

Introducing Extreme Programming into a Software Project at the Israeli Air Force

Yael Dubinsky¹, Orit Hazzan² and Arie Keren³

¹Department of Computer Science
Technion – Israel Institute of Technology
Haifa 32000, Israel
yael@cs.technion.ac.il

²Department of Education in Technology & Science
Technion – Israel Institute of Technology
Haifa 32000, Israel
oritha@techunix.technion.ac.il

³MAMDAS – Software Development Unit
Air Force, IDF, Israel

Abstract. Introducing Extreme Programming (XP) to an industrial software development team usually involves technical and professional aspects as well as social and organizational ones. The introducing of a new software development method in general and XP in particular into a software project team that operates in the army emphasizes and extends these issues. In this paper we present a process through which XP has been introduced into a 60-members software development project at the Israeli Air Force. Focus is placed on an XP workshop conducted with ten officers who worked at different teams of the project. Specifically, we present the principles according to which we facilitated the workshop, the workshop agenda and data regarding the way the participants perceive some of the XP practices. Recently, the first XP team in this project has started to work the XP way.

1 Introduction

Awareness to agile software development methods increases in the last few years. More companies seek for innovations and solutions that may answer their special needs, and find the agile methods in general ([3],[5]) and Extreme Programming (XP) ([1]) in particular as a possible answer. Experience gathered in the community indicates that the introduction of XP into an organization goes along with conceptual and organizational changes that are integrated part of the process.

In this paper we present the process of introducing XP into a software project at the Israeli Air Force. In Section 2 we describe the project setting and the preliminary phase that resulted in an XP workshop. In Section 3 we focus on the XP workshop activities. In Section 4 we bring data and its analysis regarding the way the workshop participants conceived some of the XP practices. We summarize in Section 5.

2 The Air Force Software Project

The software project we deal with is being developed by MAMDAS which is a software development unit in the Israeli Air Force. The project is being developed by a team of 60 skilled developers and testers organized in a hierarchical scheme of small groups. The software is intended to be used by thousands of users.

The third author, who is in charge of the system engineering group of this project, was requested to lead a change in the current development process to implement a new process that would enable rapid response to customers' requests and requirement changes, and would obtain feedback with respect to released features. This sub-project is named as Short-Cycles and its main goal is to change the method of work in the project itself. The duration of Short-Cycles was set to one year in which a new methodology has to be suggested and an initial team should start to implement. It was clear that an organizational and conceptual change should take place. Since the team was relatively large, such a change could not be performed over night, but rather in a gradual, stage-based process which was to be planned accordingly.

Here are three characteristics of the software project with which Short-Cycles had to deal. First, an explicit role for each project employee could not be defined nor its performance enforced; thus, it was not clear who was in charge of what. Second, work habits are so well rooted in the software development unit that even if a change were to be implemented forcefully, the accepted development process might well eventually "return through the back door". Third, communication channels between teammates and customer and among teammates were cumbersome and could not be forced; for example, in some cases no users' feedback was received and in other cases when feedback was finally received it often turned to be irrelevant.

These characteristics further explain the need for a process that would gradually guide the transition and assimilation of the new software development method. This process was also to take into account the individual interests of different people, the objections each sub-team was expected to raise, and the harmony and synergy between the different changes that were to take place in each specific defined process.

It is important to note that the Air Force leadership supported the Short-Cycles phase. Furthermore, the leadership specifically declared that, while a reduction in quantity might be accepted, quality and fitness to customers' needs were not to be compromised. It was realized that resistance may be raised by the mid-level officers.

In preparation for this transition, a Short-Cycles Group of 10 volunteers was established. Its aim was to formulate and lead the process that would end with the entire team working according to the new desired development process. The Short-Cycles Group was composed of members representing all levels of the 60-people team. The group met every two weeks.

One of the decisions of the Short-Cycles Group was to learn about the agile approach and specifically the XP method. Part of this decision was to conduct an XP workshop with project members who will be able later to evaluate and decide upon the sequel of Short-Cycles.

3 The XP Workshop

In this section we describe the XP workshop that was conducted as part of Short-Cycles. The workshop was facilitated by the two first authors and its orientation was based on our belief that Extreme Programming (XP) is best understood when a team experiences and works according to the XP values and practices in real-life software development situations.

In the 2-day workshop we aim at introducing XP while enabling the participants to experience, as far as possible, the actual construction of a software system. The idea is to show the participants just how much can be achieved in two days if the plan and time management are conducted properly, all while paying proper attention to the customer's needs. In order to achieve this goal within the time limits of a 2-day workshop, the participants complete during the workshop the development of the first iteration of a software system that they choose to develop. In addition, reflections are conducted, and a discussion is held on the suitability of XP to the organization.

Following are the principles that were applied in the 2-day workshop.

Principle 1: Experience the different XP practices as much as possible. Since we believe that real experience is needed in order to capture the main ideas of XP, hands-on experience by the participants is integrated, as much as possible.

Principle 2: Elicit reflection on experience. The importance of introducing reflective processes into software development processes in general and into the XP development environment in particular, has been already discussed ([4]). During the workshop, we aim to elicit such processes in several ways, such as asking the participants to recall and reflect on situations taken from their own experience in software development, and presenting questions in which the participants are asked to compare different situations in software development.

Principle 3: Relate to participants' feelings towards the presented software development environment. The adoption of XP requires a conceptual change with respect to what a software development process is. Consequently, it is well known that XP raises emotions that, in some cases, are in fact objections. When the audience expresses emotional statements against XP, we take advantage of this opportunity and encourage participants to express their feelings towards the subject, open it to discussion, and explain how XP addresses the issue just raised.

Principle 4: Use narratives. As an alternative to a technical explanation of the XP practices, we have adopted the storytelling approach, which has become more and more popular in management process¹. Following this spirit, we present the XP practices (in various levels of detail) using a story.

The story describes two days in the life of a software developer working in an XP environment: one business day and one development day. The appropriate XP practices are used to describe these two kinds of days. In addition to the story itself, we invite the audience to envision themselves in the described environment and to continuously compare it with their current software development environment.

Principle 5: Stick to timetable unless an interesting topic comes up. During the workshop we try to adhere to the timetable exactly as planned (see Tables 1, 2). The

¹ See at <http://www.creatingthe21stcentury.org/>

idea underlying this guideline is to illustrate how much can be achieved in a relatively short period of time. In addition, on such a tight time schedule, without too many long breaks, participants feel that their time is valued. Naturally, there are cases in which we decide not to follow this principle. This happens in cases in which we feel that an interesting topic has come up and by focusing on it for a while we may improve the workshop in a way that was not pre-planned.

Principle 6: Our preferred group size is 6-12 participants. The upper limit (12) is obvious. We require the lower limit (6) because we believe that this number of participants affords a feeling of how XP works for a team. In addition, we require that all participants commit to attend the full two days. If a participant is forced to leave the workshop for some unexpected reason, we take the opportunity to discuss the analogy of such a case to real life situations.

Principle 7: Sustainable pace (8 hours every day). Similar to the XP practice of Sustainable Pace, we make sure that the development performed during the workshop is conducted at a reasonable pace. The fact that the first iteration is developed within two such days serves to reinforce the participants' feelings about the potential contribution of XP to software development processes.

Principle 8: Workshop site. We request that the host company equips the room in which the workshop is to take place with a large table for the planning game, computers, flipcharts or a whiteboard, and a well-stocked coffee corner.

Tables 1 and 2 present schedules for a 2-day XP workshop. In order to illustrate how the days look in practice, we added hour notations to the workshop plan. The schedule presented here is a general schedule of the workshop. The details of each activity are beyond the scope of this paper.

As can be observed from Table 1, the main role in the first day is that of the customer, and the message that we try to convey in this day is that the customer is an integral part of the development environment.

Table 1. First day of the 2-day XP Workshop

10:00-10:30	30 minutes	Introduction
The 2-day story is presented briefly and references are made to what the participants can expect to experience during the next two days.		
10:30 – 11:00	30 minutes	Selection of a topic
We let the participants decide on the software they are going to develop in the workshop. This is done by asking them to suggest ideas and then taking a vote on the preferred topic.		
11:00 – 13:00	2 hours	Planning Game & Role Scheme
<ul style="list-style-type: none"> ▪ Planning game. All stages of the Planning Game are performed: telling of customer stories, allocation of releases and iterations, simple design of the first iteration, break-down into development tasks, allocation to developers, time estimation, and load balance. ▪ Role scheme. We build a role scheme together with the team, launch the created scheme, and refer to special roles during the activities. 		
13:00 – 13:30	30 minutes	Using Metaphors

We present the essence of metaphors and delve into the details of using them during the development process in general and during the planning game in particular. We also use metaphors when discussions regarding problems arose.		
13:30 – 14:00	30 minutes	Lunch break
14:00 – 15:00	1 hour	Planning Game (Cont.)
The part of task allocation, time estimations and load balancing is performed.		
15:00 – 17:15	2 hours and 15 minutes	Start of Development
XP practices (technical and organizational) are presented during development.		
17:15–18:00	45 minutes	Reflection & Summary
<ul style="list-style-type: none"> ▪ We end the day with reflection and retrospection on the first day. ▪ In preparation for the second day of the workshop, participants receive thinking homework. 		

Table 2 presents the schedule for the second day of the XP workshop, together with an explanation of the main activities conducted during each time slot. All roles are active in the second day so to bring the first iteration towards completion.

Table 2. Second day of a 2-day XP Workshop

10:00 – 10:15	15 minutes	Stand up meeting
Stand-up meeting.		
10:15 – 10:45	30 minutes	TDD
Presenting test-driven development (TDD).		
10:45 – 13:30	2 hours and 45 minutes	Software Development
<ul style="list-style-type: none"> ▪ Software development. ▪ XP practices (technical and organizational) are presented during development. ▪ Roles scheme is performed. 		
13:30 – 14:00	30 minutes	Lunch break
14:00 – 16:00	2 hours	Software Development
<ul style="list-style-type: none"> ▪ Stand up meeting. ▪ Completion of development of iteration 1. 		
16:00 – 16:30	30 minutes	Presentation
<ul style="list-style-type: none"> ▪ Presentation of the first iteration. ▪ Roles summaries. ▪ Feedback activity. 		
16:30 – 18:00	1.5 hours	Reflection & Summary

- Reflection activities.
- **Implementation of XP in the participants' organization.** This last part of the workshop is dedicated to a discussion in which the participants present their thoughts on the implementation of XP in their organization in general, and in their team in particular.
- Summary.

4 Data and Analysis

This section presents an analysis of data that was gathered during the workshop mainly by using anonymous written reflections filled in by the participants.

4.1 The Planning Game and the Roles Scheme

The first feedback was received from the participants after the first planning game had been conducted. This was the first time they participated in such an activity, yet they were not familiar with most of the XP practices. Participants were asked to reflect on the contribution of the planning game to their understanding of the project and to describe the advantages as well as the disadvantages of this practice. Nine participants (out of ten) have reported that the planning game helped them in better understanding the project's essence and its requirements, among them the officer who played the role of the customer. Only one participant reported that the planning game did not contribute at all.

Following is a sample of participants' expressions when describing the *advantages* of their first planning game: "Sharing information among teammates. Everyone knows what happens. Understanding of requirements by all teammates, acquaintance with all factors involved."; "...enables distribution to sub tasks."; "The process was quick and enabled making many decisions and tasks in short time."; "The customer was available and understand our constraints."; "We see everything in front of our eyes."; "Collaboration and better acquaintance with the people we work with."

Following are participants' expressions when describing the *disadvantages* of their first planning game: "More discussions, which were not necessarily directed to solve problems."; "Many people involved, lack of focus."; "Tendency to diversion. The customer is in pressure and doesn't remember everything."; "We didn't check duplications of subjects."; "We designed top-down and didn't verify using bottom-up."; "Sometimes, over talking causes lack of interest and concentration."; "Someone should manage the planning game so it won't be scattered."; "I didn't figure out how this will work. The planning seems very optimistic."

At this stage, before actually starting to develop the XP way and learning more about the XP practices, the above reflections can be considered premature. However, they do indicate the level of involvement of the participants and how they grasp most of the planning game benefits while still being suspicious and judgmental.

During the same reflection the participants were asked to describe their personal role in the team and how the planning game can help them perform their role. The

roles scheme we use in such workshops, described in details in [2], consists on the XP roles and adds roles that help in the management of a software project. In practice, the scheme implies that all teammates are developers while everyone has in addition a special own role. Table 3 presents participants' feedbacks according to their role.

Table 3. Planning game reflections according to roles

Personal Role	How can the planning game help you with performing your role?
coach	“ – Tasks distribution. – Accountability of all teammates. – Statements regarding times. – Work according to a method. – Coordination of expectations.”
tracker	“Getting acquainted with all tasks and time estimations to be used for control.”
customer	“Receiving a real situation regarding my requirements – what is possible and when.”
in charge of acceptance tests	“The planning game will help me realize the size and scope of the tests.”
in charge of continuous integration	“Keeping communication with the customer. Presenting results to the customer, sharing development constraints with the customer...”
in charge of design	“Enables figure exactly the structure of the system and influence it since all tasks can be easily observed.”
in charge of infrastructure	“The planning game can help me plan which work and development environment should arranged for the project. Which tools will be used...”
in charge of unit tests	“Getting acquainted with all tasks and understanding functionality before implementation.”
in charge of presentations and documentation	“Understanding of the processes in the project.”
in charge of code control	“Can help in thinking of possible ways to code...”

4.2 The Test-First Practice

Participants filled in a written reflection after their first experience with the test-first practice. Eight participants (out of nine) were in favor of the practice saying it is good, effective and ensures quality. One had no specific position. Four participants (out of nine) reported positive feelings like pleasure, innovation, and comfort, when experienced test-first. Four participants reported negative feelings like frustration, uncertainty, and time overhead. One participant didn't indicate any special feeling.

Participants were asked for the most significant *disadvantage* of the test-first practice. Five participants (out of ten) reported that the most significant disadvantage

is that test-first is time consuming; three participants claimed it can lead to the introducing of later changes in the tests; one participant claimed it lacks a global perspective; one claimed it is annoying.

Participants were asked for the most significant *advantage* of the test-first practice. Three participants (out of ten) reported that the most significant advantage is ‘thinking before coding’. Nine participants (out of ten) claimed test-first ensures the existence of tests and that it enables knowing what the minimal code that should be written is.

We have observed two main phenomena when examining the written reflections of the workshop participants and analyzing our observations. The first one is the participants’ reference to test-first as a thinking activity in general and a ‘thinking before coding’ activity in particular. This observation indicates that coding is not conceived by the workshop participants as a developer-computer simple interaction, similar to people interaction, but rather, that it requires thinking before performing even when we know what we want to say, i.e. start developing right away. This phenomenon can be explained by the common way developers refer to the coding. Specifically, developers usually start coding in an intuitive way, and therefore see test-first as a practice that enforces them to think prior to the coding.

The second observation is the contradictions and conflicts that the practice of test-first brings with. One participant claims that test-first ensures fewer bugs and consequently leads to shorter integration times. Still, this participant indicates as a disadvantage of test-first that it leads to time overhead. Another participant claims that since we think before coding, we know exactly what we are going to code. At the same time this participant indicates uncertainty as a feeling he feels while practicing the test-first approach. A third participant claims that test-first cuts off the continuity of coding while, at the same time acknowledges the convenience in using the practice. This phenomenon can be explained by using another contradiction inherent in the test-first itself. Developers are used to code and then test while test-first enforces them to perform these activities in the reversed order. Accordingly, experiencing test-first for the first time can cause mixed feelings and contradicting opinions.

4.3 The Pair Programming Practice

Participants filled in a reflection after they had developed in pairs addressing the following open question "Did you enjoy pair programming?". Eight (out of nine) reported that they enjoy pair programming. One reported "so-so". We asked which trait of your pair is the most appreciated by you. Two (out of eight) appreciate professionalism; two appreciate knowledge and acquaintance with the development environment; and the other four answers were: ability to learn, pedantry, doesn't get into panic and doesn't make pressure, agility.

4.4 Final Workshop Feedback

The final workshop feedback was received using a closed questionnaire in which the participants were asked to mark their agreement level with seven statements in a 1-5

scale (1 - weak agreement, 3 - reasonable agreement and 5 - strong agreement). Table 4 summarizes this feedback.

Table 4. Final workshop feedback of nine participants

Statement	1 slightly agree	2	3 reasonable agree	4	5 greatly agree
The workshop answers my expectations			1	4	4
XP suits me			2	6	1
XP suits my team	3	1	4	1	
XP suits my project	1	3	5		
XP suits my organization	2	3	3	1	
I'll act to assimilate XP in my team	1		5	2	1
If there will be a trial to assimilate XP in my project, I expect resistance	1		3	2	3

Table 4 shows that although participants think that XP suits them and some of them will act to assimilate it in their team, they still feel that XP suits less to their team, project and organization, and expect objections if XP is assimilated. This feedback can be explained by the fact that, indeed, in order to assimilate XP in the project, organizational and conceptual changes should occur.

5 Summary

This paper presents the introducing of XP into a software project at the Israeli Air Force. After conducting the workshop, a methodological specific process to implement XP into the unit was designed.

Recently, as a result of this process, one XP team has started to work according to XP on a new adjacent project which leans on the same infrastructure. Measures were defined and used. A new research has been established in order to assess the XP assimilation process in general and the team performance and progress in particular.

References

1. Beck, K., Extreme Programming Explained: Embrace Change, Addison-Wesley (2000).
2. Dubinsky Y. and Hazzan O., Roles in agile software development teams, 5th Int. Conf. on eXtreme Programming and Agile Processes in Software Engineering (2004) 157-165.
3. Highsmith, J., Agile Software developments Ecosystems, Addison-Wesley (2002).
4. Hazzan, O. and Tomayko, J., The reflective practitioner perspective in eXtreme Programming, Proceedings of the XP Agile Universe (2003) 51-61.
5. Reifer, D. J., How to get the most out of Extreme Programming/Agile Methods., Proceedings of the 2nd XP Universe and the first Agile Universe Conference, (2002) 185-196.