Empir Software Eng (2008) 13:303–337
DOI 10.1007/s10664-008-9065-9

The impact of agile practices on communication in software development

M. Pikkarainen · J. Haikara · O. Salo · P. Abrahamsson · J. Still

Published online: 23 May 2008

Editor: Tore Dybå

Abstract Agile software development practices such as eXtreme Programming (XP) and SCRUM have increasingly been adopted to respond to the challenges of volatile business environments, where the markets and technologies evolve rapidly and present the unexpected. In spite of the encouraging results so far, little is known about how agile practices affect communication. This article presents the results from a study which examined the impact of XP and SCRUM practices on communication within software development teams and within the focal organization. The research was carried out as a case study in F-Secure where two agile software development projects were compared from the communication perspective. The goal of the study is to increase the understanding of communication in the context of agile software development: internally among the developers and project leaders and in the interface between the development team and stakeholders (i.e. customers, testers, other development teams). The study shows that agile practices improve both informal and formal communication. However, it further indicates that, in larger development situations involving multiple external stakeholders, a mismatch of adequate communication mechanisms can sometimes even hinder the communication.

M. Pikkarainen (✉) · O. Salo · P. Abrahamsson
Technical Research Centre of Finland, Kaitoväylä 1, Box 1100, 90571 Oulu, Finland
e-mail: minna.pikkarainen@vtt.fi

O. Salo
e-mail: salo.outi@vtt.fi

P. Abrahamsson
e-mail: pekka.abrahamsson@vtt.fi

J. Haikara
Polar Electro Oy, Professorintie 5, FIN-90440 Kempele, Finland
e-mail: jukka.haikara@polar.fi

J. Still
F-Secure Corporation, Elektroniikkatie 3, 90570 Oulu, Finland
e-mail: Jari.Still@f-secure.com

The study highlights the fact that hurdles and improvements in the communication process can both affect the feature requirements and task subtask dependencies as described in coordination theory. While the use of SCRUM and some XP practices facilitate team and organizational communication of the dependencies between product features and working tasks, the use of agile practices requires that the team and organization use also additional plan-driven practices to ensure the efficiency of external communication between all the actors of software development.

Keywords Agile software development practices · Communication · Coordination theory

1 Introduction

Software organizations constantly need to react to market dynamics, new customer requirements and technological innovations (Beck 2000; Lycett et al. 2003). The degree of market dynamics and needs has increased over the past decades creating a number of fast-moving software organizations (Börjesson and Mathiassen 2004). The experiments and surveys on agile methods promise faster development thus improving the communication and collaboration inside agile teams and within the teams, customers and business units (Anderson 2003). Many organizations regard agile methods as a way of addressing key problems in software development; namely, the software takes too long to develop, costs too much and has quality issues upon delivery (Holström et al. 2006). Thus, agile methods (e.g. Extreme Programming [XP]; Beck and Andres 2004) and SCRUM (Schwaber and Beedle 2002) have been suggested as a way of responding to the changes, shortening the development time and improving communication and collaboration, especially in situations in which timing is a critical competitive advantage for an organization (Anderson 2003; Karlstrom and Runeson 2006).

One way of managing the coordination processes in an organization is to focus on communication and information transfer (Malone and Crowston 1994). Communication can be described as a way to manage relationships between producers and consumers (i.e. development teams, management and customers; Malone and Crowston 1994). This definition focuses on the efficacy of information exchange rather than quality or amount (Goles and Chin 2005). 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). It is also in a central position in all collaboration practices and processes (Paasivaara and Lassenius 2003). 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). Communication among stakeholders and software development project members and stakeholders is, however, a particular challenge for software development (Damian et al. 2000). This is not a new situation. For example, Parnas and Clements (1986) argue that the major problems in software organizations are as follows: there is no error free way to document requirements and there are changes due to external reasons and because documented knowledge is not up-to-date (Parnas and Clements 1986). Boehm and Ross (1989) argue also that the primary problem in project coordination is the demand for software projects to simultaneously satisfy the needs of so many actors; users, customers, development team, maintenance and management. All these stakeholders view and communicate on the same product from different perspectives: users require the product to be user friendly, customers seek reliability, costs and time schedule, managers strive for

the business goals, maintainers seek product documentation, modifiability and reliability, and developers seek technical challenges and fast career paths (Boehm and Ross 1989).

Both agile principles and methods provide solutions for the communication problems in companies. Agile methods, for example, suggest practices for collaboration and interaction inside and between the stakeholder groups. Although it seems that the use of agile practices would increase communication capabilities in software intensive companies, Turner (2003) argues that the companies using agile methods would also face a risk from overemphasizing tacit knowledge across a team. Agile software development does not, however, include only tacit ways of communicating (Turner 2003). Formal communication such as source codes, test cases, and a minimum, essential amount of documentation is also used in agile software development projects. Furthermore, it has been claimed 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 2003). Most of these problems may be a consequence of the lack of communication between these actors as identified in many studies (Cohn and Ford 2003; Coram and Bohner 2005; Svensson and Host 2005). The current research has, however, failed to provide valuable insight or discussion on the effects of agile practices on communication.

The agile practices investigated in this study are from eXtreme Programming (XP; Beck 1999) and SCRUM (Schwaber and Beedle 2002). These methods were chosen for a number of reasons, the foremost being the fact that they are the most popular of all agile methods (Fitzgerald et al. 2006). Secondly, they are very diverse approaches as XP is practitioner-oriented while SCRUM is focused on project management (Abrahamsson et al. 2002). By studying these two methods, this research ensures that lessons learned consider both perspectives. The goal of this paper is to increase the understanding of a customized set of XP and SCRUM practices on communication in software development. The research focuses on the research question: how do agile practices affect communication in software development teams and in an organization? The results are mainly based on qualitative data obtained during semi-structured interviews as part of an action research project.

The paper is structured as follows: Section 2 presents a conceptual model integrating the adoption of agile practices and communication, Section 3 describes the research design including the research approach and research context and Section 4 provides the empirical results on the effects of the adoption of agile practices on communication. Section 5 includes a discussion based on the research data and conceptual framework. The last section concludes the paper with some observations on the limitations of the study and an outline of avenues for future research.

2 Conceptual Background

A framework, generally, provides structured mechanisms to define phenomena in research and their links, i.e. how they relate to each other (Weick 1995). For example, a conceptual framework provides an explanation either graphically or in narrative form of the main things to be studied—the key factors, constructs and relationships between them (Miles and Huberman 1999). By using the Miles and Huberman (1999) coding approach in data analysis—setting out bins, naming them and identifying their interrelationships—can lead the researcher to a conceptual framework. The framework itself is a mechanism to help researchers decide the most important and meaningful variables for data collection and analysis (Miles and Huberman 1999). The purpose of this section is to build a conceptual framework for the focal study. The framework elements are communication in software

development and agile practices. Sections 2.1 and 2.2 provide definitions for these concepts and Section 2.3 describes the links between these concepts so as to create a framework for subsequent empirical analysis and discussion.

2.1 Communication in Software Development

Communication is an important and obvious way for team members to generate coordination processes in complex software development environments (Espinosa and Carmel 2003; Harbring 2006). The current literature defines the various characteristics of communication in the area of system or software development (Espinosa and Carmel 2003). For example,

> In software development, communication means that different people working on a common project agree to a common definition of what they are building, share information and mesh their activities (Kraut and Streeter 1995)

Goles and Chin (2005) have presented their conclusion in several research papers related to communication and argue that communication can be mostly defined as a two-way information sharing process which should happen daily in a routine manner.

Communication is characterized as one central aspect of the co-ordination theory (Malone and Crowston 1994) in which the communication is defined as an activity that is needed as a way to manage dependencies between the actors in the process. For example, a primary activity in requirements management is to transfer knowledge, such as the customer description of the developed system with the requirements needed to develop software. All the resources related to requirements analysis include knowledge that needs to be used for the analysis. Therefore, there is a considerable amount of dependencies between the tasks and resources in the whole process. (Malone and Crowston 1994). In this study the coordination theory is applied only to the perspective of communication and dependencies between the actors in the software development team.

The first dependency in coordination theory is "task–resource dependency" which means the resources are consistently defined for each particular task (Malone and Crowston 1994). The key actors in this dependency are the software developers and management, who typically are responsible for the resource allocation in software development companies. The second dependency is "Producer–consumer-dependency" means ensuring consistency between the requirements and the outputs of the software development. In this process, communication and different communication types and technologies are significant mechanisms to ensure that the output of software development process is usable by the customers (Crowston and Kammerer 1998; Malone and Crowston 1994).

The third dependency in coordination theory is the link between the task and subtasks, for example, the decomposition of the task of writing the requirements into units of work that a single developer can perform (Malone and Crowston 1994). This dependency involves the customers, management and software developers. The final dependency in co-ordination theory is the dependency between features and requirement independences. Interaction between features is a challenge, especially, for the large systems which demand continuous analysis and control of how each feature interacts with other features in the system (Crowston and Kammerer 1998). The actors in this process are the customers, management, software developers, technical architects and quality engineers (Table 1).

Communication is important for distributed, global software development, but also for inter-organizational relationships and as a strategy to manage dependencies (Malone and Crowston 1994). This is because it is a way of helping people avoid conflicts and achieve

Table 1 Communication and coordination in software development; dependencies and actors

Number	Dependency (Crowston and Kammerer 1998; Malone and Crowston 1994)	Actors
1	Task–Resource dependency	Management, Software Developers
2	Producer–consumer dependency	Customers, software developers
3	Task–subtask-dependency	Customers, Management, software developers
4	Feature-and-requirement interdependency	Customers, management, software developers, technical architects and quality engineers.

goals. Table 2 lists the different communication types, categories and techniques in the communication literature (Goles and Chin 2005).

From Table 2 it can be seen that both Herbsleb and Mockus (2003) and Kraut and Streeter (1995) divide communication into two types which are informal and formal communication, emphasizing the meaning of personal level, informal, ad-hoc communication and its support in the co-ordination as a mechanism of achieving success in software development. Formal communication refers to explicit communication such as the specification documents (Herbsleb and Mockus 2003), and the status review meetings in software development (Kraut and Streeter 1995). Informal communication refers to explicit communication via conversations among the workers in the companies (Herbsleb and Mockus 2003).

Carmel and Agarwal (2001) define the communication as a mediating factor which effects both the coordination and control activities in software development projects. Informal communication sharing in software development projects is often based on existing communication technologies such as telephone, video, audio conference, voice mail and email (Henttonen and Blomqvist 2005; Smite 2006). Usually, regular although informal, face to face communication is presented as the best way to build trust among the people in groups and therefore maintain good productivity in the software intensive companies (Henttonen and Blomqvist 2005). For example, Espinosa and Carmel (2003) have stated that there is evidence which suggests that software teams working in the same rooms achieve significantly higher productivity than the teams that are not co-located. This is because the co-location in the same room bolstered collaboration by facilitating continuous interactive communication (Espinosa and Carmel 2003). Paasivaara and Lassenius (2003) present several lower level mechanisms to manage communication and

Table 2 Communication types and techniques

Communication types	Communication techniques
Informal	Face to Face discussions in co-located or distributed teams (Espinosa and Carmel 2003)
	Informal discussions through telephone, video, audio conference, voice mail and email (Henttonen and Blomqvist 2005; Smite 2006)
Formal	Group meetings such as Weekly meetings, Steering group/milestone meetings
	Status Meetings at which the personnel present the project results (Paasivaara and Lassenius 2003)
	Formal meetings through telephone, video, audio conference, voice mail and email, progress reports (Henttonen and Blomqvist 2005; Smite 2006)
	Formal Documentation, for example specification documents (Herbsleb and Mockus 2003)

collaboration issues in the information system development field and suggest, for example, the synchronization of the main milestones and informing and monitoring (i.e. weekly meetings, progress reports, steering groups) as ways of communicating and collaborating in projects (Paasivaara and Lassenius 2003).

2.2 Agile Practices

Agile Practices have recently been recognized in many software companies as a mechanism for reducing costs and responding to changes in dynamic market conditions. The existing agile practices formulate several agile software development methods such as Extreme Programming (Beck 2000), SCRUM (Schwaber and Beedle 2002) and Crystal methodologies (Cockburn 2004). All of those follow the 12 agile principles for agile software development: iterative cycles, early delivery of working software and simplicity as defined in the Agile Manifesto (Beck et al. 2001). Many of the agile principles have been proposed to increase communication in companies as they 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. These agile principles are implemented at both project and organizational levels so as to create the adaptive products that are easier and less expensive to change, and adaptive project teams that can respond rapidly to the changes in their project's ecosystem. In this way the level of organizational agility may increase which means an increase in "the ability of both to create and respond to change in order to profit in a turbulent business environment" (Highsmith 2004). Thus, the agile practices which are the focus of this research provide a way of following the agile principles and contributing more efficient face-to-face communication in software development pilot teams and their stakeholder groups.

Agile methods cannot be used per se in complex organizational environments. This means that the adoption of methods in practice always involves customization and tailoring to some extent. XP and SCRUM are the most commonly adopted agile methods in companies (use of XP, see Drobka et al. 2004; Grenning 2001; Lippert et al. 2003; Murru et al. 2003; Rasmusson 2003; Svensson and Höst 2005; Williams and Cockburn 2003; use of SCRUM, see Dingsoyr et al. 2006; Rising and Janoff 2000). Many organizations integrate the different agile methods in their own customized agile software development approach (Fitzgerald et al. 2006; Sutherland 2001; Vriens 2003). Few studies have, however, attempted to examine the communication in the projects using agile practices. Consequently, a systematic and insightful understanding of communication in agile software development teams and between the developers and stakeholders is still missing from the field. XP and SCRUM are the two methods under focus in this study. The practices used in this study are based on the literature of Beck (2000) and Schwaber and Beedle (2002).

2.3 Framework for the Focal Study: Communication and Agile Practices

Generally, theories are constructed from specific critical bounding assumptions including the key components which are the constructs, variables, propositions and hypotheses (Bacharach 1989). "Values are the implicit assumptions in which theories are bounded" and should be explained in the research study (Bacharach 1989). This research focuses on the communication in agile software development teams. The value of the built framework lies on the assumption that communication is a significant factor which is also one of the key principles in agile software development. For example, Highsmith and Cockburn (2001) mention that informal communication between stakeholders is a significant people-factor

for developers in agile software development. Beck and Andres (2004) claims that there is a need to maximize communication between stakeholders through direct interpersonal communication.

The first variable in the study is the internal communication in the software development teams. Another variable is the external communication which in this case means the management of dependencies between the actor's development team and other stakeholder groups in the organization. Using the proposed background analysis as a basis, the case research team developed a conceptual framework for this focal study. Figure 1 presents the framework in this study which illustrates the variables, constructs and their relationships with agile practices in a continuously changing organization.

Whereas practices can be used as intellectual bins in coding structure (Miles and Huberman 1999), the use of agile practices (XP and SCRUM practices) can be defined as one of the key variables in the case framework. As shown in Fig. 1, an assumption of this study is that the use of agile practices in software development facilitates the transfer of knowledge and should beneficially affect the software development process which is based on communication. The shift towards more efficient communication is achieved by changing the communication from formal plan-driven to informal discussions between individuals, using various communication channels such as planning games, project reviews, stand-up meetings, and pair programming (Turner 2003). While the use of agile practices often heralds many benefits (Svensson and Höst 2005; Cohn and Ford 2003), some negative impact (hurdles) on communication between the actors has been identified (Murru et al. 2003; Cohn and Ford 2003; Svensson and Höst 2005). This leads to the assumption/conclusion that the use of agile practices can both improve and hinder communication between the actors in the case framework in Fig. 1.

According to the coordination theory, external communication consists of the constructs of dependencies between the different actors in software development. Figure 1 describes

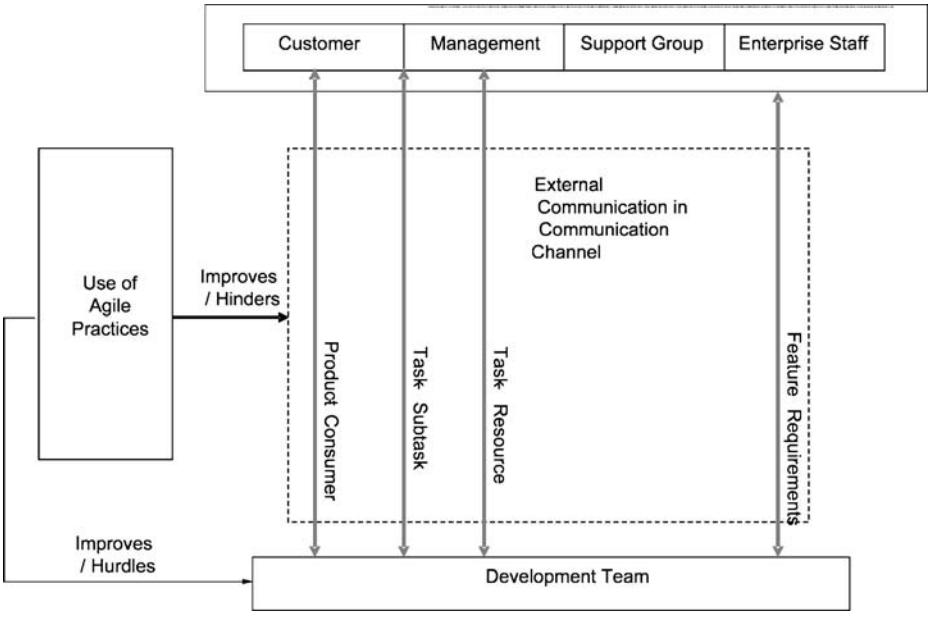

Fig. 1 Conceptual framework for the focal study

the actors and the dependencies between the case development team and stakeholders. It shows, for example, that product consumer dependency mostly effects the communication between the customer and the development team. Task and subtask dependency effects mostly the communication between the management, customer and development team. Whereas task and resource management mostly involves the management and development team, and feature and requirements dependency effects the communication between all these actors.

3 Research Design

The research presented in this paper is based on a case study and the data was mainly based on a combination of the different data collection methods, such as semi-structured interviews and observations as suggested by (Eisenhardt 1989). The data was analysed using the qualitative data analysis method of Miles and Huberman (1999).

Generally, research can be characterized as either interpretative or positivistic from a philosophical perspective (Klein and Myers 1999). This research is interpretive because it follows the principles of interpretive research study as described by Klein and Myers (1999) as follows:

- Contextualization: the research is done in an organization in which the use of agile practices can be defined as the moving target which change and cause continuous change in communication.
- Interaction: the facts are produced during interaction between the researcher and the participants in the study.
- Dialogical reasoning: original preconceptions are continuously updated based on the data found during the research process.
- Abstraction and generalization: development concepts, the generation of theory and specific implications are implemented.
- Multiple interpretations: the influences of the social context with multiple viewpoints and reasons are explained.
- Suspicion: the researchers have a critical approach to interpreting the research data, both the positive and negative impacts of agile practices on communication were studied.

To do research on communication, the unit of analysis in this research has to be first defined and that is the actors or actor groups who share the information in the software organization.

Actors or stakeholders of software development teams have been defined in many of the existing research studies. For example, with regard to coordination theory Malone and Crowston (1994) discuss actors in information transferring and coordination. The key actors who can be raised from their paper are customers, managers and individual programmers or group of programmers. Boehm (2003) presents an approach of value-based software engineering focusing on the actors and success models in the most frequent project stakeholders. He identifies the most relevant stakeholders of software engineering as developers, managers, users, maintainers, sales people and acquirers (Boehm 2003). Leon (1995) categorizes different stakeholder groups for system development defining also two other stakeholders who affect the work of a development group. These are other development teams who have interfaces with the development team and support staff in an organization, for example, the quality engineers that facilitate the team to ensure the quality of

an end product. Based on these definitions, there are five key stakeholders in software development in companies which can be defined as the unit of analysis in this study:

- "The Software Development Team" is the key unit of analysis of this study and it consists of a team of developers who are responsible for software development and the team leader who is responsible for facilitating project information to the stakeholders in an organization.
- "The Management" group is responsible for defining the strategy for an organization.
- "The External customers" can be characterized as receivers of the developed products, they can be actual customers, users or sales group who are going to sell the ready product.
- "The Enterprise Staff" are the technical staff in the company who affect and are involved in case projects or who are going to maintain the developed product (e.g. other developers who develop product components with interfaces between them others).
- "The Support Group" is a group of people from different parts of the company responsible for supporting software development projects (e.g. quality engineers or technical architects who facilitate the product development, testing and ensuring the quality of the process and end product).

3.1 Case Study Research

Since the this research aims to investigate a contemporary phenomenon in a real-life context, case studies were chosen as a suitable research approach, because a case study is appropriate in research situations where behaviour cannot be controlled and research data are collected through observation in an unmodified setting (Yin 2003). The purpose of this case study was to evaluate the impact of agile practices on communication in software development teams and in F-Secure. The case study approach was implemented iteratively based on Yin's (2003) steps for a case study research method, i.e. preparing data collection, collecting evidence, analysing case study evidence and reporting case studies.

The choice of project for the case study was made based on both F-Secure's interest to improve the software development process in which they used the customized practices of SCRUM and XP, and the researcher's interest in understanding how agile practices affect on communication in software development teams and organization.

3.2 Research Context

The research in this study was done in F-Secure Corporation during a one year time period. This section describes the research context, including the background of F-Secure and the two software development projects which were selected for analysis.

F-Secure is a corporation which produces products to protect consumers and businesses against computer viruses and other threats from the internet and mobile networks. This means that the reliability of the produced software is significant for the company success. F-Secure products include antivirus, firewall and network control solutions for internet service providers. Although the company itself is located in Finland, in Helsinki and Oulu, it also provides services through several operators in the global market.

F-Secure had previously used a plan-driven process model for product development. The potential of agile methods was noticed by the management during the year 2004. As a result

the management of F-Secure made a decision to organize training in agile software development methods for the developers and project managers. The purpose of raising agility awareness was to apply agile methods to shorten the development time and increase the quality of all the developed products as well as response to the demands of the dynamic market. The assessment was conducted in four software development projects, but Project 1 and Project 2 were selected as foci for this research because they shared factors which made them more easily comparable. Both Project 1 and Project 2 were the first projects where agile processes had been adopted in the organization. Additionally, similarities in culture and organizational structure and the historical plan-driven software development and communication mechanisms in both projects offered the possibility to compare the differences in the agile practice effects on communication mechanisms inside the pilot project team, and between the pilot project team and stakeholder groups in Table 3.

Project 1 developed a security management system. The project team consisted of six persons including a SCRUM master, four software engineers, and the quality engineer who were working in an open office space according to the SCRUM method. The product that Project 1 was developing was quite complex and the total duration of the development was 1.5 years. In addition, the project had a big stakeholder group including (1) the product owner, (2) three customers, (3) a delivery team responsible for the production of needed documentation for the product delivery, (4) another development team which was developing another part of the same product, (5) a system testing team responsible for the overall system testing of the whole product and (6) a group of top managers who were interested in the overall progress of the project.

Project 2 was carried out in co-operation between F-Secure and the research organization. The goal of the project was to develop part of a system enhancing mobile device secure capabilities which was critical for the company competitiveness. A key goal of the project was to deliver a high quality product to the market as fast as possible. This was because the competitors were already in the market with a similar product. Before the project was started several subcontractors, who had experience of a similar product, estimated that the implementation of the product would take seven man years. The need for faster production was one reason the management made a decision to apply a combined XP and SCRUM method throughout the project. The customized agile method, with practices based on XP and SCRUM, was systematically adopted and further adapted during the development using a Post-Iteration-Workshop method (Salo and Abrahamsson 2006) and an external facilitator. The core of the case project consisted of four software developers and two quality engineers who worked in an open office space. In addition the project had (1) a product manager, who was responsible for the overall product, (2) a technical architect

Table 3 Description of the evaluated projects

Project	Product	Project team	Method	Life cycle (months)
Project 1	Security system management tool	Four software engineers, a Project Leader (later on, SCRUM master) and the Quality Engineer, one month sprints	SCRUM, XP	18
Project 2	Mobile security application	Four developers, one Project Manager and Quality Engineer, 1–2 week iterations	SCRUM, XP, post-iteration-workshops	4

who made an architectural baseline before the product development (3) a system testing group responsible for the overall testing of the product and (4) a top management group interested in the progress of the project. The project team conducted five 2-week software development iterations. The team leader of the project was an expert in agile software development process and came from the research organization. Thus, the team had constant support and coaching available on the adoption of the new agile process model and practices.

3.3 Data Collection

The data for this study was collected during the two research cycles. The main data collection methods were semi-structured interviews and group interviews which were all tape recorded and transcribed. Other sources of data used in the research were documentation, observations and physical artefacts (Table 4).

The documentation reviewed during the research process was the typical documentation of agile software development projects, such as product and sprint backlogs, paper worksheets from the 17 project workshops and project tasks on the walls of the open office spaces. Furthermore, 54 daily status reports of the Project 1 were all analysed in order to gain an understanding of the software development phases and decisions. In this study, direct observation was carried out in the project two post-mortem workshops, three group interviews were held inside the development team. The semi-structured interviews were based on a pre-defined list of questions in 2-h interviews with developers, project leaders and quality engineers. Additionally, the research team members were able to visit the open office spaces of both projects.

3.4 Data Analysis

The data has been analysed in two stages. First both of the selected teams were analysed as a stand-alone entity ‘within-case analysis’ as described by Yin (2003). Secondly the case data were compared by examining the differences and similarities between the two case projects so as to gain a more sophisticated understanding of the selected cases as described by (Eisenhardt 1989).

Coding is one way to analyse research data and the code can be used on different levels of analysis to draw together data and, therefore, permit analysis (Miles and Huberman 1999). The case research is based on the framework building approach in which the

Table 4 Data sources in the research

Data source (Yin 1994)	Data sources in this research
Documentation	Product backlogs, paper worksheets of 17 iteration retrospective and reflection workshops, project data on the wall, 54 daily status reports
Archival records	–
Interviews	Six semi-structured interviews, all of them were tape-recorded and transcribed, several face-to-face discussions with the project leaders
Direct observation	Observation in two project workshops
Participative observation	Two group interviews (active facilitation)
Physical artefacts	Visiting the open office spaces

researcher identified 'bins', named them and clarified the interrelationships between them as described by Miles and Huberman (1999). The 'bins' can be events, settings, processes, practices or theoretical constructs (Miles and Huberman 1999). The framework in Fig. 1 describes the main areas to be studied, constructs of the research and key factors between them.

The coding structure adopted in this research consisted of two distinct mechanisms. Firstly, agile practices were used as 'bins' to define an ID code for each interviewee, to make the first segmentation and filter the interview data collected. Secondly, pattern coding was used as a way of grouping the summaries of previous coding in the smaller number of themes or constructs as described by Miles and Huberman (1999).

During the analysis, it became evident that the key challenges in the software development in the evaluated case projects centered on the communication aspects. This was surprising because the use of agile methods follows agile principles that are said to improve communication. Therefore, communication was selected as a key theme for the pattern coding, while the constructs of coordination theory created new 'bins' for data analysis,

In the pattern coding phase the research data were analysed from the communication perspective and especially with regard to how the developers and project leaders defined agile practices to improve or hinder/create hurdles for communication. Subsequently, the analysis focused on the internal and external, formal and informal communication as per the current literature and so as to make the analysis clearer. In pattern coding the improvements and hurdles were further categorized and mapped between the developers and others, as described in the research framework in Fig. 1.

4 Empirical Analysis

This section evaluates the communication in the case agile software development. The analysis was done by comparing two case projects in F-Secure which applied both XP and SCRUM practices using the conceptual framework presented in Fig. 1.

4.1 Use of Agile Practices

During the agile software development, both the pilot project development teams adopted many informal communication techniques. All the agile practices were deployed in a limited number of projects in the organization, except metaphor and on-site customer practices which were not used in the evaluated projects. An open office space was adopted in both projects which meant that the developers were working in the same room. Both projects also had daily meetings in which they discussed the project status and further activities.

In Project 1 the iteration length was one month while in Project 2 it was one to three weeks, which means that the iteration planning and iteration reviews were organized more often. Pair programming was applied in both cases but not regularly. A storyboard or prefer task board were used in the first phase of Project 2 and at the beginning of Project 1, but were discontinued following reflection and post-iteration workshops in both projects which were held regularly to improve the agile based software development process and tailoring of the agile practices being used.

Test driven development was optional, but not regularly used practice in both of the case project teams. Continuous integration was applied in more depth in Project 1 than in Project 2, whereas Project 2 used user stories but Project 1 did not. Additionally, Project 1 adopted

a mailing-list practice in which they communicated continuously with the whole team, instead of by individual e-mails. Project 2 used also plan-driven daily status e-mail practice in which the project status was reported to the external stakeholders by email at the end of each working day.

4.1.1 Positive Impacts on Internal Communication

This section analyses how agile practices improved internal communication in the development teams. The analysis is made through the selected dependencies in coordination theory based on the framework presented in Fig. 1.

Feature and Requirements Dependency Feature and requirement dependency can be defined as "the determination how each feature interacts with every other feature in the system" (Crowston et al. 2004). The product backlog was used in both projects as a tool to define features for the overall product. In Project 1 the interactions between the defined features were further analysed from the sprint backlog during the SCRUM planning meetings. Interactions between the features were well understood at the beginning of the project but the analysis and understanding of the interactions became complicated as the amount of features increased, according to the product backlog.

In Project 2, the features were defined in planning game meetings as stories in the task board. According to the developers, stories bring customers closer to the actual development and helped to understand previously defined features:

> "Stories were a mechanism to bring customers closer to the development and reveal the core requirements," said one Developer in Project 2.

Similarly to sprint planning meetings, iteration reviews were understood to provide also a channel for formal communication in the teams and a mechanism to facilitate dependency between features and requirements in Project 1 by providing a means of communicating requirements, working tasks and the project status also inside the team between the Developers and the Project Leader. In Project 2, the so-called releasing "ceremony" of the results of each iteration had the same affect among the team members.

Task-Subtask Dependency Task-subtask dependency is in coordination theory defined as one which:

> ...occurs when an individual or group decides to pursue a goal, and then decomposes this goal into activities (or sub goals) which together will achieve the original goal (Malone and Crowstone 1994)

Both the Management and Developers in Project 2 commented that the short iterations would be the most beneficial of the used agile practices in the project, because they facilitated "task dropping" and formulation of "man size tasks" (Developer, Project 2). In Project 1, iteration length was longer, but the comment of the Project Leader was that the iteration length was suitable because it was sufficient time to make enough features for the visible 'demo' of the product for the iteration review meeting.

Thus, the iteration in both agile software development projects started with an iteration planning meeting (sprint planning in Project 1 and planning game in Project 2) in which the purpose was to communicate the project requirements and to define goals for the subsequent iteration. Whereas Project 1 used sprint backlog as a tool for requirements and

task definition, Project 2 wrote the tasks on the wall of the open office space using task board (based on the story board in XP and project information on the wall; Beck and Andres 2004; Cohn and Ford 2003; Shukla and Williams 2002). When the iteration started, the task board was used as a tool/media type of information sharing technique between the Developers and the Project Leader in Project 2. In that case, the product backlog, working tasks and working processes were continuously tracked using the storyboard tool. In the project post-mortem, the development team, managers and other stakeholder groups commented: "It is easier to divide work into tasks than use cases when using the story board" (Developer, Project 2). Additionally, the storyboard was seen to increase visibility of common, short term goals inside Project 2.

As mentioned before, in Project 1, the task board was used at the beginning but discontinued in the middle of the development. This was because of the use of sprint backlog (i.e. excel sheet including project tasks and estimates for the next iteration) which in that case was evaluated as a valuable practice for project daily status evaluation in iteration retrospective meeting.

The open office space was regarded as an efficient way of improving informal communication during the actual development in both case projects:

> ...the team should sit together... it has been quite a success. Not a total success, because it still has interruptions, but it is still better than in our previous projects (Developer, from Project 1).

Especially in Project 1, it was said to be effective in stimulating face-to-face discussion and facilitating design problem solving which were also factors facilitating the team's achievement of goals of the defined iteration. As a consequence of the use of open office space, the Developers in Project 1 argued that there was a decreasing need for documentation due to the use of this practice:

> ... we really questioned who is going to read this, why are they going to read this, how often are they going to read this... A lot of the time we felt that the answer was no. So we didn't document it (Developer, in Project 1).

This was because the open office space was seen as a way of providing a more efficient discussion and solution forum, on a daily basis, of the project goals and status among the development team. Furthermore, the agile principle says to document only when needed.

Both case projects had daily meetings at the end of each day. The daily meetings in Project 2 were short wrap-ups in which the goal was to quickly clarify the current status and plan the next stage of the project from the goal and work task perspective. In Project 1 the daily meetings were evaluated as more effective than the previous weekly meetings. The developers claimed, for example, that the new way, compared to their previous experience of this kind of communication—in plan-driven projects, daily meetings—helped to resolve design problems more quickly. One of the Developers claimed that "most of the problems that we solved in the daily meetings were the design problems" (Developer in Project 1).

However, this opposes the idea of the daily meeting practice described by Schwaber and Beedle (2002) which indicates that the daily meeting should not be the place for design problem solving and the purpose is rather to quickly define the project status and tasks for the next working day. On the other hand, one Tester, in Project 1, thought that "the daily meetings do not offer enough information for testing" or for communication between the Tester and the Developers. Her argument was that the testing schedules are out of line with the daily meetings and additional technical meetings between the Developers and Testers are still needed in the project. In Project 2, the importance of the daily meeting increased,

especially, in that project context where two of the developers in the team were transferred to work in a separated office. In that situation the project team started to use daily meeting to go through and re-plan work tasks for project to clarify the project goals and status.

As a summary, short iterations, sprint planning, open office space, daily meetings in both projects—and task board in Project 2 and sprint backlog in Project 1—worked as mechanisms to facilitate task and subtask dependency, especially continuous 'bottom-up' goal identification among the development teams. This is because in agile software development, the team sits together in daily discussions and task definition activities that enhance both efficient information transfer and coordination as a routine part of the work. In the case agile software development projects, the teams always had the practices to hand that supported continuous evaluation of the goals of development. This type of 'bottom-up' approach to manage task and subtask dependency should, according to Malone and Crowstone (1994), have positive effects on the motivation of the actors in software engineering and effects on the success of software development teams.

Producer–Consumer and Task Resource Dependency Producer–Consumer dependency can be defined as

> Dependency in which task creates a need for resource which is again needed to develop another task (Malone and Crowstone 1994).

In Project 2, tasks were defined by the development team which also defined who is going to develop the task and handle the interfaces between them. In Project 1 the,

> customers and product owner defined high level tasks in the sprint planning meeting and then the Developers tried to estimate what were in those tasks (Developer, Project 1)

The open office space, task board and daily meetings were also practices that were interpreted to support the communication related to resource definition inside the teams. The Developers also stated that, compared to the previous plan-driven project, the Project Managers no longer had such a big role in task and resource allocation as in plan-driven software development projects.

Task-resource dependency means the resources (e.g. money, storage space, or an actor's time) are consistently defined for each particular task (Malone and Crowstone 1994). From a communication perspective, the important issue is how the communication is shared related to resource allocation e.g. scheduling of tasks in the agile software development projects. In both agile projects, the resource and schedule allocation for the work tasks were made in sprint planning meetings in co-operation between the Developers and Quality Engineers.

> First in the sprint planning meeting, we had all the stakeholders involved, when all the priorities were agreed, we defined the tasks and estimates for each task together with the team (Quality Engineer, Project 1)

Compared to plan-driven projects, the responsibility for task-resource dependency in an agile project is transferred from the Project Leader and Managers to a self-organizing team and communicated continuously along the line. Features, requirements and sprint backlog in Project 1, and stories task board in Project 2, were used as tools to handle the resource allocation in an iterative manner. Management of the scheduling and resource definition continued in both projects in daily meetings.

Key Findings Both the Developers and Project Managers claimed that the agile methods, such as open office space, daily meetings and iteration planning, would offer improved internal communication inside the development team and between the Development Team and the Project Leader. Table 5 summarizes the positive impacts and dependency effects arising from the different communication practices in the evaluated projects presented and discussed in this section.

Reflection workshops (i.e. post iteration workshops in Project 2) were used as a way of communicating problems related to the agile practices. In Project 2, reflection workshops were a mechanism for external communication between the development team, management and customers, whereas in Project 1 they can be characterized more as a mechanism for internal communication inside the development team. In both cases, however, reflection workshops had only a minor effect on the dependencies in the coordination theory. Thus, the workshops as a communication mechanism had little effect on the features, requirements or work tasks in the project.

Pair programming and continuous integration supported the tool media type of communication in both teams. The Developers felt that the pair programming was not actually a programming technique but the most efficient way to review code: "…the only real way to review code is actually by pair programming," (Developer, Project 1). This might be due to the fact that the pair programming was not used in the project regularly, on a daily basis as suggested by the eXtreme Programming method itself. In Project 2, pair programming was continuously reported as a problem in project reflection workshops. It was used as an optional practice and could not be used as an efficient way to facilitate communication about project status or goals. Continuous integration worked as an efficient mechanism for communicating the project current status to the tester group and thus making the system testing activities easier in the project. One tester noted continuous integration as

Table 5 Summary of the communication techniques used, their positive impact and the dependencies

Agile practice	Positive impact found	Dependencies
Open office space	Need for documentation may decrease, everybody had knowledge of the project status and common goals	Task–subtask
Daily meetings	Good way to keep the Developers, Project Leader and customer (in some cases) aware of the project status	Task–subtask, task–resource
Story/task board	Project status information on the wall allowed everybody see the project status at one glance	Task–subtask, task–resource
Iteration planning (sprint planning/ planning game)	Project status was well-managed and the whole project team was aware of the project plans and goals for the next iteration	Task–resource, producer–consumer, feature-and-requirement
Reflection workshops/ iteration retrospectives	An efficient way to deploy and improve agile practices	–
Pair programming	Pair Programming was defined as an efficient way to implement code review but difficult and problematic practice for daily use	–
Continuous integration	Facilitated testing, helped Quality Engineers to get information on status of the end product	–

a helpful and positive agile practice "Continuous Integration was great as I had the new build available for testing purposes all the time," (Tester in Project 2).

As shown in Table 6, reflection workshops, pair programming and continuous integration were evaluated as follows:

- Reflection workshop: as a mechanism to improve software development work
- Pair programming: as a way to review code
- Continuous integration: as a mechanism to provide new build for testing purposes

They were not evaluated as practices that would affect any dependencies in projects.

4.1.2 Negative Impacts on Internal Communication

This section discusses how agile practices can hinder internal communication inside the development teams. The analysis was done using the dependencies in coordination theory and based on the framework presented in Fig. 1.

Feature–Requirement Dependency Interactions or interfaces are often a problem in large software products, especially when there are hundreds of features supported by different people, it is difficult to evaluate and implement interfaces between features and requirements (Crowston et al. 2004). This was a problem, especially in Project 1, because according to the Developers:

> The features in product backlog weren't properly organized, they weren't prioritized... and sprint planning meetings were too short considering the huge amount of features and requirements (Developer, Project 1)

Due to the number of new customer requirements from three customers, the requirement changes appeared all the time during the project. Although the SCRUM master and product owner clocked up a lot of hours on product backlog refinement and requirements analysis, the team still felt that the new features were included in the product backlog too easily without enough detailed technical analysis.

Consequently, all the feature sets could not be completed during the project and the overall focus was often wobbly.

> ...Whoever yelled the most got what he wanted. As a team we should talk more about some things. There were some features which were unreasonable to produce...we made some useless features....technical solutions should be thought through in more

Table 6 Internal communication hurdles in the case pilot project teams post project workshops

Agile practices	Hurdles	Project 1 (% of responses)	Project 2 (% of responses)	Dependencies
Open office space	Hard to focus	100	5	Task–subtask Producer–consumer
Product backlog	Overall focus of the project	53	4	Feature-and-requirements
Product backlog and sprint planning	Long term visibility	76	5	Task–subtask, feature-and-requirement

> detail in some situations. If one customer wanted some function, they were made even though they are not needed by any other customers... (Developer, Project 1)

Thus, it seems that the agile practices, in this form, were not a good enough solution especially to handle the feature–requirements dependency in Project 1. In spite of the fact that SCRUM planning and review meetings worked well in the context of a smaller number of features, the problems appeared when the amount of features were increasing from dozens to hundreds.

These hurdles in the communication affected also the long-term visibility of the project goal and customer focus in Project 1 in Table 6. The Developers argued in the post project workshop that the goal of producing a working version of the system for iteration reviews was a good way of noticing and communicating integration problems that would otherwise (in plan-driven projects) be communicated only after the overall implementation.

As shown in Table 6, the long-term feature–requirements dependency was defined as a hurdle also in the Project 2 post-mortem. However, no effects on the long term focus were reported. This might be the result of the shorter project duration, shorter iterations and regular iteration planning and reviews in Project 2, which happened once every one or two weeks, compared with Project 1 which held them once a month.

Task–Subtask Dependency From the long-term project goal perspective, an open office space was characterized as a disadvantageous mechanism for interpersonal and group management type of communication in both development projects. In Project 2, the resistance of a few software Developers affected the opinions of the other Developers in their opposition to the open office space as well as efficient communication practices. The open office space as a practice was criticized also in Project 1:

> There are always those times that other people are holding a meeting in the open office space which affects the focus (Developer, Project 1)

This is mainly because these highly experienced developers had their own ways of doing their work in their own rooms and they were not willing to change their ways because of the agile practice deployment. Although all the Developers in Project 1 felt that it is sometimes difficult to concentrate in an open office space, it was judged to be more beneficial than problematic in the project.

In Project 2 the Developers showed resistance to an open office space during the first month of development. Due to the comments, such as: “an open office space is too noisy, it takes too much space and makes concentration difficult” (one Developer, in Project 2) the management made a decision to give up on the open office space practice and turn back to the separate rooms. It took, however, only one iteration when the change was evaluated as “harmful for the user interface design and the other communication needed in the design activities,” (a Team Leader, Project 2) and it was later reported in iteration retrospective meetings that the “project had problems in collaboration and scheduling in user interface planning” (Developer, in Project 2).

Key Findings Based on the case analysis, it was suggested that sprint planning and product backlog management were practices that affected negatively also the internal communication between the Developers and Project Leader in Project 1. This finding, however, related only to the one project and, therefore, might be owing to context factors such as the complexity of the system and number of customers, new requirements and change requests in the project.

The communication hurdles were also reported to be inside the projects and due to the open office practice, which was much criticized by the Developers in both projects. However,

this runs counter to the Developers' comments in Project 2 who felt also that the lack of open office space practice caused immediate problems in collaboration and communication of design inside of the development team.

4.2 The Impact of Agile Practices on External Communication

The main agile practices affecting the communication between the development team and stakeholder groups were iteration planning meetings, iteration reviews, daily meetings, iteration retrospectives and refused formal tool/ media type communications.

In Project 2 some additional tool/media-type of communication mechanisms were used that had not been applied in Project 1. For example, e-mails on the project status were sent to the top management always after the daily meetings, and post-iteration workshop results were sent to the top management twice a month.

4.2.1 Positive Impact on External Communication

Improvements due to the change towards agile-based software development were evaluated in a workshop, together with the pilot project team and stakeholder groups in both projects. This section analyses how the agile practices improved external communication in the two case projects applying coordination theory.

Feature–Requirement Dependency The key idea of feature and requirements dependency is the coordination of interfaces between the features and requirements and to share information on them to the project team and stakeholders (Malone and Crowstone 1994). Sprint planning meetings and reviews can be evaluated as practices positively affecting the feature–requirements dependency from the external communication perspective. This is because the use of these practices provides a systematic way to share information on the features and requirements between the many stakeholders such as the testing teams, interface projects and customers, as can be seen from the following comment:

> Sprint reviews were very positive, because we could also get a view of what other types of requirements the stakeholders had, but also what was on the stakeholders' minds and what they were thinking about in terms of needs for the service itself (Developer, Project 1)

Task–Subtask Dependency In Project 2, both the Developers and Management stated that

> the short term goals are clearer in agile software development than in a plan-driven way to plan or manage projects.

In Project 2 the project target was defined and analysed as stories for the next iteration in planning games. In that case, a Developer said:

> Stories makes it easier to divide the work into tasks than to use cases, the stories bring the customer closer to the development and reveal the core requirements (Developer, Project 2)

Thus, in both projects, it was reported that the agile practices have positive effects on external communication. On the one hand, in Project 2, the short term visibility of common goal was said to be improved also due to the tool/media type of communication with

regular status reporting and shorter iterations, which enabled more regular communication in the iteration planning and review meetings. On the other hand, in Project 1 the improvements in short term goal evaluation were evaluated to be mainly due to a group management type of communication in the regular iteration reviews, in which a version of the real software product was presented and discussed with the product owner and other stakeholders.

Producer–Consumer and Task Resource Dependency In Project 1 the customers and project owner defined the work tasks in sprint planning meetings before the Developers continued with their estimation. This assured good consistency between the requirements, high and lower level work tasks and also moved the team to examine the time and resources used for each task. The same happened also in Project 1 in which the product manager together with the customer first defined the features and priorities for the product and then the developers defined the tasks in the task board in which the tasks were again reported to the stakeholders as a part of the daily e-mails.

Key Findings As a result of this comparison, it is noted that the use of some agile practices improved the visibility and common knowledge of the short term goals in both projects. As shown in Table 7, the iteration reviews, status reporting were evaluated to improve visibility of the short term goal and focus of the project as well us understanding of the product features and requirements.

4.2.2 Negative Impact on External Communication

This section analyses and discusses the agile practice hurdles on external communication between the development team and stakeholders. The discussion is based on the dependencies in coordination theory and concepts presented in the framework in Fig. 1.

Feature–Requirement Dependency In coordination theory, feature–requirements dependency means the management and coordination of interfaces between features and requirements in system development projects (Malone and Crowstone 1994). In both the case projects, the product owner or product manager was responsible for the customer interface and the team communicated with them daily:

> ...including a development side in the talks with the customer...and...we learnt many good things by listening to the customers....we pretty much worked together in a very co-operative way regarding the definition of the scope, defining features, prioritization

Table 7 Positive impact on external communication

Agile practices	Improvements	Project 1 (% of responses)	Project 2 (% of responses)	Dependencies
Iteration reviews, status reporting	Visibility of short term goal/focus and requirements	23	8	Producer–consumer, feature-and-requirement
Iteration planning (sprint planning/ planning game)	Short term focus	29	23	Task–resource, producer–consumer, task–subtask, feature-and-requirement

> features so on...we also visited the customers together... the Product Manager visited several customers...and we really tried to really build communication with the product owner on a daily basis... (Project Leader, Project 1)

Thus, the communication between the project leaders, product managers and customers was continuous during the project. The limited time used in the sprint planning meetings for technical requirement analysis was not, however, enough for the developers and testers who felt that more time was needed to discuss with the customers especially at the end of the project when the product become more complex.

In Project 2, the stakeholders seemed to be happier with the mode of communication, although some criticism was given in the project post-mortem workshop. One difference between Project 1 and 2 was that in Project 2 additional short reports on the project's daily status were sent by e-mail to the external stakeholders of the project. The opinions of the management and project customers harvested in the project post-mortem workshop revealed that this was a positive way to keep people aware of the overall status of the project. They said:

> Information is now shared better between R&D and the management, and the management knows the status of project all the time (Developers and Managers, Project 1 and 2)

In both cases, however, the testers commented that they would clearly need more information on (1) the impacts of the requirements, (2) the architectural structure of the system and (3) its technical details. Thus, new mechanisms would be needed to coordinate dependency between the features and requirements during the agile software development.

> SCRUM does not work with the Quality Engineers. One problem is that the new features are implemented on the last day and the overall testing of the sprint results has to be done one day after the sprint ends (Tester, Project 1)

> I think there is a clear difference of approach between an agile versus a plan driven project for the Quality Engineers. And also for the other stakeholders and participants, But definitely from the quality side there is a different approach. And I don't think there are many Quality Engineers right now who understand that difference (Developer, Project 1)

In Project 2, 46% of the participants in the project post mortem workshop argued that the use of agile principles and practices hinders communication in general, and 30% said that the use of agile practices would hinder especially design documentation. In that case the Testers stated that: "we would need better documentation for the quality engineering team to work with" (Tester, Project 2).

As a conclusion on the communication gap between the quality engineering team and development team, the Quality Engineers expected the Developers to produce documentation which would describe the needed design information. An absence of communication mechanisms also affected the remote delivery team and development team who continuously required documentation in Project 1 where, for example, the need for architecture documentation was discussed in the sprint reviews. However, those documentation demands were not prioritized enough in the sprint backlog. According to the project management, the project had certain demands which were not sufficiently justified:

> There was a lot of resistance among the stakeholders because the team was not ready to write documents. The team was ready to ask questions of the stakeholders but those

> people were used to getting documents rather than having a face to face discussion… but when they said that they wanted a document they did not specify what kind of information was needed, what was the purpose and who was the audience (Team Leader, Project 1)

> There was a lot of impatience. Then the documentation issue was another one. Until we found a channel to get the information out that other people needed, those people were very unhappy (Developer, Project 1)

As a channel for information sharing, the project Developers organized several workshops with the stakeholders where the Developers reported the content, and the stakeholders themselves made the documents based on the workshop discussions. According to the Project Leader in Project 1, the stakeholders' reactions were due to the cultural change i.e. the information was previously documented, rather than communicated face to face. The stakeholders were said to be confused because they were not aware of the principles and fundamental ideologies of agile development.

Thus, the conflict arose when the stakeholders waited for the documentation while the development team intended only to communicate mainly face to face, according to the agile principles. For example, one case Developer claimed:

> Rather than creating pages that these guys never read, we just said, okay, you can have documentation workshops for a few hours where you tell us the problem areas you don't have information about (Developer, Project 1)

Task–Subtask Dependency Task and subtask dependency in coordination theory describes the way the project team and stakeholders define and maintain the common goal of the project and project working tasks (Malone and Crowstone 1994). In Project 1 the Developers and quality engineers argued that the sprint planning time was not enough for the handling of new requirements and the managers commented that there was neither enough time nor mechanisms to respond to the increasing number of requirements and communication flow between the three customers.

The developers in Project 1 felt that one of the key problems was that the project stakeholders were not aware of the changes:

> I do not think that we had too many problems within the team, problems arose when you are dealing with very slow reacting organizations and other teams who do not want to change the way they are doing the work (Developer, Project 1)

> The stakeholders were people that we had to interface with... Some of them were happy because they could see that the results were there and were delivered on a regular basis. Some of them were unhappy....they felt they have no possibility of really affecting the implementation of the project. This was not true but they felt they were not really part of the development team...this was very hard for many people, because it was against the culture that we usually have in our organization... (Project Leader, Project 1)

In this situation, one reason that left the stakeholders confused was that, although they had good possibilities to discuss with the team members with team daily, they did not have any possibilities to affect the project goals and add tasks to the project during the one month iteration. This is because in Project 1, both the Project Leaders and Developers followed the SCRUM ideology which recommends limiting the change requests and new task creation

between the project team and the stakeholders anywhere else than in the sprint planning meetings. On the other hand, the efficiency of the sprint planning and sprint reviews from a communication perspective depends largely on the stakeholder's awareness of agile principles, agile methods and their possibility to participate in the meetings. For example, in Project 1, the participation in sprint planning meetings and reviews varied during the development among the stakeholder groups:

> The stakeholders changed all the time, but the product owner was there, the business owner was there and an relevant Project Manager was there (Team Leader, Project 1)

All the customers were not able to participate in the sprint planning meetings. Therefore, the product owner was the key person regularly communicating the new requirements and, thus, contributing the customer input to the development team as suggested by the SCRUM method. Similarly, the case company staff and management groups' participation varied in the sprint planning and review meetings. The case company like other members of the development team with the same product interfaces was involved in the meetings only when it really needed some information from the pilot project developers. The top managers participated in the review meetings in order to monitor the ongoing projects.

Project 2 did not report any problems related to the stakeholder participation in the project planning games or releasing meetings. This was probably one factor which facilitated the coordination of task and subtask dependency among the team and stakeholders.

Producer Consumer and Task–Resource Dependency Resource definition was interpreted to be efficient in both development teams and the systematic meetings provided mechanisms also for the stakeholders to know when to participate in the work of the software development project.

In both the case projects a separate testing (i.e. quality engineering) team, however, waited until the end of the project before starting quality assurance activities. Whereas the XP method suggests testing and integrating continuously as part of the project work, the case Quality Engineers were used to doing the key work and, thus, allocating their resources to the end phase of the system development. This confusion in producer consumer dependency in the organization was the consequence of the lack of role definition and awareness of agile methods among the quality engineering group.

Key Findings Both post-mortem observations and interview data indicate that, due to the lack of communication mechanisms between the development team and stakeholder groups, most of the findings related to the interpersonal, group management and task type of communication problems between the team and company staff. Compared to Project 2, Project 1 had more problems with the focus and customer interface which could be due to the longer development iterations. The interface between the pilot project team and testers and the lack of communication were, however, seen as a bigger problem in Project 2 as presented in Fig. 2.

Table 8 summarizes the communication hurdles from both case projects between the Developers and stakeholder groups based on the analysis presented in this section.

4.3 Summary of the Findings

To show the effects of agile practices on communication, Fig. 3 shows the links in more detail, as well as their influence (either positive or negative) on the dependencies in both the external communication and internal communication inside the development teams. The

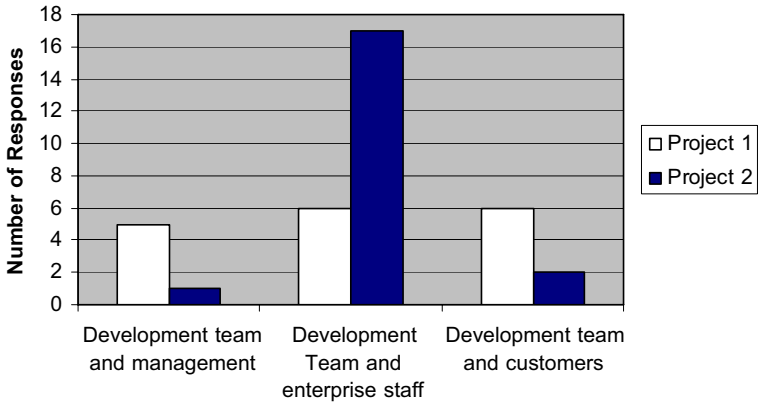

Fig. 2 External communication hurdles in the case projects

links and positive and negative factors are based on the conceptual background presented in Fig. 1. For example, the sprint planning or planning game was interpreted as a useful practice having a positive effect on all the dependencies between the developers and the stakeholders. Only one negative effect was the long term coordination of the goal of the overall product, which was difficult to relate to the dependency between the features and requirements. The daily meetings, open office space and project information on the wall were regarded as a positive practice for the management dependency between the tasks and subtasks during the project. These practices also might have positive effects on the internal communication inside the project teams. Open office space was the only practice that was interpreted to have negative effects inside the development teams. Most of the developers admitted that it is sometimes difficult to focus on that kind of environment, but generally it was evaluated as a more positive way to organize the projects.

Table 8 Communication hurdles in the case projects between the developers and stakeholder groups

Stakeholder groups	Hurdles
Enterprise staff	The team expected formal documented knowledge and the agile team wanted to give tacit interpersonal knowledge on the same topic, this caused a continual conflict between the project teams
Support group	The testing team waited for the project to end before testing activities, one tester involved in the team felt that agile practices (e.g. Sprint Planning and Daily Meetings) did not give enough knowledge for testing (2). Furthermore, testing tasks were difficult to estimate because it is difficult to evaluate how much time the testing takes (2)
External customers	The backlog was difficult to manage (2) which caused a situation in which requirements were not understandable for the Developers and the Quality Engineer, sprint planning meetings were short in relation to the huge amount of requirements (2), new features to be included in the product backlog without detailed technical analysis (1), daily meetings were too time consuming for customers
Management	Any of the development group could participate in the management level meetings, the Managers expected that the Project Leader would still be the key person with whom they should share the information

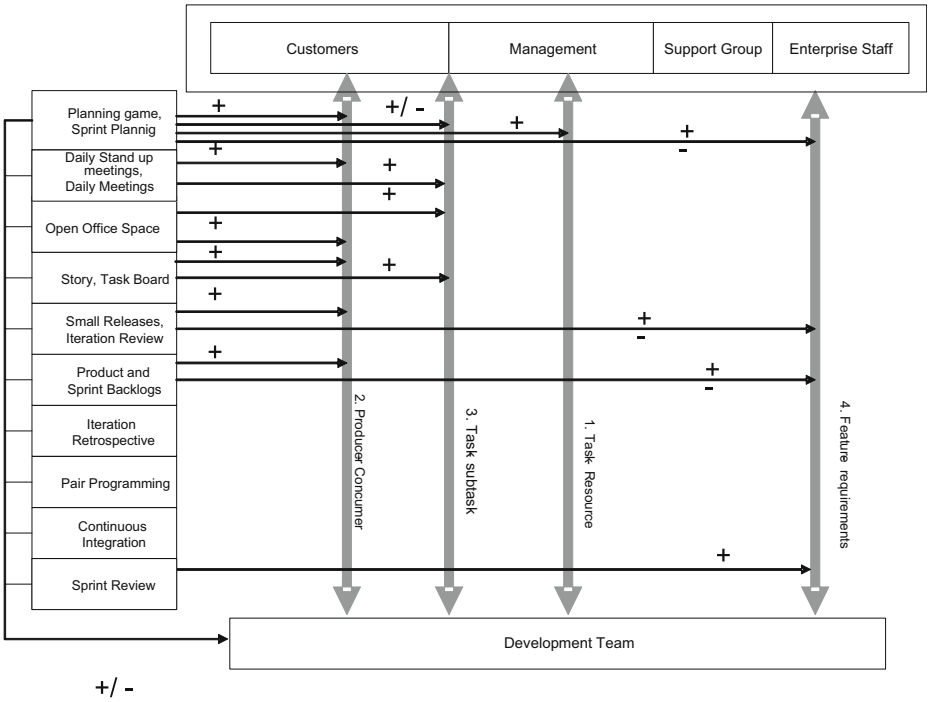

Fig. 3 Summary of the effects of agile practices on communication in case organization

The first finding of the study is that the impact of each agile practice on communication is not on the same level as all the studied agile practices in Fig. 3. It seems that some of the agile practices may have a stronger impact on both team, and the team and stakeholder, communication than the other evaluated practices. Overall, the iteration reviews, product and sprint backlogs were the practices identified as having positive effects on the feature–requirements dependency. Basically, those practices created new systematic ways to communicate between the development teams and stakeholders which also helped ensure that the created product met the demands of the customer and other stakeholders.

Furthermore, Fig. 3 reveals that some of the applied agile practices can even have negative effects on the communication between the teams and stakeholders. This is because the use of agile practices does not only offer new mechanisms to communicate but used as a "by the book" way also limits the communication to some specific, time boxed meetings that may not be enough when the complexity of the system increases. For example, it is suggested that the management of the high amount of requirements or features in the product backlog and tester role in agile software development also had minor negative effects on the dependency between the features and requirements; task and subtasks and producer–consumer. One surprising point arising from this study is that many of the XP practices, such as pair programming and continuous integration, were not regarded as having an effect on the dependencies in team coordination.

Thus, it can be assumed that in the situation in which the organization or team representatives have a clear goal to improve communication or coordination in teams or organizations, it might be best first to adopt only the defined set of agile practices that are evaluated as most beneficial to team coordination and from a communication perspective.

In a second phase, the adoption process could continue to incorporate other practices if they have some other added value for the teams or companies.

5 Discussion

One of the fundamental principles in agile software development is to increase the communication in the project teams so that formal documentation is not a necessity and the Developers can concentrate on producing working software instead of documentation (agile manifesto, Beck et al. 2001). This section presents the practical and theoretical implications from this study with regard to the current literature using the framework presented in Fig. 1.

5.1 Implications for Practice

This section presents the implications from the study for the practice. The discussion presents a reflection on the key findings with regard to the current literature, applying the key elements of the framework in Fig. 1.

5.1.1 The Impacts of Agile Practices on Internal Communication

From a practical perspective, this study suggests that agile practices—such as sprint planning, daily meetings and open office space—have positive effects on internal communication inside software development teams. This view of the positive effects of agile practices was also borne out in the studies of Korkala et al. (2006); Karlström and Runeson (2006). Based on the case analysis in F-Secure, the agile practices used in the development projects can be categorized into two groups based on what type of communication they support (i.e. informal or formal). The assumption that the amount of informal communication would increase in software development teams mainly because of sprint planning, open office space and daily meetings is mentioned by Mann and Maurer (2005), who argue that, for example, sprint planning meetings help to reduce the confusion about what should be developed from both the Developers' and customer perspectives.

This research has found that the use of sprint planning meetings could also hinder the maintenance of long term goals and analysis of interactions between the features as shown in this paper. Murru et al. (2003) and Pikkarainen and Mäntyniemi (2006) argue in a similar vein that the sprint planning meetings might cause the risk that the most demanding customers get what they want and are favoured by this approach, but the decisions are not always analysed in enough detail from a technical perspective. This may affect negatively both the overall project goal and results of the product development as shown in this paper. Murru et al. (2003) notes also that the reason for the problems with long-term goals could be that the metaphor practice from XP was not used in agile software development project. The same situation was understood also by the actors in Project 1 in the case study presented in this paper.

Rasmusson (2003) suggests that the open office space would be one of the most productive working environments for the software development. This was suggested also in this study based on the fact that the amount of informal communication seems to increase inside the teams mainly because of open office space which, according to the case Developers, also decreases the need for formal, documented communication inside the development team. While the open office space was effective in stimulating discussion and improving the pace of problem solving, the case Developers perceived the use of open

office space as distracting, especially, in the situations when they were doing programming at the same time as other Developers had a meeting in the same room.

Karlström and Runeson (2006) argue that detailed practical issues could be identified and resolved earlier in agile software development teams than in plan-driven projects. In this study the Developers argue in favour of daily problem solving in agile software development teams. For example, the Developers' way of holding daily meetings as a mechanism for design problem solving within the definition of the SCRUM method were seen to have positive effects on the communication of the design issues in the case project. This made it easier for the Developers to communicate the design and to affect the re-definition and estimation of ongoing tasks and establish the goal for the next iteration.

5.1.2 The Impacts of Agile Practices on External Communication

The strength of the agile approach lies in its focus on delivering a higher value per unit cost to the stakeholders (Turner 2003), its ability to prioritize requirements and respond to changes in stakeholder value propositions. In both the case projects, there were communication hurdles between the development team and stakeholders. Thus, it can be conceded that the agile approach in software development is not a silver bullet for optimum stakeholder coordination, and the risk (e.g. Turner 2003), of relying too much on tacit interpersonal knowledge and people willing to share it, is also a risk in agile software organizations. As Turner (2003) previously indicated, limited formal and informal communication mechanisms can hinder communication between pilot project teams in the context of agile software development. In the case presented in this paper, the communication obstacles do not, however, occur because of management resistance as reported by Drobka et al. (2004) and Grenning (2001). On the contrary, the case project managers can be characterized as key change agents in both cases. Thus, the case communication obstacles can be attributed more to factors such as the number of customers and number of features in the product backlog, and the low awareness of agile practices among the development team and stakeholder.

Coordination and Communication of Dependencies Challenging Agile Software Development The challenge of XP projects is the same as that defined in coordination theory (Malone and Crowston 1994): i.e. keeping the overall project focus and definition of tasks based on ill-defined features. In the case study, it was proposed that the combination of SCRUM and XP facilitates communication, but it did not solve all the communication glitches between the Developers in the pilot project team and external customer groups who were involved in the sprint planning to have their requirements implemented during the next iteration, at the expense of those who were not present in the planning days. Another problem seen in the case project was the lack of upfront project scope and poor feature understandability in the latter phase of the development project which affected the dependency between the features and requirements in coordination theory.

It has been reported that the use of agile practices improves informal collaboration between development teams and external customers (Cohn and Ford 2003). Thus, it could be a factor that could have an effect on producer consumer dependencies, task–subtask dependencies and feature–requirement independencies in an organization. Cohn and Ford (2003) report also that a consequence of the use of agile methods is improved communication between the developers and top management. The reasons for this are that

approaches such as SCRUM and XP accelerate project cycles, Developers typically interact with their Managers more frequently and for shorter periods.

Sprint Planning, Daily Meetings, Sprint Reviews—Both Positive and Negative Effects on Communication XP and SCRUM practices affect the external communication between the development team and stakeholders (Layman et al. 2006a). In this study, some agile practices such as sprint planning, daily meetings and sprint reviews were seen to have a positive effect on the information transfer between these actors. Rising and Janoff (2000) report similar results from SCRUM projects. They claim that short time boxed iterations in agile software development are the key reason for improved communication in software development teams. Communication does not, however, happen without systematic meetings which bring the stakeholders together to talk about the requirements and project status.

Many argue that interface management between agile and traditional teams is challenging (Boehm and Turner 2005; Lindvall et al. 2004). This kind of problem in the communication channels between the pilot project team and stakeholder groups has been surprisingly widespread among all stakeholder groups (Cohn and Ford 2003; Svensson and Höst 2005). This is partially due to the fact that agile practices seem to lack mechanisms for mass media or interpersonal information sharing between the agile teams and its stakeholders. There are also a few cases that report similar difficulties between the agile teams and company staff, such as testing teams (Lindvall et al. 2004).

Agile practices suggest that testers should be integrated into the agile team, so as to integrate and test the system during its development (Beck and Andres 2004; Cohn and Ford 2003). In the case projects, however, a separately located testing team waited for the project end before commencing quality assurance activities, and the tester involved in the project felt that agile practices did not provide enough mechanisms for communication between the Tester and customers. Therefore, the difficulties are due to the change in the Tester's role in the change towards agile based software development when tests are integrated as a part of the development process (Cohn and Ford 2003).

5.2 Implications for Theory

It has been argued that the current research has only a few case studies on agile software development (Layman et al. 2006b). Therefore, this study has an important role in increasing understanding of the agile software development process from communication perspective. Furthermore, it has been noted that the current research lack of studies which analyses agile practices in use teams and organizations with strong theoretical baseline (Pikkarainen et al. 2007). This study uses the dependencies of coordination theory to analyse the impacts of agile practices in use in communication. Although, this in depth case study focuses only on teams in one organization, it provides a starting point for research applying the coordination theory of Malone and Crowston (1994) in studying the use of agile practices in software development process.

From a theoretical perspective, this study shows that software development has other dependencies than those described in the coordination theory. The case study showed, for example, a lack of dependencies caused by the continuous software process improvement and actual development work which effects, not the working tasks, but the communication between the software Developers during the development process. For example, coordination theory does not describe the dependencies caused by the testing activities or between the separated development teams who are dealing with the same product.

According to the case studies, the problems in communication also affect the dependencies, especially, between the features and requirements as described in coordination theory. The problems within the features and requirements increase in parallel to the amount of customers, which means that the number of customers and other stakeholders can be defined as a factor which effects how much an organization should emphasize plan-driven communication mechanisms in agile software development. The problems described in co-located teams probably increase when the organization moves towards bigger distributed projects, which are the current trend in product development due to the increasing market demands and competition.

5.3 Limitations

Interpretive research assumes that both an organization and the relationships between the people in the organization are continuously changing (Klein and Myers 1999). This study was applied in F-Secure where agile practices were adopted in software development projects for the first time, and where the history of the developers was mainly based on the plan-driven, CMMI based software development process. This provided a good possibility for comparing a new agile based situation with the traditions of the organization. On the other hand, it is important to notice that the presented findings are context specific and, therefore, it can be claimed that the improvements and effects of agile practices on communication are dependent on the specific contextual factors. For example, due to the adoptive situation of agile practices, both development projects were manned by people who were new to agile development, and the fact that this was also new to the organization could be a reason for some of the problems encountered.

During the research process it was noted that the case study work of the researcher also affected future software development projects as suggested by the contextualization principle of Klein and Myers (1999). For example, workshops in which the first author worked as a key facilitator harvesting the information on improvements and hurdles related to the effects of agile practices on communication between the developers, team leaders and stakeholder activities in the case projects. This link between the results and activities of actors in F-Secure is, however, difficult to establish based on the collected research data.

Interaction in interpretative research means that the facts are produced as a part of the interaction process between the researchers and participants (Klein and Myers 1999). In this research, the collaboration between the researcher team and the case company representatives was close and continuous during the overall research period. This enabled good access to the project information and possibilities for collecting evidence from several sources.

According to Klein and Myers (1999) interpretive research should be generalized and based on concepts, generate a theory and draw specific implications on the study. Although, the use of theories might be a weakness in this research, some conceptualization took place in the area of communication and agile practices and dependencies, described by the coordination theory, were used to guide the analysis. Due to the fact that theories have not been commonly used in the research on communication in agile software development, the application of existing theories in further research on communication and agile practices would be valuable. This study makes, however, a theoretical contribution by describing a framework for the case study. In this paper, the framework is used in a company providing a mechanism for, and example of, analysis of agile practice effects on communication and coordination with qualitative research methods. In the future, the created model could be further tested in other action research or case studies with more

literature-based analysis. The framework itself could be further developed and, therefore, used to assess, for example, the success of a development team or communication between a team and organization.

Multiple interpretations in interpretive research mean that the different viewpoints and reasons behind them are clearly described. In this research, the two selected case projects can be characterized as comparable because they both were software development projects carried out in the same organizational context using the practices of XP and SCRUM. The time period of the projects and number of customer groups were different, which can be seen as factors that affected the results related to communication. In fact, it seems that the effects on the working period was not so significant from a communication perspective, but the increased amount of stakeholders increased the need to use alternative, plan-driven practices in agile software development teams.

The drawback to interpretive research means the possibility of seeing consciousness as "false" consciousness sometimes. In conclusion, this research shows that, although agile software development is supposed to improve communication, it does not give the necessary umbrella solution for all communication contexts among developers and stakeholders.

6 Conclusions

Agile methods are an innovative process being increasingly adopted in organizations because they promise organizations the possibility of improving the organization's ability to communicate more efficiently. However, only a few studies have addressed the effects of agile practices on communication.

This research seeks to understand the impacts of agile practices on communication in software development teams and in a specific organization. It sheds light on the communication in agile software development projects and generates answers about what kind of hurdles project teams may face in the communication between agile software development teams and stakeholders such as customers, management and other development teams.

This study indicates that the all agile practices used in the projects had positive effects on the communication inside the development teams. For example, sprint planning, open office space and daily meetings were suggested as efficient practices to communicate requirements, features and project tasks in agile software development teams. In the situations in which these practices were used together, it was further indicated that increased informal communication works as a factor which decreases the need for documentation in software development teams and, therefore, facilitates more productive software development than in previous plan driven situations.

The study confirms that the use of agile practices has positive effects on the external communication and facilitates dependencies between the tasks–subtasks, feature–requirements between software development teams and stakeholders. Communication hurdles, however, were still present in the communication between the agile software development team and its stakeholders. Thus, it seems that SCRUM and XP practices do not offer enough communication mechanisms in extended environments where many stakeholder groups and development teams are involved in the same software development process. On comparing the two case projects it can be assumed that the number of customers and size of the stakeholder group could be one of the key factors necessitating the adoption of new plan-driven mechanisms to manage feature–requirements dependency in software development organizations.

There are several limitations in any case study research including issues regarding the generalization of findings from the data collected. The findings apply only to case specific units in F-Secure, which used a selected set of agile practices, mainly SCRUM practices and XP practices as optional. Furthermore, the compared projects had differences which may have significant effects on the validity of the analysis.

Owing to the confidential nature of the data and extended periods of data collection, we could not rely on more objective constructs to observe the process or process changes. We also were constrained by access to a few key informants in each organization. Thus, we could only triangulate across different observations of the same data point (interviews at different time points) and across other published materials and the author's own observations of the interview data.

Since only interim, "snap shot", information of software development projects were collected and analysed, it is not possible to understand the factors of the impact of agile practices on communication from a long term perspective. The findings presented in this paper are based on the author's current knowledge gained from observations and interview data in F-Secure, the results presented are very context-specific.

Although, the analysis in the case study was quite in depth, more detailed statistics and quantitative validation based on the data would strengthen the results of this analysis. In the future, a wider sample of studies of agile software development projects would be beneficial to analyse and define the effects of agile practices on communication, especially in larger and more complex contexts such as distributed product development situations. On the other hand, it would be interesting to take plan-driven projects as a point of comparison in a qualitative analysis to further consider the implications of agile practices. Although communication can be defined as one factor that leads to productive software development, it is not the only one. There are also several other factors such as coordination, adoption and change that could be the focus of future studies on agile practices.

Acknowledgements Acknowledgements are given to the employees of VTT and F-Secure Corporation who have participated in this research. The research was conducted within the Agile ITEA project funded by the National Technology Agency of Finland (TEKES) and Nokia Foundation. A special acknowledgement to Brian Fitzgerald for his valuable advice during the writing of this paper. A special thanks also to Xiaofeng Wang, Gary Gaughan and Eoin Ó Conchúir for their comments and support during the writing of this paper.

References

Abrahamsson P, Salo O, Ronkainen J, Warsta J (2002) Agile software development methods: review and analysis. Espoo 107. VTT Publications 408. Espoo

Anderson DJ (2003) Agile management for software engineering, applying the theory and constraints for business results. Prentice Hall, Upper Saddle River, NJ

Bacharach SB (1989) Organizational theories: some criteria for evaluation. Acad Manage Rev 14(4):496–515

Beck K (1999) Embracing change with extreme programming. IEEE Comput 32(10):70–77

Beck K (2000) Extreme programming explained: embrace change. Addison-Wesley Longman, Boston, MA

Beck K, Andres C (2004) Extreme programming explained, embrace change, 2nd edn. Addison-Wesley, Boston, MA

Beck K, Beedle M, Bennekum van A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin R, Mellor S, Schwaber K, Sutherland J, Thomas D (2001) Manifesto for agile software development. Availale at: http://AgileManifesto.org. Accessed 17.7.2007

Boehm B (2003) Value based software engineering, AC, SieSoft, Software Engineering Notes, Vol 28, nro 2, p. 1–12

Boehm B, Turner D (2003) Using risk to balance agile and plan-driven methods. IEEE Comput 36(6):57–66

Boehm B, Turner D (2005) Management challenges to implement agile processes in traditional development organizations. IEEE Softw 22(5):30–38

Boehm BW, Ross R (1989) Theory-W software project management principles and examples. IEEE Trans Softw Eng 15(7):902–916
Börjesson A, Mathiassen L (2004) Successful process implementation. IEEE Softw 21(4):36–44
Carmel E, Agarwal R (2001) Tactical approaches for alleviating distance in global software development. IEEE Softw 18:22–29
Cockburn A (2004) Crystal clear, a human-powered methodology for small teams. Addison-Wesley, Boston, MA
Cohn M, Ford D (2003) Introducing an agile process to an organization. IEEE Comput Soc 36(6):74–78
Coram M, Bohner S (2005) The impact of agile methods on software project management. In: 12th International conference and Workshops on the Engineering of Computer-based Systems, Maryland, USA
Crowston K, Kammerer E (1998) Coordination and collective mind in software requirements development IBM. Syst J 37(2):227–245
Crowston K, Rybleske J, Howison J (2004) Coordination Theory. Draft for Human-Computer Interaction in Management Information Systems. Available at 14.4.2008: http://crowston.syr.edu/papers/coord2004.pdf
Damian D, Eberlein A, Shaw ML, Gaines BR (2000) Using different communication media in requirements negotiation. IEEE Softw 17(3):28–36
Dingsoyr T, Hanssen GK, Dyba T, Anker G, Nygaard JO (2006) Developing software with SCRUM in a small cross-organizational project. Developing software with SCRUM in a small cross-organizational project. In: Springer, EuroSPI 2006, LNCS 4257, pp 5–15
Drobka J, Noftz D, Raghu R (2004) Piloting XP on four mission-critical projects. IEEE Softw 21(6):70–75
Eisenhardt KM (1989) Building theories from case study research. Acad Manage Rev 14(4):532–550
Espinosa JA, Carmel E (2003) The impact of time separation on co-ordination in global software teams: a conceptual foundation. Softw Process Improv Pract 8(4):249–266
Fitzgerard B, Hartnett G, Conboy K (2006) Customising agile methods to software practices at Intel Shannon. Eur J Inf Syst 15(2):197–210
Goles T, Chin WW (2005) Information systems outsourcing relationships factors: detailed conceptualization and initial evidence. Data Base Adv Inf Syst 36(4):47–62
Grenning J (2001) Launching XP at a process-intensive company. IEEE Softw 18(6):3–9
Harbring C (2006) The effect on communication in incentive systems: an experimental study. Manag Decis Econ 27(5):333–353
Henttonen K, Blomqvist K (2005) Managing distance in a global virtual team: the evolution of trust trough technology-mediated relational communication. Strateg Change 14(2):107–119
Herbsleb D, Mockus A (2003) An empirical study of speed and communication in globally-distributed software development. IEEE Trans Softw Eng 29(6):1–14
Highsmith J (2004) Agile project management, creating innovative products. Addison-Wesley, Boston, MA
Highsmith J, Cockburn A (2001) Agile software development: the business of innovation. Computer 34 (9):120–122
Holström H, Fitzgerald B, Agerfalk PJ, Conchuir EO (2006) Agile practices reduce distance in global software development. Inf Syst Manage 23(3):7–18
Karlström D, Runeson P (2006) Integrating agile software development into stage-gate managed product development. Empir Softw Eng 11(2):203–225
Klein H, Myers M (1999) A set of principles for conducting and evaluating interpretive field studies in information systems. MIS Quarterly 23(3):67–94
Korkala M, Abrahamson P, Kyllönen P (2006) A case study on the impact of customer communication on defects in agile software development. In: Agile 2006, Minneapolis, pp 76–88
Kraut R, Streeter L (1995) Coordination in software development. Commun ACM 38(3):69–79
Layman L, Williams L, Damian D, Bures H (2006a) Essential communication practices for extreme programming in a global software development team. Inf Softw Technol 48(9):781–794
Layman L, Williams L, Cunningham L (2006b) Motivations and measurements in an agile case study. J Syst Archit 52(11):654–667
Leon G (1995) On the diffusion of software technologies: technological frameworks and adoption profiles. In: Proceedings of The Diffusion and Adoption of Information Technology, Oslo, pp 97–116
Lindvall M, Muthig D, Dagnino A, Walling C, Stupperich M, Kiefer D (2004) Agile software development in large organizations. Computer 37(12):26–34
Lippert M, Becker-Pechau P, Breitling H, Koch J, Kornstadt A, Roock S, Schmolitzky A, Wolf H, Zullighoven H (2003) Developing complex projects using XP with extensions. Computer 36(6):1–7
Lycett M, Macredie R, Pateil C, Paulk R (2003) Migrating agile methods to standardized development practice. IEEE Comput Soc 36(6):79–85
Mann C, Maurer F (2005) A case study on the impact of SCRUM on overtime and customer satisfaction. Agile, Denver
Malone T, Crowston K (1994) The interdisciplinary study of coordination. ACM Comput Surv 26(1):87–119

Miles M, Huberman A (1999) Qualitative data analysis. Sage, London
Murru O, Deias R, Mugheddu G (2003) Assessing XP at a European Internet Company. IEEE Softw 20 (3):37–43
Paasivaara M, Lassenius G (2003) Collaboration in inter-organizational software development. Softw Process Improv Pract 8(4):183–199
Parnas DL, Clements PC (1986) A rational design process: how and why to fake it. IEEE Trans Eng Manage 12(2):251–256
Pikkarainen M, Mäntyniemi A (2006) An approach for using CMMI in agile software development assessments: experiences of three case studies. Spice, Luxenburg
Pikkarainen M, Wang X, Conboy C (2007) Agile practices in use an innovation assimilation perspective: a multiple case study. ICIS, Montreal
Rasmusson J (2003) Introducing XP into greenfield projects: lessons learned. IEEE Softw 20(3):21–28
Rising L, Janoff NS (2000) The SCRUM software development process for small teams. IEEE Softw 17 (4):26–32
Salo O, Abrahamsson P (2006) An iterative improvement process for agile development. Softw Process Improv Pract 12(1):81–100
Schwaber K, Beedle M (2002) Agile software development with SCRUM. Prentice-Hall, Upper Saddle River, NJ
Shukla A, Williams L (2002) Adapting extreme programming for a core software engineering course. In: Software Engineering Education and Training (CSEET'02), Kentucky, USA
Smite D (2006) Global software development projects in one of the biggest companies in Latvia: is geographical distribution an problem? Softw Process Improv Pract 11(1):61–76
Stelzer D, Mellis W (1998) Success factors of organizational change in software process improvement. Softw Process Improv Pract 4(4):227–250
Sutherland J (2001) Agile can scale: inventing and reinventing SCRUM in five companies. Cutter IT J 14 (12):5–11
Svensson H, Höst M (2005) Views from an organization on how agile development affects its collaboration with a software development team. In: Proceedings of Product Focused Software Process Improvement, Lecture Notes in Computer Science. PROFES 2005, 487–501. Oulu, Finland
Turner R (2003) People factors in software management: lessons from comparing agile and plan-driven methods. J Def Softw Eng 22(4):4–8
Vriens C (2003) Certifying for CMM Level 2 and ISO9001 with XP@SCRUM. In; Agile 2003, Salt Lake City, UT
Weick K (1995) What theory is not, theorising is. Adm Sci Q 40(1):385–390
Williams L, Cockburn A (2003) Agile software development: it's about feedback and change. Computer 36 (6):39–42
Yin RK (1994) Case study research design and methods, 3rd edn. Sage, Newbury Park, CA
Yin RK (2003) Case study research: design and methods, Saga, Thousand Oaks, CA

Minna Pikkarainen has graduated from computer science department of University of Oulu and the topic of her Ph.D. that she is currently doing is about improvement of software development mediated with CMMI and agile practices. Minna has been working as researcher and project manager in Technical Research Centre of Finland since 1997. During that time she has worked in around 17 research projects related to software

process improvement, project management, automatic measurement, software design, requirements management, component based and distributed software development. All these projects have been made in close customer collaboration with among several industrial sectors such as telecommunication, process automation and communication. In parallel, she has participated in preparation and implementation of several international research projects (ITEA, EU) and contract customer work at European level. Previously Minna has been member of Lero, The Irish Software Engineering Research Centre doing co-operation also with Irish industry and research groups. For the past 4 years, her work and publications have been focused on research in the area of agile software development and software process improvement.

Jukka Haikara graduated from University of Oulu (Department of Information Processing Science) a few of years ago. The main topic was interaction design and agile software development, especially how to extend the interaction design process of an agile software development project. Jukka worked as a researcher in Technical Research Centre of Finland in various projects mainly concentrating on agile software development. During that time he covered topics such as usability, interaction design, documentation, product simulations, extreme programming principles and project process development, among other things. Currently, Jukka works at the leading brand in heart rate monitor business, Polar Electro Oy, as User Interface Designer, thus mainly giving his competence to the use of providing more usable products for customers.

Outi Salo earned her doctor's degree (Ph.D.) on information processing sciences from the University of Oulu in 2007. The topic of her doctoral dissertation was "Software Process Improvement in Agile Software Development Teams and Organizations". Outi works as a Senior Research Scientist at VTT Technical Research Centre of Finland. Currently, Outi is a VTT project manager in FLEXI-ITEA2 research project focusing on the scalability of agile methods in multi-site and extensive product development contexts. In addition, her work includes planning, management and implementation of research work with industry.

Outi's research interests include SPI, metrics, software development processes and methods (both agile and traditional), as well as aspects of product and project management, for instance. Previously, she has also worked as a systems analyst in industry.

Pekka Abrahamsson received his Ph.D. degree in software engineering from University of Oulu, Finland in 2002. He is a research professor at VTT Technical Research Centre of Finland. He is also an adjunct chief scientist in SINTEF, Norway. He is currently a visiting professor in Free University of Bozen-Bolzano in Italy. His research interests are centred on agile software development, software process improvement and embedded systems development. He leads large European research projects on these topics. In 2007 he was awarded a Nokia Foundation Award for his achievements in software research. He is in the editorial board of Software Process Improvement and Practice. He is VTT's representative in ISERN and a member of the ACM and IEEE.

Jari Still is F-Secure Oy's Oulu Office site manager and the head of Mobile R&D. Still has been working at F-Secure since 2000. From 1991 to 2000 Still was working as CEO of Modera Point Oy. Before 1991 Still had worked with e.g. Nokia Mobile Phones and Siemens. Still is one of the founders of the Oulu Software Forum, and also has been acting as a Chairman of the Forum (www.swforum.net). Other posts have been e.g. Chairman of revontuliryhmä, Chairman of business development group in Oulu Innovation project and member of the Board in several companies.

Related to Agile processes and product development Still is acting as a leader in FLEXI-ITEA - project at F-Secure and he is also a Chairman of the management group of FLEXI Finland -project.