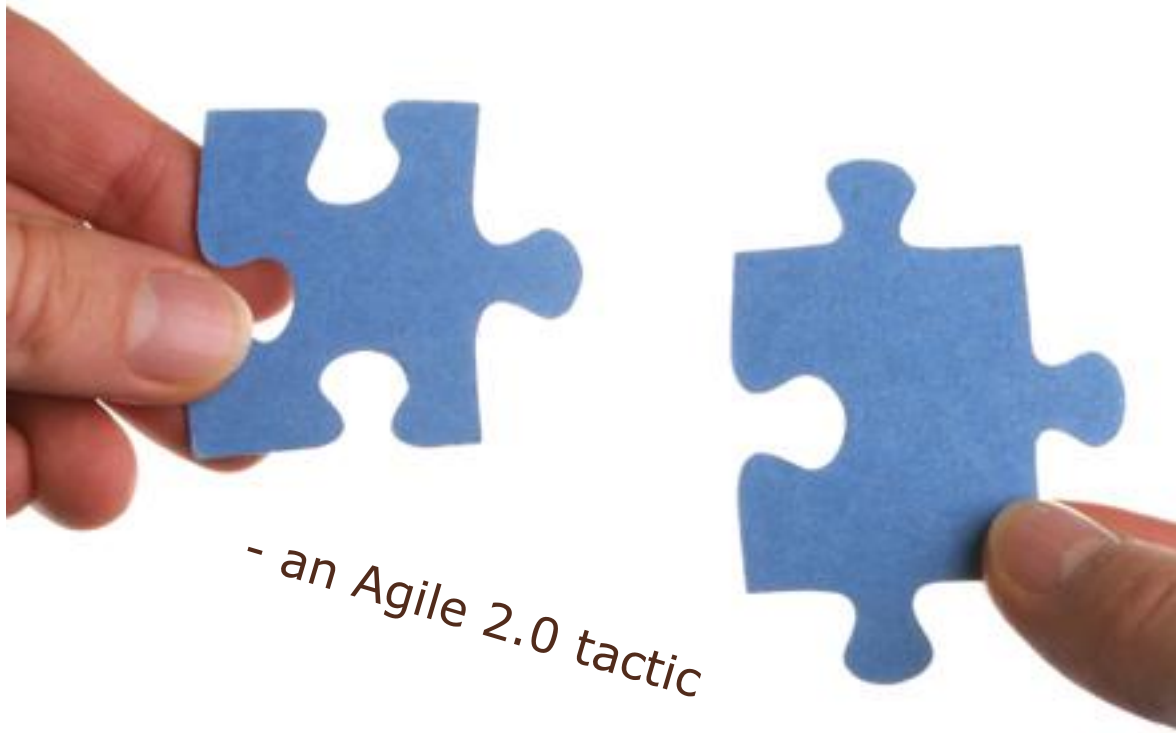
OW test model

VI REALISERER
DIN
KONKURRANSEKRAFT



Who is Erik Drolshammer?

- Consultant Objectware AS – erik.drolshammer@objectware.no

- Master in Computer Science from NTNU
  - *Master thesis*

    Improved Backward Compatibility and API Stability with Advanced Continuous Integration

  - *Depth study*

    Best Practice within Java Web Application Development

Agenda

1. Background and problem description
2. OW Test Model
3. Technical implementation
4. Summary

Background

Complex project;

- Many remote services within the project
- Integration with multiple external systems
- JMS
- OSWorkflow
- Webstart
- C, C++, Java 1.5, Java 6
- Solaris, Linux, OSX, Windows
- Performance critical data processing

Problem description

What is an integration test?
What is a system test?
How to decide which definition of a term to use?
How to synchronize the terminology in the company? Across projects?
Communication with the customer? With project manager?
Agile and TDD versus the V-model
When to run the different types of integration tests?
Before check in to the version control system?
Run by CI server?
Manual test? Automatic test?
What must be tested in an expensive environment? Can it be done in a cheaper way?
File structure for tests – how to decide?

Problem description

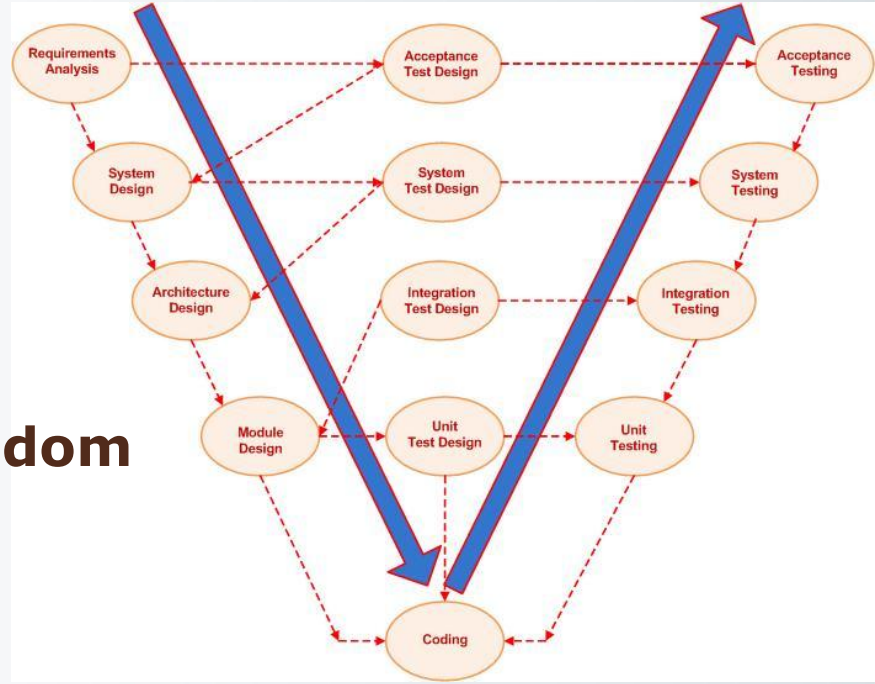
What is an integration test?
What is a system test?
How to decide which definition of a term to use?
How to synchronize the terminology in the company? Across projects?
Communication with the customer? With project manager?
Agile and TDD versus the classic?
When to run the different types of integration tests?
Before check in to the version control system?
Run by CI server?
Manual test? Automatic test?
What must be tested in an expensive environment? Can it be done in a cheaper way?
File structure for tests – how to decide?

CHAOS

The V-model is not adequate

What we often see:

- **enormous unit tests**
- **too few integration tests**
- **many broken integration tests**
- **integration tests are run to seldom**
- **hard to write system tests**
- **System tests are too fragile**



Source:
http://en.wikipedia.org/wiki/Image:V-model.JPG

Symptoms

Complex project =>
- Hard to *discuss* tests
- Hard to *write* tests
- Hard to *maintain* tests
- Hard to *change code*
- Hard to run tests

Hard to test => hard to do employ TDD properly

OW Test Model

OW Test Model

Goal
make it easier to test complex projects

How
pragmatic approach, support agile work patterns

Authors
Bård Lind and Erik Drolshammer

Overview

Terminology and categorization matrix

Design principles/drivers/guidelines

Groups and CI phases

Terminology

+ Unit tests (simple and well-defined)

- Integration test <- too inprecise

- Functional test <- too inprecise

+ Service test (it is a service oriented world)

+ System test

Tackle complexity #1

Single-responsibility principle:

”A ~~class~~**test** should have one,

and only one, reason to change.” SRP

Tackle complexity #2

Follow general OO practices also for test code, especially

- **Separation of concerns**
- **High cohesion, low coupling**

Tackle complexity #3

It *is* relevant whether a broken build is caused by bad input data or the latest change to the business logic.

-> with or without data

Tests that depend on external systems can be complex and expensive.

-> without or without environment

Categorization matrix

	No data	Data	No environment	Environment
Unit test				
Service test				
System test				

CI Phases

The longer a build takes to run, the less often people will run it.

-> speed is an important driver

Machines are better suited for mundane test tasks.

-> use the CI server for all it is worth!

Use your brain! Get the most out of your time and money.

-> Choose *what* to test carefully.

CI Phases - when to run the different tests

1 - Before check-in to VCS

2 - Multiple times a day*
3 - Nightly
4 – Weekly
5 – Release

*(e.g. hourly or every second hour)

CI Phases - when to run the different tests

1 - Before check-in to VCS } **developer**

2 - Multiple times a day*
3 - Nightly
4 – Weekly
5 – Release
} **CI server**

* (e.g. hourly or every second hour)

Categorization matrix with phases

	No data	Data	No environment	Environment
Unit test	Fast (1-2)	n/a	Fast (1-2)	n/a
Service test	Fast (1-4)	Slow (2-4)	Fast (1-4)	Slow (2-4)
System test	Slow (3-5)	Slow (3-5)	Slow (3-5)	Slow (3-5)

NOTE! The phases are only included as basis for discussion.

Groups and CI phases

- Groups are used to filter which tests to run in each phase.
  - One group for each external dependency
  - One group for each data source or provider
- One error should (ideally) result in only *one* test failure.

Implementation

Implementation requirements

Groups

Dependencies between tests and groups of tests

setUp and **tearDown** at test and group level

Dependency injection

Embedded, in-memory alternatives for heavyweight technologies

Implementation requirements

Groups

Dependencies between tests and groups of tests

setUp and **tearDown** at test and group level

TestNG

Dependency injection

Spring

Embedded, in-memory alternatives for heavyweight technologies

JMS: **ActiveMQ**

JTA**: Atomikos, JOTM**

JPA: **Spring**

EJB3: **Spring Pitchfork**

JAX-WS: **XFire**

Servlet container: **Jetty**

Database: **HSQLDB, Apache Derby**

Source: http://wiki.community.objectware.no/display/smidigtonull/Lightweight+alternatives+to+heavyweight+technologies

Indirect requirements

Continuous Integration (CI) server

Version Control System

Maven Artifact Repository

Indirect requirements

Continuous Integration (CI) server

Version Control System

Maven Artifact Repository

Enterprise Maven Infrastructure

Software stack in a real-life project

Maven (release and surefire plugins are central)

TestNG, EasyMock

Spring 2.x, ActiveMQ, HSQLDB

Continuum

Artifactory

Subversion

Test Code Location

Driver: Keep code and tests close.

	Location
Unit test	src/test in the same module as the unit under test
Service test	If the service spans multiple modules, put the tests in a separate module. Otherwise, in the same module.
System test	If the system spans multiple projects, put the tests in a separate project, otherwise in a separate module.

Some additional comments

- Write tests during development, not as a separate phase before delivery.
- unit tests > service tests > system tests
- Choose carefully what to test where.
- **Test to *ensure that the software works.***

Benefits and summary

Observations

Improved communication

More tests, but fewer lines of test code. (Minimal overlap between tests.)

Less responsibility per test

Less time spent on writing tests

Benefits

- **High complexity becomes manageable**
- **Cost reduction**
  - less effort "wasted" in test phases
  - less bugs to fix
  - easier to find cause of bugs
  - reduced ramp-up time for new developers
- **Improved quality**
  - less bugs reintroduced
  - code works as intended
  - testable code tend to be better code –> easier to understand, easier to extend

Summary

OW Test Model

- a test model that reduce test complexity
- supports agile work patterns like **TDD** and **Continuous Integration**
- can be implemented with freely available Open-Source Software

Q & A

Read more in community wiki

OW Test Model

http://wiki.community.objectware.no/display/smidigtonull/OW+Test+Model

Resources

Enterprise Maven Infrastructure

http://wiki.community.objectware.no/display/smidigtonull/Enterprise+Maven+Infrastructure

Single-Responsibility Principle:

http://www.objectmentor.com/resources/articles/srp.pdf

Next Generation Java Testing: TestNG and Advanced Concepts:

http://www.amazon.com/Next-Generation-Java-Testing-Advanced/dp/0321503104

Manifesto for Agile Software Development: http://agilemanifesto.org/