

Mohammadmahdi Farrokhy

Clean Code:

A Handbook of

Agile Software Craftsmanship

Chapter 09: Unit Tests

Test Driven Development

The importance of writing effective unit tests to ensure code **quality**, **reliability**, and **maintainability** cannot be overstated. Unit tests play a vital role in verifying the accurate functioning of individual code components and in early **detection of issues** during development.

Test code holds equal significance to production code and should maintain a high level of cleanliness. Clean tests eliminate the need for excessive documentation, as they inherently describe the scenarios pertaining to a class or even the entire software. This article delves into the topic of writing clean unit tests.

The Three Laws of TDD

Test Driven Development (TDD) is a practice where tests are written before the actual code.

TDD is guided by three laws:

- Begin by writing a failing test that cannot be compiled.
- Write just enough production code to make the failing test compile and pass logically.
- Develop the production code only to pass the test, without exceeding the requirements.

Test Driven Development

- Tests Must Be Clean.

- As the product evolves, tests evolve too.
- Dirty tests lead to challenges during modification.
- An unclear test requires additional time to encompass new code.
- Test code holds equal importance to product code.
- Test code is not a second class citizen; it demands careful **thought, design** and **attention**.
- Maintaining test code cleanliness is paramount, akin to treating product code with the same level of care.

Test Driven Development

- Tests Make You Capable.

- Unit tests empower your code by enhancing its flexibility, maintainability, and reusability.
- When you have tests in place, you gain the confidence to modify your code without fear.
- In the absence of tests, any modification becomes a potential bug.
- Regardless of your architecture's flexibility or the elegance of your design, the absence of tests leads to hesitation in making changes.
- The constant fear of hidden bugs hinders your willingness to evolve the code.

Clean Tests

The number 1 question is:

- What Keeps A Test Clean?

Readability, readability and readability. Readability might be more important for test code than product code.

The next question is:

- What Keeps A Test Readable?

What keeps product code readable. Clarity, simplicity and instructions density. You want to tell more by using less words.

Clean Tests

- One Test – One Assert

This rule might look a little much too strict, but it has a great benefit. Tests with only one assertion are fast and easy to understand. But we can be more flexible and change the rule into another:

- One Test – The Minimum Number Of Assertions Necessary.

Clean Tests

- **One Test – One Concept**

- Best practice is to test a single concept inside each test method.
- Avoid lengthy test methods that test multiple subjects consecutively.
- This principle aligns with the earlier guideline and results in test methods with minimal assertions.

Clean Tests

- F.I.R.S.T

Clean tests also follow five other guidelines:

- **Fast:** Test should run quickly.
- **Independent:** Test should not depend on other tests.
- **Repeatable:** Test should produce the same result consistently.
- **Self-Validating:** Test should have a boolean output.
- **Timely:** Test should be written before production code, not after it.