**Graduate Diploma in Information and Communication Technology**

**BCCE301 Cooperative Education Project**

**October 20, 2017**

**Analysis of JavaScript Testing Frameworks in Canterbury**

**Methodology Essay**

**On Test Driven Development (TDD)**



Swathi Soman

99162569

20/10/2017

**Department of Computing**

**Te Horo Rorohiko**

Contents

[Introduction 2](#_Toc496222814)

[Part A – Test Driven Development(TDD) 2](#_Toc496222815)

[What is TDD? 2](#_Toc496222816)

[Why TDD? 3](#_Toc496222817)

[History of TDD 3](#_Toc496222818)

[Test-driven development cycle 4](#_Toc496222819)

[Advantages of TDD 6](#_Toc496222820)

[Disadvantages of TDD 6](#_Toc496222821)

[Summary 7](#_Toc496222822)

[Bibliography 8](#_Toc496222823)

# Introduction

In the report, the methodology essay topic for the project is Test Driven Development(TDD). The report is split into three sections: 1) Theory on TDD, 2) Processes involved with the use of TDD in the project and 3) a compare and contrast of TDD used in the project with the theory

# Part A – Test Driven Development(TDD)

## What is TDD?

The method Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle. "TDD" refers a programming style which has three activities: coding, testing and design (refactor) (Agile Alliance, n.d.).

Requirements are analysed to certain automated test cases, and then the improved software is used to pass the new tests. It is an evolutionary approach to development, which combines TDD where tests are written before you write a production code to fulfil that test and refactoring (Test-driven\_development, n.d.). Refactoring is the process of changing the structure of the code without changing its behaviour; with this, the code developed would be easier to understand.

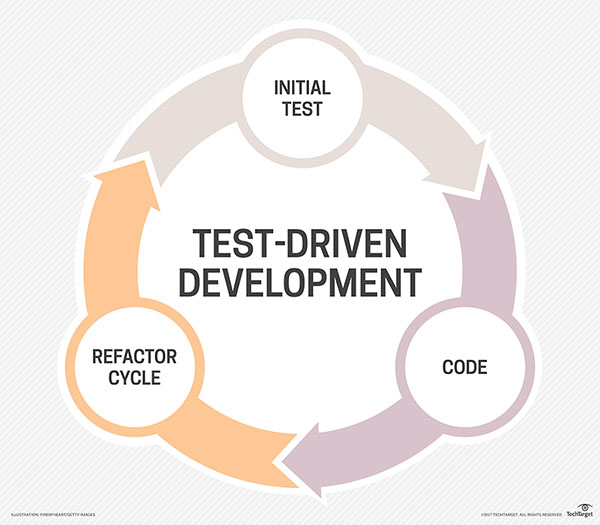
For every small functionality of an application in TDD it starts with designing and developing tests. In the below given figure1 the developer writes an automated test case that is initially failing which defines improvement or new function, then will produce a code to pass the test, and then the new code would be refactored to the acceptable standards.

Fig1: Test-Driven Development (Rouse, n.d.)

It starts with the designing and developing tests for all little functionality of an application. In TDD approach, the test which is developed first specifies and validates what the code must do. It can be shown as “red-green-refactor cycle”. Before the code is written and any new functionality is added to an application, an automated test would be written by the developer stating how the code should behave, and wait for it to turn red (fail to pass). Then the code would be written to the specification which would turn the test to green (test passes). Finally, the developer will make sure that code written will be clean and executable (refactoring) (wikipedia, 2017).

## Why TDD?

TDD is part of holistic design, development and testing process, it’s not the testing technique. The main ides of TDD is that the test would be written first not the code to ensure that the code works. The test is written first and then the code which will get all the tests to pass. This is known as a Test-First approach.

There two views in software development on why to practice TDD.

* In the first view the TDD is mentioned as the technique which can be used to specify the requirements and design before the actual code is written.
* In the second view TDD is the technique which can be used by the programmers to develop the actual code.

Regardless of the views mentioned, TDD can not only improve the code, but it can improve the complete design and the implementation of the software system (Barber, 2017).

By using TDD it solves many issues by implementing refactoring, it makes the programmers to follow good practice of coding and constant testing. A good program not just works, but it’s simple and easy to maintain. A code which is written well not only saves the time and money it can change the test time according to the requirement in business. At starting stage, the programmers feel that lot of effort must be put forth, but gradually it facilitates easier ongoing delivery. The code written by another person is always difficult to understand, it happens when each person has their own style. But it is not necessary that the code written by each person should be good and efficient. Many coders do not follow certain coding practices such as, commenting properly, providing good indentation, vertical coding, etc (helpingtesters, 2017).

## History of TDD

Kent Beck attributed the latest rediscovery of TDD, he is the creator of the extreme programming. TDD was widely accepted in Software Development community through Extreme programming and Agile software development.

* Extreme programming (XP) is a software development methodology, can be used to improve the quality of the software and its responsiveness according to the customer requirements. It is a type of agile software development which promotes short development cycles, which is used to improve the productivity of the customer and introduce checkpoints at which new customer requirements can which is intended to improve productivity and introduce checkpoints at which new customer requirements can be acquired (Extreme programming, n.d.).
* Its known that Kent Beck “rediscovered” TDD since the early days of computing which was in 1960s dates the prototype of TDD. The biggest difference in between the new and the old TDD is that the old TDD requires manual testing whereas the modern TDD is facilitated to automated. As of today, TDD is truly an automated test-driven (Barber, 2017).

## Techniques of TDD

The hardest thing in TDD is writing tests. The syntax or rules can be learned easily in very less time. The problem is how to an idea of what to be done and then verify to check if the function developed works.

When people to about TDD and use it. The question arises of how can it be used and how many functions are required, later test it. TDD is the method of developing or thinking about the code in a unique way.

Think of any kind of idea and something which is different and can be tested. Once the tests are written and it becomes a routine like writing conditions and loops. Test can be written without thinking about it lot.

* A technique is built up to address the constraints of TDD examined above with respect to the production of new techniques on teaming up classes under an easy falling flat unit test. When another strategy is created on an alternate class, TDD is recursively applied. So before making the new technique, another test is made on alternate test class, which is compared to the different class. This indicates that there are two simple tests which is coming up. So the TDD rules are changed and run the test of second class while with the new method. Once the method is done it can be confirmed that there is only one failing test and can start working on the original method.
* Upon getting the current test to pass, there are few scenarios which can be chosen to write the upcoming tests. While choosing a scenario choose those which would fail at first and not to choose a scenario which passes automatically. This shows the pattern between the pass and fail of the test. When all the scenario which indicates failure is tested, a scenario is added which can be passed. The production code gets over and the tests are still written until it confirms that the other scenarios given are correct.

#### Refactoring

Refactoring is one of the procedure of TDD and that is its main advantage. It is mainly because refactoring requires code which finishes the test scope, hence when the programmer is following TDD, the code developed would follow the requirements constantly. Refactoring of the code is the reconstructing the present code to extent the design and the structure of the code. Before refactoring test coverage is required, since test will guarantee that it will work as required and the functionality of it do not change.

Once the programmer refactors’ the function, they can run the test again to ensure that the behaviour of the test works as expected and the test pass. Thus, it will give great confidence to the developer to refactor the code that they would have avoided. The developers will never know if the changes they made have broken anything (wikipedia, 2017).

## Advantages of TDD

TDD is a software development method in which the code is tested constantly. TDD develops software of high quality in a short span of time. It saves cost, effort and much more. Below list mentions few advantages of TDD

* It is first a safety net. All the existing code is tested. As human programmers make mistakes when the code is changed or improved
* TDD will allow to spot bugs quicker. All the errors in the program while testing is identifiable easily.
* TDD focuses on good design. To make the code testable, it must have a good structure.
* TDD helps to maintain the code easily and refactor. The code implemented is of good clarity.
* Helps prevents defects – If there is any issue with the requirements or with design at the beginning it is found. If there are any small mistakes it is caught immediately and thus helps the programmers to find the mistakes.
* Helps the programmers really understand their code – The code written by the programmers are easily understandable
* Helps support refactoring as needs and design changes
* Provides early warning to design problems: If there is any errors in the code. Early warnings will be provided
* Programmers learn how to write other kinds of tests: Programmers learn different form of tests
* It encourages small steps and the principle that it is easier to keep a system working than fix it after you break it (Grenning, 2016).

## Disadvantages of TDD

Even though there are many advantages for TDD, there are few disadvantages, which makes the TDD less efficient. Below list shows some of the main disadvantages

* It consumes a lot of time and effort for the development - Takes lot of time thinking of how to write the test, which requires lot of effort.
* Concentrating on the least difficult plan now and not thinking ahead can mean major refactoring prerequisites.
* It's hard to write great tests that cover the fundamentals and keep away from the unnecessary.
* It requires significant investment and push to keep up the test suite – it must be reconfigured for most extreme esteem.
* If there is rapid change in the design, the test should be changed frequently. Most of the time would be wasted by writing tests (Hill, 2015).

# Part B – Theory on TDD

#### Reason to choose TDD

TDD was chosen due to my familiarity and my experience on working with it. I had a heap knowledge on the subject as I have worked with it and have studied the subject in SE101 JavaScript Programming. This knowledge helped me in choosing TDD which would help in developing a well-structured code. TDD fulfils the project requirements and can be used by the staffs, students or developers for their projects. It is equally beneficial for me and the future developers.

#### Applying TDD

The below mentioned steps describes about how TDD is applied and I was introduced to this topic through SE101. Which demonstrate the systematic programming on Jasmine Framework and its development using TDD.

**Write a Test**

**Refactor**

**Repeat**

**Make the test Pass**

**Test Fails**

## Test-driven development cycle

The below given steps shows the cycle of Test Driven Development cycle

#### 1: Write a failing test

Writing a test is first step of TDD. The programmer should clearly analyse the specifications and requirements before the test is written. Once a specification and requirements has been understood, the programmer can write an automated test to test given requirements. The developer writes the test and minimal code to run the test. The test fails initially as the actual functionality is not developed. The test which fails initially is the actual aspect of TDD and that is how the test begins.

#### 2: Code to pass the test

When the automated test is failed, the programmer writes a code to pass the test. As the programmer works on the code an automated test can be executed to see if the code is working. The programmer gets feedback instantly on this which increases the productivity of the programmer. The developer works on the code until the test is passing. Thus, the developer can be confident on the program developed as it is passing and the code meets the test requirements.

#### 3: Refactor

The detail of refactoring is given below; once the test passes the code can be refactored as required. The base of the code should be cleaned regularly during TDD. The newly developed code can be moved from its convenience of passing the test to where it belongs. The code duplication should be removed. There are specific guidelines for creating clean code. Re-running through each phase of refactoring the programmer can be confident that the process does not redo any functionality.

#### 4: Repetition

When the program has the test passed and has a clean and refactored code, then code can be repeated. Starting with other new test function and proceed in the same manner. If the code which was developed newly do not satisfy the test or if the test fails, the programmer must undo the new code or return to immoderate troubleshooting. As the developer goes through all the functions, the application which is developed would be actively working. One of the main advantage is TDD is that as the programmers complete the full test with the test coverage. It is extremely uncommon that a programmer would backpedal to their code and write tests for everything after having a completely actualized application (wikipedia, 2017).

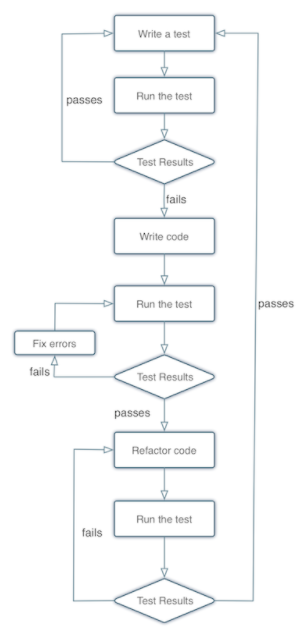
The flowchart given below describes about each phase of the TDD life cycle. At first the test is written and its run to check the test result. If the test passes the process will be repeated. If the test fails, write the code so that the test pass. When the code is written, and the code passes the test, the code will be refactored, and the test would be run to check if there are any further errors. Any further failures in the code should be refactored again. If the code passes, the process will be repeated.

Fig2: TDD Cycle (Barber, 2017)

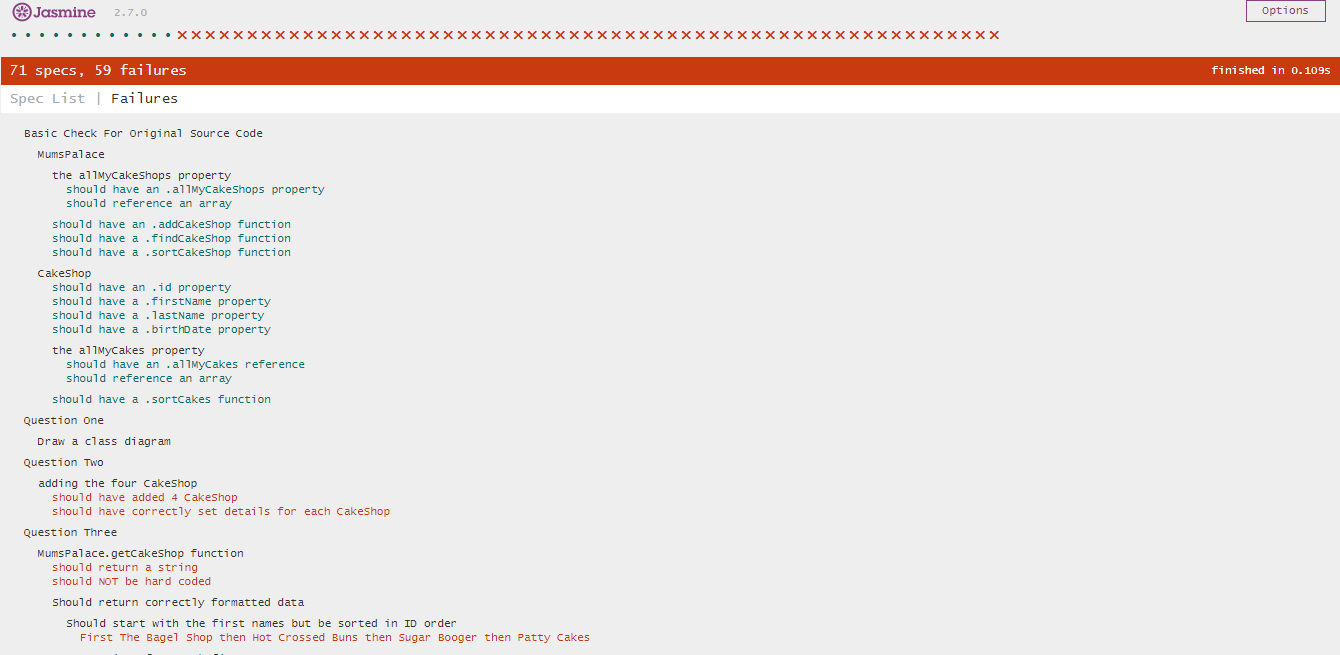
## Additional code Practice

When writing code for TDD, I usually prefer to go with a usual and an easy approach, by making a simple code it can be read and understood easily. I will also make sure that the software which I use for the code is relevant and is of latest version.

#### Example:

The example given below shows the after and before of the project.

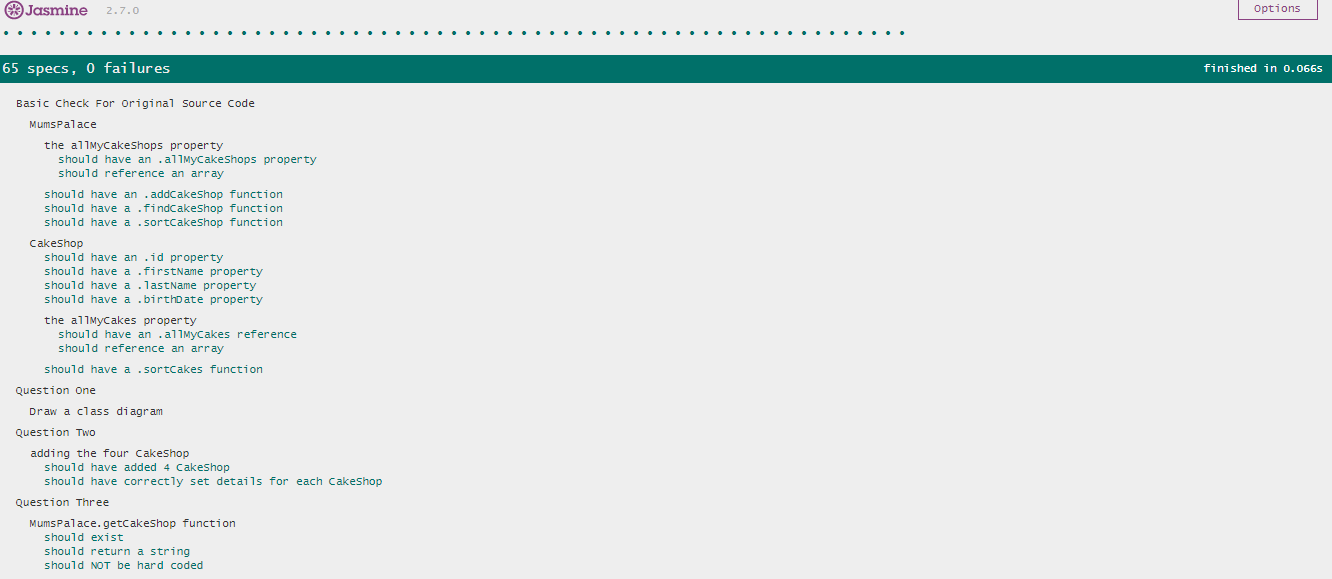
Before : The tests are written and when its executed the tests fail at first.



The above given example shows about the test which was written first and it failed. The test was written in Jasmine testing framework which is used in SE101 JavaScript Programming. The version of Jasmine used is 2.7 and which contains of lib, spec and src folders. The lib folder contains library files which is required for the program execution. Spec folder is used for writing tests. There were 7 scenarios which is edited into different test and src folder is used for writing the code which matches the test.

Seven scenarios are executed to fail at first and then the code is written to make then pass. Once the code is passed it is then refactored to make sure that the code is well structured and behaviour driven.

After: When all the test passes



Once all the tests pass spec runner is run to check if the test is passing. When all the tests are passing it is refactored again to check if there is any further modifications required and the process is repeated.

**TDD in Industry**

TDD is the powerful practice in the industry. TDD helps to deliver value quickly to the user while reducing the stress of developing the program and its release to the production environment.

In industry when TDD is being coded it is stopped now and then to check if the code can be released safely. Most of the time that can be done even if it is unfinished. TDD pattern is not red, green and refactoring it always. It may also be red, green and release at times. And after doing that there would be several refactor and release cycles in the process.

TDD can also be considered as a great tool for practising evolutionary design.

# Summary

TDD is an approach of development in which the developer writes a test that proves the piece of work being worked on, and then writes the code required to pass the test. Developer then refactors that code to eliminate duplication and overlap, then repeat until all the work is completed.

# Bibliography

*Agile Alliance*. (n.d.). Retrieved from www.agilealliance.org: https://www.agilealliance.org/glossary/tdd/#q=~(filters~(postType~(~'page~'post~'aa\_book~'aa\_event\_session~'aa\_experience\_report~'aa\_glossary~'aa\_research\_paper~'aa\_video)~tags~(~'tdd))~searchTerm~'~sort~false~sortDirection~'asc~page~1)

Barber, D. (2017, March 27). Retrieved from http://derekbarber.ca/blog: http://derekbarber.ca/blog/2012/03/27/why-test-driven-development/

*Extreme programming*. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Extreme\_programming

Grenning, J. (2016, May 11). *quora*. Retrieved from www.quora.com: https://www.quora.com/What-are-the-pros-and-cons-of-test-driven-development

Guru99. (2017). *Test Driven Development (TDD): Learn with Example*. Retrieved from test-driven-development.html: https://www.guru99.com/test-driven-development.html

*helpingtesters*. (2017, August 31). Retrieved from www.helpingtesters.com: http://www.helpingtesters.com/guide-to-test-driven-development-tdd/

Hill, S. (2015, February 23). *lean testing*. Retrieved from https://leantesting.com: https://leantesting.com/test-driven-development/

Rouse, M. (n.d.). *Tech Target*. Retrieved from searchsoftwarequality.techtarget.com: http://searchsoftwarequality.techtarget.com/definition/test-driven-development

*Test-driven\_development*. (n.d.). Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Test-driven\_development

*tutorialspoint*. (2017). Retrieved from www.tutorialspoint.com: https://www.tutorialspoint.com/software\_testing\_dictionary/test\_driven\_development.htm

*wikipedia*. (2017). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Test-driven\_development