

Integrating Agile Software Development and Software Process Improvement: a Longitudinal Case Study

Outi Salo & Pekka Abrahamsson

VTT Technical Research Centre of Finland

P.O. Box 1100, FIN-90571 Oulu, Finland

{*outi.salo, pekka.abrahamsson*}@vtt.fi

Abstract

Agile software development solutions are targeted at enhancing work at project level. Little is yet known about the relationship between agile projects and organizational capability improvement. The agile project teams are suggested to iteratively improve their behaviour in a validated manner. This paper addresses the high value of such validated software process improvement (SPI) knowledge emerging from project teams and its utilization at the organizational level SPI. In this paper it is suggested that the novel SPI methods of agile project teams also require alterations in the activities of the organizational level in order to enable the mutually benefiting co-existence of the two. Empirical results from a longitudinal case study over five software development projects are presented to illustrate the evolvement of organizational SPI mechanisms and to derive implications for integrating the agile software development and organizational SPI. The study reveals the high importance of close collaboration between the organizational and project levels throughout the projects and identifies several organizational activities needed in enhancing SPI within agile projects and in an organization.

1. Introduction

A learning organization is “skilled at creating, acquiring, interpreting, transferring, and retaining knowledge, and at purposefully modifying its behaviour to reflect new knowledge and insights” [1, p. 11]. Software process improvement (SPI) activities support the enhancing of a learning organization. Especially, the experiences of the people who actually execute a process have been identified as one of the central sources of input for an SPI program [2].

The organizational business goals have traditionally laid the foundation for all SPI initiatives as in CMMI (Capability Maturity Model® Integration) [3]. According to CMMI, SPI typically involves management steering committees developing strategies, process groups facilitating and managing

the SPI activities, and process action teams defining and implementing the improvement. The role of the practitioners is defined as performing the process [3]. Nowadays, many organizations run agile software development projects alongside their traditional ones. The principles of agile software development [4] propose that “at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly”. Thus, in the agile context, a software development team is in the central role in setting the goals, and planning and implementing SPI activities iteratively throughout a project.

Currently, there seems to be lack of empirical evidence on how the agile approach for SPI integrates to the organizational SPI activities. The existing methods for iterative adaptation and improvement of agile project teams do not seem to address the organizational learning aspect. However, it is known that “knowledge is only maximized if the organization can learn” [5, p. 11]. Thus, the process knowledge of project teams should be “shared collectively rather than limited to a privileged few” [1, p. 11]. The inability of an organization to learn systematically from agile development projects can be seen as a major obstacle in the adoption of agile methods.

In this paper it is suggested that the novel SPI methods evolved for agile project teams also require alterations in the SPI activities of organizational level in order to enable the mutually benefiting co-existence of the two. Systematically collected empirical data is provided from a longitudinal case study where five consecutive agile software development projects employed agile team reflections, as well as traditional project retrospectives to enhance systematic SPI on both, project and organizational levels.

The paper is structured as follows. Section 2 discusses the traditional and agile approaches to SPI. Section 3 suggests the integration of agile SPI approaches in traditional SPI mechanisms of organizations. Section 4 defines the research design and, in section 5, the empirical results from the case study are presented. The paper is concluded with conclusions, limitations, and future research needs.

2. Related Research

In the following sub-sections, the traditional and agile SPI mechanisms as well as their connections to organizational learning are discussed.

2.1. Traditional Approach to SPI

In traditional SPI approaches, such as Quality Improvement Paradigm (QIP) [6] and IDEALSM [7], the promoter of SPI initiatives is the organizational level and the goals set in it. Different SPI methods have been evolved in the context of traditional software development, for example, GQM [8] and CMMI [3]. QIP identifies two cycles of learning: project and organizational. The organizational learning cycle addresses the setting of the improvement goals, and planning and analyzing the results of the SPI initiatives. The project learning cycle addresses the execution (e.g., piloting) and collecting feedback from SPI initiatives (i.e., projects) for organizational analysis.

Traditionally, project retrospectives [9] (i.e., project postmortems) have been one way of harvesting process knowledge of project teams from the finished projects in order to enhance organizational learning. Thus, their focus is on improving project work in future [10]. The harvested knowledge may be relevant for the ongoing SPI initiatives of an organization or include all the experiences and improvement opportunities deemed important by the software developers. Often, the improvement opportunities that arise from the projects still need further analyzing, piloting and validation in the future projects prior to their dissemination in the organizational practices (Figure 2). Evidently, this also means a long time-span between the problem identification and process improvement. Different procedures of conducting retrospectives are suggested (e.g., [11]) but are outside the scope of this paper.

2.2. Agile Approach to SPI

In the agile principles [4], it has been suggested that “at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly”. Thus, in agile context, a software development team collectively is in the central role in setting the goals, and planning and conducting SPI iteratively throughout a project.

Currently, few techniques have been suggested to enable iterative SPI within agile software development projects. Cockburn [12] has proposed a reflection workshop technique, whereas Dingsøy and Hanssen [13] have suggested a workshop technique called

postmortem reviews. Currently, the existing agile project learning techniques seem to lack means to perceive the organizational SPI aspect. For example, they do not address the important aspects of systematically *defining, validating, packaging and storing* the SPI results of agile projects.

The empirical evidence, qualitative or quantitative, that the implemented process enhancements have actually been useful and that they have brought factual improvements to the process is needed within projects. In fact, Extreme Programming (XP) [14] suggests that the teams can change their practices only if they agree on how they will assess the effects of the change [15, p. 71]. Such practices, however, are not defined in XP. It could be argued that empirical validation of SPI actions within project teams would largely increase the value of such SPI knowledge in organizational SPI.

The Post-Iteration Workshop method (hereafter referred as PIW) (more in [16]) is based on the two existing agile reflection techniques. The initial goal of PIWs is to provide mechanisms for project teams to systematically utilize their knowledge and experience for tailoring their base process to better suit their needs. Another goal of PIW's is to provide mechanisms that enable the symbiosis of the project teams and the organizational SPI. For example, the PIW method addresses the systematic *validation of SPI actions and storing of SPI knowledge* iteratively in the ongoing projects to ensure the availability of appropriate SPI knowledge from individual project teams. The PIW method consists of six iteratively repeated steps: 1) preparation, 2) experience collection, 3) planning of improvement actions, 4) piloting, 5) follow-up and validation, and 6) packaging and storing experience.

As illustrated in Figure 1, a vital link between the agile project learning and the organizational improvement cycles is the iterative and appropriate *packaging and storing* of systematically validated and documented SPI knowledge by the project teams. This enables the transfer of the SPI knowledge from projects to organizational level as suggested in Experience Factory [17]. The action point template (defined in [18]) developed especially for storing PIW data defines a structure that supports both the project level follow-up of SPI actions and transferring of appropriate SPI knowledge to the organizational level. For every SPI action generated in a PIW, the following issues are agreed and documented in a systematic and structured manner: 1) the specific SPI action to be taken, 2) responsibilities and schedule for the action, 3) the exact problem that the action point aims at

solving, 4) the means to validate the effectiveness of the SPI action, and 5) the results (qualitative or quantitative) of the validation (updated in the following PIW after piloting). The associated problem area is also identified for each SPI action in order to ease up the filtering of data for later use. The validation data should also be stored in an organizational repository to ensure its availability.

3. Integrating Agile Software Development and Software Process Improvement

In this section the differences of traditional and agile SPI mechanisms are discussed. Secondly, the organizational activities to enhance SPI within agile projects as well as in an organization are defined.

3.1. Traditional and Agile Approaches for Organizational SPI

Two central differences of traditional and agile SPI can be identified. Firstly, the origin of SPI goals is traditionally the organizational level. However, in the agile project context the impelling power for SPI lies within individual project teams, and their experiences and learning throughout projects. Secondly, the process knowledge of project teams has traditionally been harvested from the finished projects in order to enhance project work in future [10, p. 255]. Instead, the immediate focus of agile project learning is on improving the performance of the *ongoing* project.

The traditional and agile SPI approaches are not contradictory. Rather, both the approaches can be beneficial in integrating agile software development and organizational learning. This, however, requires the integration of agile SPI mechanisms in the organizational learning cycle (Figure 1).

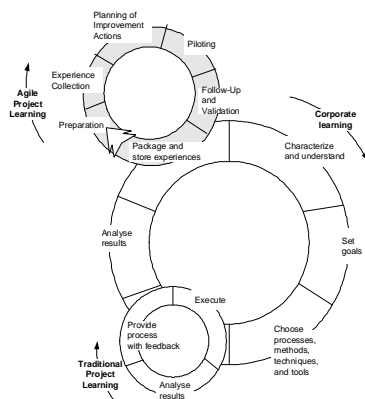


Figure 1. QIP ([17]) and agile project learning

Traditionally, a project team may participate in a retrospective after the project in order to provide the

organizational level with their experiences. Organizational SPI stakeholders, then, ensure that the harvested knowledge is appropriately stored, filtered, analyzed, piloted and disseminated from individual projects to organizational processes. Figure 2 illustrates the interaction of traditional project learning from project retrospectives and organizational learning.

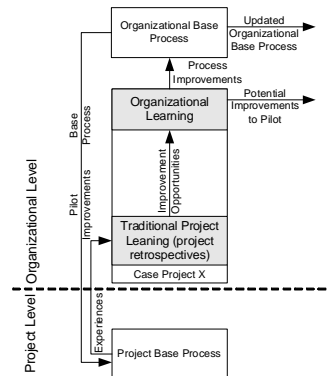


Figure 2. Interaction between project retrospectives and organizational learning

The central outputs from retrospectives (Figure 2) are the improvement opportunities (i.e., lessons-learned) identified by the project team. In the agile context four types of learning output from projects can be identified (Figure 3): 1) the metrics data collected in the projects, 2) the validated process improvements conducted iteratively within the agile software development project (including the reasoning behind the piloted improvement actions and validation data), 3) the improvement requests of agile project teams (i.e., improvement actions that would require organizational support or decision making during the project), and 4) improvement opportunities (i.e., suggestions for future improvement).

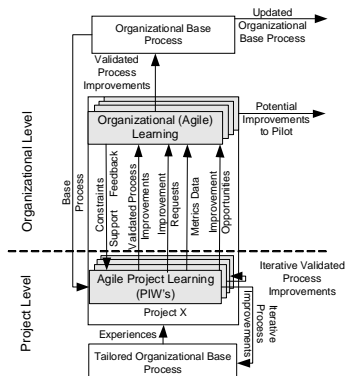


Figure 3. Interaction between agile project learning and organizational learning

The validated process improvements already piloted during the agile projects are a valuable input for the organizational SPI that the traditional project retrospectives do not provide. At its best, agile project learning can provide validated, classified, well reasoned, and adequately stored SPI knowledge and insights into the agile software development to be addressed by the organizational SPI. The organizational SPI stakeholders, then, can filter and analyze the SPI knowledge from multiple projects, and embed certain process improvements directly in the organizational practices or to plan to further pilot them later on.

3.2. Integration of Agile Software Development and Organizational SPI

The novel team-centred SPI methods evolved for agile project teams also require alterations in the SPI activities of the organizational level in order to enable the mutually benefiting co-existence of the two. Agile software development addresses the close communication between the organizational layers [4]. The need to increase the level of collaboration is also evident in integrating the iterative and rapid SPI activities of project teams and organizational SPI stakeholders in a way that benefits both the parties.

While the project teams are in a central role in improving their process within the organizational limitations, they cannot be left to do it all by themselves. For one, the project teams should be provided with organizational *facilitation* in conducting the project learning activities. For the project team, this ensures the availability of adequate knowledge on conducting the iterative learning activities (e.g., defining the improvement actions and their proper validation) and with organizational guidelines for tailoring the process. The organizational level, in return, gains on-time and first-hand process improvement knowledge from the ongoing projects. The active *participation* of the organizational level in the agile project learning activities provides valuable insights in the problems and potential solutions in the current organizational practices. By active collaboration with project teams organizational level also has an opportunity to influence the adequate definition, planning, validation of the SPI activities and packaging of the results for organizational utilization.

Project teams also require *support* in conducting some of the SPI actions they have planned to undertake. These improvement actions can be regarded as improvement requests (Figure 3) to the

organizational SPI stakeholders. Reacting to these improvement requests may be of high importance, as it ensures the implementation and validation of the SPI actions deemed important by the project team, which is not capable of accomplishing these tasks itself. This is likely to increase the motivation of the developers to participate in the agile learning activities [16].

Iterative, on-time *feedback from organizational level* considering, for example, results of the project monitoring is important for the agile project teams. They may utilize this information for learning purposes and in setting the SPI goals for the following iterations. On the other hand, the organizational level gains *feedback from the project teams* (i.e., improvement opportunities, validated improvements, improvement requests) of applying different (agile) practices in different project contexts. This is valuable input for organizational improvement in the long-run.

4. Research Design

In this section, the goals, setting, methods, and data collection of this research are defined.

4.1. Research Goals

This paper presents a proportion of a research focused on SPI in agile software development. Results have also been published in [16, 18-21]. The research goal focused on this paper is to study the integration of SPI as suggested in the agile software development context and organizational SPI. The empirical evidence was collected from five agile software development projects which participated in traditional project retrospectives as well as conducted agile project learning sessions (i.e., PIWs).

In the case organization, an organizational software engineering process group (SEPG) [3] (hereafter referred as process group) was established. It aimed at continuous and effective utilization of the SPI knowledge gained from the projects. In order to achieve this, the process group needed to evolve practices to ensure beneficial co-existence of agile project learning and organizational learning. Thus, the underlying research goal of this study was to develop mechanisms for organizations where agile projects are conducted to integrate the agile project learning in organizational level SPI. In the research organization, a goal was to utilize learning from both agile project learning (PIWs) as well as project retrospectives in devising an agile software development process to meet the needs of the volatile mobile software development environment (i.e.

Mobile-D™ [22]). For the customer organizations the goal was, naturally, to produce a high quality product as efficiently as possible.

4.2. Research Setting

All the case projects were conducted in the same open office settings at VTT, the Technical Research Centre of Finland, with different customer organizations and project teams. Table 1 illustrates the central characteristics of the case projects.

Table 1. Characteristics of case projects

Project	Size (person months)	End product	Duration	Iterations
1	8.5	Intranet app	9 weeks	3 x two weeks 3 x one week
2	10	Mobile app	9 weeks	1 x one week 3 x two weeks 2 x one week
3	5.5	Mobile app	9 weeks	1 x one week 3 x two weeks 2 x one weeks
4	5.2	Mobile app	11 weeks	2 x two weeks 7 x one week
5	7.1	Mobile app	8 weeks	1 x 1 weeks 3 x 2 weeks 1 x 1 weeks

The organization of the case study context included a steering group consisting of members of the customer and the research organizations, an off-site customer, a process group, a support group, and the respective software development teams. The teams consisted of the developers of customer organization as well as experienced students of computer science.

quality product. The decisions concerning the project level process enhancements were to be made by applying mechanisms of PIW method. Thus, all the software development projects in the case study conducted PIWs after the first four (three in Project 5) completed iterations (Figure 4). A total of 19 PIWs were conducted during the research. The workshops were held as the first activity of all iterations. The project teams also participated in a retrospective at the end of each project (PR in Figure 4).

The organizational level SPI activities were ran by the process group consisting of process improvement specialists of the research organization. Their immediate goal was to analyze and transfer process knowledge from the project teams to the organizational software development process (i.e., Mobile-D™ [22]), and, thus from a project to another. The process group had also had a vital role in supporting agile project learning. The facilitator, who was in charge of arranging and conducting the agile project learning sessions, i.e., PIWs, within the project teams was a member of the process group. Thus, she possessed expertise not only in SPI, but also in the Mobile-D™ process itself. The facilitator was an actor operating between the two organizational layers and thus in position to provide the project teams with information on, for example, organizational SPI guidelines and the process group with insights into the SPI within projects. The support group members were training (i.e., prior the project) and coaching (i.e., throughout the project) the project members on the software development process and its practices, and

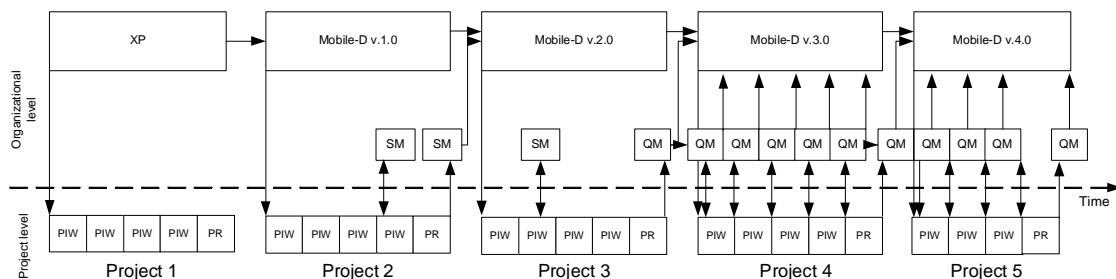


Figure 4. Evolvement of SPI approach
(PIW=post-iteration workshop, PR=project retrospective, SM=support team meeting, QM=quality meeting of process group)

The project teams were empowered to adapt the software development practices iteratively from the first iteration onwards based on their experience, objective metrics and knowledge. The aim was to optimize the daily software development work and the development process while also producing a high

providing other support needed by the project team. The support group members also belonged to the process group due to their specific context-related knowledge emerging from close collaboration with development teams.

SPI methods and activities were constantly devised, revised and adapted throughout the longitudinal study. The issue of organizational SPI was increasingly addressed towards the end of the study when, for

example, systematic process group activities were launched (quality meetings, i.e., QM in Figure 4). The first three projects lacked systematic mechanisms for storing, analyzing, transferring and disseminating the process knowledge of project teams in an organization (Figure 4). The early attempts at organizational learning took the form of relatively unorganized and irregular meetings of the support team (SM in Figure 4) where the problems and the progress of the ongoing project were discussed. At the end of Project 3, it was realized that regular and systematic means were needed to effectively improve the organizational base process (XP) to better suit the mobile software development context (i.e., Mobile-D™) beyond a single project. Thus, the quality meetings (QMs) were launched. Also, the agile project learning mechanisms (i.e., PIWs) were altered during the longitudinal case study to better meet the needs of organizational learning.

4.3. Research Approach and Methods

This research is a longitudinal case study spanning over 18 months in calendar time and over 5 sequential agile software development projects. An action research approach (e.g., [23]) was applied in all the case projects. A researcher was acting as a facilitator in both project and organizational level SPI activities. This enabled an effective way to “integrate theory with practice through an iterative process of problem diagnosis, action intervention, and reflective learning” [24]. In other words, the SPI techniques were constantly evolved throughout the case projects based on the experiences gained in practice.

The individual case projects were carried out applying a controlled case study approach [25]. It suits the study of agile methodologies particularly well because it strives for creating a software development environment, which involves a close-to-industry setting with immediate business pressure. The business pressure comes from tough time-to-market needs of real customers and real products. The controlled case study approach aims at a dual outcome: 1) fully functional software system or a software product for customer, and 2) research data on selected aspects. Thus, one central difference from laboratory experiments is that even though the project team works in laboratory-like settings, it actually produces a real software product for a real customer.

4.4. Data Collection

A large amount of research data was systematically collected from the case projects. The data included the flap-sheets containing the positive and negative

findings of project teams from the project retrospectives as well as the PIWs. The action point sheets and data analysis sheets from the PIWs were also important empirical material. In addition, the action points from the support team meetings as well as the action point lists from quality meetings were analyzed. At the end of each project, a group interview was held to record the perceptions of the developers.

5. Empirical Results

During the longitudinal case study the organizational SPI activities in agile software development context were evolved. The resulting process group activities could be categorized in two groups: 1) tasks directly related to the on-time SPI activities with the ongoing projects, and 2) long-term SPI activities focusing on improving organizational software development processes (i.e., organizational learning activities). In the following, these two aspects of the process group are further discussed with empirical evidence from the case projects.

5.1. Process Group and Agile Project Learning

The longitudinal case study revealed the importance of continuous collaboration between the process group and agile project teams. Different cooperative activities were identified as necessary to benefit the project teams as well as the process group in their SPI efforts.

Table 2 presents the amount of improvement actions of project teams in categories of “process improvements” and “other improvements”. The first ones relate to improving the practices and tools of the software development process. The “other improvements” are actions needed to meet other requests of developers, such as the set-up of new tools.

Table 2. Project level improvement actions and need for organizational support

Process Improvements of Project Teams	Project				
	1	2	3	4	5
Number of Process Improvements Requiring Organizational Support	6	16	5	8	12
Number of Process Improvements Not Requiring Organizational Support	3	9	6	2	3
Other Actions of Project Teams	Project				
	1	2	3	4	5
Number of Other Improvements Requiring Organizational Support	1	5	9	4	6
Number of Other Improvements Not Requiring Organizational Support	6	26	13	13	5

On the average, only 33 % of all the implemented “process improvements” in all of the projects were conducted without organizational *support*. Thus, for

67 % of the SPI actions resulting from agile project learning, improvement requests were passed on to the organizational support and process groups to act on. The “other improvements” required less support from the process group. On the average, as much as 72 % of these actions were conducted within the project teams. The rest of the actions (i.e., 28 %) required organizational involvement. Thus, one important task of the process group was to iteratively analyze the improvement requests of the project teams in the quality meetings and, thereafter, respond to them as agreed. Even though, at many times, the improvements were delayed or rejected, only the notion that the organizational level considered the project team requests seemed to have a positive influence on the motivation of the team. The qualitative data from the interviews of software developers also revealed the importance of organizational support in the agile project learning.

“It was good in a way that if something was bad and complained about, it was fixed.” (Project 2)

“We got our voices heard. Not necessarily would we have said those things otherwise but just bitten on the bullet till the end.” (Project 2)

“Even if the same things had been said on some other occasion, they would not necessarily have been registered - now they were.” (Project 4)

The project teams also requested support for conducting the SPI sessions (PIWs). The qualitative data from the final interviews reveals that all the project teams preferred an external facilitator who could provide new visual angle on the problematic issues, had expertise in conducting the iterative improvement yet also had knowledge of the process and authority to influence and make organizational decisions as well as position to distribute knowledge between the projects and the process group.

“If the facilitator was a team member and only the team would participate, the comments would likely change because it would feel as if you were protesting to your team mate. Not necessarily would you like to complain to another team member but when an outsider comes along things can be said more freely.” (Project 4)

“If it has to be someone from the team, it should be the project manager. However, I think it is much better to be from outside of the team, because then you have a better chance to find the relevant things. Otherwise, if the workshop had been run by the team,

the same questions and the problems that were discussed day by day would have been there.” (Project 5)

Furthermore, the *process feedback* was something that was requested by the project teams in PIWs, in project retrospectives as well as in final interviews. To ensure systematic process feedback, finally in the 6th quality meeting of Project 4, the process group agreed on a set of metrics (i.e., velocity, effort estimation and code complexity) to be iteratively analyzed and discussed separately within both the project team and the process group.

The feedback clearly seemed to benefit the project team in learning from the metrics data of the previous iteration. In Project 5, for example, velocity and estimation accuracy data were prepared and discussed with the project team after each iteration. A lot of discussion and learning was achieved through the project team interpreting the causes of delays for the tasks in the 1st PIW. One reason for underestimating the tasks was considered to be the assumption of using solo coding in the implementation even though pair-programming was generally highly used. Current studies have, however, indicated that approximately 10 to 20 % more effort is needed on pair-programmed tasks. Furthermore, the implementer of the task was not known at the time of effort estimation, which, according to the developers, would definitely affect the effort estimations. The technical difficulties in the implementation phase were something that could not be foreseen or prepared for in the estimation task.

The average estimation accuracy of Project 5 improved during the first three project iterations (Figure 5). As it can be seen, the tasks were underestimated in all the iterations, while significant iterative improvement can also be distinguished. Naturally, this trend is influenced by a lot of different factors, such as learning through monitoring and interpreting the data of previous iterations.

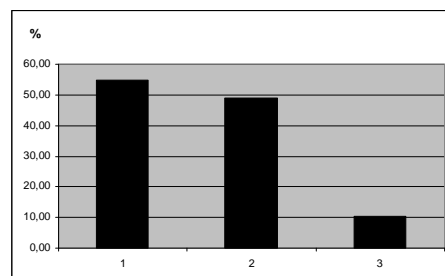


Figure 5. Effort estimation accuracy (%) in project 5

The velocity metric included both actual and planned velocity of each iteration. The actual velocity encapsulated the relative effort that was used on implementing the software development tasks defined for the iteration. By monitoring the realization of actual velocity from the previous iteration, the team was able to tune the velocity percent (Figure 6) for the scheduling of the next iteration.

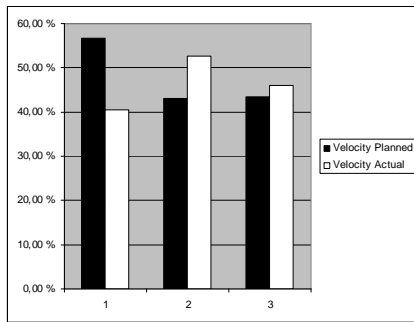


Figure 6. Improvement in planning the velocity in project 5

As is illustrated in Figure 6, the ability of the project team to estimate velocity increased towards the end of the project. The actual velocity of the project team was at its lowest at the first iteration when the software development methods and tools were still new to the project team and the set-up tasks were highly time-consuming. After the 1st iteration, the gap between the actual and planned velocity was considered too large and, accordingly, the team dropped the planned velocity from 57 % to 43 % for the second iteration. In the 2nd iteration, the project team, theoretically, could have implemented more tasks than actually scheduled for the iteration. However, as the effort estimations for the different tasks were still underestimated, there was no need for the team to plan any new tasks during the last two project iterations. In the 3rd iteration, however, the velocity estimation was highly accurate as were also the effort estimations. It is impossible to specify all the causes for this improvement in terms of estimating the velocity and the tasks. However, the iterative feedback from process group in the form of metrics data enhanced the discussing and learning within the project team and, thus, seemed to be one of the factors contributing to this advancement.

5.2. Process Group and Organizational Learning

The central goal of the process group established during the case study was to improve the

organizational software development process (i.e., Mobile-D(sm)). The aim was to utilize the process knowledge from PIW's and project retrospectives in order to adapt the original base process (i.e., XP [14]) to better suit the mobile software development context.

Thus, one of the tasks of the process group was to filter and analyze the material from the PIWs and project retrospectives in quality meetings (QMs). The process group started its operations at the end of the third project as the need and opportunity to utilize the knowledge created in project teams was realized. Prior to this, the support team was conducting the activities of process group in an unsystematic manner.

Regular QMs were held in the last two software development projects (Figure 4). They were situated at the end of every iteration prior to the agile project learning activity (i.e., PIW) and, correspondingly, after every traditional project learning session (i.e., project retrospective). The QMs that were held during the project mainly concentrated on supporting and monitoring the agile project learning activities, whereas the organizational SPI was increasingly addressed when all the PIW and project postmortem data was available after a project for analysis.

Each quality meeting yielded an action point list similar to those of the PIWs. The action points were addressed to the members of either the support or the process group. The timeframe for conducting the actions ranged from immediate actions for the ongoing project to actions for future projects (i.e., on the organizational software development process).

The process group consisted of researchers who were responsible for coordinating the improvement of the organizational process [3]. The quality manager was responsible for preparing the agenda, and pre-analyzing and filtering the empirical material from the PIWs and retrospectives. This was necessary to enable the process group to focus on the most important and urgent issues in their two-hour meeting. It was clearly realized, however, that in a situation of multiple concurrent and overlapping projects effective electronic mechanisms to store and retrieve the classified process knowledge would be necessary. The support team members were part of the process group as they were actively supporting (e.g., coaching) the software developers in their daily work and highly knowledgeable of the ongoing projects.

Table 3 illustrates the number of the implemented organizational improvement actions. The actions of the last QM in the projects originate mostly from the improvement opportunities identified at project retrospectives and validated SPI actions of the PIWs.

The SPI actions conducted during the projects, however, mostly originate from the support requests of the agile project teams. The “SPI actions” in Table 3 refer to the SPI actions taken by the process group on the organizational software development process (i.e., Mobile-D) during or after the case projects. These included, for example, changes to organizational metrics collection tools, improvements in the training of project teams and involvement of organizational testing procedures. The “other actions” refer to SPI related activities such as the supporting and monitoring of agile project learning activities, and enquiring new potential tools or techniques. These actions affected the ongoing project or future projects.

Table 3. Organizational improvement actions

Project	QM	SPI actions	Other actions
1	-	-	-
2	-	-	-
3	1 st	16	2
4	1 st	10	5
	2 nd	8	3
	3 rd	2	2
	4 th	2	1
	5 th	7	0
	6 th	5	1
	Project 4 Total	34	12
5	1 st	4	2
	2 nd	4	3
	3 rd	4	0
	4 th	6	2
	Project 5 Total	18	7
Total		68	21

The data reveals that 76 % of all the implemented improvement actions of the process group can be regarded as actual software process improvement issues. Thus, whether these actions were implemented during or after the ongoing projects, they ultimately affected the organizational software development process itself. Moreover, 20.5 % of all the improvements planned by the process group failed or were postponed for one reason or another.

6. Conclusions and Future Research

One of the cornerstones of agile software development is the iterative adaptation of software development practices during an ongoing project based on the knowledge and experience of software developers. The iterative agile project life-cycle provides an opportunity for project teams to consider how to improve their performance and pilot the agreed improvements in the ongoing projects.

Continuous improvement of organizational software processes is important in enhancing the

capabilities of an organization. The traditional approaches for organizational SPI, however, need to be altered to enable the co-existence of agile projects and organizational SPI.

This paper presents empirical evidence that has been collected in a longitudinal case study of five successive agile software development projects where organizational SPI mechanisms were evolved to suit the agile software development context. The results of the study imply that, for multiple reasons, the central aspect of successful and mutually beneficial co-existence of agile project learning and organizational learning is the *constant collaboration* between the project teams and organizational level. The empirical evidence reveals that without *support* from organizational level a majority of the improvements agreed within project teams cannot be implemented (i.e., 67 %). In specific, the project teams require the organizational level to take actions and make decisions already during the ongoing projects, and to provide their process (improvement) knowledge and facilitation skills for the agile learning activities. The project teams of the case projects also requested *continuous feedback* and *facilitation* in the project level SPI activities from organizational level. The feedback, in the form of an iteratively analyzed metric set, was found valuable for the case project team considering their agile learning process as well as for the process group in monitoring the process. By providing the project teams with an organizational facilitator who participated actively in the agile project learning activities, the process group could, for example, ensure the bi-directional transferring of insights and process knowledge as well as the appropriate tailoring of the software development process within the teams.

Furthermore, as in traditional SPI, one of the basic requirements for successful co-existence of agile project learning and organizational learning was found the systematic and appropriate packaging and storing of relevant project level process improvement knowledge, which enables a flexible and effective analysis of its contents. In the longitudinal case study, although only five consequent projects were managed by the organizational process group, the amount of emerging iterative process knowledge was extensive.

Currently, organizations are increasingly deploying agile methodologies in their software development projects. However, a systematic and large-scale adoption of agile practices in organizations is still elusive and mainly focused on project level activities. For this reason, this study was conducted in a research

environment where software developers produced real products for real customers yet in a controlled environment. The organizational process group was formed of research personnel who developed a mobile specific agile software development process (i.e., Mobile-D™) for industrial use. For one, this made it possible to provide software development organizations with early results on integrating agile software development and organizational SPI. On the other hand, this may be regarded as a limitation of this study. Thus, further research is needed, and currently conducted, to adapt the findings of this research in pure industrial settings.

Acknowledgements

The research was conducted as part of ICAROS and AgileITEA projects, both funded by TEKES (National Technology Agency of Finland). Acknowledgements to the software development staff, customers and fellow researchers.

References

- [1] D. A. Garvin, *Learning in Action*. Boston, Massachusetts: Harvard Business School Press, 2000.
- [2] J. V. Vandeville, "Organizational Learning Through the Collection of "Lessons Learned"," *Informing Science*, vol. 3, pp. 127-133, 2000.
- [3] C. M. S. E. I. SEI, "Capability Maturity Model@ Integration (CMMISM), Version 1.1," Carnegie Mellon Software Engineering Institute 2001.
- [4] Agile Alliance, "<http://www.agilemanifesto.org/principles.html>," 2001.
- [5] M. J. Earl, "Knowledge as Strategy: Reflections on Skandia International and Shorko Films," in *Knowledge in Organizations*, L. Prusak, Ed.: Butterworth-Heinemann, 1997, pp. 1-14.
- [6] V. R. Basili, "Software Development: A Paradigm for the Future," COMPSAC'89, Orlando, Florida, 1989.
- [7] B. McFeeley, "IDEAL(SM): A Users Guide for Software Process Improvement," Software Engineering Institute (SEI, Handbook CMU/SEI-96-HB-001, February 1996).
- [8] V. R. Basili, "The Goal Question Metric Approach," in *Encyclopedia of Software Engineering*, vol. 2: John Wiley & Sons, Inc., 1994, pp. 528-532.
- [9] N. L. Kerth, *Project Retrospectives: A Handbook for Team Reviews*: Dorset House Publishing, 2001.
- [10] M. von Zedtwitz, "Organizational Learning through Post-Project Reviews in R&D," *R&D Management*, vol. 32, pp. 255-268, 2002.
- [11] B. Collier, T. DeMarco, and P. Fearey, "A defined process for project post mortem review," *IEEE Software*, vol. 13, pp. 65-72, 1996.
- [12] A. Cockburn, *Crystal Clear: a Human-Powered Methodology for Small Teams*: Addison-Wesley, 2005.
- [13] T. Dingsøyr and G. K. Hanssen, "Extending Agile Methods: Postmortem Reviews as Extended Feedback," 4th International Workshop on Learning Software Organizations (LSO02), Chicago, Illinois, USA, 2002.
- [14] K. Beck, *Extreme Programming Explained: Embrace Change*: Addison Wesley Longman, Inc., 2000.
- [15] K. Beck, "Embracing Change with Extreme Programming," *IEEE Computer*, vol. 32, pp. 70-77, 1999.
- [16] O. Salo, "Improving Software Process in Agile Software Development Projects: Results from Two XP Case Studies," EUROMICRO 2004, Rennes, France, 2004.
- [17] V. R. Basili and G. Caldiera, "Improve Software Quality by Reusing Knowledge and Experience," *Sloan Management Review*, pp. 55-64, 1995.
- [18] O. Salo, "Systematical Validation of Learning in Agile Software Development Environment," 7th International Workshop on Learning Software Organizations, Kaiserslautern, Germany, 2005.
- [19] O. Salo, K. Kolehmainen, P. Kyllönen, J. Löthman, S. Salmijärvi, and P. Abrahamsson, "Self-Adaptability of Agile Software Processes: A Case Study on Post-Iteration Workshops," 5th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2004), Garmisch-Partenkirchen, Germany, 2004.
- [20] O. Salo and P. Abrahamsson, "A Post-Iteration Workshop Approach for Agile Software Process Improvement: Implications from a Multiple Case Study," *Under Review*, 2005.
- [21] M. Pikkarainen, O. Salo, and J. Still, "Deploying Agile Practices in Organizations: A Case Study," European Software Process Improvement and Innovation (EuroSPI 2005), Budapest, Hungary, 2005.
- [22] P. Abrahamsson, A. Hanhineva, H. Hulkko, T. Ihme, J. J., M. Korkala, J. Koskela, P. Kyllönen, and O. Salo, "Mobile-D: An Agile Approach for Mobile Application Development," 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA04), Vancouver, British Columbia, Canada, 2004.
- [23] J. B. Cunningham, "Case study principles for different types of cases," *Quality and quantity*, vol. 31, pp. 401-423, 1997.
- [24] F. Lau, "Toward a framework for action research in information systems studies," *Information, Technology & People*, vol. 12, pp. 148-175, 1999.
- [25] O. Salo and P. Abrahamsson, "Empirical Evaluation of Agile Software Development: A Controlled Case Study Approach," presented at 5th International Conference on Product Focused Software Process Improvement, PROFES 2004, Kansai Science City, Japan, 2004.