

Experiences of Using Extreme Programming to Support a Legacy Information System Migration Project

Juha Koskela¹, Mauri Myllyaho², Jukka Kääräinen¹, Dan Bendas², Jarkko Hyysalo², Anne Virta²

¹ VTT Technical Research Centre of Finland
P.O. Box 1100, FIN-90571 Oulu, FINLAND
Phone: +358 8 551 2206
Fax: +358 8 551 2320
juha.koskela@vtt.fi, jukka.kaarainen@vtt.fi

² Department of Information Processing Science,
P.O. Box 3000, FIN-90014 University of Oulu, FINLAND
mauri.myllyaho@oulu.fi, dan.bendas@oulu.fi, jarkko.hyysalo@oulu.fi, annevirt@paju.oulu.fi

Abstract

In the recent years, the software engineering community has been showing a growing interest in agile development methodologies. These have emerged out of the need for faster, more flexible and efficient processes for software development. Currently the best known agile method is extreme programming (XP). XP addresses issues concerning changing requirements and their cost by simplifying management tasks and project documentation. XP uses an iterative and incremental software process executed in relatively short cycles. Traditionally this type of approach would yield an increased management overhead because the management activities related to the ending and starting of iteration have to be executed for every iteration, while in the XP process these activities are minimized.

Currently, there are lots of experience reports available on applying the XP method to different types of software projects. However, there exist very few experience reports of XP projects where the database is the most central issue of the project. Thus, the objective of this paper is to report and examine experiences and empirical data while using XP practices to support a legacy information system migration project. The work was performed during 4 months, with a total effort of 1056 hours. The project followed the agile methodology principles by applying a series of XP practices (e.g. planning game and small releases). Two previous migration attempts had failed. These efforts had followed the traditional waterfall approach, finishing already in the design phase, producing mainly design documents, but no description of the technique for transferring the data. In addition, the design documents produced were highly complicated and did not completely correspond to customers' needs. This paper attempts to understand the reasons for these failures, while also compiling a list of lessons learned. This study brings empirical data supporting the idea that XP practices and documentation for need are applicable not only to pure development projects (where source code is the most important result), but also for other types of software projects, where the main activities may be design (not product) oriented. The project was an engineering success, managing to find solutions to migrating data to the format required by the new information system. The migration itself is planned to be carried out in the near future.

Keywords

Legacy system, Extreme programming, Database development

1 INTRODUCTION

In the recent years, the software engineering community has been showing a growing interest in agile development methodologies. These approaches have emerged out of the need for faster, more flexible and efficient processes for software development. Currently the best known among these methods is extreme programming (XP), which was first introduced by Kent Beck [1]. The basic idea of XP methodology is open communication between the customer and the project team as well as among the project team. In XP, the communication between stakeholders is addressed by means of face-to-face feedback and such practices as “On-site customer”, “Planning Game” and “Pair Programming”.

The objective of the paper is to report and examine experiences and empirical data, while using XP practices to support a legacy system migration project. The work was performed during 4 months, with a total effort of 1056 hours. The project followed the agile methodology principles by applying a series of XP practices. Two previous attempts at migrating the system had failed. These followed the traditional waterfall approach, and ended in the design phase, producing mostly design documents, but no description of the technique for transferring the data. This paper attempts to understand the reasons for the failures, compiling a list of lessons learned.

The paper will be composed as follows. The background concept of extreme programming will be introduced first. Then the paper will look into related research concerning software and database development using agile methods. Next, the migration project, its background and the research settings will be described. Finally, the results will be presented and discussed. The paper will be concluded with final remarks and an identification of future research needs.

2 BACKGROUND

This section introduces the basic concept of extreme programming and reviews related research.

2.1 Extreme Programming

Agile methods have gained a significant amount of attention in the field of software engineering in the last few years. Currently, extreme programming is the best known agile method. XP as a concept emerged in the late 90's along with Kent Beck's book “Extreme Programming explained: Embrace Change” [2]. Other “agile” methods have emerged along with XP (for an overview of other agile methods see, e.g., [3, 4]). XP is primarily designed for object-oriented projects using a maximum of a dozen programmers at one location [5]. This kind of situation is called “agile home ground” by Boehm [6]. The XP process is characterized by short development cycles, incremental planning, continuous feedback, and reliance on communication and evolutionary design [2]. The core of XP is made up of a simple set of common-sense practices. These practices are planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40-hour week, on-site customer, coding standards, open workspace and just rules. XP addresses issues concerning changing requirements and their cost by simplifying management tasks and documentation. XP uses an iterative and incremental software process carried out in relatively short cycles. Traditionally this type of approach would yield an increased management overhead because management activities related to the ending and starting of an iteration have to be executed for every iteration, while these are minimized in the XP process. XP may first seem fairly chaotic, but it includes several good engineering practices [7].

Product development in the XP process starts with a “planning game.” Planning game can be divided into “release planning” and “iteration planning” [8]. During the planning game, user stories are written, estimated and prioritized. In the planning game phase of XP development requirements, which are stories, are elicited, estimated and selected for release. Planning game is performed for each release. A release is divided into iterations and, therefore, may contain one or more iterations. A subset of stories based on priority and size is then selected for each iteration. This is called iteration planning. Developers then divide stories into tasks and give an estimate for each task. The next step in the XP process is the development during which iterations are produced and released.

2.2 Related research

There are lots of experience reports available on applying the XP method to different types of software projects (see e.g. <http://www.agilealliance.org>). For example, Poole and Huisman [9] report their experiences on using XP in a maintenance environment, and Elssamadisy [10] and Wright [11] discuss applying XP in large projects. However, there exist very few experience reports of XP projects where the database or the legacy information

system is the most central issue of the project. The use of agile software development in database related projects has been discussed, e.g., in [12-14]. Schuh [12] states that agile software development teams often pay little attention towards database issues. Therefore, Schuh introduces a strategy for implementing a flexible database infrastructure to complement the continuous integration-style development approach. Hassan and Elssamadisy [13] report their experiences and problems with a large XP project involving a separate database administration (DBA) team. According to Hassan and Elssamadisy [13], all the problems they had seemed to originate from the fact that they had two separate teams with different methodologies and poor communication between them. They further state that many of the problems they encountered could have been reduced by the introduction of testing, pairing, DBA involvement in the planning game, and through more collective ownership. Ambler [14] introduces the Agile Data (AD) method in his work *Agile database techniques*. The AD method emphasizes the importance of data in software-based systems and defines strategies that enable software developers to work together effectively on the data aspects of software systems. The principles of agile software development [3] constitute the basis for the AD method, while it is extended by philosophies reflecting the realities faced by data professionals (e.g. "Data", "Teamwork" and "Every project is unique") [14]. The book also addresses issues and problems concerning legacy databases. According to Ambler [14], the problems faced with legacy databases are often very difficult to fix and require a lot of learning effort.

3 PROJECT AND RESEARCH SETTINGS

This section presents the background of the project and how the project was set-up and the research data collected.

The subject of the project was the 20-year-old information system of the University of Oulu Botanical Garden, which consists in a user-searchable database of plant specimens with taxonomical designation and growth information. The system was built on top of a MUMPS hierarchical database environment, with a server running on MS-DOS, supporting text-based terminals via serial link. At the time when the migration project started, the legacy information systems contained close to 100MB of data encoded in a proprietary format, mixing database records with executable code sequences. The legacy information system including the data needed to be available for customers during the project.

Two previous attempts at migrating the system had failed. These had followed the traditional waterfall approach, and ended in the design phase, producing mostly design documents, and no description of the technique for transferring the data. The current migration project called "Unikko" (hereafter referred as "the project") was performed during 4 months, with a total effort of 1056 hours. The project team consisted of 4 developers and a project manager. The team members were 2nd-6th year students from the department of information processing science with 2-4 years of experience in programming. They were working in a project room (open workspace) in the Botanical Garden of the University of Oulu. The room was equipped to meet the requirements of XP development (e.g. board for stories/tasks). The project team could freely arrange the room as they wished. The customer's representatives were not on-site in the same room as recommended in the XP literature [e.g. 2, 5, 15]. However, they were working in the same building and available when needed (not later than next day). The project followed the agile methodology principles and applied a series of XP practices. The project team was allowed to decide which practices they found suitable for the project. Some additional practices were needed for documentation and research purposes, e.g., metrics collection and documentation according the rules of the University of Oulu.

The project team had an XP training session prior to project start-up. Quantitative and qualitative data were collected throughout the project. The team members had development diaries for time tracking and pre-defined Excel sheets were used for recording the metrics data collected during the day. The metrics responsible was monitoring the data collection systematically. The metrics and practices for gathering quantitative data were defined before the project start-up. VTT Electronics had earlier experience on how to collect metrics from more typical XP projects, such as eXpert [16, 17] and zOmbie [18]. Qualitative data was collected at the meetings with the project team after the delivery of each major milestone. The received comments were presented and analyzed in a series of Post Mortem sessions applying guidelines from [19, 20].

4 RESULTS

This section presents the quantitative and qualitative data collected during the project. Table 1 summarizes the basic quantitative data from the project. It provides basic information about the size and schedule of the project and helps the reader to understand the nature of the project.

Table 1. Background data of the project (5 persons in the core development team).

Id	Collected data	Orientation	Release 1	Release 2	Release 3	Documentation	Total
1	Calendar time (weeks)	1	6,2	4	3,8	1,6	17,4
2	Total work effort (h)	27,8	455,7	293,9	196,8	81,5	1055,7
3	Task allocated actual hours (h) *	0,0	158,8	124,9	75,5	0,0	359,2
4	Task allocated actual hours (%) **	0	35	43	38	0	34
5	# User stories implemented	0	3	3	3	0	9
6	# Tasks implemented	0	16	18	9	0	43
7	# Tasks postponed to next release	0	5	4	0	0	9
8	Required customer involvement (h)	0,0	1,2	9,8	1,1	1,9	14,0

*Task allocated actual hours (h) = amount of person hours from "Total work effort (h)" that have been allocated for tasks

**Task allocated actual hours (%) = 100 * (Task allocated actual hours [h]) / (Total work effort [h])

Table 2 summarizes the most important qualitative results of the Post Mortem sessions. The comments relate to the development practices used during the project. Table 2 also indicates which practices were used during the project.

Table 2. Comments and remarks.

XP Practice	Comments ("+" advantage, "-" disadvantage, "*" authors' remark)
Planning game	+ stories and tasks are visible and easily available on board + easy to check what needs to be done and what has already been done + planning game clarifies the execution of the project + requirements management easier in this kind of development
Small releases	+ easier to manage small releases + when all requirements are not clear at the beginning, it is easier to implement clear requirements first - scheduling turned out difficult - too many releases for project as short as this <i>* the content of releases is different in database oriented projects than usually thought (e.g. in this project the ER diagram was the central result of the first release)</i>
Metaphor	<i>* not used in this project</i>
Simple design	<i>* was identified as useful practice during Post Mortem (no other comments)</i>
Testing	<i>* not used in this project (not applicable)</i>
Refactoring	<i>* no comments, however was used fairly extensively while creating the ER diagram structure</i>
Pair programming	+ fun and efficient way of working + easier to understand complicated tasks + everybody is doing everything (not getting bored)
Collective ownership	<i>* not used in this project</i>
Continuous integration	<i>* not used in this project</i>
40-hour week	+ not working overtime + makes working more efficient <i>* working week was agreed to be less than 20 hours</i>
On-site customer	+ customer is available when needed + easy to arrange meetings + easy to discuss also unofficially (coffee breaks etc.) <i>* customer representatives worked in the same building, but not in the same room with developers</i>
Coding standards	<i>* used in some degree in this project</i>
Open workspace	+ stories and tasks are visible and easily available on board
Just rules	+ project team can use common sense when applying practices
Other guidelines	
Metrics collection	+ metrics can prove helpful in assessing the time spent in different tasks and in understanding the status of the project + helps to find personal areas that need improvement or study - metrics collection took too much effort - utilization of metrics was unclear - too many metrics and their collection was error prone
Light documentation	+ light [project] documentation is a good thing - some pre-defined document templates were unclear
Other issues	
Domain area	+ learning a new domain area was found fruitful - weak knowledge of the domain area at the beginning of the project - getting acquainted with the domain area took a lot of time - problems in understanding between team and customers

Table 2 reveals that four of the XP practices were not used during the project. These unused practices were metaphor, collective ownership, continuous integration and testing. Since the project was not a traditional

software development endeavor but rather a data migration venture, it is understandable that these practices were not used. Pair programming was used when creating ER diagrams, for example.

5 DISCUSSION

The qualitative data presented in the previous section indicates that the most successful XP practices were planning game, small releases, pair programming, 40-hour week, on-site customer, open workspace and just rules. These practices enhanced communication and feedback between all the parties involved: project team, project manager and customer. Product information, such as stories and tasks were visible for the whole project team and customers, and the work was split into understandable releases. The importance of good communication and feedback during agile development has been stressed by Williams [21] and Ambler [22]. However, task creation and estimation turned out to be somewhat difficult at the beginning. This problem has also been addressed by Abrahamsson and Koskela [17] presenting the "XP Pulse". The pulse shows that at the beginning the estimation of tasks often proves difficult, while it is likely to become more accurate in later releases.

On-site customer is one of the XP key practices. According to XP literature [e.g. 2, 5, 15], the customer optimally works in the same room with the developers. During this project the customer was not "on-site" in the same sense as in traditional XP. The project team worked in a customer's office and the customer was available when needed. This arrangement is defined as an off-site customer by Wallace et al. [23]. According to the first author's experiences of being an on-site customer in an XP project [24], the customer's location during this project was perhaps the most suitable for an XP project as it considers both developers' and customer's viewpoints; developers can still contact the customer easily and the customer is able work in a more peaceful workplace. The results of this study support this finding. The customer's representatives were available when needed during the project (not later than next working day). The project team needed a total of 14 hours of customer assistance during the project. At the beginning of the project the team experienced some communication problems when discussing with the customers. This was mainly due to a weak knowledge of the domain area. However, the project team managed to learn the specialties of the domain area during the project and was finally quite satisfied with this new knowledge they had acquired.

According to the lessons learned presented by Ambler [22], it is useful to produce just enough documentation and to update it only when needed. The findings of this study support this claim. Due to the nature of the project, the team, in fact, produced quite a lot of documentation, but only for a real purpose. According to Ambler [22], many traditional projects produce far more documentation than they need. As suggested in the agile principles [3], "Simplicity - the art of maximizing the amount of work not done - is essential", the produced documents were made as simple and as readable as possible. Earlier migration attempts had stumbled upon documentation issues. Simple documentation for a real purpose seemed to enable the project team to produce more results and to deliver more value for the customer than traditional plan-driven methodologies.

Metrics collection was considered the least welcome part of the project among the project team. The team reported that metrics collection took too much effort. However, the collection of project metrics is essential for research and project management purposes. The qualitative data shows that this was understood by the project team. The technique that was used for collecting the metrics was just too time consuming and error prone. The need for supporting project management in the means of effort collection and calculation was also identified in an earlier VTT XP project [18]. This deficiency can be fixed by constructing a computerized solution for supporting the metrics collection, processing and reporting.

In the literature, there are numerous experience reports available on applying the XP method to different types of software projects [e.g. 9, 10, 11]. In this XP case study the main focus of the project was on data migration. Compared with the previous attempts, the project was a success. This study brings empirical data supporting the idea that XP practices and documentation for need are applicable not only to software development projects, but also to other types of software activities. Furthermore, Ambler [14] also emphasizes the importance of effective teamwork in data oriented projects. The findings of this study thoroughly support this argument. The project team succeeded in working efficiently together and they also had clear goals for the project. This proved to further boost the working motivation of the team.

6 CONCLUSIONS

This paper reports the experiences of using the XP method to support a legacy information system migration project. The project followed the agile methodology principles and applied a set of XP practices. Two previous

migration attempts using different methods concerning the same information system had failed. The unsuccessful endeavors had followed the traditional waterfall approach, both finishing already in the design phase, producing mainly design documents, but no description of the technique for transferring the plant data from the old information system. Moreover, the design documents produced in the earlier projects were highly complicated and did not completely correspond to customers' needs. This study brings empirical data supporting the idea that XP concepts like “planning game”, “short releases” and “documentation for need” are applicable not only to pure development projects (where source code is the most important result), but also for other types of software projects, where the main activities may be design oriented. In conclusion, the project was an engineering success, managing to find solutions for migrating the data to the format required by the new information system, which is planned to be carried out in the near future.

Acknowledgements

The work was funded by the Technical Research Centre of Finland (VTT), the University of Oulu and by the National Technology Agency (TEKES). The authors would also like to acknowledge the contributions from several colleagues with the Technical Research Centre of Finland (VTT) and the University of Oulu. Moreover, the authors would like to thank the Unikko project team for their effort.

References:

- [1] K. Beck, "Embracing change with extreme programming," *IEEE Computer*, pp. 70-77, 1999.
- [2] K. Beck, *Extreme programming explained: Embrace change*. Reading, MA.: Addison Wesley Longman, Inc., 2000.
- [3] K. Beck, M. Beedle, A. Bennekum van, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for Agile Software Development," vol. 2002, 2001.
- [4] 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.
- [5] R. Jeffries, A. Anderson, and C. Hendrickson, *Extreme Programming Installed*. Upper Saddle River, NJ: Addison-Wesley, 2001.
- [6] B. Boehm, "Get Ready For The Agile Methods, With Care," *Computer*, vol. 35, pp. 64-69, 2002.
- [7] N. Paulk, "Extreme programming from a CMM perspective," *IEEE Software*, vol. 18, pp. 19-26, 2001.
- [8] K. Beck and M. Fowler, *Planning extreme programming*. New York: Addison-Wesley, 2001.
- [9] C. Poole and J. W. Huisman, "Using Extreme Programming in a Maintenance Environment," *IEEE Software*, vol. 18, pp. 42-50, 2001.
- [10] A. Elssamadisy, "XP on a Large Project - A Developer's View: Extended Abstract," presented at XP Universe, Raleigh, NC, 2001.
- [11] G. Wright, "Extreme Programming in a Hostile Environment," presented at XP 2002, Alghero, Sardinia, Italy, 2002.
- [12] P. Schuh, "Agility and the Database," presented at XP 2002, Genoa, Italy, 2002.
- [13] A. Hassan and A. Elssamadisy, "Extreme Programming and Database Administration: Problems, Solutions and Issues," presented at XP 2002, Genoa, Italy, 2002.
- [14] S. Ambler, *Agile database techniques: effective strategies for the agile software developer*: Hoboken, NJ, Wiley, 2003.
- [15] R. C. Martin, *Agile Software Development, Principles, Patterns, and Practices*. Englewood Cliffs, NJ: Prentice Hall, 2002.
- [16] P. Abrahamsson, "Extreme programming: First results from a controlled case study," presented at Euromicro 2003, Antalya, Turkey, 2003.
- [17] P. Abrahamsson and J. Koskela, "Extreme Programming: A Survey of Empirical Data from a Controlled Case Study," presented at ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2004), Redondo Beach, CA, USA, 2004.
- [18] J. Kääriäinen, J. Koskela, P. Abrahamsson, and J. Takalo, "Improving Requirements Management in Extreme Programming with Tool Support - an Improvement Attempt That Failed," presented at Euromicro 2004 conference in track on "Software Process and Product Improvement", Euromicro 2004, Rennes, France, 2004.
- [19] T. Dingsøyr and G. K. Hanssen, "Extending Agile Methods: Postmortem Reviews as Extended Feedback," presented at 4th International Workshop on Learning Software Organizations, Chicago, Illinois, USA, 2002.
- [20] 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," presented at 5th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2004), Garmisch-Partenkirchen, Germany, 2004.
- [21] L. Williams, "The XP Programmer: The Few-Minutes Programmer," *IEEE Software*, vol. 20, pp. 16-20, 2003.
- [22] S. Ambler, "Lessons in Agility from Internet-Based Development," *IEEE Software*, vol. 19, pp. 66 - 73, 2002.
- [23] P. Wallace, P. Bailey, and N. Ashworth, "Managing XP with Multiple or Remote Customers," presented at XP 2002, Sardinia, Italy, 2002.
- [24] J. Koskela and P. Abrahamsson, "On-Site Customer in an XP Project: Empirical Results from a Case Study," to be presented at European Software Process Improvement Conference (EuroSPI 2004), Trondheim, Norway, 2004.