

Xtreme Programming FORUM

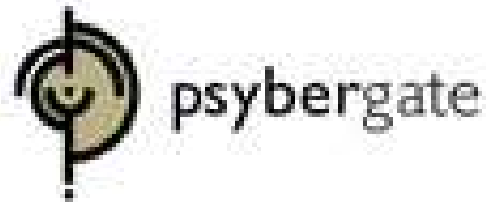


A practical view on the agile practice of test driven development

Presented by

Chris Naidoo, Psybergate

Sponsored by



Agenda

- Objectives
- What is TDD?
- TDD and XP
- The TDD LifeCycle
- Why TDD?
- General
- Demo



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
- Not about:
 - JUnit or Other Testing Frameworks
 - Test structuring
 - Mock objects
 - Test Data
 - Database testing



What is Test Driven Development (TDD)?

- TDD is a style of development / design
- 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
- It is not a testing technique, it is a development approach
- 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”

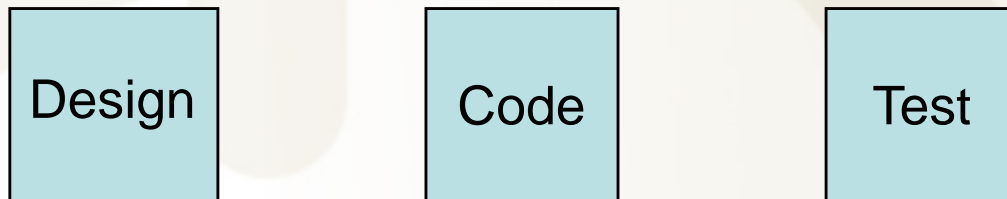
Objectives of TDD

- To enable you to develop software better and faster (by building software reliably in small increments – 1 test at a time)
- Manage fear as fear makes you :
 - Tentative
 - Communicate less
 - Shy away from feedback
 - Grumpy
- To produce clean code that works (opposite of architecture driven development in which the strategy is to make all the big decisions first)
- To get developers to produce a set of automated regression tests while they are developing 😊

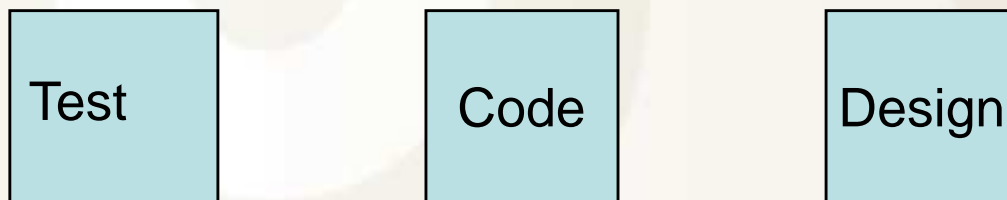


TDD Development Cycle

Traditional Development Cycle



Test-Driven Development Cycle



XP Values and TDD

- Simplicity
- Communication
- Feedback
- Courage
- Respect



XP Practices That TDD Supports

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



XP Practices That Support TDD

- Pair Programming

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



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 should contain no duplicate code



Key Attributes of a TDD Programmer

- Discipline
- Belief
- ?

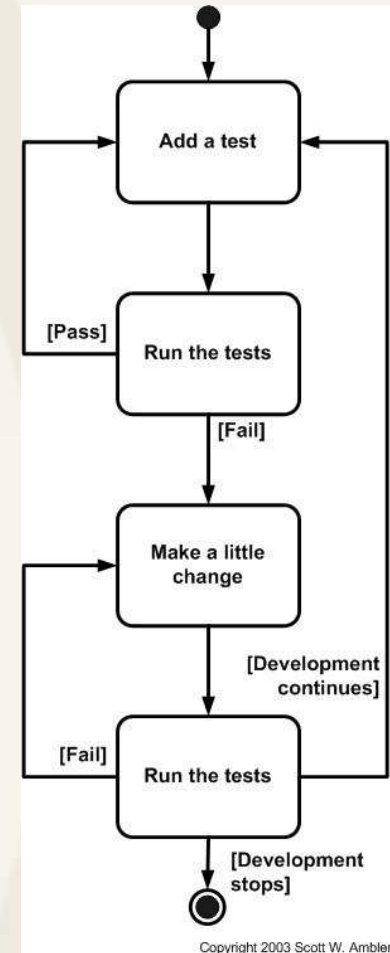
TDD LifeCycle - Overview

- Write a meaningful failing test (red)
- Make it run (green) – do the simplest thing possible
- Make it good (refactor – implementation and tests)
- Run all tests to make sure nothing is broken
- Repeat until we can think of no more tests

Red / Green / Refactor

Failure is Progress !!

All Tests Run 100% of the Time



psybergate

TDD LifeCycle – More Detail

- Think about what you want to do.
- Think about how to test it.
- Write a small test. Think about the desired API.
- Write just enough code to fail the test (RED)
- Write just enough code to pass the test (GREEN)
- Make sure ALL of your tests pass
- If you have any duplicate logic, or inexpressive code, refactor to remove duplication and increase expressiveness -- this includes reducing coupling and increasing cohesion.
- Run the tests again, you should still have the Green Bar. If you get the Red Bar, then you made a mistake in your refactoring. Fix it now and re-run.
- Repeat the steps above until you can't find any more tests that drive writing new code



Refactoring Techniques

- Create class
- Create method
- Create Interface
- Create interface implementation
- Extract method
- Extract local variable
- Extract object/field variable
- Renaming
- ...



Why TDD?

- It works with human nature (developers don't want to write tests, they want to develop)
- It prevents/manages “scope creep” – TDD helps us focus on what we need, rather than what we may need in the future
- We think about coding against an interface, rather than think about its implementation
- With TDD, we become a client of our code, and as such, design and usage of our code is improved. Often, code written without tests in mind ends up being highly coupled.
- We end up with automated tests and all the associated benefits of this
 - Increased confidence
 - Refactoring
 - Lower defect rate (enables us to discover mistakes earlier and quicker)
 - Tests communicate our functionality
 - Measure of system completeness

Difficulties in adopting TDD

- Culture in the organisation / team
- Discipline of writing the test first (pair programming helps)
- Being *able* to take small steps
- Designing for the future
- Seeing the beauty in simple solutions, and not striving for complex solutions
- You need a good IDE that supports refactoring



Who writes the tests?

- As TDD is a development approach, DEVELOPERS write the tests
 - Not a testing or a QA team
 - Not the customers (they write story tests)
- We need quick feedback, cannot rely on someone else writing our tests



When do we write tests?

- Before we write the code
- Every time we discover a bug, we write a test that isolates the problem
- Sometimes, we also write a test to illustrate and communicate an unusual circumstance



TDD Limitations

- Concurrency
- GUI's
- Frameworks
 - Open/Closed principle
- One has to ensure enough tests have been written to satisfy the requirements
- ?



How do you do TDD on existing code?

- Write tests “on demand”
 - When new functionality needs to be added to untested code, write tests for its current functionality first
 - When you fix a bug, write a test first
 - When you need to refactor, write a test/s first



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



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



Xtreme Programming FORUM



Discussion and questions

Sponsored by

