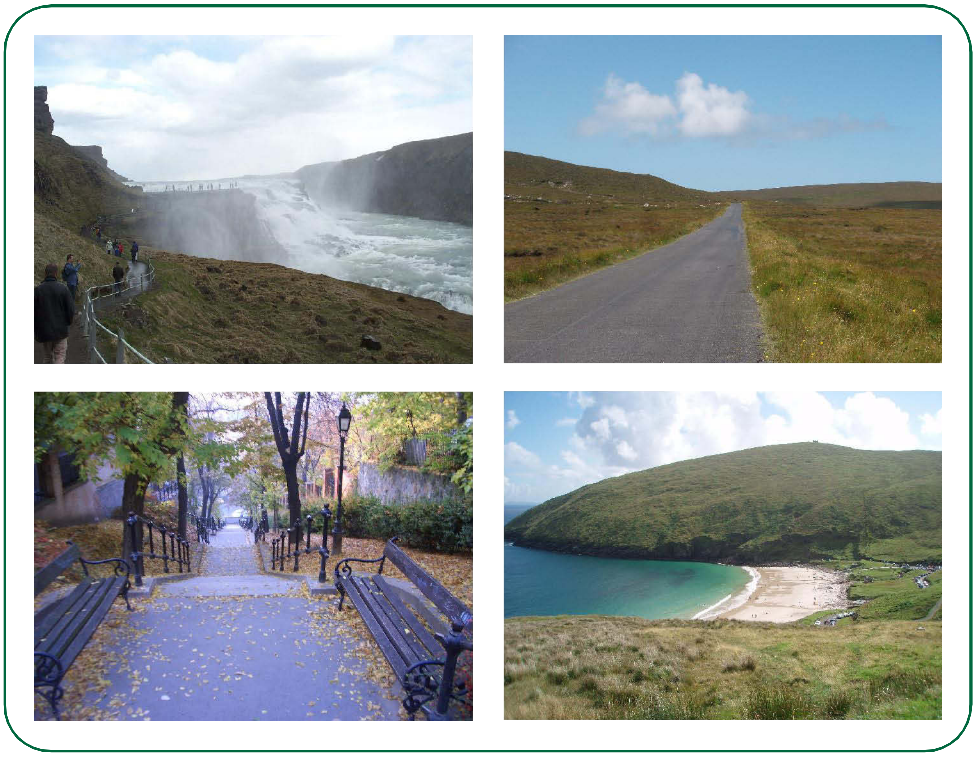
ESPOO 2008 VTT PUBLICATIONS 695



Minna Pikkarainen

Towards a Framework for Improving Software Development Process Mediated with CMMI Goals and Agile Practices

VTT PUBLICATIONS 695

Towards a Framework for Improving Software Development Process Mediated with CMMI Goals and Agile Practices

Minna Pikkarainen

Academic dissertation to be presented, with the assent of the Faculty of Science of the University of Oulu, Department of Information Processing Science, for public defence in Auditorium IT115, Rakentajantie 3, Oulu, on November 10th, 2008, at 12 noon.

ISBN 978-951-38-7121-5 (soft back ed.)
ISSN 1235-0621 (soft back ed.)

ISBN 978-951-38-7122-2 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 3, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 4374

VTT, Bergsmansvägen 3, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 3, P.O. Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax + 358 20 722 4374

VTT, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde 020 722 111, faksi 020 722 2320

VTT, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel 020 722 111, fax 020 722 2320

VTT Technical Research Centre of Finland, Kaitoväylä 1, P.O. Box 1100, FI-90571 OULU, Finland
phone internat. +358 20 722 111, fax +358 20 722 2320

Text preparing Tarja Haapalainen

Edita Prima Oy, Helsinki 2008

Pikkarainen, Minna. Towards a Framework for Improving Software Development Process Mediated with CMMI Goals and Agile Practices. Espoo 2008. VTT Publications 695. 119 p. + app. 193 p.

Keywords CMMI, agile practices, lightweight assessment, communication

Abstract

Problems in software development mainly spring from the difficulty of establishing and stabilizing the requirements, the changeability of the software and interactive dependency of the software, hardware and human beings. A software development process consists of a set of empirical and 'best' practices in software development, together with organization and management that are needed for the software product implementation.

Different process models, such as CMMI (Capability Maturity Model Integration), ISO 9001 and ISO 15504, have been developed in the last decade to support the assessment of software development processes. The main process model, examined in this thesis, is CMMI. This model was chosen as the focus of this research because it is a widely-used, beneficial approach for identifying the key weaknesses of a software development process which need immediate attention and improvement. Two of the key challenges of CMMI assessments are 1) overly heavy and time-consuming assessments and 2) the risk that the achievement of CMMI levels forces the developers to use more time writing documents than implementing the software product.

The level of interest in the use of agile practices (focusing on practices such as eXtreme Programming and Scrum) has radically increased in software organizations. Practitioners argue that the adoption of agile software development methods can solve the organizational need for a more rapid and flexible software development process, and enable improved communication in changing market situations. A brief analysis of the empirical body of knowledge reveals, however, that there are also several challenges in interactive dependency management and communication between the actors of software development in an agile context.

The objective of this study is to increase the understanding of how improvements can be made in the software development process, mediated with CMMI ‘specific’ goals and agile practices from communication perspective. This study is based on a series of case studies and data from 4 companies and 8 software development teams. To prove the importance of the improvement approach, this study starts with an evaluation of the agile practices in current use, using well established ‘innovation of adoption’ theories. The evaluation indicates that agile practices can achieve the subsequent assimilation stages differentially. The results also support the use of an adoption strategy, in which the needs of the teams are first defined before mapping the agile practice-based improvement solutions to the project level challenges.

Although the iteration retrospectives provide a practical way for improving a software development process at team level, companies need mechanisms to constantly implement improvement initiatives and share knowledge of the process status also at organizational level. To meet this gap in the current empirical body of knowledge and research, a novel framework is presented in this study. The framework can be used 1) to identify the agile practices for a plan-driven software development process and 2) to assess the software development process in a lightweight manner against the CMMI goals and agile practices.

To indicate the value of the created framework, it is important to collect empirical evidence on how agile practices actually affect communication in the software development process. This study applies coordination theory to confirm that the adoption of agile practices, such as sprint planning, an open office space, daily meetings and product backlogs improve the communication and management of requirements, features and project task dependencies in agile software development teams. Additionally, increased informal communication can in some cases decrease the need for upfront documentation in software development teams and, therefore, facilitate more productive software development than in previous plan driven situations.

Preface

During the preparation of this thesis, I have collaborated with many great people including groups of researchers and customers from many organizations. The Technical Research Centre of Finland (VTT), Tekes, the Nokia Foundation and the Irish Software Engineering Centre (Lero) provided funding for the research. A special thanks to my supervisor Pekka Abrahamsson, whose advice has been crucial to the creation of this thesis. Thanks also to Samuli Saukkonen from the University of Oulu for his reviews and Brian Fitzgerald from the University of Limerick for his great advice during the research. This research would not have been possible without my friends and colleagues Outi Salo, Annukka Mäntyniemi, Xiaofeng Wang, Fergal Mc Caffery and Kieran Conboy who jointly created the research papers with me and who supported me as a friend during the most diffficult moments of this Ph.D. journey.

While carrying out this research, I have worked on the ITEA projects Agile and Flexi. I would like to express my thanks to all these project groups, and especially to Ulla Passoja, Mika Koivupalo, Tuomo Kähkönen, Vasco Duarte and Jari Still who provided me with an opportunity to see how agile practices were adopted and used in their organizations. The manuscript of this thesis was also reviewed by Frank Maurer and Tore Dybå. I am grateful for their insightful and comprehensive comments that greatly improved the final outcome of this thesis. During the research process, I had the amazing opportunity to be in Ireland and work with all those lovely people from Lero and learn so much more about the research. From Lero I would like to thank Gary Gaughan and Eoin O'ccur, who shared an office with me for one year and always made me laugh during the most boring moments of the paper re-writing process. My thanks also to my mother, and sisters for their patience and finally thanks to my love Sami Härkönen and my cute children Jesse and Jasmin Pikkarainen, who gave me the happiness and strength every day that I needed to do this research.

Kempele, Finland

Minna Pikkarainen

Contents

List of Original Publications

I Pikkarainen, M., Wang, X. & Conboy, K. 2007. Agile practices in Use from an Innovation Assimilation Perspective. A Multiple Case Study, ICIS 2007. Montreal, Canada. 31 p.

II Pikkarainen, M. & Mäntyniemi, A. 2006. An approach Using CMMI in Agile Software Development Assessments: Experiences from Three Case Studies. SPICE 2006. Luxemburg. 22 p.

III Pikkarainen, M. & Passoja, U. 2005. An Approach for Assessing Suitability of Agile Solutions: A Case Study, The Sixth International Conference on Extreme Programming and Agile Processes in Software Engineering, Sheffield University, UK. 14 p.

IV McCaffery, F. Pikkarainen, M. & Richardsson, I. 2008. AHAA – Agile, Hybrid Assessment Method for Automotive, Safety Critical SMEs, ICSE 2008. 31 p.

V Pikkarainen, M., Salo, O. & Still, J. 2005. Deploying Agile practices in Organizations: A Case Study. In: European Software Process Improvement and Innovation (EuroSPI 2005), Budapest, Hungary. 20 p.

VI Pikkarainen, M., Haikara, J., Salo, O. & Abrahamsson, P. 2008. The Impact of Agile practices on Communication in Software Development. Empirical Software Engineering, Vol. 13, No. 3, pp. 303–337. 58 p.

List of Names and Acronyms

CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
QIP	Quality Improvement Paradigm
SEI	Software Engineering Institute
SPICE	Software Process Improvement and Capability Determination, ISO 15504
TDD	Test Driven Development
VTT	Technical Research Centre of Finland
XP	eXtreme Programming
APM	Agile Project Management
FDD	Feature Driven Design
LSD	Lean Software Development
IT	Information Technology
CR	Change Request
SME	Small-Medium Enterprise
PIW	Post Iteration Workshop
AHAA	Agile Hybrid Assessment method for Automotive, Safety Critical Small Enterprises
ASD	Adaptive Software Development
SPI	Software Process Improvement
FDD	Feature Driven Development

1. Introduction

In the mid-1990s, many developers found the initial requirements and documentation steps frustrating and difficult to implement in practice (Williams and Cockburn 2003). Requirements and plans got out of the date even in short software development projects (Williams and Cockburn 2003). Plan-driven software development methods are typically characterized as systematic engineering approaches, where continuous SPI strategies such as CMMI or ISO 15504 (i.e. SPICE) based assessments are applied in order to define improvement needs for high maturity processes (Boehm and Turner 2003a). CMM and more recently CMMI is regarded as the most popular reference model used in assessments as the first step of SPI (Agrawal, and Chari 2007) and it has been used, for example, to enhance the reduced costs of software development (Galin and Avrahami 2006). On the other hand, it has been argued that CMM is too heavy-weight a model for software development projects (Ramachandran 2005) and that the use of CMMI could lead an organization or team to an overly document-driven and structured software development approach (DeMarco and Boehm 2002, Highsmith 2002b).

Agile methods, such as eXtreme Programming (XP) (Beck 2000) and Scrum (Schwaber and Beedle 2002), promise practices for improved collaboration, communication and project management (Williams and Cockburn 2003). This is because, in agile software development, the planning is made more frequently than in so called "plan-driven software development". The constant planning enables these "planning driven teams" to respond to changes quickly (Wang et al. 2008). These are some of the reasons why agile methods have been increasingly attractive to software intensive companies (See example of the use Karlström and Runeson 2006, Cohn and Ford 2003, Drobka et al. 2004, Dybå and Dingsøyr 2008, Fitzgerald et al. 2006, Rasmusson 2003, Svensson and Höst 2005).

The usefulness of XP and Scrum practices, however, vary between organizations and projects (Salo and Abrahamsson 2008). This means that at the same time when some of the XP and Scrum practices such as (1) collective code ownership, (2) 40 h week, (3) coding standards and (4) simple design, (5) product backlogs and (6) sprint planning meetings seem to be very useful for companies, some other XP and Scrum practices such as (1) pair programming,

(2) TDD, (3) On-site customers and (4) product backlogs have been discovered to have negative effects (i.e. were not seen useful) on the software development organizations (Salo and Abrahamsson 2008).

Additionally, the deployment of agile methods demands acquiring, assimilation, transformation and exploitation of new knowledge (Cohen et al. 2004). Therefore, it is not a surprise that several case studies report the adoption of agile software development methods as a challenging activity (Svensson and Höst 2005). There are even cases in which decreased productivity rates have been reported during the deployment while the team had to take time to learn new methods (Cohn and Ford 2003). Thus, sometimes the fast introduction of agile methods in individual pilot projects can cause a situation in which the rest of the organization is left without knowledge of what and how the agile pilot projects are doing (Cohn and Ford 2003). Usually the goal is to adopt agile practices to apply the suitable agile practices as a part of the organization's previous plan-driven software development process (Manhart and Schneider 2004) and, therefore, to find a balance in both the agile and traditional approaches to take advantage of their strengths and compensate for their weaknesses (Boehm and Turner 2003a).

The relationship between CMMI and the software development process in which agile practices have been used has been discussed in several empirical reports, but only in a few research journals (Boehm 2002, Cohen et al. 2004, Glazer 2001, Highsmith 2002b, Paulk 2001). Many have criticized the use of CMMI based assessments in the software development process in which agile practices have been used. For example, Boehm (2002) argues that agile methods are a reaction against traditional methodologies, also known as plan-driven methodologies. Turner and Jain (2002) indicate that the companies using agile methods face a risk of emphasizing too much tacit knowledge and too less formal communication across the team. One reason for the concern is that the tacit, informal communication is in many cases dependent on the persons' experience and capability of sharing information between each other (Turner and Jain 2002). The software development process in which agile practices are used does not, however, include only informal communication. Formal communication, such as source code, test cases, and a minimum, essential amount of documentation is also used in the agile software development process but not in the same way or in the same extension as in the plan-driven software development process (Turner and Jain 2002).

There are only a few empirical reports in which CMMI has been used when assessing software development processes where agile practices are used. It has been, however, suggested that it is possible to achieve CMMI levels 2 and 3 process areas using Scrum and XP practices (Cohen et al. 2004, Paulk 2002, Vriens 2003). Furthermore, some argue that most XP projects, that truly follow XP practices, could be assessed at the CMMI level 2 (Glazer 2001, Kähkönen and Abrahamsson 2004, Paulk 2001). Anderson (2005) even argues that the CMMI level 5 would be possible to achieve using agile methods. Perhaps, it is, however as Jeffries (2002) points out that agile methods are: "*in some ways a 'vertical' slice through the level 2 through 5*" (Cohen et al. 2004).

Thus, organizations that are accustomed to improving their process capability utilizing standards (such as SPICE) and models (such as CMMI) seem to have limited tools to identify suitable agile practices for their specific software development context and to continuously improve the agile based software development process (Boehm and Turner 2003a). The current literature does not clearly report how agile practices contribute to the CMMI process areas (Fritzsche and Keil 2007) or how CMMI initiatives and agile practices can be used together to improve the software development process (Sidky 2007). In addition, both research and practice have much to learn about SPI efforts using CMMI (Sidky 2007).

In recognition of these problems, the question of how to improve software development processes mediated with CMMI and agile practices from communication perspective becomes relevant.

1.1 Research Questions

At the same time as agile practices are increasingly adopted in organizations (e.g. Karlström and Runeson 2006, Cohn and Ford 2003, Drobka et al. 2004, Dybå and Dingsøyr 2008, Fitzgerald et al. 2006, Rasmusson 2003, Svensson and Höst 2005), CMMI is being used as one reference models for assessing a software development process as a part of the overall SPI programs to reduce costs of software development (Galin and Avrahami 2006). Currently, many companies that have used or have a need to use CMMI based assessments have also a need for agile practices due to, for example, their needs for more effective communication and collaboration practices.

Many have argued that CMMI based assessments and agile practices could be used as a combined approach to integrate the best abilities of both the agile methods and the CMMI process areas (Boehm and Turner 2003a, Paulk 2001, Kähkönen and Abrahamsson 2004). Current research, however, shows only a few case studies related to the use of agile practices and a lack of an approach that integrates these aspects together. Therefore, there is a need for some research focusing on the following one main research question:

Q.1: How to improve the software development process mediated with CMMI goals and agile practices from communication perspective?

This question is divided into three sub-questions:

Q.1.1: How to facilitate the improvement of the software development process mediated with CMMI and the agile practices? Currently, many software intensive companies have a parallel need to 1) use the agile practices due to business demands and 2) to improve and maintain their process capability utilizing reference models such as CMMI (2006). This is the situation, especially, in companies that have a long time investment in SPI or have some other specific need to follow these well known improvement standards or models. However, the current research seems to lack an approach for the adoption of agile practices as a part of SPI program in which assessment is based on the goals of CMMI model.

Q.1.2 How to validate the improvement of the software development process mediated with CMMI and agile practices? As the CMMI model is often defined as a too heavy (Ramachandran 2005) and too document-driven approach (DeMarco and Boehm 2002, Highsmith 2002b), it is unlikely that an official assessment where the CMMI paradigm is used would suit a context of agile software development without modification. Therefore, it is important to define the extent to which the CMMI model can be applied in a lightweight manner without having the teams incur excessive documentation.

Q.1.3 Does the use of agile practices improve the communication in software development teams and between the teams and stakeholders? Communication is an important factor in software development and, thus, a relatively common success factor, when discussing change in software development projects and teams (Stelzer and Mellis 1998). Regular communication is the best way to build trust in teams (Henttonen and Blomqvist 2005) and, thus, make the software development more efficient in companies (Paasivaara and Lassenius 2003). Especially, communication among stakeholders (i.e. customers, management, other development teams) and software development project members and stakeholders is a particular challenge for software development (Damian et al. 2000). Some of the agile principles suggest that business people and developers must work together daily and project information should be shared through informal, face-to-face conversation rather than through documentation. Although it seems that the use of agile practices would increase communication capabilities in software intensive companies, it has been claimed also that the use of agile software development methods can increase the chasm among the actors in software development organizations and even lead to project failure (Boehm and Turner 2003b). Most of these problems may be a consequence of the lack of communication between these actors as identified in many studies in which agile methods are in use (Cohn and Ford 2003, Coram and Bohner 2005, Svensson and Höst 2005). The current research seems, however, lack of approach or discussion on the actual effects of agile practices on communication in software development teams and between the teams and stakeholders.

1.2 Scope of the Research

Software development can be characterized as a series of sequentially organized phases of activities (Clegg et al. 1996) such as design, programming and maintenance that are needed when implementing a software related product (CMMI 2006). Each phase operates with a defined notation and will often result in a prescript artefact such as a design document or a program (Baskerville and Bries-Heje 2001). "Process" as a concept is defined as follows:

> "*A set of activities, methods and transformations or steps that people use to develop and maintain software and its associated products*" (Humphrey 1995, Mathianssen et al. 2002).

System development is

> "*a rational scientific process, which is proposed as a subdivision of the development process into deciding what an information system must do and how it should do it*" (Fitzgerald 1996).

The main SPI model, investigated in this thesis, is the capability maturity model, CMMI (2006). This model is chosen as the focus of this research for a number of reasons, the foremost being that CMMI based assessments integrated with other assessments are a widely-used approach for evaluating the software processes within a company (Trudel et al. 2006) and indicating its key weaknesses for immediate attention and improvement (Daskalantona 1994). Secondly, the beneficial experiences of CMM and CMMI programs have been reported as a part of many studies during the past decades (Galin and Avrahami 2006, Niazi et al. 2003, Stelzer and Mellis 1998). For example, the results of CMM programs in 400 projects report improvements in productivity and development speeds in software development due to the CMMI programs (Galin and Avrahami 2006, Stelzer and Mellis 1998). Based on the results of the analysis, they report a 26 to 187% improvement in productivity, a 28–53% improvement in cycle time and a 120–650% percent return in investment due to the use of CMMI programs (Galin and Avrahami 2006).

CMMI model is provided in two alternative representations continuous and staged. These representations have the same content but different structure. (Curtis et al. 2001). This study focuses using continuous representation of CMMI. This is because the continuous representation offers more flexibility for organizations to select most relevant processes to improve based on their business goals and risks (Curtis et al. 2001).

Project planning, management and requirement management are the CMMI process areas which were examined in more detail in all the case organizations in this study. These processes were selected because previous research suggests that these process areas would provide the most significant benefit for the software development companies (McCaffery et al. 2007, Meehan and Richardson 2002).

This study focuses on investigating eXtreme Programming (XP) (Beck 2000) and Scrum (Schwaber and Beedle 2002). These methods were chosen for a number of reasons. First of all, because they are considered to be the most popular of all the agile methods (Fitzgerald et al. 2006). Secondly, they are very diverse approaches as XP is practitioner-oriented while Scrum focuses on project management (Abrahamsson et al. 2002). By studying these two methods, this study integrates lessons learned from both perspectives. The selected set of agile practices is also based on the engineering approach of using a combined set of Scrum and XP practices as described by Fitzgerald et al. (2006). Furthermore, the selection is further justified because the literature shows that there are few reports and little knowledge in the current literature on how to use and improve a combined, customized set of agile practices. This study is based on XP practices from the first edition of Beck's (2000) XP book. This is because the brief evaluation of XP practices in Chapter 2.2.3 revealed that most of the existing empirical studies, including Fitzgerald et al. (2006), that customized the approach for the agile method tailoring are based on the first, and not the second edition of XP book. In the future, the analysis could be extended to the new practices of Beck and Andres (2004).

Rogers (2003) suggests that an innovation can be an idea or practice, which is perceived as new by the adopters. Based on this definition, agile practices can be characterized as software process innovations. Most of the proprietary agile method literature portrays these methods as new, revolutionary and innovative. Theories on IT innovation adoption, consequently, bring new insights to the study of agile practices in use. Several innovation adoption theories have been built and used to explain the mechanisms and structure of the introduction and implementation of new innovations (Cooper and Zmud 1990, Davis 1989, Fichman 2001, Gallivan 2001). Gallivan's (2001) work is deemed relevant and used in the first part of this study, in which six assimilation stages have been proposed based on the previous work of Cooper and Zmud (1990). These theories provide a background for this study to investigate the adoption of agile practices and the adoption of innovation theories to determine their relevance to the outcome.

Communication is an important way for team members to coordinate complex software development environments (Malone and Crowston 1994). Therefore, communication is also characterized as one central aspect of coordination theory: as an activity that is needed as a way to manage dependencies between actors in the process (Malone and Crowston 1994). The agile principles in agile manifesto (2001) provide some solution proposal for more efficient communication in software intensive companies. This is done emphasizing the collaboration and interaction inside of the teams and between all teams and stakeholder groups. Communication was selected as the research theme of the last phase of this thesis because it has been defined as the first step of an organization towards agility. For example, Sidky (2007) claims that communication needs to be applied first (it is included as the first agile level) because agile principles emphasize "*Individuals and interactions over processes and tools*" (Sidky 2007). On the other hand, Watts Humprey (2000) puts the focus of his book on SPI on communication, thus, highlighting the significance of communication in software related organizations.

In conclusion, the research questions are investigated through case study research which was applied in four software companies and eight development teams. The four companies were selected for the research because they have previous experience or interest in CMMI based assessments and business goals that supported the use of agile practices as part of the software development process.

1.3 Structure of the Thesis

This thesis is based on 6 published research papers (I–VI). In exploring the research phenomenon outlined in this thesis, each individual paper (I–VI) contributes to the increased understanding of improvement in software development process. Different aspects of this theme have been elaborated upon the papers and described in Figure 1.

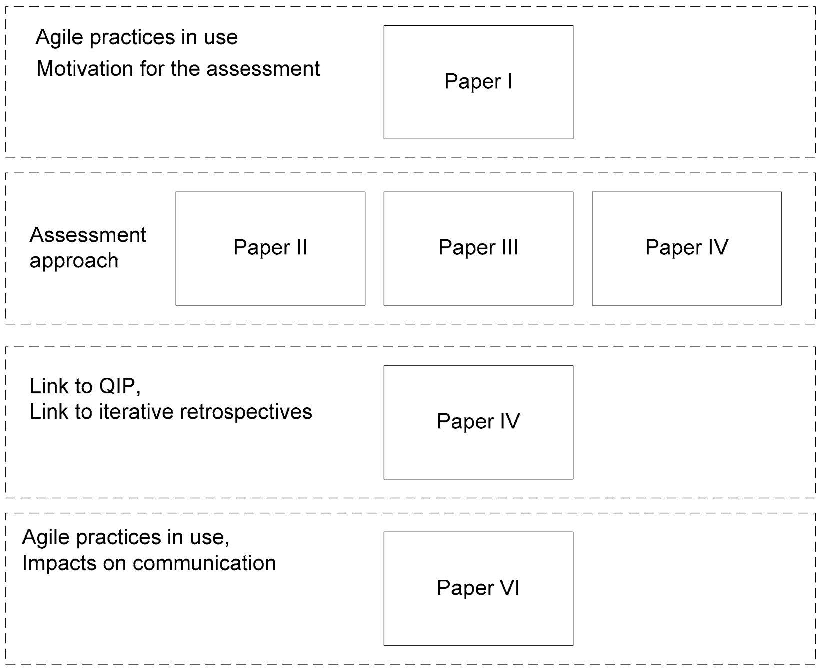


Figure 1. Different aspects of the research phenomenon.

In Paper I innovation adoption theories are applied to interpret the use of agile practices in the three development teams. The paper shows that since the use of agile practices can reach a sophisticated level of use more easily in certain areas where the project team see it as most beneficial, it might be more reasonable to use the adoption strategy in which the need of the project is evaluated to support the identification of relevant agile practices, before their use by the project teams. The author of this thesis was the first author of the Paper I. She was responsible for the case II described in the Paper (which is one of the case companies in this thesis). Cases I and III as well as the data analysis and paper writing were done in close collaboration with Xiaofeng Wang from the University of Limerick and Kieran Conboy from the University of Galway.

Because assessments are a well established approach for evaluating the current status in organizations and teams, Paper II takes the initial steps to clarify how the adoption of agile practices could be supported in assessment. As the answer seems to be to map agile practices under CMMI specific goals, Paper II uses the data of the three case companies to analyse how agile practices are mapped to the specific goals of CMMI and how the use of agile practices affects the assessment of the software development process in which agile practices have already been used. The author of this thesis is the first author of the paper and responsible for the study of all the cases described in Paper II. Paper II was written in close collaboration with Annukka Mäntyniemi from Nokia.

Paper III presents a concept of 'agile assessment', using data from case company 1. The paper describes how to do an agile assessment by focusing on improvement but also identifying the suitable agile practices for the software development organization. It has been suggested in the paper that the assessment should be lightweight – meaning that it does not need to be a complex evaluation including the full analysis of CMMI base practices. It should be based on some of the agile principles, such as face-to-face communication, rapid feedback to interviewees and organization management, and include simple documentation. The author of this thesis is the first author of the paper which was written together with Ulla Passoja, who, at that time, was working as a line manager and quality engineer in the researched case company.

Paper IV integrates the developed 'agile assessment' approach with an existing well established lightweight assessment method called 'ADEPT' (McCaffery et al. 2006, McCaffery et al. 2007) indicating that agile practices are useful and beneficial to use as a part of lightweight assessment also in safety critical software development context. One goal of the paper was also to integrate an agile assessment approach together with the ADEPT assessment method which was developed by McCaffery et al. (2007). The author of this thesis was the second author of Paper IV, but participated in both the case and paper writing work equally with the co-authors Fergal Mc Caffery and Ita Richardson. The author of this thesis participated in the described work in one of the case companies and had the main responsibility for all the agile related parts both in the case itself and in the paper writing.

All three Papers (II–IV) together reveal how an assessment mediated with the specific goals of CMMI and agile practices are used as the approach that enables companies to identify relevant improvement needs and solution alternatives for both agile and plan-driven software development processes.

Paper V integrates the developed assessment approach with steps of the SPI method called QIP (Basili 1989) showing in which steps of the SPI process the ‘agile assessment approach’ could be used. In this paper it is also shown how iterative retrospectives can be integrated with the agile assessment approach as a part of the overall organizational software process improvement. An example from one case company is used to empirically evaluate the presented research assumptions. Paper V was written jointly with Outi Salo who is the key author of several papers related to iterative software process improvement in agile context (e.g. Salo and Abrahamsson, 2007). The third author of the paper is Jari Still who at that time was working as a site manager in the evaluated case company.

In Paper VI the developed improvement approach is examined so as to determine how agile practices affect communication in software development. The paper evaluates the impacts of agile practices on communication in two agile software development teams from one case company using the dependencies of coordination theory. In the paper it is concluded that although the use of agile practices does not automatically guarantee optimized communication between the teams and stakeholders; it provides some valuable mechanisms to improve communication inside the development teams and, thus, might also decrease a need for formal documentation. The assessment was conducted together with the research team and the author gained valuable contributions for the paper from other writers. The author of this thesis is the first author of Paper VI, and she did the main work related to the literature review, case and data analysis described in Paper VI.

The summary part of the thesis is structured as following: Chapter 1 describes the research questions, scope and structure of this thesis; Chapter 2 presents the background to the study and defines the concepts of plan-driven-, agile- and hybrid software development process; Chapter 3 introduces the framework for the thesis, used for integrating the empirical results with the research questions; Chapter 4 sets out the research design including the research approach, methods and organizational context in which the research took place; Chapter 5 lists the research contribution of the empirical research; in Chapter 6 there is a discussion of the research implications with respect to the theory and practice, and Chapter 7 concludes with a summary of the results and describes the limitations and possibilities for future research.

2. Background of the Study

The purpose of the following two sections is primarily to provide some perspective and insight into the topic of this thesis, to identify some existing gaps in models supporting improvement of plan-driven, agile and hybrid software development processes and to briefly look at empirical reports to show that the framework is required to support the improvement of software development process mediated with CMMI goals and agile practices from communication perspective. In this respect, Sections 2.1 and 2.2 describe:

a) what is plan-driven and agile software development
b) what hybrid methods are currently available for the integration of these approaches
c) how existing approaches relate to SPI and the adoption of agile practices.

For this reason, Sections (2.1 and 2.2) do not provide a complete description of the methods, models and practices discussed.

2.1 Plan-Driven – Traditional Software Development

Plan-driven software development is an engineering approach in which the software is developed following specific processes which start from requirements and ends at the finished code (Boehm and Turner 2003a). There are several methodological approaches and models on how to develop the software in a plan-driven way (Boehm and Turner 2003a). Probably the best known of these approaches is the 'waterfall' model (Royce 1970) in which all the development phases are implemented, at least twice at stages after one another to be able to produce the working software. Although Royce's (1970) model is always defined as the most plan-driven way to do the software, the suggestion of Royce is actually to do a 30 month project with a 10 month pilot model which, therefore, gives some hints on an iterative development approach (Larman and Basili 2003).

"*Developers do not develop systems by completing a single task and moving to the next task following a rational sequence*" (Fitzgerald 1996).

Thus, neither the system or software development is a linear process. There is always the need to return back to the specification and designs to correct the already developed software code and, thus, continuously modify the already created documentation. Boehm (1988) claims that the use of the ‘waterfall’ life cycle can lead organizations to a document driven approach which pushes teams to write specifications, interfaces, and decision support functions that are useless and difficult to understand. The argument is that the waterfall model pursues the development projects to perform activities in the wrong order (Boehm 1988). As a solution to these problems of the waterfall type of software development life cycle, Boehm (1988) presents a ‘Spiral’ model to navigate through each phase of the system development process. He notes, however, that one of the key challenges to software project management is not only in the process order but also in the communication among the several stakeholders such as users, customers, maintenance team, management and the software development team. The problems in system development arise because all these actors view the same system in a different perspective (Boehm and Ross 1989).

In iterative development, each iteration is like its own mini-project which includes the phases of requirements analysis, design, programming and testing whereas the incremental software development can be characterized as a development in which the system grows incrementally feature by feature (Larman 2003). The ‘Spiral’ model (Boehm 1988) was developed in 1988 to describe a new, more iterative, order for the software development but also to facilitate communication and trust in the software development teams and organizations. In the 1990s, many, otherwise incremental, plan driven projects were, however, based on long iterations or increments and large well documented requirements analysis (Boehm and Turner 2003a).

SPI standards, models and approaches were developed in the late 1980s due to the so called software crisis, which existed due to late and overrun software projects (Eman and Madhavji 1999). The literature contains several theories and paradigms that describe how to perform SPI. The most well known of these models are IDEAL (McFeeley 1996), QIP (Basili 1989), and plan-do-check-act-cycle (Deming 1990). Software process assessments provide an attractive, practical way of starting the improvement of the software development process. Typically, the assessments are implemented in order to help managers and professionals to identify the most critical problems and to agree on the actions that are required to address them (Humphrey et al. 1991). The assessments are often supported by standards (e.g. ISO 15504 (2006) or ISO 9001 (Agrawal and Chari 2007)) or paradigms (e.g. capability maturity model (CMM) (Curtis et al. 2001), more recently the CMMI (2006)). These ‘standards’ (Agrawal and Chari 2007, Niazi et al. 2003) define how to achieve managed, defined and optimized software development (Boehm and Turner 2003a).

2.1.1 Assessment Approaches

Assessments facilitate improvements in the company software development and management processes (Humphrey et al. 1991):

> *“An organization characterizes the current state of its software process and provides findings and recommendations to facilitate improvement.”* (Humphrey et al. 1991)

Empirical studies have proven that assessments, integrated with the successful implementation of a change, can enable organizations to improve the speed and reduce the costs of the software development (Galin and Avrahami 2006, Niazi et al. 2003). CMMI model defines assessments as an appraisal that an organization does for itself with the purpose of improving processes (Eman and Madhavji 1999). Currently, the literature reveals three different types of assessments which are capability, software product and project assessments (Eman and Madhavji 1999). Firstly, assessments are done using a selected improvement model to evaluate development and support the capability of production (Eman and Madhavji 1999). Secondly, a system assessment aims to assess the maintainability of the software system from an architectural, design and implementation point of view. Thirdly, a project assessment focuses on evaluating that a given product will be delivered with the defined functionality, schedule, budget and quality (Eman and Madhavji 1999).

Software process assessments have been used, with all of the existing SPI models, as a mechanism to identify the strengths and weaknesses in software development projects and this information is used for improvement purposes (Humphrey et al. 1991). The Software Engineering Institute (SEI) has developed a method called The Standard CMMI® Appraisal Method for Process Improvement

(SCAMPI[SM]) (2006) to support the CMM assessment procedures in organizations. SCAMPI is a class A appraisal method, which includes detailed level instructions as well as steps and activities on how to implement the full assessment through all the maturity levels against the selected assessment model. In class A assessments, assessors typically use a large amount of evidences which is often in documented form. In depth analysis of the documentation, however, takes a great deal of time and effort by the assessment team, which, in addition needs to be highly educated about the criteria of existing standards or models (e.g. CMMI (2006) or ISO 15504 (2006)). Ratings are generated based on the descriptions of the standards. However, assessments (e.g. SCAMPI assessments) can be done in class B or C which means that the official ratings are not necessarily generated; assessments demands less resources and the amount of data e.g. documented evidence is used less than in class A assessment. For example, in a class B assessment, the same amount of documented evidence is still used but the assessment team trust face to face discussions more, such as data collected in interviews and workshops with actors in the system development. Because of the added trust, assessments in level B or C do not demand as much resources as a class A assessment but can still provide valuable results for both the teams and the organization. Although ratings are not generated in class B and C assessments, the assessment goal is more to define the strengths and improvement needs for the current organization or teams and therefore to support the process improvement rather than to rate the capability levels of the organization.

Lightweight assessment methods have been developed in order to offer techniques for implementing assessments rapidly based on the needs of small companies that have high dependencies on a low number of individuals and projects. Lightweight assessments typically follow the class C-type assessments with characteristics such as low costs, focused processes, simple assessment process and modified use of assessment process as described in Table 1.

[SM] SCAMPI, CMMI and CMM Integration are service marks of Carnegie Mellon University.

Table 1. Characteristics for lightweight assessments.

Criteria	**References**
Low costs	(Richardson 2001)
Focused processes	(Richardson 2001, Wilkie and McCaffery 2005)
Simple assessment process	(Horvat et al. 2000, Kautz 1998)
Modified use of assessment models	(Batista and Figueiredo 2000, Kautz 1998)

Although small companies have a need for the highest software product quality and fast software production, they often have problems in investing in the assessments (Batista and Figueiredo 2000). Thus, the key idea of these lightweight assessment approaches is that they focus on the most valuable process areas such as requirements management, project planning and tracking (Richardson 2001, Wilkie and McCaffery 2005) and keep the assessment process as simple as possible (Horvat et al. 2000, Kautz 1998) to avoid high costs and effort used in the SPI (Richardson 2001).

Anacleto et al. (2004) identify some existing criteria used in the proposed lightweight assessment methods. Based on their analysis they suggest 9 additional criteria for lightweight assessments which are: (1) a detailed description of the assessment process, (2) guidance for process selection, (3) a detailed definition of the assessment model, (4) support for identification of risks and improvement suggestions, (5) conformity with ISO/IEC 15504, (6) no specific software engineering knowledge required from companies' representatives, (7) tool support is provided, (8) support is provided for high-level process modeling and (9) there is public access to the developed method.

2.1.2 CMM / CMMI

The Capability Maturity Model® CMM is a model which is often used as reference model in assessments to facilitate the organization to achieve a level where continuous, optimized improvement of the software development is possible (Anderson 2005). CMM, as well the numerous other IEEE standards and guidelines, integrates some of the wisdom in the software development industry (Bamberger 1997). It has been developed by the SEI since 1986, based on the concepts developed by Humhprey (1990) and Deming (1990).

The term "discipline" is used in process relations but also in agile literature (e.g. Boehm and Turner 2003a) to describe the knowledge that is needed or available when selecting models (e.g. CMMI (2006)) or the knowledge which is integrated into the framework (Paulk 1999). CMM describes the principles and practices underlying the software process maturity (Paulk 1999). It is a model that can be used in assessment to reflect processes or software organization as a purpose to identify the strengths and improvement needs and, therefore, facilitate organizations to shift gear in the software development from ad hoc to mature, disciplined processes (Paulk 1999). CMM/CMMI were developed due to the increasing need to integrate existing models as a reference model that could be even more efficiently used in continuous process improvement in software projects and organizations. Paulk (1999) reports that due to the small differences; the topics of CMM (2006) and SPICE (2006) correlate highly to each other. CMMI is the integration of several models including elements of both CMM and SPICE. The key differences between CMM and CMMI are:

- the measurement and analysis process is added in the maturity level 2
- there is more focus on software and product development, its risk management, verification and validation instead of the organizational level processes
- the Organizational Innovation and Deployment process area has been included in maturity level 5 instead of the change management process areas.

CMMI includes both capability and maturity models, which means that it can be used in a staged and continuous way. The staged representation focuses on a set of key process areas, which are exclusively identified within the maturity levels (1–5) (CMMI 2006). The assumption of the staged representation of the CMMI is that an organization cannot achieve the next maturity levels before achieving the previous level first. Thus, an organization that succeeds to fulfils its goals in the process areas of level 1 is rated at the lowest level, called Level 1 (Initial). Other levels, repeatable (level 2), defined (level 3), managed (level 4) and optimized (level 5) can be achieved by fulfilling the goals of the process areas of each level in Table 2 (CMMI 2006). In the continuous representation, processes are often measured using the same scale of capability levels that are incomplete, performed, managed, defined, quantitatively managed and optimizing (CMMI 2006). CMMI includes 25 key process areas (CMMI 2006). Each process area contains specific and generic goals that are again dealt with by specific and generic practices (CMMI 2006).

Table 2. Process areas and purposes (CMMI 2006).

Process Area	**Specific Goal**
Maturity Level 2: Managed	
Requirements Management	Requirements of the projects, product or product components are managed, and the consistencies are identified between those requirements, project plan and work products
Project Planning	Establish and maintain plans that define project activities
Project Monitoring and Control	Provide understanding of project progress, so that corrective actions are taken
Supplier Agreement Management	Manage the acquisition of products from suppliers
Measurement and Analysis	Develop a measurement capability that is used to support management
Software process quality assurance	Provide management with appropriate visibility into the product and the software process
Software configuration management	Establish and maintain the integrity of software products throughout the project's software life cycle
Maturity Level 3: Defined	
Requirements Development	Produce and analyse customer, product and product component requirements
Technical Solution	Design, develop and implement solutions to requirements
Product Integration	Assemble products from the product components to ensure that the product is integrated properly
Verification	Ensure that specific work products meet their specific requirements
Validation	Ensure that product or product components fulfil the intended use
Organization process focus	Plan and implement organization level process improvement, based on an understanding of strengths and weaknesses
Organization process definition	Develop and maintain a usable set of software process assets

Training program	Develop individuals' skills and knowledge, so that they can perform their roles effectively
Integrated project management	Integrate management of the project and involvement of relevant stakeholders according to the process which is tailored from the organization of a standard set of process
Risk Management	Identify potential problems before they occur
Decision, Analysis and Resolution	Analyse possible decisions using a formal evaluation process
Maturity Level 4: Quantitatively Managed	
Quantitative process management	Quantitatively control the performance of the software project's process.
Quantities project management	Quantify manage the project defined process to achieve the projects set quality and process performance objectives
Level 5: Optimizing	
Organizational Innovation and Deployment	Select and deploy incremental innovative improvements
Causal Analysis and Resolution	Identify causes and defects and other problems and take actions to prevent them in the future

The achievement of related goals, maturity levels and CMMI specific practices can be assessed through practices used in the case organization. However, the main goal of the appraisal is to find out whether the goals are achieved or not, rather than whether or not the defined items exist as such (CMMI 2006). Thus, these items can be characterized as tools for the evaluators, when appraising the achievement of the goals.

Each process area includes 1-N Sub Practices which help to explain the content of the goal of the particular process area. Table 3 describes the specific goals and sub-practices for the selected requirements management, project planning and project monitoring and controlling process areas, which have been argued to be the most important processes especially in small and medium sized enterprises (Galin and Avrahami 2006).

Table 3. Sub-practices for requirements management, project planning, monitoring and controlling.

Process Area	Specific Goals (SG) and Sub-Practices (SP)
Requirements Management	SG 1.1 Obtain an Understanding of Requirements SP 1.2 Obtain Commitment to Requirements SP 1.3 Manage Requirement Changes SP 1.4 Maintain Bi-directional Traceability of Requirements SP 1.5 Identify Inconsistencies between Project Work and Requirements
Project Planning	SG 1 Establish Estimates SP 1.1 Estimate the Scope of the Project SP 1.2 Establish Estimates of Work Product and Task Attributes SP 1.3 Define Project Life Cycle SP 1.4 Determine Estimates of Effort and Costs SG 2 Develop a Project Plan SP 2.1 Establish the Budget and Schedule SP 2.2 Identify Project Risks SP 2.3 Plan for Data Management SP 2.4 Plan for Project Resources SP 2.5 Plan for Required Knowledge and Skills SP 2.6 Plan Stakeholder Involvement SP 2.7 Establish the Project Plan SG 3 Obtain Commitment to the Plan [PA163.IG103] SP 3.1 Review Plans that Affect the Project SP 3.2 Reconcile Work and Resource Levels SP 3.3 Obtain Plan Commitment
Project Monitoring and Controlling	SP 1.1 Monitor Project Planning Parameters SP 1.2 Monitor Commitments SP 1.3 Monitor Project Risks SP 1.4 Monitor Data Management SP 1.5 Monitor Stakeholder Involvement SP 1.6 Conduct Progress Reviews SP 1.7 Conduct Milestone Reviews

2.1.3 Empirical Findings

While CMMI programs have been widely reported as a beneficial approach to improve productivity and the time of the system development life cycle (Galin and Avrahami 2006), many companies have also opposed reference models like CMM and CMMI, for the following key reasons:

a) CMMI assessments are considered too heavy and time consuming (Fayad and Laitinen 1997);
b) there is no evidence available that the order of the process acquisition driven by CMMI capability levels is right (Fayad and Laitinen 1997);
c) the implementation of improvements is too time consuming (Niazi, et al. 2003) and are not implemented as planned (Herbsleb et al. 1994);
d) performance improvements are made focusing too much on processes forgetting people aspects (Laitinen and Fayad, 1998);
e) reference models, such as CMMI, are so massive, that the achievement of CMMI levels can lead to the approach in which developers use more time writing documents than implementing software (Boehm and Turner 2003a).

Challenges in the CMMI Assessments

The assessments are claimed to be wasteful, because the current assessment methods often tend to be too ‘heavy’ and expensive (Fayad and Laitinen 1997). It has been reported that even 77% of process improvements take longer than expected (Herbsleb et al. 1994). There are many reasons why the assessment costs have risen too high. For example, the organizations do not often ignore the process areas of higher levels before they have achieved the goals of the lower level (Dangle et al. 2005). This can easily delay the company’s achievement of the progress in other levels (Dangle et al. 2005).

As a response to the critique of heavy and time consuming assessments, tailored ‘lightweight’ assessment techniques have been provided in several studies (Batista and Figueiredo 2000, Horvat et al. 2000, Kautz 1998, Richardson 2001, Wilkie and McCaffery 2005). They have been developed in order to offer techniques for implementing assessments rapidly, but only in small companies that have high dependencies on a low number of individuals and projects. CMM is often used in the context of lightweight assessments (Batista and Figueiredo

2000, Horvat, et al. 2000, Wilkie and McCaffery 2005). For example, Batista and Figueiredo (2000) argue that a more pragmatic application of CMM and a simplification of the assessment methods are the key success factors for the assessments in small teams. The pragmatic application of CMMI in both class B, C assessments and in a lightweight assessment approach means that the improvement initiatives are defined based on the improvements that are most relevant for the assessed organization.

Although, the lightweight assessment approaches seem to offer attractive solutions for the problems of too time-consuming assessments, they have only been used in a small part of the whole SPI literature. At the moment, the lightweight assessment methods only focus on the needs of small organizations. The current literature lacks studies in which lightweight assessments can be integrated in the context of medium or large agile based software development companies or in the improvement of software development mediated with agile practices.

There is no evidence that the order in which the CMMI levels are driving the process acquisition is right (e.g. that it would be logical to achieve the CMMI level 1 and 2 process areas before the achievement of the CMM level 3–5 process areas) (Fayad and Laitinen 1997). For example, there is little empirical evidence that the engineering processes – such as requirements development, technical solution and product integration – should be improved later than the CMMI level 2 requirements management, project planning and controlling process areas (Fayad and Laitinen 1997).

Challenges in the CMMI Based Improvement Programs

The CMMI based improvement programs seems to demand a great deal of resources (Hareton et al. 2001). For example, the case study of 56 software organizations, that have conducted a CMM-based process improvement initiative, illustrates that the exploitation of the improvements is difficult and needs both a strong management support and staff involvement (Stelzer and Mellis 1998). Secondly, it has been argued that in many cases it takes a long time and significant effort for organizations to show the benefits of the CMMI programs (Niazi et al. 2003). For example, a survey of 138 individuals in 56 software organizations shows that 72% of the SPI programs that successfully applied the CMM based identification of weaknesses, are not actually improved (Herbsleb et al. 1994). Thus, SPI programs have often been regarded as a time consuming and long term approach whose benefits take a long time to be realized (Niazi et al. 2003).

The reason for why the CMMI initiatives take so much time to be implemented might lie in the fact that the processes often produce an environmental change which means a shift in the whole process hierarchy to achieve the identified improvements (Laitinen and Fayad 1998). This demands not only SPI team involvement but also efficient coordination and involvement of the developers.

In many cases, however, people do not have much time for software process improvement among their other software development duties (Laitinen and Fayad 1998). The process improvement is seen more as a "problem to be fixed" (Laitinen and Fayd 1998) than as an activity that really makes software development more productive or efficient. This has, in many cases, made the developers, especially, a little wary of the processes generally (Anderson 2005). Often, processes get in the way of the developers and slow the pace of software development to a frustrating level (Anderson 2005).

The reason for this might lay in the wrong focus of improvement programs as suggested by Laitinen and Fayad (1998). Although the assessments can involve the relevant people, the applied improvement programs have often focused too much on the process aspects at the expense of the people behind the actual development work (Laitinen and Fayad 1998). After all, the processes and people are interdependent and the processes are always created and used by the individuals in the development teams (Laitinen and Fayad 1998).

Impacts of the CMMI Based Improvement Programs

In some cases, the CMMI model is also perceived as too 'heavy' and too bureaucratic (Nawrocki et al. 2002). For example, Anderson (2005) points out that CMMI suggests that as many as 400 document types and 1000 artefacts are required to facilitate an appraisal. This is the main reason, why many of the CMM adopters have argued that the use of CMM itself actually increases the costs in the companies (Boehm and Turner 2003a).

2.1.4 Summary

Plan-driven software development methods have developed over time from the 'waterfall' model towards the 'spiral' model and finally to iterative, incremental software development. SPI mechanisms were developed because projects still ran over budget and time. Assessments are the starting point for SPI, but also the way to evaluate the performance of the project. In assessments, the goal is to define the strengths and weaknesses in the software development. Although CMM or CMMI are the reference models, which are most popularly and quite beneficially used in assessments, it seems that CMMI based improvement programs can easily turn out to be too heavyweight an improvement assessment, and an improvement approach that takes too much time from software development teams and companies. Sometimes, this may even drive the teams to use a too document driven software development process and cause frustration among the developers.

2.2 Agile Software Development

In spite of the ongoing SPI projects and published incremental development approaches and models in the 1980 and 1990s, software and product development projects were still troubled by costs and time overruns, low customer satisfaction and disappointed developers (Anderson 2003). According to Larman and Basili (2003), the Standish Group analysed 23 000 projects to determine the failure factors and concluded as follows:

> *"The top reasons for a project failure, according to the report, were associated with waterfall practices."*

The second reason for the failure has been the complexity of the software development projects:

> "*Anything can be complex, when complex things interact, the level of complexity goes to the roof*" (Schwaber and Beedle 2002).

Thus, organizations with standard based, highly documented, 'complex' processes were not able to respond to the challenges of the continuously changing requirements of customers, end users and business people. According to Beck (2000), the software projects faced the following problems:

a) schedule slips, the software is not ready when the deadline comes;
b) project cancelled, projects are cancelled after a long period without ever going into production;
c) systems go sour, the defect rate increases after the system has been put into production;
d) defect rate, the defect rate of the software product is so high that it is never used;
e) business misunderstood, the software never solves the business problem for which it was originally posed;
f) business changes, the software is ready for production but the business problem, for which it was meant was solved six months ago;
g) false feature rich, the software has many features which are fun to program but which do not have any added value from a customer perspective.

It is unlikely that these problems would be a consequence only of the use of plan-driven 'waterfall' development practices, but rather are a result of dynamic markets and increasing global competition which also cause additional turbulence and more changes on the products under development. Another reason for the issues has been the imperfect communication in the software development teams and in the organization as described by Cockburn (2002).

In agile software development the problem of the rigidity of a plan-driven software development process is solved with a 'planning driven' approach in which the planning has been made constant, frequent and fluid to enable the team respond to changes quickly (Wang et al. 2008). Parallel with the frequent planning the agile approaches also bring people regularly together for face to face communication. This should improve the software development if we understand software development as Cockburn (2002) states:

> "*Software development as a group game in which the team should cooperate together to achieve the defined goals*" (Cockburn 2002).

The game is co-operative because all the team members are able to help each other reach the set goals. In this game, the important element is communication. The success of the game depends a great deal on whether the communication is effective between the team members.

2.2.1 Agile Principles

On February 2001, 17 leading developers and proponents met at a workshop to think of some reasons and solutions for the ongoing crisis related to the software development process. They found that there is a need for new methods to respond to changes in software development projects and make communication in software development teams and organizations more efficient. As a result of this meeting, a manifesto for agile based software development was produced. It provides advice on how to focus the development on people, working software, customer collaboration and increase an organization's ability to respond to changes (Salo and Abrahamsson 2007).

The results of the meeting were 12 agile principles, which describe the values of the agile approach in more detail giving them a more concrete meaning. The principles of agile software development are as follows (Agile Manifesto 2001):

a) satisfy the customer through the early and continuous delivery of valuable software
b) sustainable development is promoted, facilitating an indefinite development;
c) simplicity is essential
d) welcome changing requirements, even late in the development
e) deliver working software frequently
f) working software is the primary measure of progress
g) continuous attention to technical excellence
h) business people and developers must work together daily
i) face-to-face communication is the best method of conveying information
j) the team regularly reflects on how to become more productive and efficient
k) the best work emerges from self-organising teams
l) projects built around motivated individuals.

Since the workshop, “agile thinking” through these principles has become more popular among software development teams and organizations. The principles of agile software development can be defined as fundamental ideologies that should be included in the practices of any software development method that claims to be ‘agile’ (Abrahamsson et al. 2002).

2.2.2 Agile Methods and Practices

Although the initial ideas of agile software development have been created and used already in the 1970s and 1980s, the agile methods emerged in the late 1990s and the early 2000s. Since then, they have been introduced in companies as significant mechanisms to increase the organization’s capability to respond to changes (Abrahamsson 2002). Often, an agile method is defined as a ‘just enough’ method that seeks to avoid predescribed and time consuming processes that add only little value to the software development (Cockburn and Highsmith 2001).

‘Methods’ are supposed to change and ideally improve practices in system development (Fitzgerald et al. 2002). There are several definitions of “methodology” and “methods” in the current literature (e.g. see Brinkkemper 1996, Fitzgerald et al. 2002). Based on some sources, the terms method and methodology have been used synonymously (Fitzgerald et al. 2002), whereas some research reports define methodology as “the study of systematic methods”. A method can be characterized as:

> *“A predefined and organized collection of techniques and a set of rules which state by whom and in what order the techniques are used” (Tolvanen 1998).*

Agile methods offer approaches to improve the software development process (Karlström and Runeson 2006). A number of methods are included in this family, the most notable being XP (Beck 2000), Scrum (Schwaber and Beedle 2002), Crystal (Cockburn and Highsmith 2001), Agile Modelling (Ambler 2002), APM (Highsmith 2004), FDD (Coad and Palmer 2002), and LSD (Poppendieck 2001). These developed because too much discipline kills initiative and the flexibility of software development projects, which are necessary for dynamic market environments. During the 2000s, interest in agile methods has increased dramatically (Lindvall et al. 2004). These methods have been adopted in different types of software projects and in wide-ranging application domains (Karlström and Runeson 2006).

One of the main reasons for the use of agile practices has been their ability to increase flexibility and the organization's ability to respond to changes. Another significant reason for short cycles, as described in the agile approach, lies in the communication aspect, especially, in more efficient face-to-face conversation (Larman 2003). Time boxed iterations and an incremental development approach is one of the foundations in all agile processes (Larman 2003). The fundamental idea of all agile methods is to deliver software as early as possible, in short cycles to get regular feedback both from customers and the management of the organization (Boehm and Turner 2003a). In general, agile methods tend to be lightweight processes for the software development. They emphasize user involvement and requirements prioritization and verification relying on tacit knowledge and communication as opposed to documentation (Boehm and Turner 2003a). Highsmith (2002a) states, however, that there has to be a balance between the documentation, working software and the collaboration in the agile software development teams.

XP is an agile method originally presented by Kent Beck (2000). It is a 'lightweight' methodology with four key values: communication, simplicity, feedback and courage (Beck 1999). From the communication perspective, XP faciliates correct communication, which is needed to employ the defined XP practices (Beck 1999). Simplicity in XP means the team's goal of implementing software remains as simple as possible. The value of simplicity is also connected to communication. If the code is simple, it is also easier to communicate to other people (Beck 2000). The third value, feedback in XP, mainly relates to customer collaborations. It means that the team should receive concrete feedback on their work on a daily, weekly or monthly basis. The value "feedback" also has a strong relation to communication. For example, Beck argues: "*the more feedback you have, the easier it is to communicate*" The last value is courage which means the "courage" to tackle new technical challenges and, thus, to make new innovations. According to Beck (1999), communication facilitates courage in teams because it opens the opportunity for new technical experiments.

Scrum has been pioneered by Schwaber and Beedle (2002). It is a simple process mainly focused on project management of software development (Fitzgerald et al. 2006). Scrum was originally influenced by Boehm's 'spiral' model, but it was developed based on industrial experiences to simplify the complexity of the project and requirements management in software organizations (Schwaber 2003). Scrum describes practices on an iterative, incremental time boxed process skeleton. At the beginning of the iteration, the team has a sprint planning meeting in which they decide what the team will do during the following iteration. At the end of the iteration, the team presents the results to all the stakeholders in the sprint review meetings to gain feedback on their work. The heart of Scrum is an iteration in which the self-organizing team builds software based on the goals and plans defined in the sprint planning meeting (Schwaber 2003). The team also has a daily 15 minute meeting called the daily Scrum, in which they check the status of the project and plan the activities of the next day (Schwaber 2003).

Both XP and Scrum define practices for the software development process. Beck (1999) identifies 12 key practices for the software development process, which mostly focus on software engineering. Beck (1999) argues that the XP practices are situation dependent, which means that the application of the practices is a choice which can be made based on the current development context.

Table 4 lists the practices associated with the XP and Scrum, the two methods that are the focus of this study. However, while such literature is often very detailed and prescriptive, there is often a substantial difference between the 'textbook' version of the method and the method actually enacted in practice. The selection of these agile practices are based on the Fitzgerad et al. (2006) description of the combined, customized use of Scrum and XP practices which enables the use of both XP and Scrum practices in the same project, so that sprints, small iterations, sprint planning and planning games are used as equal practices. Table 4. A Combined List of XP and Scrum Practices based on Fitzgerald et al. (2006).

Table 4. A Combined List of XP and Scrum Practices based on Fitzgerald et al. (2006).

Scrum Practices	**XP practices**	**"Text book" description**
Scrum sprints	Small releases	Put a simple system into production quickly, and then release new versions on a short cycle.
Sprint planning	Planning game	A quick determination of the scope of the next software release within a short time period, based on a combination of business priorities and technical estimates.
Daily meeting		Short daily status meeting which is official part of Scrum, XP suggests conducting it while standing up.
Post Game Sessions		At the end of each short development cycle, review the strengths and problems of the process in this cycle.
	40-hour week	Work time is limited to 40 hours per week as a rule.
	On-site customer	Include an actual user on the team, available full-time to answer questions.
	Pair programming	The code is written by two programmers on the same machine.
	TDD	Writing test code before writing the function code.
	Continuous integration	Integrate and build the system every time a task is completed – this may be many times per day.
	Collective ownership	Anyone can change any code anywhere in the system at any time.
	Simple Design	The design of the system should be as simple as possible.
	Refactoring	Programmers restructure the system, without removing its functionality.
	Coding standards	Adherence to coding rules which will facilitate communication through the code.
	Metaphor	Guide all the development with a simple shared story of how the whole system works.

2.2.3 Empirical Findings

While the empirical research on agile practices in use is growing rapidly, there are still many issues with a strong theoretical and conceptual foundation and a systematic focus on the specific problems or challenges in the software development process where agile practices are in use. This chapter gives a brief description of the empirical body of knowledge related to the use of XP and Scrum practices. Dybå and Dingsøyr (2008) carried out a systematic review of the studies of agile based software development, during the years 2000–2005. According to them, the introductions and overviews for agile software development were reported in several studies. For example, the report of Abrahamsson et al. (2002) describes the practices of 10 agile methods including XP and Scrum. Cohen et al. (2004) presents the introduction to software development where agile practices are discussed in relation to agile software development process and CMM/CMMI. Erickson and Lyytinen (2005) evaluate the research related to XP, Agile software development and agile modelling.

Use of the XP and Scrum Practices – General Reports

EXtreme Programming (XP) and Scrum have been suggested as the most used agile methods (Fitzgerald et al. 2006). XP practices have mostly been introduced among other agile methods and practices. According to Dybå and Dingsøyr (2008), more than 26 empirical reports can be found focused on the use of the XP or Scrum methods in general. 76% of the reports related to use of the XP and only 3% to Scrum practices. Due to the rapid increase of empirical research of agile practices in use, there are already a few reports on the use of Scrum published after the systematic review of Dybå and Dingsøyr (2008), which reviewed the reports related to XP and Scrum between the years 2000–2005. (E.g. the reports of Moore et al. 2007, Sutherland et al. 2007).

Although, there are many reports about the success stories of the integrated use of Scrum or XP practices, only some research and experience reports provide a critical viewpoint of the adoption of agile software development methods. For some examples of empirical studies on XP in use (Drobka et al. 2004, Grenning 2001, Karlström and Runeson 2006, Layman et al. 2006b, Rasmusson 2003); for those on Scrum in use, (Mann and Maurer 2005, Rising and Janoff 2000, Schatz and Abdelshafi 2005) and a study of the combined use of XP and Scrum (Fitzgerald et al. 2006).

Some Examples of the Use of XP Practices

Grenning (2001) describes how XP practices such as the planning games, small releases, simple design, test first development; refactoring, pair programming, collective ownership, continuous integration and 40 hour week are used in a large company developing safety critical systems. As a result of the analysis, he suggests that some XP practices, such as simple designs integrated with test first development and refactoring work quite well in the safety critical area. In that case, the managers were reported to be happy with the results of the use of XP practices. This was mainly due to the XP team ability to readily produce working software instead of a high amount of documentation (Grenning 2001). One of the biggest challenges revealed from this case was the resistance mainly due to the decreased documentation. The documentation was needed, for example, to define product requirements, sustain technical reviews, support maintenance and describe interfaces. Based on the experiences of the XP project it was still understood that the documentation was needed at least for maintenance and review purposes. One reason for this was that the pair programming as a practice was not considered efficient enough to abandon the design review processes (Grenning 2001).

Rasmusson (2003) reports XP method use experiences in ThoughtWorks excluding metaphor and on-site customer practices. He suggests that unit testing, refactoring, test-first design, and simplicity are beneficial XP practices for software development teams because they bring some improvements related to team communication. The problems in the project related to Test Driven Development, refactoring and communication with external stakeholders. TDD took a lot of time because the developers had to first learn philosophy behind TDD and to acquire enough skills for its implementation. Refactoring was used as an excuse for the delays in the project work schedule which made the customer unhappy about the practice itself. Moreover, communication was difficult with the external stakeholders such as getting database system administrators to participate in daily stand up meetings (Rasmusson 2003).

In Motorola, a selected set of XP practices was used also in the field of safety critical systems (Drobka, et al. 2004). In that case, the use of XP practices was reported to have 53% improved average quality compared to the plan-driven software development project. A key challenge in these XP projects was to define how the project changes affect the overall end date of the project. As a problem Drobka et al. (2004) mention also the issues of documentation. As in the case of Grenning (2001), the source code was not documented enough for the whole system so that a high-level architecture document was still needed to use to provide class diagrams, scenarios and a process view of the system for developers. Furthermore, acceptance tests were not enough to verify the traceability from product to customer requirements. Therefore, additional verification review meetings were used to cover this gap in the verification process (Drobka et al. 2004).

Layman et al. (2006b) report the results of a case study to understand communication in globally distributed agile software development. As communication mechanisms, instead of iterative planning meetings and daily stand up meetings the distributed teams used, for example, instant messaging systems, informal emailing and additional tool support for project tracking (Layman et al. 2006b). Based on the case study it is clear that the customer role is important for the requirements management activities and can be problematic especially when working with software at a distance (Layman et al. 2006b). One reason for the communication problems were the difficulties to communicate across the different time zones (Layman et al. 2006b).

Karlström and Runeson (2006) report experiences of the use of XP practices as a part of a milestone driven software development process. Although the use of agile practices has been defined as mainly beneficial for the teams, the report contains an example that shows that communication in the agile based software development teams can sometimes also lead to more isolated development teams which could also have negative impacts on communication on the organization level (Karlström and Runeson 2006).

Sfetsos et al. (2006) have studied the advantages and disadvantages for 30 Greek software companies to adopt XP practices. While the key success factors were the efficient communication and synergy between the persons in the software development teams using XP practices (Sfetsos et al. 2006), the study shows also that the companies can face several challenges related to the use of XP practices. One of the main challenges indicated in the study is the cultural problem which was emphasized especially between the agile and traditional teams in large distributed projects. The cultural differences had an effect mostly on the pair programming practice which required choosing the right people and to continuously rotate developer pairs. Another XP practice which was difficult to use in practice was the on-site customer. In fact, the Greek software companies ended using the telephone, fixed appointments and internet communication to handle customer communication instead of face to face meetings as suggested by the Agile Manifesto (2001).

Some Examples of the Use of Scrum Practices

Rising and Janof (2000) report on a text book use of the Scrum method in three small software development teams. As a result of the use of the sprint planning, sprints and Scrum meetings, they claim that the Scrum method facilitates the product manageability, visibility and team communication as well as ensuring frequent feedback from the customer.

Mann and Maurer (2005) report on the Scrum impacts on customer satisfaction and overtime work in the teams. Based on the empirical analysis of the case study in which Scrum was used in a development project, they reveal that although it is sometimes difficult to follow sprints of 30 days, hold daily Scrum meetings as a Scrum practice, facilitating customers to keep up to date with the development work and planning meetings helps to reduce confusion about what should be developed from the customer perspective. Additionally the study reports a statistically significant reduction in overtime work among the evaluated software development teams due to the adoption of agile practices.

Because the literature on the use of Scrum practices is relatively new, the current literature seems to lack data on the challenges related to the use of the Scrum method or its practices.

Some Examples of the Use of Individual Agile Practices of XP and Scrum

The use of individual XP practices, such as pair programming, the TDD, continuous integration or iteration retrospectives has been described in several individual reports that focus only on the use of one agile practice at the time. These reports on the use of individual XP practices can be found for example related to pair programming and TDD. Hulkko and Abrahamsson (2005) analyse 19 previously published individual research reports on the use of the pair programming practice and then compare the results between these multiple case studies. They find that pair programming is the most useful for complex tasks and facilitates the development of a more readable code as described in the literature. They note, however, that pair programming does not necessarily lead to improved productivity nor decrease the number of defects in the developed code.

Siniaalto and Abrahamsson (2007) reviewed the empirical body of evidence on existing reports related to TDD. They found 16 empirical reports on TDD studies. Based on the empirical analysis of the TDD practice in use, they conclude that TDD may improve the quality of developed software and facilitate developers but does not necessarily have an effect on the code quality.

Karlsson et al. (2000) have reported experiences about the use of the daily build activity in Ericsson as a part of continuous integration practice. According to their experiences, they have found that a daily build is actually difficult to apply in distributed software development environments.

Koskela and Abrahamsson (2005) have made a study of the on-site customer practice in agile software development concluding that actually 21% of the customer time is needed to assist the team in the on-site customer situation. On the other hand, it seems that the on-site customer is a good way of guaranteeing the strong commitment of the team to their work (Koskela and Abrahamsson 2004). Korkala et al. (2006) continue to research customer communication in an agile context focusing on the link between software quality and customer location. They show that the defect rate in the teams actually increases when the customer involvement decreases.

In agile development, regular retrospectives have often been defined as a primary mechanisms for incremental process improvement at project level (Packlick 2007). As one of the key approaches for SPI in the agile software development context, Salo and Abrahamsson (2007) present a method called post-iteration workshop (PIW) for iterative project level software process improvement in an agile context. In this method the goal is to harvest and implement improvements in an iterative manner from and in the agile software development teams. The process of the PIW method consists of two main phases, first a workshop after each development iteration in which the strengths and improvement needs of software development team are harvested and analysed and second, a phase in which the improvements need to be evaluated at organizational level and then adopted in the software development team (Salo and Abrahamsson 2007). Although there is some evidence available that the PIW method provides a valuable approach on how to improve and adopt agile practices in an iterative manner in software development teams (Salo and Abrahamsson 2007), PIW does not identify detailed mechanisms to facilitate agile practice identification for a plan driven situation or evaluation of the agile software development projects when “identifying” or “checking” the improvements between the software development projects or at an organizational level. While retrospective is one agile practice and defined as a significant improvement mechanism in agile software development, it has been argued that reflections alone are not enough to drive the degree of change needed for efficient agile practice adoption or improvement (Packlick 2007).

Combined Use of XP and Scrum

Fitzgerald et al. (2006) describe how XP and the Scrum practices can be used in a combined, customized way in software development projects, claiming that any software development method cannot be used in software development projects without tailoring. To support this argument Fitzgerald et al. (2006) present a case study of Intel Shannon as an example of how the sprint, short iterations, sprint planning and planning game can be used as equalling practices in a same team to gain such benefits as 1) reductions in a code defect density and 2) improved planning, tracking and 3) communication. This approach is significant for research purporting to understand the use of both the XP and Scrum practices but is not, yet, covered in the current literature.

XP and Scrum are in many reports suggested to be practical solutions to improve the time to market and the speed of software development. This chapter summarized the Agile Manifesto (2001) and empirical analyses of the use of XP and Scrum practices.

As noted in Chapter 2, XP and Scrum practices have been commonly adopted and used in software organizations. Most of the reports relating to agile methods have, however, been made unsystematically without strong theoretical baselines or aims. Most of the experience reports suggest that the use of XP and Scrum needs tailoring depending on the context and domain in which they have been adopted or used. Furthermore, the need for documentation, for example, seems to vary depending on the projects and developed product.

Although, several empirical reports exist on the use of Scrum and XP practices, few studies have focused on a customized approach (i.e. combined set) and use of both XP and Scrum practices. There is a dearth of reports providing a critical analysis of the software development process in which Scrum has been used. However, there are several reports on the use of individual XP practices available in current literature. For example, a quick review reveals more than 20 reports of pair programming and 17 reports of TDD.

2.2.4 Summary

Both XP and Scrum define practices for the software development process. During the 2000s interest in agile methods has increased dramatically (Dybå and Dingsøyr 2008). These methods have been adopted in different types of the software projects and in wide-ranging application domains (Lindvall, et al. 2004). Briefly look to the empirical body of knowledge (in Chapter 2.2.3) reveals that there are several communication challenges occurring in the companies using agile practices:

Documentation vs. face to face communication

- Resistance mainly due to the decreased documentation (Grenning 2001)
- Source code not enough as documentation, a high-level architecture document still needed to use to provide class diagrams, scenarios and a process view of the system for developers. (Drobka, et al. 2004).

Communication at organizational level

- Communication difficulties with the external stakeholders; participation in daily stand up meetings (Rasmusson 2003)
- Negative impacts on communication on the organization level (Karlström and Runeson 2006).

Communication in distributed environments:

- Difficulties to communicate across the different time zones (Layman et al. 2006b).

2.3 Hybrid Approaches for Improvement of Software Development

Boehm and Turner (2003a) argue that:

> "*Both the agile and planned approaches have situation-dependent shortcomings that, if left unaddressed, can lead to project failures*". (Boehm and Turner 2003a)

Therefore it is important for organizations to find a balance between these two approaches (Boehm and Turner 2003a). The current literature, however, presents only a few so called hybrid software development approaches, which use elements of both the plan-driven and agile software development. In the following sections some hybrid approaches for improving software development using both agile and plan-driven approaches have been briefly presented and discussed.

2.3.1 Risk Based Agility Evaluation

One of the methods that could be characterized as a starting point for the agile practice adoption activities is the Boehm and Turner (2003b) method to address risks particularly associated with agile and plan-driven methods. The method consists of five key steps which are (1) risk analysis, (2) risk comparison, (3) architecture analysis, (4) a tailor life cycle and (5) execution and monitoring, as seen in Figure 2.

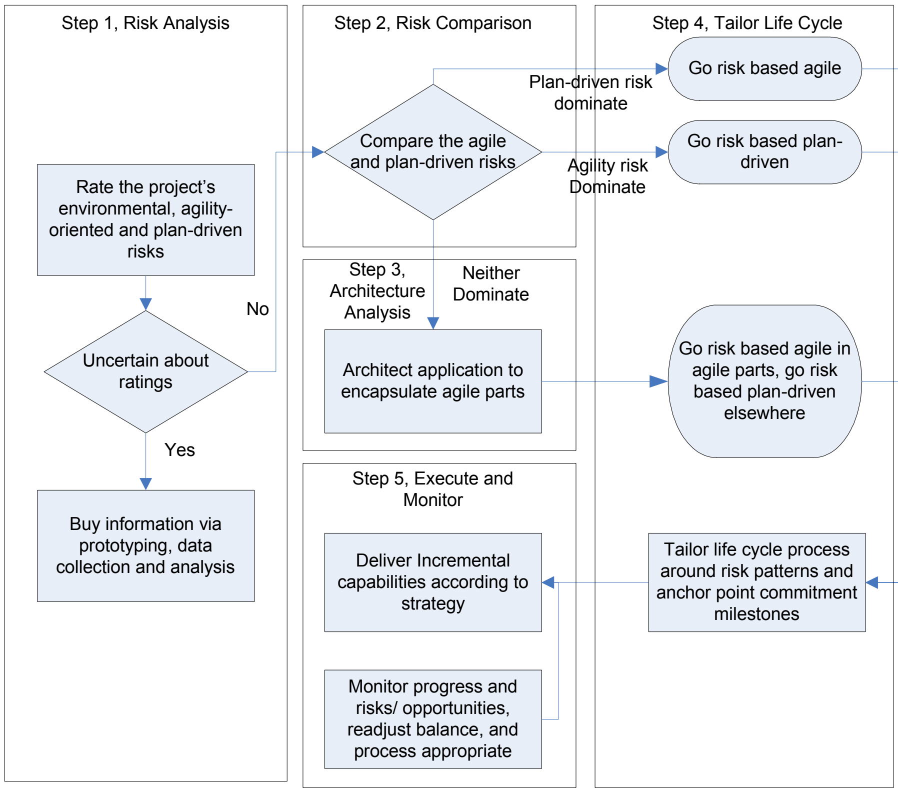

Step 1, Risk Analysis
Step 2, Risk Comparison
Step 4, Tailor Life Cycle
Go risk based agile
Plan-driven risk dominate
Compare the agile and plan-driven risks
Go risk based plan-driven
Agility risk Dominate
Rate the project's environmental, agility-oriented and plan-driven risks
No
Step 3, Architecture Analysis
Neither Dominate
Uncertain about ratings
Architect application to encapsulate agile parts
Go risk based agile in agile parts, go risk based plan-driven elsewhere
Yes
Step 5, Execute and Monitor
Buy information via prototyping, data collection and analysis
Deliver Incremental capabilities according to strategy
Tailor life cycle process around risk patterns and anchor point commitment milestones
Monitor progress and risks/ opportunities, readjust balance, and process appropriate

Figure 2. Risk based method for agility evaluation adopted by Boehm and Turner (2003a).

In the risk based method, the project environmental, agile and plan driven risks are first collected from the project, and then compared in order to define whether the project should go towards the plan-driven or agile software development process. The third phase in the method is to evaluate how the architecture affects the decision, before the life cycle process tailoring. The final step of the method is to execute and monitor the progress of the development process.

As a rating scale, Boehm and Turner (2003a) determine five agility factors affecting the selection of the agile or plan-driven way to do software development in Figure 3. These are: 1) size: the number of people in the team, 2) criticality: the product's safety criticality, 3) dynamism: the degree of requirements and change in technology 4) personnel: the skill and experience of the team and 5) culture: the support for agile software development provided by the organization culture (e.g. the developers' freedom to create technical solutions).

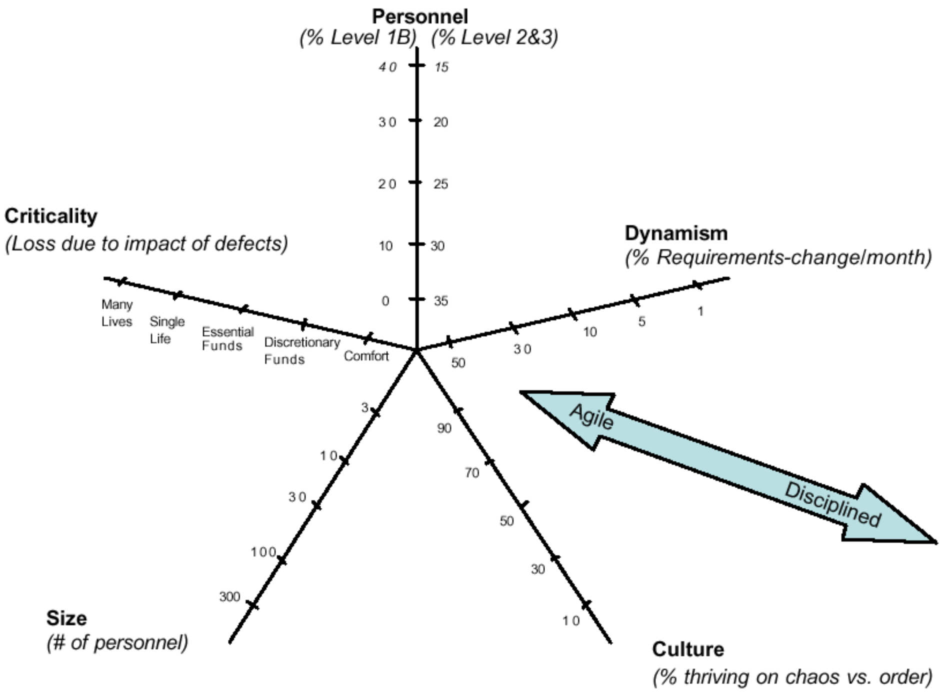


Figure 3. Boehm and Turner's five agility factors (2003a).

In Boehm and Turner's (2003a) risk based method for agility evaluation, each axis is labelled based on the respective agility dimension. When the data points of a project for each factor are joined and the resulting shape is located directly around the centre, it suggests the use of an agile method. Shapes that gather distinctly toward the periphery suggest using a plan-driven methodology. More varied shapes suggest the use of a hybrid method including both agile and plan-driven practices (Boehm and Turner 2003b).

The presented risk-based method provides comparable facts for agile based software project evaluation. While this can be used as a mechanism to start an agility evaluation and an agile pilot project selection, it does not address any specifics regarding the application of agile practices. The main challenge, for the organization, remains how to tailor the different agile practices (i.e. activities) to its specific product development context and how to apply the most suitable agile practices as part of the organization's current activities.

2.3.2 Levels for Agility Evaluation

Ahmed Sidky (2007) presents a framework for agile practice adoption in software development projects and in organizations which sets out the levels for agility evaluation. Similarly to Boehm and Turner (2005), he claims that the organization needs to evaluate the needs for agile practices before making the adoption decision. This enables an adoption approach in which the organization adopts only those agile practices that are within their current capacity (Sidky 2007). This is a suitable adoption approach, especially for organizations which are not willing to invest, time, money and effort to make changes or are unable to make changes in their current software development process (Sidky 2007). Sidky (2007) presents a framework for a project and an organization agility evaluation. The framework of Sidky (2007) deals with the agile practice adoption in four main steps, which are:

- ➢ identification of discontinuing factors: pre-assessment, in which the organization defines its capability to adopt agility
- ➢ project level assessment: identification of the highest level agility that the project can achieve
- ➢ organization readiness assessment: evaluation of the extent to which the organization is ready to support the project's target to achieve the agile level
- ➢ reconciliation: resolving of practices that the company wants and can adopt.

Agility is evaluated based on agile principles and five different agile levels, which are collaborative, evolutionary, effective, and adaptive and encompassing. This approach is also suggested for use in the diagnosis phase of the IDEAL model and suitable also for a situation in which organizations use CMMI based assessments.

Sidky (2007) suggest that an organization, moving towards agility, begins with the communication in agile level 1. This is because the agile practice advice focus on individuals and interactions over processes and tools and also because SPI literature leads companies to enhance communication and collaboration (e.g. (Sidky 2007)). Under the collaborative agility level, Sidky (2007) includes agile practices, such as coding standards, collaborative planning, collaborative teams and reflection and tune processes.

The second level in Sidky's framework is called evolutionary requirements. Its purpose is to ensure continuous delivery of the software. The argument behind the level description is based on the fact that most agile methods follow the incremental development process and aim towards a regular, evolutionary delivery of the software which needs to be achieved before the deployment of engineering practices. (Sidky 2007). The third agility level, in Sidky's model is called effective. It focuses on producing high quality software with practices such as self organizing teams, frequent face to face communication and continuous integration. The fourth level of the framework, "adaptive", aims at responding to changes with daily meetings, user stories and so called agile documentation. The final agile level 5 provides an encompassing agile environment with practices such as TDD and pair programming. (Sidky 2007).

2.3.3 Integrating CMMI and Agile Practices

According to Reifer (2003), agile practices should fit easily with CMM because the CMM represents a framework for self-improvement. Integrating plan-driven and planning driven (i.e. agile) software development has, however, often said to be a fundamental challenge which is "like oil and water" (Turner and Jain 2002). One reason for this 'thinking' might be perhaps misleading assumptions related to agile and CMM/ CMMI integration (Turner and Jain 2002).

It is often assumed that CMMI compliant processes need to be heavyweight, bureaucratic and slow-moving (Anderson 2005). Agile practices such as XP and Scrum have been said to offer a less bureaucratic way of developing quality software focusing on human centered processes. (Bos and Vriens 2004). However, the common belief has been that to follow CMMI the teams must involve documenting requirements, decisions, meetings risks, plans and effort spent on software development in order to develop high quality software. On the other hand, when an agile approach is used the teams can achieve quality software relying more on informal, lightweight documentation. (Boehm and Turner 2003a). One way to discuss this fundamental problem of CMMI and agile method integration is to compare the challenges of the CMMI to the principles of agile software development.

Challenges in CMMI Assessments vs. Iteration Retrospectives

Agile principles suggest that a team needs to reflect regularly on how to become more productive and more efficient (Agile Manifesto 2001). As previously mentioned, agile software development companies typically respond to this problem by adopting a practice of iteration retrospectives, in which the teams are continuously collecting and evaluating their strengths and improvement needs in a face to face way (Salo and Abrahamsson 2007). The problem with the iteration retrospective approach is that it does not enable mechanisms for sharing information between the teams or communicating the strengths and the improvement needs from teams to the organizational level (Salo and Abrahamsson 2007). Thus, the assessments can provide some answers to this gap by describing well established way to collect and share improvement information needed for organizational level SPI.

Heavy and time-consuming assessments may not, however, fit the organization which follows the principle of "working software is the primary measure of progress" (Agile Manifesto 2001). Thus, the assessments, even if useful for agile companies need to be implemented in the lightest way possible (i.e. not taking too much time of the teams and organizations). One solution to the problem of too heavy and time consuming assessments would be to follow the processes of lightweight assessment methods such as ADEPT (McCaffery et al. 2006, 2007, Wilkie and McCaffery 2005).

The argument that there is no evidence that the CMMI capability levels actually drives the improvements in the right order (Fayad and Laitinen 1997) might actually vitiate the use of capability levels in both traditional and agile contexts. There are, however, some reported experiences of the successful use of CMMI capability levels when implementing assessments in the context of agile software development (Glazer 2001, Kähkönen and Abrahamsson 2004, Paulk 2001). Based on these experiences it has been argued that at least some CMMI levels could be achieved using agile methods (Anderson 2005). It seems, however, that the continuous improvement, adapted as a part of the agile methods, could drive a team to achieve some of the specific practices from CMMI level 5 (e.g. continuous improvement of process performance through both incremental and innovative technological improvements) before the actual achievement of all the criteria of CMMI level 2 and 3. This is against the practice of using capability levels to assess processes in agile software development context. It should be remembered, however, that often, the goal of the organization is just to achieve business goals through the improvement of software development process. Not, actually, to have official CMMI certification (McCaffery et al. 2007).

The Challenge of CMMI Based Improvement Programs vs. Iterative Improvement

The adoption of the improvement initiatives of CMMI based assessment programs can be based on improvement models such as QIP (Basili 1989) and the IDEAL model (McFeeley 1996). It seems that there are some fundamental differences between the SPI programs that are conducted in the traditional software development teams and the SPI programs that are conducted in an agile context (Salo and Abrahamsson 2007). Firstly, CMMI based improvement programs are often based on strong management control whereas SPI in an agile context emphasizes the use of self organizing teams as the key for the SPI implementation (Salo and Abrahamsson 2007). Thus, the process of conducting SPI in agile software development is typically based on the team level improvement of their daily working practices (Salo and Abrahamsson 2007).

Impacts of the CMMI Based Improvement Programs in an Agile Context

Although it has been argued that CMMI assessments leads projects to a too document driven software development approach (Boehm and Turner 2003a) it should be highlighted that the CMMI model, although it feels heavy, does not explicitly require any particular work products or artefacts (Baker 2006). On the contrary it merely proposes collecting evidence that the goals of each process area are achieved. On the other hand, as defined in Chapter 2.2.3 agile software development does not mean a process in which documentation are not produced. On the contrary, documentation is still needed, but the level and amount of documentation seem to depend on the development environment and complexity of the developed system. Thus, there is the possibility for the organization to follow the CMMI goals and still use the agile practices in the software development (Baker 2006). Kähkonen and Abrahamsson (2004) demonstrate how to use CMMI as a framework for assessing an agile based software development process. Based on the analysis of a one case project, they suggest that assessment in the context of the agile software development is possible but requires more interpretation from assessors than the CMMI assessment for a plan-driven software development process.

2.3.4 Empirical Findings

The Boehm and Turner (2003b) risk based agility evaluation method provides a description of the risks or factors that may appear or make software development demanding in the context of agile software development. Whereas it provides a mechanism for starting the SPI by evaluating whether the project or organisation defines the risks for movement towards agility, it does not focus on identifying agile practices that could be used as a starting point in software process improvement or agility adoption. Furthermore, it remains unclear how to adopt the agility, how to assess the agility in the hybrid software projects or how to improve agile software development projects. Although Boehm and Turner (2003a) present empirical analysis based on the presented method from one case company, the risk based approach to balance agile and plan-driven methods has only been empirically evaluated in a few studies.

McCaffery et al. (2006, 2007) have integrated the risk based agility evaluation in the so called 'ADEPT' assessment method, which is developed based on the needs of small software enterprises. In the 'ADEPT' method based assessment, McCaffery et al. (2007) use the Boehm and Turner's (2003b) risk based analysis as part of an assessment to be able to evaluate an organization's capability to adopt agility as part of the improvements. Although, 'ADEPT' seems to be the only assessment method which integrates agility and SPI, it lacks a description on how agile practices should be used when doing a lightweight assessment for software organizations.

Sidky's framework (2007) opens an interesting angle on agile practice adoption. The results of the framework were evaluated from a survey which was sent to 35 participants and completed by 12. It can be argued that the framework is not yet well established from the empirical viewpoint. It does not give details on three important questions: 1) how to map agile practices with CMMI, 2) how to decrease the effort in assessments in agile practice adoption situations or 3) how to validate the impact of improvements in the context of agile software development. Furthermore, Sidky's (2007) framework could also be challenged with the same argument about the use of CMMI capability levels in the assessment situations as presented by Fayad and Laitinen (1997). It does not provide evidence as to why the adoption of agile practices should be done in some specific order, or why, for example, TDD should be adopted after the collaborative planning.

There are some case studies available that describe how the CMM/ CMMI and agile practices have successfully been used together to formulate the so called combined improvement approach (Paulk 2001, Nawrocki et al. 2002, Morkel et al. 2003, Paetch et al. 2003, Vriens 2003, Kähkönen and Abrahamsson 2004, Baker 2005 and 2006, Fritzsche and Keil 2007, Sutherland et al. 2007). Compliance with reference models usually entails the generation of documentation, which is a consequence of the used agile principle of "working software over comprehensive documentation" (Morkel et al. 2003). Although a lack of documented evidence can be defined as one problem in achieving the goals of the CMMI process areas using agile practices, the description for example of *Requirements Management* in the CMM or CMMI does not explicitly state that requirements must be documented (Nawrocki et al. 2002). In an agile context, this means that requirements can exist but are documented in some other form than in plan-driven development projects, e.g. with user story cards or a user story database (Nawrocki et al. 2002).

On-site customer and continuous integration (Paulk, 2001) are important practices from a perspective of understanding requirements. For example, the on-site customer can be characterized as a practice that directly supports the understanding of the team (Paulk 2001), when the customer is always available and capable of explaining the requirements. Paulk (2001) suggests that XP's use of (1) stories, (2) on-site customer and (3) continuous integration achieves the CMM requirement management goals. Paetch et al. (2003) presents a similar conclusion and argues that agile practices are comparable with the requirements engineering techniques, where stories and product and sprint backlogs are used.

In XP, the planning game has been characterized as a way of communicating requirements to the team and therefore increasing understanding, but also as a practice that facilitates the commitments of participants, communication of changes and checks the consistency of the plans and requirements (Schwaber and Beedle 2002). The commitment of requirements in XP teams is achieved using an on-site customer, who is mainly responsible for the story description (Paetch et al. 2003). In Scrum, the commitment is based on the product owner's (Schwaber and Beedle 2002) responsibility and involvement in the sprint planning and sprint review meetings, as well as an ideology for the self-organizing teams authorized to make decisions (Schwaber and Beedle 2002). In some reports, release reviews are reported to be comparable with the requirements review and sprint planning meetings or planning games with the process of requirements change management.

The traceability of requirements can be supported by keeping a record of the earlier stories, tasks and functional tests described in XP (Fritzsche and Keil 2007). Inconsistency can be addressed by the agile practice of writing codes specifically to meet test cases. No code is accepted before it has been verified against the defined test cases (Morkel et al. 2003).

According to Paulk (2001) Project planning in CMM could be implemented as a part of the planning games at the beginning of each iteration of XP (Paulk 2001). This is based on Humphrey's advice, "If you can't plan well, plan often." XP also requires that the team and customer are involved in both the requirements management and software project planning. This supports increasing the common understanding and commitment of relevant actors in these process activities (Paulk 2001).

In XP, requirements are typically split up into tasks, described in story cards and estimated in the planning games by the development team. From the assessment perspective, this can also be defined as one way to document the results of an iteration as a part of the project planning process of CMMI (Paulk 2001). The common understanding and commitment of a customer is achieved only if the customer is continuously available during the software development project. In many cases, this is a practice that is not possible to implement, which means that some alternative "plan-driven" approaches to address this CMMI goal may be required (Nawrocki et al. 2002). Furthermore, Paulk (2001) states that estimation planning games and small releases can be used when establishing estimations and software project planning. In Scrum, project planning is performed in sprint planning and daily meetings using product and sprint backlogs as a tool for the work tasks management (Schwaber 2003).

The Software project tracking and oversight of CMM can be applied with a large visual chart, project velocity and commitments (stories) for small releases addressed by the XP (Paulk 2001). Kähkönen and Abrahamsson (2004) also argue that iterative planning meetings, daily meetings and continuously updated task descriptions, together with the corrective actions made in iterative retrospective workshops fulfil the goals of the project monitoring and control process area in CMMI. In Scrum projects, tasks are continuously discussed and estimated in daily Scrum meetings (Schwaber 2003). This means that in agile projects, project planning is accurate and updated daily (Schwaber and Beedle 2002). Table 5 shows the mapping model between the agile (i.e. XP and Scrum) practices and CMMI requirements management, project planning, project monitoring and control process areas.

Table 5. Example of the mapping between the CMMI project planning, monitoring and controlling as well as requirements management process areas and agile practices.

Process Area	**Agile practices**	**References**
Requirements Management	User stories, On site customer and Continuous integration, Planning Game/Sprint Planning, Small Tasks/ Estimations	(Fritzsche and Keil 2007, Kähkönen and Abrahamsson 2004, Nawrocki et al. 2002, Paulk 2001)
Project Planning	Planning Game/Sprint Planning, Small releases/ Sprints, Small Tasks/ Small Tasks/ Estimations, Post Game Session	(Nawrocki et al. 2002, Paulk 2001)
Project Monitoring and Control	Small Releases/ Sprints, Planning Game/ Sprint Planning, Daily Meeting, Post Game Session, On-site Customer	(Kähkönen and Abrahamsson 2004, Paulk 2001)

2.3.5 Summary

The existing literature proposes few frameworks or mechanisms for the improvement and adoption of agile methods which can be integrated with the

SPI steps or assessments. All of these presented so called ‘hybrid’ methods that integrate aspects of both the plan-driven and agile software development approach are, however, relatively new and not so well analysed empirically in the industrial environment. The above brief review of the current literature, which maps the CMMI and agile practices, indicates that agile practices can be used when evaluating the project planning, monitoring, control and requirements management process areas in software intensive teams and organizations. On the other hand, it is clear that there is a need for agile organizations to collect and share information about the strengths and improvements needed between the several teams at organizational level.

2.4 Summary of Chapter 2

While CMMI has promised many benefits, CMMI based improvement programs have often been found to be too time-consuming and requiring an overly document driven software development approach. As indicated in Chapter 2, many organisations have recently experienced cost reductions, increased development speed or better quality or communication due to the adoption of agile practices, but have then encountered problems related to documentation and communication in the agile software development context.

While the existing hybrid models for SPI and the adoption of agile practices provide an interesting new research field, indicating the importance of research into the combined use of agile and plan-driven techniques, they do not seem to provide the needed approach for dealing with the challenges that appear when the software development process is improved and mediated with CMMI and agile practices.

3. Towards a Framework for Improving Software Development Process Mediated with CMMI and Agile Practices from Communication Perspective

This chapter first provides a discussion on how to build a framework for the research and then presents the hybrid framework that supports both the communication improvement of the software development process and adoption of agile practices.

3.1 Definition of the Framework

A framework, generally, provides structured mechanisms to define phenomena in the research and link how they relate to each other (Weick 1995). A conceptual framework is a tool for explaining, either graphically or in a narrative form, the main constructs to be studied i.e. the key factors, constructs and relationships between them (Miles and Huberman 1999). In the conceptual framework, the concepts used in the study were assembled like a “jigsaw puzzle”. The goal is to work out how the concepts fit together and relate to each other. The conceptual framework consists of patterns of concepts and their interconnections (Fisher 2007).

Use of the Miles and Huberman (1999) data analysis approach (i.e. setting out ‘bins’, naming them and clarifying their interrelationships) can also lead the researcher to the conceptual framework. The framework itself is a mechanism to help the researcher decide the most important and meaningful variables for data collection and analysis (Miles and Huberman 1999). It does not have to be complicated, but just requires a simple description of the causes and relationships between the key factors in the overall research (Fisher 2007). The research framework can have five key purposes (Schwarz et al. 2007):

1. to summarize the assumptions of a research stream
2. to provide a new focus within a research stream
3. to integrate previous research studies
4. to facilitate future research
5. to synthesize previous research in an actionable way for practitioners.

To achieve a clear framework, the graphic presentation needs notations which are systematically used during the research study. The notation for the framework presented in this thesis is based on the books of Miles and Huberman (1999) and Fisher (2007). The type of presented framework in this thesis is a cyclical, cause and effect model (Fisher 2007) which means that the framework is shown as ‘boxes’ and ‘arrows’.

3.2 Needs for the Framework

The framework of this study is based on the elements presented in Chapter 2. This section discusses the need for the framework presented in this study.

Although, CMMI based improvement programs have been used to improve productivity of software development (Galin and Avrahami 2006) and agile methods promise practices for improved collaboration, communication and project management (Williams and Cockburn 2003), the possibility combining of these approaches has been critizised, even as a fundamental challenge of software development (Turner and Jain 2002). Reasons for the critique are based on the assumption that the CMM or CMMI based software process improvement would in some context lead to the too document driven software development approach (DeMarco and Boehm 2002, Highsmith 2002b). This is against the principles of agile software development (Agile Manifesto 2001). On the other hand, the use of CMMI reference model has been defined as too time consuming and process oriented which means that the efficiency may disappear from software process improvement because the support and feedback from actual developers to software process improvement are missing (Laitinen and Fayad 1998). There seem not to be empirical evidence that the software process improvement order presented by maturity levels of CMMI would actually work in industries (Fayad and Laitinen 1997).

As claimed in Chapter 2, iterative improvement mechanisms are a valuable way to harvest feedback from development teams and give them a regular possibility to improve their way of creating software and following the software processes (Salo and Abrahamsson 2007). The agile software development approaches do not, however, provide solutions about how to share information on the strengths and improvements between the different teams (Salo and Abrahamsson 2007).

Therefore, there is a need for an approach which contains the aspects of organizational level improvement and adoption of agile practices in software intensive organizations (Salo and Abrahamsson 2007).

The framework as presented in Figure 4 was developed to increase the understanding of the link between the CMMI and agile practices, to provide an approach for starting software development improvements using both agile practices and CMM goals. The primary goal of the framework is to build a link between the concepts 'hybrid/ lightweight', 'CMMI', 'assessments', 'agile practices in use' and 'communication' and to integrate the presented research papers in this thesis. Another purpose of the framework is to focus on the research and research challenges presented in Chapter 2 and, at the same time, identify the parts that are outside the scope of study and to integrate previous research studies around the selected key concepts.

3.3 Framework for this Study

The framework of this study presented in this study illustrates:

- how to facilitate and validate software development processes mediated with CMMI and agile practices
- how the developed hybrid, lightweight improvement approach can be combined with the use of agile practices in iterative project level software process improvements
- how the use of agile practices affects the communication in software development teams and organizations.

The framework in Figure 4 has been constructed on the basis of the description of the plan-driven and agile software development variables presented in Chapter 2.

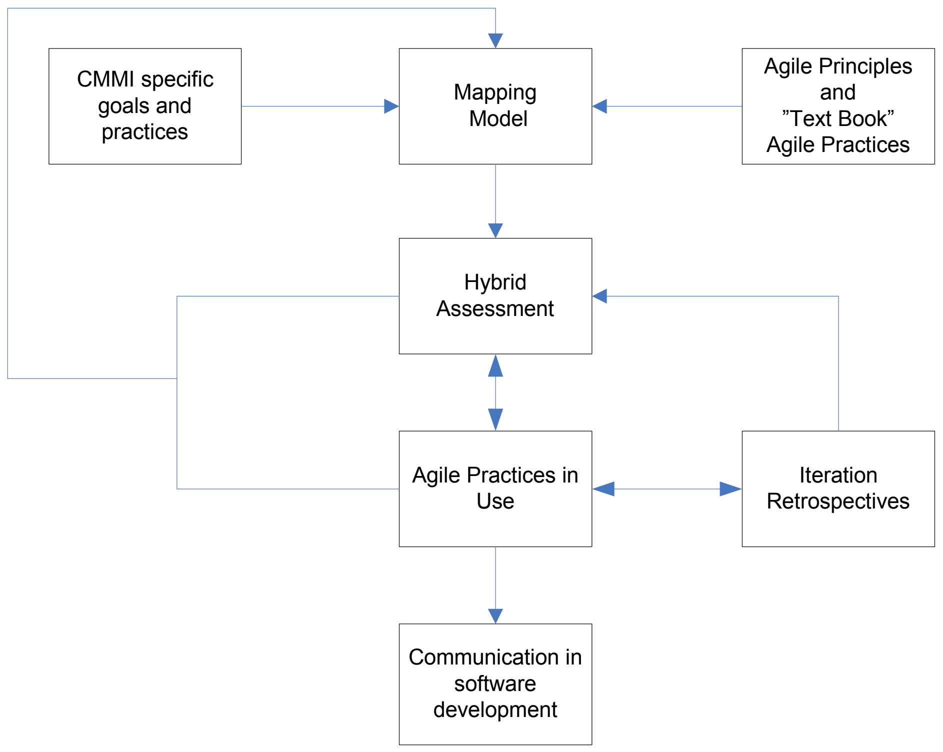

CMMI specific goals and practices
Mapping Model
Agile Principles and "Text Book" Agile Practices
Hybrid Assessment
Agile Practices in Use
Iteration Retrospectives
Communication in software development

Figure 4. Framework for this study.

The framework in Figure 4 suggests that:

- the CMMI model defines goals for each process area, and a basic way of achieving those goals is to use practices described in CMMI
- a 'Mapping Model' describes how practices in the CMMI process areas are complemented or replaced by a customized set of agile practices. The 'Mapping model' between the 'CMMI goals and process areas' and agile practices could be used as a tool when assessing the software development process with the 'hybrid assessment' method
- the 'Hybrid, assessment' produces both plan-driven and agile based improvement suggestions that may drive to 'agile practices in use'
- 'Iteration retrospectives' are used to evaluate the software development in which 'agile practices are in use'

- ‘Iteration retrospectives’ drive the ‘agile practices in use’ and work as a mechanism to harvest software process improvement needs from the team members
- the data from ‘iteration retrospectives’ can be used at an organizational level while implementing ‘hybrid assessments’
- the using of ‘agile practices improve the ‘communication in software development’
- the software development process where ‘agile practices are in use’ can be further evaluated and improved within a ‘hybrid assessment’ approach.

The relationships between the constructs of the presented framework are described in the research papers I–VI in this thesis, using the relations between the constructs and the papers from Table 6.

Table 6. Relationships between the constructs of the presented framework.

Relationships between the constructs	**Reported in the research papers**
Mapping model – CMMI goals and agile practices	Papers II, III, IV
Hybrid Assessment approach – Mapping model	Paper II, III, IV
Assessments – Agile practices in Use	Paper I
Iteration retrospectives and Assessment approach	Paper V
Iteration retrospectives – Agile practices in use	Paper I, V
The use of agile practices – impacts on communication	Paper VI

Each link between the components of the framework is explained briefly in the following sections including the description of the link between this and other frameworks described in Chapter 2.3.

3.3.1 Mapping Model and CMMI Goals and Agile Practices

Papers II, III and IV of this thesis all focus on providing a description, discussion and the key findings of the multiple case studies which have integrated CMMI specific practices, XP and Scrum practices in a real industrial context, utilizing the mapping model (as summarised in Appendix 2) as a framework for the question list creation and data analysis.

3.3.2 Hybrid Assessment Approach and Mapping Model

The research Papers II–IV describe how to apply hybrid, lightweight assessment in multiple contexts and how to produce practical improvements based on both the specific practices of CMMI and practices of Scrum and XP. The hybrid assessment presented in Papers II–IV use the same idea as defined in the Boehm and Turner (2003a) framework. The idea is to define both the agile and plan-driven risks so as to tailor the software development processes based on the assessment results which are specifically generated from the context and domain of the evaluated software organization. In this thesis, Boehm and Turner's (2003a) five agility factors were also applied in 2 case companies as a starting point for the improvement work. This was done in order to develop an understanding of the context factors of the assessed teams. During the assessment it was revealed, however, that context factors in Boehm and Turner's (2003b) framework do not provide enough detailed improvement suggestions about how to continue with improvement and agile practice adoption. Therefore, a more specific, practice based assessment approach is still needed.

At the time when the case studies were implemented in the case companies, Sidky's framework did not yet exist. Presuming that Sidky's framework will be more established for future research, it could be used as a tool in assessments to determine which agile practices should be first focused on in the assessment processes.

3.3.3 Assessments – Agile practices in Use

In Paper I the use of agile practices was analysed and discussed using the assimilation stages defined in innovation adoption theories. It was found that since the need is the key driver for a comprehensive and sophisticated use of agile practices, assessment can be also used as a mechanism to analyse the needs of companies and to identify agile practices that the software development team could first adopt.

3.3.4 Iteration Retrospectives and Assessment Approach

One purpose of Paper V was to use the QIP model as a framework to show how the improvement and adoption of agile practices can be implemented at team and organizational level and how the team level iteration retrospectives can be linked to the previously presented agile assessment approach (in Papers II and III). In one case company, the data from post iteration workshops (Salo and Abrahamsson 2007) and discussions with the workshop facilitator were used as one of the primary sources of data when doing the agile assessment. This gave new perspectives to the assessment situation as the assessors could view the improvements and implementation of the project processes as a changing, broad process instead of a snap shot description of current working practices.

3.3.5 Iteration Retrospectives – Agile Practices in Use

In Paper I it became clear that the iteration retrospective is a practice that regularly used has an impact on the assimilation levels reached in the development teams. A similar observation was made also in Paper V, in which the analysis of a post-iteration workshop disclosed its positive affects on the way the teams used the agile practices.

3.3.6 Agile Practices in Use – Impacts on Communication

Paper VI analysed the implications of the single adopted agile practices for communication in the software development teams and organizations. This evaluation is important to prove that the assessment approach is a valid way of affecting the software development processes.

3.4 Summary of Chapter 3

In this chapter the framework was presented to illustrate the problem of how to improve the software development process mediated with CMMI and agile practices. The components of the framework have been derived from the discussion in Chapter 2. The links between the constructs are described in the research Papers I–VI on the topics constructed between 2005–2008 in 4 case companies.

The framework shows the different elements that should be taken into consideration when making improvements to software development processes by integrating both plan-driven and agile aspects of software development. Therefore, the framework makes it possible for practitioners and researchers to reflect on the improvement of software development processes as a rich and complex process influenced by all the framework components and interaction between them.

4. Research Design

In this chapter there is a discussion of the research approach and methods adopted in this study. Another goal of this chapter is to provide a brief description of the companies used in this study with the basic information about the company background, technology context and use and their motivations to adopt agile practices.

4.1 Research Approach and Methods

In the following chapters, the approach and research methods of this study are discussed. In addition, this chapter contains a description of research settings including a description of context of four case companies.

4.1.1 Research Approach

The research approach in this study was to conduct the research as a series of case studies. Each step is analysed and reported in several research publications. Following the principle of interaction in interpretive research this means the construction of research data through interaction between the researchers and participants (Klein and Myers 1999). In this study, the data used is mainly qualitative but in some cases structured in a more quantitative manner (e.g. papers IV, V, VI). The data collection is based on multiple evidences as suggested by Yin (2003). The used data collection sources are semi-structured interviews and group interviews, observations, software development documentation of the case projects, workshops and collaborative meetings.

Klein and Myers (1999) argue that a dialogical reasoning principle in interpretive research means that the researcher follows the rule: improvement of the understanding of the previous research stage becomes the prejudice for the next research phase. In this research, the focus shifted among several themes which were found to be relevant to explain the observed processes in the eight different teams in the four case companies. In Table 7, these themes are categorized based on the constructs described in the published research Papers I–VI. In Table 7, these themes are categorized based on the constructs described in the published research Papers I–VI. In table 7 the papers are included in the same order as they were written.

Table 7. Research Themes.

Paper Number	Theme	Explanation
III	Agile Assessment	Assessments are used in a lightweight manner as a way to identify suitable agile practices in an organization.
II	CMMI goals and agile practices	Mapping of the CMMI goals and agile practices in Paper II, through the three case studies.
V	Deployment, Adoption of Agile practices and agile assessments, agile practices	Agile Practice adoption is analysed and integrated with the QIP model and evaluated through the case study. The post-iteration workshop method has been linked with the agile assessments and QIP steps.
IV	Hybrid Lightweight assessments and agile practices	Hybrid assessment and agile practices are presented as a new method and evaluated through the case study.
I	Agile practices in use	Data in case company 3 has been used to increase the understanding of agile practices in use and to give motivation for assessment based on agile practice adoption approach.
VI	Agile practices in use and communication	Assessment data is used to clarify the impacts of agile practices on communication in software development teams.

In the first phase of this research, the purpose was to integrate agile practices and CMMI goals in the assessment process. The results of this first step were reported in Paper III. In the second phase, deployment and adoption concepts were selected as the focus of the research and published in Paper V in the purpose to integrate the previously presented assessment mechanisms with the organizational level agile deployment, adoption and improvement. After this, the mapping model was considered in more depth through the reported experiences of four case companies in Papers II and IV. Together, these papers increase the understanding of the improvement of the software development process mediated with the lightweight CMMI based assessments and agile practices. In the last phase, the key focus turned to the agile practices in use. This was done in order to validate the importance of the presented improvement strategy and presented framework.

Firstly, the impact of agile practice in use on communication was selected as the focus for a more detailed analysis in Paper VI. Secondly, agile practices in use were also researched from an innovation assimilation perspective in Paper I.

4.1.2 Research Method

This section, provides, firstly, the reasons why the case study method was selected and, secondly, an explanation of how the data collection and analysis were applied in this particular study. As the intention behind this study was to investigate a contemporary phenomenon in a real-life context case studies are considered by the researcher to be a suitable research approach for the overall study. The case study approach was selected as the research approach in this thesis, because it is beneficial in research situations where control over behaviour is not possible and research data can be collected through observation in an unmodified setting (Yin 1994). Case study research has been implemented and reported iteratively based on Yin's (2003) steps in case study research methods i.e. preparing data collection, collecting evidence, analysing case study evidence and reporting case studies.

While a case study allows capturing of details and the analysis of many variables, the method is criticized for a lack of generalizability, a critical issue for a case study researcher. Because the corporate, team and project characteristics are unique to each case study, comparisons and generalizations of case study results are difficult and are subject to questions of external validity (Kitchenham et al. 2002). However, Walsham (1995) argues that when using a case study approach, researchers are not necessarily looking for a generalization from a sample to a population, but rather plausibility and logical reasoning through developing concepts, drawing specific implications, and contributing rich insight into the researched phenomena. This study also investigates an improvement of the software development process using individual agile practices, not agile methods in general. Therefore, the agile practice is considered as the unit of analysis of this thesis. Agile practice in this thesis can be defined as the practice that has been described in Scrum and XP.

The level of the research can be individual, group, or organization (Hovorka and Larsen 2006). As for the level of analysis, the presented research is focused on a software development team (group level) and those stakeholders (organization level) who have a direct impact on the software development teams. These actors or stakeholders have been differentially defined in existing research studies. For example, in coordination theory Malone and Crowston (1994) discuss actors in information transferral and coordination. The key actors defined in this thesis are the customers, managers and individual programmers or group of programmers. Boehm (2003) creates a theory of value based software engineering focusing on the actors and success models of the most frequent project stakeholders. He characterizes the most relevant stakeholders in the software engineering as developers, managers, users, maintainers, sales people and acquirers (Boehm 2003). On the other hand, Leon (1995) categorizes different stakeholder groups for system development, also defining two other stakeholders that have effects on the work of a development group. There are other development teams that have impacts on the work of the development team and support staff, for example, quality engineers who ensure the quality of the end product. In the CMMI product suite, (CMMI 2006) a 'stakeholder' is characterized as:

> *"A group or individual that is affected by or is in some way accountable for the outcome of an undertaking. Stakeholders may include project members, suppliers, customers, end users, and others."* (CMMI 2006).

In this thesis, the level of analysis includes not only the development team but also the managers, customers, support staff and other development teams from the organizational level who have a direct impact on the work of the evaluated software development team.

4.1.3 Collection of Empirical Evidence

Personal face-to-face interviews are considered as efficient data-gathering techniques especially for interpretive studies (Yin 2003). Also, the information gathered is likely to be more accurate than information collected by other methods, since the interviewer can avoid inaccurate or incomplete answers by explaining the questions to the interviewee (Oppenheim 1992).

During the period 2005–2008, the initial approach was to interview managers and employees in four firms that were in the process of implementing the XP and Scrum methods. Since this time period was still early for software process improvement via agile practices, the firms were chosen opportunistically based on their business goals to adopt agile practices. During this research, a total of 27 individual interviews were conducted at these four companies (Table 8). These interviews were semi structured and lasted for about 60–90 minutes each.

Most interviews were tape-recorded and transcribed verbatim. 12 additional interviews can be characterized as group interviews, in which the semi structured issues were discussed together with the software development teams and facilitated by the author of this thesis (Table 8). For these interviews, where tape recording was not possible, detailed hand written notes were taken and immediately transcribed following the interview.

Table 8. Data Collection.

Case company	1	2	3	4
Individual interviews	10	5	8	4
Group Interviews	5	1	5	1

Since one goal of these field studies was to assess the software development process and define new improvements based on project needs and agile methods focusing on the requirements, project management and engineering processes in the CMMI, each firm provided opportunities for interviewing developers, project managers and product managers. In the individual interviews 20 of the respondents were developers, 3 were also doing project management work, 6 were only project managers and 1 was quality engineer and 1 was in customer role in the agile software development project.

All the interviews were conducted in a responsive manner (Rubin and Rubin 2005, Wengraft 2001). In some interviews, the client (i.e. representative from case company) also participated in interviews in the role of the interviewer, which enabled reflection and efficient collaboration. Furthermore, the research team kept a note of the questions asked during each interview, and analysed their effectiveness, refining or adding to the set of questions via telephone or e-mail.

The documentation review and field notes were used as complementary data collection methods. The sources of documents include information system development documents, project management documents, corporate websites or brochures, and other available publications.

4.1.4 Data Analysis

Coding is often used in qualitative research, systematically labelling concepts, themes, and artefacts so as to be able to retrieve and examine all the data units that refer to each subject across the interviews (Miles and Huberman 1999). The research, presented in this thesis, is based on the framework building approach in which the researcher sets out 'bins', names them and clarifies the interrelationships between them as described by Miles and Huberman (1999). The 'bins' can be events, settings, processes, practices or theoretical constructs (Miles and Huberman 1999).

The conceptual framework describes the main areas to be studied, the key factors, the constructs of the research and the key factors between them. The coding structure, adopted in this research, consisted of two distinct mechanisms. Firstly, agile practices were used as 'bins' to define codes for each interviewee, the first segmentation and filtering was done and the interview data collected. Secondly, pattern coding was used as a way of grouping the summaries of previous codings into a smaller number of themes or constructs as described by Miles and Huberman (1999).

4.2 Research Context

Drawing conclusions from the empirical results is always difficult, because the results are largely dependent on the project settings. This study collected the data on the implementation of the XP and Scrum practices in four companies which all are working in dynamic, global markets. Agile practices were evaluated as way to improve companies ability to respond to changes within short, time boxed development cycles.

All of these organizations are operating in the embedded software development area. None of these organizations did use agile practices at the beginning of this research. Agile methods were adopted in the selected projects in the first phase, alongside the plan-driven product development process. The case companies were chosen for this study based on their need and goals to adopt agile practices and need or background to use CMMI models. The context of the case companies provided valuable context for the research related to the software process improvement mediated with CMMI goals and agile practices from communication perspective.

The research was conducted in four case companies during the years 2005–2008. During this time, 8 teams in total were selected for the evaluation. Table 9 gives an overview on the cases used in this research study.

Table 9. Overview of the Cases.

Case Company	**Projects**	**Used Agile methods**	**Evaluation of the output**
Case Company 1, Phase I	Team A Team B Team C	Plan-Driven	Paper II and III
Case Company 2	Team D	XP, Scrum	Paper II
Case Company 3 phase I and II	Team E Team F	XP, Scrum	Paper I, V, VI
Case Company 4	Team G Team H	Plan Driven	Paper IV

In two of the companies, case companies 1 and 3, the research was conducted in two stages. In the first period, the current status of the processes was analysed. In the second phase, 6 months later, the results of the previous analysis were assessed by holding second interviews and analysis. In case company 2, an evaluated project used the XP and Scrum practices whereas in case company 4, the evaluated projects were plan-driven but the organization representatives were willing to adopt agile practices as part of the SPI program.

The following chapters describe the research context of each company selected for the case study of this research.

4.2.1 Case Company 1

Teams A, B and C belong to Case Company 1 which delivers video technology solutions for 18 semiconductor manufacturer customers in six different countries in Europe, Asia and the USA. At the time of the research, Case Company 1 was a medium size company with 80 employees. All the development was located in Oulu, Finland. The development deals with domain and application engineering. This means that part of the developers in the teams do the application engineering at the same time, when the other developers integrate basic products based on specific customer needs. At the moment of the first interviews, the domain engineering teams were based on deep negotiations and requirements "freezing" before any actual software development activities.

Organizational Background

Because the company was still quite small all developers were located in the same building, daily communication between the developers, project managers and business units in all three teams was possible. Due to the efficient communication study participants said that there was no need to put so much emphasise on documentation. The need for more systematic mechanisms for communication and software development management was, however, necessary due to the increasing amount of developers and management.

Technology Context and Use

All the products that are developed in case company 1 are embedded and have more hardware components. At the time of the first interviews all the developers, in all three teams in case1 worked in the same building, some of them in the same rooms and others in separate rooms. All of the evaluated teams were also quite small, having two to five developers in which some of them design the hardware part of a whole product.

Team A has performed domain engineering activities for a previously implemented embedded product. The goal of the project has been to tailor the product to a specific customer environment based on continuous customer needs and requests for changes. Two developers have made the changes, mainly ad-hoc, as quickly as possible based on the current customer demands. The project began with long contract negotiation and specification phases following also the previously defined milestones and milestone criteria.

The goal of Team B has been to develop an embedded product with new technologies. At the first data collection period, the project had five software developers and a group of hardware developers. At that time, the project was in the specification phase having only internal customers.

The developers in Team C have conducted application engineering in parallel with domain engineering work for several customers. At the assessment moment, the project was in the implementation phase. Previously, the project only had one customer and the product manager took care of gathering customer requirements. In the later phases, the project, however, had several new customers that were all continuously making new requirements or requests for changes.

Motivation to Adopt Agile Practices

Because domain engineering projects of the teams in case 1 were typically based on deep negotiations and requirements "freezing" before the actual development activities, the management had, first, a sceptical attitude towards increasing the agility. This attitude, however, changed when new evidence of the benefits of agile software development came from within the case company 1 projects and other companies.

4.2.2 Case Company 2

Team D is part of a large international company located all over the world. It delivers mobile communication applications for the fast moving dynamic, global market. The aim of case company 2 is to provide mobile solutions in an area of imaging, games, media and businesses and also to provide equipment, solutions and services for network operators and corporations.

The work in Team D was based on case company 2's corporate research centres which develop technologies and creates competencies in technology areas vital to the company's future success. The research centre also supports the company's four business groups by interacting closely with them in order to develop new concepts, technologies, and applications. Conducting research within a cooperative and global network underpins their long-term technology visions and they cooperate with universities and other industrial players to widen the scope of technology.

Organizational Background

At the time of the interviews, the process model in used in case company 2 was based on the plan-driven software development approach, in which the management of the project followed milestones. The milestone criteria required producing specific documents in each space of the software development process.

Agile practices were first applied inside the evaluated Team D and therefore, the manager had a difficult role to combine the highly plan-driven milestone based organizational level culture and the project which internally worked on an agile basis. Since the first pilot projects, using agile methods, were successfully applied, the developers began to apply the agile software development approach on a large scale also in other development projects. During 2,5 years of time, the overall culture of the company changed radically emphasizing agile principles and quick feature deliveries.

The culture of case company 2 was, at that time, mainly based on a plan-driven software development process. Our analysis reveals that, at the assessment time, most of the management in case company 2 were not aware of agile methods or were not willing to change the plan-driven milestone routines to support software development in which agile practices were used. On the other hand, the organization had a clear willingness to pilot agile practices because, the whole development Team D was selected from other companies based on their agile software development background.

Technology Context and Use

Team D was the first project in case company 2 in which agile practices were really piloted. The goal of the evaluated case project was to develop a new generation report management system to support product development projects.

Another goal was to evaluate the success of the agile methods in the case company environment. The developers were mainly subcontractors from other organizations and selected for the pilot project because of their 2–3 years experience with agile software development. Early versions of the report management system were developed during the previous years used Excel scripts. Now the goal was to transfer the data into a new database and redevelop new software to support metrics collection and management in product projects.

Team D included four software engineers, who worked in the same office space, a project manager and customer who participated in validating the results and provided required daily technical support for the project development. The evaluated project used the selected set of agile practices from XP and Scrum methods. The product was implemented in six two week iterations.

Motivation to Adopt Agile Practices

The evaluated agile pilot project implemented by Team D was the first pilot using agile practices in the case company 2. Later on, the number of pilot projects using agile practices increased significantly. The first pilot project already provided evidence that agile practices would help teams in case company 2 to produce features of the products for earlier market delivery and, thus, created a better response to the demands of the moving market.

4.2.3 Case Company 3

Teams E and F belongs to a medium sized company (case company 3), which produces products to protect consumers and businesses against computer viruses and other threats from the Internet and mobile networks. The company is located in both Oulu and Helsinki in Finland, but the company has also many regional offices located around the world. The corporation produces services for the global market in more than 90 countries. The company was founded in 1988 and has been continuously expanding since then.

Organizational Background

At the time when the research was beginning in 2005, the case company was mainly making software using plan-driven software development methods and principles. The management of the company reported the achievement of CMMI level three and the reliability of the software was high. All the developers worked in separate rooms and, due to the demands of standards, the documentation was described as important mechanisms of communication between the developers, different projects and testing groups. The company had an iterative process model that was based on the milestones and the milestone criteria. A project status was reported to the management regularly at milestone meetings.

In 2004, the company management made a decision to pilot agile software practices as a way to respond to the needs of dynamic markets and increasing competition. The key challenge for the company was not the quality but the speed of the software development, which was evaluated as an essential factor for the particular business field that they were working in. Just before the research started, the company had organized agile method training for all the project managers and software developers to make them aware of the new approach to manage and implement the software development process. At the time of the data collection, the company was just implementing their first pilot projects using Scrum and XP practices.

The goal of the research team was to help the company in moving towards a new agile based software development. This was done by several interviews, workshops and collaborative meetings which all supported the company in the agile practice deployment.

Technology Context and Use

Case company 3 delivers products for securing computers, networks and mobiles against the increasing complexity of computer viruses, worms, hackers and other threats that appear in the field of information systems. Assessments were conducted for the four software development teams that all were the first projects where agile processes were adopted.

Team E focused on developing a tool for security system management. The project team consisted of six persons including four software engineers, a Scrum master and the quality engineer. The core of the project worked in an open office space. The project had been ongoing for over a year and the Scrum method had been used in the project for about five months at the moment of the first interview period. The second interviews were held after the project had finished, when the development had continued for 1.5 years.

The goal of Team F was to develop a mobile security application. The core of the case project consisted of four software developers and two quality engineers, who worked in an open office space. The project team conducted five software development iterations. The team leader of the project was an expert in the agile process and was provided by the research organization. Thus, the team had constant support and coaching available for adopting the new agile process model and technologies.

Motivation to Adopt Agile Practices

At the moment of the research case in Teams E and F, company 3 was adopting agile software practices in the first pilot projects. Due to successful experiences (e.g. the pilot projects showed a radical increasing in the speed and quality of the software development process), the company management made a decision to apply the agile practices on a large scale at the corporation level. Feedback for the process model creation was collected from several projects.

4.2.4 Case Company 4

Teams G and H belong to Case company 4 that is a small company in Ireland with a parent company in Denmark. The company is developing safety critical products with a key focus on the application areas such as test systems for the electrical and functional testing of complex electronic systems and Network and Command & Control systems. The domain of case company 4's products is currently in the space industry, but the company is also making products for the automotive domain. Case company 4 consisted of 6 developers and three managers in Ireland. The parent company in Denmark has 50 employees in total.

Organizational Background

The case company 4's Irish unit was founded only one year ago, which means that the software development practices were just developing. The culture of the company was highly plan-driven, which means that development was based on a long specification phase and development cycles.

Technology Context and Use

At the moment of the interviews, the company had two ongoing teams working on two projects, both having the goal to make a commercial system for the automotive industry. Both projects built an embedded system. The projects also used commercial shelf-components as one of the key strategies for efficient development.

The case projects in company 4 had their own project manager and the six developers changed between the projects when needed. Both of the projects had been running in the case company 4 for half a year. In Team H, the software development was dealt with in three, three-month increments, whereas in another project, the overall development life cycle was six months. The parent company had been mainly responsible for the requirement's definition and communication with customers in both projects. Team I of case company 4 was mainly working similarly to Team H. Team I had, however, created the requirements specification document interpreting the requirements document made by the parent company.

The communication between the parent company and case company 4 was occurred but not in regular manner. Most of the changes to the requirements and the overall systems were managed in an unsystematic way, but the development, verification and validation of the products were well implemented, which is important for the safety critical domain.

Motivation to Adopt Agile Practices

Teams H and I began the change towards the use of agile practices in software development process based on the assessment results made with the AHAA, Agile Hybrid Assessment method for Automotive, Safety Critical SMEs. Based on the assessment results, the developers and management of case company 4 identified the particular areas in which they could use agile practices to solve their current problems and, thus, improve their software processes.

5. Research Contributions

In exploring the research phenomenon defined in this thesis, each individual paper contributes to improving the software development process mediated with agile practices. The following chapters describe the research contributions including the key findings for each selected research paper.

5.1 PAPER I: Agile Practices in Use from an Innovation Assimilation Perspective. A Multiple Case Study

Paper I focuses on one of the key elements of the thesis: the combined use of XP and Scrum practices. It is based on the strong lens of existing well-established innovation adoption theories (Davis 1989; Cooper and Zmud 1990; Fichman 2001; Gallivan 2001) and it presents three teams (including a team from a case company 3) which compare and evaluate the agile practices in use from an innovation assimilation perspective. Paper I is based on the Rogers (2003) argument that an innovation can be an idea or practice, which is perceived as new by its adopters. This is because this definition leads to the characterization, which is "*Agile practices are software process innovations*". The investigation of the adoption processes of agile practices is critical because "*it provides another level of explanation, describing the importance of the agile practices in a system development and adoption setting*" (Hovorka and Larsen 2006).

As a result, Paper I suggests that while agile practices addressing the needs of a software development team or an organization have the potential to be reutilized or infused, it is first sensible to identify the challenges in the software development project or organization and then to select the relevant agile practices to adopt in the area, rather than take the whole set of agile practices of some method as such. This supports the approach in which the adoption of agile practices begins by identifying the areas of a software development process in which the improvement is implemented through a CMMI goal and agile practice-mediated framework, which is one of the contributions of this thesis.

Another finding from the study presented in Paper I is that iteration retrospectives are a so called 'superior' practice, in which regular use may lead teams to a more comprehensive use of agile practices. This supports the argument that the use of iteration retrospectives affects the adoption and improvement of agile practices and needs, therefore, to be linked with the assessment (i.e. the data of iteration retrospectives might be valuable while doing assessments for a company where agile methods are in use).

Key Findings of the Paper

- ➢ It is sensible to identify and select the relevant agile practices to adopt, based on the needs of the software development project or organization in the areas of most need.
- ➢ Iteration retrospectives drive the assimilation level at which agile practices are used.

5.2 PAPER II: An Approach Using CMMI in Agile Software Development Assessments: Experiences from Three Case Studies

As recognized in the first paper of this thesis, organizations have a need for assessment mechanisms that take the agile context into account, to show how to validate the extent to which the customised method(s) in action meet the CMMI model. To cater for this, Paper II begins the development of the assessment approach focusing on the mappings between the CMMI goals and agile practices. The mapping and experiences of the assessment approach has been presented through three different organizations (in case companies 1–3) and the experiences of a total of seven team assessments.

Paper II uses CMMI goals, integrated with agile practices, as a tool to identify agile practices based on the current need for software development projects. It suggests that the use of agile practices increases the discipline in the CMMI based assessment situation and helps assessors not to wrongly interpret the different phases of the development project. As shown in the research selected in this thesis, the mappings can be used to clarify the connections between the agile and plan-driven processes and thus, increase understanding of the assessment results and provide concrete practical solution proposals for the company on how to improve software development process.

Among other agile practices, Paper II reveals that, for example, knowledge of agile practices such as product and sprint backlogs; sprint planning, sprint reviews; self-organizing teams in industries brings some new approaches for the assessment of requirements management, project planning, monitoring and the controlling process area of the CMMI. Furthermore, the paper proposes that compared to the plan-driven software development projects, it seems that an agile based approach enables project members to prioritize requirements as well as plan and monitor project tasks. On the other hand, the agile approach does not automatically lead to the success of software development process. Moreover, agile projects have improvement needs that are possible to identify using the CMMI model as a framework for analysis.

Although CMMI and agile practices seem to be integral elements, there are many challenges that the assessors face when evaluating software development processes in the context of an agile software development. The use of the CMMI goals enables the assessors to cover all the relevant issues in the assessment situation at the same time as the agile principles lead the assessors to seek evidence of the goal achievement in other forms than the traditional word documents. For example, the use of the agile principle: "*working software over comprehensive documentation*" leads easily to the situation in which the evidence for assessments is not often available as a form of official documentation. This does not mean that the evidence does not exist. On the contrary, evidence can be collected using face-to-face communication with the groups of developers and project management. The ideology of the self-organizing teams and efficient communication in the team makes the group interview situations more open and, therefore, easier from an assessor perspective.

Key Findings of the Paper

- Mapping the CMMI process areas to agile practices is a valuable tool for identifying suitable agile practices for organizations that use the plan-driven software development approach.
- Lightweight, hybrid assessments, integrated with CMMI goals and agile practices can be used as a way to validate software development process using agile methods.

5.3 PAPER III: An Approach for Assessing Suitability of Agile Solutions: A Case Study

Paper III describes the results of a research study, in which agile practices are used to identify improvements for case company 1. It focuses on the requirements and project management process areas of the CMMI. It presents an assessment approach to make an assessment in organizations that have a goal to increase the ability to respond to changes in dynamic market situations.

From the research framework perspective in Figure 4 the paper contributes to the link between hybrid assessments and agile practices. For example, it suggests that lightweight assessments (i.e. assessment with low costs, focused processes, simple assessment processes and modified use of assessment processes) are a sensible way of identifying suitable agile practices based on the project needs in a software organization. This supports the use of lightweight assessment methods in assessments made in the context of agile software development. The paper also proposes that the assessment integrated with agile practices (i.e. agile assessment) increases the understanding of what agile practices would suit the organizational culture, current working methods and environment. The findings, presented in Paper III, support the assumption that the use of agile practices improves project monitoring, risk management and requirement traceability in plan-driven product development.

Key Findings of the Paper

- Lightweight assessments integrated with agile practices is a sensible way of identifying suitable agile practices in plan driven software organizations.
- Lightweight, hybrid assessments, integrated with agile practices, can be used as a way of validating suitable agile practices in a plan driven organization.

5.4 PAPER IV: AHAA – Agile, Hybrid Assessment Method for Automotive, Safety Critical SMEs

Paper IV presents the ‘AHAA’ Agile, Hybrid Assessment method for Automotive industries, integrating the lightweight ‘ADEPT’ assessment method (McCaffery et al. 2006, 2007, Wilkie and McCaffery 2005) within the agile assessment approach presented in Papers II and III. It uses data from the assessment in case company to describe the practical application of the presented method in a new domain. From the framework perspective in Figure 4, Paper IV integrates the assessment approach (mediated with agile practices) presented in Papers II and III with the ‘ADEPT’ method and specific process areas of Automotive SPICE^{TM} (Automotive SPICE). By doing this, the paper creates a new method called the AHAA – A Agile, Hybrid Assessment Method for Automotive for SMEs working in the safety critical domain. Furthermore, the paper provides a description of how the AHAA method was developed and how the ‘ADEPT’ method and ‘agile assessment’ approach were integrated. Paper IV shows that is sensible to use agile practices as part of the lightweight assessment also in the safety critical domain. Additionally, Paper IV provides some new evidence on how to integrate requirements management and project planning, monitoring, controlling and configuration management process areas of CMMI and agile practices and how to use the mapping model to generate the needed information for creating questions, analysing data and to identify suggestions for suitable agile practices based on the improvement needs of the one case company. For example, the AHAA assessment provided 7 actions to address the 14 issues highlighted during the assessment of the requirements management process within case company 4.

The AHAA recommendations enable this to be resolved by adopting a combination of plan-driven and agile based actions. For example, according to the participants of assessments, case company 4 will adopt plan-driven practices such as introducing requirements capturing templates (to ensure that the requirements are complete and verifiable) and developing a procedure for handling CRs, in addition to introducing agile based practices of iterative software development cycles, backlog-based requirement databases, continuous requirement analysis and requirement prioritisation with the customer and parent company.

Key Findings of the Paper

- Hybrid assessment, integrated with agile practices, is a sensible way of identifying currently suitable agile practices, also in the context of an automotive security critical software organization.

5.5 PAPER V: Deploying Agile Practices in Organizations: A Case Study

Paper V presents a framework that maps the SPI and agile deployment steps together in a model that can be used in the organization's continuous SPI. It uses data from case company 3 to summarize how assessments can be used to facilitate the SPI in organizations, both at team and organizational levels. Paper V helps to understand the link between assessments and agile practices in the presented framework, but also shows how the agile practice deployment and the SPI fit together. Paper V integrates the assessment approach – presented in Papers II, III and IV with the QIP model and iterative improvements required by agile methods – in the form of steps for the deployment of agile practices.

Based on the agile principle: "The team regularly reflects on how to become more productive and efficient" (Agile Manifesto (2001)), processes in the agile software development should be iteratively improved using the reflection workshops. As presented in Paper II, the continuous iterative process improvement makes a "snap shot" type of assessment challenging in the context of agile software development. In Paper V, this challenge becomes a bonus as continuing discussions and the shared documentation between the assessor and reflection workshop facilitator can be used as additional data for assessments.

Paper V suggests that lightweight assessments could be used in two phases of the QIP model and three phases of the agile deployment model. Therefore, the paper integrates the assessment approach as a part of a larger agile practice adoption and continuous SPI context. The empirical evidence, from the case study, illustrates how case organization 3 was able to employ and benefit from the deployment mechanisms. Particularly the management found the results of the lightweight assessments useful in monitoring the deployment process and in drafting an organization-specific agile process model, alongside their own plan-driven product development process.

Key Findings of the Paper

- Use of the results of the iteration retrospectives or post iteration workshops (PIWs) enables assessors to form an overall picture of agile software development projects.

5.6 PAPER VI: The Impact of Agile Practices on Communication in Software Development

The last paper of this thesis uses empirical data collected during the assessments in case company 3 in order to explore the impact of agile practices on communication in two software development teams. Paper VI investigates how agile practices affect communication. Based on the analysis, it suggests that some agile practices, such as an open office space, sprint planning, daily meetings and sprint reviews would have a stronger effect on the communication between the development team and stakeholders than the other XP or Scrum practices. Paper VI proposes that agile practices, without plan-driven mechanisms, do not provide the needed support for external communication. Contextual factors – such as the number of customers – can affect the need for plan-driven mechanisms to cover these communication and collaboration needs.

Thus, Paper VI reveals that although the improvements especially in the area of the communication seems to happen due to the adoption of agile practices, there are still room for improvements from communication and coordination perspective especially when the amount of stakeholders are increasing. Thus, additional mechanisms are still needed to cover, for example, the link between the software development team and its stakeholders, which still remains a challenge in the case projects.

Key Findings of the Paper

- The use of customized agile practices is not a silver bullet. There are many communication challenges in the context of agile software development which means that improvement is still needed.

5.7 Summary of Chapter 5

Table 10 summarizes the findings of this research through a conceptual framework. It also maps the findings to the research questions presented in Section 1.1. These empirical findings, although they must be regarded somewhat tentatively since they are based on a limited sample, are of interest and will be investigated further as part of future research in this area.

The results of this thesis suggest that these different conceptual foundations (agile practices and CMMI) should be viewed as complementary, rather as competition or an incompatible approach. In fact, the study shows that the integrated lightweight assessment approach, used as a part of the continuous SPI, is a sensible way to help software companies to make decisions on the adopted agile practices, especially when the goal is to improve requirements, project management processes and adopt agile practices.

Table 10. Findings of the study.

Research Question	Assumptions of the framework	Findings	Reported in the paper(s)
Q.1.1	Hybrid assessment produce both plan driven and agile based improvement suggestions that may drive towards the use of agile practices.	Lightweight, hybrid assessments, integrated with agile practices, is a sensible way to identify suitable agile practices in a plan driven organization.	III, IV, V
	The mapping model between the CMMI goals and process areas and agile practices could be used as a tool when assessing the software development process with the hybrid assessment method.	The CMMI process area and agile practice mapping is a valuable tool to identify suitable agile practices for a plan-driven organization.	II
Q.1.2	Data of iteration retrospectives can be used at organizational level while implementing hybrid assessment.	Use of the results of the iteration retrospectives (or PIWs) helps assessors formulate an overall picture of the agile software development project.	V
	Iteration retrospectives drives the level of agile practices in use.	Iteration retrospectives drives the assimilation level in which agile practices are used.	I
	The software development process in which agile practices are used can be further validated within a hybrid assessment approach.	Lightweight, hybrid assessments, integrated with the CMMI goals and agile practices can be used as a way to further validate the software development process.	II
Q.1.3	Individual agile "text book" agile practices can be used when improving the software development process.	It is sensible to identify or select relevant agile practices to adopt, based on the challenges of the software development project or organization in the area where it is currently most beneficial.	I
	The assessment producing both agile and plan-driven improvement suggestions may drive the use of agile practices.		
	The use of agile practices may improve communication in software development.	The use of customized agile practices improves communication in the teams but is not a silver bullet; there are still also many communication challenges in the context of software development process in which agile practices are in use. In some cases, agile practices can even hinder communication related to software development process.	VI

6. Discussion

The research presented in this thesis has implications both for theory and practice. In Chapter 6 there is a discussion of the findings of this thesis against the current literature and a discussion of the implications of the papers both for research and practice.

6.1 Implications for the Research

The research presented in this study supports increased attention to be given to the improvement of software development process, mediated with agile practices. Aggregating the different perspectives of improvement and examining the results through the framework leads to stronger results in terms of research validity than looking at each of the framework components in isolation.

Since Paper I shows that agile practices addressing the needs of an organization have the potential to be reutilized or infused, it is sensible to identify areas that an organization needs to improve and then select a relevant, customized set of agile practices to adopt rather than attempting to adopt a whole set of practices in an agile method. In this approach, described in papers II, III and IV, the practice is firstly to identify the organization's challenges, then map them to agile practice based solutions so as to understand what agile practices should be introduced and why.

This study contributes new evidence on the integration of CMMI and agile practices. For example, the CMMI model and agile methods have often been presented as approaches which are opposed to each other or at least difficult to integrate. This research increases the understanding on how to combine CMMI process areas and agile practices in an approach that supports finding beneficial XP and Scrum practices in the specific team context. In this approach, the mapping model between the CMMI goals has an important role.

The concept of agility is based on agile manufacturing and it has been generally defined as agility of enterprises to manage unexpected changes (Highsmith 2004). Although, the primary goal of this research was not to clarify the concept of agility, this study anyway contributes to the understanding of agility in terms of the use of individual agile practices. For example, based on the study in Paper I, it is suggested that the use of agile practices is not a straightforward or simple activity. This is because the use of agile practices is a changing phenomenon that can reach different levels of assimilation also inside the software development teams. From the research perspective, this means that there is a need to establish what a deep, sophisticated level use of agile practices actually means in software development teams.

While there has been some work linking theories from organizational management literature to organizational level agility (e.g. Hovorka and Larsen 2006) and examining communication in the agile context (e.g. Korkala et al. 2006, Layman et al. 2006b, Turner and Jain 2002), little attention has been paid to combining agile practices and software development process improvements and communication aspects. In addition, it is useful to link practices that are well grounded in industries with existing well establish theories. This is an approach that may lead to a more comprehensive set of findings. This study links agile practice to innovation adoption (Davis 1989, Cooper and Zmud 1990, Fichman 2001, Gallivan 2001) and coordination (Crowston and Kammerer 1998, Malone and Crowston 1994) theories. This can be a starting point for many future studies in which well established theories are used to understand communication, coordination and innovation in software development processes in which agile practices are used.

Both communication and coordination seem to be interesting concepts as a focus for research in software development processes using agile practices. While both of these concepts have been mainly used in theoretical studies made in the distributed software development context, it seems to be worth studying their extension also to co-located software development environment. Since the amount of research is currently increasing in the field of organizational agility and software development process or teams using agile practices, many researchers tend to base the concept of agility on the description of the Agile Manifesto (2001). One interesting finding in this research related to communication is, however, that the principles and values in the Agile Manifesto (2001) do not necessarily guarantee improved communication between the team and its stakeholders. For future research there is a clear need to establish the link between the Agile Manifesto (2001) and the use of agile practices.

6.2 Implications for the Practice

From an empirical viewpoint, this study provides results for the practical implications on two levels of software engineering: firstly, in SPI which is related to the CMMI level 5 goal to establish continuous software process improvement supporting incremental process, and secondly on the practical level of the software development process in teams and organizations. The following sections discuss the implication of this study. The discussion is based on the findings of this study, reflected against the relevant literature in this field.

6.2.1 Implications for Continuous SPI

The CMMI process area and agile practice mapping is a valuable tool to identify suitable agile practices for a plan-driven organization.

The assessment approach, presented in this thesis, is a novel strategy for assessment among others presented by Nielsen and Pries-Heje (2003). In this new strategy, the CMMI process area and agile practice mapping are used either as a tool to find a suitable combination of XP and Scrum practices for a plan-driven organization or as a tool for evaluating and continuously improving agile based software development projects, according to the criteria of lightweight assessment.

The mapping model was created first on a theoretical level by integrating agile practices under the specific goals of CMMI in the selected process areas in Paper III. Then the model was integrated as a part of the list of assessment questions which was used to create a framework for interviews as identified in Paper IV. During the overall research process the mapping model was used to create interview questions, which were analysed and continuously updated case by case as described in Paper II. Although, this research study, therefore, gives some examples of mapping between the goals of CMMI project planning, monitoring and controlling and requirements management process areas and agile practices, the results of this research only look at a narrow scope of this larger research area, which would be worthwhile pursuing in future research. As argued by Reifer (2003): "SEI could also provide leadership and stimulate the process community to develop recommended agile practice mappings for the SWCMM." This is because Level 5 organizations should have a technology change-management process to adopt innovations, such as XP into a normal practice (McCaffery et al. 2007) and working mechanisms to also continuously improve processes in agile software development organizations.

The fact that the mapping model presented in this thesis is focused on three process areas from CMMI level 2 does not mean that the approach would not be relevant to the other CMMI Levels. In fact, implications for SPI presented in this thesis focuses vertically on CMMI level 5 which encourages continuously improving the process performance through incremental and innovative changes. It seems also that the iterative continuous improvement approach integrated with lightweight assessments would be a valuable practice to support CMMI level 5 achievements related to the software development process in which agile practices have been used.

Lightweight assessments integrated with CMMI goals and agile practices are a sensible way of identifying suitable agile practices in a plan driven organization.

As in plan-driven processes, agile based software development also requires frequent inspection and adaptive responses (McCaffery et al. 2007). Agile software development is not easy to assess using CMMI (McCaffery et al. 2007). As CMMI specific practices differ from agile practices, it is difficult for assessors to analyse projects in an agile context (Kähkönen and Abrahamsson 2004). This thesis supports Anderson's (2003) argument that CMMI is a useful model to assure that the most significant software development viewpoints, related to the selected process focus, were also taken into account in the assessment in the context of the agile software development. This thesis reveals that an analysis of software development processes using agile practices brings some new discipline for assessments in the context of plan-driven software development. For example, the knowledge of agile practices facilitates assessors together with case company representatives to define more specific improvement suggestions for software development process.

The hybrid lightweight assessment approach provided in this thesis is different from other assessment approaches because it provides an organization with a combination of plan-driven and agile-based suggestions as to how to improve their software development. During the research process, Boehm and Turner's (2003a, b) risk based method was applied when defining the suitable projects for lightweight assessment purposes and analysing the data relating to contextual factors such as criticality, personnel, dynamism, culture and size. The method did not, however, provide enough practical improvement suggestions for the companies on how to improve software development processes using agile practices. The assessment approach presented in this thesis is said to be lightweight for the following reasons:

- first of all it is integrated into the existing Lightweight assessment method called ADEPT (McCaffery et al. 2007) which has been empirically evaluated in several companies and described in several journal papers (Wilkie and McCaffery 2005, McCaffery et al. 2006, 2007)
- secondly, it follows 7 of the 9 criteria outlined by Anacleto et al. (2004), for the development of lightweight assessment methods: low cost, detailed description of the assessment process, guidance for process selection, detailed definition of the assessment model, support for identification of risks and improvement suggestions, no specific software engineering knowledge required from companies' representatives, and tool support is provided. The exceptions to the criteria outlined by Anacleto et al. (2004) are that no support is provided for high-level process modelling and only the authors currently have access to method and the conformity with ISO/IEC 15504 is provided only in AHAA method
- thirdly, the approach was designed to adhere to criteria for lightweight assessments such as low costs (Richardson 2001), focused processes (Richardson 2001, Wilkie and McCaffery 2005), simple assessment process (Horvat et al. 2000, Kautz 1998) and modified use of assessment models (Batista and Figueiredo 2000, Kautz 1998).
- fourthly, the provided lightweight assessment method also shares some of the requirements of the Adept method (Wilkie and McCaffery 2005, McCaffery et al. 2006, 2007) meaning that the assessment is implemented without the purpose of certification (ratings are not required), both preparation and assessment time is minimized
- finally, the evidence collection during the assessment was mainly based on face-to-face discussions (what developers and managers say) instead of documented evidence and the purpose was to keep assessment documentation as lightweight as possible.

In this thesis it is claimed that hybrid integrated lightweight assessments, as reported in Papers II, III and IV, can be used to identify suitable agile based solutions for the improvement needs of teams using the plan-driven software development approach. This is possible to do in different product environments, such as embedded software development (Paper III) or safety critical, automotive software development (Paper IV). During the research, the agile practice based assessment approach integrated with the 'ADEPT' method (McCaffery et al. 2007) was described in practical steps on how to conduct assessments, so as to identify suitable agile practices for an organization or to improve the agile software development related to the requirements management and project management process areas.

Lightweight, hybrid assessments, integrated with the CMMI goals and agile practices can be used as a way to validate the software development process

It has been argued that even the CMMI Level 4 or 5 could be achieved using agile methods (Sutherland 2001). Based on this argument, it is assumed that the lightweight assessment can be applied in organizations that use the agile approach (McCaffery et al. 2007). This thesis sheds light on this research area by providing some new knowledge about the use of CMMI goal based lightweight assessments in the context of software development in which agile practices are used.

It is shown that CMMI, as a model, does not explicitly prescribe or require any particular work products. Rather, its purpose is to provide evidence that the processes related to the defined goals are performed as suggested by Baker (2006). This is the key reason, why the CMMI based assessments can be conducted in agile software development process, whose purpose is to produce "working software, instead of comprehensive documentation" (Agile Manifesto 2001). This is also why the lightweight assessment mechanisms, as they emphasize human communication based evidence in assessment situations (Wilkie and McCaffery 2005), were found to be suitable in the context of agile software development.

Iteration retrospectives drive the assimilation level in which agile practices are used

Iteration retrospectives have been successfully used as a mechanism to evaluate software development where other agile practices are in use (Salo and

Abrahamsson 2007). This approach was also shown to be beneficial for organizations and to motivate developers to participate in the daily software process improvement (Salo and Abrahamsson 2007). In this thesis, it is shown using assimilation stages from the innovation of adoption theories (Davis 1989, Cooper and Zmud 1990, Fichman 2001, Gallivan 2001) that iteration retrospective actually drives teams towards a more efficient and sophisticated use of agile practices.

The link between assessments and iteration retrospectives enable the assessor to forme an overall picture of the agile software development project

This thesis creates a link between the iterative project level SPI (where iteration retrospectives are used), as described by Salo and Abrahamsson (2007) and the organizational level SPI (using hybrid assessments). By presenting a framework on how to deploy and continuously improve the software development using agile practices, it was found that CMMI can be used as a way to improve software development in teams using Scrum or XP practices (Fritzsche and Keil, 2007). There are, however, several strategies for selecting the way to apply the assessment in companies (Nielsen and Pries-Heje 2002). The assessments, presented in this thesis, are based on the CMMI model and a software process assessment approach. It is also integrated with "day to day project management" (Nielsen and Pries-Heje 2002), creating a link between the assessments and the data of iterative retrospectives.

6.2.2 Implications for Agile Practice Adoption and Communication

It is sensible to identify and select relevant agile practices to adopt based on the challenges of the software development project or the area in an organization where it is currently most beneficial. There are some empirical studies that support a similar approach for the integrated process improvement and adoption of agile practices in companies, see (Packlick 2007, Sidky 2007, Svensson and Höst 2005). For example, Svensson and Höst (2005) present cases of agile practices being introduced in a software maintenance and evolution organization via assessment.

There are many communication challenges in the context of agile software development

The challenge of XP projects lies in the same problems defined in coordination theory (Malone and Crowston 1994): i.e. keeping the overall project focus and the independence of the defined work tasks. In this study, it is proposed that the use of a combination of Scrum and XP facilitates communication. In fact, some agile practices such as sprint plannings, daily meetings and sprint reviews were revealed to have a more positive effect on information transfer between actors than other agile practices. Rising and Janoff (2000) report similar results from the Scrum projects. They claim that the short time boxed iterations in the agile software development are a key reason for improved communication in the software development teams.

The use of agile practices may improve communication but does not solve all the communication challenges between developers and external stakeholder groups (Cohn and Ford 2003, Coram and Bohner 2005, Svensson and Höst 2005). In fact, many argue that interface management between agile and traditional teams are challenging (Boehm and Turner 2005, Lindvall et al. 2004).

7. Conclusions

In this thesis a framework is proposed that draws insights from both the literature on the use of agile methods as well as SPI and CMMI. There are three key contributions in this thesis.

First contribution is to provide a brief review of the literature with regard the terms of ‘assessment approaches’, ‘CMMI’ and ‘the use of XP and Scrum practices from communication perspective’. By highlighting the limitations and proposing solutions to the problems in these areas, this study generates a new understanding of the improvement of software development processes mediated with CMMI and agile practices.

Secondly, on the basis of this review, a hybrid framework is suggested to facilitate organizational level improvement of requirements management, project planning, project monitoring and controlling process areas of CMMI. Based on the framework the improvements on communication of software development process can be done in lightweight manner via CMMI goals, project level iterative improvement mechanisms and a combination of XP and Scrum practices. It is also shown that the use combined set of XP and Scrum practices improves some communication aspects inside of the development teams but may not provide enough practices for the overall communication or coordination of dependencies in broader system development where external stakeholders are involved.

Thirdly, the research described in the scientific research papers in this thesis can be regarded as a starting point for integrating well established theories in the use of agile practices and to modify software process improvement methods to fit better with improvement work in the context of software development process using agile practices.

7.1 Answers to the Research Questions

In this research, three research questions (Q.1–3) were posed, which are set out below with a summary of the results.

Q.1 ‘How to improve software development process mediated with CMMI goals and agile practices from communication perspective?’

This thesis presents a framework which goal is to support improvement of software development process from communication perspective using goals of CMMI and agile practices. The presented framework integrates previously developed ADEPT assessment method with agile practices and therefore, provides a new hybrid assessment method that supports both deployment of agile practices and assessment of plan-driven software development processes. The presented assessment method does not, however, support certification. On contrary, it suggests improvements in industries using lightweight, adaptive, 1–2 days assessments bracing against CMMI goals and agile practices. This is occurred more using informal (i.e. face to face) than documented formal evidence and based on the CMMI level 5 goals addressing both iterative improvement with project teams and SPI activities in organizational level.

The implementation of framework reveals that although the assumption was that use of agile practices improves the communication between the software development teams and their stakeholders, finding the balance between the amount of informal and formal communication seems still to be a problem especially in the project in which the agile practices are used. Although, the combined use of Scrum and XP practices for instance has been shown to improve the communication inside of the development teams, it does not seem to guarantee the efficient enough communication in the condition in which the amount of customers or stakeholder teams with feature interdependences are increasing. In the following, more detailed answers are provided also for the three sub research questions 1.1, 1.2 and 1.3:

Q.1.1 ‘How to facilitate the improvement of the software development process mediated with CMMI and agile practices?’

Firstly, the mapping model was developed to provide an example of how agile practices could complement the CMMI specific practices in the selected process areas. The development of the mapping model was started by focusing on the requirements and project management process areas of the CMMI. This was done without the goal of achieving the complete model, but rather for presenting some examples on how a mapping model could be implemented and used for assessment.

The second phase of the framework is a hybrid assessment which is based on the principles and methods of the existing lightweight assessment approaches that were originally developed based on the needs of small to medium size of enterprises. The facilitation of the improvement can be done using the lightweight assessment approach, in which the assessment questions are based on the mapping model. The research reveals that a lightweight CMMI based assessment is suitable for an agile context because it does not require documented evidence during the assessment and it seems to provide a mechanism to identify practical agile based improvement suggestions for the software development teams and organizations.

Q.1.2 'How to validate the improvement of the software development process mediated with CMMI and agile practices?'

In the third phase of the presented framework, the hybrid assessments are used to evaluate the software development process against the CMMI goals. This is based on the assumption used among the lightweight assessment methods (i.e. ADEPT) that achieving the CMMI maturity levels does not require that projects achieve all the practices of the CMMI but emphasizes the achievement of the CMMI goals instead. As a conclusion of the assessments using CMMI goals and agile practices, it can be stated that although XP and Scrum give valuable practices for companies to achieve some of the CMMI goals, at least in some context, they do not seem to replace the plan-driven practices from CMMI. For example, in requirements management process area, the use of agile methods might facilitate the achievement of commitment between the stakeholders (in the case where the systematic meetings are used regularly and right people are attending to those meetings) but does not necessary quarantee the needed traceability of the requirements or efficient requirements impact analysis. Similarly in the project planning process area the use of agile practices such as small releases, sprint planning, daily meetings might enable the constant planning including more efficient scope evaluation and estimation activities in co-operation with teams and business units but lack, for instance, efficient risk management practices. Thus, some of the practices described in CMMI are still needed in agile software development, depending on the context of the software development e.g. the number of customers and other stakeholders and overall complexity of the developed system.

Q.1.3 'Does the use of agile practices improve the communication in software development teams and between the teams and stakeholders?'

To indicate the value of the created framework, it is important to collect empirical evidence on how agile practices actually affect the software development process from the communication perspective. In this research it is indicated that most of the agile practices used in the projects had positive effects on the communication inside the development teams. For example, sprint planning, open office space, collective code ownership and daily meetings were suggested as efficient practices to improve communication related to requirements, features and project tasks in agile software development teams. In the situations in which these practices were used together, it was apparent that increased informal communication decreases the need for documentation in software development teams and, therefore, facilitates more productive software development than in previous plan driven situations. The study confirms that the use of agile practices has also some positive effects on the external communication and facilitates dependencies between the tasks-and subtasks as well as features and requirements. This occurred especially in the communication between the software development teams and stakeholders. Communication hurdles, however, might still be encountered in the communication between the agile software development team and its stakeholders.

7.2 Limitations of the Thesis

This research was implemented as a series of case studies to understand agile practice adoption and improvement in software intensive organizations. Owing to the confidential nature of the data and the extended periods of data collection, the research team could not rely on more objective constructs to observe process or process changes. The research team was also constrained by access to a few key informants in each organization who were managers or developers. Thus, it was only possible to triangulate across different observations of the same data point (interviews at different time points) and across other published materials and the researchers and research team's own observations. Since only interim, "snap shot" information of software development projects has been collected and analyzed, it is not possible to understand the factors involved in improvements of software development processes from a long term process perspective.

In addition, due to the lack of a reference model for agile practices (e.g. a standard), the mappings presented in this thesis were done based on the researcher team's current knowledge gained from the literature and personal experience. Thus, the mappings presented are also subjective and context-specific. Furthermore, the author's role as a lead assessor in the case companies 1–3 can be considered as a factor of bias in this research. However, the case study research method was considered a suitable and practical method in rapidly changing software development organizations.

The concepts used in this thesis and their interpretation need further evaluation and extension to make them more adequate for the study of the improvement of the software development process mediated with CMMI goals and agile practices. One possible avenue for further research is to examine agile practices beyond those covered in this study i.e. XP and Scrum. Methods such as the LSD, FDD, APM, Crystal and the Adaptive Software Development are all methods that could be assessed.

The study was also based on the first edition of the XP book. This was done for two reasons. Firstly, because the first version of the XP book is empirically evaluated in many research studies and secondly because there is a lack of research available on the combined use of XP and Scrum practices. This is the situation even if the use of a customized set of practices of these two methods seems to be the increasing trend in software companies. In the future, the study could, however, be extended to cover XP practices from the new version of XP (Beck and Andres 2004).

The proposed approach, as implemented in the pilot scheme, is also limited to the predefined three process areas and was implemented only in four case organizations. However, the empirical evidence from these organizations suggests that the method could be also valuable for other software development contexts in different software development organizations. Four case studies is too small a number to validate or refute the framework. Therefore, a larger number of rich case details and case studies would be necessary for further evaluation of the framework.

7.3 Future Research

First of all, the lightweight assessment approach presented in this doctoral thesis is a good starting point for work towards a combination of the CMMI process areas and agile practices. Future research can go down several routes.

Firstly, the research can continue with other process areas of CMMI. For example, requirements development, technical solution, product integration, validation and verification are process areas that would be valuable to map with XP practices, in several in-depth case studies or action research studies.

Secondly, one possible avenue for further research is to examine agile method practices beyond those covered in this study i.e. XP and Scrum. Methods such as LSD, FDD, APM, Crystal and ASD are all methods that could be assessed. For example, (Sidky 2007) uses a larger set of agile practices, which could be valuable from the perspective of future research.

Thirdly, research on the adoption of agile practices could be based on a stronger theoretical framework or could continue with a more quantitative approach by means of a large-scale survey. This could be used to determine the levels of the agile methods and agile practice assimilation across the information system development community with results that can be more generalizable than those contained in this research. This could reveal interesting insights, such as which agile methods are most assimilated and why. A future study could also examine the barriers and facilitators affecting this assimilation. Further research could also examine the effectiveness of agile method adoption. This study was descriptive in nature with the objective being to understand the extent of assimilation, but there was no attempt made to correlate the assimilation to effectiveness or success.

Finally, one of the most obvious ways to continue this study would be to further develop, validate and evaluate the presented framework. The validation could be done by empirical case studies or in a survey with large amount of more generalizable research data.

References

Abrahamsson, P. 2002. The Role of Committment in Software Process Improvement. Oulu Doctoral Thesis. Oulu: University of Oulu. 162 p.

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. 2002. Agile Software Development Methods: Review and Analysis. Espoo. 408. VTT Publications 478. 107 p. http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf.

Agrawal, R. & Chari, K. 2007. Software Effort, Quality and Cycle Time: A Study of CMM Level 5 Projects. IEEE Transactions on Software Engineering, Vol. 33, No. 3, pp. 145–155.

Agile Manifesto, 2001. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. Manifesto for Agile Software Development. http://AgileManifesto.org. Accessed 02.07.2008.

Ambler, S. 2002. Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. New York: John Wiley & Sons Inc. 384 p.

Anacleto, A., von Wangenheim, C. G., Salviano, C. F. & Savi, R. 2004. "Experiences gained from applying ISO/IEC 15504 to small software companies in Brazil", 4th International SPICE Conference on Process Assessment and Improvement, Lisbon, Portugal. Pp. 33–37.

Anderson, D. J. 2003. Agile Management for Software Engineering, Applying the Theory and Constraints for Business Results. Prentice Hall. 0-13-142460-2. 336 p.

Anderson, D. J. 2005. Stretching Agile to Fit CMMI Level 3. Agile Development Conference, Denver.

Automotive SIG, 2007. The SPICE User Group, Automotive SPICE^{TM} Process Reference Model, available from http://www.automotivespice.com.

Baker, S. W. 2006. Formalizing Agility, Part 2: How an Agile Organization Embraced the CMMI. Agile 2006 Conference, Minneapolis, Minnesota.

Baker, S. W. 2005. Formalizing Agility: An Agile Organization's Journey toward CMMI Accreditation. Agile Conference, Denver, Colorado, USA.

Bamberger, J. 1997. Essence of the Capability Maturity Model. Computer, Vol. 30, No. 6, pp. 112–114.

Basili, V. R. 1989. Software Development: A Paradigm for the Future. COMPSAC '89 Conference, Orlando, Florida. Pp. 471–485.

Baskerville, R. & Bries-Heje, J. 2001. A Multible Theory Analysis of Diffusion of Innovation Technology Case. Information Systems Journal, Vol. 11, No. 3, pp. 181–212.

Batista, J. & Figueiredo, A. D. 2000. SPI in a very Small Team: a Case with CMMI. Software Process Improvement and Practice, Vol. 5, No. 4, pp. 243–250.

Beck, K. 1999. Embracing Change with Extreme Programming. IEEE Computer, Vol. 3, No. 10, pp. 70–77.

Beck, K. 2000. Extreme Programming Explained: Embrace Change. Addison-Wesley Longman, Inc. 190 p.

Beck, K. & Andres, C. 2004. Extreme Programming Explained: Emprace change, second edition. Boston: Addison-Wesley.

Boehm, B. 2002. Get Ready For The Agile Methods, With Care. Computer, Vol. 35, No. 1, pp. 64–69.

Boehm, B. 1988. A Spiral Model of Software Development and Enhancement. Computer, Vol. 21, No. 5, pp. 61–72.

Boehm, B. 2003. Value-Based Software Engineering. Computer, Vol. 3, No. 3, pp. 33–41.

Boehm, B. & Turner, D. 2005. Management Challenges to Implement Agile Processes in Traditional Development Organizations. IEEE Software, Vol. 22, No. 5, pp. 30–38.

Boehm, B. & Turner, R. 2003a. Balancing Agility and Discipline. Balancing Agility and Discipline. In: Balancing Agility and Discipline – A Guide for the Perplexed. Addison-Wesley. 0-32-118612-5. 304 p.

Boehm, B. & Turner, R. 2003b. Using Risk to Balance Agile and Plan-Driven Methods. IEEE Computer Society.

Boehm, B. W. & Ross, R. 1989. Theory-W software project management principles and examples. IEEE Transactions on Software Engineering, Vol. 15, No. 7, pp. 902–916.

Bos, E. & Vriens, C. 2004. An agile CMM, 4th Conference on Extreme Programming and Agile Methods – XP/Agile Universe, pp. 129–138.

Brinkkemper, S. 1996. Method Engineering: Engineering of Information Systems. Development Methods and Tools. Information and Software Technology, Vol. 38, No. 4, pp. 275–280.

Clegg, S. R., Waterson, P. E. & Axtell, C. M. 1996. Software Development Knowledge Intensive Work Organizations. Behaviour and Inofmation Technology, Vol. 15, No. 4, pp. 237–249.

CMMI. 2006. Capability Maturity Model® Integration for Development, Version 1.2, Technical Software Engineering Institute. Report CMU/SEI-2006-TR-008. http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html.

Coad, P. & Palmer, S. 2002. Feature-Driven Development. NJ.: Prentice Hall.

Cockburn, A. 2002. Agile Software Development. Boston: Addison-Wesley. 0-201-69969-9 278.

Cockburn, A. & Highsmith, J. 2001. Agile Software Development: The People Factor. Computer, Vol. 34, No. 11, pp. 131–133.

Cohen, D., Lindvall, M. & Costa, P. 2004. An Introduction to Agile Methods. Elsevier Academic Press. 0-12-012162 2-67.

Cohn, M. & Ford, D. 2003. Introducing an Agile Process to an Organization. IEEE Computer, Vol. 36, No. 6, pp. 74–78.

Cooper, R. B. & Zmud, R. W. 1990. Information Technology Implementation Research: A Technological Diffusion Approach. Management Science, Vol. 36, No. 2, pp. 123–139.

Coram, M. & Bohner, S. 2005. The Impact of Agile Methods on Software Project Management. 12th IEEE International Conference and Workshops on Engineering of Computer based Systems Conference, Potsdam, Germany.

Crowston, K. & Kammerer, E. 1998. Coordination and Collective Mind in Software Requirements Development. IBM Systems Journal, Vol. 37, No. 2, pp. 227–245.

Curtis, P., Phillips, D. M. & Weszka, J. 2001. CMMI – The Evolution Continues. System Engineering, Vol. 5, No. 1, pp. 7–18.

Damian, D., Eberlein, A., Shaw, M. L. & Gaines, B. R. 2000. Using Different Communication Media in Requirements Negotiation. IEEE Software, Vol. 17, No. 3, pp. 28–36.

Dangle, K. C., Larsen, P. & Zelkowitz, M. V. 2005. Software Process Improvement in Small Organizations: A Case Study. IEEE Software, Vol. 22, No. 6, pp. 68–75.

Daskalantona, M. K. 1994. Achieving Higher SEI Levels. IEEE Software, Vol. 11, No. 4, pp. 17–24.

Davis, F. D. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly, Vol. 13, No. 3, pp. 319–339.

DeMarco, T. & Boehm, B. 2002. The Agile Methods Fray. IEEE Computer, Vol. 31, No. 6, pp. 90–92.

Deming, W. E. 1990. Out of the Crisis. Massachusetts Institute of Technology, Center of Advanced Engineering Study. Cambridge. 0-26-254116-5. Second edition. 207 p.

Drobka, J., Noftz, D. & Raghu, R. 2004. Piloting XP on Four Mission Critical Projects. IEEE Software, Vol. 21, No. 6, pp. 70–75.

Dybå, T. & Dingsøyr, T. 2008. Empirical Studies of Agile Software Development: A Systematic Review. Information and Software Technology 10.1016/j.infsof.2008.01.006.

Eman, K. E. & Madhavji, N. H. 1999. Elements of Software Process Assessment and Improvement. IEEE Press.

Erickson, J. & Lyytinen, K. 2005. Agile Modelling, Agile Software Development, and Extreme Programming: The State of Research. Journal of Database Management, Vol. 16, No. 4, pp. 88–100.

Fayad, M. & Laitinen, M. 1997. Process Assessment Considered Wasteful. Communications of the ACM, Vol. 40, No. 11, pp. 125–128.

Fichman, R. G. 2001. The Role of Aggregation in the Measurement of IT-related Organizational Innovation. MIS Quarterly, Vol. 25, No. 4, pp. 427–455.

Fisher, C. 2007. Researching and writing a dissertation a guidebook for business students. Prentice Hall.

Fitzgerald, B. 1996. Formalized Systems Development Methodologies: a critical perspective. Information Systems Journal, No. 6, pp. 3–23.

Fitzgerald, B., Hartnett, G. & Conboy, K. 2006. Customising Agile Methods to Software Practices at Intel Shannon. European Journal of Information Systems, Vol. 15, No. 2, pp. 200–213.

Fitzgerald, B., Russo, N. L. & Stolterman, E. 2002. Information Systems Development – Methods in Action. McGraw-Hill Education.

Fritzsche, M. & Keil, P. 2007. Agile Methods and CMMI: Compatibility or Conflict? Software Engineering Journal, Vol. 1, No. 1, pp. 9–26.

Galin, D. & Avrahami, M. 2006. Are CMM Program Investment Beneficial? Analysing Past Studies. IEEE Software, Vol. 23, No. 6, pp. 81–87.

Gallivan, M. 2001. Organizational Adoption and Assimilation of Complex Technological Innovations: Development and Application of a New Framework. The DATA BASE for Advances in Information Systems, Vol. 32, No. 3, pp. 51–85.

Glazer, H. 2001. Dispelling the Process Myth: Having a Process Does Not Mean Sacrificing Agility or Creativity. CrossTalk. The Journal of Defense Software Engineering, pp. 27–30.

Grenning, J. 2001. Launching XP at a Process-Intensive Company. IEEE Software, Vol. 18, No. 6, pp. 3–9.

Hareton, K., Leung, N. & Terence, C. F. 2001. A process framework for small projects. Software Process Improvement and Practice, Vol. 6, No. 2, pp. 67–83.

Herbsleb, J., Carleton, A., Rozum, J. & Siegel, D. 1994. Benefits of CMM-based software process improvement: Initial results. CMS/SEI-94-TR-013. Pittsburgh: Carnegie Mellon University.

Henttonen, K. & Blomqvist K. 2005. Managing Distance in a Global Virtual Team: The Evolution of Trust Trough Technology-Mediated Relational Communication. Strategic Change, Vol. 14, No. 2, pp. 107–119.

Highsmith, J. 2004. Agile Project Management, Creating innovative products. Addison-Wesley.

Highsmith, J. 2002a. Agile Software Development Ecosystems. Addison-Wesley. Boston. 0201760436. 448 p.

Highsmith, J. 2002b. What Is Agile Software Development? Crosstalk, pp. 4–9.

Horvat, R. V., Rozman, I. & Györkös, J. 2000. Managing the Complexity of SPI in Small Companies. Software Process Improvement and Practice, Vol. 5, No. 1, pp. 45–54.

Hovorka, D. S. & Larsen, K. R. 2006. Enabling Agile Adoption Practices through Network Organizations. European Journal of Information Systems, Vol. 15, No. 2, pp. 169–168.

Hulkko, H. & Abrahamsson, P. 2005. A Multiple Case Study on the Impact of Pair Programming on Product Quality. International Conference on Software Engineering. ICSE Conference, Louis, Missouri, USA.

Humphrey, W. S. 1995. A Discipline for Software Engineering. Addison-Wesley, Longman, Inc. 0-201-54610-8 816.

Humphrey, W. S., Snyder, T. R. & Willis, R. W. 1991. Software Process Improvement at Hughes Aircraft. IEEE Software, Vol. 8, No. 4, pp. 11–23.

ISO. 2006. (SPICE) ISO TR 15504. Part 5. Information technology – Software process assessment – Part 5: An exemplar Process Assessment Model, JTC 1/SC 7. ISO TR 15504. Geneva: International organisation of standardisation.

Jeffries, R. 2002. Extreme Programming and the Capability Maturity Model http://www.xprogramming.com/xpmag/xp_and_cmm.htm.

Karlsson, E.-A., Anderson, L.-G. & Leion, P. 2000. Daily Build and Feature Development in Large Distributed Projects. Conference on Software Engineering (ICSE 2000), Limerick, Ireland. Pp. 649–658.

Karlström, D. & Runeson, P. 2006. Integrating agile software development into stage-gate managed product development. Empirical Software Engineering, Vol. 11, No. 2, pp. 203–225.

Kautz, K. 1998. Software Process Improvement in Very Small Enterprises: Does it Pay Off. Software Process Improvement and Practice, Vol. 4, No. 4, pp. 209–226.

Kitchenham, B. A., Pfleeger, S. L, Pickard, L. M., Jones, P. W., Hoaglin, D. C., Eman, K. E. & Rosenberg, J. 2002. Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering, Vol. 28, No. 8, pp. 721–734.

Klein, H. K. & Myers, M. D. 1999. A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. MIS Quartely, Vol. 23, No. 1, pp. 67–94.

Korkala, M., Abrahamsson, P. & Kyllönen, P. 2006. A Case Study on the Impact of Customer Communication on Defects in Agile Software Development. Agile 2006 Conference. Pp. 76–88.

Koskela, J. & Abrahamsson, P. 2004. On-Site Customer in an XP Project: Empirical Results from a Case Study. EuroSPI 2004 Conference. Pp. 1–11.

Kähkönen, T. & Abrahamsson, P. 2004. Achieving CMMI Level 2 with Enhanced Extreme Programming Approach. Profes Conference, Japan.

Laitinen, M. & Fayad, M. 1998. Surviving a process performance crash. Communications of the ACM, Vol. 41, No. 2, pp. 83–86.

Larman, C. 2003. Agile & Iterative Software Development. Addison-Wesley. Boston. 340 p.

Larman, C. & Basili, V. R. 2003. Iterative and Incremental Development: A Brief History. IEEE Computer, Vol. 36, No. 6, pp. 47–56.

Layman, L., Williams, L. & Cunningham, L. 2006a. Motivations and Measurements in an Agile Case Study. Journal of Systems Architecture, Vol. 52, No. 11, pp. 654–667.

Layman, L., Williams, L., Damian, A. & Bures, H. 2006b. Essential Communication Practices for Extreme Programming in a Global Software Development Team. Information and Software Technology, Vol. 48, No. 9, pp. 781–794.

Leon, G. 1995. On the diffusion of software technologies: technological frameworks and adoption profiles. The Diffusion and Adoption of Information Technology Conference, Oslo. Pp. 97–116.

Lindvall, M., Muthig, D., Dasnino, C. W., Stupperich, M., Kiefer, D. & Kähkönen, T. 2004. Agile Software Development in Large Organizations. Computing Practices, Vol. 37, No. 12, pp. 38–46.

Malone, T. & Crowston, K. 1994. The Interdisciplinary Study of Coordination. ACM Computing Surveys, Vol. 26, No. 1, pp. 87–119.

Manhart, P. & Schneider, K. 2004. Breaking the Ice for Agile Development of Embedded Software: An Industry Experience report. 26th International Conference of Software Engineering Conference, Washington, DC, USA.

Mann, C. & Maurer, F. 2005. A case study on the Impact of Scrum on Overtime and Customer Satisfaction. Agile 2005 Conference, Denver.

Mathianssen, L., Pries-Heje, J. & Ngwenyama, O. 2002. Improving Software Orgnanizations, From Principle to Practice. Addison-Wesley.

McCaffery, F., Richardson, I. & Coleman, G. 2006. Adept – A Software Process Appraisal Method for Small to Medium-sized Irish Software Development Organisations. EuroSPI06 Conference, Finland, Joensuu.

McCaffery, F., Taylor, P. & Coleman, G. 2007. Adept: A Unified Assessment Method for Small Software Companies. IEEE Software, Vol. 24, No. 1, pp. 24–31.

McFeeley B. 1996. A Users Guide for Software Process Improvement. Pittsburgh: Carnegie Mellon University.

Meehan, B. & Richardson, I. 2002. Identification of Software Process Knowledge Management. Software Process: Improvement and Practice, Vol. 7, No. 2, pp. 47–56.

Miles, M. & Huberman, A. 1999. Qualitative Data Analysis. London: Sage.

Moore, R., Reff, K., Graham, J. & Hackerson, B. 2007. Scrum at a Fortune 500 Manufacturing Company. Agile 2007 Conference, Washington D.C.

Morkel, W. H., Kourie, D. G. & Watson, B. W. 2003. Standards and Agile Software Development. SAICSIT 2003 Conference. Pp. 178–188.

Nawrocki, J., Jasinski, M., Walter, B. & Wojciechowski, A. 2002. Extreme Programming Modified: Embrace Requirements Engineering Practices. International Conference of Requirements Engineering, Essen, Germany.

Niazi, M., Wilson, D. & Zowghi, D. 2003. A maturity model for the implementation of software process improvement: an empirical study. The Journal of Systems and Software, Vol. 74, No. 2, pp. 155–172.

Nielsen, P. A. & Pries-Heje, J. 2002. A Framework for Selecting an Assessment Strategy. A Framework for Selecting an Assessment Strategy. In: Improving Software Organizations – from principle to practice, Addison-Wesley. Pp. 185–198.

Oppenheim, A. N. 1992. Questionnaire Design, Interviewing and Attitude Measurement. Questionnaire Design, Interviewing and Attitude Measurement. New York.

Paasivaara, M. & Lassenius, G. 2003. Collaboration in Inter-organizational Software Development. Software Process Improvement and Practice, Vol. 8, No. 4, pp. 183–199.

Packlick, J. 2007. The Agile Maturity Map. A Goal Oriented Approach to Agile Improvement. Agile 2007 Conference, Washington D.C.

Paetch, F., Eberlein, A. & Maurer, F. 2003. Requirements Engineering and Agile Software Development. 12th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprices Conference. Pp. 308–313. 0-7695-1963-6.

Paulk, M. 2002. Agile Methodologies and Process Discipline. CrossTalk The Journal of Defense Software Engineering, pp. 15–18.

Paulk, M. 1999. Analyzing the Conceptual Relationship Between ISO/IEC 15504 (Software Process Assessment) and the Capability Maturity Model for Software. International Conference on Software Quality, Austin, Texas, USA.

Paulk, M. C. 2001. Extreme Programming from a CMM Perspective. Software, Vol. 18, No. 6, pp. 19–26.

Poppendieck, M. 2001. Lean Programming. Software Development Magazine, No. 2, pp. 71–75.

Ramachandran, M. 2005. A Process Improvement Framework for XP Based SMEs. XP Conference, Sheffield, UK. Pp. 202–205.

Rasmusson, J. 2003. Introducing XP into Greenfield Projects: Lessons Learned. IEEE Software Vol. 20, No. 3, pp. 21–28.

Reifer, D. 2003. XP and the CMM. IEEE Software, Vol. 20, No. 3, pp. 14–15.

Richardson, I. 2001. Software Process Matrix: A Small Company SPI Model. Software Process Improvement and Practice, Vol. 6, No. 3, pp. 157–165.

Rising, L. & Janoff, N. S. 2000. The Scrum software development process for small teams. IEEE Software, Vol. 17, No. 4, pp. 26–32.

Rogers, E. M. 2003. Diffusion of Innovations. New York: The Free Press. Fifth Edition.

Royce, W. 1970. Managing the Development of Large Software Systems. IEEE WESCON Conference. Pp. 1–9.

Rubin, H. & Rubin, I. 2005. Qualitative Interviewing: The Art of Hearing Data. Thousand Oaks, CA: Sage.

Salo, O. & Abrahamsson, P. 2007. An Iterative Improvement Approach for Agile Development: Implications from multiple case study. Software Process: Improvement and Practice, Vol. 12, No. 1, pp. 81–100.

Salo, O. & Abrahamsson, P. 2008. Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum, Software, IET, Vol. 2, No. 1, pp. 58–64.

Schatz, B. & Abdelshafi, I. 2005. Primavera Gets Agile: A Successfull Transition to Agile Development. IEEE Software, Vol. 22, No. 3, pp. 36–42.

Schwaber, K. 2003. Agile Project Management with Scrum. Microsoft Press. Washington.

Schwaber, K. & Beedle, M. 2002. Agile Software Development with Scrum. Prentice-Hall. Upper Saddle River, NJ. 0-13-067634-9 158.

Schwarz, A., Mehta, M., Johnson, N. & Chin, W. W. 2007. Understanding Frameworks and Reviews: A Commentary to Assist us in Moving Our Field Forward by Analyzing Our Past. The DATA BASE for Advances in Information Systems, Vol. 38, No. 3, pp. 29–50.

SEI. 2006. Standard CMMI®Appraisal Method for Process Improvement (SCAMPISM) A, Version 1.2: Method Definition Document. CMU/SEI-2006-HB-002. http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06hb002.pdf, available 020108.

Sfetsos, P., Angelis, L. & Stamelos, I. 2006. Investigating the extreme programming system. An empirical study. Empirical Software Engineering, Vol. 11, No. 2, pp. 269–301.

Sidky, A. 2007. A Structured Approach to Adopting Agile Practices: The Agile Adoption Framework. Doctoral Thesis. Virginia Polytechnic Institute and State University. 236 p.

Siniaalto, M. & Abrahamsson, P. 2007. Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage. International Symposium on Empirical Software Engineering and Measurement Conference, Madrid, Spain.

Stelzer, D. & Mellis, W. 1998. Success Factors of Organizational Change in Software Process Improvement. Software Process Improvement and Practice, Vol. 4, No. 4, pp. 227–250.

Sutherland, J. 2001. Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies. Cutter IT Journal, Vol. 14, No. 12, pp. 5–11.

Sutherland, J., Jakobsen, C. R. & Johnson, K. 2007. Scrum and CMMI Level 5: The Magic Potion for Code Warriors. Agile 2007 Conference, Washington D.C.

Svensson, H. & Höst, M. 2005. Introducing an Agile Process in a Software Maintenance and Evolution Organization. 9th European Conference of Maintenance and Reengineering Manchester, UK.

Tolvanen, J. P. 1998. Incremental Method Engineering with Modelling tools. Doctoral Thesis. University of Jyväskylä. 301 p.

Trudel, S., Lavoie, J. M, Paré, M. C. & Suryn, W. 2006. The small company-dedicated software process quality evaluation method combining CMMI and ISO/IEC 14598. Software Quality Journal, Vol. 4, No. 1, pp. 7–23.

Turner, R. & Jain, A. 2002. Agile Meets CMMI: Culture Clash or Common Cause. 1st Agile Universe Conference, Chicago.

Walsham, G. 1995. Interpretive Case Studies in IS Research: Nature and Method. European Journal of Information Systems, Vol. 4, No. 1, pp. 74–81.

Wang, X. Oconchuir, E. & Vidgen, R. 2008. A paradoxical Perspective on Contradictions in Agile Software Development. European Conference of Information Systems (ECIS) 2008. Galway, Ireland.

Weick, K. 1995. What Theory Is Not, Theorising Is. Quarterly, Vol. 40, No. 1, pp. 385–390.

Wengraft, T. 2001. Qualitative Research Interviewing. London: Sage.

Wilkie, F. G. & McCaffery, F. 2005. Evaluation of CMMI Process Areas for Small to Medium-sized Software Development Organizations. Software Process Improvement and Practice, Vol. 10, No. 2, pp. 189–202.

Williams, L. & Cockburn, A. 2003. Agile Software Development It is about Feedback and Change. Computer, Vol. 36, No. 6, pp. 39–42.

Vriens, C. 2003. Certifying for CMM Level 2 and ISO9001 with XP@Scrum. Agile Development Conference, Salt Lake City, Utah, USA.

Yin, R. K. 1994/2003. Case Study Research Design and Methods. Saga Publications. Thousand Oaks, California. Second Edition. 192 p. 076192552X

Yin, R. K. 2003. Case Study Research: Design and Methods. Thousand Oaks, California.

Appendix 2: Publications 1–VI of this publications are not included in the PDF version. Please order the printed version to get the complete publication (http://www.vtt.fi/publications/index.jsp)

Appendix 1: Mapping Model

Appendix 1 shows the mapping model which was created based on the literature review and the four described case studies 1–4.

CMMI specific goal	**Scrum Practices**	**XP Practices**
Manage requirements	Requirements continuous analysis for product and sprint backlogs, requirements analysis for 4 week iterations in sprint planning, sprint reviews (demo presentation after the review), daily meetings, Self-organizing teams	Analysis of requirements for one- to two-week iterations, user stories, on-site customer, daily stand up meetings, continuous integration, small tasks and estimations
Establish estimates	Sprint planning, tasks and effort estimations for 4-week iterations, self-organizing teams	Planning games, stories, story boards, tasks and effort estimations for one- to two-week iterations, task and effort descriptions on the wall
Develop a Project Plan	Sprint planning, sprints, product backlog, sprint backlog, daily meetings, tasks and effort estimations for 4-week iterations, self-organizing teams	Planning game, small releases, story board, small tasks and effort estimations for one- to two-week iterations, task descriptions in the wall
Obtain commitment to the Plan	Sprint planning, sprint review; self-organizing teams	On-site customer, planning game
Monitor project against the plan	Sprint planning, sprints, sprint review, daily scrum meeting, scrum metrics, reflection workshops	Planning game, small releases, daily stand up meetings, on-site customer

Series title, number and report code of publication

VTT Publications 695
VTT-PUBS-695

Author(s)
Pikkarainen, Minna

Title

Towards a Framework for Improving Software Development Process Mediated with CMMI Goals and Agile Practices

Abstract

Problems in software development mainly spring from the difficulty of establishing and stabilizing the requirements, the changeability of the software and interactive dependency of the software, hardware and human beings. A software development process consists of a set of empirical and ‘best’ practices in software development, together with organization and management that are needed for the software product implementation. Different process models, such as CMMI (Capability Maturity Model Integration), ISO 9001 and ISO 15504, have been developed in the last decade to support the assessment of software development processes. The main process model, examined in this thesis, is CMMI. This model was chosen as the focus of this research because it is a widely-used, beneficial approach for identifying the key weaknesses of a software development process which need immediate attention and improvement. Two of the key challenges of CMMI assessments are 1) overly heavy and time-consuming assessments and 2) the risk that the achievement of CMMI levels forces the developers to use more time writing documents than implementing the software product.

The level of interest in the use of agile practices (focusing on practices such as eXtreme Programming and Scrum) has radically increased in software organizations. Practitioners argue that the adoption of agile software development methods can solve the organizational need for a more rapid and flexible software development process, and enable improved communication in changing market situations. A brief analysis of the empirical body of knowledge reveals, however, that there are also several challenges in interactive dependency management and communication between the actors of software development in an agile context.

The objective of this study is to increase the understanding of how improvements can be made in the software development processes from communication perspective, mediated with CMMI ‘specific’ goals and agile practices. This study is based on a series of case studies and data from 4 companies and 8 software development teams. To meet the gaps in the current empirical body of knowledge and research, a novel framework is presented in this study. The framework can be used 1) to identify the agile practices for a plan-driven software development process and 2) to assess the software development process in a lightweight manner against the CMMI goals and agile practices.

To indicate the value of the created framework, it is important to collect empirical evidence on how agile practices actually affect communication in the software development process. This study applies coordination theory to confirm that the adoption of agile practices, such as sprint planning, an open office space, daily meetings and product backlogs improve the communication and management of requirements, features and project task dependencies in agile software development teams. Additionally, increased informal communication can in some cases decrease the need for upfront documentation in software development teams and, therefore, facilitate more productive software development than in previous plan driven situations.

ISBN
978-951-38-7121-5 (soft back ed.)
978-951-38-7122-2 (URL: http://www.vtt.fi/publications/index.jsp)

Series title and ISSN	Project number
VTT Publications 1235-0621 (soft back ed.) 1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)	23490

Date	Language	Pages
October 2008	English	119 p. + app. 193 p.

Keywords	Publisher
CMMI, agile practices, lightweight assessment, communication	VTT Technical Research Centre of Finland P.O. Box 1000, FI-02044 VTT, Finland Phone internat. +358 20 722 4520 Fax +358 20 722 4374

Organizational maturity indicators, such as the CMMI levels or SPICE ratings, have become important for software development. Customer organizations often rely on them when selecting a supplier, as the results of these assessments can serve as an indicator of process maturity. At the same time, agile methods continue to gain popularity due to increasing speed and quality demands. It has been argued that the CMMI model is too heavy-weight for software development projects adopting agile practices and that its use would lead to an overly document-driven software development approach. This presents a challenge to enable organizations, relying on CMMI as an indicator of process maturity, to also benefit from using agile methodologies such as XP and Scrum. The purpose of this thesis is to increase understanding of how to improve the software development process mediated with the CMMI and the agile practices. The work was done empirically in 4 companies and based on 6 scientific research papers, written jointly with an international group of researchers and published in well-established peer-reviewed scientific fora.

In order to answer the gaps in the current empirical body of knowledge and research this study introduces a framework, based on a hybrid assessment approach, and starts the evaluation of the impact of agile practices from the communication perspective. The framework can be used to identify the agile practices for a plan-driven software development process and to validate the software development process against CMMI goals and agile practices.

Julkaisu on saatavana	Publikationen distribueras av	This publication is available from
VTT	VTT	VTT
PL 1000	PB 1000	P.O. Box 1000
02044 VTT	02044 VTT	FI-02044 VTT, Finland
Puh. 020 722 4520	Tel. 020 722 4520	Phone internat. + 358 20 722 4520
http://www.vtt.fi	http://www.vtt.fi	http://www.vtt.fi

ISBN 978-951-38-7121-5 (soft back ed.) ISBN 978-951-38-7122-2 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1235-0621 (soft back ed.) ISSN 1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)