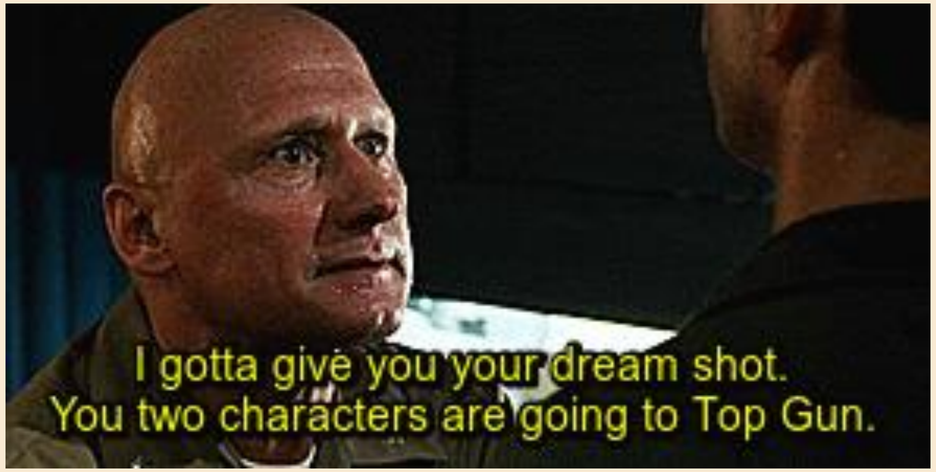
ALCOR ACADEMY TDD – FLYING

..Some of the key takeaways


I gotta give you your dream shot.
You two characters are going to Top Gun.

Source: https://imgur.com/gallery/2Z0x67t

Kristoffer Steen – 01.07.2022

bouvet

Connascence

- Two ore more elements are connascent if a change in one element require change in the others

https://media.giphy.com/media/lbldukokUp8lr6tH70/giphy.gif

- Coupling, Cohesion and Connascence highly influence each other

Connascence

- Measured in 3 dimensions
  - Degree – Size of it’s impact. Low is good
  - Locality – Are the entities that are connascent close to each other or far from each other? Close is good
  - Strength – How likely is it that you have to make compensating changes in connascent elements? Weak is good!

Connascence – The 10 categories by strength

10. Manual task (not in source code)
9. Identity (Depending on a specific instance)
8. Value (Eg. New Time(27, 15, 12))
7. Timing (..of when different calls return)
6. Execution order
5. Position (Eg. Order of args to method, array, etc)
4. Algorithm (Eg. Same algorithm several places)
3. Meaning / Convention (Eg. Magic strings, integers)
2. Type (Eg. Type of args to method)
1. Name (Eg. Name of args to method)

Discoverability: Dynamic

Have to run the code to see

Discoverability: Static

Only need to look at the code

Command vs Queries

- **Commands** change the state of something outisde the application, but does not return anything.

- **Queries** returns the state of something, but does not alter it’s state
  - Separating them makes the code more readable/comprehensible (higher cohesion)
  - It’s also easier to write and keep track of

Test Doubles – The three types

I used to call all of them «mocks», turns out it's not that simple

- Dummy: Usually for filling parameter lists.

- Stub - A test double for a Query
  - Instead of querying a real database
  - Pre-programmed into returning certian values based on certain input
  - A Fake is a «hand made» Stub

- Mock - A test double for a command
  - Instead of changing a real database or printing on real paper, we create a mock that accepts and terminates the call
  - And and the assert of your test, you verify that a call was made, and it was passed right arguments
  - A spy is a «hand made» Mock

- An object can be both a Mock and a Stub at the same time
  - And are usually set up using the same syntax when using a framework

Test Doubles – Benefits

- Not having to implement or worry about components far away from what we are creating / testing
- Not having to depend on internal or external systems
    - They might be offline, slow and their response might change from time to time.

[Source: https://makeagif.com/gif/crash-and-burn-huh-mav-QPNFHE](https://makeagif.com/gif/crash-and-burn-huh-mav-QPNFHE)

Test Types and Boundaries

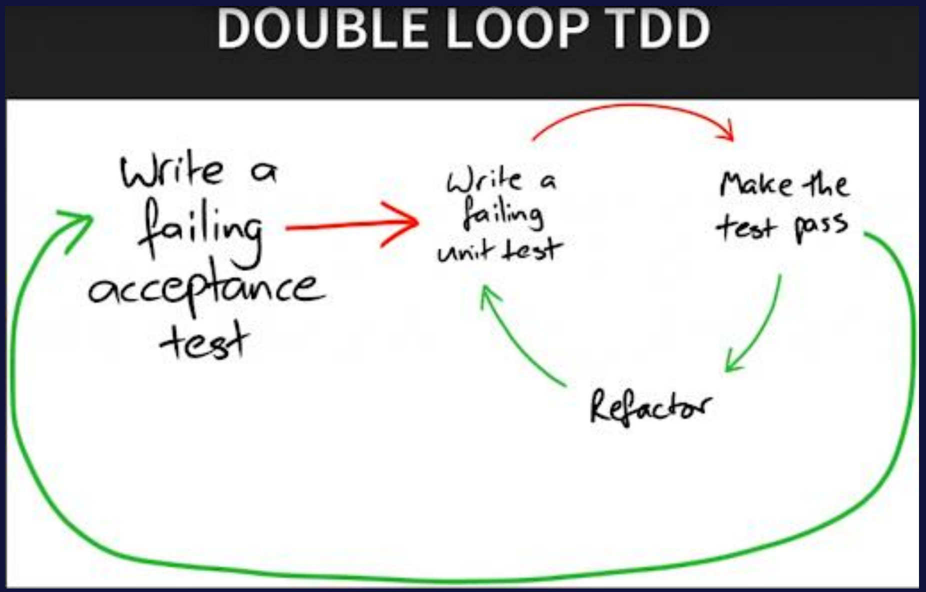
- End to end tests
  - Tests a complete flow of a requirement.
  - Includes the Views on one side and also slow (external) resources like databases and other APIs on the other side
  - Very valuable, but slow in execution so the amount must be limited
- Integrations Tests
  - Tests the part of the application that uses external systems
  - Can also be slow
- Unit Tests
  - Tests small and atomic behaviours
  - Very fast -> Can have a lot of these
- Acceptance Tests
  - Might look very similar to a unit test, but it covers more code/a bigger portion of the requirements
  - Slower than unit tests, but still much faster than End to End tests, due to external/slow systems being replaced by test doubles

https://media.giphy.com/media/26AHLNr8en8J3ovOo/giphy.gif

ATDD – Acceptance driven TDD

..Where we apply all that we have learned

- Write a failing acceptance tests that defines a use case
  - Remember that it's testing the public behaviour of one component, but will also force the implementation of other close component of that one
  - Components «further away» should be simulated using test doubles
- ..Then write multiple failing unit tests for the same module
  - These will test more specific (and edge) cases of that module
  - Important to advance slowly using baby steps
  - It's these unit tests that will drive forward the design of the components
  - In time, when all these tests are passing, the acceptance test will also pass



Source: Waldemar Mękal via https://medium.com/moonpigtech/

ATDD ctd.

..When writing the tests and implementation..

- Remember the outside in mindset
  - From high level responsibilities to low-level details
  - From the main goal to the steps to reach it
  - Follow the flow of dependency when
- First create a (subset of a) walking skeleton that compiles
  - Then add the needed code, each abstration layer at the time
  - Run the tests all the time and let that decide what the next task should be
    - Fix compilation errors before making tests run

Thanks, any questions?

https://gifs.com/gif/you-can-be-my-wingman-anytime-top-gun-1986-81loDW

kristoffer.steen@bouvet.no