# Productivity of Test Driven Development: A Controlled Experiment with Professionals

Gerardo Canfora[1], Aniello Cimitile[1], Felix Garcia[2], Mario Piattini[2], and Corrado Aaron Visaggio[1]

[1] RCOST- Research Centre on Software Technology
University of Sannio, Italy
{canfora, cimitile, visaggio}@unisannio.it
[2] ALARCOS Research Group- Information Systems and Technologies Department
UCLM-Soluziona Research and Development Institute
University of Castilla-La Manch Paseo de la Universidad, 4 – 13071 Ciudad Real, Spain
{Felix.Garcia, Mario.Piattini}@uclm.es

**Abstract.** With the growing interest for Extreme Programming, test driven development (TDD) has been increasingly investigated, and several experiments have been executed with the aim of understanding if and when it is preferable to the traditional practice of testing the code after having written it (named TAC in the paper). However, the research concerning TDD is at its beginning and the body of knowledge is largely immature. This paper discusses an experiment carried out within a Spanish software company with the aim of comparing productivity in TDD and TAC.

## 1 Introduction

Test driven development (TDD) belongs to the set of the extreme programming [1] practices, even if it might be adopted in any kind of software development process. According to TDD, unit testing results drive code development. As a first step, the developer defines the classes of the system in terms of public interfaces. Then, the test suite for each class is written: the suite must contain all the tests helpful for verifying that each method in the class exposes the correct behavior. Finally, the body of each method is completed throughout an iterative process, consisting of two activities: to execute the tests and, when some of them fail, to change the code in order to remove the bugs that potentially caused the failure. The process ends when all the tests succeed. TDD is widely considered a practice of code development rather than code testing, although the role of unit testing is relevant for defining the design strategy to adopt. TDD might be considered also an alternative to the traditional approach of testing the code after having written it, named 'test after coding' (TAC) in this paper. Recently some researchers investigated TDD: some experiments [2], [4], [7], and [9] produced evidence about the improvement of code quality achieved with TDD; however, some authors [2], and [6] did not find particular differences between TDD and TAC. There is not a wide consensus about the relationship between TDD and productivity, despite several studies obtained evidence that TDD is able to increase the productivity with respect to TAC [3], and [4]. Since both the practices involve

J. Münch and M. Vierimaa (Eds.): PROFES 2006, LNCS 4034, pp. 383 – 388, 2006.

coding and testing in a tightly interleaved process, we aim at understanding differences in the productivity in the two practices. We have carried out an experiment with the collaboration of professionals working in a Spanish Software Company, aiming at meeting the following research goal: **Analyse** *Test Driven Development and Test After Coding* **With the purpose of** *comparing them* **With respect to** *productivity* **From the point of view of** *the developers* **In the context of** *a group of professionals.* The work comprises two research questions:

R.1 *Is test driven development more productive than TAC from the viewpoint of testing?* In this case, productivity is intended as the time needed to write and execute assertions.

R.2 *How is the time employed in the two practices?* In order to have a deep insight of the two practices, we try to understand if the increasing of productivity means reducing the time for developing or for coding.

The paper proceeds as follows: section 2 describes the experimental design; data are analyzed in section 3; and, finally, section 4 draws the conclusions.

## 2 The Experiment Characterization

The experiment aimed at testing the following null hypotheses:

**$H_{01}$ :** there is no difference in the productivity between test driven development and test after coding (helps to answer the research question R.1).

**$H_{02}$ :** there is no correlation between productivity and the number of assertions in test driven development (helps to answer the research question R.2).

**$H_{03}$ :** there is no correlation between the productivity and the number of assertions in test after coding (helps to answer the research question R.2).

The experiment was carried out in the facilities of the company Soluziona Software Factory located in Ciudad Real (Spain). The variables used in the data analysis are described in Table 1.

**Table 1.** Variables used in the experiment

| Variable | Description |
|---|---|
| **MeanTPA** | *Mean Time per Assertion.* It is the time required to write and execute an assertion in the test suite. In both the practices the time for executing the assertion includes also the changing in the code, suggested by the failures of the test. It is assumed as an indicator of the *productivity.* |
| **AssertTot** | *Total Number of Assertions*. It is the total amount of assertions in the project. It is an indicator of the *quality of test* in the overall project. The greater is the number of assertions the greater is the number of aspects of the code which are tested. |
| **MeanAPM** | *Mean Assertion per Method*. It is the mean number of assertions written for a method. It is an indicator of the *accuracy of testing*. A high value of this metrics could indicate that all the methods of the classes received the same attention in the test. |