Various Kinds of Testing

Developer Testing is Overrated

- Developer testing (automated unit/integration/acceptance tests) is not sufficient for revealing all application defects. Also other techniques are needed, for example code reviews, exploratory testing and usability testing.
  - Different kinds of testing finds different kinds of problems.

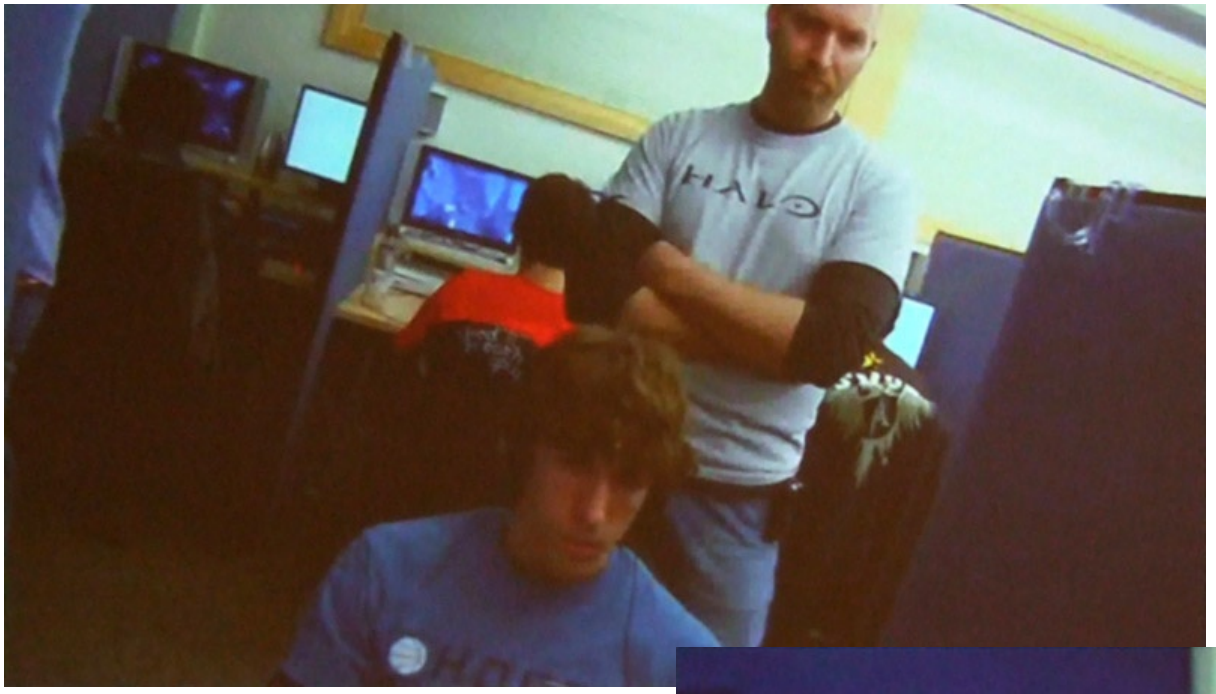
- Presentation:
  (22 min)

http://www.infoq.com/presentations/francl-testing-overrated
http://railspikes.com/2008/7/11/testing-is-overrated
http://railspikes.com/2008/12/2/testing-is-overrated-great-talk

Usability Testing

- Find a test user who *has not yet used the system* (preferably someone from the system's end users) and give him some *realistic task* that he should do with the system.
- Look from behind and write down *everything* that the user does and says. The actions are more valuable information than what the users says. (Recording the video and audio is also possible, but they are slower to analyze than good handwritten notes.)
- *Do not hint* the user anything about how to accomplish the task or how to use the system. Avoid putting pressure on the user.
- When the user has problems with the system, find out the usability problem that caused it and fix it. Then iterate.
- A couple of users is enough to find most of the issues. Can be done also with a paper prototype, before the system has been implemented.

http://en.wikipedia.org/wiki/Usability_testing
http://www.useit.com/alertbox/20010805.html



GDC SF 2009,
Halo in the Laboratory,
Team Participation

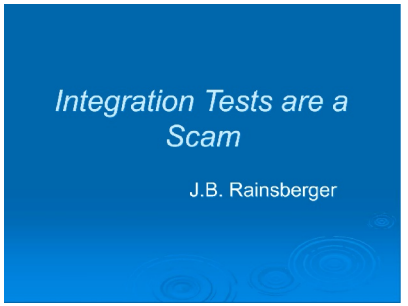
Exploratory Testing

- Automated tests are good for making sure known bugs do not reappear. Manual tests are needed for finding new unknown bugs.
- A skilled tester can bring much value to an agile team.
- Exploratory testing is simultaneous learning, test design, and test execution. Some examples of test plans:
  - "Define work flows through DecideRight and try each one. The flows should represent realistic scenarios of use, and they should collectively encompass each primary function of the product."
  - "We need to understand the performance and reliability characteristics of DecideRight as decision complexity increased. Start with a nominal scenario and scale it up in terms of number of options and factors until the application appears to hang, crash, or gracefully prevent user from enlarging any further."
  - "Test all fields that allow data entry (you know the drill: function, stress, limits)"

http://www.satisfice.com/articles/et-article.pdf
http://testobsessed.com/2007/02/19/test-heuristics-cheat-sheet/
http://softwareeducation.wordpress.com/2009/09/17/36-testing-heuristics/
http://www.indicthreads.com/1324/agile-teams-miss-out-by-having-a-narrow-focus-on-testing/

Integration Testing

- *Integration test* is any test whose result (pass or fail) depends on the correctness of more than one interesting behavior.
  - When an integration test fails, it's not clear where the failure is. You waste time in finding it instead of fixing it.
  - Integration tests tend to be slower than focused tests → more time executing them → run them less often → false sense of security about the system.
- When showing *basic correctness* (not performance, reliability, security etc.), write only focused object tests. Test one thing at a time. You should not need to write integration tests.
- Presentation:
  (0:00-27:30 / 93 min)



http://www.infoq.com/presentations/integration-tests-scam
http://www.jbrains.ca/permalink/239

Acceptance Testing

- When the customer wants some feature to be developed, acceptance tests are written for it. They are used as:
  - Communication tool: Promote discussion with the customer, to make sure that we understand eachother and we are *building the right thing*. (In contrast, unit tests make sure were are *building it right*. Both are needed.)
  - Acceptance criteria: Passing the acceptance tests is part of the feature's *"definition of done"*.
  - Regression/integration tests: Acceptance tests are automated black box tests. They are run as part of a continuous integration build.
- Can make sense in medium to large programs, as maintaining the acceptance tests has some overhead.
- There are various frameworks with different styles, but at the moment they have rough edges.

http://fitnesse.org/FitNesse.UserGuide.AcceptanceTests
http://www.extremeprogramming.org/rules/functionaltests.html
http://www.objectmentor.com/resources/publishedArticles.html → Craftsman #24-#42

Concurrency

- Given *lastIdUsed* is **93**, when two threads call the *incrementValue()* method, then what is the end state?

```
public class IdGenerator {
    int lastIdUsed;

    public int incrementValue() {
        return ++lastIdUsed;
    }
}
```

  - Thread 1 gets **94**, thread 2 gets **95**, and *lastIdUsed* is **95**.
  - Thread 1 gets **95**, thread 2 gets **94**, and *lastIdUsed* is **95**.
  - Thread 1 gets **94**, thread 2 gets **94**, and *lastIdUsed* is **94**.

Clean Code p321

Example Concurrency Bug
(Guice 1.0)

6.3.2009 19:01:18 net.orfjackal.dimdwarf.tasks.TransactionFilter filter
INFO: Task failed, rolling back its transaction
java.lang.NullPointerException
at com.google.inject.InjectorImpl.injectMembers(**InjectorImpl.java:673**)
at com.google.inject.InjectorImpl$8.call(InjectorImpl.java:682)
at com.google.inject.InjectorImpl$8.call(InjectorImpl.java:681)
at com.google.inject.InjectorImpl.callInContext(InjectorImpl.java:747)
at com.google.inject.InjectorImpl.injectMembers(InjectorImpl.java:680)
at net.orfjackal.dimdwarf.serial.InjectObjectsOnDeserialization.afterResolve(
at net.orfjackal.dimdwarf.serial.ObjectSerializerImpl$MyObjectInputStream.re
at java.io.ObjectInputStream.checkResolve(ObjectInputStream.java:1377)
at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1329)
at java.io.ObjectInputStream.readObject(ObjectInputStream.java:351)
at net.orfjackal.dimdwarf.serial.ObjectSerializerImpl.deserializeFromStream(
at net.orfjackal.dimdwarf.serial.ObjectSerializerImpl.deserialize(ObjectSeriali
...

http://dimdwarf.sourceforge.net/ commit 96b49611 and before (fixed on 2009-08-17)
http://groups.google.com/group/google-guice/browse_thread/thread/7cdff66a4b7acc9d

Concurrency Testing

- Concurrency bugs are generally not repeatable – a bug might show up once in a million times.
  - **Every unexplained stack trace** must be investigated thoroughly, *until it has been explained and fixed*, even if the test passes on the next run! Concurrency bugs are real, even if they show up rarely.
- Best advice regarding concurrency: **Don't do it.**
  - Use design patterns and architectures which let you write non-concurrent code, but which then execute it concurrently.
  - Know your libraries: Which classes are not thread-safe? What concurrency tools are there?
    - `java.util.Collections.synchronized*()`
    - `java.util.concurrent.{*, atomic.*, locks.*}`
    - *Java Language Specification, 3rd edition, chapter 17*

Clean Code ch13 p182, p186
http://java.sun.com/docs/books/jls/third_edition/html/memory.html

Concurrency Testing

- Restrict concurrency to as few classes as possible.
- Make sure the code works single-threaded. Don't hunt normal and concurrency bugs simultaneously.
- Run the tests with more threads than processors, to encourage context switching.
  - In IDEA, hold down Shift+F10 to launch tests until 20-50+ processes are running concurrently. After CPU load goes down, look for failed tests and stack traces in log output.
- Instrument your code to try and force failures.
  - Insert calls to `Thread.yield()` or `sleep()` into strategic places to encourage context switching.
  - Use a tool which does the same thing automatically, for example ConTest (http://www.alphaworks.ibm.com/tech/contest).

Clean Code ch13 p187, p188

Choosing a Testing Framework

- Some points to think about:
  - How easy is it to add a new test?
  - How easy is it to add a new fixtures?
  - How readable is the test code?
  - Can any characters be used in test names?
  - Does it support my way of organizing test?
  - Does it integrate well in my development environment?
  - Does it integrate with my preferred mocking frameworks?
  - When tests are run, are they completely isolated or are the side-effects of different tests in the same class visible?
  - Is it possible to run tests in parallel, on multiple CPU cores?
  - Tests may be written in a different programming language.

Some Unit Test Tools

- **JUnit** (Java) http://www.junit.org/
- **TestNG** (Java) http://testng.org/
- **JDave** (Java) http://www.jdave.org/
- **JBehave** (Java) http://jbehave.org/
- **RSpec** (Ruby) http://rspec.info/
- **Specs** (Scala) http://code.google.com/p/specs/
- **ScalaTest** (Scala) http://www.artima.com/scalatest/
- **ScalaCheck** (Scala) http://code.google.com/p/scalacheck/
- **EasyB** (Groovy) http://easyb.org/
- Some comparisons:
  - http://www.codecommit.com/blog/java/the-brilliance-of-bdd
  - http://www.artima.com/weblogs/viewpost.jsp?thread=251945

Some Mock Object Tools

- **JMock** (Java) http://www.jmock.org/
- **EasyMock** (Java) http://easymock.org/
- **Mockito** (Java) http://mockito.org/
- **PowerMock** (Java) http://code.google.com/p/powermock/
- **RSpec** (Ruby) http://rspec.info/documentation/mocks/

Some Acceptance Test Tools

- **FitNesse** (Java) http://fitnesse.org/
- **Robot Framework** (Python, Java, any) http://code.google.com/p/robotframework/
- **Cucumber** (Ruby, any) http://cukes.info/
  - http://blog.josephwilk.net/ruby/outside-in-development-with-cucumber-and-rspec.html

Some Continuous Integration Tools

- **CruiseControl** http://cruisecontrol.sourceforge.net/
- **TeamCity** http://www.jetbrains.com/teamcity/

"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly."

- http://www.martinfowler.com/articles/continuousIntegration.html

Course Material

- http://www.infoq.com/presentations/francl-testing-overrated
  http://railspikes.com/2008/7/11/testing-is-overrated
  http://railspikes.com/2008/12/2/testing-is-overrated-great-talk
- http://www.satisfice.com/articles/et-article.pdf
- http://www.extremeprogramming.org/rules/functionaltests.html