Test Driven Development

Driving the development using tests

- The stories in the sprint/iteration backlog should be relatively fine-grained. They shouldn’t be too large.
- Stories should be able to be implemented within a matter of hours to a days or so. The time take to implement a story should be a fraction of a sprint.

From user stories to tests

- The test cases provide a way to validate and verify and to know that we are done.

3Cs

Ron Jeffries talks about the 3Cs.

1. C → Card
    a. Index card, on which you write the user story.
    b. On the back of the index card, you might write the test you want to write, potentially.
2. C → Conversation
    a. This is what the team — the testers, the business analysts, the programmers — does.
3. C → Confirmation
    a. This is where the test comes in. On the back of the index card, you might write the test you want to write, potentially.
    b. The testers and the business analysts might want to write down the test based on the domain and the application being built.

INVEST

1. **I → Independent**
    a. The stories should be independent of each other.
2. **N → Negotiable**
    a. Stories should be negotiable.
    b. You can discuss and determine what the real feature should be.
3. **V → Valuable**
    a. The story should have a real business value to the application.
    b. If something is a backend process that you're trying to do to get the application up and going, don't bring it up as a story. Keep them as other tasks to be completed.
4. **E → Estimable**
    a. Stories shouldn't be too big.
5. **S → Small**
    a. They should be small in size.
6. **T → Testable**

Levels of Tests

- Unit tests are at the lowest level.
- You can functional tests, acceptance tests, UI tests, and so on (each increasing in the level).

Who drives the tests?

- Unit tests are predominantly driven by the programmers
- Functional tests and Acceptance tests are predominantly driven by business analysts and testers.
- Programmers have to support the business analysts and testers for them to be able to integrate and run these higher level and automated tests.

- Similarly, the testers have to supports the programmers when writing automated tests, so that when the programmers do evolve the code, fix errors, or enhance and add features, there can be a quick feedback loop that the code that worked before continues to work now.
- You can write the stories as executable documentations.
    - You can write them using tools such as R-spec, Cucumber, ECB, SpecFlow (.NET or C#), FIT (Framework for Integration Testing), FITNess, or Jasmine (for JS).
    - These are Behaviour Driven Development (BDD) tools, where you can write functional testing or acceptance tests.

Benefits of test driving

1. Regression
    a. You want to make sure that the code that was working before continues to work now.
2. Design
    a. You are able to influence your code's design through these test cases.
    b. "This code is not testable" ⇒ "This code and its design sucks!"
    c. The more we do TDD, it change the way we think, design, how we approach a problem. The design benefit we get becomes less.