Introducing Extreme Programming – An Experience Report

Daniel Karlström

Dept. Communication Systems,

Lund University,

Box 118, SE-221 00 Lund,

Sweden.

Daniel.Karlstrom@telecom.lth.se

Abstract

This paper presents a single case study reporting the experiences of introducing extreme programming (XP) *in a small development project at Online Telemarketing in Lund, Sweden. The project was a success despite the fact that the customer had a poor idea of the system required at the start of the development. This success is partly due to the introduction of practically all of the XP practices. The practices that worked best were the planning game, collective ownership and customer on site. The practices that were found hard to introduce and not so successful were small releases and testing.*

INTRODUCTION

Extreme programming (XP) [1] is a methodology that has received much attention during 2000 and 2001. XP is a package of several practices and ideas, most of which are not new. The combination and packaging of all of these is, however new. One of the features that makes XP different to most other methodologies is that it is centred on the developer and gives him or her more responsibility in the creation of the product. This paper provides an experience report from the introduction of XP at Online Telemarketing in Lund, Sweden. The company decided to use XP to develop a sales support system for use in their principal line of business, telemarketing. The paper presents a brief introduction to qualitative research methodology, which can be said to be the research methodology used for this experience report in Section 2. Section 3 contains a brief introduction to the company, followed by an introduction to the development project in section 4. The experiences of the XP practices are accounted for in section 5 and a discussion of the quality of the conclusions is accounted for in section 6. Finally the conclusions and a summary are presented in section 7.

So far, relatively few experience reports have been made available with regards to XP. Especially, well structured reports of attempts to fully introduce XP are rare. Experience reports not only provide insight into specific situations in which the method may work and not work, but also provide practical examples to illustrate the method. Organisations considering XP can gain much needed prior experience of what to expect when introducing practices, irrespective if they are implementing one practice or implementing XP fully.

METHODOLOGY

The majority of the information presented in this experience report was gathered in two different ways. The first way was by direct observation of the developers during the course of the project, and the second was by interviews with both the developers and the development management. The interviews with the developers gave a lot of information about attitudes towards different XP practices, while the interviews with the management gave information mostly about how the practices were being followed.

As the information presented is of a qualitative nature, a brief discussion of qualitative methodology and threats is in order. It should be mentioned that the study performed is not intended to be a complete formal qualitative investigation and that auditing is not used to validate the results [7]. This kind of validation is only applicable and practical in much larger studies. By addressing the methodology behind the research techniques we can at least make an informed attempt at improving the quality of the information obtained.

In qualitative research the trustworthiness of the investigation, which is usually called validity in quantitative research, can be addressed using four criteria: *credibility*, *transferability*, *dependability* and *confirmability* [6].

These criteria are briefly summarised below. Further information can be found in the works of Lincoln [6], Robson [7], and Miles and Huberman [8]. References are made in the summaries below to corresponding criteria in quantitative validity theory. Wohlin et al. [9] contains a comprehensive quantitative validity section.

- *Credibility* corresponds to internal validity in quantitative research. The aim of this criterion is to ensure that the subject of the enquiry has been accurately identified and described. This can be achieved by, for example, triangulation of sources or methods.
- *Transferability* corresponds to external validity in quantitative research. This criterion addresses how far outside the observed domain the results are applicable.

- *Dependability* addresses whether the process of the study produces the same results, independent of time, researcher and method.
- *Confirmability* addresses the issue of researcher biases and ensures that the researcher affects the results as little as possible.

An attempt to evaluate the study according to these four criteria is performed in the quality of conclusions section, section 6.

ONLINE TELEMARKETING

Online Telemarketing is a small company specialising in telephone-based sales of third party goods. The company has its head office in Lund, Sweden, and regional branches in Uppsala, Visby and Umeå. Recently the company has expanded internationally with operations in Denmark, Norway and Finland. The company consists of a small core of fulltime staff that manages and supports a large number of temporarily employed staff. This implies that the company has a very flat organisation. The primary task of the temporary staff is performing the actual sales calls.

Management realised in the autumn of 2000 that a new sales support system would be required and started planning for a system for use within the company. 'Commercial off the shelf' (COTS) alternatives were evaluated but discarded due to being too expensive and due to the fact that it would be both difficult and expensive to incorporate specialised functionality. The management at Online Telemarketing had several novel ideas for features not present in the systems available on the market that they considered crucial for the future expansion and business success of the company.

The person responsible for systems development at Online Telemarketing realised that the lack of detailed requirements from management and the fact that no similar systems had been created before meant that traditional development with a big up-front design and detailed requirements documents would prove expensive and not very efficient. An alternative was found in XP [1].

THE DEVELOPMENT PROJECT

Project overview

Online Telemarketing decided on a strategy for developing the product that involved using their own system responsible person and employing part time developers to perform the coding work. To start with, four systems-engineering students were employed part time in parallel with their coursework at the university. After three months a further four people were employed and integrated into the development team in order to increase the absolute velocity of the project. The developers were employed as regular employees and there was no connection whatsoever between their position at Online and their university course-work. The employees were selected by interviewing applicants answering adverts placed throughout the student community both virtual and real.

The product was coded using Microsoft Visual Basic and SQL in a Microsoft development environment. The customer for the project was internal at Online Telemarketing and no considerations were made for eventually selling the product outside the company.
The size of the product is estimated to approximately 10 000 lines of code after all the initial functionality has been developed. The development was started in December 2000 and the first functional system was launched in mid April 2001. The system has been in full commercial operation since the end of August 2001.

Roles

The traditional XP roles described by Beck in [1] were assigned to the various members of the team at the start of the project. The employed developers assumed the roles of programmer. They also assumed the roles of testers, working together with the customers to create and run functional tests. The senior management at Online assumed the role of customer, as they were the people who had the original idea of the system. The tracker's responsibilities were assumed by the IT executive at Online as he had a good overview of the work performed by the group and was in direct contact with the developers daily. The coach role was assumed mainly by the IT executive, but at the beginning of the project, when XP was new to the team, the author shared some of the coach's responsibilities. Finally the Online senior management also assumed the roles of bosses for the project as they were providing all means for the development, such as computers, location and funding.

Configuration management

The configuration management was solved by a simple solution. As there were no branches in the configuration management and the system was relatively small, the team used a checkout directory to copy source code manually instead of using a tool for this purpose. This solution proved effective during the first part of the project when only two pairs of programmers were working. Common sense, combined with the fact that all the developers were in the same room, made sure that the configuration management worked well. As the product grew, and the number of developers doubled, problems did arise on occasion. One of the effects of the problems was work being deleted on a few occasions due to versions overwriting each other because of misunderstandings. When this showed to be causing problems for the developers, a quick and dirty solution was introduced. Using simple text files to administrate copies to checkout directories, the problem was solved.

Awareness of what was happening in the product was intended to be handled by the developers sitting in the same room and communicating all the time. The problem that became apparent with this strategy was

when people were absent or working different schedules.

Code is integrated continuously several times a day and several times for each task. The alternative of employing a tool for the configuration management might, in retrospect, have been a more effective solution. The basic system of copying files to checkout folders solves the basic issues addressed by these tools and, as no configuration branches were to be used at all, the simple solution worked once the communication problems were fixed.

EXPERIENCES OF THE XP PRACTICES

This section discusses the experiences gathered through the observations and interviews made at Online Telemarketing practice by practice. This strategy of structuring the experiences seemed at the time to provide the most complete account for describing the experiences gained. If the experiences were not structured by practice it was thought that experiences not thought vital for this group might not be included and thereby not available to other groups.

The developers were introduced to XP by a half-day seminar with an introduction and an *extreme hour* exercise [10]. The developers had guidance from the coach regarding how they should implement XP at all times. XP books [1, 11, 12] were also made available to them. The developers were also instructed to look at XP websites to keep up to date on recent developments in the XP community [10, 13, 14, 15].

The planning game

Using story cards proved to be one of the greatest successes of all the XP practices. The story cards provided all parties involved with a picture of the status of the work and an overview of the product as a whole. Approximately 150 stories have been implemented in total. The stories were written by the customer and then prioritised together with the development manager as he had the best overview of the technical status of the product. The estimation worked well once the management understood the three levels of prioritisation [1, 2, 3].

New stories were added continuously during the whole project. This was due to the fact that the management did not have a clear picture of the product at the start of the project. This meant that functionality was continuously added during the entire project. The time estimation of the stories was difficult at first due to the lack of practical experience of estimating, but after a few weeks the estimating worked very well according to the group members. The estimation quality was not confirmed using quantitative methods. The whole group performed estimations together during planning meetings.

Breaking the stories into tasks was difficult for the developers to grasp. The developers ended up drawing flow charts for the work, which was not the idea. Some of the story cards were very similar to tasks, i.e. at a too detailed a level for story cards. The problem was thought to be due to the difficulty of setting some kind of a common detail level for the stories.
The developers selected the stories to develop in conjunction with the development manager. This way of working with stories and tasks is an area that was continuously looked at and improved during the course of the project.

Small releases

Creating a minimal framework for each part of the system proved to take longer than the following smaller releases. The very first iteration took much more time than intended due to lack of experience in using the XP methods and traditional development thinking dominating. Once a complete bare working system was implemented, however, small releases were easier to implement.

During the long initial releases it was important to keep good communication between the customers and developers so that the project did not proceed in the wrong direction. As an afterthought, this practice seems fundamental to the success of XP. Maybe more effort should have been exerted to keep the initial release time shorter.

Metaphor

The system metaphor created before the actual start of the development was a little too detailed. It was almost an attempt at a complete requirements document. This was partly due to the fact that this document was written before the XP methodology was first thought of for the project. The document was not altered after XP was selected as the preferred development method. The metaphor document was also not properly updated as the system evolved during the course of the project. This is most probably also due to the too detailed level of the system metaphor. A common picture of the system was gained throughout the project by looking at the system directly and discussing individual cards. This common picture could have been improved by creating an accurate system metaphor.

Simple design

The development team has strived to implement the simplest possible solution at all times in accordance with this XP practice. A further evaluation of this practice was deemed to be difficult to perform in a reasonable amount of time.

The philosophy of always assuming simplicity was thought to have saved time in the cases where a much larger solution would otherwise have been implemented. Time was also believed to have been saved due to the fact that developers did not have to cope with a lot of unnecessarily complicated code.

Testing
Test-first programming was difficult to implement at first. Determining how to write tests for code proved difficult to master. The developers thought that the tests were hard to write and they were not used to thinking the test-first way. It was found difficult to see how many tests were enough to satisfy that the desired functionality would be implemented correctly.

The VB Unit test structure [12, 13] was used to create the automatic unit tests. VB Unit takes quite a long time to get used to and set up according to the developers. The unit tests that were written take less than one minute to run in total. The whole set of tests were run each time new code was integrated. During the course of the project the developers started to ignore writing tests first, especially when the project came under time pressure a few months in. The developers understood why tests are important but thought it involved too much work and did not see the short term benefits. It is believed that this was due to the inexperience of the developers. A more rigorous approach to the testing practices would most probably have been preferable.

The developers found programming by intention difficult. Programming by intention involves deciding the functionality and structure of the code in advance so that the test cases can be created beforehand. The development manager, who is experienced in coding these kinds of systems, found this way of working natural. He actually found that the way he usually worked was very close to the way described by XP. It was found that database code was much easier to write test code for than business rule code. The graphical user interface (GUI) code was also, as expected beforehand, hard to write automated tests for. Because of the limited nature of the GUI it was decided that an automated test tool for GUI testing would probably take longer to take into practice than manual user testing.

As the project came under pressure to release the fully operational version, the test first method of working ceased completely. The time pressure was due to the expansion of the company into a new region earlier than first expected. This meant that a portion of the new system was desired to go into operation earlier than the initial planning.
The functionality of the system was tested by the customer before each release as well as spontaneously during the development. When the functionality was not as the customer had intended it, a correction card was written. At first the customer just interrupted the developers when they found the functionality inconsistent with the desired functionality, but this was found to be too disruptive so a correction card strategy was adopted. The functional testing provided a good view of how the product is progressing.

Refactoring
No tools for refactoring were used in the project. All project members performed minor refactoring continuously. No major refactoring of the code was performed, but assessment of the code was performed continuously regarding the benefits of a major refactoring in case it was necessary. No education or training was given either beforehand or during the course of the project in refactoring methods or theory such as those presented by Fowler [16].

Pair programming
The developers used pair programming at all times. The only exceptions were when illness intervened or the developers had demanding schedules at the university. The developers adopted pair programming cautiously at first, but then gradually started to work naturally and effectively in the pairs.

The fact that the developers had no prior professional experience probably made the introduction of pair programming much easier than if they had been used to working in a traditional single-programmer manner.

When alone, the programmers often seemed to seize and get stuck when solving a problem. Also the tendency to carry on with a nonworking solution seemed more frequent. The developers found it easier to keep their concentration on the task at hand when working in pairs.

The development leader estimates that the pair programming produced the code faster than if the same programmers would be working separately. However the inexperience of the developers made them much slower than experienced professionals.
The pair programming worked excellently when introducing new people into the project. For the first part of the project the pairs were been fixed so that the developers could synchronise their schedules easily, but during the second phase of the project when 8 people were working full time on the project, the pairs were changed continuously. The original 4 developers also chose their own pair-programming buddy, but the second group were assigned into pairs by management.

Collective ownership
Collective ownership worked well in the project. This contributed to solving some minor irritation among the developers due to defects found in the code. When the programmers thought of defects as a group issue, rather than someone else's 'private' defect the irritation disappeared and a constructive atmosphere was created. The only problem observed in this practice was due to the configuration management or rather lack of effectiveness in the communication in the handling of the configuration management. The developers were on occasion afraid to change parts of the code due to the risk of loosing work if not in direct contact with the other pairs.

Continuous integration
Continuous integration proved to be natural in the development environment created for the project. As soon as code was finished it was integrated into the product. The ease with which this practice was implemented is notable in itself.

40-hour week
As the developers all worked part time, 20-hours per week, this practice was adjusted to accommodate this. Only the development manager and senior management worked full time.

On site customer
The customer was available throughout the course of the project. This worked very well. The only problems were the flexible work hours of both developers and management and everyone's busy schedules. While the senior management of the company had the role of customer, they were not been able to devote all of their available time to this project, because of other meetings and responsibilities in running the company. At the start of the project the customer had many opinions on the functionality in the product. As soon as a release was made the customer wanted to modify or add to it. This decreased during the course of the project, partly due to the system evolving into what the customer wanted and partly due to that the customer became better at writing story cards describing the desired functionality to the developers more efficiently.

Coding standards
A coding standard document was created at the start of the project. This was used extensively at first and added to when needed. After a while the developers became more relaxed and used the coding standard less. This was at the time identified as an issue and was re-enforced with success. The outcome of this practice has however not been evaluated by comparing sections of the actual code with the coding standard.

QUALITY OF CONCLUSIONS
In this section the criteria discussed in section 3, methodology, are discussed with regards to the research methodologies employed in this paper.

- *Credibility*
  The fact that both interviews and observations were used in the study increases credibility. The resulting observations do not seem to be incredible. The resulting observations seem to be correct when reviewed by the development manager at Online Telemarketing.

- *Transferability*
  The experiences from introducing XP in this project should be of considerable help to other projects introducing XP, either in part or fully. Consideration should be taken to the facts that the developers were working part time and were otherwise university students, not full time, experienced professionals.

- *Dependability*
  Due to the limited nature of the study in this experience report it is difficult to assess the dependability of the study.

- *Confirmability*
  The confirmability is increased by the review by the development manager at Online Telemarketing.

The quality of the conclusions is increased by the triangulation of qualitative research methods. Both interviews and direct observations were used and the results were reviewed by a representative of the participating subjects.

SUMMARY AND CONCLUSIONS
In conclusion, the project at Online Telemarketing was a success. The product was created and is now functioning live. The experiences of the actual XP practices are a mostly successes, but also a few failures. All of these experiences are relevant to projects considering introducing XP.

The *planning game* was easy to introduce and effective. This can be partly due to the fact that the extreme hour, used to initially introduce XP, focuses on the *planning game*, as does extensive parts of the XP literature [e.g. 1, 11, 12].

Small releases proved difficult for the first releases for each part of the system. Even though they were expected to take a little longer than the other releases, they took longer than planned. An increased focus on only creating an absolute minimal framework system might help this.

The system *metaphor* was too complicated to start with. This resulted in a metaphor document that did not evolve with the system. It should not be difficult to keep the metaphor up to date if it is simple from the start.

Simple design was thought to work well, but was not verified by code inspections. The developers believed that by thinking in terms of simple solutions as much as possible, they saved a lot of time by not having to try to understand unnecessarily complicated code.

Testing was found to be one of the hardest practices to implement. It requires careful preparation of the testing unit and also a strict discipline among the testers to always write the tests first. The *testing* practice was the first practice to cease when the project came under pressure.

Refactoring was performed on a small scale all the time. This is, however, natural in normal programming. Larger scale refactoring was not performed, although the possibility of large scale refactoring was continuously evaluated.

Pair programming worked excellently for the developers in the project. It seemed to help them solve difficult problems faster and identify potential dead-end solutions earlier. The *pair programming* also worked very well when introducing new people into the project

Although it was different from what the developers were used to from the start, *collective ownership* proved to be effective for the team spirit.

Continuous integration was not hard to implement and was found a natural way to work in the development environment created in the project.

The *on-site customer* practice worked well. The customer solved many misunderstandings of functionality early and was available to complete or clarify any poorly written story cards. As the customer did not really know the full extent of the product at the start of the project, this practice appears to be one of the major reasons for the success of the project.

The *coding standard* practice worked well. When the developers started to get sloppy in the middle of the project, the development manager enforced the coding standard again.

Keeping in mind the issues raised in the quality of conclusions section, section 7, these experiences should be of interest to any development team considering introducing XP.

ACKNOWLEDGEMENTS

This work was partly funded by The Swedish Agency for Innovation Systems (VINNOVA), under a grant for the Center for Applied Software Research at Lund University (LUCAS). The author would also like to thank Johan Norrman (Online Telemarketing) and Per Runeson (LTH) for their contributions to the paper.

REFERENCES

[1] Beck, K., *Extreme Programming Explained: Embrace Change,* Addison Wesley, 1999.

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

[3] Martin, R., C., "Extreme Programming Development through Dialog", *IEEE Software*, July/August 2000, pp.12-13.

[4] Haungs, J., "Pair Programming on the C3 Project", *IEEE Computer*, February 2001, pp. 118-119.

[5] Hicks, M., *XP Pros Take it to the Extreme,* ZDNet eWeek News, last confirmed 010903, http://www.zdnet.com/eweek/stories/general/0,11011,2714342,00.html .

[6] Lincoln, Y., S., Guba, E., G., *Naturalistic Inquiry,* Sage Publications, 1985.

[7] Robson,C., *Real World Research,* Blackwell Publishers, Oxford, 1993.

[8] Miles, M.B., Huberman, A.M., *Qualitative Data Analysis*, Sage Publications, 1994.

[9] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A., *Introduction to Experimentation in Software Engineering*, Kluwer Academic Publishers, 2000.

[10] Multiple Authors, *The Extreme Programming Roadmap,* last confirmed 010903, http://www.c2.com/cgi/wiki?ExtremeProgrammingRoadmap .

[11] Beck, B., Fowler, M., *Planning Extreme programming,* Addison Wesley, 2000.

[12] Jeffries, R., Anderson, A., Hendrickson, C., *Extreme Programming Installed,* Addison Wesley, 2000.

[13] Jeffries, R., (Ed.), *XProgramming.com,* last confirmed 010903, http://www.xprogramming.com .

[14] Wells, D., *Extreme Programming: A Gentle Introduction,* last confirmed 010903, http://www.extremeprogramming.org/ .

[15] Multiple authors, *Extrem Programmering,* (in Swedish), last confirmed 010903, http://oops.se/cgi-bin/wiki?ExtremProgrammering .

[16] Fowler, M., *Refactoring: Improving the Design of Existing Code,* Addison Wesley, 2000.