Introduction to Automated Tests

Paulo Cheque

Summer 2009

About

- Language
- Types of tests
- Introduction to test frameworks

Many communities

- A lot of terminology and dialects:
  - Agile communities
  - Formal communities
  - QA communities
  - Tools
  - Clients

Standards Specifications

- - BS 7925-2:1998. Software Component Testing.
- - DO-178B:1992. Software Considerations in Airborne Systems and Equipment
- Certification, Requirements and Technical Concepts for Aviation (RTCA SC167).
- - IEEE 610.12:1990. Standard Glossary of Software Engineering Terminology.
- - IEEE 829:1998. Standard for Software Test Documentation.
- - IEEE 1008:1993. Standard for Software Unit Testing.
- - IEEE 1012:1986. Standard for Verification and Validation Plans
- ...

Jargons

- Defect/Mistake/Error/Failure
- Verification
- Validation
- Inspection
- Debugging
- …

Terms

- Black-box/White/Gray/Glass
- Functional Tests
- Structural Tests

- Test
- Test case
- Test suite

Symbols

- SUT/AUT: System/Application Under Test
- TFD/POUT: Test-First Development / Plain Old Unit Test
- TAD: Test-After Development
- TDD: Test-Driven Development / Design
- BDD/EDD/STDD: Behaviour/Example/Story Test Driven Development
- TDDD: Test-Driven Design Databases

Double

- ***Dummy***: A simple object to make a test executable
- ***Fake***: Lightweight object that contains a false implementation
- ***Stub***: Object with data to be used in tests
- ***Mock***: Object with an appropriate interface that contains data and behavior that will be used in tests
- ***Spy***: Object to capture indirect calls

Type of Automated Tests

- Unit
- Integration
- Acceptance
- Interface
- Smoke
- Sanity
- Mutation
- ...
- Performance
- Load
- Stress
- Longevity
- Security
- Installation
- ...

Unit test

- Unit: Class/Module or Method/Function
- Unit or “mini-integration”

- Test a feature, not an implementation
- Basic but very important test
- Solid test
- Avoid debugging

Integration tests

- In general: Test that involve more than one unit

- Verify errors in the relationship between modules, components or layers that may work fine individually

Acceptance

- Test a feature from the user's point of view
- Assurance for a client that software works perfectly and correctly

- Clients, users and developers may (must) work together to write tests
  - Implementation details hidden
  - Client language – Domain Specific Language

Interface Tests

- User Interface
  - Console
  - GUI
  - WUI
- => Acceptance Tests
- => Usability
- => Layout Tests

Smoke Tests

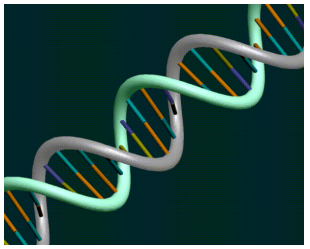
- Rapid, extensive and superficial tests
- After software installation
- Example:
  - Visit a Web page and verify that 404 text is not visible

Sanity Tests

- A smoke test
- Appropriate to find huge errors in algorithms
- Use particular technique of an algorithm to identify possible errors
  - Theorems
  - Mathematical properties

Mutation tests

- Test a test
- *"Tests may show some errors, but not its absence"* - Dijkstra
- Infinite possibility of errors?
  - Competent developer
- Mutant: SUT with few changes
- Run a suite of tests under mutants
- Analysis of results: Mutants and Tests



Performance tests

- Evaluate response time of a specific module
- Profilers help finding bottlenecks
- Unit test suites without mocks may be used like profilers (this is not the main goal of those suites)
- PS: Don't evaluate computational complexity

Load tests

- Simulate a lot of users and requests
  - Simultaneous or inside a time interval
- Test
  - Infra-Structure
  - Hardware
  - Network bandwidth
  - Database and servers

Stress tests

- Maximum load test
- Identify maximum amount of:
  - users
  - requests
  - data
- Limitations of an environment

Longevity tests

- Test to verify performance and correction within a large period of time
- Find problems with memory leakage
- Find cache errors

Other types...

- Tests of:
  - Installation: Verify if all components are interconnected properly without hardware incompatibility
  - Recovery: Appropriated reaction after errors
  - Configuration: Different configurations and distinct environments (portability)
  - ...
- Alfa / Beta versions

Security tests

- Useful for applications exposed to bad intentioned users
  - Web, banks ...
- Test:
  - Permissions errors
  - Errors in model layer (SQLInjection ...)
  - Bugs in servers
  - Fragility to attacks like DoS (Stress tests)

Pertinent information

- **Machine**: CPU, Memory, HD, Swap ...
- **OS**: Name/Distribution, Version, Kernel, Locale ...
- **Servers**: Name-Version, Pertinent files,
  - Tomcat 5.5, Jboss 4.0.4, (/etc/init.d/jboss)
- **Database**: Name-Version, Machine, Schema
- **Network**: IP, Firewall, Port, DNS ...
- **Dependencies**: Name-Version
  - JDK 1.5.0_06, Ruby 1.8
- **Product**: Name, Version, installation data, logs

Unit tests

- Main
- X-Unit frameworks
  - Test case
  - Verification
  - Exceptions
  - Results
  - Report

Main

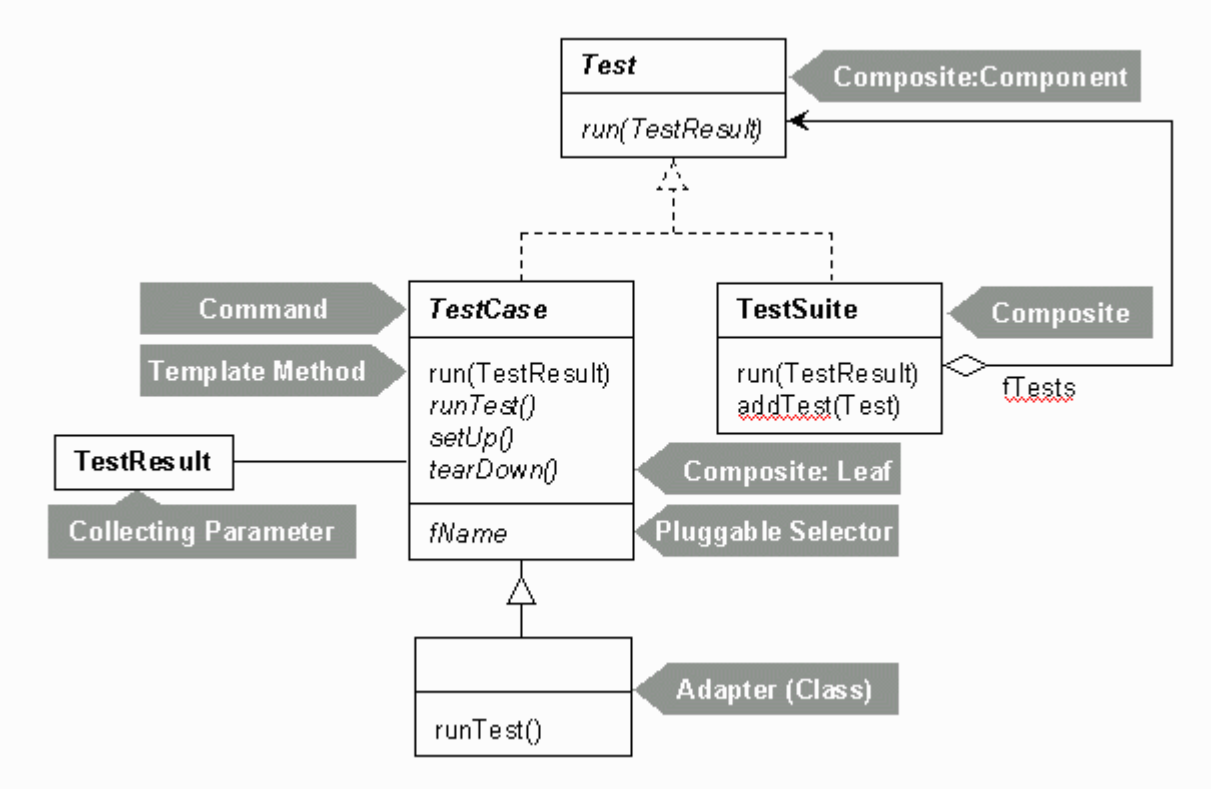
```
public class MainExample {


    public static void main(String[] args) {
        if(!new Double(Math.sqrt(-1)).isNaN())
            throw new RuntimeException("Ops!");
        // Or
        assert new Double(Math.sqrt(-1)).isNaN() == true;
    }


}
```

x-Unit

- Article:
  http://junit.sourceforge.net/doc/cookstour/cookstour.htm



Test case

- Conventions
  - `public void testSomeMethodName() { ... }`
- Annotations

```
@Test public void someMethodName() { ... }
```

- Configurations
  - xml...

Verification

- AssertTrue / AssertFalse
- AssertEquals
- AssertSame
- Fail
- Expected Exceptions

- Friendly messages

- Hamcrest: Matchers + AssertThat

Demonstration

- C
- Java
- Scala
- Groovy

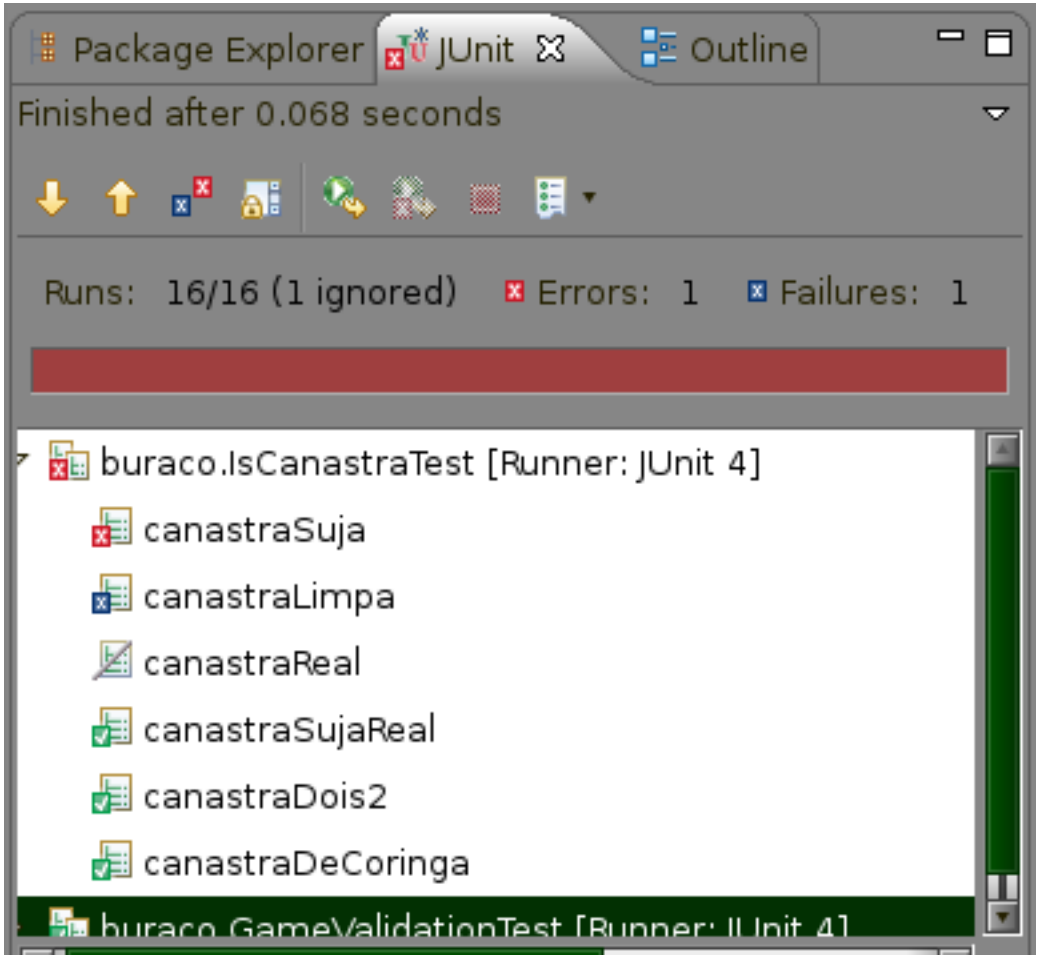
Setup e Teardown

- Test cases must be independent:
  - From each other
  - From number of times it has been executed
  - From external factors

- Setup: Prepare environment to test
- Teardown: Clean environment for other tests

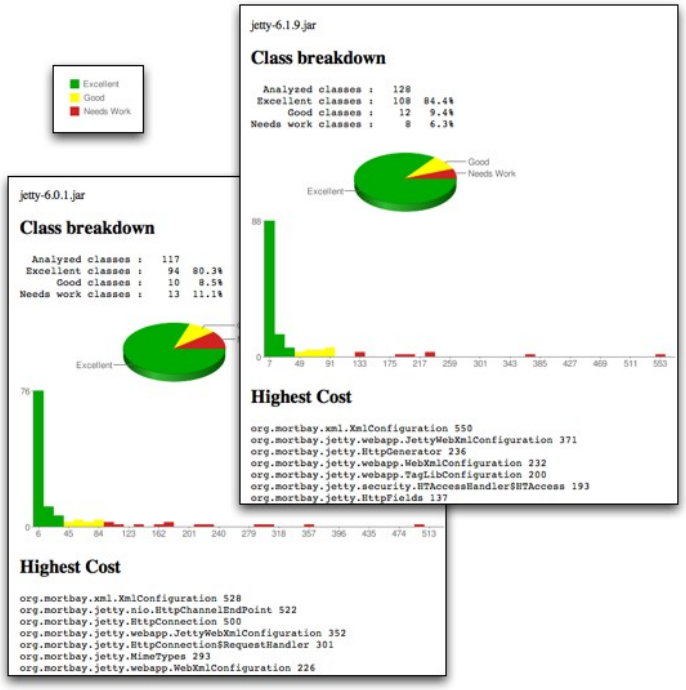
Results

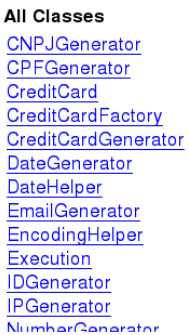
- Skip
- Fail
- Error
- Success

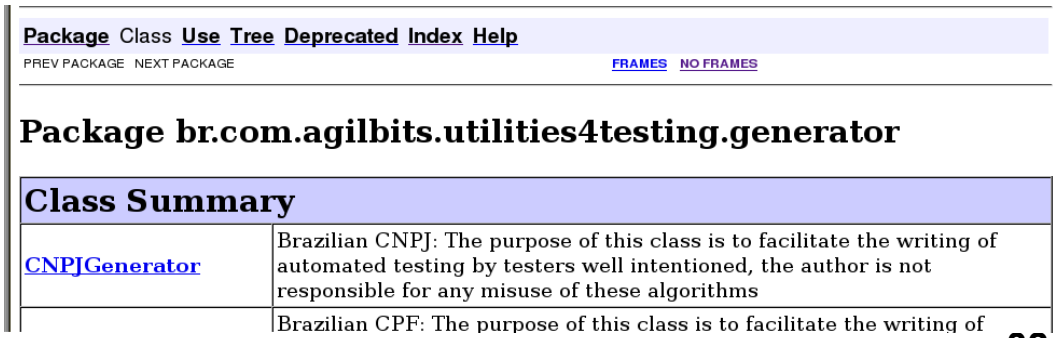


Reports

- Language
  - API-Docs
- Tools
  - StackTrace
  - Logs
  - Screenshots
- Metrics







Some tools

- Unit tests:
  - CxxTest (C++): http://cxxtest.sourceforge.net
  - CUnit: http://cunit.sourceforge.net
  - JUnit (Java): http://www.junit.org
  - DUnit (Delphi): http://dunit.sourceforge.net
  - VBUnit (Visual Basic): http://www.vbunit.com
  - TestNG (Java): http://testng.org
  - RSpec (Ruby): http://rspec.info/

- http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks

+some tools

- Mock Objects:
  - SevenMock (Java): http://seven-mock.sourceforge.net
  - EasyMock (Java): http://www.easymock.org/
  - JMock (Java): http://www.jmock.org
  - Rhino.Mocks (.NET): http://www.ayende.com/projects/rhino-mocks.aspx
  - SMock (Smalltalk): http://www.macta.f2s.com/Thoughts/smock.html
  - Mockpp (C++): http://mockpp.sourceforge.net
  - GoogleMock (C++): http://code.google.com/p/googlemock

+some tools

- User Interface tests for Desktop:
  - Fest: http://fest.easytesting.org/swing
  - Jemmy: http://jemmy.netbeans.org
  - Marathon: http://www.marathontesting.com
- Web interface tests:
  - Selenium: http://www.openqa.org/selenium
  - Watir: http://wtr.rubyforge.org
- Acceptance tests:
  - Fit: http://fit.c2.com

+some tools

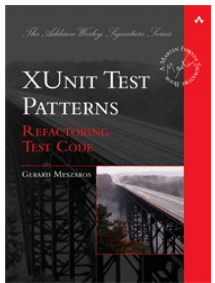
- Mutation tests:
  - Jabuti (USP São Carlos): http://jabuti.incubadora.fapesp.br
  - Jester: http://jester.sourceforge.net
  - Heckle: https://rubyforge.org/projects/seattlerb
- Performance / Load tests:
  - JMeter: http://jakarta.apache.org/jmeter
- Others:
  - JPDFUnit: http://jpdfunit.sourceforge.net

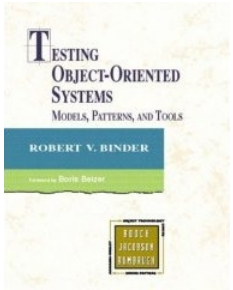
Some links

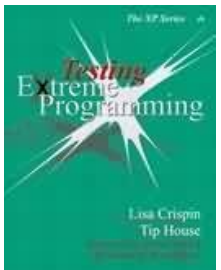
- http://www.testing.com
- http://www.opensourcetesting.org
- http://xunitpatterns.com
- http://www.mockobjects.com
- http://java-source.net/open-source/testing-tools
- http://www.junit.org
- http://www.agilcoop.org.br

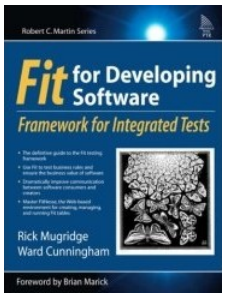
Some Books

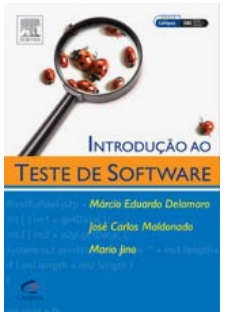
- Gerard Meszaros, “xUnit Test Patterns, Refactoring Test Code”, Addison-Wesley, 2007
- Robert V. Binder, “Testing Object-Oriented Systems”, Addison-Wesley Professional, 1999
- L. Crispin, T. House, “Testing Extreme Programming”, Addison-Wesley, 2005
- R. Mugridge, W. Cunningham, “Fit for Developing Software”, Prentice Hall, 2006
- M. Delamaro, J. Maldonado, M. Jino, “Introdução ao Teste de Software”, Campus, 2007











Contact

http://www.agilcoop.org.br

http://ccsl.ime.usp.br

http://qualipso.org

agilcoop@agilcoop.org.br

paulocheque@agilcoop.org.br