



**Unit Testing
and
Test Driven Development (TDD)**

Training provided from ADT
Volvo Information Technology

Who are we?

- Micael Andersson, Volvo Information technology
 - Java EE ADT
 - Mobile Solutions iOS/Android
- Gerardo, Grégoire
 - Supervisors for this course

Outline of the course

- Session 1 & 2:
 - Background: Unit Testing
 - Junit : Basics
 - Junit : **Advanced topics**
 - **Test Driven Development**
 - Breaking dependencies: **Moquito**
 - Background: Integration Testing
 - **Integration testing**
- Session 3 (& 4):
 - **Refactoring**
 - Requirements driving the test

Course material - Java

- These slides...
- Online resources
 - www.junit.org
 - code.google.com/p/mockito/
 - www.dbunit.org
 - strutstestcase.sourceforge.net

About Tests ...

- Everybody knows they should, but few actually do
- “Why isn’t this tested before”?
 - Because it has been too expensive, difficult, cumbersome to test
 - Because we have been too busy
 - Because things have changed



Discussion: In case the application
you currently work on lacks tests;

- In your opinion; what’s the main reason for this?



Quality Assurance precedes Quality Assessment

- Testing is about Quality **Assurance**, not just Quality **Assessment**
- Quality Assessment only indirectly affect quality
- Testing *reveals information*
- Testing helps *focus project activity*



Test Automation Goals

- Tests should be S.M.A.R.T:
- **S**elf Checking
- **M**aintainable
- **A**ct as documentation
- **R**epeatable and Robust
- **T**o the point – provide "defect triangulation"



Manual Tests are ...

- Repetitive
- Error-prone
- Difficult to test other units than the User Interface
- yet ...
- a (Manual) Test Process must be present in order to automate it!



Critical Success Factors for Automated Tests

- **Repeatability and Consistency**

- Once the test is complete, it should **pass repeatedly**, whether it executes by itself or within a test suite.
- When a completed test fails, we need to quickly and accurately pinpoint the cause: did the test **uncover a bug** in the system, or is **the test itself faulty**?

- **Readability**

- The tests are the definitive **reference for the system requirements**.

- **Maintainability**

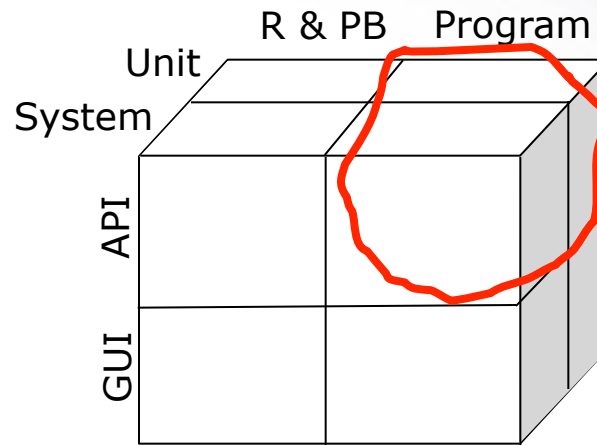
- Iterative, test-first development yields as **much (or more)** test code than system code
- Thus we have to be as concerned (or more) with the maintenance costs of test code as compared to system code.

Testability

- Testability consists of two fundamental characteristics:
 - **Visibility** – the tester can see (and understand) what happens within the system (i.e. can observe important aspects of the internal state of the system)
 - **Control** – the tester can force interesting things to happen within the system (i.e. can control its behavior)
- Testability doesn't just happen. **It must be designed and built into a system.** Writing tests before designing and building the system (a.k.a. Test-First or Test-Driven Development) is a great way of achieving good testability.

Classifying Automated Tests

- **Granularity**
 - Entire system
 - Individual units
- **Point of Contact**
 - Existing User Interface
 - Testability API
- **Test Case Production**
 - Record & Play Back
 - Hand Written (programmatic)



Test (and build) automation

