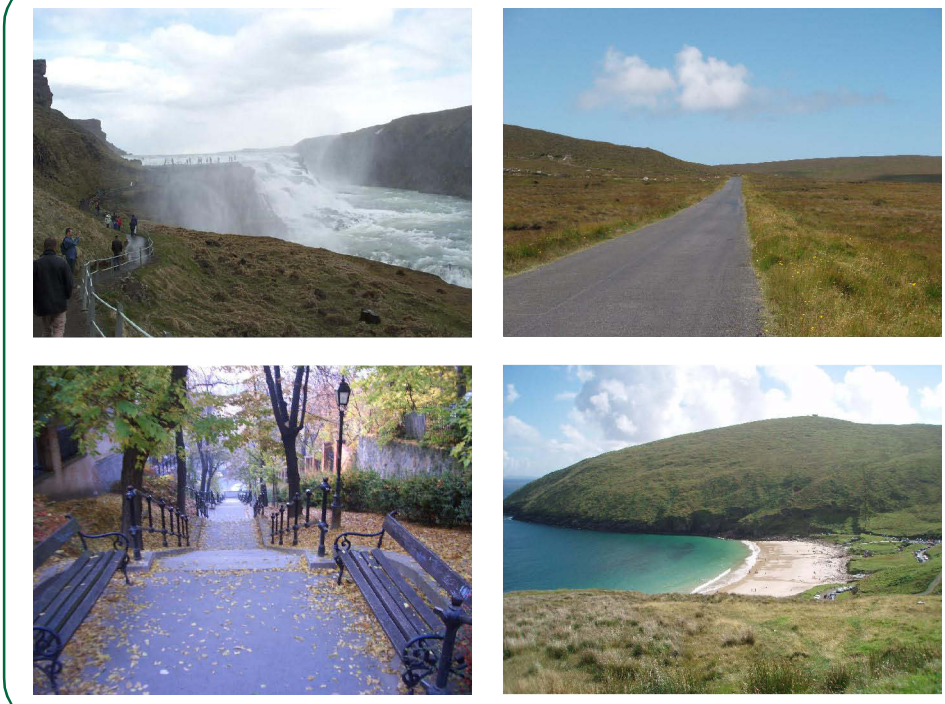
ESPOO 2008 VTT PUBLICATIONS 695



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)

Copyright © VTT Technical Research Centre of Finland 2008

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 difficult 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

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)

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.

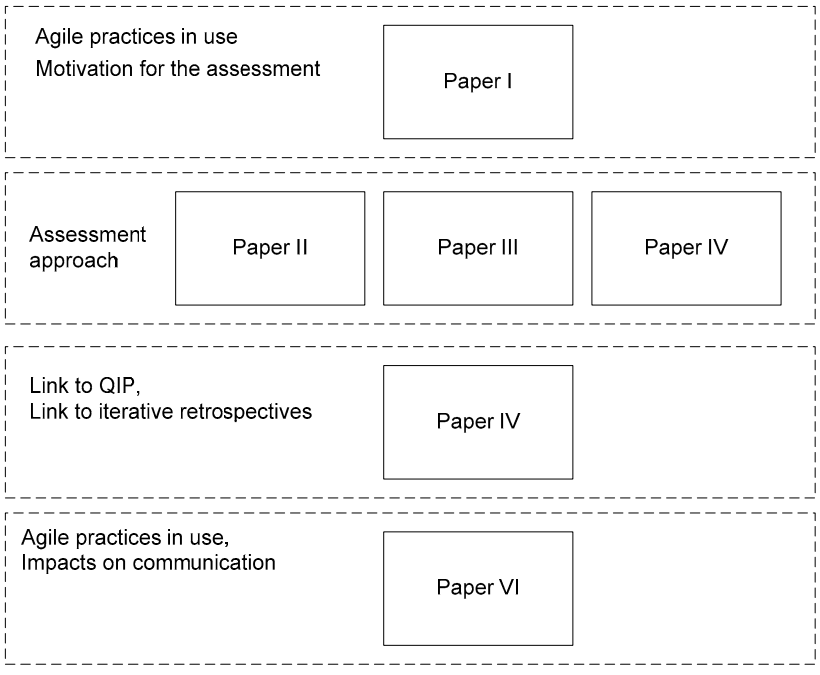


Figure 1. Different aspects of the research phenomenon.

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).

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
VTT
PL 1000
02044 VTT
Puh. 020 722 4520
http://www.vtt.fi

Publikationen distribueras av
VTT
PB 1000
02044 VTT
Tel. 020 722 4520
http://www.vtt.fi

This publication is available from
VTT
P.O. Box 1000
FI-02044 VTT, Finland
Phone internat. + 358 20 722 4520
http://www.vtt.fi

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)