An XP Experiment with Students – Setup and Problems

Thomas Flohr and Thorsten Schneider

Software Engineering Group, University of Hannover,
Welfengarten 1, 30167 Hannover, Germany
{Thomas.Flohr, Thorsten.Schneider}@Inf.Uni-Hannover.de

Abstract. Designing experiments to be carried out with students as subjects in an XP setup is a difficult task: Students lack experiences with XP, there are limited resources, the experiment might not be taken seriously and other effects interfere. This paper presents an experiment using student subjects examining test-first in comparison to classical-testing. We proved several hypotheses about test coverage, number of test-cases, contacts with customer, acceptance for test-first, development speed and not required features. While designing the experiment we noticed that it is useful to include some additional XP techniques on top of test first, because of our special setup and the demands we had. Despite careful planning and conduction of the experiment we still faced a number of problems. In this paper we also discuss the problems with our experimental setup.

1 Introduction

Extreme Programming (XP) by Kent Beck [1] is the best-known agile Method. XP is a collection of several practices to be obeyed during software development projects.

The effectiveness of most XP practices is only weakly confirmed by empirical data gained through experiments. A single XP technique can either be examined in isolation, in a complete XP setup, or in a mixed setup in which some other XP techniques are applied as well. In winter semester of 2004/05 the Software Engineering Group of the University of Hannover conducted an experiment and for practical reasons (e.g. limited hardware resources and realistic setup) we decided to choose the last alternative, in which some other XP techniques are applied at the same time. Objects of investigation were test-first and classical-testing. For the automatic tests we used the JUnit framework.

We designed two experimental setups: a test-first setup including test-first and some other XP techniques (pair programming, on-site customer and user stories) and a classical-testing setup including the same XP techniques, but test-first was replaced by a classical-testing approach.

– Students should implement a useful program (e.g. not a small and abstract programing task) because students do not take synthetic tasks seriously.

F. Bomarius and S. Komi-Sirviö (Eds.): PROFES 2005, LNCS 3547, pp. 474–486, 2005.

- As in real life the requirements on the program are sometimes vague.
- The experiment should take place at the same location and time for all participants to create equal conditions for everyone and also because of our limited resources.
- Implementation during the experiment is restricted to our computer lab. Homework is not allowed in order to maintain comparability and control.

To meet these criteria we carefully designed a setup. Nevertheless we faced problems while designing and conducting the experiment. The focus of this paper lies on a description of the experiment's setup, the problems we faced and how we avoided some of them. This paper does not include the results of the experiment, because the experimental run is not completely finished.

1.1 Related Work

So far most techniques of XP were studied in isolation in controlled experiments. Case studies often observed more complex XP settings with multiple XP techniques.

Müller and Hagner observed 19 graduate students with some experience in XP [2]. In their experiment test-first was compared to traditional development separately. The students were divided in two groups to apply one of the approaches each. Each group had to implement the main class of a graph library which only contains declarations of methods. The experiment included unit and acceptance tests. Objects of investigation was programming speed, reliability of the final code and program understanding (reuse of code). The authors conclude that test-first for a traditional developer is not necessarily faster than traditional development.

In a further case study Müller and Tichy report about experiences, when realizing an XP course with students [3]. The study focused on pair programming, iteration planning, test-first, refactoring and the question of the best size of a team for an XP-project. One result was that pair programming is no problem for students, but test-first is hard to learn in the beginning.

Noll and Atkinson [4] report about a case study dealing with a comparison between XP and traditional development at a university. A class was divided into 4 teams; each consisted of 6 to 8 students and each team had to develop a web-based room reservation system. Two teams followed XP and two teams followed the traditional approach. One result was that the teams following the traditional approach delivered less robust code, but their solutions had more features. The test-first teams delivered more robust code with less features, but the user interface was sloppy and difficult to use. It was also observed that requirements were interpreted by the teams to their favor, despite an always available customer. The authors advise to prefer a strong customer (e.g. professor) and to outline the differences between XP and traditional development better.

Lindvall et al. [5] report about an eWorkshop about different empirical findings in agile Methods. Objects under discussion were team size, personnel, reliability, training requirements, refactoring and documentation in agile projects. The authors conclude that 10% of qualified personnel in an agile project with pair programming