Self-Adaptability of Agile Software Processes: A Case Study on Post-Iteration Workshops

Outi Salo[1], Kari Kolehmainen[1], Pekka Kyllönen[1], Jani Löthman[2], Sanna Salmijärvi[2], and Pekka Abrahamsson[1]

[1] VTT Technical Research Centre of Finland,
P.O. Box 1100, FIN-90571 Oulu, Finland
{Outi.Salo; Kari.Kolehmainen; Pekka.Kyllonen; Pekka.Abrahamsson}@vtt.fi
[2] Department of Information Processing Science,
P.O.Box 3000, FIN-90014 University of Oulu, Finland
{Jani.Lothman; Sanna.Salmijarvi}@oulu.fi

Abstract. None of the agile methods are claimed to fit all development situations. A team should attempt to adapt the methods and practices to fit their specific needs. For that reason agile principles call for self-reflection on a regular basis in order to identify where and how to make improvements. While some systematic approaches on how to execute this self-reflection process effectively have already been proposed, little empirical evidence currently exists. This paper reports empirical results based on a study where a project team conducted a self-reflection process called “post-iteration workshop” in order to improve and optimize the adopted practices in an XP project. Both qualitative and quantitative data were collected from four 1-2 hour workshops. The results show that with less than 4% effort it is possible to hold post-iteration workshops that significantly help to improve and optimize practices and enhance the learning and satisfaction of the project team.

1 Introduction

Agile methodologies and principles [see e.g., 1] place emphasis on incremental software development with short iterations, adaptation to changing requirements, close communication, self-organizing teams, and simplicity [2]. While all of them are challenging to implement in practice, relying on self-organizing teams is an ambitious goal in itself.

Agile proponents have noted that "each situation calls for a different methodology" [2, p. 184]. Thus, one of the principles behind agile manifesto (www.agilemanifesto.org/principles.html) suggests that the team should regularly reflect on how to become more effective, and fine-tune and adjust its behavior accordingly. Cockburn refers to "the mystery of how to construct a different methodology for each situation without spending so much time designing the methodology" [2, p. 184]. Some systematic approaches have been proposed on how to execute this self-reflection process effectively. Cockburn [2] suggests a methodology-growing technique including a team reflection workshop after each

iteration. Furthermore, Dingsøyr and Hanssen [3] have suggested a learning mechanism called postmortem reviews to be used as an extension for agile software development methods. It works towards making good use of the experiences of project participants at the end of iteration to enhance the development process. In agile software development, one iteration may last from one to four weeks [4]. In terms of knowledge management, post mortem reviews could be described as a method that targets "dynamic interaction that facilitates the transformation of personal knowledge into organizational knowledge" [5, p.14]. The idea of postmortems in software development projects is not a new one. In recent years different postmortem techniques have been used in traditional software development approaches [examples 6, 7]. They suggest that each project should conclude with postmortem review to analyze our shortcomings in order to learn and improve [7]. Postmortem reviews have been found to be effective as a tool for organizational learning and productive from the software process improvement (SPI) point of view [example 8]. However, they are not suitable, as such, to an agile software development environment since they focus on traditional software development approaches involving long durations, rich and detailed documentation and large projects [see example 7].

However, little empirical evidence on using either team reflection workshops or the lightweight postmortem reviews in agile software development exists. This paper presents empirical results from a case study (eXpert) where a project adopting Extreme Programming (XP) method systematically reflected its practices after each increment in a session that combined elements from both the Cockburn's team reflection workshop [2] and the lightweight postmortem review technique suggested by Dingsøyr and Hanssen [3]. This technique, as presented here, is referred to as a post-iteration workshop. The case study presented here is the first among an ongoing series of Agile case studies conducted at the Technical Research Centre of Finland.

This paper is composed as follows. The following section presents the research design including the method, the research target and settings. The paper continues with the results, experiences of the post-iterations workshops and conclusions. The paper concludes with final remarks.

2 Research Design

In this section, the research method, data collection, the post-iteration workshop technique (i.e., research target), and the research setting are described.

2.1. Research method and data collection

The research method used in this study was action research [9] that can be seen as one form of case study. The focus is more on what practitioners do rather than what they say they do [10]. The resulting knowledge should guide the practice [11]. In action research, the modification of reality requires the possibility of researcher intervention [12]. In the post-iteration workshops the researchers' acted in the role of a moderator and participated in the generation of positive and negative findings and

enhancing the process with the project team. However, they did not participate in the actual software development work, but acted more as a support team for the developers.

Quantitative and qualitative research data was collected on a) effort used on workshops, b) quantity of findings and c) their content and, d) quantity and e) content of suggested and actual process enhancements (i.e. action points). Furthermore, developers maintained diaries to record their negative and positive perceptions on the process. A final interview was also held for the project team at the end of the project.

2.2. Research target: Post-Iteration Workshop technique

The research aims to study how a short iterative reflection session is suitable for self-adapting the practices during an Agile software development project. The existing reflection techniques (i.e. lightweight postmortem review and team reflection workshop techniques) were examined beforehand. The aim was to combine and adopt these techniques in order to attain effective self-adaptability with minimal effort and high impact. As a result, a post-iteration workshop technique was constructed.

As suggested in the postmortem review technique, the problem-solving brainstorming method called KJ method [6] was adopted in the post-iteration workshops. It was used for *generating experiences* from the project team and collecting and structuring this data. As a result, the project team generated positive experiences, i.e. the practices that should remain the same, on post-it notes and placed them individually for display on a flip chart with clarifying comments. The findings were then grouped and the groups were labeled to simplify the discussion on the emerged topic areas. Similarly, the negative findings were placed on display and grouped in order to identify the problem area. The reason for using KJ for generating experiences in post-iteration workshops instead of more free discussion, as suggested by Cockburn [2], was its controllability and effectiveness as a result of strict procedures.

Both techniques suggested prioritizing the negative findings and analyzing only the most important ones. However, in post-iteration workshop technique all the findings were considered to be equally important (whether positive or negative) and were included in further discussion. Furthermore, the amount of post-it notes was not limited in any way, as reported to be in the case of the lightweight postmortem review technique [3]. Moreover, a root cause analysis technique called the Ishikawa diagram for analyzing the underlying causes, as suggested in lightweight postmortem review technique, was considered but not included. As an alternative, the Cockburn's suggestion of analyzing the negative issues and collecting improvement suggestions along with discussion was followed using the organized flipchart of negative findings as a guide.

The post-iteration workshops ended with the generation and agreement on the improvement actions for the next iteration, i.e. list of action points. Finally, the list of action points from the previous workshop was revised to find out what improvements had actually taken place and which ones were not implemented for whatever reason.

2.3. Research setting

The case study was conducted in a software development project (eXpert) where a team of four developers implemented an intranet application for managing the research data obtained over years at a Finnish research institute. The project lasted eight weeks and the size of the resulting product was 10000 lines of code (see more details in [15]). The development team followed the XP process as suggested by Beck [4]. The team consisted of experienced university students to confirm comparability to practitioners in industry as suggested in [13]. The development team worked in a co-located environment with an on-site customer (a representative of a management organization), as suggested in XP practices [14].

The project members were novice on using agile software development methods. They were guided to adopt all the central XP practices including planning game, small releases, metaphor, simple design, testing practices, refactoring, pair programming, collective ownership, continuous integration, 40-hour week, on-site customer, and coding standards [4]. However, the project had the freedom to adapt the practices based on their experiences from the first iteration onwards. The decisions concerning process enhancements were to be made in the post-iteration workshops.

The project team worked a 24-hour week in four days, in other words from Monday to Thursday. As proposed by the 40-hour week rule, no overtime was recommended. The possible overtime was compensated in the following iteration. The project consisted of five iterations during the eight-week period. The first four iterations were the actual software development iterations and the last one was a corrective iteration. The first three iterations lasted for two weeks and the last two iterations for one week each. A post-iteration workshop was held after each of the iterations. Only the first four workshops are comparable and as a result included in the analysis presented in this paper. The last workshop can be regarded as post-project workshop that concentrates on the experiences from the entire project instead of the previous iteration. It is a valuable part of software process improvement (SPI) in an Agile organization and will be reported thoroughly in the near future.

3 Case Study Results

In this section, the results of the post-iteration workshops are presented and interpreted. Each post-iteration workshop concentrated on the experiences gained from the previous iteration. At the end of this section, the perceptions of the project team are summarized.

3.1 Post-Iteration Workshop Findings

Table 1 presents the costs of post-iteration workshops in terms of effort usage. The data includes the effort of the four software developers. Results show that the effort

spent reduced from iteration to iteration. In other words, the duration of a workshop went down from over 2.5 hours to less than one hour per session. It should be noted that due to the shorter duration of the fourth iteration (i.e. one week) the proportion of effort rises even though the actual effort spent is lower. Also, one factor that presumably increases the duration of workshop in the eXpert case study is the fact that the amount of findings was not limited and all of the findings were considered equally important (i.e., no prioritization).

Table 1: Cost of post-iteration workshop

Iteration	Effort on post-iteration workshops (in hours)	Total project effort (in hours)	Effort spent on post-iteration workshops (%)
1	10,7	195,4	5,5 %
2	7,3	189,7	3,8 %
3	4,0	193,7	2,1 %
4	3,7	110,7	3,3 %
TOTAL	**25,7**	**689,5**	**3,7 %**

In Dingsøyr and Hanssen's [3] study, the effort spent on lightweight postmortem reviews was around 4.7% and the duration of one workshop was roughly 1.4 hours per person (calculated from their data). Cockburn [2] estimates a minimal duration from two to four hours. In this study, the average effort was 3.7% and the average duration of the workshop was 1.6 hours. The percentual effort spent on post-iteration workshops may seem somewhat high. However, it should be noted, that in eXpert the project team worked a 24-hour week which increases the percentual effort proportion comparing to a "normal" 40-hour week.

Findings of the post-iteration workshops are shown in Figure 1 including positive and negative issues, and how many improvement actions they were followed by.

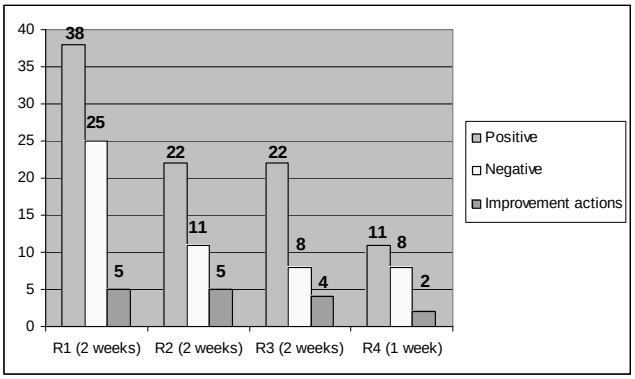

Fig. 1: Number of findings and improvement actions from the post-iteration workshop

The four post-iteration workshops resulted a total of 93 positive and 52 negative findings. Figure 1 shows the declining trend in both positive and negative findings. The positive findings decreased from 38 to 11 and negative from 25 to 8 findings per

workshop. Furthermore, the implemented process changes lessened during the project. This finding is in-line with that of Cockburn [2], who argued that the changes needed in the process will be much smaller after the second and subsequent increments.

However, other factors (than process satisfaction) might also influence the decline in positive and negative findings. For example, as the team became more accustomed to the adopted practices, their weaknesses and rewards may have been taken for granted. Moreover, as the post-iteration workshops were relatively close to each other (from one to two weeks apart) the team did not find it necessary to repeat the findings except for the most disrupting ones. However, the repeated positive and negative findings were recorded also in the subsequent workshops.

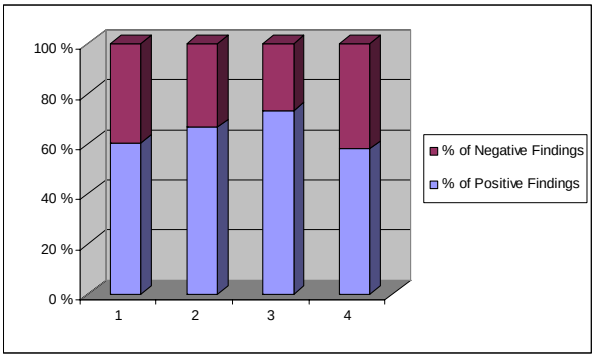

Fig. 2: Ratio of positive and negative findings

Figure 2 illustrates how the satisfaction of the project team evolved during the project by examining the ratio between positive and negative findings. During the first three iterations, the proportion of positive findings rises from 60% to 73% indicating increased satisfaction. In the fourth post-iteration workshop the proportion of positive findings dropped by 15%, while the amount of negative findings remained the same. The trend of three first iterations suggests that the process actually improves as a result of post-iteration workshops. However, this analysis is not yet strong enough to draw any conclusions. Furthermore, the closer examination of research data reveals that the topics causing negative findings became fewer during the project. The "topics" here refer to the post-it note groupings made according to the KJ-method to identify the specific problem areas for the findings. The data reveals that the criticism of the project team focuses on nine topic areas at the beginning of the project and declines rapidly to only two issues at the end. This analysis again supports the assumption that the XP process practices had actually self-adapted to the needs of the project team, i.e. increased their satisfaction for the process.

To explain the growing satisfaction that the data implicates, it should be reported that all except one suggestion for enhancing the software process practices were actually implemented. Alone, this power to influence on the daily working practices is likely to raise the positive atmosphere among the project team. Furthermore, the rapid and visible effect of the process changes is likely to satisfy the developers. The

spread between the negative findings and process changes can be explained by the fact that several of the negative findings needed no actions but were rather misunderstandings or other issues solved by learning through discussion during the post-iteration workshops.

Table 2 demonstrates the top five positive and negative findings during the entire project. Interestingly, the top positive finding was the controversial pair programming practice. It continued to appeal to team members during the entire project. Noteworthy is also the fact that all of the top five positive topics belong to the practices of XP. The top negative finding was time tracking. Due to the research character of the project, the collection of measurement data was heavy and time tracking detailed. Testing only became a negative issue towards the end of the project when the motivation of the outside testing group clearly decreased. Code commenting and time estimating findings generated mostly from the lack of proper standards and instructions. Test-driven development was found to be difficult as it was the first time the project team had encountered it and an experienced coach was not available.

Table 2: Top five of the positive and negative findings

	Top 5 positive findings	**Top 5 negative findings**
1	Pair programming	Time tracking
2	Short iterations	Testing
3	Continuous integration	Code commenting
4	On-site customer	Effort estimation
5	Refactoring	Test-first development

Table 3 presents a summary of the improvement suggestions and actions on the top 5 negative issues to provide an overview on what kind of enhancements arose from the post-iteration workshop in the eXpert case study. The lines in an *italic* font indicate suggestions that were not implemented during the project as the others are the actual action points for the next iteration.

Table 3: Process improvement suggestions

Practice	**Improvement Suggestions/actions**
Time Tracking	• New column to project hours-table for dump task (PM2) • Own task for refactoring comments (PM2) • Developing personal process for time tracking (PM3) • *Tool support for time tracking (PM5)*
Testing	• Internal audits (code reviews) (PM2) • Pre-release testing (PM1)
Code Commenting	• Refactor old comments (PM2) • Commenting on test code also (PM2) • Coding standards needed (PM2) x3 • *Improve commenting: do immediately and in detail (PM5) x3* • *Coding standard should be agreed at the beginning (PM5) x2*
Time Estimating	• Improve task descriptions: More exact tasks (PM3) • *Analyzing time estimates of previous releases (PM3)*
Test First	• Modify test cases to find errors (PM2)

3.2 Developer's perceptions

The project team felt that the post-iteration workshop technique was an effective and convenient way to learn because it summarized the previous iteration and forced each team member to think about its difficulties and negative aspects. This way each member was able to learn and improve their own actions and pay attention to negative issues even if they were not always written down as improvement suggestions. In addition, post-iteration workshops were seen as an efficient and honest way to improve the process because they actually forced the process to take a better direction. The project team found it very creative to discuss experiences and solutions in a group and to criticize the things that were done sloppily or could have been done better. The issues were brought to light and the improvement actions turned out to be successful, according to the project members. As an example, pre-release testing was brought as a new practice to the process and relieved the actual release testing with customer.

The opinion of the project team was that the post-iteration workshop, as applied in eXpert, didn't take too much effort yet still improved the weak aspects of the process significantly. When the development was done in short cycles the things agreed in post-iteration workshops stayed clear in the mind and the effects of improvements were noticeable in the following iterations. All developers were confident that post-iteration workshop was a way to get better outcome from the development process. All also favor using this technique in future projects when applicable.

4. Conclusions and Further Work

Agile software development relies on self-organizing teams and the Agile principles suggest that the team should regularly reflect on how to become more effective, and fine-tune and adjust its behavior accordingly. While some systematic approaches have been proposed on how to execute this self-reflection process effectively, little empirical evidence exists as yet. This paper has served for this purpose. Two known self-reflection approaches were combined and 4 post-iteration workshops were held on an XP project. The case study presented (eXpert) is the first in an ongoing series of Agile case studies conducted at the Technical Research Centre of Finland and provides a baseline for further replications for the progress of the post-iteration workshop technique.

Based on our experiences, the KJ method (see section 2.2. for details) proved to be an effective tool in adapting practices in an XP project, as was suggested by Dingsøyr and Hanssen [3]. The quantitative and qualitative findings from the case study support the assumption that with less than 4% effort it is possible to hold post-iteration workshops that concretely help improving and optimizing practices, and enhance the learning and satisfaction of the project team. The empirical data from the case study shows that the post-iteration workshops fine-tuned the development process and increased the project team’s satisfaction. A strong indication of the benefit of the post-iteration workshop was the positive remarks made by the developers.

However, this study lacks evaluation of the effects of process improvements on, for example, the effectiveness of the process or quality of the product. One reason for this is that the existing project level SPI techniques, including the post-iteration workshops, lack a detailed procedure for the follow-up of software process improvement actions, as well as their support with, for example, measurement data. Furthermore, the existing techniques lack important aspects in enhancing the extensive learning in the future projects. As a result, the post-iteration workshop technique has been evolved and is currently being applied for further evaluation in a third XP case study (bAmbie).

References

[1] P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, "New directions on agile methods: A comparative analysis," presented at International Conference on Software Engineering (ICSE25), Portland, Oregon, 2003.

[2] A. Cockburn, *Agile Software Development*. Boston: Addison-Wesley, 2002.

[3] T. Dingsøyr and G. K. Hanssen, "Extending Agile Methods: Postmortem Reviews as Extended Feedback," presented at 4th International Workshop on Learning Software Organizations (LSO'02)), Chicago, Illinois, USA, 2002.

[4] K. Beck, *Extreme Programming Explained: Embrace Change*: Addison Wesley Longman, Inc., 2000.

[5] I. Nonaka and H. Takeuchi, *The Knowledge-Creating Company*, 1995.

[6] R. Scupin, "The KJ Method: A Technique for Analyzing Data Derived from Japanese Ethnology," *Human Organization*, vol. 56, pp. 233-237, 1997.

[7] B. Collier, T. DeMarco, and P. Fearey, "A defined process for project post mortem review," *IEEE Software*, vol. 13, pp. 65-72, 1996.

[8] M. J. Tiedeman, "Post-mortems-methodology and experiences," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 176-180, 1990.

[9] J. B. Cunningham, "Case study principles for different types of cases," *Quality and quantity*, vol. 31, pp. 401-423, 1997.

[10] D. Avison, F. Lau, M. Myers, and P. A. Nielsen, "Action Research," *Communications of the ACM*, vol. 42, pp. 94-97, 1999.

[11] P. Oquist, "The epistemology of action research," *Acta Sociologica*, vol. 21, pp. 143-163, 1978.

[12] G. I. Susman and R. D. Evered, "An Assessment of the Scientific Merits of Action Research," *Administrative Science Quarterly*, vol. 23, pp. 582-603, 1978.

[13] M. Höst, B. Regnell, and C. Wohlin, "Using Students as Subjects - A Comparative Study of Students and Professionals in Lead-Time Impact Assessment," *Empirical Software Engineering*, vol. 5, pp. 201-214, 2000.

[14] K. Beck, "Embracing Change with Extreme Programming," *IEEE Computer*, vol. 32, pp. 70-77, 1999.

[15] P. Abrahamsson, "Extreme Programming: First Results from a Controlled Case Study," presented at 29th Euromicro Conference, Belek-Antalya, Turkey, 2003.