ISSN No:-2456-2165

Learning and Understanding: Test-Driven Development in Software Development

Prakriti Dhang
Department of Computer Science,
Malmö University, Sweden

Abstract:- Test Driven Development (TDD) is one of the important approaches in software development. Sometimes, it is very difficult to understand what the codes of a system are doing, so to have a clear understanding one needs to write test cases first with the expected output, for these type of development TDD is required. This is to make the code cleaner and fault free. In software engineering, one has to undergo some testing to meet the company's goal and customers need. One of the software development approaches in software engineering is Test-Driven Development (TDD). An attempt is made to reveal the necessity of Test-Driven Development (TDD) for software development. On reviewing the literatures, challenges and problems were identified while adopting TDD. In this paper, a conclusion is made on TDD as an essential approach in software engineering.

Keywords:- *Software Development, Software Engineering, Test Cases, Test-Driven Development.*

I. INTRODUCTION

Software Engineering is a structured way of creating, designing, implementing, testing, and maintaining software application [1]. For successful production of software application one need to develop some principal, algorithms, tools, and system. Figure 1 shows all the activities that include in the software development process.

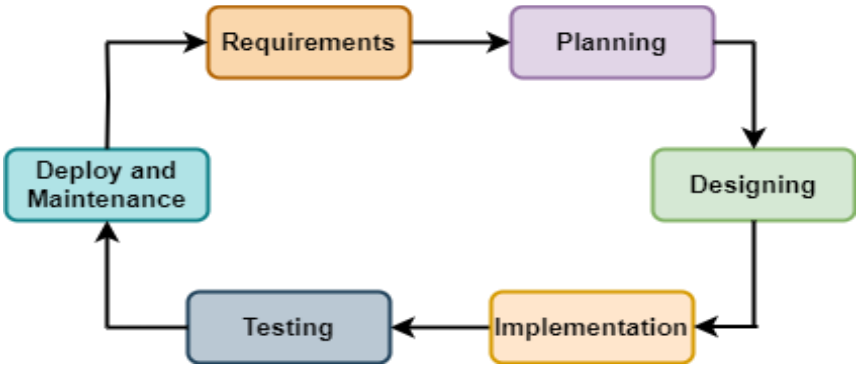

Fig. 1: Software Development Life Cycle

As stated [2]: "While testing any system, it can only show the presence of bugs, it will not show the absence of bugs".

The above statement says on undergoing the testing process it shows only the faults and the locations of the faults in the system. Testing is a good way to find any bugs in the program. So, after every completion of the development cycle, one needs to undergo a testing process. Testing is one of the processes in the software development cycle where testing is a process [3] of finding faults or defects in a software application or in other words one can say testing is to verify software application but not validating the applications.

The main objective of a software engineer is to produce a constructive valuable, qualitative, and reliable products to the customers. Test-Driven Development is one of the structured of testing where one write test cases first then start writing the code and these are written in small chunks [4]. Test-driven means to evaluate the condition of software applications, which means to determine the quality of software applications. TDD is one of the best practices in Extreme Programming(XP) [3]. But now in recent days, TDD can be written in many other programming languages

The paper is divided into several sections. In section 2, the background of TDD and TDD cycle is explained with the merits of using TDD in software development. Section 3 illustrates the literature review of using TDD, findings, and problem faced by the developer while using TDD. Section 4 will be the conclusion of my study with future work. The paper ends with a list of references.

II. TEST- DRIVEN DEVELOPMENT

Test-Driven Development (TDD) is a software development process [3] where the development cycle is divided into short development cycles which includes writing test cases, little modification and finally eliminating the duplicates and these cycles are repeated after each completion of a cycle. So when one writes test cases one needs to fulfill some criteria. The criteria that are needed to be fulfilled are an input value, some boolean conditions, and an expected result. Initially in TDD, one writes the test cases and then run those test to check whether it will pass or fail. After failing the test one modifies the code to pass the test and then refactor

ISSN No:-2456-2165

it. Initially, each test case fails as one has not implemented the code, and this ensures that the test works and achieved a fault. In this way, the functionality of the test can be implemented [3]. The process of TDD with a flowchart diagram is explained in the following section.

A. Test-Driven Development (TDD) process

The TDD process goes through the following steps. In figure 2, each process of Test-Driven Development [1,3,5] is explained.

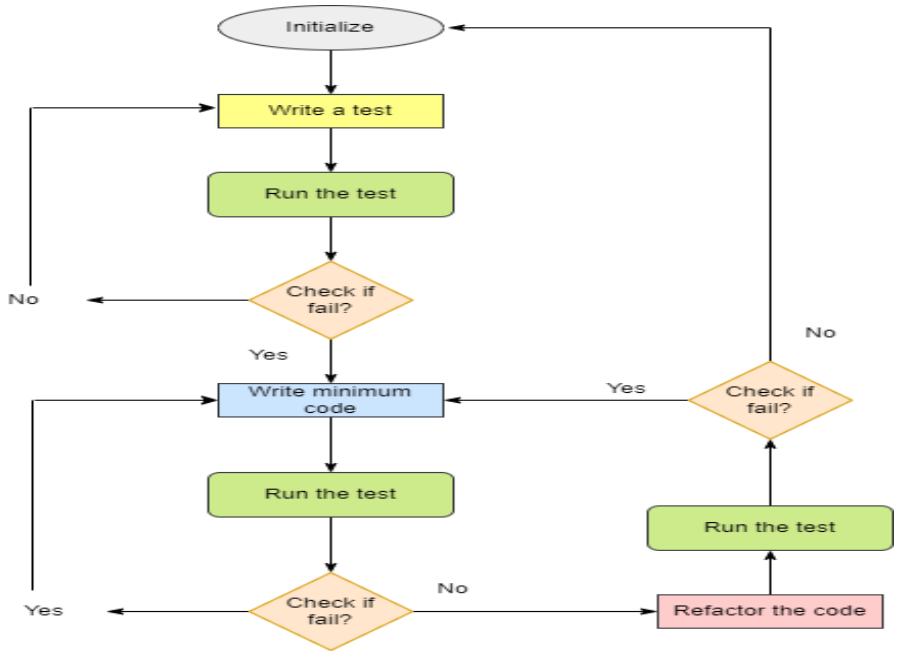

Fig. 2: Test-Driven Development (TDD) process

- **Initialize:** The first step of Test-Driven Development (TDD) process is identifying the functions or features that are required to write test.
- **Write a test:** In this step, one write a test that defines the improvement of a function. Writing a test is important for the developer because the developer needs to understand the requirement and specification of features clearly so as to execute the tests further without any difficulties.
- **Runs to check if the test fails:** After the test is written, tests are executed. This test generates a report defining whether the test has passed or fail. On the report basis, it will proceed to the next step. If it passes it will go to the intial step to write test cases for the next segment of a code and if it does not pass it will move to next step where one make changes in the code. The report is maintained throughout the process. This step mainly focuses on examining the behavior and outputs of a test.
- **Writing minimum code to pass the test:** In this step, as discussed, after the test fails it modifies some functions with a minimum change in the code so to pass the test. It then runs the test to check the status of the test. If it passes this time it goes for refactoring step.
- **Refactor the code and test again:** The final step is the refactor existing code, where all test runs successfully, and move to the next new function for implementation as shown in the above flowchart diagram. Basically, in refactor it checks for any duplication in the test. Refactoring is used to improve the internal structure of a code without changing any functions.

In this way, the whole process is repeated for the next new bunch of functionality.

B. Merits of using Test-Driven Development (TDD)

One can gain interest in using TDD, which are listed below [1,3,5].

- Code coverage is an ideal way of identifying defects or faults in the development process. These coverage is tested after writing the test cases [1,3]. This informs about how much one's code is covering.
- In TDD, for each bug separate tests are created [5]. If one creates a separate test for each bug, the developer spends less time in debugging.
- TDD reduces the cost of regression testing as one can check no new bugs have occurred even when one modifies in the code.
- TDD ensures good quality of code and product, as tests are written first.
- TDD shares a report of the test in documentation after each successful execution of the test. The document describes all the tests and helps programmers to have a clear idea of what each segment of code supposed to do [1,3].

III. LITERATURE REVIEW

Test Driven Development (TDD) to a program developer refers in developing code structure and evaluating test functionality [1]. TDD is a combination of three factors which includes test-first design, test implementation, and refactoring where one first design the test cases then start implementing and run to check the result, after getting expected result one eliminates the duplicate code using refractor. Initially, the test cases fail when executing the first time, then writing a small amount code to satisfy the test condition to pass, then refactoring it to improve the functionality and structure of the code, and ensure that this

ISSN No:-2456-2165

time the tests will pass [3,6] even after modification. Test cases are written in a framework such as JUnit, so whenever one modifies a code, the test can be re-run to make sure the test cases that passed earlier will pass after the modification of the code as well. A mantra [7] for TDD was discovered called "TDD mantra: Red-Green-Refactor". In figure 3 the *"RED-GREEN-REFACTOR"* of Test Driven Development (TDD) lifecycle is shown, where red means tests fail after writing a test. Green means to pass the test. Refactor means eliminating all the duplicates from the code and if it works it starts with a new function of the code.

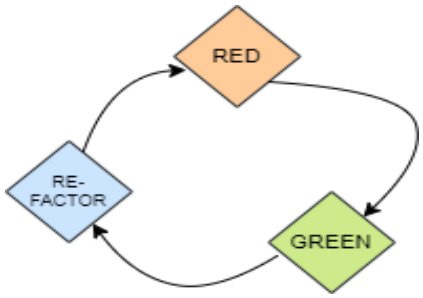

Fig. 3: Test-Driven Development Lifecycle

Test Driven Development (TDD) repeatedly repeats the steps of writing test cases that fail, passing them with minimum code change, and then refactoring the tests with modifications. After achieving the expected test results at each stage, the ideas of writing code increases

This leads to an increase in confidence and an increase in productivity as the developer knows once it work correctly it will works in the future.

As discussed above, test cases are written before developing the code. To write test cases, the developer must clearly understand the specification and the requirements [3] The developer can achieve the requirements and specifications through the use case diagram in Unified Modeling Language(UML). UML diagrams are the most interactive way of describing the interaction between a system and the user [8]. Class diagrams are used to write the methods. For better understanding, the internal and external behavior of a system, state diagrams, and activity diagrams are required for TDD approach, whereby looking at these diagrams one can write all possible test cases. As an example, possible test cases are shown in table 1 (taken from my academic project work), there can be more test cases.

Table 1: Test Cases in Test Driven Development (TDD)

Test cases with expected and actual results					
Test Cases	**Test Steps**	**Input**	**Excepted result**	**Actual result**	**Pass/Fail**
Test Case 1	User Identification: User logged in with user-id and password	Correct input of user-id and password	A message displays: Successful login	A message displays: Successful login	Pass
		Incorrect input	Failure message displays	Failure message not displayed	Fail
		Incorrect input	Failure message displays	Failure message displays	Pass
		Text field blank	Message displays: Fill blank spaces	Message displays: Fill blank spaces	Pass
Test Case 2	Documents submission	Checks for the file size of the document	Message displayed as invalid file size	Message not displayed	Fail
		Checks for the file size of the document	Message displayed as invalid file size	Message displayed as invalid file size	Pass

JUnit is one of the testing framework [9] of TDD in java platform. In this testing framework, one write test cases and run it. A developer can write a test in any testing framework [3,5] like PyUnit which is a testing framework for Python and there is a testing framework for .Net as well and many more testing frameworks are available to write test cases.

Few companies admit that adopting TDD can affect the productivity, internal quality, external quality, and test quality if there are dealing with the complex system and some companies admit that productivity will increase with a TDD approach [10]. As with TDD, one can improve external quality of code as in external quality faults are detected quickly, it also improves the internal quality and test quality as it is observed that fixing faults in code is much easier than designing. In [11] the study was based on verifying the skills required to apply TDD in industries. A quasi-experiment based on the analysis of the level of external software quality and productivity has been made among 30 people from the industry. The people were divided into three groups on the basis of having knowledge of TDD. The analysis shows that there is no such difference in the external software quality and productivity. Apart from this, it has been observed statistically that having knowledge about TDD could increase the external software quality and productivity. A [12] with 30 undergraduate third-year students from Computer Science background has undergone a certain procedure to analyze the effects on the internal quality of software as well as the developer's productivity when using TDD and retainment of TDD over months. Analyzation was based on two statistical approaches descriptive and inferential statistics. Descriptive statistical analyzation was plotted in a boxplot and inferential statistical was displayed in a tabular format. It has been shown that the external quality of the software product and the productivity of the developers affects neither of them while using TDD. Rather, the differences are only in producing tests among TDD users and non-TDD users. Students who were using TDD produced more tests than the students who were not using TDD. Thus, on producing more

number of tests one can retain TDD for better improvement in writing tests.

TDD is not only to apply in development of software, TDD can also be applied in the development of a randomized algorithm [13]. Randomized algorithms are those types of algorithms which help to solve the problems when having more results that are not expected to have. In this study, a framework for Junit called "ReTest" has been developed. A case study, with ten participants which were further divided into two groups according to their experiences with programming knowledge. All the participants have been trained about TDD and developed some function using TDD with ReTest framework. Analysis has been done by asking questions and has been found that the ReTest framework will be useful to the developers leading with such type of algorithms. An analysis of using Test-Driven Development approach in startup companies [14] discussed the merits and demerits of using Test-Driven Development. For analyzing, a set of research questions for two companies has been made. The analysis process was followed by thematic analysis on interview data. The analyzation results show the demerits of TDD that if a developer not having proper knowledge of using TDD, will unable to write test cases. The main advantage of Test Driven Development (TDD) is that one write the test cases before start developing the code. By this method, one can estimate the expected outcome. A developer knows the test will fail initially but after modifying the code, it will pass at a point and one also has to verify the previous tests, that the tests passed before is still passing even after modification. Once the function is implemented [3], this process helps the developer, to not to miss any section of the code.

IV. PROBLEMS AND CHALLENGES

Here are some problems and challenges on using the TDD method are listed from the review.

The main problem with TDD is understanding the requirements of the tests. If a developer doesn't know how to write tests, then the developer will not able to write the required code for the test. One can find another problem with time consumption where the company changes the design of the system repeatedly, one also need to change test cases simultaneously. As one write a test, then run the test with some modification and make sure that it will still run even after the modification. If one doesn't change accordingly the whole code will stop working and one has to repeat the whole process again.

TDD model has some limitations while applying to GUI development [15]. The major issue was in the user interface where requirements were not specified clearly. The issue may be found with the window size, templates and this becomes a problem when undergoing regression testing and maintenance. Challenging part in TDD is dealing with a complex test. For a complex test, effective test cases required in order to fulfill the requirements and these test can be difficult to build and more time will take.

V. FINDINGS

In [3,6,7] one can achieve better test coverage and confidence in developing and implementing code when using TDD method.

Many Software companies support Test Driven Development (TDD), as TDD produce the code simpler and reduces a time in implementating the code. TDD produces better result and design which increases the software quality. TDD generates a report to help software engineer to know about each segment of code so that they can a write test on the report basis.

Using TDD is not the ultimate process, after using TDD one need to go through a system testing [1] which is one of the levels in software testing where one needs to validate the system. In the validation process, it checks whether the system has fulfilled the customer's need or not, it also checks the quality, reliability and also checks that the system is giving output as per expectation.

VI. CONCLUSION AND FUTURE WORK

Test Driven Development (TDD) is used in the software development process as it is easy to adaopt. This is because TDD helps the developer to focus on small segments of code at a time. As TDD has some merits and demerits but still developers are adopting the use of TDD as one say this is a productive way of developing software. The main benefit of TDD is one can analyze the design before start coding. Once the developer acquires knowledge of using TDD, it is much easier to handle test cases. The ultimate conclusion is that the Test Driven Development (TDD) process is one of the important approaches in improving test quality, external quality, internal quality of software products.

In the future, more studies can be carried out on TDD in large companies dealing with complex problems.

ACKNOWLEDGEMENT

The work is supported by Department of Computer Science, Malmö University.

REFERENCES

[1.] Sommerville I, “Software Engineering”, 10th ed, England: Pearson, 2016. pp. 102-134, 227-252.
[2.] Dijkstra EW, "The Humble Programmer." , October 1972, Comm. ACM 15, NUmber: 10, Volume: 15
[3.] Krampell M, Unit Testing and TDD, PPT.
[4.] Jeffries R, Melnik G, ``Guest Editors' Introduction: TDD--The Art of Fearless Programming", Published 2007 in IEEE Software, DOI:10.1109/MS.2007.75.
[5.] Nair J, "An Introduction to Test-Driven development", TestLodge, Updated: September 2018, Published: February 2018.
[6.] Madeyski L, Kawalerowicz M: “Continuous Test-Driven Development — A Novel Agile Software Development Practice and Supporting Tool”, in Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software

Engineering (ENASE), ANGERS, 2013: p.260-267. DOI: 10.5220/0004587202600267.
[7.] Beck K, "Test Driven Development: By Example", 2002, Addison-Wesley, Boston, MA, USA.
[8.] Bennett S, Skelton J, and Lunn K, "Schaum's outline of UML", 2001, New York: McGraw-Hill, ISBN: 0-07-709673-8.
[9.] Astels D ,"Test Driven development: A Practical Guide", 2003, Prentice Hall Professional Technical Reference.
[10.] Turhan B, Diep M, Layman L,H.Erdogmus,"How Effective is Test Driven Development" ,October 2010.
[11.] Fucci D, Caivano D, Romano S, Scanniello G, Juristo N, Teresa M, and Thuran B, "Towards an operationalization of test-driven development skills: An industrial empirical study", 2015, Information and Software Technology, Volume 68, December 2015, pp 82-97, https://doi.org/10.1016/j.infsof.2015.08.004
[12.] Fucci D, Caivano D, Romano S, Scanniello G, Juristo N, Teresa M, and Thuran B, "A Longitudinal Cohort Study on the Retainment of Test-Driven Development", 2018, ESEM, October 2018, Oulu, Finland, arXiv:1807.02971v1 [cs.SE].
[13.] Ivo AS, Guerra EM, Porto SM, Choma J, and Quiles MG, "An approach for applying Test-Driven Development (TDD) in the development of randomized algorithms", 2018, Ivo et al. Journal of Software Engineering Research and Development (2018) 6:9, doi: 40411-018-0053-5.
[14.] Kenigbolo MS, "A case study of Test Driven Development", 2017, ResearchGate, Thesis, DOI: 10.13140/RG.2.2.27852.92803.
[15.] Dhandapani S, "GUI Development in TDD Model-Case Study" ,July 14, 2016, Journal of Software, doi: 10.17706/jsw.11.11.1139-1144.