

Chapter 5: Test Driven Development



Tuesday, February 11, 2025

The Jury is In: TDD works!



How can you consider yourself to be a professional if you do not know that all your code works?



How can you know all your code works if you don't test it every time you make a change?



How can you test it every time you make a change if you don't have automated unit tests with very high coverage?

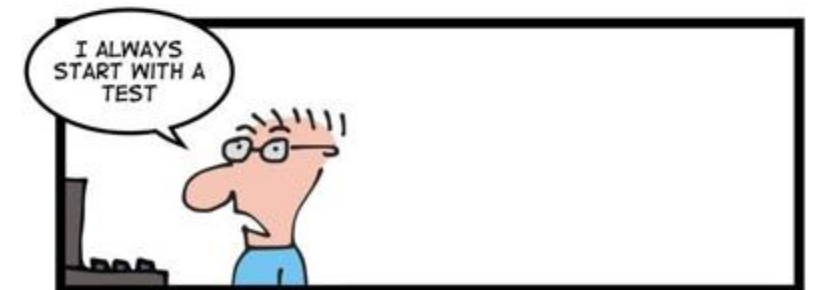
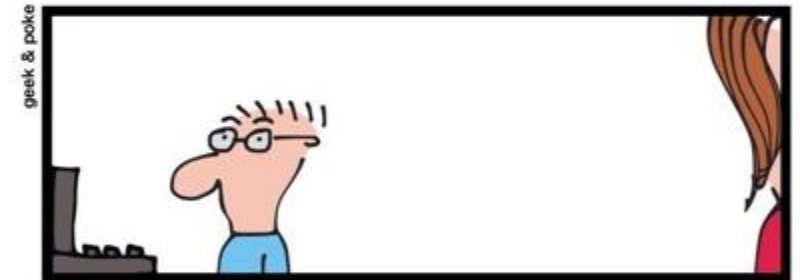


How can you get automated unit tests with very high coverage without practicing TDD?

The Three Laws of TDD

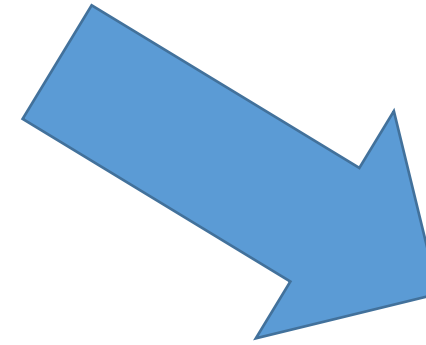
1. You are not allowed to write any production code until you have first written a failing unit test.
2. You are not allowed to write more of a unit test than is sufficient to fail—and not compiling is failing.
3. You are not allowed to write more production code that is sufficient to pass the currently failing unit test.

SIMPLY EXPLAINED

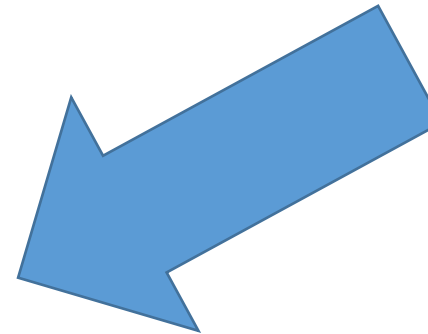


TDD

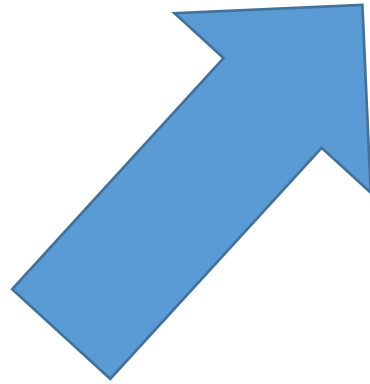
Portion of a
Unit Test



Unit Test
Fail to
compile



Write
Production
Code



The Litany of Benefits



Certainty

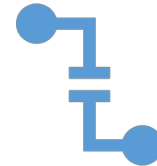
Run test any time you change the code

- Whenever you make a change you run the unit test



Defect Injection Rate

The number of defects that were discovered and reported during a particular integration of product development



Courage

When you have a suite of tests that you trust, then you lose all fear of making changes.

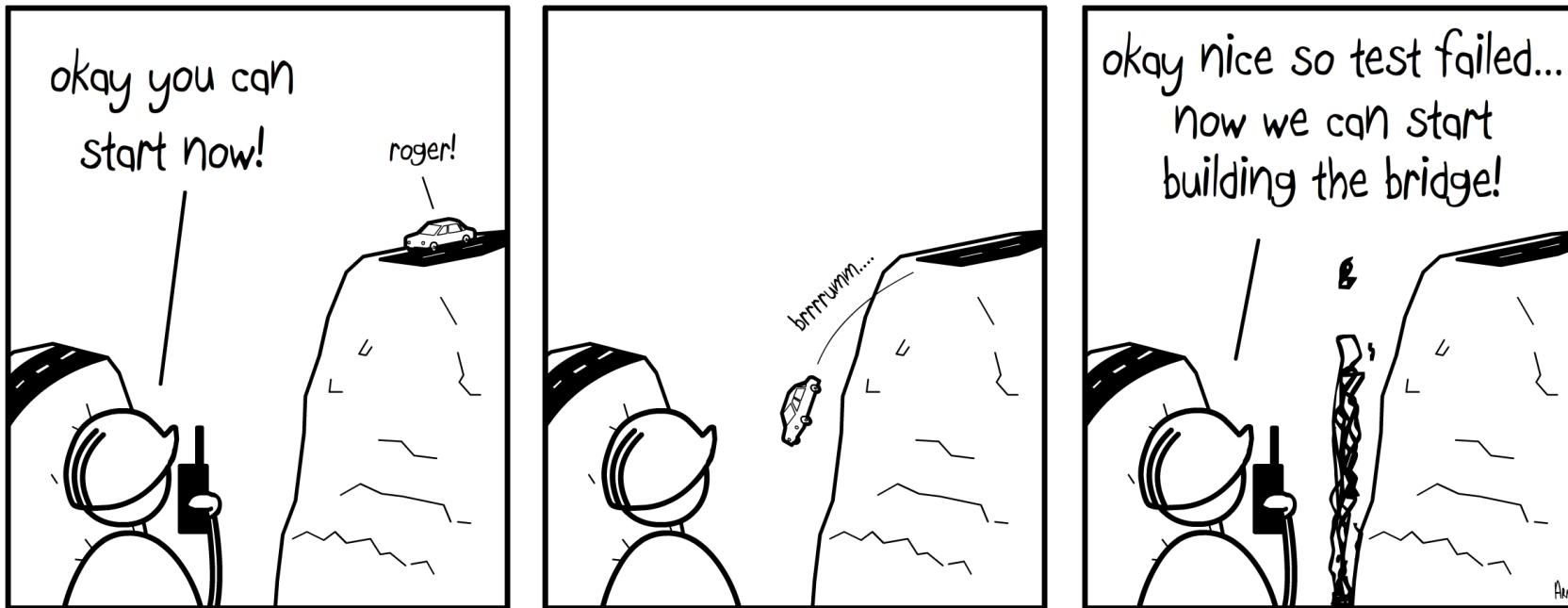
When you see bad code, you simply clean it on the spot.

The Litany of Benefits (con't)

- Documentation
 - The unit tests are documents.
 - They describe the lowest-level design of the system.
 - They are unambiguous, accurate, written in a language that the audience understands, and are so formal that they execute.
 - They are the best kind of low-level documentation that can exist.
- Design
 - Test first forces you to think about good design.
 - “But I can write my tests later” No , you can’t
- The Professional Option
 - Discipline that enhances certainty, courage, defect reduction, documentation, and design.

What TDD is Not



- Following the three laws does not guarantee any of these benefits.
 - You can still write bad code even if you write your tests first.
 - Indeed, you can write bad tests.




Chapter 5: Key Points

- Test Driven Development is a method that combines test-first development where you write a test before you write just enough production code.
- Three Laws of TDD:
 1. You are not allowed to write any production code until you have first written a failing unit test.
 2. You are not allowed to write more of a unit test than is sufficient to fail — and not compiling is failing.
 3. You are not allowed to write more production code that is sufficient to pass the currently failing unit test.
- TDD allows you to think about good design

Professional Tip of the Day #1: How to give constructive criticism about someone's code

- How do you tell someone their code is horrible?
 - common issues in code: excessive global variables, lack of modularity, poor naming conventions, inefficient algorithms, etc.
- Instead of just pointing out flaws, focus on **asking questions** and **providing helpful alternatives**.
-  Ask Thoughtful Questions
 - What made you decide to use a global variable here?
 - Have you considered breaking this function into smaller pieces?
-  Offer Constructive Suggestions
 - That is interesting. I usually do mine this way because [insert reason why you are better]
 - Does that way work? I usually [insert how you how you do it]

Professional Tip of the Day #1 (con't)

-  **Encourage Discussion**
 - *That's an interesting way to solve the problem! Here's how I approach it—what do you think?*
 - *Does your approach handle [specific edge case]? We could test it to see how it behaves.*
- **Key Takeaways:**
 - Phrase feedback as **curiosity, not criticism**
 - Offer **alternatives, not just problems**
 - Encourage **collaboration, not judgment**

Professional Tip of the Day #2: Introduction to Stock Compensation






- **Why This Matters for Software Engineers:**
 - 💡 "Many tech companies offer stock compensation—understanding these options helps you negotiate effectively."
 - 💡 "Stock options can increase your total compensation but come with risks and tax considerations."
- 💰 **How to Get Financial Guidance (Even if You're New to Investing)**
 - 💠 **University Career Centers** – Many offer free financial literacy workshops.
 - 💠 **Low-Cost Fiduciary Advisors** – Groups like **XY Planning Network** or **NAPFA** list **fee-only** financial planners who charge flat fees (not commissions).
 - 💠 **Online & Community-Based Resources** – Websites like **Bogleheads**, **r/personalfinance**, and **local nonprofit financial literacy programs** offer free guidance.
 - 💠 **Employer Resources** – Some companies offer **free financial advisory sessions** as part of their benefits.

Professional Tip of the Day #2: Introduction to Stock Compensation (con't)








- These are the most common stock options in tech. But there are other forms of equity to know about.

Type	What Is It	Key Features
RSUs	Shares granted as compensation	Free, taxed when vested
ESOs	The right to buy stock at a set price	Only profitable if stock rises
ESPP	Employee discount stock purchase plan	Buy shares at a discount

Professional Tip of the Day #2: Introduction to Stock Compensation (con't)

-  **Vesting Schedule:** When do you actually receive your shares? (Cliff vs. graded vesting)
-  **Tax Implications:** RSUs = taxed as income when vested; ESOs = taxed when exercised/sold
-  **Liquidity Risk:** Can you actually sell your shares? (Startups vs. public companies)
-  **Expiration Dates:** Some stock options expire if not exercised within a set period
-  **Stock is great, but you need to understand when you actually get paid and how much tax you owe.**

What Questions to Ask in a Job Offer?

- **Title:** *Negotiating Your Stock Compensation*
-  *Before accepting an offer, ask your employer:*
-  *What type of stock options am I receiving? (RSUs, ESOs, ESPP?)*
-  *What is the vesting schedule? (How long until I own the shares?)*
-  *What happens if I leave the company? (Do I lose unvested shares?)*
-  *What are the tax implications?*
-  *Can I negotiate for more stock?*
-  **"Stock compensation is negotiable! If the salary is fixed, ask for more RSUs."**