Test Driven Development

Alexander Olivero

Alexander.Olivero@milliman.com

March 1, 2019

Overview

Kent Beck

- Creator of Extreme Programming
- One of the 17 signatories of the Agile Manifesto
- Decades of experience in and consulting around software development

Definition

Test Driven Development (TDD) is a software development process that relies on the repetition of a very short development cycle. Requirements are turned into very specific test cases, then the software is improved to pass the new tests.

The Process

1. Quickly add a test
2. Run all tests and see the new one fail
3. Make a little change
4. Run all tests and see them all succeed
5. Refactor to remove duplication

Considerations

We all want to strive for what Robert Martin describes as “clean code that works.” With TDD we first solve the “that works” part of the problem, then we solve the “clean code” part.

You are writing tests for a reader, not just a computer. You need to answer “what was this programmer thinking about,” even when you are the programmer.

Helpful Hints

- Triangulate - only generalize code when there are two examples or more
- Leave old tests until you're confident that the changes encompass the logic being tested
- Never be more than one change away from a green bar
- Do not couple tests
- Leave the last test broken when ending a programming session

Constraints

Performance

We would like our tests to run as quickly as possible; namely, if we use similar objects in several tests, we would like to create them once for all tests.

Isolation

We would like the success or failure of one test to be irrelevant to other tests. If test share objects and one test changes the objects, following tests are likely to change their results.

Why use TDD?

"TDD is a way of managing fear during programming, so write tests until fear becomes boredom."

When stuck on big design ideas, work on something small. Even small progress compounds over time.

When you break a test on a team project, it means you didn't know enough to program what you just did.

Paradox: By not considering the future of your code, you make it much more likely to be adaptable in the future.

The Vicious Cycle

The more stress you feel, the less testing you will do.

The less testing you do, the more errors you will make.

The more errors you make, the more stress you will feel.

Bonus: Tips for Programming

Dave Ungar Shower Method

If you know what to type, then type. If you don't, take a shower and stay in the shower until you know what to type.

Taking Breaks

- scale of hours - have water at desk for biology to make you get up
- scale of a day - commitments after hours make you stop
- scale of a week - weekend plans get your conscious mind off work
- scale of a year - vacation helps reset completely