Test Driven Development

(by Chris Naidoo)

psybergate

Before we start…

- Explore audience experience with TDD
- Explore audience experience with Agile

psybergate

Objective

- To give an overview of TDD – introductory
- To demonstrate the basic steps in TDD
- To give you an opportunity to share your experiences with TDD – so that we can all learn
- You will not be an expert at the end of this talk
- Not about:
  - JUnit or Other Testing Frameworks
  - Test structuring
  - Mock objects
  - Test Data
  - Database testing

psybergate

What is Test Driven Development (TDD)?

- Well established technique – style of development / design – NOT a testing technique
- It is a rapid cycle of testing, coding and refactoring
- In TDD, the concept of programming and unit testing are not separate activities
- In TDD, we write a test first to drive the development process
- Strictly speaking, in TDD, we only write new code if an automated test has failed
- TDD Mantra: “Only ever write code to fix a failing test”
- Tests are first class artefacts

psybergate

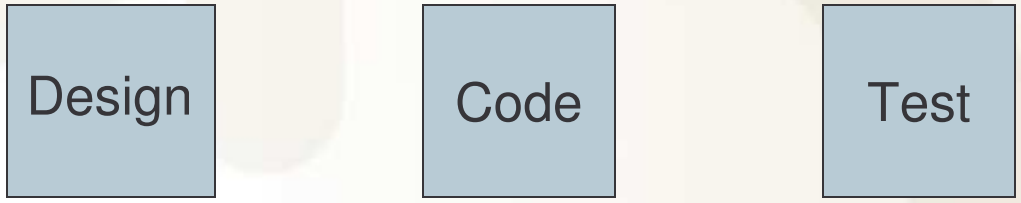
Benefits of TDD

- You to develop software better and faster (by building software reliably in small increments – 1 test at a time)
- You produce well-designed, well-tested, and well-factored code in small verifiable steps
- You get fast feedback on the state of your code
- You feel more confident about the software you are producing / less fear
- Cleaner code that works
- You get a set of automated regression tests as part of the development process
- Tests are often better system documentation for your programmers – they represent the developers interpretation of the requirements.
- When you write a test, you are a writing a client to your code – this often results in better testable code, and better designed code

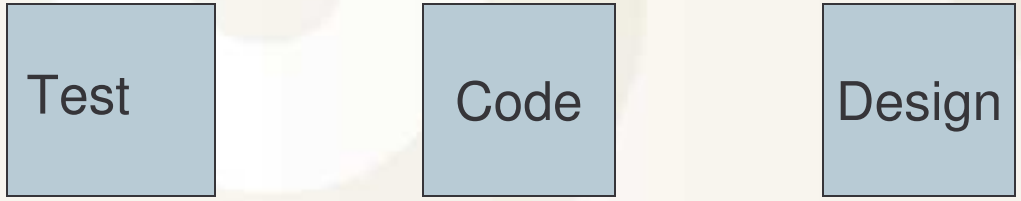
psybergate

TDD Development Cycle

Traditional Development Cycle



Test-Driven Development Cycle



psybergate

Agile Values and TDD

- Simplicity
- Communication
- Feedback
- Courage
- Respect

psybergate

Agile Practices that TDD Supports

- Simple/Incremental Design
- Refactoring
- Collective ownership
- Continuous Integration
- 40 hr week / Energised Work
- Short/small releases
- Pair Programming

psybergate

NOTE…

You do not need to do Agile in order to do TDD

psybergate

TDD and Simple Design

- Do the simplest thing that could possibly work (YAGNI)
- The best design is the simplest design that runs all the test cases
- Code/tests must communicate everything you want to communicate
- The system (including tests) should contain no duplicate code

psybergate

Key Attributes of a TDD Programmer

- Discipline
- Belief
- Be able to think/design/develop in small steps
- Admire elegance of simplicity
- ?

psybergate

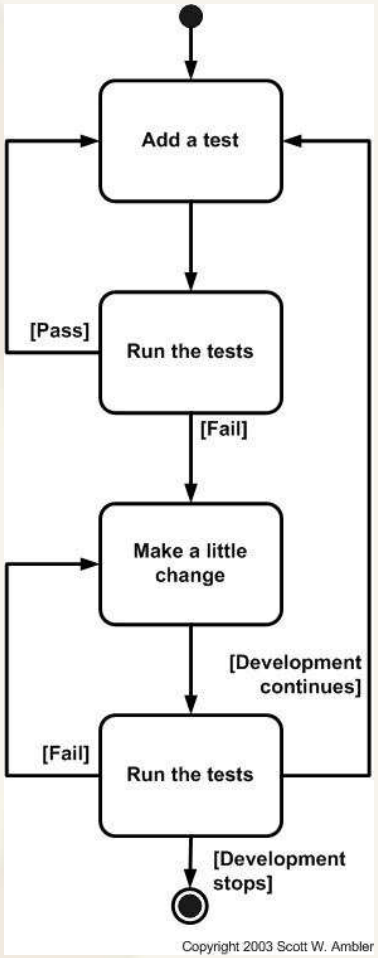
TDD LifeCycle - Overview

- *Think* of a simple, meaningful test
- Write a failing test (red)
- Make it run - fast (green) – do the simplest thing possible
- Make it good (refactor – implementation and tests) – refactor in small steps
- Run all tests to make sure nothing is broken
- Repeat until you can think of no more tests

Red / Green / Refactor

Failure is Progress !!

All Tests Run 100% of the Time



psybergate

Further Reading

- Extreme Programming Explained (Kent Beck) – 1st and 2nd Edition
- Test Driven Development (Kent Beck)
- http://www.testdriven.com (online community/forum for TDD)
- Test Driven : Practical TDD and Acceptance TDD for Java Developers – Lasse Koskela

psybergate

DEMO

- Simple Domain
- Keeping a list of tests to be written / things to do
- Writing the test first
- Programming by Intention – writing code as if another piece of code exists
- Red / green / refactor
- Small tests
- Quick feedback
- Patterns
- IDE Refactoring
- Real production system

- *Key question is :*
  - *“what do I want to test for”, rather than “how do I want to implement this”*

psybergate

Questions

?

psybergate