Test Driven Development

Dr. Thomas Carroll
thomas.carroll@manchester.ac.uk

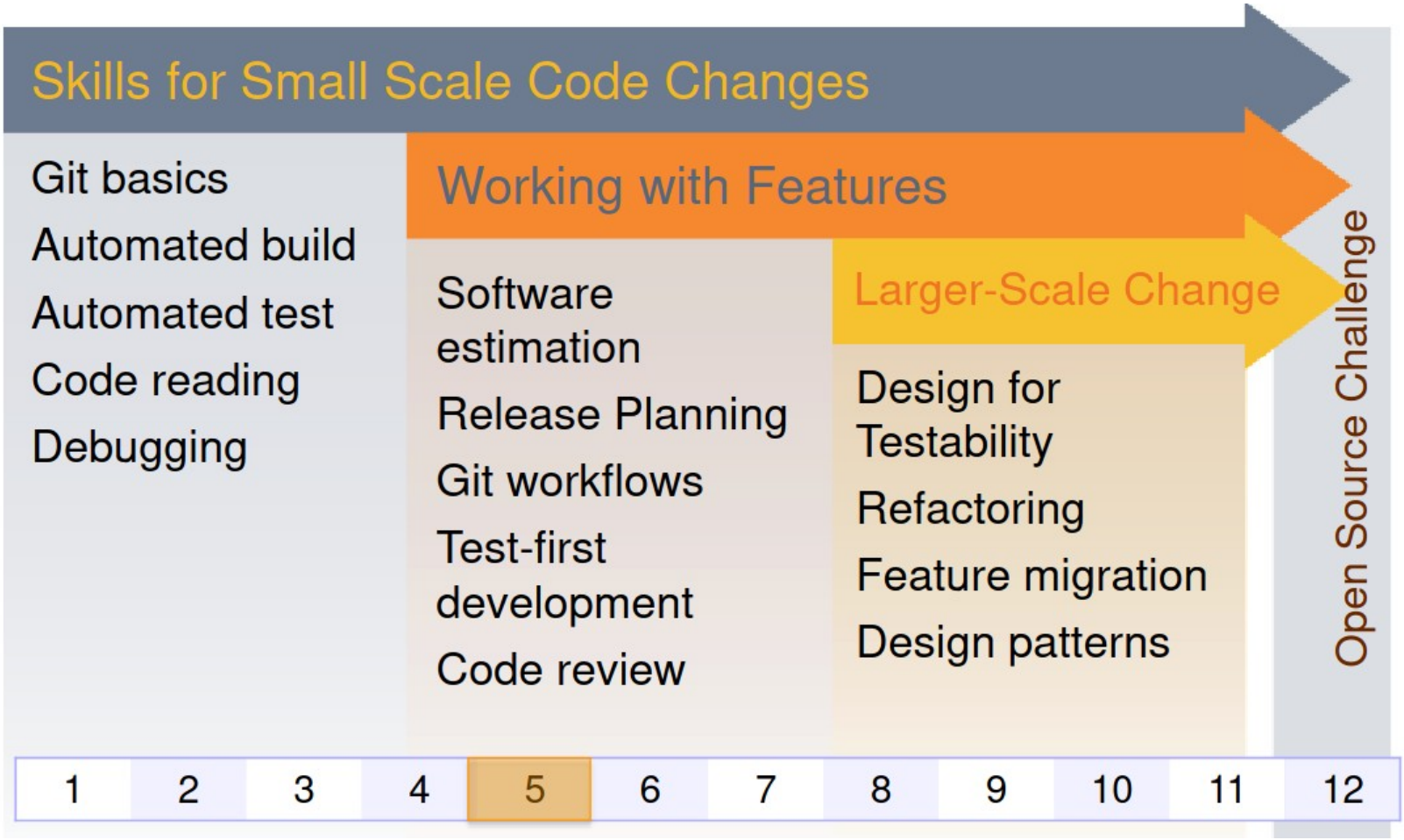
~~Test Driven Development~~
How to Keep Your Customer Happy and Keep Yourself (partially) Sane

Dr. Thomas Carroll
thomas.carroll@manchester.ac.uk


WHAT ARE YOU WORKING ON?
TRYING TO FIX THE PROBLEMS I CREATED WHEN I TRIED TO FIX THE PROBLEMS I CREATED WHEN I TRIED TO FIX THE PROBLEMS I CREATED WHEN...

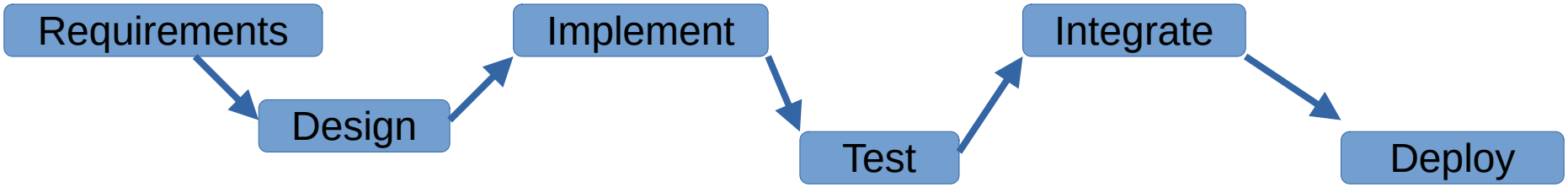
Learning Outcomes

- Understand concept of Test Driven Development
- Understand what the limits of TDD are
- Be able to Implement a basic TDD-led program


Skills for Small Scale Code Changes
Git basics
Automated build
Automated test
Code reading
Debugging
Working with Features
Software estimation
Release Planning
Git workflows
Test-first development
Code review
Larger-Scale Change
Design for Testability
Refactoring
Feature migration
Design patterns
Open Source Challenge
1
2
3
4
5
6
7
8
9
10
11
12

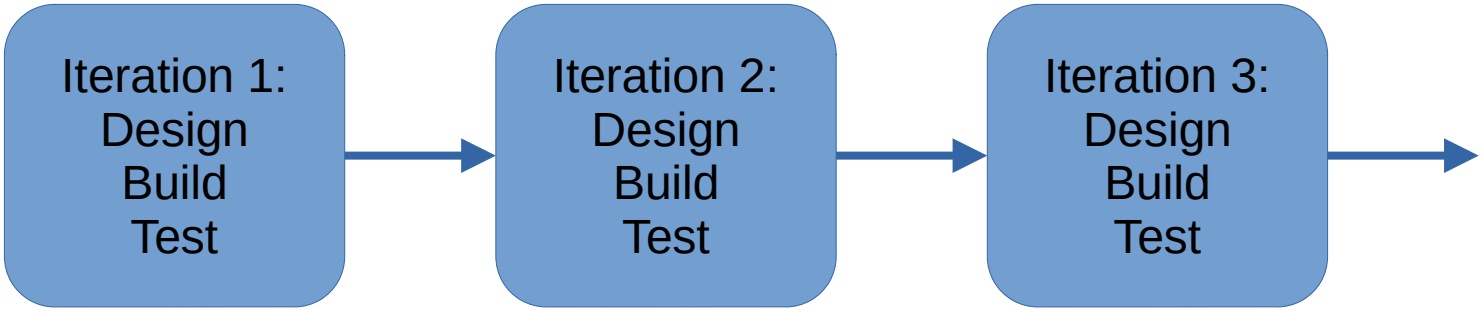
In The Beginning...

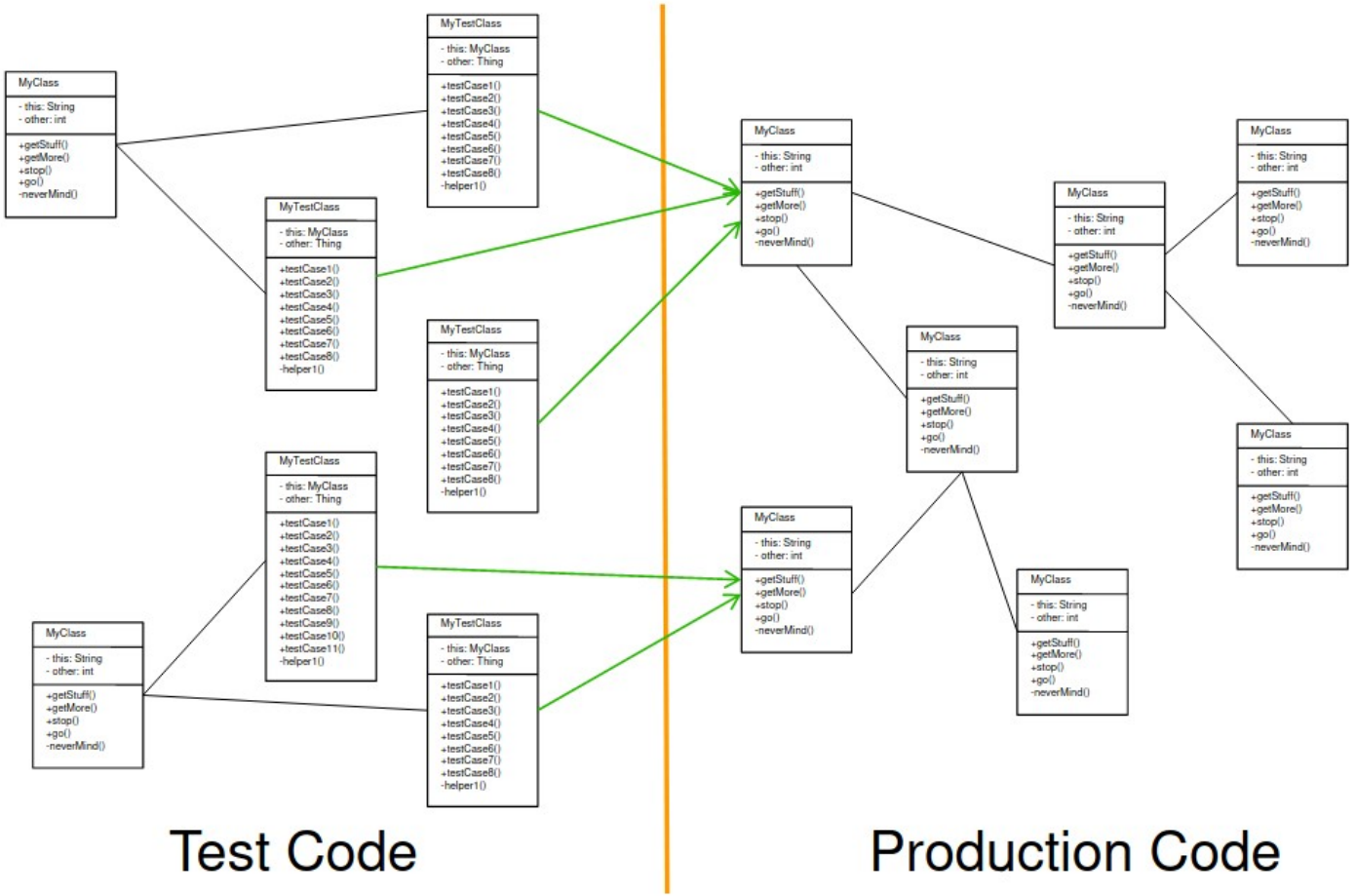
- Testing was performed as a separate task **after** code was developed
- Sometimes tests were designed before code was written, but testing was still left until later



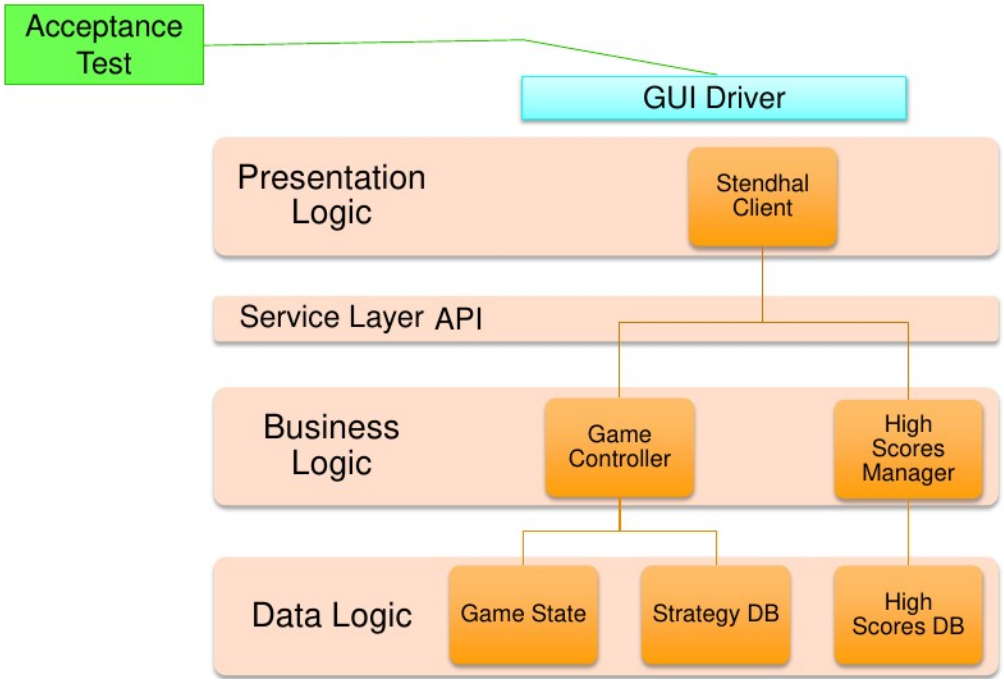
Agile Methodology

- Agile methods use:
  - Small, value driven increments
  - Testing inteleaved with requirements, design, and development
  - Fail fast

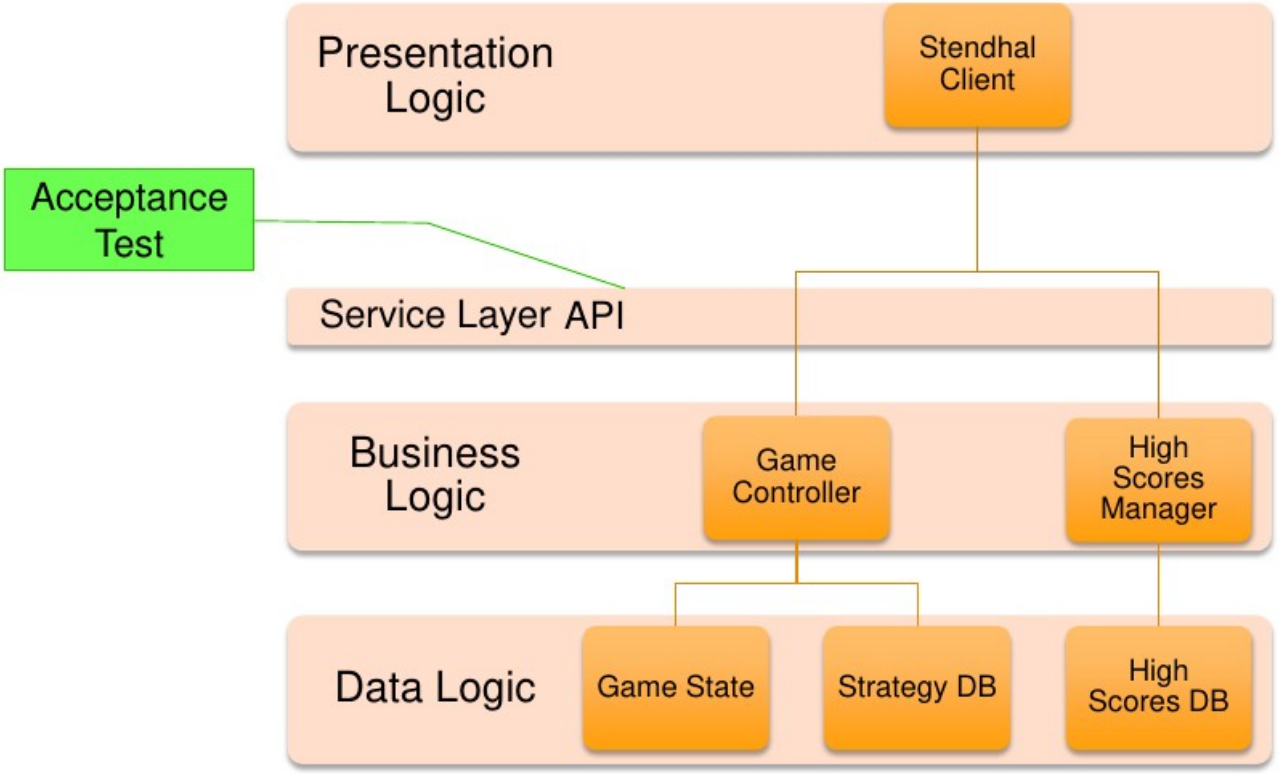



MyClass
- this: String
- other: int
+getStuff()
+getMore()
+stop()
+go()
-neverMind()
MyTestClass
- this: MyClass
- other: Thing
+testCase1()
+testCase2()
+testCase3()
+testCase4()
+testCase5()
+testCase6()
+testCase7()
+testCase8()
+testCase9()
+testCase10()
+testCase11()
-helper1()
Test Code
Production Code

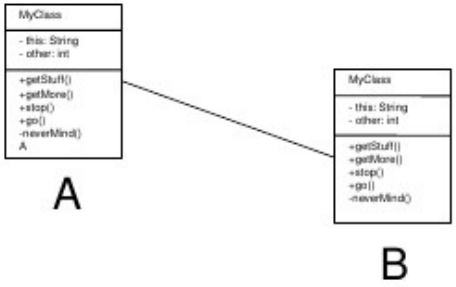
Outside-In Method



Middle-Out Method

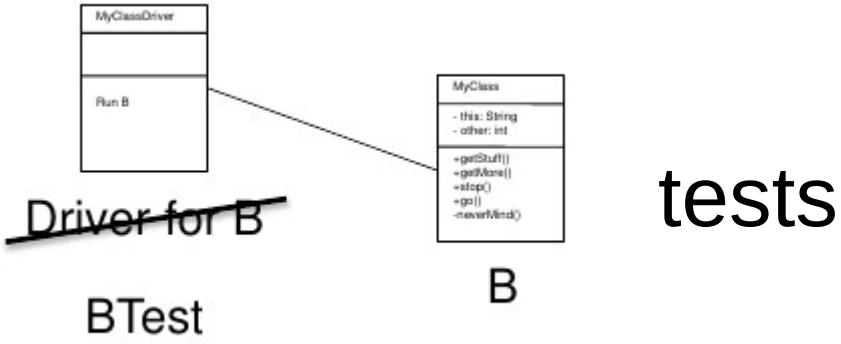


Drivers and Stubs



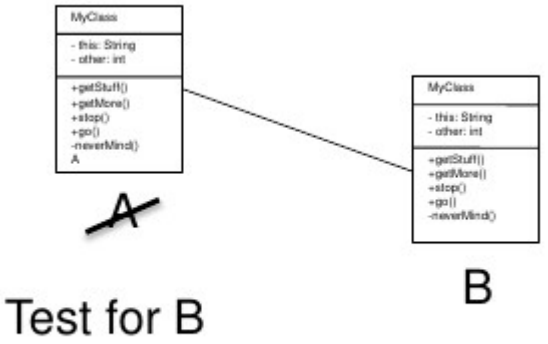
- A depends on B, which do you implement first?
- How do you test your work on A, whilst someone else works on B?

Testing Class B

- A isnt readv
- 

tests
- This is normal “Test After Development”

Testing Class B



- B isnt ready (or doesn't exist)
- We use a **stub** of Class B
- These tests will compile, but fail until we correctly implement B

TEST DRIVEN DEVELOPMENT

TDD Process

1. Decide Behaviour
- What behaviour should your program exhibit?

2. Write Tests
- Use “the perfect API” or “Wishful Thinking”
- Ensure Tests compile by creating any needed stubs. (IDE will help)

3. Run Tests
- Tests should FAIL initially
- Tests will start to PASS as you implement more

4. Write Code
- Make your tests pass as quickly as possible
-Make refinements until all tests pass

5. Refactor
- When all tests pass, you can refactor your program
- Does the design need improving?

Failing Tests

- Acceptance Tests:
  - Customer Centric
  - Defined Scenarios
- Unit Test:
  - Tests specific unit of code

- Failing unit tests are always a problem – they need to pass!
- Failing Acceptance tests are OK, if you have not yet implemented these parts