

Why I Use Test Driven Development

Alex Thayn

What does TDD mean to me?

- My hope for using a TDD approach is that I am able to create more testable code which consequently makes my code easier to maintain and extend.
- It requires a shift in the development mindset.
- Focusing on tests first allows me to begin development with only the requirements in mind and ensure my software will have the correct behavior before I write even one line of production code.

Why I write tests

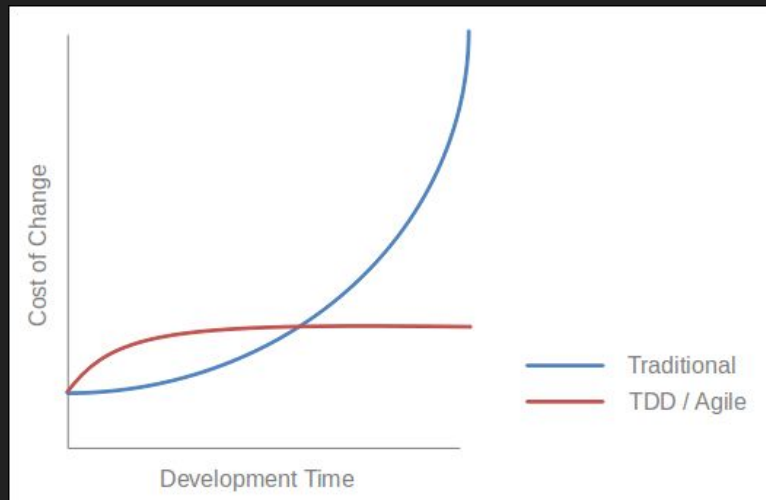
- It creates a short and continuous feedback loop
- More reliable code
- Increased confidence in code
- Saves time in the long run
- Allows me to focus on only one aspect of development at a time

Are there downsides?

- TDD has a huge up-front cost and slows down development
- There is more code to maintain, not only production code but all our tests have to be maintained
- Testing can be a challenge to learn and apply
- Early refactoring production code can sometimes lead to tests needing to be refactored






















Do the benefits outweigh the costs?

- Yes! Having a test suite prevents many defects and helps us save time and money in the long run.
- TDD helps clarify requirements which saves unnecessary rework.
- It also helps programmers understand their code more fully and provides a safety net when you need to refactor.



Examples in real life - Boggle

- During development of a Boggle game the tests I wrote the test saved my butt many times. I had a build pipeline setup and was alerted any time my tests failed.
- This continuous feedback allowed me to fix issues early and quickly because they were caught so soon.

Boggle					Edit	Queue	...
History		Analytics*					
Commit		Build #	Branch	Queued ↓	Duration		
 Merged PR 5: Added more game logic and did some refactoring CI build for alex.thayn1		20181201.5	 master	2018-11-30 - 22:35	6:41.650		
 Updated the scoring method and the units tests for that method CI build for alex.thayn1		20181201.4	 gameLogicImpleme...	2018-11-30 - 22:13	6:55.258		
 Finally got guesses to be added to the UI when the user submits one. R... CI build for alex.thayn1		20181201.3	 gameLogicImpleme...	2018-11-30 - 21:20	7:06.662		
 Fixed unit test that was missing a parameter CI build for alex.thayn1		20181201.2	 gameLogicImpleme...	2018-11-30 - 18:56	6:26.512		
 Refactored some view model logic, implemented more game logic suc... CI build for alex.thayn1		20181201.1	 gameLogicImpleme...	2018-11-30 - 18:45	6:41.034		
 removed unnecessary code CI build for alex.thayn1		20181130.9	 gameLogicImpleme...	2018-11-30 - 11:26	6:32.187		
 added some extra tests for the game logic CI build for alex.thayn1		20181130.8	 gameLogicImpleme...	2018-11-30 - 11:15	7:05.666		

Examples in real life - Boggle

I tested everything from game logic such as scoring to my data service which stored and returned database values.

Game Score Tests

```
[TestCase("bear", 1)]
[TestCase("BEAR", 1)]
[TestCase("home", 1)]
[TestCase("four", 1)]
[TestCase("can't", 0)]
[TestCase("1234", 0)]
[TestCase("----", 0)]
[TestCase("abcd", 0)]
public void Test4LetterScoreIsCalculatedCorrectly(string word, int expectedScore)
{
    BoggleGame.SubmitGuess(word);
    Assert.AreEqual(expectedScore, BoggleGame.Score);
}
```

Data Service Tests (using moq)

```
[SetUp]
public void Setup()
{
    dataServiceMock = new Mock<IDataService>();
    dataServiceMock.Setup(a => a.GetAllPlayers()).Returns(new List<Player>()
    {
        new Player(){Name = "Superman"}
    });

    dataServiceMock.Setup(a => a.GetAllGames()).Returns(new List<Game>()
    {
        new Game(){Score = 10294, PlayerId = 0}
    });
}

[Test]
public void TestDataServiceGetPlayers()
{
    var vm = new MainScreenViewModel(new MainViewModel(), dataServiceMock.Object);

    Assert.AreEqual(vm.Players[0].Name, "Superman");
}
```