Abstract

This is an article about TDD and TLD, and how they differ, and when to use one over the other.

1 Introduction

The objective of this article is to research the best approach to test code during development. The two different testing strategies covered in this article will be test driven development (TDD) and test last development (TLD). The article will try and explain the difference between the two, and through the perspective of multiple articles by different authors, and conclude on which strategy is the most efficient way to probably test your application code.

The article is build up of four sections:

- **Probelm definition 1.1:** This will define excatly what the probelm needed to solved is.
- **Hypothesis 2:** This will explain what the right answer might be.
- **Analysis 3**: This will be an analysis of the hypotese.
- **Conclusion 4**: This will a conlusion of the Analysis.

1.1 Problem definition

It is easy to develop an application from a list of speccifications, but it is a lot harder to make it bug free and stabel. To help avoid these problems, developers use the strong tool of testing code. But how should a program be tested. This articel will look at two different approaches:

- Test Driven Development (TDD)
- Last Test Development (LTD)

to try and find the anwser to which of the two strategies is best applicapel to development.

2 Hypothesis

This article goes into the Analysis with hypothesis, that TDD is the superior testing strategy compared to TLD.

The general philosophy in many articles and teaching institutes is that TDD is the right way to go, when testing code, compared to other strategies like TLD. The hypothesis is that TDD is way more effective at covering all the code, as the developer programs the test on the way to the finished product. Where TLD is very time consuming, it is often placed at the end of the development cycle. This makes it harder to account for the time needed for a full system testing. It is easier to forget certain functions and can result in a lot of technical debts. Where in the TDD approach, the developer team account for the testing when developing the function.

3 Analysis

3.1 TDD vs TLD - Pros and cons

Each of the two test strategies has their pros and cons. Some of the most common pros and cons can be seen in the tabel figure 1

TDD	
Pros	**Cons**
Code becomes modular	Takes a lot of extra coding
Decect bugs early	Can result in major refactorring later
Cheaper to fix	Require often reconfiguration of suites
Refactorring becomes easier	Hard to make well defined tests
Faster development	A lot of wasted time in constant changing code
Cleaner code	Steep learning curve

TLD	
Pros	**Cons**
Easy to do and understand	Expensive to maintaine
Code is ls less complex	Testing can become too wide
Does not require extra refractoring	Bugs are discovered late in development
Has a lot less code	No pre-testing of new function introducing new bugs

Table 1:

4 Conclusion

It is hard to say which method is the best for testing. According to Susan Hammond [1], it is naïve to claim that TDD is better, because it is not simple to define or measure if something is TDD.

The conclusion is therefore not going to point to TDD or to TLD as the best way to go. Ideally it is up to every individual team to decide from project to project, should we use TDD or TLD for this project. Understandably this will not really happen, the more realistic scenario will be, that the team will stick to one method, and use that exclusively. This might overtime prove just as good, considering that the team will become a lot better to that testing method.

jacob [2]

References

[1] Susan Hammond. Risk/reward analysis of test-driven development. https://etd.auburn.edu/bitstream/handle/10415/6890/Susan%20Hammond%20Dissertation.pdf?sequence=2&isAllowed=y.

[2] Christian Peters. When to use tdd and when not to use tdd. https://blog.ncrunch.net/post/when-not-to-use-tdd.aspx.