1. INTRODUCTION

1.1. Background

Software development life cycle (SDLC) consists of several stages that should be done to develop software applications. One of the testing methods in the SDLC is Agile Testing which is the software testing phase of Agile Software Development. The advantage of Agile Testing is the strong collaboration between related parties and all of them are responsible for the quality of the resulting test, so this will support effective and efficient software testing [1]. One of the models in the agile method is Test Driven Development (TDD). TDD is a software development strategy that creates a test case as the first step and then writing and refactoring the code [2]. With TDD, before writing implementation code, developers will write automated unit test cases for the new functionality they are going to implement. After writing test cases, developers write implementation code to pass these test cases. developer writes multiple test cases, implements code, writes multiple test cases, implements code, and so on [3]. TDD implements a software development process that integrates writing software tests with building the solution itself. TDD has two main concepts, that are unit testing and test-fist. unit test writing test cases that independently validate individual or similar small unit methods. By following unit tests, developers can identify and localize bugs more easily. Whereas the test-first approach of TDD establishs that writing unit tests should ideally define writing the solution itself. However, it is important to note that the process is gradual, test first does not suggest that all test code be completed before starting any part of the solution. As, writing unit tests just writes the appropriate solution unit, before the process is repeated for other units [4].

Code coverage is a way to measure the extent of which a series of tests run a software system [5]. Code coverage evaluation is a matter of identifying the parts of a program that are not executed in one or more running programs. Developers and testers use code coverage to ensure that all or most of the statements in a program have been executed at least once during the testing process. Thus, measuring code coverage is important for code testing and validation during development [6]. The results of the code coverage can be used as a standard in measuring the quality of the applications that are built. The higher the result of code coverage, it will show that the better the quality of the application. On the other hand, the lower the code coverage results, the lower the quality of the applications built. The low code coverage results indicate that there is still numerous code lines that have not been tested. This can have an impact not only on the appearance of bugs/errors at unpredictable times but can also increase the required maintenance costs. To avoid this, it is necessary to strive for the resulting code coverage value to be in the range of 80% and above [5]. Developers often change software in a way that causes tests to fail. When this occurs, the developer must determine whether the failure was caused by an error in the code under test or in the test code itself. In frequent cases, the developer must either fix the failed test or remove it from the test suite. Fixing the test is time consuming but worthwhile, because removing the test reduces the ability of the test set to detect regressions [7]. One of the benefits of implementing TDD is code coverage, which means that all code will be tested. Thus, we have high code coverage and the code is protected from any potential regression.

In 2003, Boby George and Laurie Williams conducted a research on Test Driven Development in an industrial environment. The research, entitled An Initial Investigation of Test-Driven Development in Industry [8], focuses on how programmers evaluate two different developments, that are development using TDD and development using a waterfall approach. From the research it resulted that TDD developers produce higher quality code, which passes 18% more functional black box test cases. However, the TDD developer pair required 16% more time for development. In that study, it was assessed only in terms of programmers but did not show how the value of code coverage resulted from different developments.

This research aims to compare the code coverage results between the software development process that uses TDD and without using TDD. To compare these processes, we use the case study of using a point of sales application, called “dRetail”. This application is developed using Java and the code coverage testing tool used is JaCoCo, which is a software tool used for measuring code coverage for Java. By carrying out this study, we will be able to investigate in the program code there are conditions or object loops that should not be needed in order to optimize the process of running the program that was created and can achieve the desired test case results, for that whether the application of TDD will improve results code coverage indicating that applications generated from TDD are of better quality.

1.2. Framework

Based on the above background, the formulation of the problem is

RQ1: Measuring the value of code coverage on the test driven development approach used for software development.

RQ2: Analyze the difference between manual testing without TDD and automated TDD testing.

The framework of the problem in this final project is a case study using the dRetail application. This application is built using the Java programming language so that the code coverage testing tool used is JaCoCo, which is a software tool used for measuring code coverage for Java.

1.3. Objectives

The objectives to be achieved in this research is to be able to apply the Test Driven Development method to application development and ensure that the code coverage value obtained is higher than the application development without using the Test Driven Development method.

1.4. Writing Sequences

The next part is part 2, that is related studies which is an explanation of the literature study related to the Final Project topics taken. The next part, which is part 3, contains the system that was built, that is an explanation of the methods and stages of the research. Furthermore, there is part 4, that is evaluation, which is the result of testing and test analysis. Then in section 5, the conclusion contains a summary of the final results obtained from the test results. The last section contains a bibliography containing supporting literature sources from this research.