

ANALYSIS OF THE IMPACT OF TEST BASED DEVELOPMENT TECHNIQUES TO THE SOFTWARE LIFE CYCLE

Experimental Planning

January 2018

1. Responsible

Msc. Antonio Quiña

Universidad Técnica del Norte (UTN) - Ecuador

Email: aquina@utn.edu.ec

Ing. Luis Alberto Cisneros

Polytechnic of Leiria - Portugal

Email: 2160085@my.ipleiria.pt

2. Context

Software development presents many challenges to work software development teams, ranging from understanding the needs of customers, until providing quality solutions that guarantee the users' expectations. In response to this and within Agile Software Development, several techniques are presented like: Test Driven Development (TDD), Acceptance Test Driven Development (ATDD) and Behavior Driven Development (BDD); which aim to improve quality, productivity and simplicity of design [1], [2]. This allows the programmers to focus on the development of the functionalities and at the same time reducing the time spent in other tasks like: code maintenance, bugs fixing, among others.

3. Objectives

3.1 General Objective

Analyze the impact on software quality and developer productivity, produced by applying test-based techniques in software development.

3.2 Specific Objectives

To achieve the general objective, the following specific objectives are proposed:

- Teach a group of computer systems engineering students about:
 - Testing;
 - junit;
 - Incremental Test-Last (ITL);
 - Test-Driven Development (TDD);
 - Behavior-Driven Development (BDD).
- Present one workshop about development and execution of **code katas**¹ with the trained students, to put into practice the learned techniques.
- Tabulate the obtained results from the workshop.

¹ A code kata is an exercise in programming which helps a programmer hone their skills through practice and repetition [7].

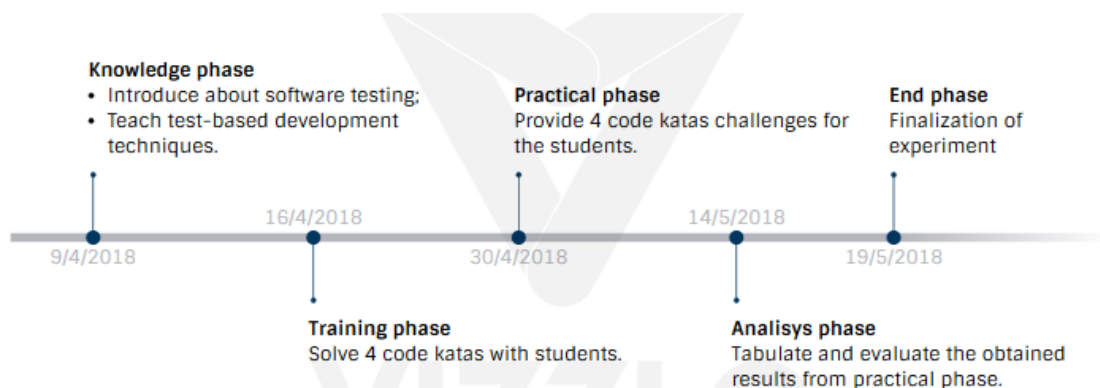
- Analyze and evaluate the results, in order to show the incidence in the quality of the software developed and productivity with the mentioned techniques.

4. Methodology

Considering previous TDD experimentation put into effect by O. Dieste and N. Juristo [3], [4], we propose and identify the following steps:

- Analyze and choose **4 code katas** as for the training phase as the practice phase. These programming exercises are found on sites like:
 - <http://www.codekatas.org>: videos in different programming languages where different katas are solved and explained.
 - <http://codekata.com>: Dave Thomas's site, which contains 21 exercises.
 - <http://codingdojo.org/KataCatalogue/>: catalog of many katas with English descriptions.
 - <https://github.com/garora/TDD-Katas>: katas specifically focused on practicing TDD.
- Apply the **metric of function point** for each **code kata**. This metric allows the evaluation of the functionality of a software at any stage of life cycle [5], [6].
- Define development times for the **code katas** proposed for the practical phase.
- Select a group of university students (**around 20**) and evaluate their knowledge through a survey.
- Introduce knowledge about: Testing, JUnit, Incremental Test-Last (ITL), Test-Driven Development (TDD) and Behavior-Driven Development (BDD).
- Train the students applying the techniques TDD and BDD, solving **2 code katas** (training phase).
- Explain and provide **2 code katas** taken as challenges for the students to solve with TLD, TDD and BDD (practical phase).
- Increase functionalities in some specific exercises.
- Evaluate each programming exercise using: **external quality metric** (*O. Dieste and N. Juristo experiments*), **productivity metric** (*O. Dieste and N. Juristo experiments*), **McCabe's cyclomatic complexity** (internal quality) and **Branch coverage** (internal quality).

5. Experiment Timeline



6. Expected results

The comparison of the "**code katas**" solved with the techniques: TDD and BDD, with those that use TLD; must have the following effects:

- Observe an increase in the understanding of needs.
- Visualize quality improvements in the code.
- Obtain a lower number of errors.
- Observe greater ease to add new features.

7. Required infrastructure

Computers with Internet access.

8. Place of realization

Laboratories of the "Facultad de Ingeniería en Ciencias Aplicadas" in "Universidad Técnica del Norte", Ibarra – Ecuador.

9. References

- [1] R. Martinez, "TDD, una metodología para gobernarlos a todos," 2017. [Online]. Available: <https://www.paradigmadigital.com/techbiz/tdd-una-metodologia-gobernarlos-todos/>. [Accessed: 11-Nov-2017].
- [2] R. A. Rodríguez Morillo, "Test Driven Development (TDD): Ventajas y desventajas - La Oficina de Proyectos de Informática." [Online]. Available: <http://www.pmoinformatica.com/2012/12/test-driven-development-ventajas-y.html>. [Accessed: 17-Jan-2018].
- [3] O. Dieste, E. R. Fonseca, G. Raura, and P. Rodríguez, "Efectividad del Test-Driven Development: Un Experimento Replicado," *Rev. Latinoam. Ing. Softw.*, vol. 3, no. 3, p. 141, 2015.
- [4] D. Fucci, H. Erdogmus, and B. Turhan, "A Dissection of Test-Driven Development : Does It Really Matter to Test-First or to Test-Last?," *IEEE Trans. Softw. Eng.*, vol. 6, no. 1, pp. 1–20, 2015.
- [5] Ifpug, "Function Point Counting Practices Manual," *Group*, vol. on06/23/. 2010.
- [6] F. Sánchez, "Medida del tamaño funcional de aplicaciones software," *Univ. Castilla-La Mancha*, 1999.
- [7] D. Thomas, "CodeKata." [Online]. Available: <http://codekata.com/>. [Accessed: 16-Jan-2018].