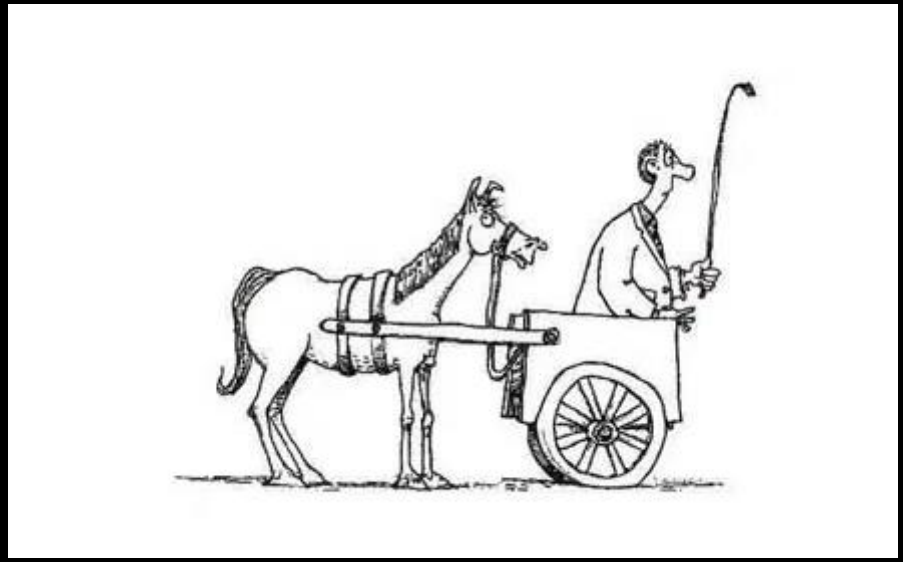
Testing Methodologies

Classic TDD, outside in & something ~~completely~~ slightly different

In the beginning

- No tests
- Manual testing
- Test in debug
- QA testing
- Testing as an afterthought


TEST DRIVEN
DEVELOPMENT

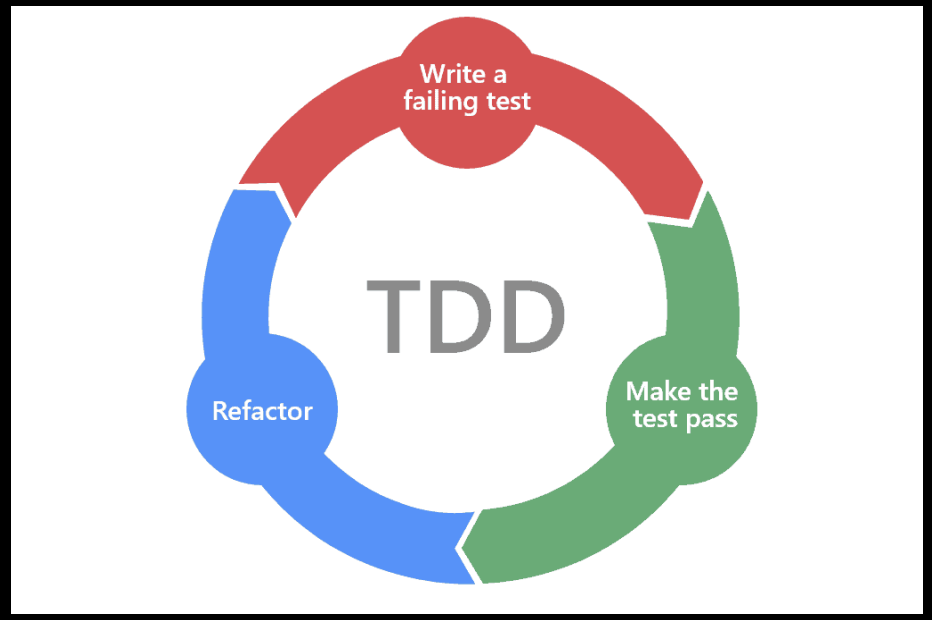


What is Classic TDD?

“Do the simplest thing that could possibly work”

Kent Beck

Red/Green/Refactor



- Write a failing test
- Make the test pass
- Refactor
- Rinse & repeat

Advantages of Classic TDD

Fail fast

Document as you go

Enable change

Disadvantages of Classic TDD

- Too technical
- Lose sight of business requirements
- Tests can become trivial and concentrate too much on edge cases

Outside-in TDD

- Acceptance Test Driven Development
- London school of TDD
- Mockist TDD

Acceptance Tests

- put the business requirements first
- seek to define the needs of the user
- are concerned with the system as a whole
- can be written in non-technical language

Gherkin

Given I have a dog

When she is happy

Then she wags her tail

Outside-In design

- Start on the outside
- Public interface
- High level behaviour
- Mock internal methods

Outside-In design

User needs

“my dog needs to be able to show her emotions”

Scenarios:

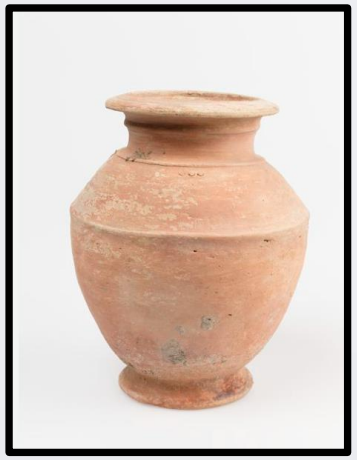
Given I Have a dog
When she is happy
Then she wags her tail

Given I have a dog
When she is scared
Then she growls

etc

The difference with this approach is that, rather than discovering the design through refactoring, in outside in development we will make design decisions about the public interfaces and use stubs and mocks to fake our implementation, before drilling down into the detail later.

Requirement: a vessel to carry things

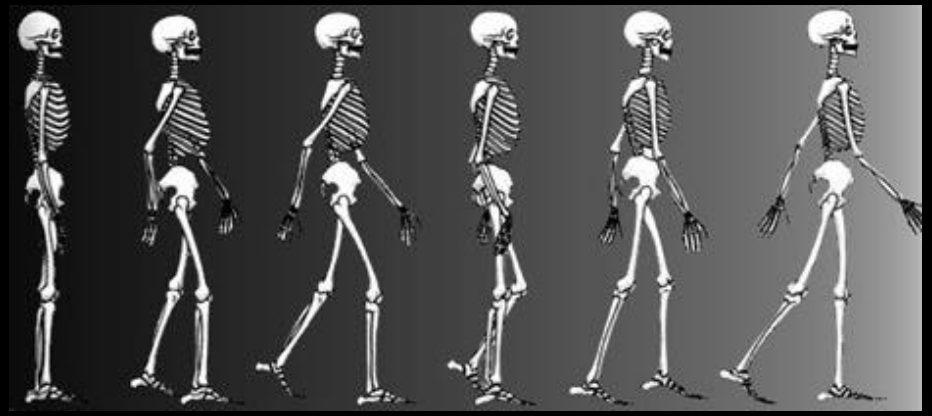


Delivery:

Walking Skeleton

"A Walking Skeleton is a tiny implementation of the system that performs a small end-to-end function. It need not use the final architecture, but it should link together the main architectural components. The architecture and the functionality can then evolve in parallel."

Alistair Cockburn



And now for something completely different

Characterization Tests

In computer programming, a characterization test is a means to describe the actual behaviour of an existing piece of software, and therefore protect existing behaviour of legacy code against unintended changes via automated testing

Wikipedia

When a system goes into production, in a way, it becomes its own specification. We need to know when we are changing existing behaviour regardless of whether we think it's right or not.

Michael Feathers

Case Study

- We have an upcoming audit to allow our product to be certified under the medical device regulations. We have to be able to demonstrate the product does what we say it does
- Current tests are slow, brittle and ambiguous.
- For instance:

```
… MonsterTestCase02()
```

Can anyone tell me what this does?

Our requirements:

- to produce human readable tests that anyone can understand
- to test all public methods
- to produce a living document of the tests and incorporate this into the build pipeline
- to work with fake, consistent data that is defined as part of the testing
- to leave the system under test unchanged, even when flaws are detected (for now)

Choices

- Specflow
- No mocking of data
- Instead use Entity Framework to populate empty database per test
- Arrange: what we put in the database
- Act: request some data
- Assert: that the correct data is returned

Progress

- 75% of public methods under test to date
- Good statement and branch level coverage
- Bugs documented
- Prepared to refactor legacy code and introduce new features with high level of confidence

In Summary

- Classic TDD starts on the inside and works out
- Design emerges through iterative refactoring
- Acceptance TDD starts on the outside and works in
- Design emerges by sketching out public interfaces and mocking implementation
- Characterisation tests confirm what does happen not what should happen

Questions?

Thank you

Andrew Benham
Andy.Benham@fdbhealth.com

References:

https://alcor.academy/
https://en.wikipedia.org/wiki/Characterization_test
https://michaelfeathers.silvrback.com/characterization-testing
Alistair Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams