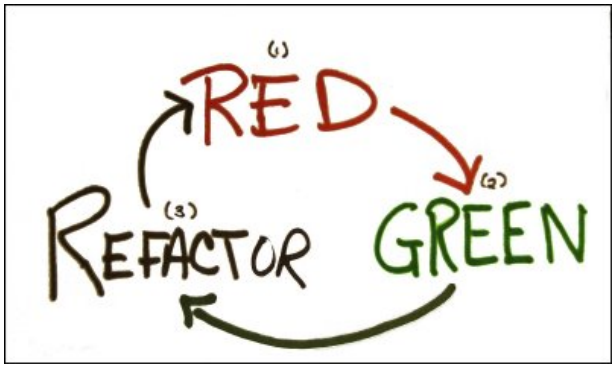
Test-Driven Development

Why, How and Strategies for Success

Gustav Boström
gustav.bostrom@crisp.se

Definition of TDD

- What is TDD?
  - Tests drive the design of the system
  - The process:
    - Failing test
    - Functioning test
    - Refactoring
    - Repeat
  - Tests are written first



Why TDD?

- Your safety net
- No tests -> Fear of change -> No Agility
- Fosters good design
- Saves resources
  - Manual testing
  - Less superfluous code (YAGNI)
- Gives you a good nights sleep before release!

Different kinds of tests

- Unit Tests
- Integration Tests
- Acceptance Tests

Unit tests

- Tests a specific component independently of it's environment
- Makes assumptions on the behaviour of surrounding components
- Technically oriented
- Runs fast
- Xunit frameworks
  - A xUnit-test is NOT automatically a unit test

"Never in the field of software development have so many owed so much to so few lines of code", Martin Fowler speaking of JUnit

Mocking and stubbing

- Short-circuit surrounding components and simulate behaviour
- Necessary to make proper unit tests
- Frameworks examples:
  - Mockito (Java)
  - Moq (.Net 3.5)

Mockito example

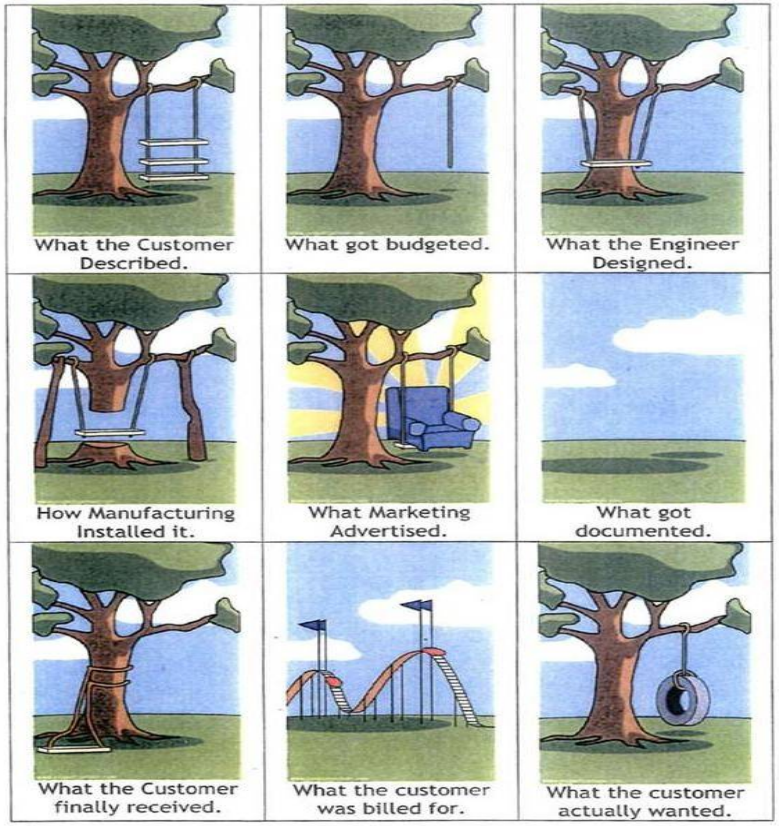

```
import static org.junit.Assert.*;
import org.junit.Test;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.verify;

public class TestForeignTransfer {
	@Test
	public void performForeignTransfer () {
		Account account = new Account();
		account.setBalance(1000);
		ForeignTransferFacade foreignTransferFacadeMock =
mock(ForeignTransferFacade.class);
		account.setForeignTransferFacade(foreignTransferFacadeMock);

		String ibanNumber = "CZ676676766767667";
		account.depositToIBANNumber(500,ibanNumber);
		assertEquals(500,account.getBalance());
		verify(foreignTransferFacadeMock).deposit(500, ibanNumber);
	}
}
```

Integration Tests

- Tests that test several integrated components together
- Example: Tests with calls to the database
- Often also written using a xUnit-framework
- Special frameworks available. E g: dbUnit
- OK if they are long-running

Acceptance Tests

- Tests expressed in a language your customer understands – Business oriented
- A Communication Tool first – Testing second
- Executable specifications
  - Concordion
  - FIT/FITnesse
  - Robot Framework
  - Cucumber



Source: Whitney Hess

Concordion example

Splitting Names

To help personalise our mailshots we want to have the first name and last name of the customer. Unfortunately the customer data that we are supplied only contains full names.

The system therefore attempts to break a supplied full name into its constituents by splitting the name around whitespace.

Examples

The full name John Smith will be broken into first name John and last name Smith.

Results generated by concordion™
in 110 ms on 19-Sep-2007 at 15:16:32 BST

Concordion Example – Making the specification executable

```
<div class="example">

        <h3>Example</h3>

        <p>
            The full name
            <span concordion:execute="#result = split(#TEXT)">John Smith</span>
            will be broken into first name
            <span concordion:assertEquals="#result.firstName">John</span>
            and last name
            <span concordion:assertEquals="#result.lastName">Smith</span>.
        </p>

    </div>
```

Concordion Example – Making the specification executable

```
package example;

import org.concordion.integration.junit4.ConcordionRunner;
import org.junit.runner.RunWith;

@RunWith(ConcordionRunner.class)
public class SplittingNamesTest {

    public Result split(String fullName) {
        Result result = new Result();
        String[] words = fullName.split(" ");
        result.firstName = words[0];
        result.lastName = words[1];
        return result;
    }

    class Result {
        public String firstName;
        public String lastName;
    }
}
```

TDD and Continuous Integration

- Test as soon as the code has changed
  - -> Quick feedback -> Less energy spent on fixing
  - Demands fast execution of tests
- Automate running of the test suite
- Run tests before check-in to SCM
- Red lamp when it fails for communication

USB- or Ethernet-controlled power sockets

Jenkins

TFS

Strategies for success

Strategy

Skills

Measure

Love

Create a Test Strategy - Case study

- Agile organization
  - 10 developers
  - No testers
  - 1 Product Owner
  - New production release every 2 weeks

Background

- A large refactoring demanded higher test coverage
  - Needed coverage for the courage to change
- Current test coverage around 20%

The System

- CRM
- Web application
  - Spring
  - Layered architecture
    - Web
    - Service
    - Data access layer
  - Integration with external systems

Test Strategy

- Strive for 100% test coverage of backend-code helped by unit tests and mocking framework
- Test each layer on it’s own
- Database layer tested with integration tests
- Critical User Stories tested using Acceptancetests expressed in text and tables
- User interface mainly tested by manual exploratory testing

Sample workflow

- Write Acceptance test
  - Product owner writes **text**
  - Developers make the test **executable**

Paralell

- Write Unit tests for backend components, mock subcomponents (Like the Database components)
- Write Integration tests for Database Components
- Check-in when tests are green
  - IntelliJ/TeamCity Remote Run does this automatically

Learn the skills - How to start

- Learn a Unit test framework
- Start with unit testing new code
- Pair program with those who know
- Automate in your Continuous Integration platform
- Write tests when making changes to old code (You will discover why there are benefits with Test First …)
- Improve communication with customers through Executable Specification tests

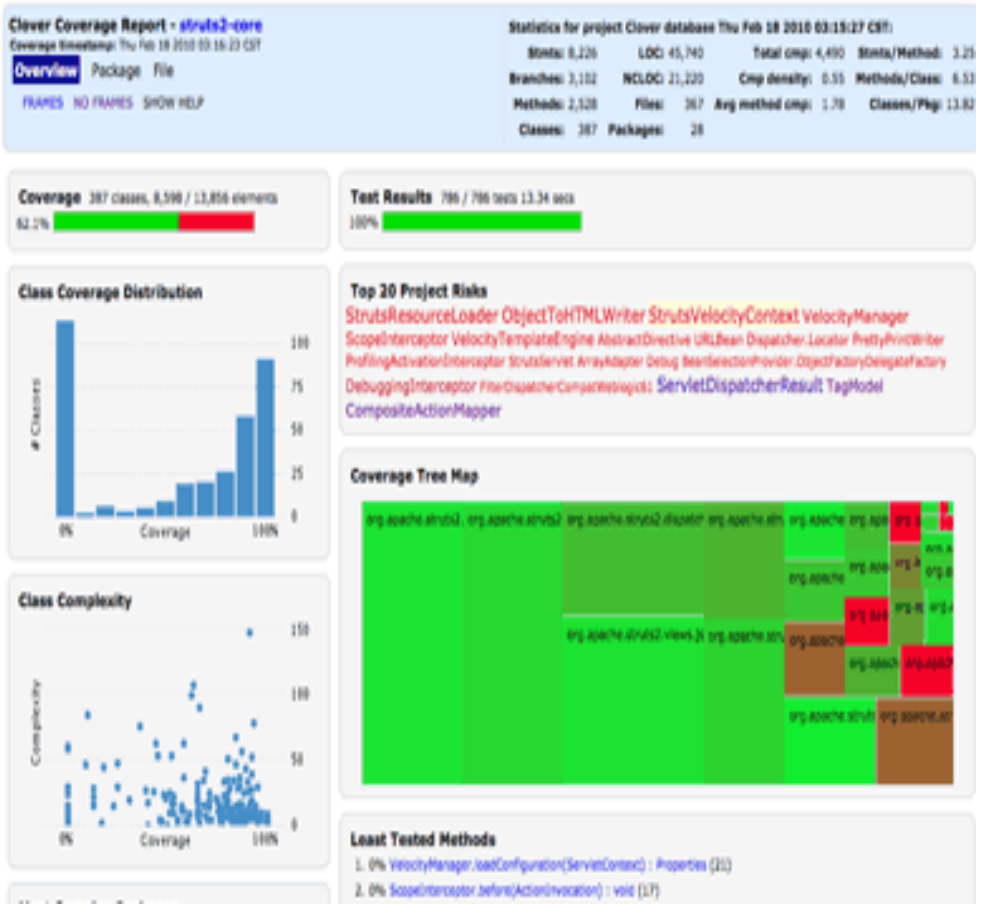
Measure test coverage wisely

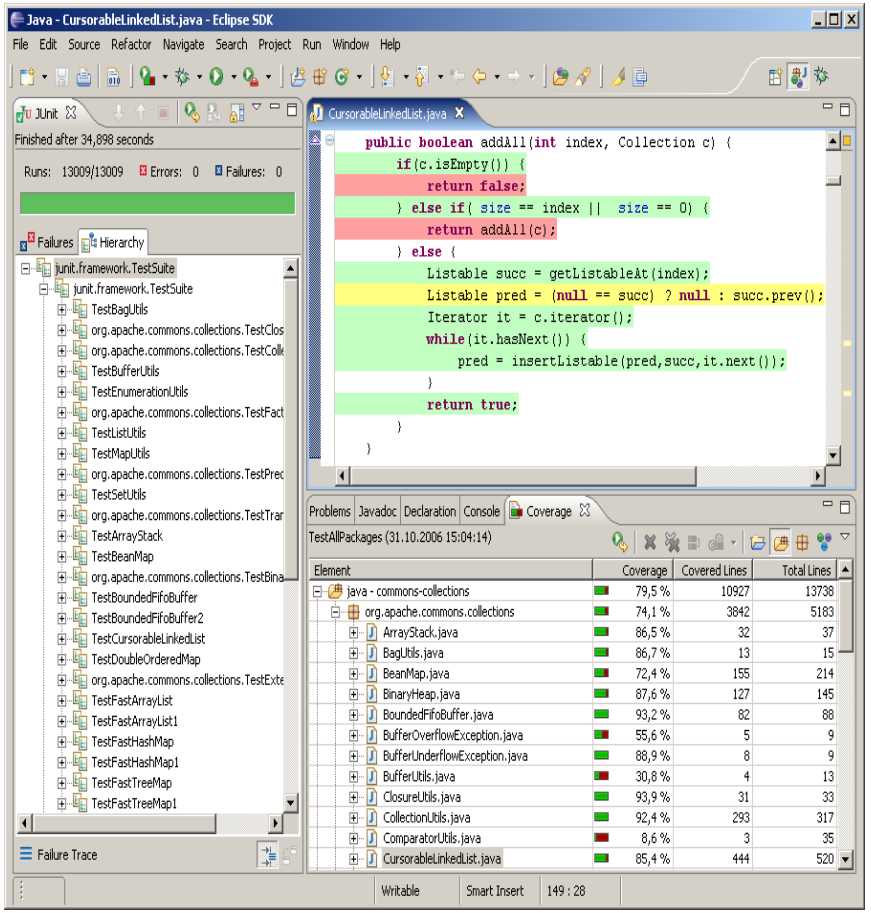
- How do you know you've tested enough?
- Which tests test what parts of the code?
- Tools:
  - Clover (Java)
  - Emma (Java)
  - NCover (.Net)
  - Dot Cover (.Net)
  - Testdriven.NET (.Net)

What test coverage is acceptable?

- If you develop ”Test first” you automatically get good coverage. Not unfeasible to acheive close to 100% with little effort.
- Prioritize the coverage on the most important parts
  - Look at the complexity
  - Which parts change the most?
  - Which parts break the most?

Clover and Emma example





Love your test suite like it was your code

- Your tests need love and nurturing or else they will die
  - Refactor and remove duplication
  - Fix or remove broken tests immediately
- Give them lots of exercise
  - Run them as much as practical in CI
  - Keep them fit and fast

Love your testers too

- The right tool for the job
- Testing is a profession – You can learn from it
- If it's too hard to automate, maybe it's not worth it

?

Referenser

- Working effectively with Legacy Code, Michael Feathers
- Continuous Delivery, Jez Humble, David Farley
- Clover test coverage tool, http://www.atlassian.com/software/clover/
- Emma test coverage, http://emma.sourceforge.net/
- Testdriven.NET, http://testdriven.net/
- Nunit, http://www.nunit.org/
- Selenium, http://seleniumhq.org
- Concordion, http://www.concordion.org/
- Moq, http://code.google.com/p/moq/
- Mockito, http://mockito.org/