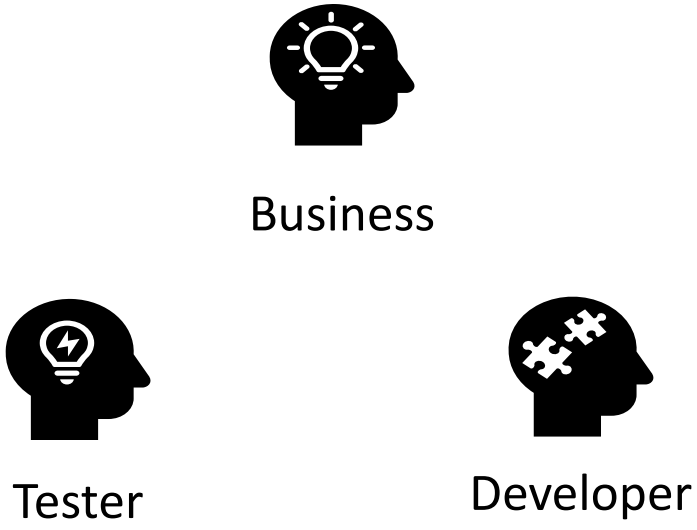
Behavior **D**riven **D**evelopment – Why ?

Introduction BDD & ATDD - Why

Definition : *« A collaborative process that creates a shared understanding of requirements between the business and the agile teams».*



Behavior Driven Development – How ?

3 Amigos / Refinement

Output

```
Feature: User login

  Scenario: Successful login with valid credentials
    Given the user is on the login page
    When the user enters a valid username and password
    Then the user should be redirected to the dashboard

  Scenario: Failed login with invalid credentials
    Given the user is on the login page
    When the user enters an invalid username and password
    Then the user should see an error message
```

Behavior Driven Development – Benefits

Higher confidence in the quality of the product

Increased individual productivity and motivation

Increased knowledge of the product domain

Reduced time to complete tasks

A lower number of production issues

The delivery process costs less money

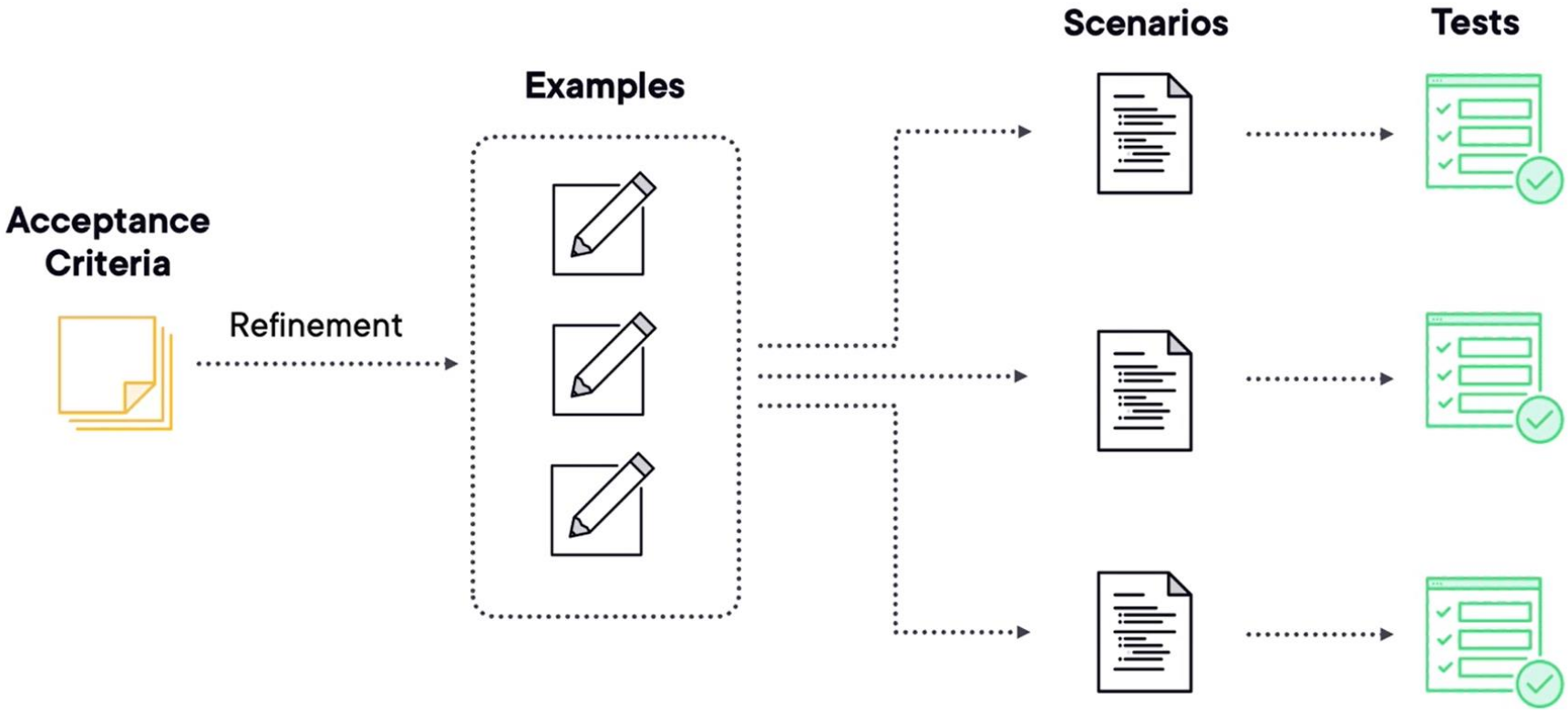
Accessible documentation

Easy onboarding process

Reduced dependency

Improved maintainability

Behavior Driven Development



Resources

- https://app.pluralsight.com/library/courses/behavior-driven-development-big-picture/table-of-contents
- https://cucumber.io/blog/bdd/example-mapping-introduction/
- https://capgemini.sharepoint.com/sites/ITpourtous/SitePages/S%C3%A9rie-sur-le-testing.aspx

Test Driven Development – Why ?

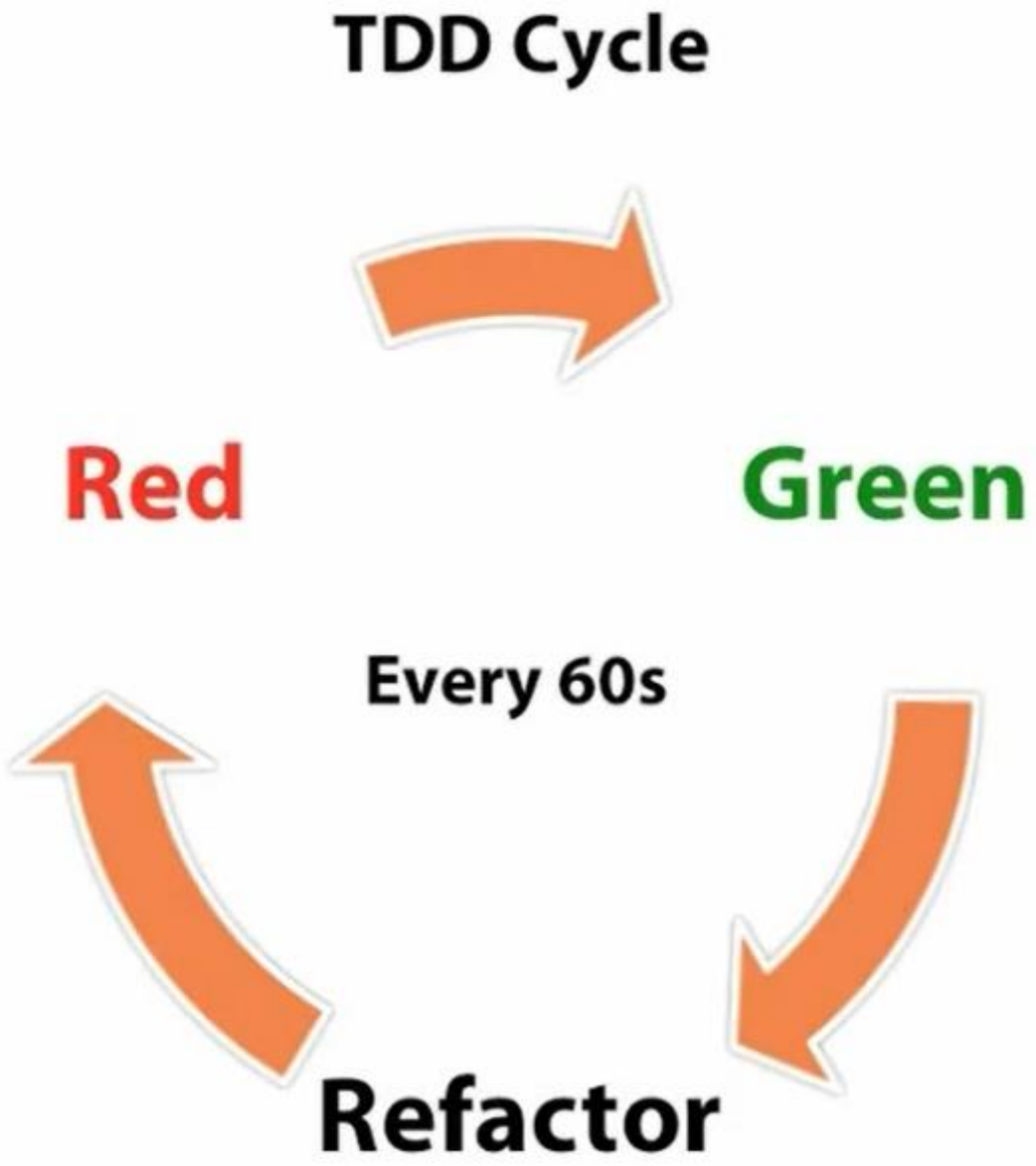
Definition : *« A software development practice that emphasis writing tests before writing the actual code »*.

- Code Coverage
- Less Technical Debt
- Prevent bugs
- Easier maintenance
- Documentation
- Refactoring with confidence
- Cleaner Code

TDD – How ?

- Write a failing test
- Make it pass
- Refactor

Small and concise (modularize), better design.



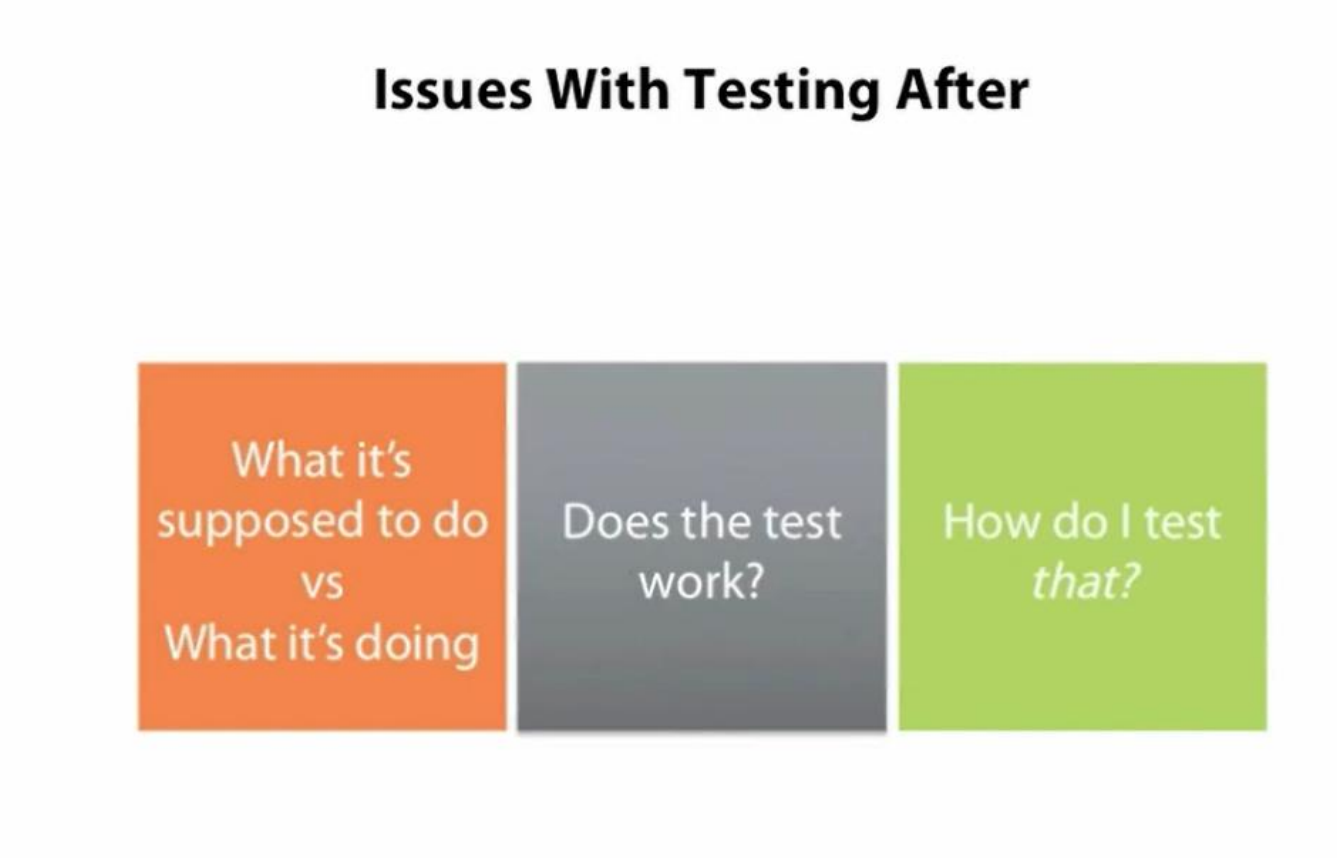
No TDD – Issues

What it's supposed to do vs What it's doing.

Your code becomes the specification for your test.

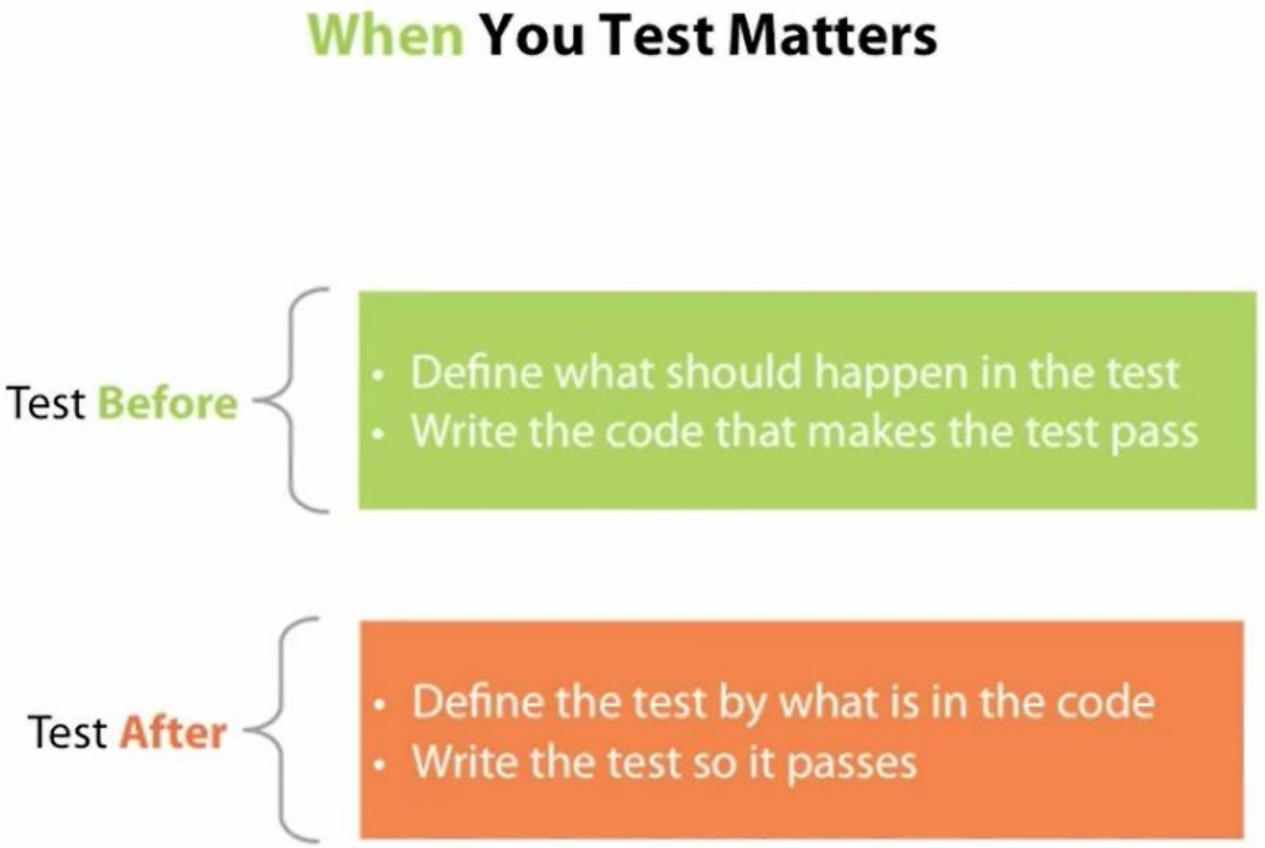
Does the tests work ? You will end up modifying your test and not your code.

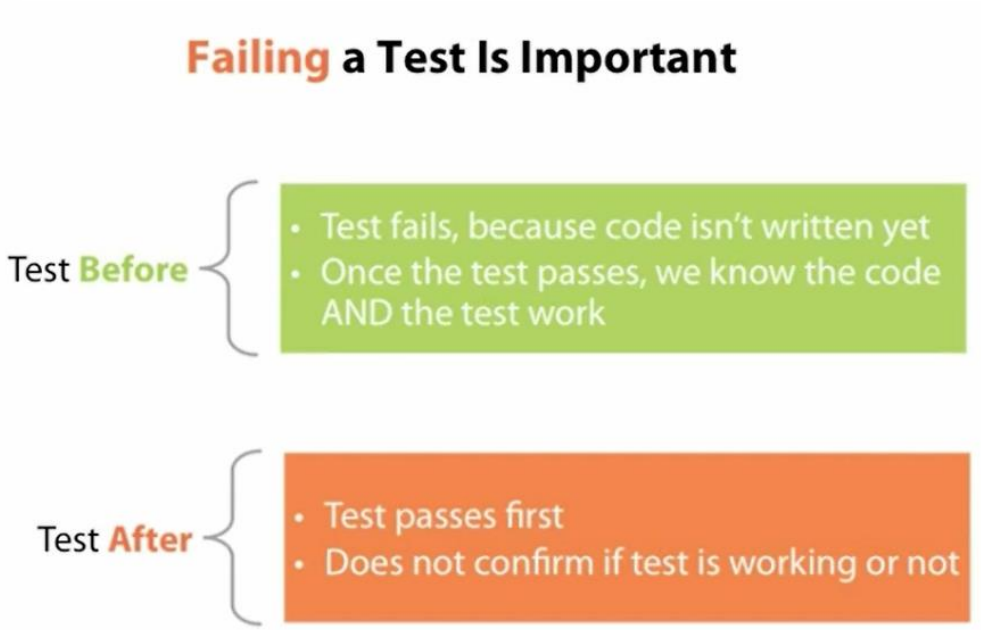
How do I test the code I've just written ?



TDD – As a Spec

The test is the codified version of the spec.





Test When Needed

Test Before

- Code is not written until the test already exists

Test After

- Code is written without thinking about testing
- Can be hard to simulate specific errors

Effects of Refactoring

Able to clean up code with **refactoring** as our 3rd step of TDD

Refactoring in **Test After** often leads to broken tests

Broken Tests lead to not refactoring or not writing tests

Recap

- **Writing tests after suffers from 3 ailments**
  - Supposed To vs Does
  - Is the test working?
  - Unsure **HOW** to test
- **Testing after reduces refactoring**