Comparative Study of Test-Driven Development and Behavioral-Driven Development

Myint Myint Moe
University of Computer Studies (Hpa-an), Kayin State, Myanmar

Abstract:- **TDD and BDD were popularized by XP (Extreme Programming) is an agile software development framework. Test-Driven Development (TDD) is a software development technique which applies unit tests to incrementally deliver small pieces of functionality. TDD is a developer-focused process.In TDD (Test Driven Development), first come tests and then the code. The minimal piece of code is written in order to pass the designed test. In other words, it is the process of testing the code before its accretion writing. If the code passes the test, then developers can carry on to its refactoring. Behavioral Driven Development (BDD) is a customer-focused process. It is based on the full and clear understanding of the system or module behavior but in the terms of business/client. The tests for TDD are generated by developers for developers. The test for BDD can be created by testers or technical managers.**

Key words: **TDD, BDD Developer, End User, Tester.**

I. INTRODUCTION

Test-Driven Development is the easiest one to reply. It is a test-first software development methodology that it demands writing test code before writing the real code that will be tested. In Kent Beck's good book:The style here is to write a few lines of code, then a test that should run, to write a test that won't run, then write the code that will produce it run.After figuring out how to write one small piece of code, now, instead of just coding on, developers want to get immediate feedback and practice "code a slight, test a slight, code a slight, and test a slight." So developers immediately write a test for it.So TDD is a lower-level, technical methodology that developers use to make clean code that works.

Behavior-Driven Development is a methodology that was generated based on TDD, but evolved into a process that doesn't concern only developers and testers, but instead deals with the entire team and all important stakeholders, technical and non-technical. BDD started out of a few simple questions that TDD doesn't answer well: how much tests should developer write? What should developer actually testand what shouldn't developer? Which of the tests developer write will be in fact important to the business or to the overall quality of the product, and which are just my over-engineering?Business stakeholders and domain experts often can advise engineers what kind of tests sound like they would be useful but only if the tests are high-level tests that deal with important business aspects. BDD imposes as business-like tests and reserves the word "test" for low-level, technical checks such as data validation. The important part is that while tests can only be created by programmers and testerscan be collected and analyzed by designers, analysts, and so on.

This paper is composed in six sections: In section 1. Introduction 2.Objectives, 3. Background Theory, 4. Motivation,5. Contribution and 6. Conclusion.

II. OBJECTIVES

The aim of TDD is to produce the code clearer, simple andbug-free. TDD starts with designing and developing tests for small functionality of an application. The goals of Behavioral-Driven Design (BDD) is to verify that the application meets the specification; to validate that the design does what the customer wants; to help the customer understand the use of the application; and to ask questions about the behavior of an application before and during development. Behavioral-Driven Development (BDD) is perhaps the biggest source of confusion. When applied to automate testing, BDD is a set of best practices for writing greattests. BDD can be used together with TDD and unit testing methods.BDD address is implementation detail in unit tests. In TDD, the test is written to examine the implementation of functionality, but as the code develops, tests can give false results. BDD is also atest-first approach, but differs by testing the actual behavior of the system from the end users perspective.

III. BACKGROUND THEORY

Test-Driven Development (TDD) is a simple process. Test-Driven Development (TDD) concentrates on the "inside-out" outlook and generates tests from a developer's perspective. The methodology focuses explicitly on unit tests. The developer accepts a requirement and then converts it into a defined test case. Then the developer writes the code to pass those specific test cases only. This practice is purposed to prevent unnecessary updates that do not address the requirements. Test-Driven Development (TDD) forces developers to concentrate on product requirements before writing code, a fundamental difference from traditional programming where developers write unit tests after the writing the code.Behavior Driven-Development (BDD) concentrates on the "outside-in" outlook,which is related to

business benefits. The process is very alike to TDD. BDD demands guidance from developers, testers, and users to assure answers to the "whys" behind a user story. BDD is mainly an extension of the TDD methodology. The developer defines a test case, tests code to check that the test case will fail. Next, the developer writes the code necessary to pass the test case and then tests the code to assure compliance. Where BDD disagrees from TDD is how the test case is specified. BDD tests cases exist in a way that defines the desired behavior. The clear language of BDD test cases achieves it simple for all stakeholders in a development project to understand.

3.1Test-Driven Development (TDD)

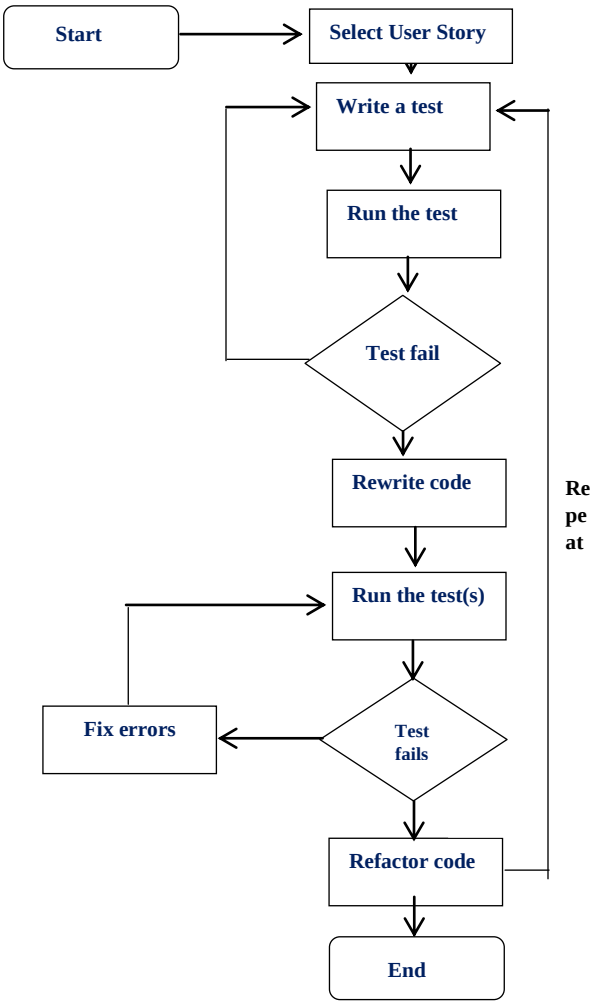

Figure: 1 Test-Driven Development flow

The TDD process is presented in Figure 1, and consists of the following steps:

1) Select a user story,
2) Write a test that fulfills a small task of the user story and run this test. Then produces a failed test,
3) Re-write the production code necessary to implement the feature,
4) Execute the pre-existing tests again, where any failed test is existent. When the code is correct, completely and finally go to the refactoring stage.
5) As the refactoring stage is finished, the correct production code is produced and the user can select new user story again.
6) This method produces some benefits, focus on the commitment of increasing the quality of the software product and the productivity of programmers.

3.2 Behavioral- DrivenDevelopment (BDD)

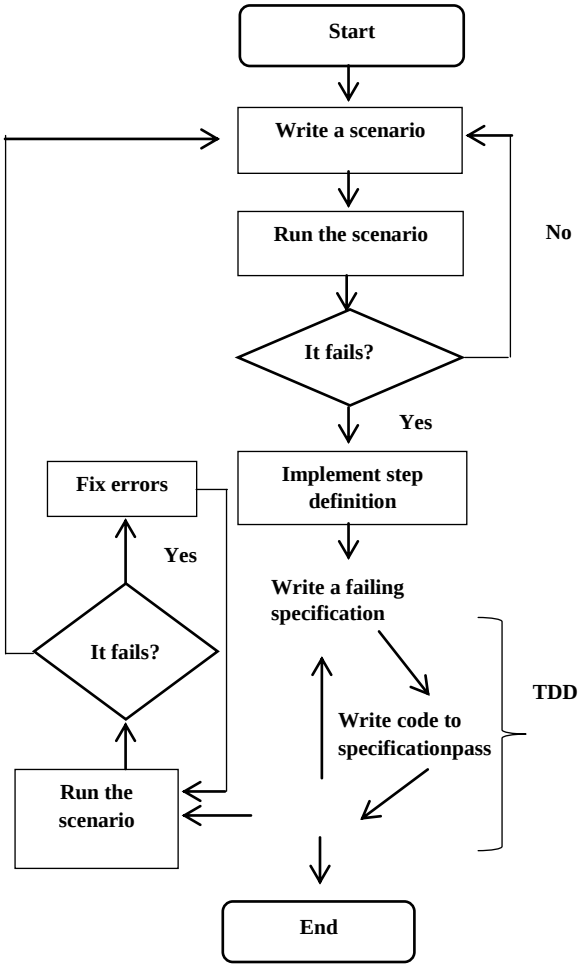

Figure: 2Behavior -Driven Development flow

BDD, initially proposed by Dan North, is a synthesis and refinement of software engineering practices that help teams generate and deliver higher quality software quickly. The BDD process is similar to TDD and follows these steps:

1) Write a scenario;
2) Run the scenario that fails;
3) Write the test that corresponds to the specifications of the scenario;

4) Write the simplest code to pass the test and the scenario, and lastly;
5) Refactor to eliminate duplication.

3.3 Advantages and Disadvantages of TDD

The best TDD can do, is make sure the code is doing what the programmer thinks it should do.

- Helps prevents defects
- Helps document code with executableexamples
- Helps programmers really understand their code
- Helps support refactoring as needs and design changes
- Encourages better design (more cohesive modules that are loosely coupled)
- Provides early warning to design problems (when they are easier to fix)
- Creates an automated regression test suite, basically for free
- Programmers learn how to write other kinds of tests
- It encourages small steps and the principle that it is easier to keep a system working

Do not apply points of TDD:

- A challenge to learn
- Hard to apply to legacy code
- Lots of misconceptions that keep programmers from learning it

3.4 Advantages and Disadvantages of BDD

If software team plans to implement BDD, here are a few points that will benefit the software team.

- Software teamis no longer defining 'test', but is defining 'behavior'.
- Better communication between developers, testers and product owners.
- Because BDD is explained using simple language, the learning curve will be much shorter.
- Being non-technical in nature, it can reach a wider audience.
- The behavioral approach defines acceptance criteria prior to development.

Even the best development approaches can have problems and BDD is no exception. Some of them are:

- To work in BDD, prior experience of TDDis required.
- BDD is incompatible with the waterfall approach.
- If the requirements are not properly specified, BDD may not be effective.
- Testers using BDD need to have sufficient technical skills.

3.5 Difference between TDD and BDD

- BDD focuses on the behavioral aspect of the system rather unlike the TDD focuses on the implementation aspect of the system.
- TDD leans towards the developer-focused side of things; the BDD is where the step of making it more customer-focused comes in.
- BDD is usually done in very English-like language, and often with further tools to make it easy for non-techies to understand. This allows much easier collaboration with non-techie stakeholders, than TDD.
- BDD focuses on the behavioral aspect of the system rather than the implementation aspect of the system that TDD focuses on. BDD gives a clearer understanding as to what the system should do from the perspective of the developer and the customer. TDD only gives the developer an understanding of what the system should do.

IV. MOTIVATION

According to the study, there has found TDD compared Traditional techniques. In addition, there has discovered TDD compared incremental Test- Last development. This paper describes TDD compared BDD.

V. CONTRIBUTION

Test-driven development (TDD) is a development technique where developer must first write a test that fails before you write new functional code. BDD is a software development approach that has developed from TDD. It differs by being written in a shared language, which improves communication between tech and non-tech teams and stakeholders. In both development approaches, tests are written advance of the code, but in BDD, tests are more user-focused and based on the system's behavior.This paper investigates about TDD and BDD. It studies pros and cons of TDD and BDD. This explores differences of TDD and BDD.

VI. CONCLUSION

Test-Driven Development is a process for when developers write and run your tests. Following it produces it possible to have a very high test-coverage. Test-coverage refers to the percentage of codes that is checked automatically, so a higher number is better. TDD also reduces the probability of having bugs in developer's tests, which can otherwise be difficult to track down.BDD illustrates the methods of developing a feature based on its behavior. The behavior is basically explained in a very simple language which can be understood by everyone in the team who is responsible for the

development.BDD is a better approach that has actually evolved from TDD, as a way to eliminate the shortfalls of TDD. So there is positively no damage in implementing both approaches-one to support the quality of the code the developer writes, and has the other to support the behavior of the system defined by the product owner.

REFERENCES

[1]. ShawetaKumar, Sanjeev Bansal; "Comparative study of Test-Driven development with Traditional Techniques"; (IJSCE); ISSN: 2231-2307, Volume-3, Issue-1, March 2016.
[2]. Luis A. Cisneros, Marisa Maximiano, Catarina I. Reis, José Antonio Quiña Mera; "An Experimental Evaluation of ITL and TDD"; ICSEA 2018 : The Thirteenth International Conference on Software Engineering Advances; Copyright (c) IARIA, 2018. ISBN: 978-1-61208-668-2.
[3]. Helen Johnson, works at QATestLab; Feb10, 2017; www.quora.com/profile/ Helen-Johnson-76.
[4]. Kamil Nicieja, CEO of Ada and author of "Writing Great Specifications"; Feb 11, 2017; www.quora.com/profile/
[5]. Vinai Amble; May 31, 2018; http://dannorth.net/introducing-bdd.
[6]. Manoj; November 5, 2012; September 19, 2014;https://vibhuaggarwal.wordpress. com/
[7]. Sergey Sergyenko; Updated 19 February 2014; http:// code.google.com/
[8]. DuckDuckGo; www.code.google.com/p/concordion/
[9]. Shanmugam Lakshmanan; March 25, 2017; www.codeqa.com/
[10]. Kevin Dunne; http://www. gasymphony.Com/ system from the end users perspective