SNAPIT™

Work Based Project

Test Driven Development with Modular Software

Current Situation

Monolithic Software

Little Design before development

Minimal Testing

Decision made to move towards modular software with tests

Test Driven Development

Iterative development cycle, tests first then code

Makes software easy to change, maintain and understand

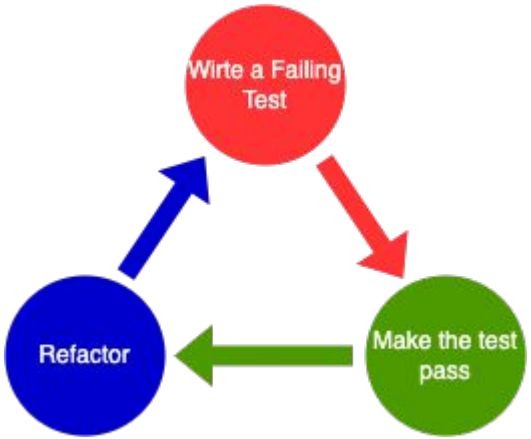
Prevent bugs in code

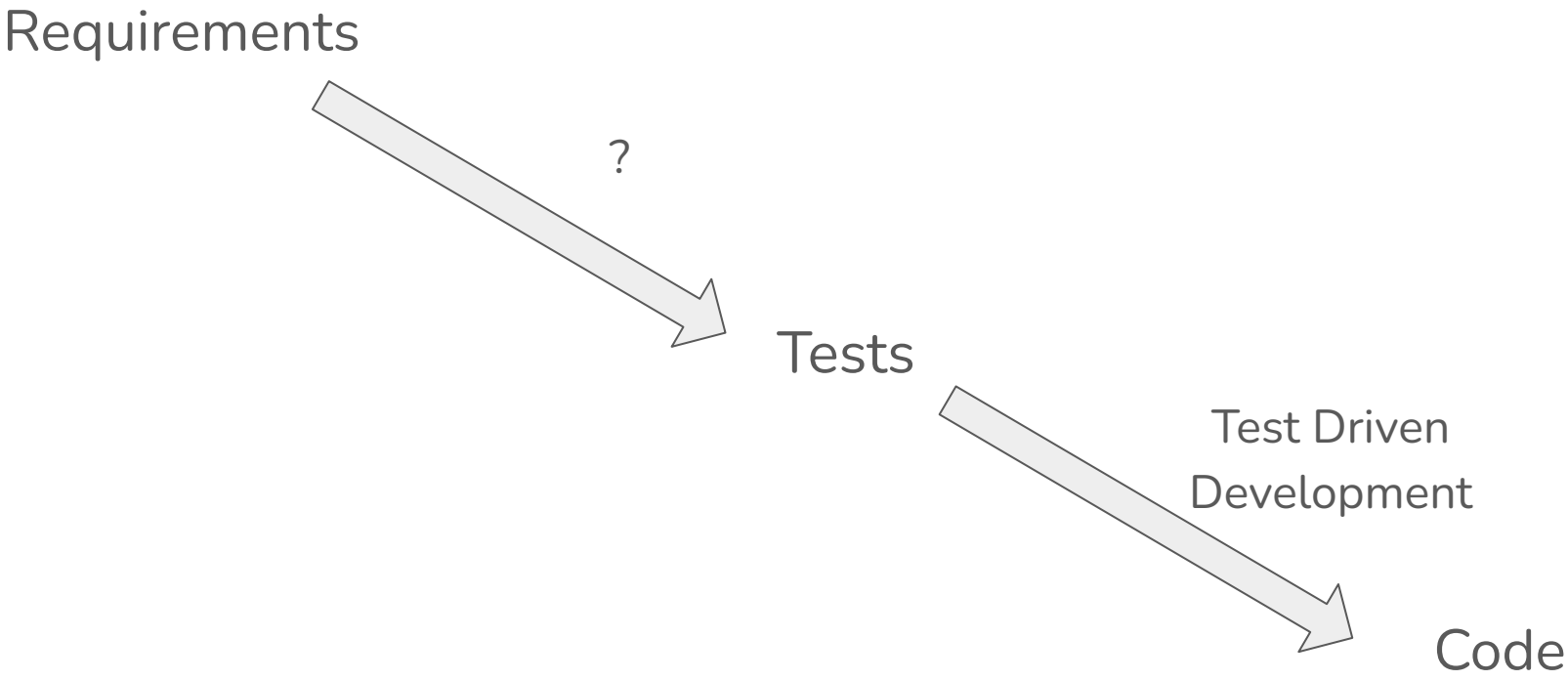
Test Driven Development Cycle

Red - Write a test

Green - Write just enough code to make the test pass

Refactor - Improve the quality of the code




Requirements
?
Tests
Test Driven
Development
Code

Behaviour Driven Development

Communication, Collaboration, Documentation

Shared Understanding of the System

System features are defined as Behaviours

Helps build the right system

Gherkin

Business readable, Domain Specific Language

Concrete Examples of how the system works

“Living” Documentation

Extension of user stories with Narrative

New people can understand the system by reading the features

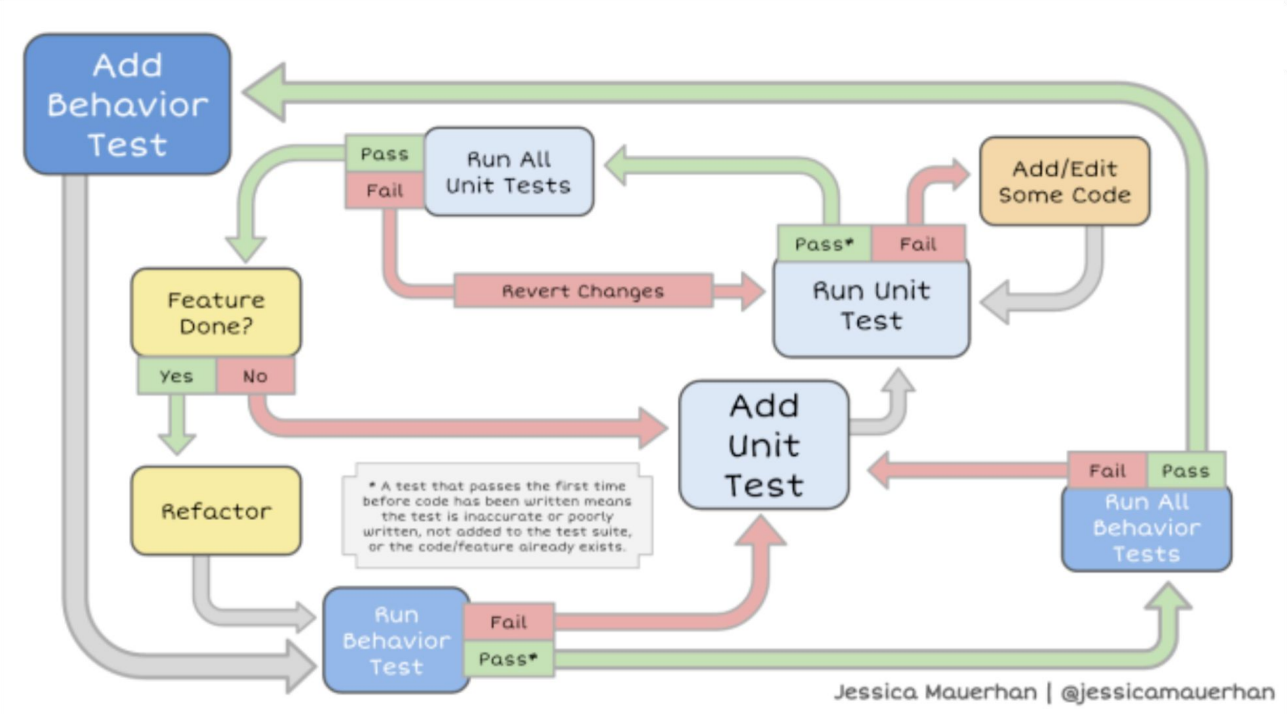
Feature: Requesting rides

In order to get to my destination on time
As a verified user
I want to request a ride

Rule: Unverified users cannot request rides
Rule: User cannot request ride for more than 6 people

Scenario: A user requests a ride for 4 people
Given I am a verified user
And there is a van with 6 seats nearby
And there is a car with 3 seats next to me
When I request a ride for 4 people
Then I should see that I am being connected to the van

Double Loop workflow



https://jmauerhan.wordpress.com/talks/double-loop-tdd-bdd-done-right/

Conclusion

Test Driven Development ensures that we build the system right

Behaviour Driven Development ensures that we build the right system

Documentation is valuable and is written as part of the development process