Title	Agile Practices in Use from an Innovation Assimilation Perspective: a Multiple Case Study
Author(s)	Wang, Xiaofeng; Conboy, Kieran
Publication Date	2007
Publication Information	Pikkarainen, M., X. Wang and K. Conboy (2007): "Agile Practices in Use from an Innovation Assimilation Perspective: a Multiple Case Study", in the proceedings of the 28th International Conference on Information Systems (ICIS 2007), Montreal, Dec 10-12, 2007.
Item record	http://hdl.handle.net/10379/1607

AGILE PRACTICES IN USE FROM AN INNOVATION ASSIMILATION PERSPECTIVE: A MULTIPLE CASE STUDY

Minna Pikkarainen
Lero – The Irish Software Engineering Research Centre
Limerick, Ireland
minna.pikkarainen@lero.ie

Xiaofeng Wang
Lero – The Irish Software Engineering Research Centre
Limerick, Ireland
xiaofeng.wang@lero.ie

Kieran Conboy
National University of Ireland, Galway
Galway, Ireland
(kieran.conboy@nuigalway.ie)

Abstract

Agile methods have been adopted by many information systems development (ISD) teams and organizations in recent years. However, while agile method research is growing, many studies lack a strong theoretical and conceptual base. Innovation adoption theories provide new perspectives on analysing agile methods. This paper is based on an exploratory study of the application of innovation theory to agile practices in use, focusing in particular on the later stages of assimilation i.e. acceptance, routinization and infusion. Three case studies were conducted involving agile method projects, using semi-structured interviews. One key finding is that specific needs of the adopting teams may drive the relevant agile practices in use to a deeper level of assimilation. Another key finding indicates the period of agile use does not have a proportional effect on their assimilation stages. Therefore, one needs to be cautious when using time as a measure of agile practice assimilation.

Keywords: agile, method, systems development, practice in use, innovation adoption, assimilation stages, routinization, infusion, extreme programming, Scrum

Introduction

Agile methods represent quite a popular initiative which complements previous critiques of formalised methods (e.g. Baskerville et al., 1992), and have been well received by practitioners and academics. There is also evidence to suggest that use of agile methods has been growing rapidly since their inception (Boehm, 2002, Boehm and Turner, 2004). A number of methods are included in this family, the most notable being *eXtreme Programming (XP)* (Beck, 2000), *Scrum* (Schwaber and Beedle, 2002), the *Dynamic Systems Development Method (DSDM)* (Stapleton, 1997), *Crystal* (Cockburn, 2001), *Agile Modelling* (Ambler, 2002), *Agile Project Management (APM)* (Highsmith, 2004), *Feature Driven Design* (Coad and Palmer, 2002), and *Lean Software Development (LSD)* (Poppendieck, 2001). While the emergence of agile methods has been primarily industry and not research led, agile methods are the subject of a rapidly growing body of research activity. In addition, a large contingent of these empirical studies focus on agile methods "in action" (Fitzgerald 1997) or agile practices used in real-world contexts (Rasmusson 2003; Grenning 2001; Murru et al. 2003; Lippert et al. 2003; Fitzgerald et al. 2006). Few studies of agile methods in use, however, are based on a strong theoretical and conceptual foundation This trend is particularly symptomatic of agile method tailoring research- many such studies exist, providing a descriptive account of how an textbook agile method was implemented and modified, but there is little cumulative tradition where each author compares and contrasts their account against other previous tailoring studies. Therefore there is significant benefit to be gleaned from analysing agile method use through the lens of existing well-established theory - innovation adoption theory being one such example

According to Rogers (1983), an innovation can be an idea or practice which is perceived as new by adopters. Based on this definition, agile practices can certainly be characterized as software process innovations. The vast majority of the proprietary agile method literature portrays these methods as new, revolutionary and innovative. The agile manifesto is highly illustrative of this, setting out key values and principles of agile methods, and in essence drawing a firm distinguishing line between these methods and the heavyweight, bureaucratic methods that went before. Theories on IT innovation adoption, consequently, may bring new insights in the study of agile practices in use. Several innovation adoption theories have been built and used to explain the mechanisms and constructs of the introduction and implementation of new innovations (Davis 1989; Cooper and Zmud 1990; Saga and Zmud 1994; Fichman 1999; Gallivan 2001). Gallivan (2001)'s work is considered particularly relevant to this study, in which six assimilation stages have been proposed based on the previous work of Cooper and Zmud (1990) and Saga and Zmud (1994). The later three stages of assimilation have particular relevance for the study of agile practices in use, i.e., acceptance, routinization and infusion. The investigation of the adoption processes of agile practices is critical, following the line of argument of Hovorka and Larsen (2006), because it provides another level of explanation, describing the importance of the agile practices in a system development and adoption setting.

The goal of this study is to explore the application of innovation adoption theories in agile research, and to obtain a better understanding of agile practices in use by so doing. The agile practices investigated in this study are from eXtreme Programming (XP) (Beck, 2000) and Scrum (Schwaber and Beedle 2002). These methods were chosen for a number of reasons, the foremost being the fact that they are by far the most widely used of the agile method family (Fitzgerald et al. 2006). Secondly, they are very diverse approaches as (XP) is very prescriptive and practitioner-oriented while Scrum is primarily a project management method (Abrahamsson et al. 2002). By studying these two methods, this piece of research ensures that lessons learned considered both perspectives. Finally, existing research has shown that XP and Scrum are often combined by teams and organisations (Fitzgerald et al. 2006), and observing this phenomenon is further motivation underpinning the choice of methods.

To achieve the goal of this study the remainder of the paper is organized as follow. The next section describes the conceptual background, starting with a description of agile practices, and a review of innovation adoption theory. An interpretation of the theories is also presented in terms of agile practices in use, and is followed by the proposed framework adopted in this research. The research approach is then justified and outlined, and this is followed by a description of the three cases. The use of the agile practices is described and the key findings are then presented. These are then discussed in light of the existing literature. The last section concludes with final remarks, limitations and suggestions for future work.

Conceptual Background

Agile Practices: Methods versus Method-in-Action

There are many books, journals and articles explaining the various agile methods in existence. Table 1 lists the practices associated with XP and Scrum, the two methods which 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 'vanilla' version of the method and the method actually enacted in practice. Fitzgerald et al (1997) refer to the latter as the "method-in-action". Prescribed practices are constantly tailored to suit the specific needs of teams and human nature inevitably leads to diverse interpretations and implementations of a method. However, empirical research of agile practices used in real-world settings is relatively sparse, compared with laboratory-style experiments, controlled case studies or anecdotal experience reports (for some examples of empirical studies on XP in use, see Grenning (2001), Murru et al. (2003), Rasmusson (2003), Williams et al. (2003), Drobka et al. (2004) and Svensson and Host (2005); for those on Scrum in use, see Rising and Janoff 2000 and Dingsoyr et al. 2006; Fitzgerald et al. (2006) study the combined use of XP and Scrum). Few studies have attempted to understand agile practices in use with the help of appropriate theoretical lenses. Consequently, systematic and insightful understanding of agile practices in use is yet to be achieved.

Table 1: A Combined List of XP and Scrum Practices (adapted from Beck, 2000 and Schwaber & Beedle, 2002)

XP Practices	Scrum Practices	Explanation
Pair Programming		Code is written by two programmers on the same machine.
Testing		Continually write tests, which must run flawlessly for development to proceed. Write test code before writing function code.
Metaphor		Guide all development with a simple shared story of how the system works
Collective Ownership		Anyone can change any code anywhere in the system at any time.
Refactoring		Programmers restructure the system, without removing functionality, to improve code, performance, simplicity, and flexibility.
Coding Standards		Adherence to coding rules which will facilitate communication through code.
Simple Design		The design of the system should be as simple as possible.
Continuous Integration		Integrate and build the system every time a task is completed - this may be many times per day.
40-Hour Week		Work time is generally limited to 40 hours per week.
On-Site Customer		Include an actual user on the team, available full-time to answer questions.
Small Releases	Sprints	Put a simple system into production quickly, then release new versions on a very short cycle.
Planning Game	Sprint Planning	Prioritisation of scope for next release based on a combination of business priorities and technical estimates.
	Architecture	System architecture modification and high-level design regarding implementation of backlog items.
	Post Game Sessions	Reflect on method strengths and weaknesses after eachg cycle.
	Daily Meetings	Short daily status meeting.

Innovation Adoption Theory

Several innovation adoption studies represent the core theories, models and frameworks in this area, including Davis (1989), Fichman (1992, 1994, 2001) and Gallivan (2001). They build innovation adoption models, and measure the

adoption stage of innovations in various environments. While many of these exist, and have many commonalities, they are based on many different levels of analysis, such as individual, group and organisation.

Davis (1989)'s Technology Acceptance Model (TAM) focuses on evaluating usefulness, use and user acceptance aspects on adoption of information technologies. A conclusion of the study is that users adopt technologies primarily based on the functions they perform and secondarily on how hard or easy to use the technologies. Fichman (1994) develops measurement scales for innovation adoption using the concepts of assimilation stages from Meyer and Goes (1988). An assimilation stage can be defined as a description of how deeply an adopted innovation penetrates the adopting unit (company, group or individuals) (Fichman 1992, 2001). Gallivan (2001) suggests a six-staged model for describing innovation assimilation based on the work of Cooper and Zmud (1990) and Saga and Zmud (1994). The stages in the model are defined as follow.

- Initiation, a match is identified between an innovation and its application in the organization;
- Adoption, a decision to adopt an innovation is made in the organization;
- Adaptation, the innovation is developed, installed and maintained, and organizational members are trained to use the innovation;
- Acceptance, members are committing to using the innovation, which means that the innovation is employed in the organization;
- Routinization, usage of the innovation is encouraged as a normal activity in the organization; the innovation is no more defined as something out of ordinary;
- Infusion, the innovation is used in a comprehensive and sophisticated manner, which results in increased organizational effectiveness. Three different facets of infusion are described:
    - Extensive use: using more features of the innovation;
    - Integrated use: using the innovation to create new workflow linkages among tasks;
    - Emergent use: using the innovation to perform tasks not in the pre-conceived scope.

Gallivan (2001) concludes that, although the adoption of innovations always requires some degree of top-down, organizational level decision, it is usually not possible to occur without a degree of bottom-up assimilation of developers. Furthermore, Gallivan (2001) argues that assimilation stages describe how deeply an innovation penetrates the adopting unit which can be company, division, workgroup, or individuals.

Zmud and Apple (1992), Gallivan (2001) and other models discussed above are based upon what Zmud and Apple (1992) call an unfreeze-refreeze model of innovation adoption which implies a sequence of the stages. In terms of the later three assimilation stages, acceptance can be seen as the end of an unfreezing phase while routinization and infusion are increasingly refreezing phases.

The Assimilation Stages of Agile Practices in Use

This section presents the interpretation of the innovation adoption concepts in terms of agile practices in use. Rogers (1983) contends that a practice perceived as new by adopters can be characterized as an innovation; therefore, agile practices can be defined as process innovations for an adopting ISD team. This study focuses on agile practices in use i.e. where initiation, adoption and adaptation have already occurred. Therefore, in the context of this study the later stages of assimilation i.e. acceptance, routinisation and infusion are more relevant. A key distinction between acceptance and routinization is whether a practice becomes a routine part of a development process, regardless how it is used, "by the book", or in a tailored way. If a practice is seen as being routinized in a process, the three facets of infusion (i.e. extensive use, integrated use and emergent use) will be used as indicators to decide if it reaches infusion stage. This is consistent with the key assumption abovementioned, that is, the acceptance, routinization and infusion of agile practices happen in a sequential manner.

The three facets of infusion need to be re-considered to suit the study. Extensive use is to use more features of an innovation, which makes more sense if an agile method is investigated as a whole. Since the study is focused on individual agile practices, intensive use, rather than extensive use, seems more relevant and applicable here. The intensive use of a practice can be defined as the more frequent or strengthened use of the practice than suggested by

the method. The definition of integrated use also needs to be adapted. The interconnected use of an agile practice with other agile practices is regarded as an integrated use of that practice, rather than new linkages of workflow that the practice gives rise to. The definition of emergent use, instead, can be taken directly without modification.

Table 2 summarizes the interpretation of the concepts of innovation assimilation in terms of agile practices in use. The concepts, together with the key assumption, enable an appropriate analysis of the assimilation stages of agile practices in use.

Table 2. The Later Assimilation Stage of Agile Practices

Assimilation stages	Conceptualization in studying agile practices in use
Acceptance	An agile practice is used only when needed, or in some special situations.
Routinization	An agile practice is used regularly, as a routine part of the process.
Infusion	An agile practice is not only routinely used, but also in a comprehensive and sophisticated way, which is indicated by: - Intensive use: an agile practice is used more frequently or more intensively than suggested by the method; - Integrated use: an agile practice is used in an interconnected way with other practices; - Emergent use: an agile practice is used in the areas not prescribed by the method. These uses tend to be routinized as well.

Take pair programming as an example. It is at the acceptance stage if it is only used by the team in some specific situations, not as a routine of the development process. For instance, the team only pair on some complex tasks from time to time. If routinization is achieved, pair programming is used continuously throughout the project with little or no exception. The infusion stage of pair programming means that it is not just used continuously, but is also used in some modified manner, for example, as a way to communicate and demonstrate XP practices to team members (Rasmusson 2003), or as a way to implement peer-discipline within the team.

Research Approach

The objective of this study is to explore the application of innovation adoption theories in agile research, and to obtain a better understanding of agile practices in use through the perspective of later assimilation stages discussed in the previous section. Since the study is exploratory in nature, with the intention of investigating contemporary phenomenon in a real-life context, case studies are considered by the researchers to be a suitable research approach. The use of cases is also beneficial where control over behaviour is not required or possible, as research data can be collected through observation in an unmodified setting (Yin 2003). While case study allows the capture of detail and the analysis of many variables, the method is criticized for a lack of generalizability, a critical issue for case study researchers. 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 generalization from a sample to a population, but rather plausibility and logical reasoning through developing concepts and theory, drawing specific implications, and contributing rich insight.

This study uses a multiple-case design. The rationale behind a multiple-case design is that it allows a cross-case pattern search, which reassures the researcher that "the events and processes in one well-described setting are not wholly idiosyncratic. At a deeper level, the aim of multiple case study is to see processes and outcomes across many cases, to understand how they are qualified by local conditions, and thus to develop more sophisticated descriptions

and more powerful explanations" (Miles and Huberman 1994, p. 172). Researchers become more confident in a theory when similar findings emerge in different contexts and build up evidence through a family of cases.

Following these suggestions, three cases have been selected for this study, which allow a meaningful and stark comparison. Each case is based on an ISD team who have adopted a set of agile practices from XP, Scrum or both, and have used them for some time. Other relevant decisions are the unit and level of analysis. This study investigates individual agile practices in use, not agile methods in general, thus each agile practice is considered a suitable unit of analysis. The level of analysis is at team level. The study examines how each agile practice has been used and which assimilate stage it has reached in a team as a whole, rather than concerning the individual developer's reaction towards the adopted agile practices.

Background to the Cases

Drawing conclusions of empirical results is always difficult because the results largely depend upon project settings, especially empirical research of agile ISD (Layman et al. 2006). The context of the ISD team plays an important role in field study. Table 3 provides an overview of the profiles of the three cases.

Table 3. The Profiles of the Three Cases

	Team A	Team B	Team C
Team size	8	6	4
Team composition	7 developers, 1 development manager.	4 developers, 1 tester, 1 Scrum master	3 developers, 1 project manager
Location	Partially distributed in two continents	Collocated in an open office space	Collocated in an open office space
Development method	XP	Scrum/XP	XP
Years of experience of agile methods	4.5	1.5	5
Type of system developed	Commercial product	Commercial product	External application
Company background	Medium, service-oriented solution provider	Medium, providing security solution and services	Small software house, specialized in security, management systems

Team A is an ISD team in a multi-national company. The company provides service-oriented architecture solutions. XP has been introduced and adopted company-wide more than 4 years ago. Before embarking on XP, the company already had an open culture for developers to communicate, collaborate and help each other. So XP was not viewed as a dramatically different way of development, rather, it was considered just explicating the way things should be done naturally. XP provides structure and discipline to the team. The company had intention to adopt XP in full scale, and the team had formal training on XP, but the practices have been adopted only loosely or partially, and some practices have met strong resistance. Generally speaking, test first and continuous integration are considered to be used to their most effects. Pair programming suffered biggest resistance.

Team B resides in a medium-sized company that provides IT services in security domain. It is the first ISD team that adopted agile methods in the company. The objective of the company to adopt agile methods is to respond to new threats in market significantly faster than its competitors. Before adopting agile methods, the company had based its systems development on the CMMI goals and practices, therefore, had a strong plan-driven culture. A project manager was seen as a key person responsible for a software development project. When the senior management made the decision to adopt agile practices, large Scrum and XP trainings have been organized for all project managers and developers. The project manager in Team B played the role of a key change agent who brought the

ideas from Scrum, such as self-organizing teams, and later on the XP practices to the team. The team members were provided with the possibility to select agile practices that they wanted to use as a part of their ISD process.

Team C is an ISD team in a software house specialized in network security and management systems. The team had failed to deliver its last project before embarking on a collaborative effort with an XP training laboratory. The team had carried out intensive training for 6 months before they returned home and applied the XP practices in their on-going activities. They successfully completed several projects with the development approach they learnt. They felt they reached their goals of developing software "good, fast and cheap", and "working in an enjoyable way". The team is the only one in the company using XP till now. The team see a clash between the way they work and the culture of the company.

Data Collection

Data was collected primarily through personal face-to-face interviews, which is considered the superior data gathering technique for interpretivist studies such as this (Yin, 2003). Personal interviews are also well suited for exploratory research because they allow expansive discussions which illuminate additional factors of importance (Yin, 2003, Oppenheim, 1992). 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).

A guiding script was prepared for use throughout the interviews to establish a structure for the direction and scope of the research, to ensure the researcher covered all aspects of the study with each respondent, to manufacture some element of distance between the interviewer and interviewee, and to permit the researcher to compare and contrast responses (McCracken, 1988). The researcher circulated the guiding questions in advance to allow interviewees to consider their responses prior to the interview. The questions were largely open-ended, allowing respondents freedom to convey their experiences and views, and expression of the socially complex contexts that underpin software development and agile method use (Yin, 2003, Oppenheim, 1992).

The interviews lasted between 50 and 120 minutes with the average being approximately 85. The interviews were conducted in a responsive (Rubin and Rubin, 2005, Wengraf, 2001), or reflexive (Trauth and O'Connor, 1991) manner, allowing the researcher to follow up on insights uncovered mid-interview, and adjust the content and schedule of the interview accordingly. Furthermore, the researcher kept a diary of questions asked during each interview, and analysed their effectiveness, making refinements and additions to the set of questions prior to the next meeting. In order to aid analysis of the data after the interviews, all were recorded with each interviewee's consent, and were subsequently transcribed, proof-read and annotated by the researcher. In any cases of ambiguity, clarification was sought from the corresponding interviewee, either via telephone or e-mail.

Documentation review and field notes were used as complementary data collection methods. Sources of documents include ISD documents, project management documents, corporate websites or brochures, and other publications available.

Data Analysis

The unit of analysis is at the ISD team level as opposed to the individual developer or indeed the organisation within which the team sits. The data has been analyzed in two stages. Following analysis of each case as a stand-alone entity or what Yin (2003) refers to as 'within-case analysis' , the researchers engaged in cross-case comparison. Within-case analysis takes the form of a detailed case writing-up for each case. Though descriptive, it is central to the generation of insight. The intention is to become intimately familiar with each case (Eisenhardt 1989). The focus of the data analysis is on the comparison of the three cases. Cross-case comparison is intended to enable a better understanding of different use modes of the agile practices and the assimilation stages they have reached. The data analysis process has been guided by the concepts and their interpretations presented in the previous section. They play as sensitizing and sense-making devices in the 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 data units that refer to each subject across the interviews. The coding structure adopted in

this research consisted of three distinct mechanisms. Firstly, an ID code was attached to each interviewee. Secondly, a classification schema was built, acting as what Miles and Huberman (1999) call a set of "intellectual bins", used to segment and filter the interview data collected. Finally, pattern coding was used in order to "identify any emergent themes, configurations or explanations" (Miles and Huberman, 1999). This approach aims to aggregate and summarise the previous codes, identifying themes and inferences across all.

Findings

Agile Practices in Use through an Assimilation Perspective

This section discusses how each XP and Scrum practice has been used in the three cases, analysing adoption from an assimilation perspective.

Small Releases (Scrum Sprints)

Team A's organisation has a general policy to issue a major product release every 12 months and a minor one every 6 months. The team collaborate with other teams on a product line and all follow 6-week milestones between the releases to coordinate their work. At the end of each milestone, there is a requirement to produce a demonstratable delivery from each team. Within the 6-week period, the teams are not obliged to use 2-week iterations. They are left to their own device to decide how to approach the milestones. Team A keeps 2-week iterations out of project management rather than technical development concerns. There is no delivery or acceptance test at the end of 2-week iterations, and so small releases practice is embodied at the 6-week milestone level, not at the 2-week XP iteration level.

Team B follows time-boxed monthly iterations as suggested by Scrum. Monthly sprint is considered a suitable iteration length for the type of product the team develop. Shorter iterations could not accommodate the features that need to be implemented for the working prototype. At the end of each sprint, prototypes are always presented to the product owner and various groups of stakeholders. Then sprint reviews are held to collect feedback from the stakeholders, to increase product visibility and to ensure the product fulfils the demands of the customers. Compared with Team A, Team B has used the small releases practice in a more stable and routinized way.

Team C uses this practice in a more sophisticated way, whereby one-week iterations have been used. The team have experimented with both longer and shorter iterations, and finally opted for one-week length which the team feel is the optimal pace for them. Generally the team prefer fixed-length iterations, but depending on the status of a project, the team do vary iteration length. However, no user requirement change is allowed to be introduced during an iteration. The customers can check the progress of the development anytime they want, or clarify their understanding of the requirements, but they can only introduce changes at the beginning of the next iteration. At the end of each iteration, the team always deliver a piece of working software to the customers, either in test environment or directly in the live environment. Working software is the only delivery at the end of an iteration, with no accompanying documentation. The customers test the deliverable according to the acceptance tests they write with the help of the team during planning games. The team guide the customers to write acceptance tests in such a way that it can identify precise business scenarios from business perspective and only address business complexity. The technical complexity is internal to the team and therefore should be hidden from the customers.

Planning Game (Sprint Planning)

Team A start with a planning meeting between the product manager and the managers of the teams involved at the start of each iteration. Within Team A, since 2-week iterations are not used as a routine, the planning game is only used when the team feel there is a need to plan, which then is based on the common-sense judgement, for example: "*Should this piece of work be documented, if yes, better to have a plan for it" (engineering manager, Team A).*

In contrast, Team B has a regular sprint planning meeting at the beginning of each monthly sprint, in which the requirements are defined at a more detailed level and put into a sprint backlog: "*In the first few iterations I met with product owner, we prioritised the backlog…then in the sprint planning meeting we saw…,whether there is something missing of the list, we reprioritised features and then took the items in the following sprint*" (Scrum

master, Team B). Developers do not have much influence on how the requirements are selected and analyzed. They are, however, responsible for defining tasks and providing estimates for implementing them in that sprint. Customers are partially involved in a sprint planning meeting, but the product owner, who plays the role of customer proxy, is always involved in the meeting. "*Team, product owner and delivery team are always in sprint planning meetings.*" (developer, Team B)

Team C not only uses planning games routinely, but also uses different techniques to make them more effective. Each iteration starts with a planning game in which all the team members and the customers participate. User requirements are broken down and specified as user stories. Then the team analyse each user story and assign estimates to it. The estimates then are compared with the capacity of the team to implement user stories in that iteration. This gives the customers and the team an idea of what user stories can be implemented in that iteration. If not all stories can be implemented, the customers prioritize the stories and choose the ones that are to be delivered at the end of the iteration. Then the stories are further divided into tasks. Both tasks and stories are written on a quarter A4-sized paper cards. The team use different colours to distinguish between different types of cards: green for user stories, blue for tasks and red for spikes (exploratory tasks aiming at solving any issues emerged during the course of development). All the cards are then stuck to a whiteboard and arranged in categories. All information about user stories and acceptance tests are recorded in the project wiki, to which the customers also have access. Through planning as frequent as every iteration and using various techniques to support this activity, the attitude of the team towards planning is: *"We do not plan the whole project from beginning to the end, it is useless." (Developer, Team A)*

Daily Stand-up Meeting (Daily Meeting)

Team A adopted the idea of stand-up meetings, but do not use it on a daily basis. They typically have a quick stand-up meeting at 12 o'clock in a day when they feel they have not communicated sufficiently for a while.

Daily meetings in Team B is a short session focused on answering the questions including "What did I work on yesterday?", "What do I plan on working today?" and "What is getting in my way?" Daily meetings are held regularly for disseminating information and analysing metric data of the project status "*those meetings gave a daily view into what was going on in the project*, Scrum master, Team B. Team A always uses metrics in the meetings for planning and project monitoring. Daily meetings are also an effective problem solving mechanism. Developers feel that most problems, including design issues, are actually solved based on the discussions in daily meetings.

Team C uses this practice more intensively than what is described "by the book". The team has renamed this practice by a metaphoric term "steering". They use it as a daily planning mechanism. In each working day, the team will have a brief steering session in the morning, to quickly plan what to do in that day, and to express technical obstacles and ask others for help. The session is intended to be very short. Steering can happen more than once a day. Sometimes, in addition to the morning steering, the team will have a steering session in the afternoon, to plan the work of the remaining day. Steering, as the name implies, is constant adjustment of where the team goes in the light of what happens around.

Retrospective

In Team A only the team manager formally reflected on their development process, but not on a regular basis. It is considered unnecessary by the team since they believe there are no issues with their process.

> *(at team level do you do this kind of retrospective?) yeah we have, we are not doing it regularly… maybe it's helpful, I mean again that's, yeah, those are the things get lost. (Engineering manager, Team A)*

Team B holds retrospective workshops regularly at the end of each sprint to reflect on their process, to decide how to adopt other practices effectively. Retrospective also provides a mechanism to motivate the team. For instance, to motivate developers to automate unit testing and to try test-driven development *"In retrospectives we had several discussions with the team, why is the unit test numbers so low, why is the code coverage so going down instead of maintaining" developer, Team B.*, the Scrum master uses retrospective workshops to present the concrete measurement data of the test coverage.

Retrospective happens more frequently in Team C, to the extent of every day. The team name this practice "feedback" and it is the first thing they do in a working day. During a feedback session the team members reflect on

the previous working day. The feedback is not only on the development process, but also on the feelings team members had, anxieties felt, what the team have achieved or whether something went wrong. The team record the feedbacks in the team wiki to remember the lessons learnt. The team have even extended the practice to self retrospectives. They believe that self reflection is a precondition for effective retrospective on their process, and effective communication and collaboration among the team.

40-hour Week

Team C is the only one that has followed the rule of 40-hour week. Based on this rule, the team have developed a framework to integrate other practices, including daily retrospective and daily stand-up meeting. A typical working day is eight hours, time-boxed into 15 working units (as shown in Figure 1). One unit is composed of 25-minute working time followed by a 5-minute break. During the day there are 2 long breaks, 15 minutes each. The team actually use a timer to help them to keep this pace. The timer rings after each 25 minutes, to remind the team members to have a break.

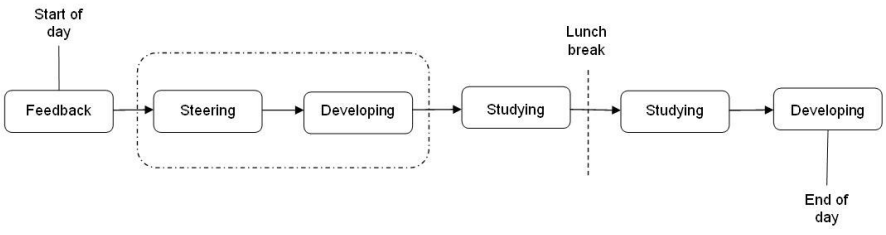

Figure 1. The Daily Pace of Team C

Team B does not follow the rule of 40 hour week but the team has applied the idea of time boxed iterations: " *iterations in the team are time boxed and follows the same rule every time"* developer, Team B. - -iteration starts always with half day sprint planning meeting, continue with daily meetings and end with half day iteration retrospective and half day sprint review meeting.

On-site Customer

On-site customer has not been fully implemented in any of the three teams. Team A and Team B use internal resources as customer proxies instead of real customers. Even the customer proxies are not always on-site. In Team A, product managers of the company play this role. Although the team try to communicate with them whenever possible, they are not always available to the team. For Team B the product owner and the Scrum master communicate daily with the customers. The real customers sometimes participate in sprint planning and review, but not regularly.: "*involvement of customer, even through conference calls, through onside visits, would be beneficial.*" developer, Team B. In contrast, Team C manage to involve their customers in every planning meeting and acceptance tests weekly, and communicate frequently with them within the weekly iteration, to achieve the same effect that on-site customer can bring to them.

Pair Programming

Team A does not use pair programming on coding tasks, but rather primarily as a trouble shooting tool. Pair estimation of user stories has been used, sometimes successfully, sometimes not. Team B pairs on code review and new staff training "*…the only real way to review code is actually pair programming*". Neither Team A nor Team B treats pair programming as a routine practice in their development process. In Team C, instead, developers always program in pairs on real code. They share one desktop, one using keyboard (as a driver), the other using mouse (as a navigator). Role switching happens when the navigator has some new idea. He can take the keyboard from the driver and start to write code, which is a preferred way of communicating ideas rather than discussion. Pairing is self-arranged and is not fixed. Pair rotation happens frequently. Between the two developers, generally the developer who is responsible for the task stays, the other goes to pair with a different developer.

Testing

This practice has been used by all the three teams, though Team B has only adopted it partially at a later stage of the project. Team B feel that testing activities are too complex and time consuming, and are not as primary as creating new features rapidly. As a consequence, this practice has not been used regularly at the beginning of the project. "*we weren't use test driven development, so we wrote the component first and then the unit test… later on, when we were quite more disciplined then we started writing the tests first and then the component*" developer, Team B. In comparison, both Team A and Team C use testing coherently and consistently, especially test first. It is a part of the development routine. The two teams always write unit test code before writing function code. In addition, Team A has developed a comprehensive testing framework that combines test first and continuous integration, as shown in Figure 2. Team C, instead, discovered an emergent use of test first. They use it to learn third party software. To understand how a piece of software works, rather than reading documentation, they write tests and run them on it to understand its functionalities.

Continuous Integration

Continuous integration has been implemented consistently in Team C. In Team B, setting up the continuous integration framework was found difficult and time consuming, "*around in the middle of the project we had stable build environment"* developer, Team B. When the problems with the build environment were solved, the team B started to use continuous integration hourly. Team C continually integrate the well-refactored function code of different stories. Team A, as mentioned in "testing practice" section, have a more comprehensive test-oriented development framework which has also been adopted company-wide in a consistent way (see Figure 2). The framework integrates testing first and continuous integration to guarantee the high quality of the code.

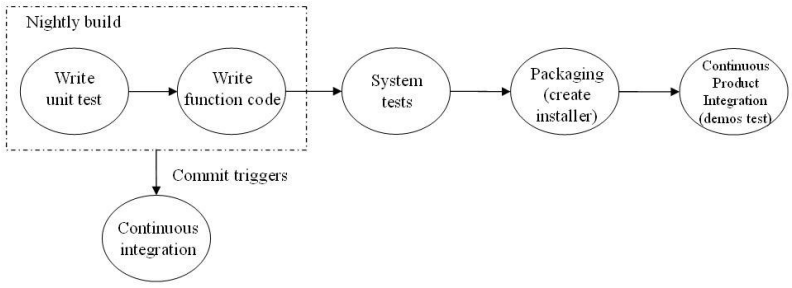

Figure 2. The Test-driven Framework of Team A

Collective Ownership

In all three teams collective ownership has been endorsed by the team members and regarded as a natural result of team collaboration. Team A has achieved it on the code, as the practice is intended, but it goes beyond the scope of code in Team B and Team C. They are sharing other forms of working results within their members as well. For example, both teams have their team email addresses which they use to communicate with their customers. Any achievements are shared among team members. Another example from Team C is that when the team members respond to a customer's request, they make the contact to the customer all together, and talk to the customer all together. No one claims the ownership of the solution. They collectively own it.

Simple Design

Simple design is not evident in Team A and Team B, but is highly embraced by Team C. Team C believes that design emerges from code; therefore simple design is embodied by simple code. Writing simple code is always what the team strive to achieve. Being simple is not only important in coding and designing; the team generalize the value

behind the practice and apply it to the communication. They believe that they have to be simple in order to effectively communicate with each other. The team's understanding of simple design goes beyond the system:

> *Good practices are to write simple stories, keep the system simple and your code simple by refactoring, and also studying new technologies that can help you to make things simple in a quick manner… learning also is a (way to) increase your skill and so is useful for managing complexity. I talked about simplicity, but all the XP values deal with complexity in some way. (Developer, Team C)*

Refactoring

Team A and B do not refactor every piece of code. They do so only if there is a good reason for it. They strive for working code that is useful, not perfect. Team C does not refactor every piece of code either; they do so when there are duplicated behaviours in the system. Compared with Team A, Team C does refactoring in a relatively more consistent and regular way. They refactor both test code and function code.

Coding Standards

All the teams conform to a set of coding standards they had even before they applied agile methods. It is a well established routine, but no evidence of any mode of use that indicates infusion has been found in any of the three teams.

The Factors in Agile Practice Assimilation

This section presents the findings regarding the factors in the assimilation of agile practices. Several patterns have emerged through the comparative analysis of the agile practices used in the three cases, combined with their organizational contexts. To better illustrate these patterns, Table 4 re-organizes the agile practices used in the three cases along the dimension of the three later assimilation stages.

Table 4. A Summary of the Agile Practices at Different Assimilation Stages in the Three Cases

Assimilation stage	Team A	Team B	Team C
Acceptance stage	- Small releases - Planning game - On-site customer - Retrospective - Pair programming - Refactoring	- On-site customer - Pair programming - Testing - Refactoring - Continuous integration	---
Routinization stage	- Collective ownership - Coding standards	- Scrum Sprints - Scrum planning - Retrospective - Daily meeting - Coding standards	- On-site customer - Continuous integration - Refactoring - Coding standards
Infusion stage	- Testing - Continuous integration	- Collective ownership	- Small releases - Planning game - Daily stand-up meeting - 40-hour week - Retrospective - Collective ownership - Pair programming - Testing - Simple Design

Agile practices in use do not reach the same assimilation stage simultaneously, even if they are adopted at the same time.

Team A started to use all the practices listed in Table 4 at the beginning of their XP adoption. But only testing and continuous integration have reached the infusion stage. Some practices are routinized. Many others remain at the acceptance stage after more than 4 years of usage. Team C is in a similar situation, though in a slightly more concentrated manner. All the adopted agile practices have been routinely used, but not all have reached the infusion stage. Team B is different than the other two cases in that the team did not adopt all the practices together. They adopted different practices at the different stages of the project. However, the practices that were adopted at the same time, for example, Scrum sprint planning and pair programming, are at different assimilation stages.

The period of use of agile practices does not have a proportional effect on their assimilation stages.

This finding overlaps, to a certain extent, the previous one. However, it is more focused on a cross-case comparison of the assimilation stages of the same agile practice, rather than on a within-case comparison of the assimilation stages of different practices. Team A and Team C have both adopted agile practices for 4 to 5 years. However, as shown in Table 4, many practices remain at the acceptance stage in Team B while in Team C they have reached the infusion stage, for example, small releases and retrospective. In contrast, small releases and retrospective in Team B have reached routinization after only one and a half year of use. It may be argued that, based on this analysis, the duration of use does not have a proportional effect on the assimilation stage that agile practices reach.

The agile practices addressing the specific needs of the adopting team reach deeper assimilation levels.

In Team A, agile practices addressing management issues remain at the acceptance stage while development practices, such as testing and continuous integration, have reached the infusion stage within the same timeframe. One possible explanation can be found in the high quality need of Team A, which is a consequence of the software product line approach the company adopts. A problem with product line is that all the defects or bugs in one component can be repeated in all other components and products in the product line. This characteristic imposes a need of high quality. Quality related practices, such as test first and continuous integration, can address this need. Instead, the company of Team A has always an open culture. Developers communicate and collaborate freely. Mutual help goes on amongst developers. There is no need to emphasize these aspects through adopting the relevant agile practices. As a result, these agile practices, such as daily stand up meetings, do not reach the infusion stage even if they have been used for more than 4 years.

This finding can be supported fatherly by Team B in which the management practices of Scrum have achieved the routinization fairly quickly in only one year and a half of usage, because the team regarded these practices as a solution for the communication and change request management between the team and their customers.

Regular retrospective may drive agile practices to later assimilation stages.

Regular retrospectives happen in Team B and Team C, where most practices have reached either the routinization or infusion stage. In Team A, where most practices remain at the acceptance level, instead, retrospective does not happen at all. Since the purpose of retrospective is to review and improve the use of the used agile practices, retrospective would be one of the key mechanisms to drive agile practices to reach a deeper level of assimilation.

Discussion

Agile practices have been used in different modes and have reached different stages of assimilation in the three cases presented in this paper. The emergent patterns reveal the factors in the assimilation of agile practices. The implication of the findings are discussed in this section.

One implication can be drawn from the finding that the duration of usage does not affect the assimilation stage of an agile practice. It resonates to the argument of Fichman (1999). Innovation adoption studies are concerned with understanding the deployment of innovation by following its spread across a population of potential adopters over time (Fichman 1999). However, Fichman (1999) reminds that for some technologies it may be inappropriate to assume that in most organizations these later assimilation stages will automatically follow earlier stages. The finding

of this study indicates that time may not be an appropriate indicator in the evaluation of the assimilation stages of agile practices. This is an important aspect to take into consideration for the studies that intend to understand how agile practices have been routinized and infused in a team or even at a company level. Time is an important dimension in this kind of study, but one needs to be cautious when using it as to measure the levels that agile practices penetrate into adopting units.

Another finding suggests that high needs may lead relevant practices to their comprehensive or sophisticated use and a deeper level of penetration in adopting organizations. This finding provides an insight of how to effectively adopt agile practices. Since the agile practices addressing the needs of a adopting organization have potential to be routinized or infused, it is sensible to identify the areas that an organization needs to improve, and then select relevant agile practices to adopt rather than taking the whole set of practices in an agile method. Therefore, this study supports the adoption approaches presented in Svensson and Host (2005), Pikkarainen and Passoja (2005), and Pikkarainen and Mäntyniemi (2006). These approaches identify firstly the organization challenges, then map them with agile based solutions (i.e., practices), to understand what agile practices should be introduced and why they should be, before the actual deployment of agile practices. For example, Svensson and Host (2005) presents experiences of introducing agile practices in a software maintenance and evolution organization. Their study is focused on analyzing which agile practices are easiest to introduce in a development project. It claims a similar finding that agile practices should be introduced in the area where it is currently seen as most beneficial. For example, code related practices were introduced because of the phase in which they were maintaining and supporting several systems. In the study of Pikkarainen and Mäntyniemi (2006) agile adoption is started with identifying the areas of an ISD process that needs improvement through a CMMI goal-mediated framework.

The innovation adoption theories used in the study presume that infusion of agile practices could not happen before they are routinely used and become a part of the normal organizational life. This is in agreement with Zmud and Apple (1992) who show empirical evidences to support the argument that an organization could not achieve high level infusion without having high level routinization first. Zmud and Apple (1992) suggest that routinization and infusion are capturing important aspects of innovative organizational behaviour. Teams using agile practices as a routine part of their development work may in a good position to discover innovative ways of using them, thus have potentials to be innovative. On the other hand, this study also finds that some level of integrated or emergent use of agile practices can occur also in the acceptance stage of assimilation. For example, pair design is an emergent use of pair programming, but it is not used regularly either in Team A or Team B. Can integrated or emergent use at the acceptance stage be viewed as an innovative behaviour even though it is yet to be routinized? It is a question yet to be explored.

An examination of the use modes of retrospective in the three cases gives rise to an assumption that there might be a link between the frequency of retrospectives and the assimilation stages that agile practices can reach. The effects of iteration retrospective on development teams have been studied in Salo and Abrahamsson (2006). They show that teams are able to do continuous, small but effective improvements through iteration reflective workshops. Meanwhile, the reflective mode of thinking may improve the developers' understanding of their own processes (Hazzan and Tomayko 2003), which may in turn lead agile practices to deeper levels of penetration in the adopting organizations.

Finally, in this study integrated use of agile practices is re-defined as interconnected use and is taken as an indicator of infusion stage. Many studies present examples of interconnected use of agile practices (Rasmusson 2003; Vanderburg 2005; Fitzgerald et al. 2006; Layman et al. 2006). Rasmusson (2003) discovers that introducing test driven development technique depends on team learning which happens during pair programming. This study showed that agile practices are not isolated or independent, and they can be used in an integrated way. Layman et al. (2006) find that pair programming can be used as a framework to integrate other agile practices. These studies show that interconnected use of agile practices may provide new schemes for ISD.

Conclusion and future work

Many studies have investigated adoption and application of agile practices in real-world contexts, but few have attempted to understand routinization and infusion of agile practices. This paper contributes to the literature by making a first attempt to conceptualize the later assimilation stages of agile practices and to explore agile practices in use through an innovation assimilation perspective. Three cases have been presented and compared to evaluate the assimilation stages of the agile practices they have used. Among the key findings are that time may not be an

appropriate indicator of assimilation stages of agile practices, instead, needs of an adopting unit may be a driving force. Agile practices addressing the needs of a software development team have potentials to achieve later stage of assimilation. The methodological contribution of this study is that it shows a possible way to apply the concepts from innovation adoption theories in evaluating the use of agile practices in systems development.

There are obvious limitations of any case study research including issues regarding generalisation of findings from the data collected. Another specific methodological limitation of this study is that, since only snapshots of agile practices in use have been taken and analyzed, it is not possible to understand the factors in the assimilation of agile practices from a process perspective. A research question associated to the process perspective is how different agile practices reach assimilation stages. The concepts from innovation adoption theories and their interpretation need further evaluation and extension to make it more adequate for the study of agile practices. 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 Lean Software Development, Feature Driven Development, Agile Project Management, Crystal and Adaptive Software Development are all methods that could be assessed. Secondly, a more quantitative approach could be adopted by means of a large-scale survey. This could be used to determine the levels of agile method and agile practice assimilation across the ISD community, with results that are more generalisable than those contained in this research. This could reveal interesting insights such as which agile methods are most assimilated and why. The 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 to correlate assimilation to effectiveness or success.

Acknowledgements

This research was conducted within the ITEA Flexi project which Lero, the Irish Software Engineering Research Centre is one of the research partners funded by Enterprise Ireland. Part of the work was done in VTT, Technical Research Centre of Finland funded by National Technology Agency of Finland (TEKES) and Nokia Foundation in Finland. A special acknowledgement to Brian Fitzgerald for his valuable advice on the research and the paper.

References

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. "Agile Software Development Methods: Review and Analysis", *VTT Electronics, Espoo*, 2002, pp. 1-107.

Ambler, S. W. *Agile Modeling: Best Practices for the Unified Process and Extreme Programming*, John Wiley & Sons., New York, 2002.

Baskerville, R., Travis, J. and Truex, D. “Systems without Method”, In *The Impact of Computer Supported Technologies on Information Systems Development,* Kendall, K., DeGross, J. and Lyytinen, K. (Ed,), Elsevier Science Publishers, North Holland, 1992, pp. 241-269.

Beck, K. *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Reading, Mass., 2000.

Beck, K. and Andres, C. *Extreme Programming Explained: Embrace Change*, *Second Edition*, Addison-Wesley, Boston, 2004.

Boehm, B. “Get Ready for Agile Methods, with Care”, *IEEE Computer* (35), 2002, pp. 64-69.

Boehm, B. and Turner, R. *Balancing Agility and Discipline: A Guide For The Perplexed*, Addison-Wesley, Boston, MA, 2004.

Coad, P. and Palmer, S. *Feature-Driven Development*, Prentice Hall, Englewood Cliffs, NJ, 2002.

Cockburn, A. *Crystal Clear: A Human-Powered Software Development Methodology for Small Teams*, Addison-Wesley, Reading, MA, 2001.

Cooper, R. B., and Zmud, R.W. "Information Technology Implementation Research: A Technological Diffusion Approach", *Management Science* (36:2), February 1990, pp. 123-139.

Davis, F. D. "Perceived usefulness, perceived ease of use, and user acceptance of information technology", *MIS Quarterly* (13:3), 1989, pp. 319-339.

Dingsoyr, T., Hanssen, G. K., Dyba, T., Anker, G. and Nygaard, J. O. "Developing Software with Scrum in a Small Cross-organizational Project", in *Proceedings of the Thirteenth European Conference on Software Process Improvement,* Richardson, I., Runeson, P. and Messnarz, R. (Ed.), Joensuu, Finland, October 2006, pp. 5-15.

Drobka, J., Noftz, D. and Raghu, R. "Piloting XP on Four Mission Critical Projects," *IEEE Software* (21:6), November 2004, pp. 70-75.

Eisenhardt, K. M. “Building Theories from Case Study Research”, *Academy of Management Review* (14:4), October 1989, pp. 532-550.

Fichman, R. G. "Information Technology Diffusion: A Review of Empirical Research", in *Proceedings of the Thirteenth International Conference on Information Systems*, Dallas, December 1992, pp. 195-206.

Fichman, R. G. "The Role of Aggregation in the Measurement of IT-related Organizational Innovation", *MIS Quarterly* (25:4), 2001, pp. 427-455.

Fitzgerard, B., Hartnett, G. and Conboy, K. "Customising Agile Methods to Software Practices at Intel Shannon", *European Journal of Information Systems* (15:2), April 2006, pp. 200-213.

Gallivan, M. "Organizational Adoption and Assimilation of Complex Technological Innovations: Development and Application of a New Framework", *The DATA BASE for Advances in Information Systems* (32:3), Summer 2001, pp. 51-85.

Grenning, J. "Launching Extreme Programming at Process-Intensive Company", *IEEE Software* (18:6), November 2001, pp. 27-33.

Hazzan, O. and Tomayko, J. “The Reflective Practitioner Perspective in eXtreme Programming”, in *Proceedings of Extreme Programming And Agile Methods - XP/Agile Universe,* Maurer, F. and Wells, D. (Ed.), New Orleans, LA, USA, August 2003, pp. 51-61.

Highsmith, J. *Agile Project Management*, Addison-Wesley, Boston, MA, 2004.

Hovorka, D. S. and Larsen, K. R. "Enabling Agile Adoption Practices through Network Organizations", *European Journal of Information Systems* (15:2), April 2006, pp. 159-168.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E. and Rosenberg, J. "Preliminary Guidelines for Empirical Research in Software Engineering", *IEEE Transactions on Software Engineering* (28:8), August 2002, pp. 721-734.

Layman, L., Williams, L. and Cunningham, L. "Motivations and measurements in an agile case study,", *Journal of Systems Architecture* (52:11), 2006, pp. 654-667.

Layman, L., Williams, L., Damian, D. and Bures, H. "Essential Communication Practices for Extreme Programming in a Global Software Development Team", *Information & Software Technology* (48:9), 2006, pp. 781-794.

Lippert, M., Becker-Pachau, P., Breitling, H., Kock, J., Kornstädt, A., Roock, S., Schmolitzky, A., Wolf, H. and Züllighoven, H. "Developing Complex Projects Using XP with Extensions", *IEEE Computer Society* (36:6), June 2003, pp. 67-73.

McCracken, G. *Qualitative Research: The Long Interview*, Sage Publications, Beverly Hills, CA, 1988.

Meyer, A.D. and Goes, J.B. "Organizational Assimilation of Innovation: a Multilevel Contextual Analysis", *Academy of Management Journal* (31:4), 1988, pp. 897-923.

Miles, M. and Huberman, A. *Qualitative Data Analysis*, Sage, London, 1999.

Miles, M. B. and Huberman, A. M. *Qualitative Data Analysis: an Expanded Sourcebook*, Saga, London, 1994.

Murru, O., Deias, R. and Mugheddu, G. "Assessing XP at European Internet Company", *IEEE Software* (20:3), May 2003, pp. 37-43.

Nawrocki, J., Jasinski, M., Walter, B. and Wojciechowski, A. "Combining Extreme Programming with ISO 9000", in *Proceedings of the First EurAsian Conference on Information and Communication Technology*, Tehran, Iran, October 2002, pp. 786-794.

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

Pikkarainen, M. and Mäntyniemi, A. "An Approach for Using CMMI in Agile Software Development Assessments: Experiences of Three Case Studies", in *Proceedings of the Sixth International SPICE Conference,* Luxembourg, May 2006, pp. 121-129.

Pikkarainen, M. and Passoja, U. "An Approach for Assessing Suitability of Agile Solutions: a Case Study", in *Proceedings of the Sixth International Conference on Extreme Programming and Agile Processes*, Baumeister, H., Marchesi, M. and Holcombe, M. (Ed.), Sheffield, UK, June 2005, pp. 171-179**.**

Poppendieck, M. “Lean Programming”, *Software Development Magazine* (9:5), 2001, pp. 71-75.

Rasmusson, J. "Introducing XP into Greenfield Projects: Lessons Learned", *IEEE Software* (20:3), May 2003, pp. 21-28.

Rising, L. and Janoff, N. “The Scrum Software Development Process for Small Teams”, *IEEE Software* (17:4), July 2000, pp. 26-32.

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

Saga, V.K. and Zmud, R.W. "The Nature and Determinants of IT Acceptance, Routinization and Infusion", in *Proceedings of the IFIP TC8 working conference on diffusion, transfer and implementation of information technology*, Levine, L. (Ed.), Amsterdam, North Holland, 1994, pp.67-86.

Salo, O. and Abrahamsson, P. "An Iterative Improvement Approach for Agile Development: Implications from multiple case study", *Software Process: Improvement and Practice* (12:1), January 2007, pp. 81-100.

Schwaber, K. and Beedle, M. *Agile Software Development with Scrum*, Prentice-Hall, Upper Saddle River, NJ, 2002.

Stapleton, J. *DSDM: Dynamic Systems Development Method*, Addison Wesley, Harlow, England, 1997.

Svensson, H. and Host, M. "Introducing an Agile Process in Software Maintenance and Evolution Organization", in *Proceedings of the Ninth European Conference on Software Maintenance And Reengineering*, Manchester, UK, March 2005, pp. 256-264.

Trauth, E. and O'Connor, B. "A study of the interaction between information, technology and society", In *Information Systems Research: Contemporary Approaches and Emergent Traditions,* Nissen, H., Klein, H. and Hirschheim, R. (Ed.), Elsevier, North Holland, 1991, pp. 131-144.

Vanderburg, G. "A Simple Model of Agile Software Processes - or - Extreme Programming Annealed", *ACM SIGPLAN Notices* (40:10), October 2005, pp. 539-545.

Walsham, G. "Interpretive Case Studies in IS Research: Nature and Method", *European Journal of Information Systems* (4), 1995, pp. 74-81.

Wengraf, T. *Qualitative Research Interviewing*, Sage, London, 2001.

Williams, L., Maximilien, E. M. and Vouk, M. "Test Driven Development as a Defect Reduction Practice", in *Proceedings of the Fourteenth International Symposium on Software Reliability Engineering*, Denver, Colorado, November 2003, pp. 34-45.

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

Zmud, R. W. and Apple, L. E. "Measuring Technology Incorporation/Infusion", *Journal of Product Innovation Management* (9:2), June 1992, pp. 148-155.