International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume: 3 | Issue: 4 | May-Jun 2019 Available Online: www.ijtsrd.com e-ISSN: 2456 - 6470

# Comparative Study of Test-Driven Development (TDD), Behavior-Driven Development (BDD) and Acceptance Test–Driven Development (ATDD)

**Myint Myint Moe**

University of Computer Studies, Hpa-An, Kayin State, Myanmar

***How to cite this paper***: Myint Myint Moe "Comparative Study of Test-Driven Development (TDD), Behavior-Driven Development (BDD) and Acceptance Test–Driven Development (ATDD)" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-3 | Issue-4, June 2019, pp.1-3, URL: https://www.ijtsrd.com/papers/ijtsrd23698.pdf

IJTSRD23698

## ABSTRACT

TDD, BDD and ATDD were introduced by XP (Extreme Programming) is an agile software development framework. They are unit testing approaches. TDD, BDD and ATDD are a software development technique which uses unit tests to incrementally deliver small pieces of functionality. TDD is a developer-focused process. In Test Driven Development (TDD), 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 accrual writing. If the code passes the test, then developers can proceed to its refactoring. Behavior-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 created by developers for developers. The test for BDD can be written by testers or technical managers. Acceptance Test-Driven Development (ATDD) is towards the developer-focused side of things. ATDD is a technique where the entire team collaborates to define the acceptance criteria of a story before the implementation actually begins. These acceptance tests are supported necessary information. Using the Given-When-Then format, ATDD approach can implement.

***KEYWORDS***: *TDD, BDD, ATDD, Developer, End User, Tester*

## 1. Introduction

Test-Driven Development is the easiest one to apply. It is a test-first software development methodology. Before writing the actual code, it requires writing test code that will be tested. In Kent Beck's good book: The style is to write a few lines of code then a test that will make it run.

After figuring out to write one small piece of code, developers want to get immediate feedback and practice "code a little, and test a little." So developers immediately write a test for it. TDD is a low-level, technical methodology that developers use to produce clean code that works.

Behavior-Driven Development based on TDD is a methodology. BDD evolved into a process that doesn't concern only programmers and testers, but deals with the entire team and all important stakeholders, technical and non-technical. Business stakeholders and domain experts often can determine engineers what kind of tests emit like they would be useful but only if the tests are high-level tests that deal with important business aspects. BDD calls 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 developers and testers can be collected and analyzed by designers, analysts, and so on.

ATDD is a collaborative application where users, testers, and developers define automated acceptance criteria early in the development process. ATDD aids to ensure that all project members understand precisely what needs to be done and implemented. This process usually involves establishing the criteria first, most often from a user outlook. Thereafter, acceptance tests are developed and run to detect the outcomes of failure with the right code based on. Small code is then developed to run the program, more acceptance tests are run again, and the results are validated. Before the final program is developed for use, refactoring is then carried out based on the results of the acceptance tests.

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

## 2. Objectives

The primary goal of TDD is to accomplish the code clearer, simple and bug-free. Test-Driven Development starts with designing and developing tests for small functionality of an application. The goals of Behavior-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. Behavior-Driven Development (BDD) is perhaps the biggest source of confusion. When applied to automate testing, BDD is a set of best practices for writing great tests. BDD can use together with TDD and unit testing methods. BDD address is implementation detail in unit tests. The test of TDD is written to analyze the implementation of functionality, but as the code evolves, tests can give fail results. BDD is also a test-first approach, but differs by testing the actual behavior of the system from the end users perspective. ATDD intends to support collaboration among the user, developer, and tester to ensure that acceptance tests exist before writing any code. The acceptance test is written from the users outlook and function as a requirement for how the software should function.

## 3. Background Theory

Test-Driven Development (TDD) is a simple process. Test-Driven Development (TDD) focuses on the "inside-out" perspective and creates tests from a developer's perspective. The methodology focuses specifically on unit tests. The developer catches a requirement and then converts it into a particular test case. Then the code is written by the developer to pass those specific test cases only. This practice is intended to prevent unnecessary updates that do not address the requirements. Before writing code, TDD forces developers to focus on product requirements, a fundamental difference from traditional programming where unit tests are written by developers after the writing the code. Behavior Driven-Development (BDD) focuses on the "outside-in" perspective. These are related to business outcomes. The process is very similar to TDD. BDD requires guidance from developers, testers, and users to ensure answers a user story. BDD is largely an addition of the TDD methodology. The developer defines a test case, tests code to analyze that the test case will fail. Next, the code is written by the developer necessary to pass the test case and then tests the code to ensure compliance. BDD tests cases exist in a way that defines the desired behavior. The clear language of BDD test cases produces it simple for all stakeholders in a development project to understand. Acceptance Test-Driven Development (ATDD) is an increasingly popular development method. ATDD is closely related to Test Driven Development (TDD) because of its highly collaborative approach. Acceptance Test Driven Development needs participation from customer-facing team members to provide end user stories to the development/testing team. These stories are refined into Acceptance Tests that lead the development process.

### 3.1 Test-Driven Development (TDD)

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.

This method produces some benefits, focus on the commitment of increasing the quality of the software product and the productivity of programmers.

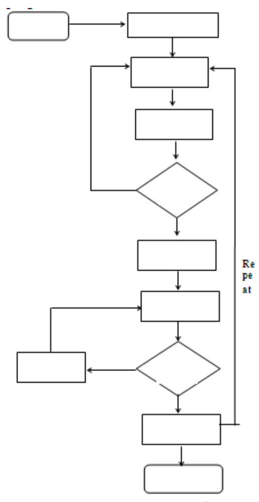

Figure: 1 Test-Driven Development flow

### 3.2 Behavior Driven-Development (BDD)

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. Writes 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.

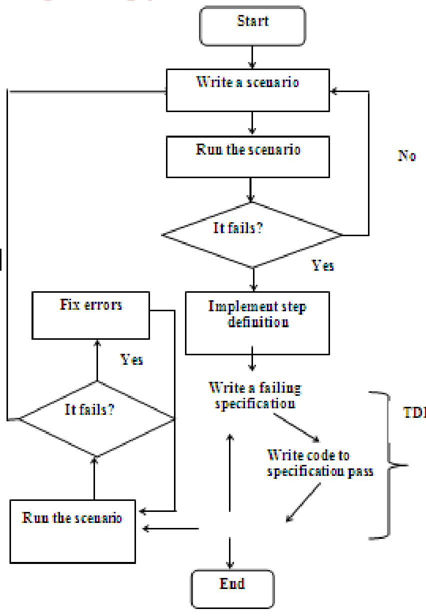

Figure: 2 Behavior -Driven Development flow

### 3.3 Acceptance Test-Driven Development

Acceptance test- driven development (ATDD) is a development methodology based on communication between the business customers, the developers, and the testers. Before developers begin coding, ATDD encompasses acceptance testing, but highlights writing acceptance tests. The ATDD process follows these steps:

1. Select user story;
2. Write acceptance test;
3. Implement user story;
4. Run acceptance test
5. Make little change/Refactor

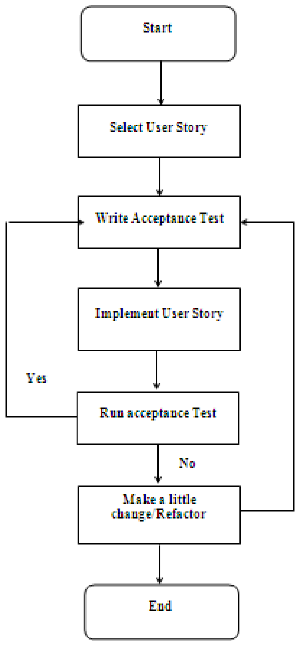

Figure 3: Acceptance Test- Driven Development flow

### 3.4 Advantages and Disadvantages of TDD

The benefits of TDD:

- ➢ Helps prevents defects
- ➢ Helps document code with executable examples
- ➢ 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

The drawback of TDD:

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

### 3.5 Advantages and Disadvantages of BDD

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

- ➢ Software team is 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 TDD is 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.

### 3.6 Advantages and Disadvantages of ATDD

**Advantages:**

- ➢ Improve communication and collaboration between project stakeholders
- ➢ Shared understanding of what a successful implementation means
- ➢ Better coverage of business expectations
- ➢ Faster feed back

**Disadvantages:**

- ➢ New methodology that requires rigor and discipline
- ➢ Find the right balance between people/process/tool

### 3.7 Difference between TDD, BDD and ATDD

- ➢ 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 permits 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 provides 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.
- ➢ ATDD focuses on capturing requirements in acceptance tests and uses them to drive the development. ATDD focuses on external quality of the software.
- ➢ ATDD leans towards the developer-focused side of things like TDD does. The BDD is where the step of making it more customer-focused comes in.
- ➢ TDD focuses on the low level, ATDD on high level.

## 4. 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 and ATDD.

## 5. Contribution

Test-driven development (TDD) is a development technique where developer must first write a test that fails before you write new functional code. In development approaches, tests are written ahead of the code, but in BDD, tests are more user-focused and based on the system's behavior. ATDD leans towards the developer-focused side of things. This permits much easier collaboration with non-techie stakeholders, than TDD. This paper investigates about TDD, BDD and ATDD. Its study pros and cons of TDD, BDD and ATDD. This explores differences of TDD, BDD and ATDD.

## 6. 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 BDD 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 the other to support the behavior of the system defined by the product owner. Acceptance test-drive development makes the implementation process more effective. These techniques have the same goal: write just enough code, reduce developer efforts, build to detailed requirements and continuously test the product to ensure it meets business user expectations.

**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; Feb 10, 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]. Duck DuckGo; www.code.google.com/ p/concord ion/

[9]. Shanmugam Lakshmanan; March 25, 2017; www.codeqa.com/

[10]. Kevin Dunne; http://www. gasymphony.com