Teaching Software Process Improvement trough Extreme Programming practices, study case: UAA

Arturo Barajas Saavedra, Francisco J. Álvarez Rodríguez, Jaime Muñoz Arteaga¹, Jiani Cardoso²

¹Universidad Autónoma de Aguascalientes, Avenida Universidad 940, Colonia Ciudad Universitaria, Aguascalientes, Ags., 20100. México
²Pontifícia Universidade Católica do Rio Grande do Sul, BR 472 Km. 3, Uruguaiana, RS
ar.ba.saa@gmail.com, {fjalvar, jmunozar}@correo.uaa.mx, jiani@pucrs.br

Abstract. This paper describes the practical results of an experiment created in order to analyze the software process improvement by the use of the practices and rules of Extreme Programming methodology. In the experiment a group of 40 students in the Universidad Autónoma de Aguascalientes took part to create a small system, to measure their productivity and to measure how much does this rules and practices are applicable in the real world.

1. Introduction

Extreme Programming (XP) is actually a deliberate and disciplined approach to software development. XP empowers your developers to confidently respond to changing customer requirements, even late in the life cycle. XP improves a software project in four essential ways; communication, simplicity, feedback, and courage. XP is a collection of the “best practices” in software development called rules and practices each of which supports others, and are supported by several others in turn. [1], [2].

2. Problematic

When using a heavyweight methodology everything becomes administrative work that was designed to be a historical record rather than a tool to assist in problem solving and project management. [4]

In order to improve software processes is necessary to reduce their time and costs, and to increase the quality and the scope of their products and of the processes themselves. Through the use of a lightweight methodology for developing a software product, we can focus on the main and most important activities like planning, designing, coding, and testing according to the pending activities and the current status of the software process, instead of doing it weeks or months before executing the activities.

3. Methodology

In this section we will explain the activity executed in order to get information and data about the performance of 20 pairs of programmers working according to the rules and practices of the Extreme Programming methodology.
3.1. Used methodology

The experiment was divided into two parts, the first one is the delivery of the user stories, and the second part is the simulation of a change in the requirements. The first delivery of the user stories were provided to the students and it described the necessity of a computing system capable of calculate different measures (perimeter, area and volume) of six different geometric figures. In the second iteration there was a change in two of the six figures, which means a change in the 33.33% of the original requirements.

The products that the pairs must provide or deliver to us were: CRC cards, Iteration planning, Release planning, Source code and Executable code.

3.2. Experiment objectives

The first objective is to prove that the communication in the work team is increased and it is much more effective [4]. The second objective is to prove that the productivity is increased due to the pair programming, as we can see from the research in [3]. The third objective is to prove that the time is decreased. The forth objective is to prove that the number of LOC, as well size, effort and time is much more predictable with this methodology.

4. Experimental Results

As we can see in the figure 1, the mean time spent in the Coding area of the software system is significantly longer than in the other three areas in the first iteration. In the second iteration is almost the same mean time for the four areas, this is because the programmers just had to create two new modules instead of six. Also, in the Testing area a longer time was spent, due to the practices and rules that focus on testing all the modules and codes in order to be released with no bugs.

![Figure 1. Spent time per area chart](image)

According to the structure of the rules and practices of XP, this is the expected behavior of the software process, because this methodology emphasizes in the Coding and Testing areas, leaving in a second plane the rigorous documentation and planning.

The figure 2 shows how the time is distributed in the four areas of the software development process.

The figure 3 shows the mean time spent in the construction of each module. As we can see this is a very predictable time due to its small standard deviation.
6. Conclusions

In accordance with results derived from our experience based on the study of 20 pairs of programmers working with the rules and practices of the Extreme Programming methodology and from the pupils’ experience of working with a lightweight methodology like XP, we proposed the use of this methodology in small and medium size software processes.

The experiment objectives were achieved, so we based our proposal in them, because these achieved objectives proves that the Extreme Programming methodology improves the software process as we mention in the section 3.

References


