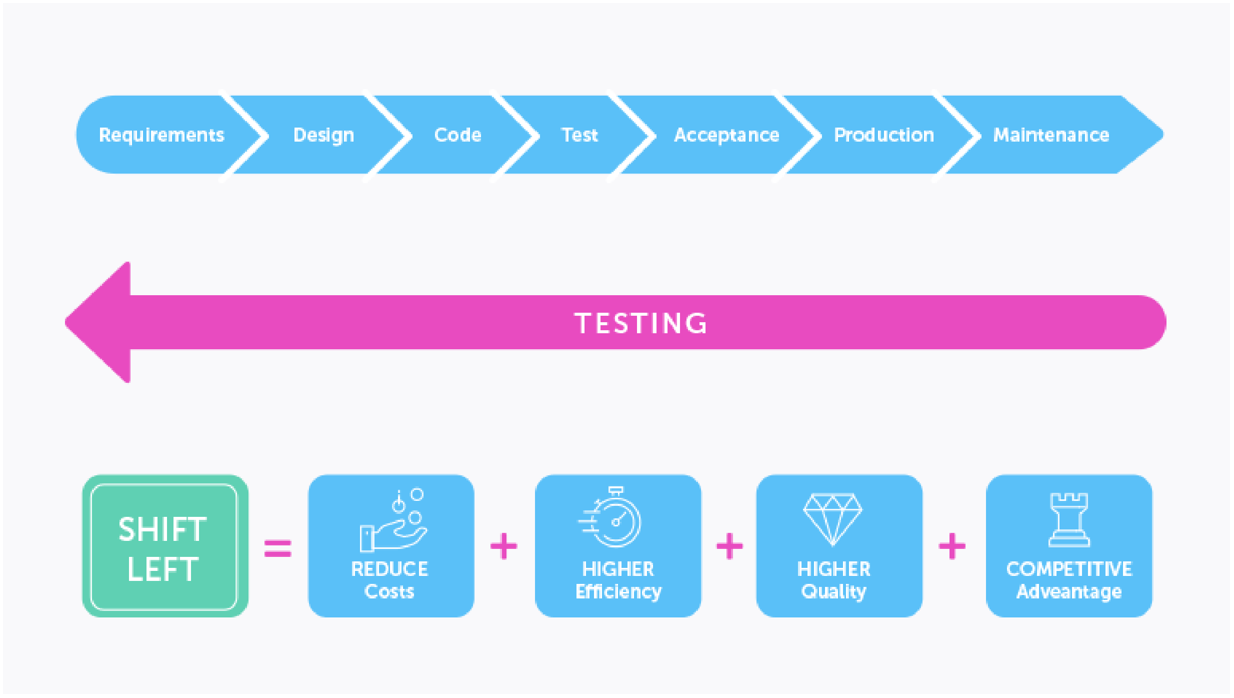
# Flutter In Action: TDD : Shifting LEFT with RIGHT approach

*Posted by Khushboo Uchat May 24, 2021*

I am back with one more blog as part of my CE learning journey. Here I will talk about my personal experience with TDD. If you are not aware of shift left approach then let me tell you in a brief. **Shift left approach means you are moving your testing one stage before than its usual place in software delivery life cycle.** The idea is to improve quality by moving tasks to the left as early in the lifecycle as possible. To enjoy the blog, I recommend you get yourself familiar with TDD basics. If not then I will still try to cover basics wherever possible.



## What is TDD :

Lets not be bookish and have the exact definition. In a very simple words, here the test cases are written before development starts. Initially all the test cases fail and our goal is to make them success. It is also called Test First Development. Usually the test cases are automated as it is quite large in number for the whole application and these cases are called Unit Test Cases as it is created to test a particular isolated block of code known as Unit. It has three phases which is self explanatory

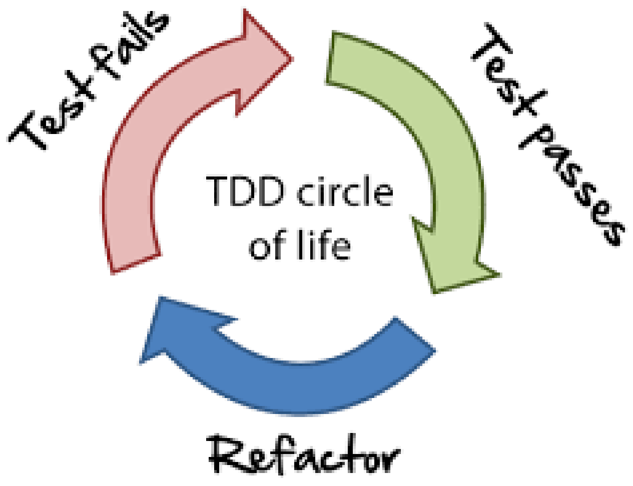
— Write a single unit test to verify that some criteria is met.

— Run the failing test (non compiling code counts as a failing test).

— Write just enough code so that the test passes.

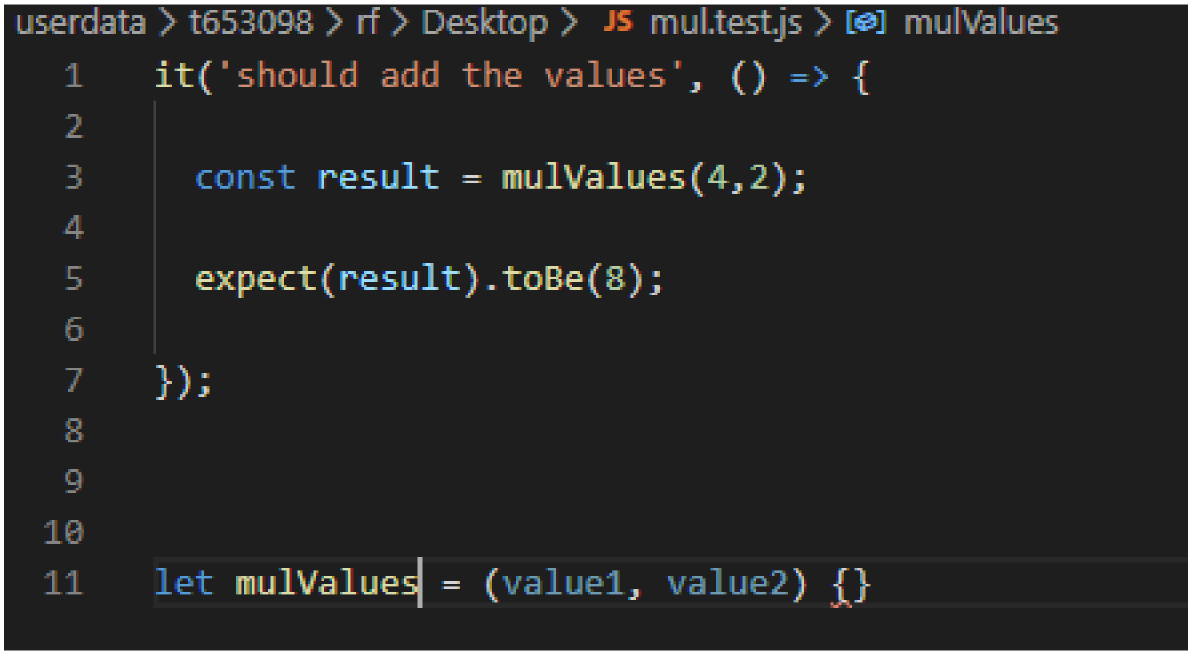
— Refactor the code making sure that the test still passes.

— Start again, incrementally testing and developing your application.



My experience with TDD :

I came to know TDD back in 2015 when I joined Deloitte. I joined Deloitte as Front End Engineer and my profile was to develop the application on Angular. I used Jasmine framework to write test cases which comes fully loaded when it comes to Angular CLI. Lets say I am writing a function to multiply numbers then before writing actual function, I write the test case in jasmine as below


userdata > t653098 > rf > Desktop > mul.test.js > mulValues

```
it('should add the values', () => {

  const result = mulValues(4,2);

  expect(result).toBe(8);

});



let mulValues = (value1, value2) {}
```

Now when I run test, it will fail. so we know that we have to implement the method to make it pass. So when we add the method mulValues it will pass. This example is the simplest example of TDD which one can follow. When I first write many test cases as part of TDD, it became the confidence booster and TDD became habit and I never looked back

After 2015, I was part of the projects where TDD is being followed from the beginning as well as we shifted from usual development method to TDD. I have seen the journey to adopt TDD and the challenges it imposes and learning curve for every team member. From my experience I conclude the below points.

- If you follow TDD as practice from beginning of the project then it empowers you with the advantages it brings to table. Once you start the development and try to move to TDD in between, it will not be that much effective and you have to juggle a lot between speed and quality.

- Do not blindly follow TDD without understanding the values it brings and whether it is useful for you or not otherwise you will end up writing lots of code just for simple test which might be done more easily by other ways.

- Do not always trust on 100% automation and work with balanced mix of Unit and Functional testing. Sometimes if there is a bug in test case itself then there needs to be a fallback to catch those errors.

- Do not compromise the design. This may sound silly but it has been observed that sometimes the focus is lost from ultimate goal of development and shifts to increase numbers of test cases and code coverage (that green bar infatuates a lot .. !!)

In UBS, when I joined Arena, we were only three with lots of development effort to meet the deadline. Due to situations, we started follow a simple Agile approach. Now our system is stable and to **focus more on detecting the bugs at early stage**, **I have suggested TDD approach to start with frontend development**. Though it might not be easy but will surely benefit in long run. I have also written a sample test cases in React to present an example in React. You can find at below.

https://devcloud.ubs.net/t653098/tdd-learning-shopping-cart

## Advantages of TDD :

- In TDD, you know exactly when to stop and you write just enough code to make the test pass. It will avoid lots of redundant code.
- One feature at a time can be taken as you are bifurcating your code into various tests.
- Modular code is being written as part of TDD because your test cases are already divided in small pieces
- Easy to debug the code written by TDD style.
- Bugs can be detected early in lifecycle which will save cost and time both as it can be expensive to fix them at later stage of development

### Disadvantage of TDD :

- It is not an replacement of manual testing completely when it comes to UI at least. The bugs those are introduced in test cant be detected (because ultimately it is all code)
- It will slow down the velocity of your team or the speed of the development as now for every feature you have to pass through the phases test, develop and refactor.
- Lots of code to maintain.
- It might not be easy for every team member to adopt TDD as it imposes lots of new challenges of its own with steep learning curve.

### Challenges of TDD and trade offs :

- Automated vs Manual Testing : From my experience with TDD, I have noticed that once the TDD is in place, people shift all their focus on automated test cases. We have to balance between functional and unit test cases as manual and automatic respectively as I mentioned for UI especially, it is very difficult to write all the automation in place.
- Test at the cost of Design : When you start writing lots of test cases as part of TDD, it is possible that you can go in wrong path which might not what the actual design is meant for. Writing lots of test cases give you confidence but you should always focus on design first.
- Test suites can grow larger and wild over time : When your application grows larger, your test suite also grows and maintaining it can be a task. You might be writing lots of code just to detect one line of change.

So basically when you decide to follow TDD, You should just not follow it because it is being talked about or it is something in trend. As per your project and requirements, it **might not be the best choice for you when**. For e.g.

- When your code is very small
- When you have shorter development cycle
- When your code is frequently changing

It is necessary to balance the trade offs of TDD with right approach and then you can shift left in your development cycle.
40 Views

Jerry Le Donne
Jun 9, 2021 2:19 PM
TDD may appear to slow you down but you pick up velocity by having less defect review meetings because you catch your defects earlier. TDD along with Software Engineers in Test (SET) have an impactful productivity boost as documented in the book "How Google Tests". This approach was industrialized by Microsoft.