5 points, respectively. As observed, management in both teams sustained the transferred knowledge by defining some of the transferred knowledge (e.g., software processes in the areas of sprint planning, sprint executions, and sprint reviews) as their standard practices and provided environments to support its use. This strategy somewhat forced their team members to regularly use that knowledge.

Based on these findings, we consider that our defined problems in the part of an efficient and effective software process were solved and the transfer process was recognized as successful. Owing to time limitations of this study, we could not continuously monitor and audit the use of the transferred knowledge in both teams after the case projects' completion. Nevertheless, we hope to follow up their software practices in the future.

On the case study side transferring knowledge within teams: Sprint retrospective meetings are places for lessons learned. Hence, they are good place to discuss and get feedback on integration of the transferred knowledge and existing practices and its use. When integration solutions exist, using and sustaining the transferred knowledge can then be continued in next iterations. However, if the transferred knowledge cannot be integrated into their exiting practices, the transferred knowledge is more likely to be abandoned. As observed, there were no noticeable situations being able to be indicated that the transferred knowledge needs to be discarded or reverted back into the earlier stages of the transfer process.

During the Integration stage, it is important to ensure that the recipient can use the transferred knowledge without any the source's support, take any remedial action to better the

understandings of that knowledge, and assimilate that knowledge into his/her knowledge packages. Thus, three main activities in this stage are sustaining, monitoring, and auditing the use of that knowledge. For sustaining, as aforementioned, management in both teams defined some of the transferred knowledge as their standard practices and provided environments to support its use. For monitoring, the sources in both teams allowed the recipients to make a mistake and correct it by themselves. However, when the recipients encountered a tough problem, the source helped solving that problem with the recipients. For auditing, the use of the transferred knowledge was mostly audited at sprint review meetings through oral questions and answers that served as a quality check on whether the learning had taken place. The feedback from both sides of the source and the recipient was shared. The feedback may have been logged in lessons learned documents or impediment backlogs, and then used to plan remedial activities for further and better use. Once the transferred knowledge is routinized effectively, the transfer process can be recognized as successful.

Based on the descriptions above, there were many difficulties leading to knowledge transfer ineffectiveness, backward stages, and unfortunately knowledge transfer abandonment in the two case studies. Providing a clearer view, Table 7-13 summarizes the difficulties that were observed in the two case studies and should be taken into account for future practice. In this table, "C" represents "CAT" and "T" represents "TOT". For instance, the "T" falling into the "Implementation" and the "Lack of commitment in terms of time" means that we found a lack of commitment in terms of time at the Implementation stage of the transfer process in the TOT team. Hence, TOT should pay more attention to this difficulty for future practice.

Table 7-13. The difficulties of knowledge transfer found in two case studies

Context	**Stage** / **Difficulty**	**Initiation**	**Implementation**	**Ramp-up**	**Integration**
Source	Insufficient support from the source			T	T
Recipient	Lack of motivation		C		
	Lack of absorptive capacity		C, T	C, T	C, T
Knowledge	Knowledge not perceived as easy to use or access	T	T	C, T	
Relational	Lack of commitment in terms of time		T	T	T
Situational	Lack of intensive communication or inappropriate communication channels		T	T	T
	Strong embeddedness of the old routine			C, T	
	Incompatibility between the transferred knowledge and existing old routines		T		

Owing to time limitations of this study, we could not carry out an empirical case study to test the knowledge transfer framework in real-life software project. However, the demonstration above shows that the framework has a high degree of compatibility with Scrum-oriented software development. In the future, we hope to empirically evaluate and improve the framework in the Thai telecommunications industry in order to raise more confidence on its usability and practicality.

7.4 Summary

Only a software development approach may not guarantee that a software project will get a smooth landing. This is because success in software project requires efficient and effective processes of software development and knowledge transfer, stakeholders' expertise and experience, and the ability to transfer, acquire, and apply knowledge to solve any development problems. This emphasizes that the presence of sufficient knowledge on teams is crucial, depending upon the uniqueness of the required outcomes. Even though many approaches to knowledge transfer in software development have been proposed, how to achieve software process and product quality enhancement through knowledge transfer still remains a challenge. To overcome this challenge, this study was carried out in three steps. First, 27 highly visible knowledge transfer studies which explain all or part of a knowledge transfer process have been reviewed. Most reviewed studies place an emphasis on investigating influential antecedents that affect knowledge transfer effectiveness, software quality, and software productivity. Some explore knowledge transfer mechanisms that facilitate the flow of common knowledge to address unstructured situations, investigate knowledge transfer barriers, or affect performance improvement in software projects. Nonetheless, how to drive knowledge transfer into action is scarce. The findings also reveal that knowledge transfer in software development can be viewed as a communication process between the source and the recipient engaged in teams through communication channels for their learning and applying knowledge.

Drawing on a connectionistic perspective and communication-based knowledge transfer research, this study is based on Szulanski's model. In this model, a transfer process flows through four distinct stages: Initiation commencing with all events leading to the decision to transfer, Implementation commencing with the decision to transfer, Ramp-up commencing when the recipient starts using the transferred knowledge, and Integration commencing after the recipient achieves satisfactory outcomes. The findings reveal that the transfer process should consist of six components: problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes. In each stage of the transfer process, a set of these components interact with others as multi-directional and play an important role depending upon the stage's functionality.

Based on the significance of the findings, the second step was to develop a knowledge transfer framework. The framework represents a clearer understanding of knowledge transfer primarily involved in transferring knowledge into action which could serve as guidance for planning knowledge transfer activities. In the framework, the comprehensive descriptions of the six components have been presented. Under each component, a list of activities has been designed. Under each activity, a list of key questions that should be considered has also been suggested. Owing to relationships between knowledge transfer components and stages, an activity flow has been illustrated under each stage.

Starting with the Initiation stage, it is triggered by all events leading to the decision to transfer knowledge. A transfer begins when the required knowledge meets a need. The discovery of the need leads to the search for valuable knowledge, which in turn triggers to seek potential solutions. However, there are antecedents that make difficulties to initiate this stage (e.g., difficulties of knowledge contents). Hence, antecedents must be assessed in order to reduce failure chances. From this view, there are eight activities that are involved in this stage and pertinent to the components of problems, antecedents, knowledge, and mechanisms. An activity flow of this stage begins with defining goals and objectives, identifying and prioritizing problems, defining knowledge, classifying knowledge, assessing

knowledge, locating knowledge, assessing antecedents, and developing a plan. Once the potential solution planning is done, the actual activities can then be executed directly through the Implementation stage.

The Implementation stage begins with the decision to transfer the knowledge. When it takes place, the plan should be followed. Moreover, resources flow between the source and the recipient. Hence, antecedents supporting communications between them should be established (e.g. establishing several communication channels, strong commitment, and reward systems). During this stage, the required knowledge is often tailored to suit the expected needs and to pre-empt problems experienced in the past. However, the transfer process may be abandoned or re-initiated if the knowledge is deemed unsuitable. From this view, there are eight activities that are involved in this stage and pertinent to the components of antecedents, knowledge, and mechanisms. An activity flow of this stage begins with developing supportive antecedents, tailoring knowledge, managing antecedents, and managing the plan. If the knowledge is not perceived as suitable, this stage then continues with assessing knowledge and antecedents as well as reviewing problems and the plan.

The Ramp-up starts when the recipient starts using the transferred knowledge. The recipient typically uses the knowledge ineffectively at first, but gradually identifies and rectifies unexpected problems until being able to achieve satisfactory outcomes. During the use, the recipient may request additional support from the source in solving problems. However, the recipient may abandon or re-initiate the transfer process if there are too many difficulties to use it. Thus, impeding antecedents must be removed. From this view, there are 12 activities that are involved in this stage and pertinent to the components of problems, antecedents, mechanisms, knowledge application, and outcomes. An activity flow of this stage begins with making use of the knowledge, focusing on the problems, monitoring and supporting the use of the knowledge, and maintaining antecedents and the plan. After the use of the knowledge with more experience, this stage then continues with auditing the use of the knowledge, reviewing the problems, and assessing the outcomes. If the outcomes are not satisfied, this stage continues with improving the outcomes, assessing antecedents, and reviewing the plan. Once satisfactory outcomes are achieved, the transfer process then flows through the Integration stage.

The Integration stage begins after the recipient gained desired outcomes with the transferred knowledge. Knowledge application and its integration with existing practices gradually become routinized into standard practices. At this stage, the Integration activities are carried out to ensure that the recipient can use the transferred knowledge without any support from the source and can take any remedial action to improve the understanding of the transferred knowledge and integrate it into his/her practices. Moreover, this stage primarily looks at the efforts required to minimize problems and deal with challenges to the routinization of the transferred knowledge. When the knowledge transferred presents too many difficulties, it is unlikely to become part of routines and therefore sustained in a practice. Hence, antecedents must be reviewed and managed. From this view, there are nine activities that are involved in this stage and pertinent to the components of problems, antecedents, knowledge, mechanisms, and knowledge application. An activity flow of this stage begins with integrating knowledge; sustaining, monitoring, supporting and auditing the use of the knowledge; assessing and maintaining antecedents; as well as reviewing the problems and the plan. Once the recipient can integrate the transferred knowledge into his/her knowledge packages and use it without any support, the transfer process is then recognized as successful.

It is also important to demonstrate how to apply the framework in real-life software projects, which was the third step. Owing to time limitations of this study, we have to use the findings of our case studies in Chapter 5 for demonstrations. The demonstration shows that most of the transfer processes were recognized as successful as the participants were satisfied with the transferred knowledge (i.e., the software process maintenance framework), integrated it into their existing practices, and sustained it in their teams. However, they required time to apply the transferred knowledge effectively. However, there were some transfer processes recognized as ineffective due to insufficient support from the source, a lack of motivation, a lack of absorptive capacity, knowledge perceived difficult, a lack of commitment in terms of time, a lack of intensive communications, and strong embeddedness of their existing practices. Some were recognized as failure due to incapability between the transferred knowledge and their organizational practice and no readiness to apply the transferred knowledge. Therefore, the participants should pay more attention to these difficulties for improving their knowledge transfer on further software projects. The demonstration also shows the great compatibility between the framework and Scrum-oriented software development. In the future, we hope to further improve our knowledge transfer framework by carrying out empirical case studies in the Thai telecommunications industry which is the first focus of this study. Since the framework provides a general conceptual lens of knowledge transfer in software development; in case positive results are gained, empirical case studies with an improvement of the framework may be performed in other industries. This is in order to increase the generalizability of the framework.

Chapter 8

The Framework for Transferring Software Project Management Approaches into the Thai Telecommunications Industry

This chapter proposes a framework for transferring software project management approaches into the Thai telecommunications industry which itself consists of two components: the developed software process maintenance framework presented in Chapter 4 and the developed knowledge transfer framework presented in Chapter 7. The framework aims at contributing to the improvement of software development performance by providing a comprehensive set of project management, software development, and knowledge transfer processes. Giving a better understanding, this chapter then demonstrates the application of the framework, based on the findings of case studies in Chapter 5.

8.1 Introduction

The development of a software project requires efficient and effective software development processes (hereafter referred to as "software processes"), stakeholders' expertise and experience, knowledge transfer activities, and the ability to transfer, acquire, and apply knowledge to solve problems occurring during software development [29]. These elements significantly lead to quality of software development and software products. Achieving this quality, a framework for transferring software project management approaches into the Thai telecommunications industry is required, which aims at contributing to the improvement of software development performance. This leads to the following research question.

RQ8-1: How should a framework for transferring software project management approaches into the Thai telecommunications industry be constructed?

In the framework, there are two components which are frameworks themselves: the developed software process maintenance framework presented in Chapter 4 and the developed knowledge transfer framework presented in Chapter 7. First, the proposed software process maintenance framework in this context means a framework for software process development and improvement. It aims at providing the "what" to improve with a software process assessment mechanism and the "how" to implement with and a comprehensive set of project management and software development processes. The framework consists of two components which are a Software Development Maturity (SDM) model and an integrated PMBOK-Scrum model. This study intends to minimize changes of the software processes which software development teams (hereafter referred to as "teams") are already familiar with. Therefore, the SDM model has been constructed to provide what software processes need immediate and sustainable improvement, based on Capability Maturity Model Integration (CMMI) and Critical Success Factors (CSFs) approaches. The integrated PMBOK-Scrum model has been constructed to provide how to implement

software processes, based on Project Management Body of Knowledge (PMBOK) and Scrum approaches.

Second, the proposed knowledge transfer framework aims at providing guidance for planning knowledge transfer activities. The framework describes six components and four stages of knowledge transfer. The six components consist of problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes. "Problems" lead to knowledge transfer and in turn help teams to define what "Knowledge" is required and what "Mechanisms" fit their software development contexts. "Antecedents" in this study mean determining factors of the ease or difficulty of knowledge transfer. Weak antecedents lead to new problems, whilst supportive antecedents enable satisfactory outcomes. In addition, designing and selecting "Mechanisms" depends upon required knowledge and software development environments. Suitable mechanisms lead to transfer effectiveness; otherwise, unexpected problems may occur and expected outcomes are unlikely to be achieved. In addition, "Knowledge Application" can bring about knowledge retention and may lead to a new consideration of the underlying problem or the identification of new problems. When satisfactory "Outcomes" are achieved, sustaining knowledge use is more likely to take place. This shows that these components are connected with others through a multi-directional set of interactions and occur at the same or different times and more than once. Based on Szulanski's model, the knowledge transfer process flows through four distinct stages starting with Initiation beginning with all events leading to the decision to transfer, Implementation beginning with the decision to transfer, Ramp-up beginning when the recipient starts using the transferred knowledge, and Integration beginning after the recipient achieves satisfactory results. Owing to the relationships between knowledge transfer components and stages, a list of activities has been designed under each component. A list of key questions that should be considered has been suggested under each activity. A flow of relevant activities has also been illustrated under each stage. Moreover, guidance on how to apply the software process maintenance framework and the knowledge transfer framework together in real-life software projects is also important. This leads to the following research question.

RQ8-2: How can the developed framework for transferring software project management approaches into the Thai telecommunications industry be performed?

In consequence of time limitations of this study, we could not carry out empirical case studies in the Thai telecommunications industry. However, we use our prior case studies in two Thai telecommunications companies (i.e., CAT Telecom Public Company Limited and TOT Public Company Limited) (presented in Chapter 5) as a base for a demonstration. This chapter is organized as follows. The following section presents the developed framework for transferring software project management approaches into the Thai telecommunications industry. This is then followed by the descriptions of the application of the framework.

8.2 The Framework for Transferring Software Project Management Approaches into the Thai Telecommunications Industry

Quality software processes and products require an efficient and effective software process and a knowledge transfer process. Thus, a framework for transferring software project management approaches into the Thai telecommunications industry can be depicted in Figure 8-1.

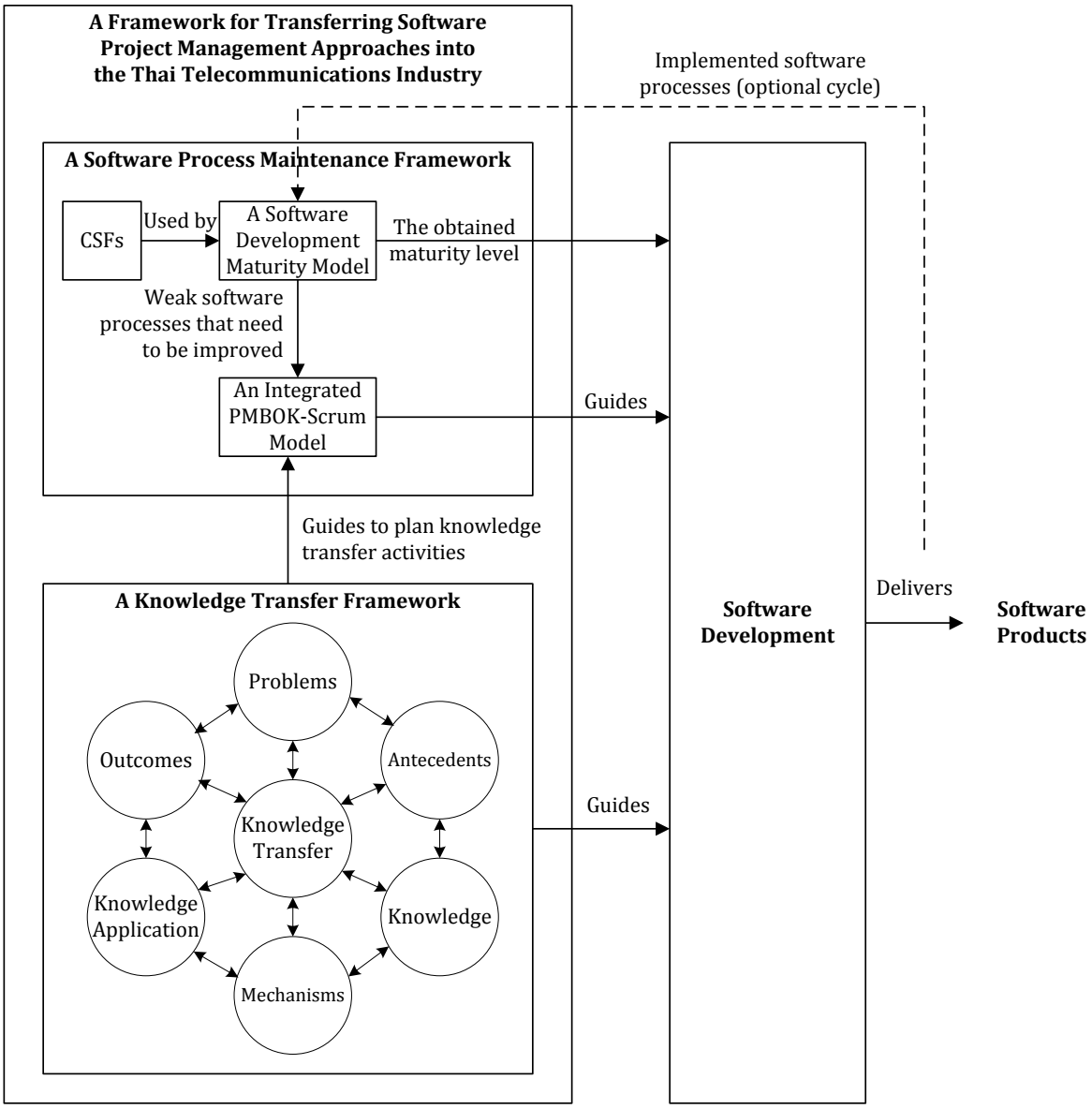


Figure 8-1. The proposed framework for transferring software project management approaches into the telecommunications industry

Answering the RQ8-1 "How should a framework for transferring software project management approaches into the Thai telecommunications industry be constructed?", we

connect together a software process maintenance framework presented in Chapter 4 and a knowledge transfer framework presented in Chapter 7 into an umbrella framework. The software process maintenance framework advocates software process improvement though the SDM model and providing a comprehensive set of project management and software development processes through the integrated PMBOK-Scrum model; while the knowledge transfer framework provides guidance for planning knowledge transfer activities and transferring knowledge in action. To apply the two frameworks together, there are four main steps which are outlined as follows.

Step 1: Assessing the existing software process. Starting with the software process maintenance framework, practitioners can first evaluate their existing software processes using the SDM model. The obtained results can then be used for setting software project's goals and objectives and planning for improving the software processes and products.

Step 2: Planning a software process and a knowledge transfer process. In this step, the integrated PMBOK-Scrum model and the knowledge transfer framework can be used together to create a software project plan by defining a necessary set of project management, software development, and knowledge transfer processes. As software development in this study is executed in the Scrum way, this step is divided into two sub-steps: (i) Scrum planning meetings and (ii) sprint planning meetings.

Step 3: Executing the plan. In this step, the software project plan is executed, inspected, and adjusted through iterations until the completion of the software project, using the integrated PMBOK-Scrum model and the knowledge transfer framework as guidance. This step is divided into two sub-steps: (i) Scrum executions consisting of daily meetings and executions and (ii) sprint review and sprint retrospective meetings.

Step 4: Evaluating the implemented software process and closing the software project. It is the final step that is reached when the final software product is rendered ready for release and distribution. In this step, many project closure activities may be executed, e.g., ensuring the work of the software project being acknowledged, obtaining user acceptance, conducting post-project review, recording impacts of tailoring to any software processes, documenting lessons learned for use on further software projects, and closing out procurements [38]. It is an optional cycle that the implemented software processes may be again evaluated through the SDM model in order to compare the overall performance between before- and after- software development.

For a better understanding, the next section demonstrates the application of the framework for transferring software project management approaches into the Thai telecommunications industry (hereafter referred to as "the umbrella framework").

8.3 Application of the Framework for Transferring Software Project Management Approaches into the Thai Telecommunications Industry

Owing to time limitations of this study, we could not evaluate the usability and practicality of the umbrella framework in real-life practice. Answering the RQ8-2 "How can the developed framework for transferring software project management approaches into the Thai telecommunications industry be performed?", our prior case studies in two Thai telecommunications companies (i.e., CAT Telecom Public Company Limited (CAT) and TOT Public Company Limited (TOT)) (presented in Chapter 5) are used to demonstrate the application of the umbrella framework. The data collection was carried out through on-site observations, individual interviews, and questionnaires during November 2010 - February 2011 (see Section 5.2 in Chapter 5 for more details).

Based on the findings in accordance with the application of the software process maintenance framework presented in Chapter 5 and the application of the knowledge transfer framework presented in Chapter 7, the application of the umbrella framework can be described in the following four steps.

Step 1: Assessing the existing software process (as illustrated in the blue area in Figure 8-2) describes the maturity of the software processes implemented in the CAT and TOT teams and the software processes required immediate improvement.

Step 2: Planning a software process and a knowledge transfer process (as illustrated in the light green area in Figure 8-2) describes how practitioners set up and planned a software project under two sub-steps: Scrum Planning Meetings and Sprint Planning Meetings.

Step 3: Executing the plan (as illustrated in the yellow area in Figure 8-2) describes how practitioners managed and developed the software project as well as validated and verified the software products under two sub- steps: Sprint Executions and Sprint Review and Retrospective Meetings.

Step 4: Evaluating the implemented software process and closing the software project (as illustrated in the red lines in Figure 8-2) is the final step that is reached when the final software product is made ready for delivery. Project closure activities (e.g., obtaining user acceptance, completing project records, and documenting issues and lessons learned) are performed. In order to compare and record the overall performance of software development, the implemented software processes through the SDM model may be evaluated again.

Figure 8-2 illustrates a four-step flow of our umbrella framework.

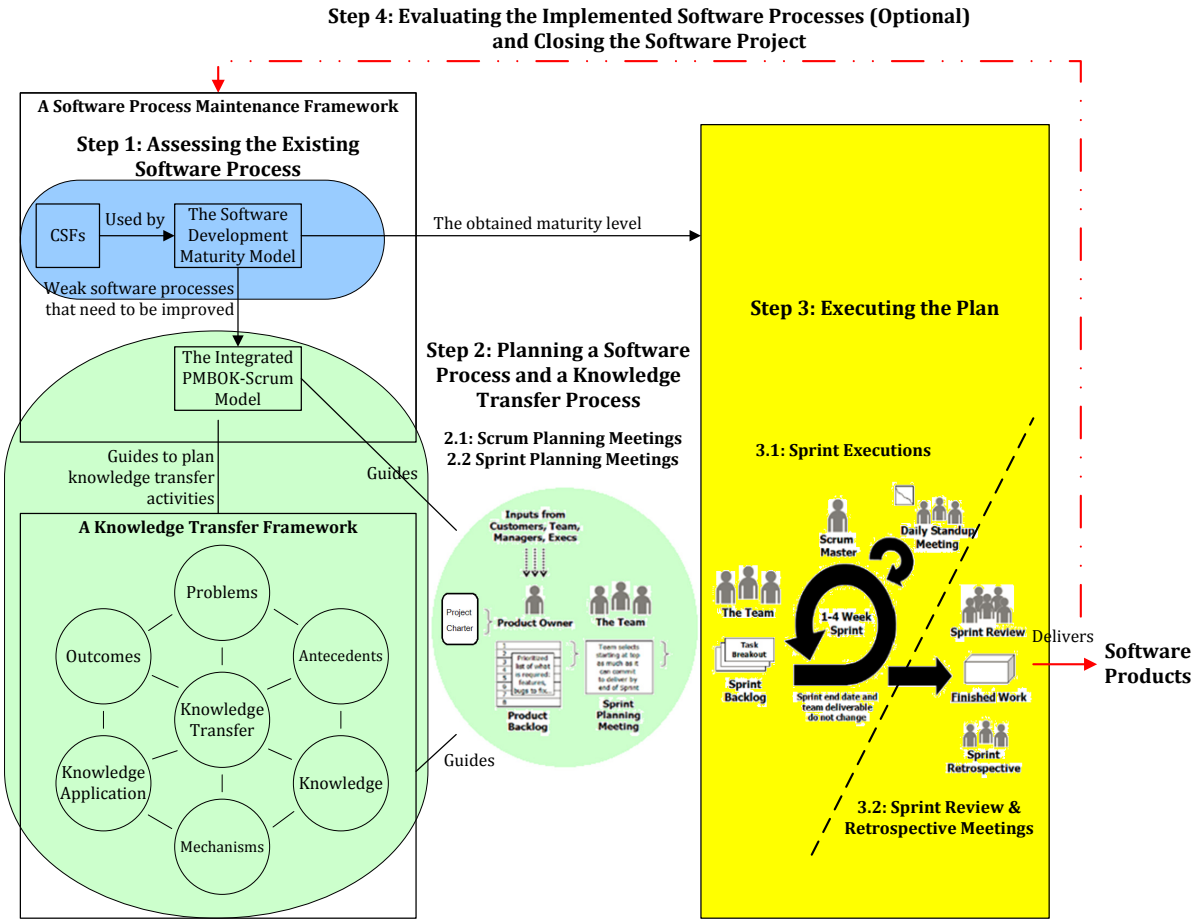


Figure 8-2. A four-step flow of the framework for transferring software project management approaches into the telecommunications industry (Scrum's source: [102])

Step 1: Assessing the Existing Software Process

Team members carried out together an assessment of their existing software process before developing software, using the SDM model. At the threshold of 7 for an assessment calculation as guided by Motorola [215], the assessment results reveal that the CAT team stood at the maturity level 2-"Managed" of the SDM model. This is because only one factor "Team Size" belonging to the maturity level 3-"Defined" was not fully implemented and the factor belonging to the maturity level 4-"Optimizing" was not focused at all. On the other hand, the TOT team stood at the maturity level 1-"Initial" of the SDM model. At the case study time, TOT usually used an outsourcing method for their existing software projects. Hence they decided to start from scratch for the case project. Based on the assessment results, the CAT team had to sustain strong CSFs and improve software practices corresponding to the CSFs "Team Size" and "Reviews"; while the TOT team had to improve all CSFs.

Step 2: Planning a Software Process and a Knowledge Transfer Process

Step 2: Planning a software process and a knowledge transfer process describes how practitioners set up and planned a software project into two sub-steps which are Step2.1: Scrum Planning meetings and Step 2.2: Sprint Planning Meetings.

Step 2.1: Scrum Planning Meetings

After assessment, the teams set up and planned the software projects. The case project in CAT developed additional Web-based functionalities bundled into their ongoing software project, while the case project in TOT developed a small decision support application. Both software projects were non-life critical. As CAT used PMBOK for their existing software projects, all related project documents (e.g., project charter, program roadmap, project management plans, and software design) were thus used in the case project. On the other hand, TOT usually used an outsourcing method for their existing software projects and the case project was a new project, all documents required by the case project thus needed to be created. Consequently, the TOT team began the case project with developing a project charter. The project charter includes, e.g., project goals, objectives, characteristics, and stakeholders. Apart from business aspects, the assessment results can be used to define technical goals and objectives with respect to software process improvement and knowledge transfer in software development. Project types and characteristics needed to be assessed in order to understand its criticality and the degree of required project management, as suggested by the SDM model. A stakeholder analysis was also performed. Supporting knowledge transfer, teams should have additionally analyzed what skills, experience, and knowledge each stakeholder had; the degree to which each stakeholder had that knowledge; and each stakeholder's characteristics in terms of motivation, credibility, capability, absorptive capacity, relationships amongst team members, and commitment. These characteristics can be used as knowledge transfer antecedents. Antecedents in this context mean determining factors of the ease of difficulty of knowledge transfer. Once the project charters were approved, the case projects were then formally authorized.

Step 2.2: Sprint Planning Meetings

The Initiation stage of a knowledge transfer process is triggered by all events leading to the decision to transfer, e.g., the search for problems, necessary knowledge, and potential solutions. Hence, knowledge transfer activities of the Initiation stage can be performed at sprint planning meeting parts I and II.

A. Sprint Planning Meeting Part I

Albeit the CAT team had been developing an ongoing software project for almost two years, a project roadmap was not enlightened to the team. At the beginning of both cases, the product owners explained the project roadmap in order to draw the team a big picture. It was recognized as an important practice that can drive the team into the right direction. When observed and asked what project management aspects were considered, the TOT team followed PMBOK guidance by planning integration and configuration management in a simple way; while the CAT team used their existing integration management plan.

The CAT team was not concerned about whether the scope would become broader since they had continued enhancing the application's functionality. However, in order to prevent any risk from enlarging the project's scope, the CAT team considered the capacity of network and application architecture. On the other hand, the TOT team developed a simple scope management plan to prevent scope creep, used together with product backlogs. During gathering user and technical requirements, both teams may have used those requirements, the obtained assessment results, and the defined goals and objectives to identify potential problems (as knowledge transfer requirements) for software process and product improvement. Typically, business, organizational, and technological knowledge is transferred during this step. All requirements were then logged into a product backlog with estimated effort in terms of time, and prioritized using a relative weighting approach.

After the verification of the scope, the identified requirements were then scheduled into iterations. Time was also not a major constraint in both teams. Rather than attempt to build the entire application from ground zero or through long-term iterations, both teams used small iterations lengthening between 1-4 weeks. The CAT team initially started using 2-week iterations due to time given for learning agile during the development, but later preferred one-week iterations to deliver features. The main reason was to keep work motivation. On the other hand, the TOT team initially used 2-week iterations. However, it was not enough to produce a meaningful functionality due to two main reasons as claimed by the Scrum master. First, they did not well assess the appropriate techniques and tools before using them, nor did they analyze the data quality from the source systems before development. Second, the Scrum master needed to transfer programming techniques to the developer in the team. Hence, they needed more time to create meaningful and valid features. As knowledge training and coaching requires time, adequate independence from the software development tasks needed to be provided to team members involved in the knowledge transfer. This shows that the amount of user and technical requirements needed to be implemented in each iteration was sometimes reduced.

Owing to internal development, both teams did not emphasize cost management. To guarantee software quality, the CAT team followed the Scrum validation and verification ways; whilst the TOT team used a simple PMBOK quality management plan and managed through sprint reviews. Concerning human resource aspects, the CAT team was formed with the same team responsible for the existing ongoing software project, composing of a product owner, a Scrum master (also acted as a developer), a developer and a tester. This helped to reduce time to learn business logic, programming languages, and development tools. On the other hand, the TOT team was formed with only two members, i.e., a Scrum master who had multi-projects and multi-roles (i.e., product owner, developer, and tester) and a developer, due to the small size of the software project and their available resources at that time. To accelerate software development, the product owner in CAT had full authority to make decisions but not in TOT. Team members in TOT had much experience in telecommunications but not software development; whilst team members in CAT had 7-15 years of experience in software development. This supports that the degree of transfer of software development knowledge required for the TOT team was higher than that required for the CAT team.

Both teams followed the Scrum communication mechanism through sprint planning meetings, daily meetings, and sprint review meetings; except through sprint retrospective meetings only in the CAT team. Considering development environments, the CAT team worked approximately 60% in co-location and approximately 40% in distributed sites; while the CAT team worked fully in co-location. Both teams cultivate informal communication for

collaborating on work and transferring knowledge. Creating more chances of communication, the CAT team established several communication channels, e.g., mobiles, phones, face-to-face communications, emails, instant messaging, and e-conferencing. Since management in TOT had a heavy workload due to multi-projects, the team was not fully approachable. Dealing with this situation, they used approximately 70% for mobiles and only 30% for face-to-face communications. Both teams used non face-to-face media for necessary explanation and feedback when the product owners were remote and for technical knowledge exchange when team members worked in different sites. This enabled them to obtain quick feedback.

Moreover, the CAT team planned risk management with short-term and long-term solutions using risk and impediment backlogs; whilst the TOT team created a simple PMBOK risk management plan for the overall project and used risk and impediment backlogs for iterations. The main reason for this, as observed in the TOT team, was that they preferred to get familiar with PMBOK risk management for further complex software projects. Those plans were continuously reviewed and adjusted during the case projects. Concerning procurement management, it was not performed in both case projects; however, both teams were planning to acquire outsourcing teams for future software projects.

Once all related project management plans were finished and the first set of prioritized requirements was obtained to be implemented, the software development flowed into part II of the sprint planning session.

B. Sprint Planning Meeting Part II

Iteration's goals and objectives were described to team members. All requirements were then considered. For user and technical requirements, they were broken down into tasks and logged into a sprint backlog with estimated task efforts and assigned responsible persons. As observed, this practice seemed to provide smaller and manageable tasks to the teams. For each knowledge transfer requirement, the teams should have defined and classified what knowledge and what type of the knowledge was needed for transfer. In this case, a knowledge transfer requirement may be either a user requirement or a technical requirement. According to eTOM business process areas, software applications developed in both CAT and TOT teams can be classified into the Strategy, Infrastructure & Product (SIP) process area. It means that five knowledge types (i.e., human, organizational, relational, project management and technological knowledge) may have all been required for transfer. The teams should have assessed the required knowledge by considering its values, complexity, and accessibility and then selected the most suitable knowledge. For instance, the CAT team intended to apply a standard software process improvement method (e.g., Capability Maturity Model-CMM, Capability Maturity Model Integration-CMMI, Software Process Improvement and Capability Determination-SPICE, and Six Sigma). Owing to high suitability for their business purposes and having its knowledge source in their team, the CAT team planned to be certified in CMMI.

After getting suitable knowledge, teams should have found a credible source having such knowledge and located it in teams. Knowledge sources can be both inside and outside team members (e.g., developers and consultants) and non-human (e.g., project documents, organizational policies, and information systems). For instance, the Scrum master in CAT having CMMI knowledge may have been assigned to be a knowledge source. However, the knowledge transfer requirement being considered may be discarded or re-assessed if such knowledge cannot be located. It may not be plausible to teach everybody for every task.

Therefore, target recipients need to be defined. Suitable communication channels or Information and Communication Technologies (ICTs) for transfer should have also been planned. This may have been based on the communication management plan. The details of the source, the recipients, their roles and responsibilities, the knowledge areas, and the ICTs should have been logged into sprint backlogs and clearly clarified to all related team members. After breaking down all requirements into tasks and scheduling them, a sprint Burndown chart was developed.

Both teams identified and prioritized risks and impediments, and then logged them into backlogs with short-term and long-term solutions. Apart from software development aspects, management should have been assessed influential antecedents to understand what facilitated and hindered a knowledge transfer process. They may have used the results of the stakeholder analysis together with the current observation results for measuring influential antecedents in the source (i.e., motivation, capability, and credibility), recipient (i.e., motivation and absorptive capacity), and relational (i.e., commitment and relationship between the source and the recipient) contexts. Hindering antecedents may be logged into an impediment backlog with strategic solutions. As observed, management (especially in the TOT team) as a main source having multi-roles and multi-projects led to inadequate communications with the recipient and a lack of commitment in terms of time. Moreover, a developer in the TOT team as a recipient lacked absorptive capacity due to less prior software-development-related knowledge, while developers in the CAT team as recipients somewhat lacked motivation or interest in project management knowledge that could in turn led to a lack of absorptive capacity to learn and use that knowledge. This shows that teams had to build up the extent of communication, commitment, absorptive capacity, and motivation.

It is essential to prepare materials for software development and knowledge transfer. For knowledge transfer, the source may have (1) put together a package for their areas of expertise into forms of, e.g., documentation and presentation slides, and (2) prepared transfer environments, e.g., programming environment in the TOT case project. Once the planning and resources were done, the actual software development and knowledge transfer was directly executed in the sprint execution. For the knowledge transfer process, it flows from the Initiation stage through the Implementation stage. It is important to note that some of actual knowledge transfer activities may be performed during the sprint planning without actual planning, e.g., on-the-fly transfers of how to break down the work into tasks.

Step 3: Executing the Plan

Step 3: Executing the plan describes how practitioners managed and developed the software project as well as validated and verified software products. The descriptions are divided into two sub-steps which are Step 3.1: Sprint Executions consisting of Daily Meetings and Executions and Step 3.2: Sprint Review and Retrospective Meetings.

Step 3.1: Sprint Executions

For the knowledge transfer process, resources flow between the source and the recipients during the Implementation stage. Once the recipients start using the transferred knowledge, the transfer process then flows through the Ramp-up stage. Hence, transfer

activities of the Implementation and Ramp-up stages can be executed through sprint executions.

A. Daily Meetings

Daily meetings are places to coordinate work, synchronize efforts, and tackle anticipated problems. The meetings took place around 5-15 minutes with non-fixed place and time in both teams. When asked how to perform the three Scrum daily-meeting questions, the daily-meeting questions of "What did you do yesterday?" and "What will you do today?" were not asked every day. One issue we found is that the Scrum master in CAT felt *"Asking these two questions every day seems like micromanaging or not having confidence in the team."* Hence, he asked these two questions approximately few times a week. This implies that management did not perform strong micromanagement in their existing software projects. In contrast, a developer in CAT *said "It's a normal thing to do."* However, both CAT and TOT teams emphasized on the occurring impediments, which are related to the third daily-meeting question of "What impediments are in your way?". In the view of product owners, these questions' discussion was recognized as important. Noticeably, the product owner in CAT encouraged and facilitated support to the team to perform it. During the meetings, management should have also observe hindering antecedents (e.g., personal problems, conflicts within teams, inability to learn and apply the transferred knowledge) in order to minimize chances of knowledge transfer failure. After discussing on the three questions and observing the current situation in the teams, all related plans (e.g., project management plans, sprint backlogs, risk backlogs, impediment backlogs, and Burndown charts) were adjusted.

B. Executions

Management in both teams managed each iteration based on their management plans. For knowledge transfer aspects, impediment backlogs and related management plans (e.g., human resource and risk management plans) may have been used to develop supportive antecedents. For instance, communication, absorptive capacity, motivation, and commitment needs enhancement. Both teams realized that only face-to-face conversations could not be held for all software processes. They thus increased the volume of communications by using other media, e.g., phones and instant messaging. Absorptive ability may have been enhanced by allowing them to learn by doing and making a mistake. Learning by doing can strengthen the understanding of tasks through gaining accumulative practical experience. The source who feels threatened to lose his/her importance or authority is likely to demonstrate non-cooperative behavior with the recipient. A clear vision of his/her future may be communicated early for motivation. Nevertheless, sources in both teams as observed were greatly willing to transfer their knowledge to their team members. To increase motivation, management in CAT motivated team members through conversations and intended to establish a reward system on future software projects; whilst management in TOT considered the case project results as one of key performance indicators for the annual staff performance appraisal. In addition, it is important to make awareness and get commitment for knowledge transfer from all key stakeholders and team members. Management in both teams should have secured both key stakeholders' and team members' commitment especially through the Implementation and Ramp-up stages of the transfer process.

For a given user and technical requirement, during the coding stage, the CAT team followed their coding standard for having easily maintainable and expandable code, pursued simple design, and used code refactoring to allow for improving existing code to support new functionalities of the software application, as suggested by the SDM model. These practices were not only used for this case study, but also implemented into their existing software projects. They employed a configuration management system for controlling individual check-in, check-out, and continuous integration of their source code and applications. It was also used for supporting quality assurance. Their development environment closely mirrors the production environment to guarantee quality and minimize unexpected risks. Additionally, unit and integration tests were performed against test cases to ensure work completeness.

For a given knowledge transfer requirement, the required knowledge is transferred to the recipients and tailored until suitable for the current software development context. During transferring, the source should assess whether an amount of the required knowledge is sufficient for accomplishing the focused requirement. However, if the knowledge being transferred is neither suitable nor effectively tailored, the transfer process is likely discarded or re-initiated. As observed, the CAT team largely exchanged technological knowledge and discussed on how to adapt it for improving their development techniques and software maintainability. During conducting project documents, organizational knowledge (e.g., organizational templates, standards, and policies) was engaged. Most of those documents were informal and less detailed. This implies that most transferred knowledge was more likely to become human knowledge residing in individual team members. On the other hand, the source in the TOT team used on-the-job training to transfer programming techniques to the recipient. Since the recipient had no experience with the programming language, more time and effort for sharing and learning that knowledge was highly required.

Once the recipients start using the knowledge, the transfer process then flows from the Implementation stage through the Ramp-up stage. During this Ramp-up stage, the recipients gradually ramps up to work performance and satisfaction by using the transferred knowledge to accomplish the requirement. Typically, recipients use the transferred knowledge ineffectively at first. This situation was noticeable in the TOT team. The developer who was received the programming techniques at first use had requested support from the source and taken time to use that knowledge and solve any occurring problems. However, the volume of support is typically decreased when the recipient receives deeper understandings of the transferred knowledge through gaining practical experience. During the recipient's use of the transferred knowledge, the source should also monitor and then audit in order to ensure that the recipient can use it appropriately and effectively. If there are any occurring problems or it is unlikely to accomplish the focused requirement, the transfer may be either discarded or reverted back to the earlier stages (i.e., the Implementation or Initiation stages). In case of reverting back to the Implementation stage, the source should re-tailor the required knowledge to fit into the current software development situations and make use the transferred knowledge again. In case of reverting back to the Initiation stage, the source should re-assess the required knowledge. Once getting the most suitable one, the actual knowledge transfer with the revised related plans is executed again. Nevertheless, some of those problems may be considered as new knowledge transfer requirements. Solutions for those problems may be discussed either through daily meetings or once those problems occur.

Step 3.2: Sprint Review and Retrospective Meetings

In sprint review meetings which are places for showing the team's accomplishment during sprint executions, both teams held approximately 30-90 minutes. The software products and the knowledge transfer outcomes (i.e., work performance in terms of efficiency and effectiveness as well as work satisfaction in terms of perceived usefulness and perceived ease of use) were verified and validated against the sprint backlog.

The product owners then determined which requirements have been completed against acceptance criteria, clarified the team the reasons for work acceptance and rejection, and discussed until all team members accepted with the results and/or solutions for product modification.

In case of team members unsatisfied with the results, the transfer process may be either continued at the Ramp-up stage or reverted back to the earlier stages (i.e., Implementation and Initiation stages). For instance, developers in the CAT team misunderstood user requirements and in turn delivered the wrong work. In this case, the transfer process needed to be reverted back to the Implementation stage. The project owner as the source had to re-transfer the knowledge and ensure developers' understandings by informal oral questions and answers. As observed, if the rejected work could be fixed within approximately 15 minutes, both the Implementation and Ramp-up stages of the transfer process were re-circled during the sprint review meetings. Otherwise, the rejected work and the Implementation and Ramp-up stages of the transfer process needed to be re-executed in the next iterations.

In case of team members satisfied with the results, the transfer activities of the Ramp-up stage will cease and the transfer process then flows from the Ramp-up stage through the Integration stage. The transfer activities of the Integration stage may be discussed in either sprint review meetings or sprint retrospective meetings, and executed in the next iterations. In this study, those activities are discussed in sprint retrospective meetings.

Sprint retrospective meetings are places for lessons learned by discussing on what went well, what did not, what could be improved in the next iterations. The TOT team did not yet concentrate on this kind of meetings due to no sufficient time of the Scrum master. Only the CAT team performed these meetings holding approximately 20-30 minutes.

In case things went well, they should discuss how to integrate the transferred knowledge or what went well into their standard practices, based on the compatibility with, e.g., their organizational standards, regulations, and cultures. That knowledge (or what went well) is then executed in the next iterations. However, if incompatibility is found, that knowledge may be discarded. During the Integration stage, the transferred knowledge is gradually routinized. Hence, it is important to ensure that the recipients can use that knowledge effectively without any the source's support, take any remedial action to better understandings of that knowledge for improving their work performance and products, and assimilate that knowledge into their knowledge packages. Three main activities that the source needs to plan and perform in the next iterations are sustaining, monitoring, and auditing the use of that knowledge. For instance, management in CAT and TOT facilitated supportive environments and defined some of the transferred knowledge as their standard practices to sustain the use of the transferred knowledge. For monitoring, the sources in both teams allowed the recipients to make a mistake and correct it by themselves. However, when the recipients encountered tough problems, the source helped solving that problem with the recipients. For auditing, the use of the transferred knowledge was mostly audited at sprint

reviews and retrospectives through oral questions and answers. This served as a quality check on whether the learning had indeed been taken place. Feedback on these activities should be used to plan remedial activities for further and better use of the transferred knowledge.

In case things did not go well, the related transferred knowledge can be recognized as a failure. In case improvement is needed, the related transferred knowledge is reverted back to earlier stages of the transfer process, depending on the team's purposes and encountered problems. For instance, if that knowledge requires an additional amount to be transferred, that knowledge should be reverted back to the Initiation stage. If that knowledge requires re-tailoring to fit into the current software development circumstances, that knowledge should be reverted back to the Implementation stage. If that knowledge requires continuous use until the recipients achieving satisfactory results, that knowledge should be reverted back to the Ramp-up stage. If that knowledge has already been integrated into their standard practices but still requires continuous use until the recipients can use it effectively, that knowledge can be recognized as being at the Integration stage. However, once the transferred knowledge is routinized effectively, the transfer process can be recognized as successful.

During these two kinds of meetings, all related plans (e.g., project management plans, sprint backlogs, risk backlogs, impediment backlogs, lessons learned, and Burndown charts) were reviewed and adjusted. This iteration was then formally closed and the software development flows into the next iterations.

Step 4: Evaluating the Implemented Software Processes and Closing the Software Project

Before releasing the final products, the teams performed various types of software testing (e.g., integration, system, and user acceptance tests) against test cases and acceptance criteria to ensure software product quality. Many closure project activities (e.g., completing all required deliverables, getting final acceptance of the project results, and documenting project performance, issues, and lessons learned) were also executed. Concerning project performance in both aspects of software development and knowledge transfer, work performance in terms of efficiency and effectiveness as well as work satisfaction in terms of perceived usefulness and perceived ease of use was evaluated as follows.

Efficiency can be measured by software quality. Two key variables used to represent work efficiency in this study are team productivity [233, 234] and achieved doneness. Productivity can be considered by using velocity metrics. Velocity is the amount of requirements (or backlog items) successfully delivered in an iteration. Achieved doneness is a ratio of the amount of the tasks that the product owner accepts over the amount of the tasks that the team said was done at the sprint review. Effectiveness is often associated with doing the right things; therefore, two key variables used to represent software development effectiveness in this study are defect reduction and customer/team satisfaction [233, 234].

In the CAT team, the velocity was increased from 14 in the first iteration to 30 in the last iteration. The achieved doneness increased from 64.29% in the first iteration to 100% in the last iteration. Defects were reduced from 5 in the first iteration to zero in the last iteration. Based on the questionnaire findings, the average rated scores of (1) the increased work productivity, (2) the increased work effectiveness, (3) the increase work performance, and (4) the improved quality of software process and product were 4.33, 4.67, 4.67, and 4.33 out of 5 points. In the TOT team, we used only the questionnaire findings to analyze their work

performance using the same set of variables tested in the CAT team. Their rated scores were 4, 5, 5, and 5 out of 5 points, respectively.

Perceived usefulness refers to the degree to which users/team members believe that using the knowledge and the software products/services would enhance their performance. Perceived ease of use refers to the degree to which users/team members believe that using the knowledge and the software products/services would be free of effort. Based on the interview findings, both CAT and TOT teams were strongly satisfied with their work and the proposed software process maintenance framework; whilst the questionnaire findings reveal that the mean value of perceived usefulness and perceived ease of use rated by both teams were 4.357and 4.2 out of 5 points, respectively.

Concerning project issues and lessons learned, the above results show that better work performance and work satisfaction was gained as direct results; whereas cultivating teamwork as observed was also gained as indirect results of software development. However, both teams were required building up management and team commitment, collaboration, intensive communications, and knowledge sharing environments. Moreover, the findings reveal that some of Scrum's weaknesses can be overcome to some extent in the following knowledge areas: (1) integration management (i.e., providing configuration management and details of many types of testing), scope management (i.e., a clearer sense of product's direction), time management (i.e., improving the predictability of time estimate for the whole project according to the scope management plan), and technical aspects (e.g., data quality techniques, simple design, and code standard as suggested by the SDM model). Owing to small software projects of our case studies, this limits our ability to argue whether or not some Scrum's weaknesses (i.e., limited support for high quality assurance, large teams, outsourcing, and accurate cost estimate for the whole project) can effectively be overcome by the software process maintenance framework. Besides, the software processes implemented in both teams indicate that the software process maintenance framework partially conforms to approaches offering similar features (e.g., project management and software development processes, continuous software process improvement, coding standards, simple design, refactoring, and continuous integration), e.g., CMMI and eXtreme Programming (XP); whereas the knowledge transfer framework is greatly compatible with scrum-oriented software development. At this stage, the umbrella framework partly promises an improvement of software development performance, as a result of the software process maintenance framework. In other words, we cannot yet give assurances about the knowledge transfer framework component due to time limitations of this study for evaluating the knowledge transfer framework in real-life practice.

Furthermore, practitioners may evaluate the implemented software processes through the SDM model in order to compare the overall performance between before- and after-software development. However, software process improvement is a long-team approach, similarly to CMMI. It requires approximately 4.5-24 months for moving from one maturity level to an higher one [227]. Owing to the short project duration of software development in both teams, they decided not to perform an assessment of their implemented software processes after software development.

8.4 Summary

Quality software development requires an efficient and effective software process and a knowledge transfer process. Albeit agile software development methods offering effective software development processes are available, they provide limited project management processes (e.g., procurement management and high software quality assurance). To enable agile software development processes to be more efficient, an adequate set of project management processes is thus required. Moreover, software development is a knowledge-intensive activity and its project consists of people with varying backgrounds and knowledge. Hence, guidance on getting knowledge transfer into actions in software development is also necessary. Since many problems pertinent to software development, project management, and knowledge transfer (e.g., ill-defined requirements, a lack of project management competence, a lack of teamwork, and a lack of provision and support of training to teams) have been found in the Thai telecommunications industry which is the first focus of this study and available solutions dealing with these problems are still required, this chapter has therefore proposed a framework for transferring software project management approaches into the Thai telecommunications industry to fulfill this gap.

The framework aims at contributing to the improvement of software development performance in terms of efficiency and effectiveness. The framework consists of two components which are frameworks themselves: the software process maintenance framework presented in Chapter 4 and the knowledge transfer framework presented in Chapter 7. First, the software process maintenance framework aims at providing the "what" to improve through the SDM model and the "how" to implement integrated project management and software development processes through the integrated PMBOK-Scrum model. Second, the knowledge transfer framework aims at providing guidance for planning knowledge transfer activities. It has been developed, based on Szulanski's model. In this study, knowledge transfer can be defined as a dyadic process between the source and the recipient engaged in teams through communication channels for their learning and applying software-development-related knowledge. In the framework, a knowledge transfer process consists of six components (i.e., problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes) and flows through four distinct stages (i.e., Initiation, Implementation, Ramp-up, and Integration). In the transfer process, the six components are connected with each other through a multi-directional set of interactions and play an important role in different stages. To provide guidance on planning the transfer process into action, a set of activities under each component are guided as an activity flow in each stage.

This chapter has then described how to apply the frameworks through four main steps. The first step starts with the software process maintenance framework. Practitioners can first assess their existing software processes, using the SDM model. The obtained assessment results can be used for setting technical goals and objectives and planning for improving quality software processes and products. Second, the integrated PMBOK-Scrum model and the knowledge transfer framework can then be used together to create a software project plan by defining a necessary set of project management, software development, and knowledge transfer processes. Third, the software project plan can be then executed, inspected, and adjusted through iterations until the completion of the software project, using the integrated PMBOK-Scrum model and the knowledge transfer framework as guidance. Fourth, it is the final step reached when the final software product is rendered ready for release and distribution. The software project is then closed with many project closure activities, e.g., obtaining user acceptance, conducting post-project review, and closing out procurement.

Besides, it is optional that the implemented software processes can be evaluated again through the SDM model after the software development. This is in order to compare the overall performance of before- and after- software development.

Giving a better understanding, this chapter has demonstrated the application of the frameworks in real-life software projects, based on the findings of case studies in Chapter 5. The overall results reveal that work performance in terms of efficiency and effectiveness was increased. Work satisfaction in terms of perceived usefulness and perceived ease of use was perceived. Not only was the increased work performance and work satisfaction gained as a direct result, but also cultivating collaborative teamwork, informal frequent communications, and knowledge sharing culture were also gained as in directed results. However, both teams were required building up management and team commitment, collaboration, intensive communications, and knowledge sharing environments. Moreover, the findings reveal that some of Scrum's weaknesses can be overcome to some extent in the following knowledge areas: (1) integration management (i.e., providing configuration management and details of many types of testing), scope management (i.e., a clearer sense of product's direction), time management (i.e., improving the predictability of time estimate for the whole project according to the scope management plan), and technical aspects (e.g., data quality techniques, simple design, and code standard as suggested by the SDM model). Owing to small software projects of the case studies, this limits our ability to argue whether or not some Scrum's weaknesses (i.e., limited support for high quality assurance, large teams, outsourcing, and accurate cost estimate for the whole project) can efficiently and effectively be overcome by the software process maintenance framework. Based on these findings, we conclude that the problems in the industry as identified in Chapter 2 were to some extent solved successfully; whilst the process transferring new knowledge (i.e., the software process maintenance framework) to the case study teams was perceived as successful.

Because of time limitations of this study, the umbrella framework has not yet been fully evaluated in real-life software projects. Based on only the full evaluation results of the software process maintenance framework, we cannot yet give assurances about the knowledge transfer framework component. However, the software process maintenance framework promises an improvement of software development performance in terms of efficiency and effectiveness; whilst the knowledge transfer framework was designed and constructed based on the positive evaluation results of the first framework. Therefore, there is a great likelihood that the usability and practicality of the knowledge transfer framework can be perceived. In the future, we hope to carry out additional practical tests of the frameworks before finally handing them over to industry partners. Since the frameworks provide a general conceptual lens of software development and knowledge transfer; in case positive results are gained, case studies with an improvement of the frameworks may be performed in other industries. This should help increase the generalizability of the results of this study.

Chapter 9

Conclusions

9.1 Summary of Findings

Chapter 2: In this study, we first investigated software development situation in the Thai telecommunications industry, which is the main focus of this study. We used interviews with two in-house and outsourcing software development teams (hereafter refer to as "teams") working in two of the largest Internet services companies in Thailand. The findings reveal that typical problems (e.g., a lack of project management competence, a lack of management commitment, a lack of training support, and a lack of knowledge transfer) still exist. These problems can be classified into two categories that are software development processes (hereafter refer to as "software processes") and knowledge transfer processes. Dealing with these problems, we have proposed a framework for transferring software project management into the Thai telecommunications industry. It consists of two components which are frameworks themselves: a software process maintenance framework and a knowledge transfer framework. Based on the findings, we have identified two sets of Critical Success Factors (CSFs) as requirements for the sound development of the frameworks. The first set consists of 12 CSFs for the software process maintenance framework, aiming at improving software development performance in terms of efficiency and effectiveness. They are agile software development process, appropriate methods, techniques, and tools, data quality, management commitment, organizational environment, project management process, project type, team capability, team environment, team size, training support, and user involvement. The second set consists of 11 CSFs for the knowledge transfer framework, assisting in organizing knowledge transfer during software development. They are a source's motivation, a source's capability, a source's credibility, a recipient's motivation, a recipient's absorptive capacity, usefulness of knowledge and its ease of use, good relationship, commitment, extensive communication, and organizational culture.

Chapter 3: For improving software development performance in terms of efficiency and effectiveness, we have delved into the prior literature that forms the foundation of the proposed software process maintenance framework. This performed through a systematic literature review. As the framework in this study means a framework for continuous Software Process Improvement (SPI) and development, the systematic literature review thus focused on two parts: agile software development integration with SPI and with traditional project management.

In the first part of agile software development integration with SPI, there are many well-known SPI methods, e.g., Capability Maturity Model Integration (CMMI), Six Sigma, and Control Objectives for Information and Related Technology (COBIT). CMMI is the most suitable for this study. This is because CMMI aims to optimize the development activity in every stage for improving software process and product quality. Looking into what existing research results we can build on, the findings show that most of the reviewed papers propose SPI mechanisms by mapping CMMI key processes with agile practices, especially Scrum. According to the CMMI and Scrum, the findings show the positive theoretical and empirical results of blending CMMI levels 2, 3, and 5 to Scrum practices. We also found that there is

only one CSF "reviews" that is vital to achieving CMMI level 5. This helps us to design the SDM model, emphasizing on these four maturity levels. Second, some researchers suggest that an agile method should be adopted as prerequisite to CMM/CMMI. As the first step to move towards SPI is software process assessment, this leads us to conduct an assessment approach to guide practitioners to improve their agile software process and prepare for achieving CMMI-based process improvements in future. From this point of view, there were two interesting aspects that those research results do not cover yet. First, there is no study emphasizes on dealing with CSFs in order to get agile software processes continuously improved and become more mature. Second, there is no study provides guidance to cope with Scrum weaknesses in both managerial and technical aspects. Bridging these gaps, a search for agile practices to fulfill Scrum's managerial and technical weaknesses is required. Those agile practices need to be mapped with the related CSFs in order to guide practitioners on how to implement the CSFs though agile practices. These results were used to design and develop Software Development Maturity (SDM) model.

In the part of agile software development integration with traditional project management, the findings reveal that integration of agile and traditional processes can overcome agile shortcomings and achieve software development efficiency and effectiveness. With agile, Scrum is the most widely used. With tradition project management, Project Management Body of Knowledge (PMBOK) is recognized for being used more than Projects in Controlled Environments (PRINCE2). As this study aims at minimizing changes that teams are already familiar with, Scrum and PMBOK are thus considered for this study. Although the findings reveal that there is a great possibility to apply PMBOK in agile software development (i.e., Scrum software development in particular), all of the reviewed papers neither specifically offer a theoretical integrated PMBOK-Scrum model nor apply it in real-life software projects. Bridging this gap, a theoretical integrated PMBOK-Scrum model was required. Two proposed models were used to construct the software process maintenance framework as two core components.

Chapter 4: To bridge the gaps identified in Chapter 3, the software process maintenance framework was constructed to assist in providing the "what" to improve through an SDM model and the "how" to implement software processes through an integrated PMBOK-Scrum model. The SDM model was created with a threefold objective: to appraise an organization's current software process through the identified CSFs, to get the current maturity level rating from the model, and to identify which software processes demand immediate and sustainable improvement. The SDM model consists of three dimensions: maturity stage, CSFs, and assessment. First, the maturity stage dimension contains four CMMI-based maturity levels: "Level 1-Initial", "Level 2-Managed", "Level 3-Defined", and "Level 4-Optimizing". Second, the CSF dimension contains 13 CSFs affecting the successful agile software development (i.e., 12 CSFs identified in Chapter 4 and the additional CSF of "reviews" identified in Chapter 3). Based on the perception of CMMI process area division amongst different CMMI maturity levels; the identified CSFs were categorized into three categories: foundation, standardization, and support. The foundation category contains the CFSs that support to establish project management processes, necessary process discipline, and commitments amongst key stakeholders. It can be linked to the maturity level-2 "Managed". The standardization category containing the CSFs that support the design of systematic structures can be linked to the maturity level-3 "Defined". The support category containing CSFs to support continuous SPI activities can be linked to the maturity level-4 "Optimizing". As a guide on how to implement the CSFs, a list of agile practices has been designed under each CSF. These agile practices consist of 67 agile practices in total. They were derived from the findings of a literature survey on worldwide agile software projects

and a questionnaire-style information collection on local agile software projects in three Thai companies. Third, in the assessment dimension, an assessment instrument successfully developed and tested at Motorola has been adapted to assess agile practices. This instrument can be applied at many levels, e.g., organization, department, and project levels. The results of the SDM model can be used to guide practitioners on their current software development maturity and weak practices that demand immediate and sustainable improvement.

The integrated PMBOK-Scrum model aims to assist in establishing, designing, and planning a comprehensive set of project management and software development processes. It was developed by merging the core entities of the PMBOK meta-model with the core entities of the Scrum meta-model. To support practitioners who are responsible for planning a software project with a comprehensive set of project management and software development processes and to ensure the consistency of the integrated PMBOK-Scrum model, a set of eight constraints is provided.

In order to support the application of the framework, a prototype tool has been created as a Web-based application, using the Java language and a MySQL database. It helps an end user (e.g., a project manager and a team leader) to get insight into the organization's current maturity by assessing the identified CSFs through the list of practices required by the SDM model. Weak practices as a part of assessment results will be used to plan the project together with the defined information (e.g., project, phase, and activity) required by the integrated PMBOK-Scrum model. At this stage, the prototype tool provides limited support, i.e., developing plans, assigning resources to tasks, and analyzing workloads. After planning, the defined process is then validated and prepared in an eXtensible Markup Language (XML) file format for export to the organization's project planning tools.

Chapter 5: It is important to perform a reality check on whether the software process maintenance framework is applicable in real-life software projects. The evaluation of the framework was performed through two case studies in the Thai telecommunications industry from November 2010 to February 2011. The evaluation was split into two phases: the first phase performed at CAT Telecom Public Company Limited (CAT) and the second phase performed at Public Company Limited (TOT). The main goal of the first phase is to provide an analysis of the application of the framework and the participants' knowledge transfer mechanism; whilst the second phase involves collecting only interesting data which offers our double check on certain factors and issues in the case studies. The data collection of both phases was carried out through on-site observations, individual interviews, and questionnaires. The findings reveal that the framework is perceived as acceptable in terms of usefulness and ease of use. It promises to provide the improvement of software development performance in terms of efficiency and effectiveness. However, the significant degree of improvement depends up the maturity of software development.

Based on the findings, we identified certain software practices under five CSFs that were efficiently and effectively implemented in both cases (i.e., project management process; user involvement; appropriate methods, techniques, and tools; team capability; and team environment) and four additional CSFs that were efficiently and effectively implemented in the CAT team (i.e., management commitment, agile software engineering process, organizational environment, and reviews). The better the CSFs are well implemented, the better the increased software development performance can be gained.

Furthermore, we identified eight certain challenges that need to be addressed for further improvement. These include a lack of consistent self-discipline on backlog administration, a lack of appropriate workload allocation and awareness of their roles and

responsibilities, a lack of team self-management, the need of team leaders who can make a decision and guide teams in the right direction, a lack of balanced agile and disciplined environments, a lack of intensive face-to-face communications, less-detailed documentation, and a lack of sufficient knowledge transfer.

Besides, we found six certain practices that both cases needed to make changes for adapting the framework. There were clearly explaining project goals, objectives, and roadmaps to all team members; using both iterative and ongoing project management plans throughout the software projects; working together between users and team members from iterative planning to closure through continuous communications; freezing requirements during iterations; testing, reviewing work, and collecting lessons learned in short-time iterations; and cultivating shared-value environments through sufficient knowledge transfer.

Knowledge transfer is crucial to success in software development. During transferring new knowledge (e.g., the framework and software-development-related knowledge) from the authors to the teams or amongst team members, the participants considered four factors (i.e., the knowledge's usefulness and ease of use, suitability with the organizational or team cultures, and compatibility with the existing software processes) to decide whether or not to use new knowledge. Once all of these factors were satisfied by all team members, the transferred knowledge was used. Otherwise, the transferred knowledge was more likely to be rejected. Some transferred knowledge was directly used; whilst some was tailored to fit into their software development environments. Once the expected outcomes (e.g., work performance and work satisfaction) from using the transferred knowledge were satisfied, the transferred knowledge was integrated into their standard practices. Otherwise, it was either re-tailored until being to solve their occurring problems or meeting their objectives, or continuously used until achieving the expected outcomes. During the transfer process, the findings reveal that team members' motivation, absorptive capacity, credibility, capability or the knowledge source's reservoir of knowledge, communication frequency, good relationships between team members, and key stakeholder commitment significantly affects the knowledge transfer success. The more the quality of these factors exists in the teams; it is more likely to gain knowledge transfer effectiveness. This suggests that the participants should continuously assess, implement, and improve these factors in order to achieve successful knowledge transfer.

Based on these findings, we identified requirements for successful adaptation of the framework. In the organization context, the framework requires management to motivate changes, support hybrid agile and disciplined environments, and cultivate collaborative self-management. In the software process context, teams must iteratively inspect and adapt the integrated project management and software development processes to fit into any circumstances. To do so, it is important to have adaptive people who understand both traditional and agile software development approaches on teams. In the knowledge transfer context, the following factors are required to be existed in teams. Those factors includes knowledge's source motivation, capability and credibility; knowledge recipient's motivation and absorptive capacity; knowledge usefulness and ease of use; good relationships between team members, commitment; frequent communications; and (supportive) organization culture. Practitioners should continuously assess and improve these factors for successful knowledge transfer. As how to successfully organize transfer knowledge still remains a challenge for the organizations. The findings regarding the participants' knowledge transfer mechanism and the identified knowledge transfer factors were used to design and develop a knowledge transfer framework.

Chapter 6: For organizing knowledge transfer during software development, we have delved into the prior literature that forms the foundation of the proposed knowledge transfer framework. The literature review has been presented in three sections. The first section examined what are the differences in how knowledge transfer is defined in the literature and what we can learn from those differences. The findings reveal that the connectionistic epistemology which refers to knowledge residing in human connections is in this study considered the most suitable for software development. Based on the connectionistic perspective, knowledge transfer should be viewed as a communication process between the source and the recipient engaged in teams through communication channels for their learning and applying knowledge. Based on communication-based models, we have considered Szulanski's model. Typically, knowledge transfer has its components. The second section thus scrutinized its common components and how individual components interact amongst them. The findings reveal that knowledge transfer consists of six common components: problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes. These components are connected with others through a multi-directional set of interactions. They can occur at the same or different times and more than once. The third section had highlighted what are differences in the 27-highly-visible literature in knowledge transfer in software development. The findings reveal that all of these studies neither put an emphasis on all of the six components nor do they clearly offer comprehensive descriptions and relationships between those components. The ones providing guidance on how to put knowledge transfer into action are scarce. Consequently, a knowledge transfer framework has been proposed, aiming at covering the six components, providing guidance for planning knowledge transfer activities, and contributing to an effective knowledge transfer amongst software development team members.

Chapter 7: In the knowledge transfer framework, we have elaborated the six components. "Problems" lead to knowledge transfer and in turn help teams define what "Knowledge" is required and what "Mechanisms" fit their software development contexts. "Antecedents" in this study mean determining factors of the ease or difficulty of knowledge transfer. Weak antecedents lead to new problems, whilst supportive antecedents affect transferability, the ability to use the knowledge, and satisfactory outcomes. Besides, designing and selecting "Mechanisms" depends upon required knowledge and software development environments. Suitable mechanisms lead to transfer effectiveness. Otherwise, unexpected problems may occur and expected outcomes are unlikely to be achieved. In addition, "Knowledge Application" can bring about knowledge retention. It may lead to a new consideration of the underlying problem or the identification of new problems. When satisfactory "Outcomes" are achieved, sustaining knowledge application is more likely to occur. This shows that these components are connected with others through a multi-directional set of interactions.

Based on Szulanski's model, a knowledge transfer flows through four distinct stages (i.e., Initiation, Implementation, Ramp-up, and Integration). Owing to relationships between knowledge transfer components and stages, a list of activities has been designed under each component. A list of key questions that should be considered has been suggested under each activity. A flow of relevant activities has also been illustrated under each stage.

Starting with the Initiation stage, it is triggered by all events leading to the decision to transfer knowledge. A transfer begins when the required knowledge meets a need. The discovery of the need leads to the search for valuable knowledge, which in turn triggers to seek potential solutions. However, there are antecedents that make difficulties to initiate this stage (e.g., difficulties of knowledge contents). Hence, antecedents must be assessed in order

to reduce failure chances. From this view, activities pertinent to the components of problems, antecedents, knowledge, and mechanisms are involved in this stage. Once the potential solution planning is done, the actual activities can then be executed directly through the Implementation stage.

The Implementation stage begins with the decision to transfer the knowledge. When it takes place, the plan should be followed. Moreover, resources flow between the source and the recipient. Hence, antecedents supporting communications between them should be established (e.g. establishing several communication channels, strong commitment, and reward systems). During this stage, the required knowledge is often tailored to suit the expected needs and to pre-empt problems experienced in the past. However, the transfer process may be abandoned or re-initiated if the knowledge is deemed unsuitable. From this view, activities pertinent to the components of antecedents, knowledge, and mechanisms are involved in this stage.

The Ramp-up starts when the recipient starts using the transferred knowledge. The recipient typically uses the knowledge ineffectively at first, but gradually identifies and rectifies unexpected problems until being able to achieve satisfactory outcomes. During the use, the recipient may request additional support from the source in solving problems. However, the recipient may abandon or re-initiate the transfer process if there are too many difficulties to use it. Thus, impeding antecedents must be removed. From this view, activities pertinent to the components of problems, antecedents, mechanisms, knowledge application, and outcomes are involved in this stage. Once satisfactory outcomes are achieved, the transfer process then flows through the Integration stage.

The Integration stage begins after the recipient gained desired outcomes with the transferred knowledge. Knowledge application and its integration with existing practices gradually become routinized into standard practices. At this stage, the Integration activities are carried out to ensure that the recipient can use the transferred knowledge without any support from the source and can take any remedial action to improve the understanding of the transferred knowledge and integrate it into his/her practices. Moreover, this stage primarily looks at the efforts required to minimize problems and deal with challenges to the routinization of the transferred knowledge. When the knowledge transferred presents too many difficulties, it is unlikely to become part of routines and therefore sustained in a practice. Hence, antecedents must be reviewed and managed. From this view, activities pertinent to the components of problems, antecedents, knowledge, mechanisms, and knowledge application are involved in this stage. Once the recipient can integrate the transferred knowledge into his/her knowledge packages and use it without any support, the transfer process is then recognized as successful.

For a better understanding, the knowledge transfer framework has been demonstrated, based on the findings of the case studies in Chapter 5. The demonstration shows that most of the transfer processes were recognized as successful as the participants were satisfied with the transferred knowledge (i.e., the software process maintenance framework), integrated it into their existing practices, and sustained it in their teams. However, they required time to apply the transferred knowledge effectively. However, there were some transfer processes recognized as ineffective due to insufficient support from the source, a lack of motivation, a lack of absorptive capacity, knowledge perceived difficult, a lack of commitment in terms of time, a lack of intensive communications, and strong embeddedness of their existing practices. Some were recognized as failure due to incapability between the transferred knowledge and their organizational practice and no readiness to apply the transferred knowledge. Therefore, the participants should pay more attention to these difficulties for

improving their knowledge transfer on further software projects. The demonstration also shows that the knowledge transfer framework has a high degree of compatibility with Scrum-oriented software development. Even though the framework has not been empirically tested yet, it was designed and constructed based on the positive results of the case studies, regarding transferring new knowledge (i.e., the software process maintenance framework) into the cases. This implies that there is a great likelihood that the framework is practical in real-life software projects.

Chapter 8: As the developed software process maintenance framework and the developed knowledge transfer framework are required to solve the software development problems found in the Thai telecommunications industry; both frameworks have been integrated into an umbrella framework, called a framework for transferring software project management approaches into the Thai telecommunications industry. There are four steps for applying the umbrella framework. First, starting with the software process maintenance framework, practitioners can first evaluate their existing software processes, using an SDM model. The obtained results can be used for setting technical goals and objectives and planning for improving quality software processes and products. Second, the integrated PMBOK-Scrum model and the knowledge transfer framework can then be used together to create a software project plan by defining a necessary set of project management, software development, and knowledge transfer processes. Third, the plan is then executed, inspected, and adjusted through iterations until the software project completion, using the integrated PMBOK-Scrum model and the knowledge transfer framework as guidance. Last, it is optional that the implemented software processes can be evaluated through the SDM model after the software development. This is in order to compare the overall performance of before- and after- software development.

For a better understanding, the umbrella framework has been demonstrated, based on the findings of the case studies in Chapter 5. Albeit the umbrella framework has not fully been tested in real-life practice, the findings of the evaluation of the software process maintenance framework indicate the generation of positive effects by (1) improving software development efficiency and effectiveness (e.g., increasing productivity, reducing rework, enhancing customer/team satisfaction) and (ii) cultivating collaborative teamwork, informal frequent communications, and knowledge sharing culture. Nevertheless, we hope to carry out case studies to evaluate the usability and practicality of the umbrella framework in the future. To guide future work directions, next sections present a summary of our theoretical contributions, implications for future research design, implications for practice, limitations of this study, and recommendations for future work.

9.2 Research Contributions and Implications

This section discusses theoretical contributions and potential implications that can be classified into two categories: implications for future research design and implications for practice.

9.2.1 Theoretical Contributions

The major theoretical contribution of this study is a framework for transferring software project management approaches into the Thai telecommunications industry aiming at contributing to the improvement of software development performance. As mentioned, it

consists of two core components. First, a software process maintenance framework assists in measuring, planning, and improving project management and software development processes. Second, a knowledge transfer framework offers guidance for planning knowledge transfer activities. Giving a comparative picture, we perform two comparisons between our theoretical contributions and the existing theoretical literature on (i) agile software development integration with software process improvement and with traditional project management and (ii) knowledge transfer.

The first comparison is between our software process maintenance framework and the relevant existing theoretical literature on agile software development integration with software process improvement and with traditional project management. As our software process maintenance framework consists of a software development maturity model and an integrated PMBOK-Scrum model, we therefore separate details into two domains: software process assessment approaches and hybrid agile-disciplined approaches.

Considering software process assessment approaches, McCaffery et al. [138] propose an assessment method providing what process areas are most applicable for firms wishing to be automotive software suppliers, by integrating CMMI process areas, Automotive SPICE processes, and several agile practices. Petersen and Wohlin [140] propose a lean measurement method used to assess the performance of a software process through a set of individual inventories (i.e., requirements, test cases, change requests, faults and failures, and fault-slip-through) and an analysis of the situation aiming at determining the causes for high inventory level and quality problems. In other words, this method can be used to continuously identify wastes in software development. Traditional process maturity models (e.g., CMMI and ISO/IEC 15504) aim to provide process repeatability and predictability; whilst an Agile Process Maturity Model is designed to enhance agile capability (not to rate an organization's adoption level), to provide process visibility and adaptivity, and to offer guidance for putting agile processes and practices into context and adopting the right strategies and techniques for an organization [28]. McCaffery et al. [138] said that SPI provides the first step to move towards software quality and assessments are a critical part of this process, whilst Khan et al. [136] suggest that an agile method should be adopted as prerequisite to CMM/CMMI. Having a different purpose, we have developed a software development maturity model guiding practitioners on what CSFs affecting software development need implementation and improvement through agile processes, covering both management and development processes. The software development maturity model enhances software development capabilities, provides where an organization is in its adoption level, and help practitioners to prepare themselves for going for CMMI-based process improvements.

Considering hybrid agile-disciplined approaches, we consider models that aim at enhancing project management in agile software development, not software process improvement. Callegari and Bastos [8] propose a model for software project management based on PMBOK and its integration with Rational Unified Process (RUP). Zaki and Moawad [145] propose a new hybrid agile-disciplined model consisting of six phases which are (1) Inception, to set up a project with five main activities which are start-up activities, aspects evaluation activities, gathering requirements and building backlogs, architectural activities, and building a prototype; (2) Planning, to set up the project boundaries; (3) Iterative Assessment, to customize agile and traditional processes; (4) Iterative Building, to build the product; (5) Production, to deliver the product; and (6) Closure, to close the project when there are no longer new requirements for implementation, when the product is not delivering the desired outcomes, or when the product is too expensive for further development. Having a similar purpose, we have developed an integrated PMBOK-Scrum

model providing the "how" to implement CSFs and develop software with a comprehensive set of project management and software development processes. The integrated PMBOK-Scrum model starts with initiation, planning and customizing integrated software process, executions, reviews and retrospectives, and closure. Overall, the major difference comparing to other relevant literature is that our software process maintenance framework provides both the "what" and the "how" to develop and improve software process and product quality.

Regarding the second comparison between our theoretical knowledge transfer framework and the existing theoretical literature on knowledge transfer and, the existing knowledge transfer models and frameworks can be classified into the following categories.

- Antecedent-based models; placing an emphasis on investigating influential antecedents that enable or impede the ability to share and learn from knowledge transfer interactions, e.g., motivation, capability, absorptive capacity, relationship between a source and a recipient, and communication frequency [40, 73, 74, 79, 84, 86, 91, 92, 273].
- Component-based models; articulating components of knowledge transfer. For instance, Albino et al. [280] propose a knowledge transfer framework having four components which are actors involved in knowledge transfer, the context where interactions take place; the knowledge content transferred between actors, and the media by which the transfer is carried out.
- Antecedent- and component- based models; considering antecedents as one of knowledge transfer components. For instance, Becker and Knudsen [253] argue that knowledge transfer must include antecedents, mechanisms, and outcomes. Ward et al. [257] propose a knowledge transfer framework consisting of problems, knowledge, antecedents, knowledge transfer activities, and knowledge utilization, while Martinkenaite [255] proposes an integrative framework illustrating the relationship between antecedents and outcome of knowledge transfer.
- Process-based models; describing knowledge transfer that flows though many stages or processes. For instance, Nevis et al. [272] propose an organizational learning model describing three stages of a transfer process which are knowledge acquisition, sharing, and utilization. Jackson and Klobas [310] propose a knowledge creation and sharing process model describing six major processes: internalization, which describes recipient's knowledge absorption; personal knowledge creation, which can be done through routinization or transformation; externalization, which is the knowledge expression in a symbolic form; objectivation, which is the creation of shared, social constructs that represents a group's understanding; legitimation, which is a process whereby knowledge is authorized and standardized; and reification, which is a process in which concepts harden in the minds of group and attain an existence.
- Process- and antecedent- based models, paying attention on both descriptions on a knowledge transfer flow and antecedents affecting knowledge transfer performance. For instance, Szulanski [103] proposes a knowledge transfer model describing four distinct stages of knowledge transfer: Initiation, beginning with all events leading to the decision to transfer; Implementation, beginning with the decision to transfer; Ramp-up, beginning when the recipient starts using the transferred knowledge; and Integration, beginning after the recipient achieves satisfactory outcomes. This study also describes barriers to the transfer process, e.g., unproven knowledge, causally ambiguous and arduous relationship.

Considering more details of the above models and frameworks, albeit most of them view knowledge transfer as a process, a small number of studies provide insight into the

transfer process. Moreover, only understandings of interactions between antecedents, components, and processes of knowledge transfer may remain doubt to practitioners on how to drive them into action. From this view; however, only few studies (e.g., Jackson and Klobas [310] and Ward et al. [257]) provide such guidance. For a comprehensive understanding and providing guidance on how to put knowledge transfer into action in order to ensure the maximization of knowledge transfer performance, we consequently propose an antecedent-, component-, and process- based framework. The framework consists of six components which are problems, antecedents, knowledge, mechanisms, knowledge application, and outcomes. To provide guidance, a transfer activity flow of each of four knowledge transfer stages is provided. Within each activity, a set of questions are guided towards action planning.

9.2.2 Implications for Future Research Design

The implications for future research design have been drawn from the practical experiences of the authors. For each implication, the experience is described and the actions for future researchers to improve their research design are presented.

Questionnaires: For designing an SDM model, several agile practices in real-life software projects need to be explored for a various set of CSFs. It was inevitable to collect data through questionnaire information collections/surveys with many questions. Although the respondents returned their completed questionnaires as expected, some of their reaction indicated that they struggled to complete the questionnaires. The authors often needed to put much effort into encouraging the respondents to complete the questionnaires. Hence, future researchers are argued to take appropriate data collection methods and designs into account in order to maximize chances of receiving completed data with sufficient validity and reliability.

Case Studies: Case studies were expected to finish as originally planned; however, this was achieved only in CAT. The main reason that it was not achieved in TOT as they committed is that the case project was interrupted by key participants' multi-projects and/or unavailable-for-full-time participation in the case project. Owing to time limitations of the authors to stay in Thailand, this situation has led to many anticipated problems, e.g., ineffective communications between the authors and the team and the limited ability to perform on-site observations. Therefore, future researchers should secure commitment of all participants in order to prevent anticipated problems leading to ineffective knowledge transfer from the authors to the participants and ineffective data collections. Another related implication is that in order to investigate the more precise potential of software methods developed, it is important to investigate it on intermediate or mature software development teams. Thereby, the case selection needs to address "What are the outstanding characteristics of the case that makes it worth researching?". Moreover, both flexible and fixed research designs need to be performed in order to gain a deeper understanding.

Interviews: Individual interviews were expected to be carried out at the end of the case studies and to flow naturally. These were achieved in almost all cases. The prepared questions were sometimes answered during developing the case projects, e.g., the case study teams' existing software processes and their problems always encountered during software development. However, they reaction as observed sometimes indicated that they did not feel comfortable to openly express their opinions in public. From this experience, future researchers should be able to notice respondents' feelings, based on a particular culture. We strongly suggest that researchers should clearly understand at least the culture of the country

that case studies are taking place. Moreover, the individual interviews provide two main situations that need to be deemed, due to their available time and interesting issues discovered during the interviews. First, as we experienced in CAT, future researchers should be prepared to deviate from the prepared script in order to gain a deeper understanding of the explanations of interesting issues being given by the respondents, albeit this may run out of time or be unable to ask all prepared questions. Second, in case of the respondents having very limited available time as we experienced in TOT, future researchers should be able to recognize what the prepared questions are highly required answers and meanwhile manage such limited time to gain a clear understanding of the explanations of interesting issues being given by the respondents.

9.2.3 Implications for Practice

Through the application of a software process maintenance framework, the overall empirical results reveal that practitioners were able to deal with their typical software development problems (e.g., inability to cope with changing requirements, schedule problems, and insufficient time to review and remove redundant codes) and in turn led to increased software development performance. However, there was a conflict between the obtained results of a software practice assessment that should reflect their current software practices and the actual problems encountered during the case projects. Albeit a knowledge transfer framework has not yet been evaluated in real-life software practice, its application demonstration based on our two case studies reveals that the framework has potential to improve their knowledge transfer activities, and consequently the improvement of software development performance. Thereby, the potential implications of our findings as guidelines for practice are highlighted as follows.

1. The better the organization can implement CSFs suggested by the SDM model, the better the organization can achieve efficiency and effectiveness of software development and higher maturity levels. All key stakeholders (e.g., management, key users, and team members) should be aware of this.
2. Assessing existing software practices with the minimum bias shall provide the most precise results. Otherwise, some important software areas that need improvement may be overlooked.
3. Before adaptation of the software process maintenance framework, practitioners should consider and establish strategies to deal with the certain CSFs, challenges, necessary changes, and requirements identified in Chapter 5 in order to gain a higher degree of successful adaptation of the framework.
4. When applying the software process maintenance framework more generally in an organization, it is expected to generate positive effects by (i) increasing software development performance in terms of efficiency (e.g., reducing rework and increasing productivity) and effectiveness (e.g., reducing defects and increasing customer/team satisfaction); and (ii) cultivating collaborative teamwork, informal frequent communications, and knowledge sharing culture. These effects are in turn expected to benefit the implementation of a knowledge transfer framework by maximizing the possibility of the source and the recipient engaged in a software project to transfer, learn, and apply knowledge to solve any problems and accomplish work effectively.
5. Knowledge transfer shall be actively encouraged as normal practices and recognized as an integral aspect of software development activities. This is

expected to build an acceptance of informal activities in the software development team or the organization. This is one of practical ways to build up new internal knowledge sources critical to knowledge-intensive software development success.

6. Developing software processes to assist in the implementation of a knowledge transfer process, e.g., review and retrospective meetings to collect lessons learned, is expected to ensure that explicit knowledge resides within the organization and tacit knowledge resides within team members.

9.3 Focus and Limitations of this Study

This section discusses the potential limitations that exist with this study as it was designed and implemented.

1. As mentioned, although there are indications that some of the major findings of this research might actually be of a more general nature and hence of a wider applicability, we have decided to limit the analysis to the Thai telecommunication industry for five reasons. First, the telecommunications industry was chosen as the research domain since it is a significant and highly developed area of the Thai economy. Moreover, implementing and deploying its elements (e.g., advanced mobile networks) is likely to stimulate innovation in the development of the software industry [95]. Hence, focusing on the telecommunications industry may also benefit the software industry. Second, telecommunications is a high competitive industry. Companies in this domain do consequently depend upon quickly rolling out higher quality of services and products and innovation through efficient and effective software development and knowledge transfer mechanisms. Third, the perspectives and results of this research are presumably easier to transfer into an already developed industry. Fourth, the setting of this study was determined by ÖAD (the Austrian Agency for International Cooperation in Education and Research) and the Higher Education Commission of Thailand who support this study in the form of a scholarship. Hence, the economically most beneficial contribution of this study is knowledge that can be transferred into the Thai telecommunications companies. Last, as the sample of the participating companies was limited to the Thai telecommunications industry, it would be too risky to draw more general conclusions. This is because they cannot be substantiated by data from our case studies. Consequently, we at this stage limit our proposed frameworks (i.e., a software process maintenance framework and a knowledge transfer framework) and conclusions to software development in the Thai telecommunication industry. Nevertheless, we hope to further investigate, modify, and test our framework in other industries in order to proof its general applicability.
2. The current trend towards adopting agile methods in Thailand is just at the initial stage [188, 189], as supported by only few years of agile experiences of the majority of respondents on our questionnaire-style information collection presented in Chapter 4. This results in the limited ability to collect and generalize data for designing our SDM model. However, the results from our questionnaire-style information collection on the utilization of agile practices in three companies in Thailand are consistent with the recent empirical results from the software industry in Thailand, presented in Chookittikul et al. [188]. In other

words, almost all common agile practices identified in Chookittikul et al. [188] (i.e., refactoring, whole team, unit testing, coding standards, and small release) were also found in our results of the questionnaire-style information collection. This raises more confidence in our data generalization.

3. According to our case studies of a software process maintenance framework, the participants were previously inexperienced in agile software development. Even though the overall findings reveal that non-agile teams can gain increased software development performance with integrated agile-disciplined processes, this limits our assurance that the framework can indeed be used as a possible alternative to agile teams to manage and develop software. However, as mentioned, the current trend towards adopting agile methods in Thailand is just at the initial stages. This implies that a majority of companies in the Thai telecommunications industry may still currently either use traditional software development methods or have traditional software development environments. As the result of the continuation of using the framework on other software projects in the participating organizations, this implies that generalizability should more or less be increased. Hence, this study may provide generable results to companies or software projects having contexts similar to the cases.
4. In our case studies, we did not have history documents of the participating companies' existing software projects. This leads to a limited ability to compare software development results between before- and after- use of the software process maintenance framework. However, the overall findings prove that their software development performance in terms of efficiency and effectiveness was improved.
5. Our case projects were relatively small in terms of the team size and the project duration. Hence, a software process maintenance framework has a limited ability to promise that it can overcome major shortcomings of agile methods in some management aspects, e.g., high quality assurance and procurement management.
6. Our case studies were focused on state-owned enterprises, not private companies who are leaders in the overall Thai telecommunications market, e.g., Advanced Info Service Public Company Limited (AIS), Total Access Communication Public Company Limited (DTAC), and True Corporation Public Company Limited (TRUE) [226]. This limits generalizability of the results for the Thai telecommunications industry. However, the relevance or similarities between our findings and the findings of other cases were described and the contexts of the case projects were pointed out in order to make explicit to what degree the results are generalizable.
7. Because of time limitations of this study, a knowledge transfer framework and a framework for transferring software project management approaches into the Thai telecommunications industry (so called "the umbrella framework") have not yet been fully evaluated in real-life software projects. Based on only the full evaluation results of the software process maintenance framework, we cannot yet give assurances about the knowledge transfer framework component. However, the knowledge transfer framework was designed and constructed based on the positive results of our prior case studies, concerning the successful transfer of new knowledge (i.e., the software process maintenance framework) into the cases. Therefore, there is a great likelihood that the knowledge transfer framework is practical in real-life software projects.

9.4 Possibilities for Further Research and Practical Work Building on and Extending the Results of this Thesis

Throughout this research project, especially the results of the case studies we have presented in Chapter 5 and the application demonstrations of a knowledge transfer framework described in Chapter 7 and a framework for transferring software project management approaches into the Thai telecommunications industry described in Chapter 8 have opened several areas to be explored in the future as follows.

1. A software process maintenance framework may need to carry out more case studies, especially in major Thai telecommunications players in terms of total market share, e.g., AIS, DTAC, and TRUE. This is in order to confirm and compare results with the existing results from our case studies in CAT and TOT. There are two questions in case that there is any negative indication or it is possible to follow up the application of the software process maintenance framework in CAT and TOT. In case practitioners prefer to use their existing traditional software development methods, the questions are "What are the root causes of why agile-oriented and/or hybrid agile-disciplined methods do not work for Thai organizations doing more traditional software development?" and "How can those root causes be overcome?"
2. The design and results of a software process maintenance framework reveal that it can be used as a general framework. Hence, it would be good to evaluate the framework in other industries such as government, banking, and manufacturing. This is in order to compare similarities and differences between results in various industries. As influential factors affecting the successful software development can be changed over times and may be different in different industries, we hence suggest that influential factors should also be re-investigated.
3. One issue that arose during construction of the knowledge transfer framework was that of how to maximize the capacity of an organization's existing communication channels or Information and Communications Technologies (ICTs) that can be employed for effective software development and knowledge transfer. Albeit this issue is out of the dissertation's scope, it is interesting to explore a mechanism dealing with such issues. This should benefit an organization by reducing software development and/or knowledge transfer costs and increasing existing communication channels' and ICTs' capacity.
4. As a knowledge transfer framework and a framework for transferring software project management approaches into the Thai telecommunications industry could not be fully evaluated in real-life software projects due to time limitation of this study; consequently, both frameworks shall be evaluated and improved for better usability and practicality. In case the knowledge transfer framework is perceived as usable and practical, it may be evaluated in other industries and/or with other possible frameworks. For instance, it may be used as an add-on component of a promising evaluation framework for e-Government services [348] to test in the Thai e-Government area.
5. It is important to improve the shortcomings of our prototype tool and add more features to enhance the prototype tool's usability in supporting the use of the software process maintenance framework. Moreover, features to support the use of the knowledge transfer framework shall be created and incorporated into the prototype tool. This is in order to facilitate practitioners when using both the

software process maintenance framework and the knowledge transfer framework together.

Whilst further case studies are needed to evaluate and refine the frameworks, increase the generalizability of the results, and extend the results of this study; the evidence we have collected from the use of the software process maintenance framework and the demonstrations of the use of the knowledge transfer framework and the framework for transferring software project management approaches into the Thai telecommunications industry, is encouraging and reveals that the frameworks can be used as an alternative means to software development. In other words, we are encouraged that the agile and disciplined methods can be integrated and sufficient knowledge transfer amongst team members should be implemented to increase software development performance and satisfy users with quality software. However, practitioners are novices when a technology changes the ways to develop software or the nature of the tasks the practitioners perform. Thereby, the use of the technology (e.g., the frameworks) requires time and experience to gain more efficiency and effectiveness. Finally, I shall leave this dissertation towards generalizability and extensibility for future researchers.

References

1. WebsiteOptimization, *US Broadband Penetration Drops to 27th Place Worldwide - July 2011 Bandwidth Report*, 2011.
2. N. Porrawatpreyakorn, et al., "Requirements for a Knowledge Transfer Framework in the Field of Software Development Process Management for Executive Information Systems in the Telecommunications Industry," *The 3rd International Conference on Advances in Information Technology*, Communications in Computer and Information Science 55, B. Papasratorn, et al., eds., Springer Berlin/Heidelberg, 2009, pp. 110-122.
3. W. Jirachiefpattana, "The Impact of Thai Culture on Executive Information Systems Development," *Proc. The 6th International Conference Theme 1, Globalization: Impact on and Coping Strategies in Thai Society*, 1996, pp. 97-110.
4. J. Highsmith and A. Cockburn, "Agile Software Development: The Business of Innovation," *IEEE Computer*, vol. 34, no. 9, 2001, pp. 120-127.
5. K. Kumar and R.J. Welke, *Methodology Engineering: A Proposal for Situation Specific Methodology Construction*, John Wiley & Sons, 1992, p. 257-269.
6. P. Abrahamsson, et al., "New Directions on Agile Methods: A Comparative Analysis," *Proc. The 25th International Conference on Software Engineering*, IEEE, 2003, pp. 244-254.
7. T. Gilb, *Principles of Software Engineering Management*, Addison-Wesley, 1998.
8. D.A. Callegari and R.M. Bastos, "Project Management and Software Development Processes: Integrating RUP and PMBOK," *Proc. International Conference on Systems Engineering and Modeling*, IEEE, 2007.
9. N. Ionel, "Critical Analysis of the Scrum Project Management Methodology," *Proc. The 4th International Economic Conference on European Integration - New Challenges for the Romanian Economy Oradea Romania*, 2008, pp. 435-441.
10. A. Shalloway, et al., *Lean-Agile Software Development: Achieving Enterprise Agility*, Addison-Wesley Professional, 2009.
11. D. Turk, et al., "Limitations of Agile Software Processes," *Proc. The 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering*, Springer-Verlag, 2002, pp. 43-46.
12. G. Spafford, "The Benefits of Standard IT Governance Frameworks," 2003; www.itsmwatch.com/itil/article.php/2195051.
13. W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, 1989.
14. M. Lehman, "Why Is Process Important?," *Proc. The 1st International Conference on the Software Process*, IEEE, 1991, pp. 4 (panel discussion).
15. S. Huang, et al., "Adoption-Centric Software Maintenance Process Improvement via Information Integration," *Proc. The 13th IEEE International Workshop on Software Technology and Engineering Practice*, IEEE, 2005, pp. 25-34.
16. O. Ngwenyama and P.A. Nielsen, "Competing Values in Software Process Improvement: An Assumption Analysis of CMM from an Organizational Culture Perspective," *IEEE Transactions on Software Engineering*, vol. 50, no. 1, 2003, pp. 100-112.
17. SEI, *CMMI for Software Engineering, Version 1.1, Staged Representation (CMMI-SM, V1.1, Staged)*, CMU/SEI-2002-TR-029, Software Engineering Institute, 2002.
18. P.V. Martins and A.R. Silva, "A Comparative Study of SPI Approaches with ProPAM," *Proc. The 6th International Conference on the Quality of Information and Communications Technology*, 2007, pp. 100-109.
19. D.R. Goldenson and J.D. Herbsleb, *After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success*, Software Engineering Institute, 1995.

20. M. Niazi, et al., "A Model for the Implementation of Software Process Improvement: An Empirical Study," *The 5th International Conference on Product Focused Software Process Improvement*, Lecture Notes in Computer Science 3009/2004, F. Bomarius and H. Iida, eds., Springer Berlin/Heidelberg, 2004, pp. 1-16.

21. E. Bos and C. Vriens, "An Agile CMM," *The 4th Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2004*, Lecture Notes in Computer Science 3134, C. Zannier, et al., eds., Springer, 2004, pp. 129-138.

22. J. Diaz, et al., "Mapping CMMI Level 2 to Scrum Practices: An Experience Report," *The 16th European Conference on Software Process Improvement* 42, R. V. O'Connor, et al., eds., Springer Berlin/Heidelberg, 2009, pp. 93-104.

23. C.R. Jakobsen and K.A. Johnson, "Mature Agile with a Twist of CMMI," *Proc. Agile Conference*, IEEE, 2008, pp. 212-217.

24. A.S.C. Marçal, et al., "Blending Scrum Practices and CMMI Project Management Process Areas," *Innovations in Systems and Software Engineering*, vol. 4, no. 1, 2008, pp. 17-29.

25. J. Sutherland, et al., "Scrum and CMMI Level 5: The Magic Potion for Code Warriors," *Proc. The 41st Annual Hawaii International Conference on System Sciences*, IEEE, 2007, pp. 466.

26. M. Pikkarainen and T. Huomo, "Agile Software Development of Embedded Systems: Agile Assessment Framework," 2005; http://www.agile-itea.org/public/deliverables/ITEA-AGILE-D4.1_v1.0.pdf.

27. M. Pikkarainen and A. Mäntyniemi, "An Approach for Using CMMI in Agile Software Development Assessments: Experience from Three Case Studies," *Proc. The 6th International SPICE Conference*, 2006, pp. 121-129.

28. IBM, "The IBM Agile Process Maturity Model," 2009; ftp://ftp.software.ibm.com/software/emea/de/rational/neu/The_IBM_Agile_Process_Maturity_Model_EN_2009.pdf.

29. B.S. Sandhawalia and D. Dalcher, "Knowledge Flows in Software Projects: An Empirical Investigation," *Knowledge and Process Management*, vol. 17, no. 4, 2010, pp. 205-220.

30. S. Henninger, "Case-Based Knowledge Management Tools for Software Development," *Automated Software Engineering*, vol. 4, no. 3, 1997, pp. 319-340.

31. S. Faraj and L. Sproull, "Coordinating Expertise in Software Development Teams," *Management Science*, vol. 46, no. 12, 2000, pp. 1544-1568.

32. I. Attarzadeh and S.H. Ow, "Project Management Practices: Success versus Failure," *Proc. International Symposium on Information Technology*, IEEE, 2008, pp. 1-8.

33. M. Ceschi, et al., "Project Management in Plan-Based and Agile Companies," *IEEE Software*, vol. 22, no. 3, 2005, pp. 21-27.

34. T. Chow and D.-B. Cao, "A Survey Study of Critical Success Factors in Agile Software Projects," *Journal of Systems and Software*, vol. 81, no. 6, 2008, pp. 961-971.

35. H.G. Gemuenden and T. Lechler, "Success Factors of Project Management: The Critical Few-An Empirical Investigation," *Proc. Portland International Conference on Management and Technology* IEEE, 1997, pp. 375-377.

36. A. Shalloway and J.R. Trott, *Lean-Agile Pocket Guide for Scrum Teams*, Lean-Agile Press, 2009.

37. H. Thomas and J. Tilke, "Best Practice Methodologies for the Project Management Office: PMBOK and PRINCE2," 2009; http://www.ca.com/us/default.aspx.

38. PMI, *A Guide to the Project Management Body of Knowledge (PMBOK Guide), Fourth Edition*, Project Management Institute, Inc., 2008.

39. SEI, *CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.1, Staged Representation (CMMI-SE/SW/IPPD/SS, V1.1, Staged)*, Software Engineering Institute, 2002.

40. K.D. Joshi, et al., "Knowledge Transfer within Information Systems Development Teams: Examining the Role of Knowledge Source Attributes," *Decision Support Systems*, vol. 43, no. 2, 2007, pp. 322-335.

41. S.A. Slaughter and L.J. Kirsch, "The Effectiveness of Knowledge Transfer Portfolios in Software Process Improvement: A Field Study," *Information Systems Research*, vol. 17, no. 3, 2006, pp. 301-320.

42. D. Arnott, et al., "Executive Information Systems Development in an Emerging Economy," *Decision Support Systems*, vol. 42, no. 4, 2007, pp. 2078-2084.

43. IMF, *World Economic Outlook, April 2011*, International Monetary Fund, 2011.

44. P. Poon and C. Wagner, "Critical Success Factors Revisited: Success and Failure Cases of Information Systems for Senior Executives," *Decision Support Systems*, vol. 30, no. 4, 2001, pp. 393-418.

45. M. Kirlidog, "Information Technology Transfer to a Developing Country: Executive Information Systems in Turkey," *Information Technology & People*, vol. 9, no. 3, 1996, pp. 55-84.

46. K.C. Laudon and J.P. Laudon, *Management Information Systems*, Prentice Hall, 2009.

47. J.H. Nord and G.D. Nord, "Executive Information Systems: A Study and Comparative Analysis," *Information and Management*, vol. 29, no. 2, 1995, pp. 95-106.

48. W. Jirachiefpattana, et al., "Executive Information Systems Development in Thailand," *Implementing Systems for Supporting Management Decisions: Concepts, Methods, and Experiences*, Implementing Systems for Supporting Management Decisions: Concepts, Methods, and Experiences, P. Humphreys, et al., eds., Chapman & Hall, 1996, pp. 203-224.

49. M.C. Lacity and L.P. Willcocks, *Global Information Technology Outsourcing: In Search of Business Advantage*, Wiley, 2001.

50. R.Y. Wang and D.M. Strong, "Beyond Accuracy: What Data Quality Means to Data Consumers," *Journal of Management Information Systems*, vol. 12, no. 4, 1996, pp. 5-34.

51. C. Batini, et al., "Methodologies for Data Quality Assessment and Improvement," *ACM Computing Surveys*, vol. 41, no. 3, 2009.

52. R.Y. Wang, et al., "A Framework for Analysis of Data Quality Research," *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 4, 1995, pp. 623-640.

53. W. Chutimaskul, et al., "The Quality Framework of e-Government Development," *Proc. The 2nd International Conference on Theory and Practice of Electronic Governance* ACM, 2008, pp. 105-109.

54. K.M. Calo, et al., "A Quantitative Framework for the Evaluation of Agile Methodologies," *Journal of Computer Science and Technology*, vol. 10, no. 2, 2010, pp. 68-73.

55. A.C.C. França, et al., "An Empirical Study on the Relationship between the Use of Agile Practices and the Success of Scrum Projects," *Proc. The 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ACM, 2010.

56. J.A. Livermore, "Factors that Impact Implementing an Agile Software Development Methodology," *Proc. SoutheastCon*, IEEE, 2007, pp. 82-86.

57. S.C. Misra, et al., "Identifying Some Important Success Factors in Adopting Agile Software Development Practices," *Journal of Systems and Software*, vol. 82, no. 11, 2009, pp. 1869-1890.

58. M. Othman, et al., "A Review on Project Management and Issues Surrounding Dynamic Development Environment of ICT Project: Formation of Research Area," *International Journal of Digital Content Technology and its Applications*, vol. 4, no. 1, 2010, pp. 96-105.

59. S. Tong, et al., "Analyse Changing Risk of Organizational Factors in Agile Project Management," *Proc. The 1st International Conference on Information Science and Engineering*, IEEE, 2009, pp. 4188-4193.

60. A. Cockburn and J. Highsmith, "Agile Software Development: The People Factor," *IEEE Computer*, vol. 34, no. 11, 2001, pp. 131-133.

61. R. Turner and B. Boehm, "People Factors in Software Management: Lessons from Comparing Agile and Plan-Driven Methods," *The Journal of Defense Software Engineering*, 2003, pp. 4-8.

62. J. Iivari and N. Iivari, "The Relationship Between Organizational Culture and the Deployment of Agile Methods," *Information and Software Technology*, vol. 53, no. 5, 2011, pp. 509-520.

63. D.E. Strode, et al., "The Impact of Organizational Culture on Agile Method Use," *Proc. The 42nd Hawaii International Conference on System Sciences*, IEEE, 2009, pp. 1-9.

64. M. Cohn and D. Ford, “Introducing an Agile Process to an Organization,” *IEEE Computer*, vol. 36, no. 6, 2003, pp. 74-78.

65. S. Nerur, et al., “Challenges of Migrating to Agile Methodologies,” *Communications of the ACM*, vol. 48, no. 5, 2005, pp. 73-78.

66. P.E. McMahon, “Bridging Agile and Traditional Development Methods: A Project Management Perspective,” *The Journal of Defense Software Engineering*, 2004.

67. T. Dybå and T. Dingsøyr, “Empirical Studies of Agile Software Development: A Systematic Review,” *Inf. Softw. Technol.*, vol. 50, no. 9-10, 2008, pp. 833-859; DOI 10.1016/j.infsof.2008.01.006.

68. R. Hoda, et al., “The Impact of Inadequate Customer Collaboration on Self-Organizing Agile Teams,” *Information and Software Technology*, vol. 53, no. 5, 2010, pp. 521-534.

69. M. Korkala, et al., “A Case Study of Customer Communication in Globally Distributed Software Product Development,” *Proc. The 11th International Conference on Product Focused Software*, ACM, 2010, pp. 43-46.

70. M. Lindvall, et al., “Empirical Findings in Agile Methods,” *Proc. Extreme Programming and Agile Methods – XP/Agile Universe 2002*, 2002, pp. 197-207.

71. B. Schatz and I. Abdelshafi, “Primavera Gets Agile: A Successful Transition to Agile Development,” *IEEE Software*, vol. 22, no. 3, 2005, pp. 36-42.

72. K.D. Joshi, et al., “Knowledge Transfer Among Face-to-Face Information Systems Development Team Members: Examining the Role of Knowledge, Source, and Relational Context,” *Proc. The 37th Hawaii International Conference on System Sciences*, IEEE, 2004, pp. 1-11.

73. S. Sarker, “Knowledge Transfer and Collaboration in Distributed U.S.-Thai Teams,” *Journal of Computer-Mediated Communication*, vol. 10, no. 4, 2005.

74. A. Upadhyaya and S. Krishna, “Antecedents of Knowledge Sharing in Globally Distributed Software Development Teams,” *Proc. The 15th European Conference on Information Systems*, 2007, pp. 727-738.

75. J. Kotlarsky and I. Oshri, “Social Ties, Knowledge Sharing and Successful Collaboration in Globally Distributed System Development Projects,” *European Journal of Information Systems*, vol. 4, no. 1, 2005, pp. 37-48.

76. M. Yuan, et al., “Antecedents of Coordination Effectiveness of Software Developer Dyads From Interacting Teams: An Empirical Investigation,” *IEEE Transactions on Engineering Management*, vol. 56, no. 3, 2009, pp. 494-507.

77. J. Zhang, et al., “The Effect of Organizational/Technological Factors and the Nature of Knowledge on Knowledge Sharing,” *Proc. The 39th Hawaii International Conference on System Sciences*, 2006, pp. 74a.

78. S. Sarker, et al., “Knowledge Transfer in Virtual Information Systems Development Teams: An Empirical Examination of Key Enablers,” *Proc. The 36th Annual Hawaii International Conference on System Sciences - Track 4* IEEE, 2003, pp. 119a.

79. Z. Al-Salti, “Knowledge Transfer and Acquisition In IS Outsourcing: Towards a Conceptual Framework,” *Proc. UK Academy for Information Systems Conference*, 2009.

80. N.H. Arshad, et al., “IT Outsourcing and Knowledge Transfer in Malaysia,” *Proc. The 2nd International Congress on Engineering Education*, IEEE, 2010, pp. 16-21.

81. R. Gregory, et al., “Breaching the Knowledge Transfer Blockade in IT Offshore Outsourcing Projects: A Case from the Financial Services Industry,” *Proc. The 42nd Hawaii International Conference on System Sciences*, IEEE, 2009.

82. J.Y. Park, et al., “The Role of IT Human Capability in the Knowledge Transfer Process in IT Outsourcing Context,” *Information and Management*, vol. 48, no. 1, 2011, pp. 53-61.

83. A. Mohamed, et al., “Influencing Factors of Knowledge Transfer in IT Outsourcing,” *Proc. The 10th WSEAS International Conference on Mathematics and Computers in Business and Economics* 2009, pp. 165-170.

84. H.L. Yun, “Knowledge Transfer in ISD Offshore Outsourcing Project,” *Proc. International Conference on Computer Engineering and Technology*, IEEE, 2009, pp. 487-491.

85. N. Dayasindhu, “Embeddedness, Knowledge Transfer, Industry Clusters and Global Competitiveness: A Case Study of the Indian Software Industry,” *Technovation*, vol. 22, no. 9, 2002, pp. 551-560.

86. D.-G. Ko, et al., “Antecedents of Knowledge Transfer from Consultants to Clients in Enterprise System Implementations,” *MIS Quarterly* vol. 29, no. 1, 2005, pp. 59-85.

87. L. Hongli and Z. Lei, “Knowledge Transfer in Knowledge Network of IT Consulting Company,” *Proc. International Conference on Information Management, Innovation Management and Industrial Engineering*, IEEE, 2009, pp. 490-495

88. Y. Malhotra and D.F. Galletta, “Role of Commitment and Motivation in Knowledge Management Systems Implementation: Theory, Conceptualization, and Measurement of Antecedents of Success,” *Proc. The 36th Hawaii International Conference on System Sciences*, IEEE, 2003.

89. C.-L. Hsu and J.C.-C. Lin, “Acceptance of Blog Usage: The Roles of Technology Acceptance, Social Influence and Knowledge Sharing Motivation,” *Information & Management*, vol. 45, no. 1, 2008, pp. 65-74.

90. C.-J. Chen, et al., “The Role of Intellectual Capital in Knowledge Transfer,” *IEEE Transactions on Engineering Management* vol. 56, no. 3, 2009, pp. 402-411.

91. E.T.G. Wang, et al., “Improving Enterprise Resource Planning (ERP) Fit to Organizational Process Through Knowledge Transfer,” *International Journal of Information Management*, vol. 27, no. 3, 2007, pp. 200-212.

92. Q. Xu and Q. Ma, “Determinants of ERP Implementation Knowledge Transfer,” *Information and Management*, vol. 45, no. 8, 2008, pp. 528-538.

93. L.Z. Cantú, et al., “Generation and Transfer of Knowledge in IT-Related SMEs,” *Journal of Knowledge Management*, vol. 13, no. 5, 2009, pp. 243-256.

94. C. Tiexin, et al., “The Influence Factors of Knowledge Transfer in Project Management: An Empirical Survey,” *Proc. The 4th International Conference on Wireless Communications, Networking and Mobile Computing*, IEE, 2008, pp. 1-7.

95. K. Dasgupta, *The Economic Benefits from Investment in Advanced Mobile Infrastructure and Sevices: The Case of Thailand*, LECG Ltd, 2009.

96. M. Griffiths, “Using Agile Alongside the PMBOK,” *Proc. PMI Research Conference*, 2004.

97. K. Schwaber, “SCRUM Development Process,” *Proc. The 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications* ACM, 1995, pp. 117-134.

98. P. Fitsilis, “Comparing PMBOK and Agile Project Management Software Development Processes,” *Advances in Computer and Information Sciences and Engineering*, T. Sobh, ed., Springer Netherlands, 2008, pp. 378-383.

99. J.R. Persse, *Implementing the Capability Maturity Model*, Wiley, 2001.

100. C.R. Jakobsen and J. Sutherland, “Scrum and CMMI - Going from Good to Great, Are you ready-ready to be done-done?,” *Proc. Agile Conference*, IEEE, 2009, pp. 333-337.

101. B. Mutafelija and H. Stromberg, *Systematic process improvement using ISO 9001:2000 and CMMI*, Artech House, 2003.

102. J. Sutherland and K. Schwaber, *The Scrum Papers:Nuts, Bolts, and Origins of an Agile Process*, 2007.

103. G. Szulanski, “Exploring Internal Stickiness: Impediments to the Transfer of Best Practice within the Firm,” *Strategic Management Journal*, vol. 17, no. Winter Special, 1996, pp. 27–43.

104. T.H. Davenport and L. Prusak, *Working Knowledge: How Organizations Manage What They Know*, Harvard Business School Press, 1998.

105. K.D. Joshi and S. Sarker, “Examining the Role of Knowledge, Source, Recipient, Relational, and Situational Context on Knowledge Transfer Among Face-to-Face ISD Teams,” *Proc. The 39th Annual Hawaii International Conference on System Sciences-Track 7*, IEEE, 2006, pp. 148c.

106. C. Wallin and I. Crnkovic, “Three Aspects of Successful Software Development Projects "when are projects canceled, and why?"” *Proc. The 29th Euromicro Conference*, IEEE, 2003, pp. 368-374.

107. E. Ellmer, “Improving Software Processes,” *Proc. The 1995 Software Engineering Environments*, IEEE, 1995, pp. 75-83.

108. G. Seshagiri, “Continuous Process Improvement-Why Wait till Level 5,” *Proc. The 29th Hawaii International Conference on System Sciences*, IEEE, 1996, pp. 681-692.

109. P. Allen, et al., “PRISMS: An Approach to Software Process Improvement for Small to Medium Enterprise,” *Proc. The 3rd International Conference on Quality Software*, IEEE, 2003.

110. K.C. Dangle, et al., “Software Process Improvement in Small Organizations: A Case Study,” *IEEE Software*, vol. 22, no. 6, 2005, pp. 68-75.

111. N. Ramasubbu, et al., “Leveraging Global Resources: A Process Maturity Framework for Managing Distributed Development,” *IEEE Software*, vol. 22, no. 3, 2005, pp. 80-86.

112. B. Fitzgerald and T. O'Kane, “A Longitudinal Study of Software Process Improvement,” *IEEE Software*, no. May/June, 1999, pp. 37-45.

113. T.M. Somers and K. Nelson, “The Impact of Critical Success Factors Across the Stages of Enterprise Resource Planning Implementations,” *Proc. The 34th Hawaii International Conference on System Sciences*, IEEE, 2001, pp. 8016.

114. M. Niazi, et al., “A Maturity Model for the Implementation of Software Process Improvement: An Empirical Study,” *Journal of Systems and Software*, vol. 74, no. 2, 2005, pp. 155-172.

115. M. Niazi, et al., “A Framework for Assisting the Design of Effective Software Process Improvement Implementation Strategies,” *Journal of Systems and Software*, vol. 78, no. 2, 2005, pp. 204-222.

116. M.-L. Huotari and T.D. Wilson, “Determining Organizational Information Needs: The Critical Success Factors Approach,” *Information Research*, vol. 6, no. 3, 2001.

117. V.K. Khandelwal and J.R. Ferguson, “Critical Success Factors (CSFs) and the Growth of IT in Selected Geographic Regions,” *Proc. The 32nd Hawaii International Conference on System Sciences*, IEEE, 1999, pp. 13 pp.

118. A. Pellow and T.D. Wilson, “The Management Information Requirements of Heads of University Departments: A Critical Success Factors Approach,” *Journal of Information Science*, vol. 19, no. 6, 1993, pp. 425–437.

119. C.K. Tyran and J.F. George, “The Implementation of Expert Systems: A Survey of Successful Implementations,” *ACM SIGMIS Database*, vol. 24, no. 1, 1993, pp. 5-15; DOI 10.1145/154421.154422.

120. SEI, *Process Maturity Profile of the Software Community, 2002 Mid-Year Update*, Software Engineering Institute, 2002.

121. ISO, “ISO/IEC 15504,” 19 December 2009 2004; http://www.iso.org/.

122. R.W. Hoerl, “Six Sigma and the Future of the Quality Profession,” *Quality Progress*, vol. 31, no. 6, 1998, pp. 35-42.

123. Gartner, “Balancing Six Sigma and the Capability Maturity Model (CMM/CMMI),” 6 December 2009; http://www.gartner.com/4_decision_tools/measurement/measure_it_articles/2003_0424/bal_cmm.jsp.

124. K.D. Shere, “Comparing Lean Six Sigma to the Capability Maturity Model,” *The journal of Defense Software Engineering*, 2003.

125. IT-Governance-Institute, *COBIT Framework, 3rd Edition*, Information Systems Audit and Control Foundation, 2000.

126. S.W. Ambler, “Agile Survey Results Summary,” 16 July 2009 2008; http://www.ambysoft.com/downloads/surveys/AgileAdoptionRates.ppt.

127. Danube-Technologies, “Danube Technologies Sees Strong Growth/Scrum Emerges as Leading Method for Agile Software Development,” 14 July 2009 2008; http://www.agilejournal.com/agile-news/807-danube-technologies-sees-strong-growthscrum-emerges-as-leading-method-for-agile-software-developmnt.

128. VersionOne, *3rd Annual Survey: 2008 “The State of Agile Development”*, VersionOne, 2008.

129. Digital-Onion, “Success with Scrum: It’s all About Leadership,” 16 July 2009 2009; http://www.digitalonioninc.com/.

130. PMI, “Statistics of Interest as of 31 March 2009,” 2009; http://search.pmi.org/.

131. B. Kitchenham and S. Charters, *Guidelines for performing Systematic Literature Reviews in Software Engineering*, EBSE 2007-001, Evidence-Based Software Engineering, 2007.

132. D.J. Anderson, “Stretching Agile to fit CMMI Level 3-The Story of Creating MSF for CMMI® Process Improvement at Microsoft Corporation,” *Proc. Agile Conference*, IEEE, 2005, pp. 193-201.

133. S.W. Baker, “Formalizing Agility, Part 2: How an Agile Organization Embraced the CMMI,” *Proc. Agile Conference*, 2006, pp. 154.

134. S. Cohan and H. Glazer, “An Agile Development Team’s Quest for CMMI® Maturity Level 5,” *Proc. Agile Conference*, IEEE, 2009, pp. 201-206.

135. T. Kähkönen and P. Abrahamsson, “Achieving CMMI Level 2 with Enhanced Extreme Programming Approach ” *Proc. The 5th International Conference on Product Focused Software Process Improvement*, Springer Berlin/Heidelberg, 2004, pp. 378-392.

136. M.I. Khan, et al., “Agile Methodology in Software Development (SMEs) of Pakistan Software Industry for Successful Software Projects (CMM Framework),” *Proc. International Conference on Educational and Network Technology*, 2010, pp. 576-580.

137. R. Leithiser and D. Hamilton, “Agile Versus CMMI - Process Template Selection and Integration with Microsoft Team Foundation Server,” *Proc. The 46th Annual Southeast Regional Conference on XX*, 2008, pp. 186-191.

138. F. McCaffery, et al., “AHAA - Agile, Hybrid Assessment Method for Automotive, Safety Critical SMEs ” *Proc. The 30th International Conference on Software Engineering*, 2008, pp. 551-560.

139. A. Omran, “Agile CMMI from SMEs Perspective,” *Proc. The 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, 2008, pp. 1-8.

140. K. Petersen and C. Wohlin, “Software Process Improvement through the Lean Measurement (SPI-LEAM) Method,” *Journal of Systems and Software*, vol. 83, no. 7, 2010, pp. 1275–1287.

141. F.J. Pino, et al., “Using Scrum to Guide the Execution of Software Process Improvement in Small Organizations,” *Journal of Systems and Software*, vol. 83, no. 10, 2010, pp. 1662-1677.

142. G. Rong, et al., “SCRUM-PSP: Embracing Process Agility and Discipline,” *Proc. The 17th Asia Pacific Software Engineering Conference*, 2010 pp. 316-325.

143. O. Salo and P. Abrahamsson, “An Iterative Improvement Process for Agile Software Development,” *Software Process: Improvement and Practice*, vol. 12, no. 1, 2007, pp. 81-100.

144. L. Williams, et al., “Driving Process Improvement via Comparative Agility Assessment,” *Proc. Agile Conference*, IEEE, 2010, pp. 3-10.

145. K.M. Zaki and R. Moawad, “A Hybrid Disciplined Agile Software Process Model,” *Proc. The 7th International Conference on Informatics and Systems*, 2010, pp. 1-8.

146. M. Lepasaar, et al., “Models and Success Factors of Process Change,” *The 3rd International Conference on Product Focused Software Process Improvement*2188/2001, F. Bomarius and S. Komi-Sirviö, eds., Springer Berlin/Heidelberg, 2001, pp. 68-77.

147. A. Rainer and T. Hall, “A Quantitative and Qualitative Analysis of Factors Affecting Software Processes,” *Journal of Systems and Software*, vol. 66, no. 1, 2003, pp. 7-21.

148. D. Stelzer and W. Mellis, “Success Factors of Organizational Change in Software Process Improvement,” *Software Process: Improvement and Practice*, vol. 4, no. 4, 1998, pp. 227-250.

149. F. Guerrero and Y. Eterovic, “Adopting the SW-CMM in a Small IT Organization,” *IEEE Software*, vol. 21, no. 4, 2004, pp. 29-35.

150. M. Niazi, et al., “Critical Success Factors and Critical Barriers for Software Process Improvement: An Analysis of Literature,” *Proc. Australasian Conference on Information Systems*, ACIS, 2003.

151. M. Niazi, et al., “Implementing Software Process Improvement Initiatives: An Empirical Study,” *The 7th International Conference on Product Focused Software Process Improvement*, Lecture Notes in Computer Science 4034/2006, J. Münch and M. Vierimaa, eds., Springer Berlin/Heidelberg, 2006, pp. 222-233.

152. M. Niazi, et al., "Organisational Readiness and Software Process Improvement " *The 8th International Conference on Product-Focused Software Process Improvement*, Lecture Notes in Computer Science 4589/2007, J. Münch and P. Abrahamsson, eds., Springer Berlin/Heidelberg, 2007, pp. 96-107.

153. A. Cockburn, *Agile Software Development*, Addison-Wesley, 2002, p. 215-218.

154. P.E. McMahon, "Lessons Learned Using Agile Methods on Large Defense Contracts," *The Journal of Defense Software Engineering*, 2006, pp. 25-30.

155. A. Qumer and B. Henderson-Sellers, "An Evaluation of the Degree of Agility in Six Agile Methods and Its Applicability for Method Engineering," *Information and Software Technology*, vol. 50, no. 4, 2008, pp. 280-295.

156. B. Boehm, "Get Ready for Agile Methods, with Care," *IEEE Computer*, vol. 35, no. 1, 2002, pp. 64-69.

157. S.d. Sousa, "The Advantages and Disadvantages of Agile Scrum Software Development," 2009; http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-agile-scrum-software-development.html.

158. A. Jain, "Post Workshop Progress Report," *Proc. CSE Annual Research Review & Executive Workshop*, 2002.

159. M. Hecht, et al., "Fostering Adoption, Acceptance, and Assimilation in Knowledge Management System Design," *Proc. The 11th International Conference on Knowledge Management and Knowledge Technologies* ACM, 2011, pp. 1-8.

160. L.G. Tornatzky and M. Fleischer, *The Processes of Technological Innovation*, D.C. Heath & Company, 1990.

161. W.H. DeLone and E.R. McLean, "The DeLone and McLean Model of Information Systems Success: A Ten-Year Update," *Management Information Systems*, vol. 19, no. 4, 2003, pp. 9-30.

162. T.-P. Liang, et al., "Adoption of Mobile Technology in Business: A Fit-Viability Model," *Industrial Management & Data Systems*, vol. 107, no. 8, 2007, pp. 1154-1169.

163. M. Fishbein and I. Ajzen, *Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research*, Addison-Wesley, 1975.

164. I. Ajzen, "From Intentions to Actions: A Theory of Planned Behaviour," *Action Control: From Cognition to Behavior*, J. Kuhl and J. Beckmann, eds., Springer, Heidelberg, 1985.

165. F.D. Davis, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *MIS Quarterly*, vol. 13, no. 3, 1989, pp. 319-340.

166. R.L. Thompson, et al., "Personal Computing: Toward a Conceptual Model of Utilization," *MIS Quarterly*, vol. 15, no. 1, 1991, pp. 125-143.

167. V. Venkatesh, et al., "User Acceptance of Information Technology: Towards a Unified View," *MIS Quarterly*, vol. 27, no. 3, 2003, pp. 425-478.

168. E.M. Rogers, *Diffusion of Innovations (Fifth Edition)*, The Free Press, 2003.

169. G.C. Moore and I. Benbasat, "Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation," *Information Systems Research*, vol. 2, no. 3, 1991, pp. 192-222.

170. A.H. Tolba and M. Mourad, "Individual and Cultural Factors Affecting Diffusion of Innovation," *Journal of International Business and Cultural Studies*, vol. 5, 2011, pp. 1-16.

171. I. Ajzen, "The Theory of Planned Behavior," *Organizational Behavior and Human Decision Processes*, vol. 50, 1991, pp. 179-211.

172. Y. Gao, "Applying the Technology Acceptance Model (TAM) to Educational Hypermedia: A Field Study," *Education Multimedia and Hypermedia*, vol. 14, no. 3, 2005, pp. 237-247.

173. V. Venkatesh and F.D. Davis, "A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field," *Management Science*, vol. 46, no. 2, 2000, pp. 186-204.

174. J. Choudrie and Y.K. Dwivedi, "Towards a Conceptual Model of Broadband Diffusion," *Journal of Computing and Information Technology*, vol. 12, no. 4, 2004, pp. 323-338.

175. M. Pikkarainen, *Towards a Framework for Improving Software Development Process Mediated with CMMI Goals and Agile Practices*, VTT Publications 695, VTT Publications, 2008.

176. M.C. Paulk, et al., *Capability Maturity Model[SM] for Software, Version 1.1*, Software Engineering Institute, 1993.

177. J.J. Jiang, et al., "An Exploration of the Relationship between Software Development Process Maturity and Project Performance," *Information and Management*, vol. 41, no. 3, 2004, pp. 279-288.

178. B. Pitterman, "Telcordia Technologies: The Journey to High Maturity," *IEEE Software*, vol. 17, no. 4, 2000, pp. 89-96.

179. G. Yamamura, "Software Improvement Satisfied Employees," *IEEE Software*, vol. 16, no. 5, 1999, pp. 83–85.

180. V. Khandelwal and R. Natarajan, *Quality IT Management in Australia: Critical Success Factors for 2002*, Technical Report No. CIT/1/2002, University of Western Sydney, 2002.

181. K. Schwaber, *Agile Project Management with Scrum*, Microsoft Press, 2004.

182. CORPORATE-Office-of-the-Under-Secretary-of-Defense-for-Acquisition, "Excerpts from Fall 1987 Report of the Defense Science Board Task Force on Military Software," *ACM SIGAda Ada Letters*, vol. 8, no. 4, 1988, pp. 35-46.

183. N. Porrawatpreyakorn, et al., "Requirements for a Software Process Maintenance Framework for Executive Information Systems in the Telecommunications Industry," *Journal of Global Management Research*, vol. 6, no. 1, 2010, pp. 7-18.

184. R.R. Willis, et al., *Hughes Aircrafts Widespread Deployment of a Continuously Improving Software Process*, Software Engineering Institute, 1998.

185. M. Lindvall, et al., "Agile Software Development in Large Organizations," *IEEE Computer*, vol. 37, no. 12, 2004, pp. 26-34.

186. K. El-Emam, et al., "Modelling the Likelihood of Software Process Improvement: An Exploratory Study," *Journal of Empirical Software Engineering*, vol. 6, no. 3, 2001, pp. 207-229.

187. D. Dorenbos and A. Combelles, "Lessons Learned around the World: Key Success Factors to Enable Process Change," *IEEE Software*, vol. 21, no. 4, 2004, pp. 20-21.

188. W. Chookittikul, et al., "Reducing the Gap between Academia and Industry: The Case for Agile Methods in Thailand," *Proc. The 8th International Conference on Information Technology: New Generations*, IEEE, 2011, pp. 239-244.

189. R. Morien and O. Tetiwat, "Agile Software Development Methods Adoption in Thailand - A Survey of Thai Universities," *Proc. Information Systems Education Conference*, 2007.

190. H. Coolican, *Research Methods and Statistics in Psychology*, Hodder and Stoughton, 1999.

191. N. Baddoo and T. Hall, "De-Motivators for Software Process Improvement: An Analysis of Practitioners' Views," *The Journal of Systems and Software*, vol. 66, no. 1, 2003, pp. 23-33.

192. J.C. Nunnally, *Psychometric Theory*, McGraw-Hill, 1978.

193. A. Acharya, "Agile-From Chaos to Success," *Advances in Computational Sciences and Technology*, vol. 3, no. 1, 2010, pp. 17-22.

194. S. Augustine, et al., "Agile Project Management: Steering from the Edges," *Communications of the ACM*, vol. 48, no. 12, 2005, pp. 85-89.

195. S. Berczuk, "Back to Basics: The Role of Agile Principles in Success with and Distributed Scrum Team," *Proc. The Agile 2007*, IEEE, 2007, pp. 382-388.

196. D. Cohen, et al., "An Introduction to Agile Methods," *Advances in Computers*, vol. 62, 2004, pp. 1-66.

197. M. Coram and S.A. Bohner, "The Impact of Agile Methods on Software Project Management," *Proc. The 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems* IEEE, 2005, pp. 363-370.

198. M. Cristal, et al., "Usage of SCRUM Practices within a Global Company," *Proc. International Conference on Global Software Engineering*, IEEE, 2008, pp. 222-226.

199. G. DeHondt-II and A. Brandyberry, "Programming in the eXtreme: Critical Characteristics of Agile Implementations," *e-Informatica Software Engineering Journal*, vol. 1, no. 1, 2007, pp. 43-58.

200. J. Highsmith, "Innovation & Quality in Healthcare IT: The Agile Revolution," 16 June 2010 2004; http://www.agileprojectmgt.com/docs/healthcareit.pdf.

201. K.H. Judy and I. Krumins-Beens, "Using Agile Practices to Spark Innovation in a Small to Medium Sized Business," *Proc. The 40th Annual Hawaii International Conference on System Sciences*, IEEE, 2007, pp. 275.

202. N.B. Moe, et al., "Understanding Shared Leadership in Agile Development: A Case Study," *Proc. The 42nd Hawaii International Conference on System Sciences*, IEEE, 2009, pp. 1-10.

203. M. Qasaimeh, et al., "Comparing Agile Software Processes Based on the Software Development Project Requirements," *Proc. The 2008 International Conference on Computational Intelligence for Modelling Control & Automation*, IEEE, 2008, pp. 49-54.

204. A.S. Sidky and J.D. Arthur, "Agile Adoption Process Framework," *CoRR abs/cs/0612092*, 2006.

205. A.S. Sidky and J.D. Arthur, "Determining the Applicability of Agile Practices to Mission and Life-critical Systems," *Proc. The 31st Annual IEEE Software Engineering Workshop*, IEEE, 2007, pp. 3-12.

206. J. Vanhanen, et al., "Practical Experience of Agility in the Telecom Industry," *Proc. The 4th International Conference on Extreme Programming and Agile Processes in Software Engineering*, Springer-Verlag, 2003, pp. 279-287.

207. V. Agashe, "Agile: Key to Addressing Data Quality ", June 16, 2010 2009; http://vishagashe.wordpress.com/2009/07/25/agile-key-to-addressing-data-quality/.

208. S.W. Ambler, "Agile Master Data Management (MDM)," 16 June 2010 2008; http://www.agiledata.org/essays/masterDataManagement.html.

209. S.W. Ambler, "Evolutionary/Agile Database Best Practices," 16 June 2010 2010; http://www.agiledata.org/essays/bestPractices.html.

210. R.G. Mathieu and O. Khalil, "Data Quality in the Database Systems Course," *Data Quality* vol. 4, no. 1, 1998.

211. M. Moseley, "Agile Data Governance: The Key to Solving Enterprise Data Quality Problems. Information Management Special Reports," 16 June 2010 2008; http://www.information-management.com/specialreports/2008_105/10001919-1.html?pg=2.

212. N. William, et al., "Data Quality and Agile Methods: A BT Perspective," *Proc. The 11th International Conference on Information Quality*, 2006.

213. H. Xu, "Data Quality Issues for Accounting Information Systems' Implementation: Systems, Stakeholders, and Organizational Factors," *Journal of Technology Research*, vol. 1, 2000, pp. 1-11.

214. C.V. Bullen and J.F. Rockhart, *A Primer on Critical Success Factors*, Working Paper No. 69, Massachusetts Institute of Technology, 1981.

215. M.K. Daskalantonakis, "Achieving Higher SEI Levels," *IEEE Software*, vol. 11, no. 4, 1994, pp. 17–24.

216. B. Henderson-Sellers, et al., "Third Generation OO Processes: A Critique of RUP and OPEN from a Project Management Perspective," *Proc. The 7th Asia-Pacific Software Engineering Conference*, IEEE, 2000, pp. 428-435.

217. B. Hailpern and P. Santhanam, "Software Debugging, Testing, and Verification," *IBM Systems Journal*, vol. 41, no. 1, 2002, pp. 4-12.

218. I. Sommerville, *Software Engineering (7th Edition)*, Pearson Addison Wesley, 2004.

219. K. Beck, "Embracing Change With Extreme Programming," *IEEE Computer*, vol. 32, no. 10, 1999, pp. 70-77.

220. K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, Prentice-Hall, 2002.

221. P. Runeson and M. Höst, "Guidelines for Conducting and Reporting Case Study Research in Software Engineering," *Empirical Software Engineering*, vol. 14, no. 2, 2009, pp. 131-164.

222. ICT-(Ministry-of-Information-and-Communication-Technology), *Thailand Progress Report in 2010 AFACT Year Book*, The 28th AFACT Meeting in Japan, 2010.

223. NTC, "รายงานสภาพตลาดโทรคมนาคม ณ สิ้นไตรมาสที่ 2 ปี 2554," 2011; http://www.nbtc.go.th/index.php?option=com_content&view=article&id=235&Itemid=1.

224. Jasmine, "ข่าวในแวดวงโทรคมนาคม: ฮัลโหลต่างประเทศแข่งเดือด "ทรู" ดัมพ์แหลกปลุกโทร.บ้าน," 2010; http://www.jasmine.com/jasmineweb/press/template_industry-th.asp?ID=1071.

225. C. Wohlin, et al., *Experimentation in Software Engineering - An Introduction*, Kluwer Academic Publishers, 2000.

226. NTC, "รายงานสภาพตลาดโทรคมนาคม ณ สิ้นไตรมาสที่ 2 ปี 2553," 2010; http://www.ntc.or.th/uploadfiles/MK2011_1.pdf.

227. SEI, *Process Maturity Profile: CMMI for Development SCAMPI Class A Appraisal Results 2009 End-Year Update*, Software Engineering Institute, 2010.

228. S. Collyer, "Project Management Approaches for Dynamic Environments," *International Journal of Project Management*, vol. 27, no. 4, 2009, pp. 355-364.

229. K. Beck, *Planning Extreme Programming*, Addison-Wesley, 2000.

230. M. Cohn, *Agile Estimating and Planning*, Prentice-Hall, 2005.

231. S.C. Misra, et al., "Identifying Some Critical Changes Required in Adopting Agile Practices in Traditional Software Development Projects," *International Journal of Quality & Relability Management*, vol. 27, no. 4, 2010, pp. 451-474.

232. Agile-Manifesto, "Manifesto for Agile Software Development," June 16, 2010 2001; http://agilemanifesto.org/.

233. A. Tiwana, "Impact of Classes of Development Coordination Tools on Software Development Performance: A Multinational Empirical Study," *ACM Transactions on Software Engineering and Methodology*, vol. 17, no. 2, 2008, pp. 11:11-11:47.

234. T. Rojas and M. Pérez, "A Comparison of Three Information System Development Methodologies Related to Effectiveness/Efficiency Criteria," *Proc. International Symposium on Applied Corporate Computing*, 1995.

235. B. Shackel, "The Concept of Usability," *Proc. IBM Software and Information Usability Symposium*, IBM Corporation, 1981, pp. 1-30.

236. V. Bruno and G. Al-Qaimari, "Usability Attributes: An Initial Step toward Effective User-Centered Development," *Proc. OZHI2004*, 2004.

237. O. Frandsen-Thorlacius, et al., "Non-universal Usability?: A Survey of How Usability is understood by Chinese and Danish Users," *Proc. The 27th International Conference on Human Factors in Computing Systems*, ACM, 2009, pp. 41-50.

238. O. Laitenberger and H.M. Dreyer, "Evaluating the Usefulness and the Ease of Use of a Web-based Inspection Data Collection Tool " *Proc. The 5th Software Metrics Symposium*, IEEE, 1998, pp. 122-132.

239. D. Karlström and P. Runeson, "Integrating Agile Software Development into Stage-gate Managed Product Development," *Empirical Software Engineering*, vol. 11, no. 2, 2006, pp. 203-225.

240. L. Layman, et al., "Essential Communication Practices for Extreme Programming in a Global Software Development Team," *Information and Software Technology*, vol. 48, no. 9, 2006, pp. 781-794.

241. K. Vlaanderen, et al., *Case Study Report: Agile Product Management at Planon*, Technical Report UU-CS-2009-005, Department of Information and Computing Science, Utrecht University, 2009.

242. S.W. Ambler, "Best Practices for Agile/Lean Documentation," 16 June 2010 2001; http://www.agilemodeling.com/essays/agileDocumentationBestPractices.htm.

243. S. Ratanotayanon, et al., "After the Scrum: Twenty Years of Working without Documentation," *Proc. The 8th International Conference on Software Engineering and Knowledge Engineering*, 2006, pp. 194-199.

244. D. Spann, "Agile: Changing the Organization," *The Executive Update*, 7 October 2011 2005; http://aamngt.com/files/changingtheorg.pdf.

245. J.P. Kotter, "Leading Changes: Why Transformation Efforts Fail," *Harvard Business Review*, vol. 73, no. 2, 1995, pp. 59-67.

246. A. Cockburn and J. Highsmith, "Agile Software Development: The People Factor," *Computer*, vol. 34, no. 11, 2001, pp. 131-133.

247. S.T. Solansky, "Leadership Style and Team Processes in Self-Managed Teams," *Journal of Leadership & Organizational Studies*, vol. 14, no. 4, 2008, pp. 332-341.

248. M. Alavi and D.E. Leidner, "Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues," *MIS Quarterly*, vol. 25, no. 1, 2001, pp. 107-136.

249. B.D. Janz and P. Prasarnphanich, "Understanding Knowledge Creation, Transfer, and Application: Investigating Cooperative, Autonomous Systems Development Teams," *Proc. The 38th Annual Hawaii International Conference on System Sciences - Track 8*, IEEE, 2005, pp. 248a.

250. J. Sapsed, et al., "From Here to Eternity?: The Practice of Knowledge Transfer in Dispersed and Co-located Project Organizations," *European Planning Studies*, vol. 13, no. 6, 2005, pp. 831-851.

251. L. Argote and P. Ingram, "Knowledge Transfer: A Basis for Competitive Advantage in Firms," *Organizational Behavior and Human Decision Processes* vol. 82, no. 1, 2000, pp. 150-169.

252. E. Darr and T. Kurtzberg, "An Investigation of Partner Similarity Dimensions on Knowledge Transfer," *Organizational Behavior and Human Decision Processes*, vol. 82, no. 1, 2000, pp. 28-44.

253. M.C. Becker and M.P. Knudsen, "Intra and Inter-Organizational Knowledge Transfer Processes: Identifying the Missing Links," 20 April 2011 2006; http://www3.druid.dk/wp/20060032.pdf.

254. V. Ward, et al., "Developing a Framework for Transferring Knowledge into Action: A Thematic Analysis of the Literature," *Journal of Health Services Research & Policy*, vol. 14, no. 3, 2009, pp. 156-164.

255. I. Martinkenaite, "Antecedents and Consequences of Inter-Organizational Knowledge Transfer-Emerging Themes and Openings for Further Research," *Baltic Journal of Management*, vol. 6, no. 1, 2011, pp. 53-70.

256. G.T. Timbrell, et al., "Impediments to Inter-Firm Transfer of Best Practice in an Enterprise Systems Context," *Proc. The 7th Americas Conference on Information Systems*, 2001, pp. 1084-1090.

257. V. Ward, et al., *Knowledge Brokering: Exploring the Process of Transferring Knowledge into Action*, University of Leeds, 2010.

258. J.L. Cummings and B.-S. Teng, "Transferring R&D Knowledge: The Key Factors Affecting Knowledge Transfer Success," *Journal of Engineering and Technology Management*, vol. 20, no. 1-2, 2003, pp. 39-68.

259. S.I. Tannenbaum and G.M. Alliger, *Knowledge Management: Clarifying the Key Issues*, IHRIM, 2000.

260. M. Jelavic, "Socio-Technical Knowledge Management and Epistemological Paradigms: Theoretical Connections at the Individual and Organisational Level," *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 6, 2011, pp. 1-16.

261. M. Venzin, et al., "Future Research into Knowledge Management," *Knowing in Firms: Understanding, Managing and Measuring Knowledge*, G. von-Krogh, et al., eds., Sage Publications, 2000, pp. 26-66.

262. I. Nonaka, et al., "SECI, Ba and Leadership: A Unified Model of Dynamic Knowledge Creation " *Long Range Planning*, vol. 33, no. 1, 2000, pp. 5-34.

263. P.M. Senge, *The Fifth Discipline: The Art and Practice of the Learning Organization*, Doubleday Business, 1990.

264. G.V. Krogh and J. Roos, "Conversation Management," *European Management Journal*, vol. 13, no. 4, 1995, pp. 390-394.

265. B. Curtis, et al., “A Field Study of the Software Design Process for Large Systems,” *Communications of the ACM*, vol. 31, no. 11, 1988, pp. 1268-1287.

266. Y. Duan, et al., “Identifying Key Factors Affecting Transnational Knowledge Transfer,” *Information and Management*, vol. 47, no. 7-8, 2010, pp. 356-363.

267. C.E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, University of Illinois Press, 1949.

268. K.M. McKenzie, “Exchanging 'PayLoad' Knowledge: Interpersonal Knowledge Exchange within Consulting Communities of Practice,” Swinburne University of Technology, 2002.

269. E. Hollnagel and D.D. Woods, *Joint Cognitive Systems: Foundations of Cognitive Systems Engineering*, CRC Press, 2005.

270. E.M. Rogers, *A History of Communications Study: A Biographical Approach*, The Free Press, 1994.

271. L.H. Ling, “From Shannon-Weaver to Boisot: A Review on the Research of Knowledge Transfer Model,” *Proc. International Conference on Wireless Communications, Networking and Mobile Computing*, IEEE, 2007, pp. 5439-5442.

272. E.C. Nevis, et al., “Understanding Organizations as Learning Systems,” *Sloan Management Review*, vol. 36, no. 2, 1995, pp. 73-85.

273. A.C. Inkpen and A. Dinur, *The Transfer and Management of Knowledge in the Multinational Corporation: Considering Context*, Working Paper 98-16, Carnegie Bosch Institute, 1998.

274. A.L. Chua and S.L. Pan, “Knowledge Transfer and Organizational Learning in IS Offshore Sourcing,” *Omega*, vol. 36, no. 2, 2008, pp. 267-281.

275. K. Yokozawa, et al., “A Conceptual Model for the International Transfer of Japanese Management Systems,” *Proc. The 14th International Annual EurOMA Conference*, 2007.

276. V.A. Cooper and S. Lichtenstein, “Supporting Knowledge Transfer in Web-Based Managed IT Support,” *Journal of Systems and Information Technology*, vol. 12, no. 2, 2010, pp. 140-160.

277. C.Y. Li and C.T. Hsieh, “The Impact of Knowledge Stickiness on Knowledge Transfer Implementation, Internalization, and Satisfaction for Multinational Corporations,” *International Journal of Information Management*, vol. 29, no. 6, 2009, pp. 425-435.

278. G. Elwyn, et al., “Sticky Knowledge: A Possible Model for Investigating Implementation in Healthcare Contexts,” *Implementation Science*, vol. 2, no. 44, 2007.

279. E.W.K. Tsang, “Transferring Knowledge to Acquisition Joint Ventures: An Organizational Unlearning Perspective,” *Management Learning*, vol. 39, no. 1, 2008, pp. 5-20.

280. V. Albino, et al., “Knowledge Transfer and Inter-Firm Relationships in Industrial Districts: The Role of the Leader Firm,” *Technovation*, vol. 19, no. 1, 1999, pp. 53-63.

281. S. Betz, et al., “Knowledge Transfer in IT Offshore Outsourcing Projects: An Analysis of the Current State and Best Practices,” *Proc. The 5th IEEE International Conference on Global Software Engineering*, IEEE, 2010, pp. 330-335.

282. J. Soini, “An Approach to Knowledge Transfer in Software Measurement,” *Informatica*, vol. 31, 2007, pp. 437-446.

283. C. Scott and S. Sarker, “Examining the Role of the Communication Channel Interface and Recipient Characteristics on Knowledge Internalization: A Pragmatist View,” *IEEE Transactions on Professional Communication*, vol. 53, no. 2, 2010, pp. 116-131.

284. L. Pérez-Nordtvedt, et al., “Effectiveness and Efficiency of Cross-Border Knowledge Transfer: An Empirical Examination,” *Journal of Management Studies*, vol. 45, no. 4, 2008, pp. 714-744.

285. K.U. Koskinen, “Metaphoric Boundary Objects as Co-ordinating Mechanisms in the Knowledge Sharing of Innovation Processes,” *European Journal of Innovation Management*, vol. 8, no. 3, 2005, pp. 323-335.

286. B.K. Brockman and R.M. Morgan, “The Role of Existing Knowledge in New Product Innovativeness and Performance,” *Decision Sciences*, vol. 34, no. 2, 2003, pp. 385-419.

287. M. Strohmaier, et al., “Knowledge Problems in Process-Oriented Organizations: A Pattern Approach,” *The 3rd Conference Professional Knowledge Management - Experiences and Visions*, K.-D. Althoff, et al., eds., 2005, pp. 241-244.

288. S. Sarker, et al., “Knowledge Transfer in Virtual Systems Development Teams: An Exploratory Study of Four Key Enablers,” *IEEE Transactions on Professional Communication*, vol. 48, no. 2, 2005, pp. 201-218.

289. A.K. Gupta and V. Govindarajan, “Knowledge Flows Within Multinational Corporations,” *Strategic Management Journal*, vol. 21, 2000, pp. 473-496.

290. W.M. Cohen and D.A. Levinthal, “Absorptive Capacity: A New Perspective on Learning and Innovation,” *Administrative Science Quarterly*, vol. 35, no. 1, 1990, pp. 128-152.

291. G. Szulanski, “Appropriating Rents from Existing Knowledge: Intra-firm Transfer of Best Practice,” INSEAD, Fontainbleau, 1995.

292. R. Reagans and B. McEvily, “Network Structure and Knowledge Transfer: The Effects of Cohesion and Range,” *Administrative Science Quarterly*, vol. 48, no. 2, 2003, pp. 240-267.

293. R.T.A.J. Leenders, et al., “Virtuality, Communication, and New Product Team Creativity: A Social Network Perspective,” *Journal of Engineering and Technology Management*, vol. 20, no. 1-2, 2003, pp. 69-92.

294. P.R. Carlile, “A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development,” *Organization Science*, vol. 13, no. 4, 2002, pp. 442-455.

295. S.M. Jasimuddin, “Exploring Knowledge Transfer Mechanisms: The Case of a UK-Based Group within a High-Tech Global Corporation,” *International Journal of Information Management*, vol. 27, no. 4, 2007, pp. 294–300.

296. S.O.S. Syed-Ikhsan and F. Rowland, “Knowledge Management in a Public Organization: A Study on the Relationship between Organizational Elements and the Performance of Knowledge Transfer,” *Journal of Knowledge Management*, vol. 8, no. 2, 2004, pp. 95-111.

297. M. Alavi, et al., “An Empirical Examination of the Influence of Organizational Culture on Knowledge Management Practices,” *Journal of Management Information Systems*, vol. 22, no. 3, 2005, pp. 191-224.

298. C. Liyanage, et al., “Knowledge Communication and Translation-A Knowledge Transfer Model,” *Journal of Knowledge Management*, vol. 13, no. 3, 2009, pp. 118-131.

299. A.M. Ortiz-Laverde, et al., “Knowledge Processes: On Overview of the Principal Models,” *Proc. The 3rd European Knowledge Management Summer School*, 2003, pp. 1-6.

300. A.C.L. DeMeyer, “Tech Talk: How Managers are Stimulating Global R and D Communication,” *Sloan Management Review*, vol. 32, no. 3, 1991, pp. 49–58.

301. T.J. Allen, *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information within the R and D Organization*, MIT Press, 1977.

302. S. Watson and K. Kewett, “A Multi-Theoretical Model of Knowledge Transfer in Organizations: Determinants of Knowledge Contribution and Knowledge Reuse,” *Journal of Management Studies*, vol. 43, no. 2, 2006, pp. 141-173.

303. M. Gilbert and M. Cordey-Hayes, “Understanding the Process of Knowledge Transfer to Achieve Successful Technological Innovation,” *Technovation*, vol. 16, no. 6, 1996, pp. 301-312.

304. J. Wan, et al., “Research on Knowledge Transfer Influencing Factors in Software Process Improvement,” *Journal of Software Engineering and Applications*, vol. 3, no. 2, 2010, pp. 134-140.

305. G.T.M. Hult, et al., “Organizational Learning in Global Purchasing: A Model and Test of Internal Users and Corporate Buyers,” *Decision Sciences*, vol. 31, no. 2, 2002, pp. 293–325.

306. S.F. Slater and J.C. Narver, “Market Orientation and the Learning Organization,” *Journal of Marketing*, vol. 59, no. 3, 1995, pp. 63–74.

307. B.D. Janz and P. Prasarnphanich, “Understanding the Antecedents of Effective Knowledge Management: The Importance of a Knowledge-Centered Culture,” *Decision Sciences*, vol. 34, no. 2, 2003, pp. 351–384.

308. S. Chen, "Task Partitioning in New Product Development Teams: A Knowledge and Learning Perspective," *Journal of Engineering and Technology Management*, vol. 22, no. 4, 2005, pp. 291-314.

309. J. García, et al., "Design Guidelines for Software Processes Knowledge Repository Development," *Information and Software Technology*, vol. 53, no. 8, 2011, pp. 834-850.

310. P. Jackson and J. Klobas, "Building Knowledge in Projects: A Practical Application of Social Constructivism to Information Systems Development," *International Journal of Project Management*, vol. 26, no. 4, 2008, pp. 329-337.

311. I. Oshri, et al., "Knowledge Transfer in Globally Distributed Teams: The Role of Transactive Memory," *Information Systems Journal*, vol. 18, no. 6, 2008, pp. 593-616.

312. T.L. Roberts, et al., "Utilizing Knowledge Links in the Implementation of System Development Methodologies," *Information and Software Technology*, vol. 43, no. 11, 2001, pp. 635-640.

313. O. Steen, "Practical Knowledge and Its Importance for Software Product Quality," *Information and Software Technology*, vol. 49, no. 6, 2007, pp. 625-636.

314. O. Volkoff, et al., "Enterprise Systems, Knowledge Transfer and Power Users," *The Journal of Strategic Information Systems*, vol. 13, no. 4, 2004, pp. 279-304.

315. M.J. Leseure, et al., "Adoption of Promising Practices: A Systematic Review of the Evidence," *International Journal of Management Reviews*, vol. 5, no. 3-4, 2004, pp. 169-190.

316. J. Chen and R.J. McQueen, "Knowledge Transfer Processes for Different Experience Levels of Knowledge Recipients at an Offshore Technical Support Center," *Information Technology & People*, vol. 23, no. 1, 2010, pp. 54-79.

317. W.D. Hendricson, et al., "Electronic Curriculum Implementation at North American Dental Schools," *Journal of Dental Education*, vol. 68, no. 10, 2004, pp. 1041-1057.

318. J.A. Nickerson and T.R. Zenger, "A Knowledge-Based Theory of the Firm-The Problem-Solving Perspective," *Organization Science*, vol. 15, no. 6, 2004, pp. 617-632.

319. L.D. Kiel, *Knowledge Management, Organizational Intelligence and Learning, and Complexity: v. 3*, EOLSS Publishers Co., Ltd, 2009.

320. A.H. Gold, et al., "Knowledge Management: An Organizational Capabilities Perspective," *Journal of Management Information Systems*, vol. 18, no. 1, 2001, pp. 185-214.

321. P. Neergaard, "Configurations in Quality Management," *Scandinavian Journal of Management*, vol. 18, no. 2, 2002, pp. 173-195.

322. S.L. Ahire and T. Ravichandran, "An Innovation Diffusion Model of TQM Implementation," *IEEE Transactions on Engineering Management*, vol. 48, no. 4, 2001, pp. 445-464.

323. W.A. Taylor and G.H. Wright, "A Longitudinal Study of TQM Implementation: Factors Influencing Success and Failure," *Omega*, vol. 31, no. 2, 2003, pp. 97-111.

324. J. Bessant, et al., "Putting Supply Chain Learning into Practice," *International Journal of Operations & Production Management*, vol. 23, no. 2, 2003, pp. 167-184.

325. J.A.-M. Coyle-Shapiro and P.C. Morrow, "The Role of Individual Differences in Employee Adoption of TQM Orientation," *Journal of Vocational Behavior*, vol. 62, no. 2, 2003, pp. 320-340.

326. S.L. Brown and K.M. Eisenhardt, "Product Development: Past Research, Present Findings, and Future Directions," *The Academy of Management Review*, vol. 20, no. 2, 1995, pp. 343-378.

327. S.C. Goh, "Managing Effective Knowledge Transfer: An Integrative Framework and Some Practice Implications," *Journal of Knowledge Management*, vol. 6, no. 1, 2002, pp. 23-30.

328. M.T. Hansen, "The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge across Organization Subunits," *Administrative Science Quarterly*, vol. 44, no. 1, 1999, pp. 82-111.

329. E. Fernández, et al., "Typology and Strategic Analysis of Intangible Resources: A Resource-Based Approach," *Technovation*, vol. 20, no. 2, 2000, pp. 81-92.

330. R. Chan, "Knowledge Management for Implementation in SMEs," *3rd Annual SAP Asia Pacific*, Institute of Higher Learning Forum, 1999.

331. A. Antonova, et al., “Knowledge Management and Learning in the Organizational Context,” *Proc. 3rd E-Learning Conference*, 2006, pp. 63-67.

332. A. Patel, “Current Status and Future Directions of Software Architectures for Telecommunications,” *Computer Communications*, vol. 25, no. 2, 2002, pp. 121-132.

333. J. Qi, et al., “Knowledge Management in OSS-An Enterprise Information System for the Telecommunications Industry,” *Systems Research and Behavioral Science*, vol. 23, no. 2, 2006, pp. 177-190.

334. TeleManagementForum, *Enhanced Telecom Operations Map (eTOM) - The Business Process Framework for the Information and Communications Services Industry - Release 5.0*, GB921, TeleManagement Forum, 2005.

335. H.H. Chang, et al., “Knowledge Characteristics, Implementation Measures, and Performance in Taiwan Hospital Organization,” *International Journal of Business and Information*, vol. 4, no. 1, 2009, pp. 23-44.

336. I. Bouty, “Interpersonal and Interaction Influences on Informal Resources Exchanges between R&D Researchers Across Organizational Boundaries,” *Academy of Management Journal*, vol. 43, no. 1, 2000, pp. 50-65.

337. L. Argote, et al., “Knowledge Transfer in Organizations: Learning from the Experience of Others,” *Organizational Behavior and Human Decision Processes*, vol. 82, no. 1, 2000, pp. 1-8.

338. J. Fulk and G. DeSanctis, “Electronic Communication and Changing Organizational Forms,” *Organization Science*, vol. 6, no. 4, 1995, pp. 337-349.

339. N. Gorovaia and J. Windsperger, “The Use of Knowledge Transfer Mechanisms in Franchising,” *Knowledge and Process Management*, vol. 17, no. 1, 2010, pp. 12-21.

340. A.C. Inkpen and A. Dinur, “Knowledge Management Processes and International Joint Ventures,” *Organization Science*, vol. 9, no. 4, 1998, pp. 454-468.

341. V.C. Sheer and L. Chen, “Improving Media Richness Theory: A Study of Interaction Goals, Message Valence, and Task Complexity in Manager-Subordinate Communication,” *Management Communication Quarterly*, vol. 18, no. 1, 2004, pp. 76-93.

342. J.F.L. Hong and T.V. Nguyen, “Knowledge Embeddedness and the Transfer Mechanisms in Multinational Corporations,” *Journal of World Business*, vol. 44, no. 4, 2009, pp. 347-356.

343. B. Nicholson and S. Sahay, “Embedded Knowledge and Offshore Software Development,” *Information and Organization*, vol. 14, no. 4, 2004, pp. 329-365.

344. S.B. Basri and R.V. O'Connor, “Knowledge Management in Software Process Improvement: A Case Study of Very Small Entities,” *Knowledge Engineering for Software Development Life Cycles: Support Technologies and Applications*, M. Ramachandran, ed., 2011, pp. 273-288.

345. M.M. Crossan, et al., “An Organizational Learning Framework: From Intuition to Institution,” *The Academy of Management Review*, vol. 24, no. 3, 1999, pp. 522-537.

346. D. Lee, et al., “The Role of Organizational and Individual Characteristics in Technology Acceptance,” *International Journal of Human-Computer Interaction*, vol. 25, no. 7, 2009, pp. 623-646.

347. H.A. Shibly, “Human Resources Information Systems Success Assessment: An Integrative Model,” *Australian Journal of Basic and Applied Sciences*, vol. 5, no. 5, 2011, pp. 157-169.

348. S. Funilkul, et al., “An Evaluation Framework for e-Government Services Based on Principles Laid Out in COBIT, the ISO 9000 Standard, and TAM,” *Proc. The 17th Australasian Conference on Information Systems*, AIS Electronic Library, 2006, pp. Paper 3.

349. J.C. Henderson and S. Lee, “Managing I/S Design Teams: A Control Theories Perspective,” *Management Science*, vol. 38, no. 6, 1992, pp. 757-777.

350. U. Lichtenthaler, “Absorptive Capacity, Environmental Turbulence, and the Complementarity of Organizational Learning Processes,” *The Academy of Management Journal*, vol. 52, no. 4, 2009, pp. 822-846.

351. G.S. McMillan, et al., "The Impact of Publishing and Patenting Activities on New Product Development and Firm Performance: The Case of the US Pharmaceutical Industry," *International Journal of Innovation Management*, vol. 7, no. 2, 2003, pp. 213-221.

352. L.R. Newey and A.D. Shulman, "Systemic Absorptive Capacity: Creating Early-to-Market Returns through R&D Alliances," *R&D Management*, vol. 34, no. 5, 2004, pp. 495-504.

353. A.J. Davies and A.K. Kochhar, "A Framework for the Selection of Best Practices," *International Journal of Operations & Production Management*, vol. 20, no. 10, 2002, pp. 1203-1217.

354. A. Petroni, "Critical Factors of MRP Implementation in Small and Medium-Sized Firms," *International Journal of Operations & Production Management*, vol. 22, no. 3, 2002, pp. 329-348.

355. T. Guimaraes, "Field Testing of the Proposed Predictors of BPR Success in Manufacturing Firms," *Journal of Manufacturing Systems*, vol. 18, no. 1, 1999, pp. 53-65.

356. J.M. Beyer, et al., "Contrasts in Enacting TQM: Mechanistic vs. Organic Ideology and Implementation," *Journal of Quality Management*, vol. 2, no. 1, 1997, pp. 3-39.

357. S. Sundaresan and Z. Zhang, "Facilitating Knowledge Transfer in Organizations through Incentive Alignment and IT Investment," *Proc. The 37th Annual Hawaii International Conference on System Sciences*, IEEE, 2004.

358. G. Szulanski, "Appropriability on the Challenge of Scope: Bank One Routinizes Replication," *The Nature and Dynamics of Organizational Capabilities*, G. Dosi, et al., eds., Oxford University Press, 2000.

359. S. Drew, "BPR in Financial Services: Factors for Success " *Long Range Planning*, vol. 27, no. 5, 1994, pp. 25-41.

360. K.-C. Kim and I. Im, "The Effects of Electronic Supply Chain Design (e-SCD) on Coordination and Knowledge Sharing: An Empirical Investigation," *Proc. The 35th Annual Hawaii International Conference on System Sciences*, IEEE, 2002, pp. 2149 - 2158

361. D. Šmite and C. Wohlin, "Software Product Transfers: Lessons Learned from a Case Study," *Proc. The 5th IEEE International Conference on Global Software Engineering*, IEEE, 2010, pp. 97-105.

Appendices

Appendix A: Semi-Structured Interview Guide for Chapter 2

Questions for warm-up:

1. What is your professional background?
2. What is your role and responsibility in software development lifecycles?

Main body of the interview:

1. What software development methods are used for software projects?
2. What are your project characteristics?
3. What are your team characteristics?
4. How is your working environment, e.g., relationships between team members, communications, and sufficient commitment and support from management?
5. How do you perform software development (e.g., what software process and how to perform)?
6. What are challenges and issues that you encountered in software projects? Can those challenges and issues be solved? Why and why not?
7. What are the critical problems?
8. How is the degree of requirements volatility? How do you deal with it?
9. Is the work done and delivered as committed? Why and why not?
10. Are you satisfied with the software products? Why and why not?
11. How do you share knowledge to your team members?

Appendix B: Questionnaire for Chapter 4

Company: _____

Name: _____

Education (Bachelor's, Master's, or Doctoral degrees):_____

Current position: _____

Years in that position: _____

Years in agile software development:_____

Agile methods currently used (if any, e.g., Scrum, eXtreme Programming (XP), and Feature Driven Development (FDD): _____

Please rank the following agile practices from 1 to 5 (1=Not at all important or not implemented; 2=Not very important; 3=Quite Important; 4=Very Important; and 5=Extremely important and need to be implemented) as well as suggest your agile practices that your organization is implementing for success in agile software development.

(Note: Cronbach's alpha (α) was used to evaluate the reliabilities of this questionnaire instrument. The results reveal that the Cronbach's alpha scale of the entire scale is 0.878 and the Cronbach's alpha scales of the items range between 0.864 and 0.888 as presented in the Table. All items having the coefficient of above 0.7 demonstrate acceptable reliability [192].)

Influential Factors	**Lists of Practices**	**Rating**	**α**
Agile Software Development Process (SD)	SD1. A project has been established with well-defined coding standards up front. SD2. A project has been established by pursuing simple design. SD3. A project has been established with rigorous refactoring activities. SD4. A project has been established with right amount of documentation. SD5. A project has been established with correct integration testing. SD6. A project has been established with short increments. SD7. Most important features have been first delivered. Your agile practices: 1. 2. 3.		0.873 0.874 0.879 0.871 0.878 0.878 0.875
Appropriate Methods, Techniques, and Tools (MT)	MT1. Appropriate methods, techniques and tools have been assessed and performed. Your agile practices: 1. 2. 3.		0.887

Influential Factors	Lists of Practices	Rating	α
Data Quality (DQ)	DQ1. Plans or strategies to address data quality problems have been performed.		0.872
	DQ2. Common data standards or guidelines have been conducted.		0.874
	DQ3. Software development teams have their own working environments.		0.871
	DQ4. Basic skills have been trained to people who relevant to data quality.		0.872
	DQ5. Data governance to ensure the quality, availability, integrity, security, and usability has been performed.		0.878
	DQ6. Database regression testing has been performed.		0.882
	DQ7. Many types of database testing (e.g., database input, database output, stored procedures, column constraints, default column values) have been performed.		0.888
	DQ8. The data aspects of software have been modeled iteratively and incrementally.		0.885
	Your agile practices: 1. 2. 3.		
Management Commitment (MC)	MC1. Management provides strong commitment and presence.		0.884
	MC2. Management supports the software development.		0.880
	MC3. Management is willing to participate in assessment and development activities.		0.876
	MC4. Management is committed to provide training and resources.		0.876
	Your agile practices: 1. 2. 3.		
Organizational Environment (OE)	OE1. Cooperative organizational culture has been established instead of hierarchical culture.		0.875
	OE2. Oral culture placing high value on face-on-face communication has been established.		0.865
	OE3. Agile has been promoted and accepted throughout the organization.		0.877
	OE4. All the key stakeholders are involved in development and improvement activities.		0.873
	OE5. Management has provided strong leadership-collaboration; meaning management understands that collaboration on information to make informed decisions and trusting individuals to apply their competency in effective ways is important.		0.875
	OE6. Facility with proper agile-style work environment has been established.		0.878
	OE7. Reward system appropriate for agile software development has been promoted amongst the management and team members.		0.878
	Your agile practices: 1. 2. 3.		

Influential Factors	Lists of Practices	Rating	α
Project Management Process (PM)	PM1. Agile-oriented project management process has been followed. PM2. Cost evaluation has been done up front. PM3. Risk analysis has been done up front. PM4. A process has been established to monitor and track the progress of the project. PM5. Strong face-to-face communication has been established as a primary communication method. PM6. Teams have honored their regular working schedule. PM7. Work has been done to continuously improve a project management process. Your agile practices: 1. 2. 3.		0.878 0.878 0.882 0.887 0.879 0.881 0.868
Project Type (PT)	PT1. Project characteristics (i.e., extreme, complex, or high-change) have been assessed the suitability of software process development PT2. Project criticality (i.e., life-critical and non-life-critical) has been assessed the suitability of software process development. Your agile practices: 1. 2. 3.		0.872 0.876
Reviews (RE)	RE1. Organization has developed a review process for development and improvement requirements. RE2. Work has been done to continuously monitor existing software development processes. RE3. Organization has developed a process in order to review influential factors of software development. RE4. Responsibilities have been assigned to conduct continuous software process development and improvement reviews within organization. RE5. All the key stakeholders are involved in software process development and improvement reviews. Your agile practices: 1. 2. 3.		0.871 0.876 0.864 0.873 0.876
Team Capability (TC)	TC1. People have been selected as team members who have high competence and expertise. TC2. People have been elected as team members who have great motivation. TC3. People have been selected as project managers or team leaders who have an adaptive management style. TC4. People have been selected as project managers or team members who are knowledgeable in an agile process. TC5. People who have track record of different successful software projects have been selected for development activities. TC6. Role and responsibilities have clearly been assigned to each team		0.874 0.877 0.868 0.871 0.873 0.868

Influential Factors	Lists of Practices	Rating	α
	member. TC7. A process has been established to monitor the progress of each team member. TC8. A process has been established to collect and analyze the feedback data from each team member and to extract the main lessons learned. Your agile practices: 1. 2. 3.		0.878 0.873
Team Environment (TE)	TE1. Collocation of the whole team has been established. TE2. Coherent and self-organizing teamwork has been established. TE3. A project has been established with no multiple independent teams. TE4. A process has been established to monitor the progress of each team. TE5. A process has been established to collect and analyze the feedback data from each team and to extract the main lessons learned. TE6. A process has been established to distribute the lessons learned to the relevant stakeholders and team members. TE7. Team members are aware of their roles and responsibilities during software development and improvement. Your agile practices: 1. 2. 3.		0.883 0.879 0.875 0.882 0.879 0.873 0.880
Team Size (TS)	TS1. Project team size has been assessed for the suitability of the project. Your agile practices: 1. 2. 3.		0.874
Training Support (TR)	TR1. Appropriate training has been provided to team members for developing the skills and knowledge needed to perform the software project. TR2. Sufficient resources and additional time to participate in training will be provided to team members. TR3. Training program activities are reviewed on a periodic basis. TR4. All future group or individual trainings of software development are planned. Your agile practices: 1. 2. 3.		0.869 0.869 0.866 0.864
User (Staff) Involvement	UI1. The software development effort has been staffed by people who indicated interest and commitment in the effort. UI2. A project has been established with good user relationship. UI3. A project has been established with user commitment, collaboration,		0.875 0.875 0.880

Influential Factors	Lists of Practices	Rating	α
	and participation. UI4. Users directly involving the project have had full authority. UI5. Work has been done to facilitate team members during software development. UI6. Work has been done to allocate the time necessary to make user participation successful. Your agile practices: 1. 2. 3.		0.883 0.879 0.876

Appendix C: Semi-Structured Interview Guide for Chapter 5

The respondent was informed about the overall interview topics, the important to record the interview, and that only the researcher would have access to the transcript. The respondent is asked if he/she would agree to the interview being recorded. Facilitating discussions, the interview questions are matched with their relevant research questions in Chapter 5 presented as follows. A list of those research questions (RQs) is also presented.

RQ5-1: How can the developed software process maintenance framework be executed efficiently and effectively in the given context?

RQ5-2: What changes are necessary to adapt the developed software process maintenance framework?

RQ5-3: What are the challenges that impact software development, using the developed software process maintenance framework?

RQ5-4: How do practitioners transfer new knowledge into their existing software processes?

RQ5-5: What is the developed software process maintenance framework perceived usefulness and ease of use?

RQ5-6: What are the requirements for successful adaptation of the developed software process maintenance framework?

Questions for warm-up:

Research Question	Interview Question
RQ5-1	1. What is your professional background? (e.g., how long in the company, how long in the telecommunications area, and how long in the software development area) 2. What is your role and responsibility in the case project and earlier software projects?

Main body of the interview with respect to the Software Development Maturity (SDM) model:

Research Question	Interview Question
RQ5-1, RQ5-2	1. What is your existing software process improvement approach? How has it been done in earlier software projects?
RQ5-1	2. Do you agree with the obtained SDM assessment results? Why and why not?
RQ5-3	3. What are the challenges or issues that you encountered in the case project, using the SDM model? 4. What are pros and cons of the SDM model?

RQ5-5	5. Is the SDM model useful? Is the SDM model easy to understand and follow? 6. How do you satisfy with the SDM model? 7. Please rank from 1 to 5 your satisfaction with the SDM model. (1= Strongly Dissatisfy and 5= Strongly Satisfy) 8. Will you continue to use the SDM model?

Main body of the interview with respect to the integrated PMBOK-Scrum model:

Research Question	**Interview Question**
RQ5-1, RQ5-2	1. How do you perform software development? How was it done in earlier software projects? 2. How is your working environment, e.g., relationships between team members, communications, and sufficient commitment and support from management? What are pros and cons of your working environments? How was it done in earlier software projects? 3. What project management aspects are planned? 4. How do you deal with changes in the case project? How was it done in earlier software projects? 5. Is the work done and delivered as committed? Why and why not? How was it done in earlier software projects? 6. How much do you perform documentation? How was it done in earlier software projects? 7. How is your improvement of team performance and productivity? How was it in earlier software projects? 8. How do you perform the verification and validation? How was it done in earlier software projects?
RQ5-1, RQ5-2, RQ5-4	9. How does the team communicate? What communication channels are established? How was it done in earlier software projects? 10. How do you share relevant project information with team members? How was it done in earlier software projects? 11. Do key stakeholders and team members attend all main meetings? How was it done in earlier software projects?
RQ5-1, RQ5-3	12. How do you build the team? What are the challenges or issues that you encountered regarding this matter?
RQ5-2, RQ5-3	13. What are the challenges or issues that you encountered in the case project vs. in earlier software project? Can those challenges or issues be solved in the case project?
RQ5-3	14. What are pros and cons of the software processes (e.g., sprint planning, sprint executions, sprint reviews, and sprint retrospectives)?
RQ5-4	15. How do you share knowledge to your team members? 16. How to integrate the implemented integrated PMBOK-Scrum processes into your existing practices?
RQ5-5	17. Is the integrated PMBOK-Scrum model suitable to your software development? 18. Is the integrated PMBOK-Scrum model useful? Is the integrated PMBOK-Scrum model easy to understand and follow? 19. Are you satisfied with the model? 20. Please rank from 1 to 5 your satisfaction with the integrated PMBOK-Scrum model. (1= Strongly Dissatisfy and 5= Strongly Satisfy) 21. Will you continue to use the integrated PMBOK-Scrum model? 22. Do you have any other opinion about the framework experience?

Main body of the interview with respect to the prototype tool:

Research Question	Interview Question
RQ5-3	1. What are the challenges or issues that you encountered, using the tool? 2. What are pros and cons of the tool?
RQ5-5	3. Is the tool useful? Is the tool easy to use? 4. Are you satisfied with the tool? Why and why not? 5. Please rank from 1 to 5 your satisfaction with the tool. (1= Strongly Dissatisfy and 5= Strongly Satisfy) 6. Will you continue to use tool?

Appendix D: TAM-Based Questionnaire for Chapter 5

Please mark with *X* in the box that indicates your desire.

1. Software Process Maintenance Framework

Perceived Usefulness:

Item	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
PU1. Using the framework would enable me to accomplish my job more quickly.					
PU2. Using the framework would enhance my job performance.					
PU3. Using the framework would improve the quality of software process and product.					
PU4. Using the framework would increase my productivity.					
PU5. Using the framework would enhance my effectiveness on the job.					
PU6. Using the framework would make it easier to do my job.					
PU7. I would find the framework advantageous in my job.					

Perceived Ease of Use:

Item	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
PEOU1. Learning to use the framework would be easy for me.					
PEOU2. My interaction with the framework would be clear and understandable.					
PEOU3. It was easy to become skilful using the framework.					
PEOU4. It was easy to remember how to perform tasks using the framework.					
PEOU5. I would find the framework easy to use.					

Intention to Use:

Item	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
IU1. Assuming the framework would be available on my job, I would use the framework to assess the implemented process and software development maturity on a regular basis in the future.					
IU2. Assuming the framework would be available on my job, I would use the framework to develop an integrated PMBOK-Scrum process on a regular basis in the future.					
IU3. I would continue using the framework to assess the implemented process and software development maturity.					
IU4. I would continue using the framework to develop an integrated PMBOK-Scrum process.					

2. Software Process Assessment and Development Tool

Perceived Usefulness:

Item	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
PU1. Using the tool would enable me to accomplish my job more quickly.					
PU2. Using the tool would improve job performance.					
PU3. Using the tool would improve the quality of the software process and product.					
PU4. Using the tool would increase my productivity.					
PU5. Using the tool would enhance my effectiveness on the job.					
PU6. Using the tool would make it easier to do my job.					
PU7. I would find the tool advantageous in my job.					

Perceived Ease of Use:

Item	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
PEOU1. Learning to use the tool would be easy for me.					
PEOU2. My interaction with the tool would be clear and understandable.					
PEOU3. It was easy to become skilful using the tool.					
PEOU4. It is easy to remember how to perform tasks using the tool.					
PEOU5. I would find the tool easy to use.					

Intention to Use:

Item	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
IU1. Assuming the tool would be available on my job, I would use the tool to assess the implemented process and software development maturity on a regular basis in the future.					
IU2. Assuming the tool would be available on my job, I would use the tool to define an integrated PMBOK-Scrum process and plan a project on a regular basis in the future.					
IU3. I would continue using the tool to assess the implemented process and software development maturity.					
IU4. I would continue using the tool to define an integrated PMBOK-Scrum process and plan a project.					

CURRICULUM VITAE

Nalinpat Porrawatpreyakorn

Faculty of Information Technology

King Mongkut's University of Technology North Bangkok, Thailand

Postal Address: Nawamintararachinee Tower, Floors 4-7
King Mongkut's University of Technology North Bangkok (KMUTNB)
518 Pibulsongkram Road, Bangsue, Bangkok, Thailand, 10800

Phone: (66) 2912-2019, (66) 2913-2500-24 ext. 2701, 2704, 2714

Email: nalinpat.por@gmail.com

Website: http://www.it.kmutnb.ac.th/

Education

2009-2012: Ph.D. in Computer Science, University of Vienna, Austria

2003-2006: Master of Science in Electronic Business, King Mongkut's University of Technology Thonburi, Thailand

1998-2002: Bachelor of Science in Applied Computer Science, King Mongkut's University of Technology North Bangkok, Thailand

Professional Experience

2006-2007: Senior Officer
Basel II Program Office, Risk Management Department
TMB Bank Public Company Limited, Thailand

2004-2005: Programmer
OLAP Unit, Management Information System Department
Thai Wacoal Public Company Limited, Thailand

2002-2004: Programmer
System Support Unit, Treasury Division
Bangkok Bank Public Company Limited, Thailand

Publications

1. Porrawatpreyakorn, N., Quirchmayr, G., and Chutimaskul, W. 2009, 'Requirements for a Knowledge Transfer Framework in the Field of Software Development Process Management for Executive Information Systems in the Telecommunications Industry', in Papasratorn, B., Chutimaskul, W., Porkaew, K., and Vanijja, V. (eds), Proceedings of the

3rd International Conference on Advances in Information Technology, Springer Berlin Heidelberg, Bangkok, Thailand, vol. 55, pp. 110-122.
2. Porrawatpreyakorn, N., Quirchmayr, G., and Chutimaskul, W. 2010, 'Requirements for a Software Process Maintenance Framework for Executive Information Systems in the Telecommunications Industry', Journal of Global Management Research, vol. 6. No. 1, pp. 7-18.
3. Porrawatpreyakorn, N., Quirchmayr, G., and Chutimaskul, W. 2010, 'A Prototype for the Support of Integrated Software Process Development and Improvement', in Papasratorn, B., Chutimaskul, W., Porkaew, K., and Vanijja, V. (eds), Proceedings of the 4th International Conference on Advances in Information Technology, Springer Berlin Heidelberg, Bangkok, Thailand, vol. 114, pp. 94-105.