

Systematical Validation of Learning in Agile Software Development Environment

Outi Salo

VTT Technical Research Centre of Finland
P.O. Box 1100, FIN-90571 Oulu, Finland
Outi.Salo@vtt.fi

Abstract. This paper illustrates implications from four case studies in which Agile software development teams conducted iterative project retrospectives to improve and adapt their software development processes. It was detected that the existing techniques lack a systematic approach to iteratively validate the implementation and effectiveness of software process improvement actions with both quantitative and qualitative data. Also, the case studies revealed that the organizational level can only benefit from the learning of project teams if the knowledge and reasoning behind the process improvements is converted into such an explicit format that it can be utilized for learning in organizational level also. Thus, this paper illustrates how these deprivations were accomplished in the case projects with the support of a structured template.

1 Introduction

Agile software development offers both need and opportunity for adapting and improving the software development process rapidly and effectively. One of the Agile principles (<http://agilemanifesto.org/principles.html>) is that the team should regularly reflect on how to become more effective, and tune and adjust its behavior accordingly. The short development cycles provide continuous and rapid loops to iteratively enhance the process during Agile software development projects.

Some techniques suggest how the learning of the project team could be iteratively transferred into concrete software process improvements in Agile software development, namely a reflection workshop technique [1] and postmortem reviews [2]. While both of these techniques most likely accomplish this goal, they both seem to lack the means to systematically validate the effects of the improvements. However, Extreme Programming (XP) [3], for one, requires that the teams can change the rules only if they also agree on how they will assess the effects of the change [4]. Nor do the existing techniques provide guidance on how the tacit or even explicit knowledge and learning of the project teams can be converted into an explicit form that can be utilized in organizational learning as well.

In this paper, an extension for the existing project retrospective techniques is proposed to support the systematic implementation and validation of software process improvements in Agile projects using both quantitative and qualitative data, and to

provide organizational level with insights into the projects. The propositions presented in this paper are particularly suitable for Agile software development context where the continuum of short iterations provide the opportunity for such activities.

2 Research Context

The research was conducted at VTT Technical Research Centre of Finland in four Agile case studies (Table 1). The case projects used an XP [3] based software development process that evolved during the case projects to include relevant extensions such as SPI activities.

Table 1. Characteristics of the case projects

Characteristic	eXpert	zOmbie	bAmbie	uniCorn
Team size	4 developers	5.5 developers	4 developers	6 developers
Total team effort	7.5 PM	10 PM	5.5 PM	5.2 PM
End product	Intranet app	Mobile app	Mobile app	Mobile app
Iterations	3 x two weeks 3 x one week	1 x one week 3 x two weeks 2 x one week	1 x one week 3 x two weeks 2 x one week	2 x two weeks 7 x one week

In all case projects, project retrospectives were conducted iteratively to improve and adapt the software development process. The last retrospective in every project was considered a project postmortem, contributing more directly to the organizational level software process improvement. All the previous retrospectives, however, concentrated on the iterative learning and improvement of a single project yet also supported a “bottom-up approach” for organizational learning.

The technique used, i.e. Post-Iteration Workshops, (hereafter referred as PIWs) combines and adapts certain elements from both the workshop technique [1] and the postmortem review technique [2] (as presented in [5]). Furthermore, it includes modifications and extensions evolved and experimented in the series of case studies.

3 Systematic Validation of Process Improvements

Soon after the first case study it was realized that a systematic way to validate the effects of learning (i.e., the process improvement actions agreed by the project team in the retrospectives) was needed for two specific reasons. Firstly, the project itself could control the implementation of the agreed process enhancements and evaluate whether they actually improved the process. Secondly, the organization needed a way to examine how and why individual projects adapted their base process, and how successfully. Without this kind of explicit knowledge supported with qualitative and quantitative data, the process changes in organizational level would be a shot in the dark rather than validated learning to be shared and diffused organizationally.

A structured template (i.e., an Action Point List) was generated to provide guidance for project team on the issues to be considered for every process improvement action and information for both project and organizational levels of how the SPI in individual projects made progress. It was filled in every PIW session with the moderator and the project team. Also, the list from the previous PIW was updated regarding the validation of earlier SPI actions according to the validation plan. Figure 1 illustrates the structure of the template along with a set of examples from the case projects.

Improvement Topic: Test-Driven Development				
Finding	Action Point	Actor	Validation plan	Validation
Deficiency and lacking of TDD test cases caused by absent knowledge of TDD.	Support team will coach the team the next planning day Analyse the quality and coverage of test cases (1 st and 2 nd iteration)	Project Manager Tracker	Team will interpret the data from 1 st and 2 nd iteration in the next PIW to evaluate the improvement.	Situation not improved based on the experiences of the team and the metrics data. More support needed for the next iteration (=> new ap).
Improvement Topic: Configuration management				
Finding	Action Point	Actor	Validation plan	Validation
Corrections made during release day caused irrecoverable situation.	Baseline made at the beginning of release day (instead at the end)	Project Manager	Team will discuss the improvement of the situation in the next PIW.	Working fine according to the team. Ongoing practice.
Improvement Topic: Task estimation				
Finding	Action Point	Actor	Validation plan	Validation
Task estimations too inaccurate in release 1. => release delayed.	Analyse task data to evaluate the cause of delays in each task.	Tracker	Team interprets the data in the next PIW to find ways to improve.	Data revealed too big tasks. Splitting needed.
Splitting of tasks needed to improve effort estimation.	Task will be split to max. 4 hours size. Analysis of effort estimations.	Project Manager Tracker	Interpret the metrics data in the next PIW to see if effort estimations improved.	Smaller task size seemed to improve effort estimation. 4 hours is a good size for task when possible.

Fig. 1. Example of filled Action Point List

In the PIW's, the problematic issues that the software developers had faced during the previous iteration were generated and structured using the KJ method [6], as also suggested in postmortem review technique [2]. These grouped and labeled findings were then the basis for focus group discussion [7] that aimed for discovering and agreeing process enhancements for the subsequent iteration. The "Improvement Topic" field in Action Point List refers to the labels of the grouped findings that the project team generated on the flap board during the PIW. This way the process improvements could be traced all the way back to the groups of findings in certain workshop. The purpose of the "Finding" field is to provide sufficiently detailed explicit knowledge of the specific problems in each topic area based on the discussion of the project team. "Action point" field clarifies what is the mutually agreed concrete action to be taken in the next iteration to improve the situation. "Actor" defines the

responsibilities for each action point to ensure its implementation. The “Validation plan” field is used to record the project team’s decisions on how the success of the improvement will be assessed effectively and also to reveal the schedule of the action. The validation should be carefully considered for each action point. The qualitative data (i.e. experience) of the software developers is always needed in validating if a certain process enhancement really is an improvement. Often, however, quantitative verification is also needed, especially for the organizational level. In such a case the availability and analysis of the metrics needed should be planned at this point. The “Validation” field is filled in the next PIW as the validation is carried out as planned including the possible interpretations of analyzed data. In practice, the data interpretation and sharing of experiences during a group discussion often generated new improvement opportunities and action points. Also, one goal of validation is to agree if the specific process improvement should be further employed or rejected.

It can be said that the changes made in the project level were usually fairly small, yet effective enough to ease up the daily work of the project team and increase their motivation [5, 8]. The more radical decisions on improving the process were made in the organizational level often based on the ideas generated by the project teams. These improvement opportunities (e.g., changes to organizational data tracking tools) could not be done on the project level alone.

4 Discussion

Agile software development provides an opportunity to iteratively generate and implement enhancements during the software development process. In the case studies, the PIWs were conducted based partly on the two existing Agile techniques, namely a reflection workshop technique [1] and postmortem reviews [2]. They were found to be effective and motivating in improving and adapting the software development process in Agile case projects [5, 7]. However, it was realized that the earlier suggested techniques lacked the mechanisms to support the iterative implementation and evaluation of the process improvement actions, and thus, they were included in the PIW technique. Also, it was realized that the organizational level will benefit from the learning of projects only if the knowledge and reasoning behind the process improvements are converted into such an explicit form that can be utilized for learning on the organizational level as well.

Thus, this paper discusses how the systematic implementation and verification of the software process improvements as well as the flow of process knowledge from projects to organizational level can be supported with a template. It guides the project team to the issues to be considered for every process improvement action, including the quantitative follow-up of the process enhancements. The sequential “Action Point Lists” also provide a means to transfer the knowledge between project and organizational levels and should be supplemented with the analysis and interpreted of quantitative data when needed. As the organizational level can view concrete chains of reasoning behind the software process improvement actions - supported with both quantitative and qualitative data, it can gain new ideas and insights from single projects which is essential if learning is to take place on the organizational level [8].

The focus of this paper is to point out the importance of the systematic follow-up and validation of software process improvements in iterative SPI cycles during Agile software development and to discuss how the benefits of this kind of project level software process improvement should be utilized in organizational level also. This paper is an important part of research of continuous software process improvement in Agile software development context. The thorough analysis of the research data for validating the proposed solution and as well as the presentation of the PIW technique are out of the scope of this paper.

5. Acknowledgements

This study is done in Agile-ITEA project funded by TEKES (National Technology Agency of Finland). It is part of larger research topic of continuous software process improvement in Agile software development context. Acknowledgements to the software developers of all the case projects for their thorough participation in the process improvement activities. Also, sincere thanks to Dr. Pekka Abrahamsson, Hanna Hulkko and Minna Pikkarainen for their valuable comments.

References

1. Cockburn, A., *Agile Software Development*. The Agile Software Development Series, ed. A. Cockburn and J. Highsmith. 2002, Boston: Addison-Wesley. 278.
2. Dingsøy, T. and G.K. Hanssen. *Extending Agile Methods: Postmortem Reviews as Extended Feedback*. in *4th International Workshop on Learning Software Organizations (LSO'02)*. 2002. Chicago, Illinois, USA.
3. Beck, K., *Extreme Programming Explained: Embrace Change*. 2000: Addison Wesley Longman, Inc. 190.
4. Beck, K., *Embracing Change with Extreme Programming*. IEEE Computer, 1999. **32**(10): p. 70-77.
5. Salo, O., et al. *Self-Adaptability of Agile Software Processes: A Case Study on Post-Iteration Workshops*. in *5th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2004)*. 2004. Garmisch-Partenkirchen, Germany: Springer.
6. Scupin, R., *The KJ Method: A Technique for Analyzing Data Derived from Japanese Ethnology*. Human Organization, 1997. **56**(2): p. 233-237.
7. Kerlinger, F.N. and H.B. Lee, *Foundations of Behavioral Research*. Fourth Edition ed. 2002: Harcourt College Publishers. 890.
8. Salo, O. *Improving Software Process in Agile Software Development Projects: Results from Two XP Case Studies*. in *EUROMICRO 2004*. 2004. Rennes, France: IEEE Computer Society Press.
9. Garvin, D.A., *Learning in Action*. 2000, Boston, Massachusetts: Harvard Business School Press. 256.