Introdução aos Testes Automatizados – Parte II

Paulo Cheque

09/02/2009 Verão 2009

Sobre

- Linguagem
- Tipos de Testes
- Introdução aos Arcabouços de Teste

Muitas comunidades

- Linguajar distinto:
  - Comunidades Ágeis
  - Comunidades Formais
  - Comunidade de QA
  - Ferramentas
  - Clientes

Padrões

- - 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
- ...

Jargões

- Defeito/Engano/Erro/Falha
- Verificação
- Validação
- Inspeção
- Depuração
- ...

Termos

- Caixa-Preta/Branca/Cinza/Vidro/Transparente
- Teste Funcional
- Teste Estrutural

- Teste
- Casos de Teste
- Bateria de Testes

Siglas

- 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

Dublês

- ***Dummy***: Objeto utilizado apenas para permitir a criação/execução do teste
- ***Fake***: Objeto leve que contém uma implementação falsa
- ***Stub***: Objeto contendo informações que serão utilizadas pelos testes
- ***Mock***: Objeto que possui uma interface apropriada e que contém informações e comportamentos que serão utilizados pelos testes
- ***Spy***: Objeto que captura chamadas indiretas

Tipos de Testes Automatizados

- Unidade
- Integração
- Aceitação
- Interface
- Fumaça
- Sanidade
- Mutação
- ...

- Desempenho
- Carga
- Estresse
- Longevidade
- Segurança
- Instalação
- ...

Teste de Unidade

- Unidade: Classe/Módulo ou Método/Função
- Unidade ou “mini-integração”

- Teste do código fonte com foco na funcionalidade
- Teste básico e muito importante
- Teste sólido
- Evita depuração

Teste de Integração

- Em geral: Teste que envolve mais de uma camada

- Verifica erros da relação entre módulos, componentes e camadas, que podem funcionar perfeitamente individualmente

Aceitação

- Testa as funcionalidades a partir do ponto de vista do cliente
- Garantia pro cliente que o software faz o que deve e faz corretamente

- Clientes e Desenvolvedores podem trabalhar em parceria para a escrita dos testes
  - Detalhes de implementação ocultos
  - Linguagem do cliente

Teste de Interface

- Interface de Usuário
  - Console
  - GUI
  - WUI
- => Teste de Aceitação
- => Teste de Usabilidade
- => Teste de Layout

Teste Fumaça

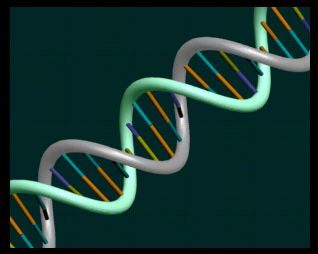
- Testes Rápidos, Superficiais e Abrangentes
- Realizados após a instalação do software
- Exemplo:
  - Acesse uma página Web e verifique que não apareceu o texto “404”

Teste de Sanidade

- Um Teste Fumaça
- Apropriado para buscar erros absurdos em algoritmos
- Utilizar técnicas relacionadas ao algoritmo para identificar possíveis falhas
  - Teoremas
  - Propriedades matemáticas

Teste de Mutação

- Teste dos Testes
- *"Testes podem mostrar a presença de erros, não a sua ausência"* - Dijkstra
- Infinitas possibilidades de erros?
  - Programador competente



- Mutante: SUT com pequenas alterações
- Executar os testes sob os Mutantes
- Análise dos Resultados: Mutantes e Testes

Teste de Desempenho

- Avaliar o tempo de resposta de um módulo específico
- Profilers ajudam a encontrar gargalos
- Baterias de testes podem ser utilizados como profilers
- Obs: Não avalia a complexidade computacional

Teste de Carga

- Simula muitos usuários e muitas requisições
  - Simultâneas ou dentro de um intervalo de tempo

- Testa
  - Infra-Estrutura
  - Hardware
  - Banda de rede
  - Banco de dados e Servidores

Teste de Estresse

- Teste de Carga Máxima
- Identificar quantidade máxima:
  - De usuários
  - De requisições
  - De dados
- Limitações do ambiente

Teste de Longevidade

- Teste que verifica a performance e correção do sistema dentro de um grande período de tempo
- Encontrar problemas com vazamento de memória
- Encontrar erros de cache
  - Não atualização

Outros tipos...

- Teste de:
  - Instalação: Verifica se todos os componentes se interconectaram e se não houve incompatibilidades com o hardware
  - Recuperação: Reação adequada após erros
  - Configuração: Configurações diferentes e ambientes distintos (portabilidade)
  - ...
- Versões Alfa / Beta

Teste de Segurança

- Útil para aplicações expostas a usuários mal-intencionados
  - Web, Caixas eletrônicos ...
- Testa:
  - Falhas de permissão na interface
  - Falhas no modelo (SQLInjection ...)
  - Bugs nos produtos de servidores
  - Fragilidade de ataques DoS (Teste de estresse)

Informações Pertinentes

- **Máquina**: CPU, Memória, HD, Swap ...
- **SO**: Nome/Distribuição, Versão, Kernel, Locale ...
- **Servidores**: Nome-Versão, Arquivos Pertinentes
  - Tomcat 5.5, Jboss 4.0.4, (/etc/init.d/jboss)
- **Bancos de Dados**: Nome-Versão, Máquinas, Schema
- **Rede**: IP, Firewall, Portas, DNS ...
- **Dependências**: Nome-Versão
  - JDK 1.5.0_06, Ruby 1.8
- **Produto**: Nome, Versão, data instalação, logs

Os Primeiros Testes de Unidade

- Main
- Arcabouços x-Unit
  - Casos de Teste
  - Verificações
  - Exceções
  - Resultados
  - Relatórios

Main

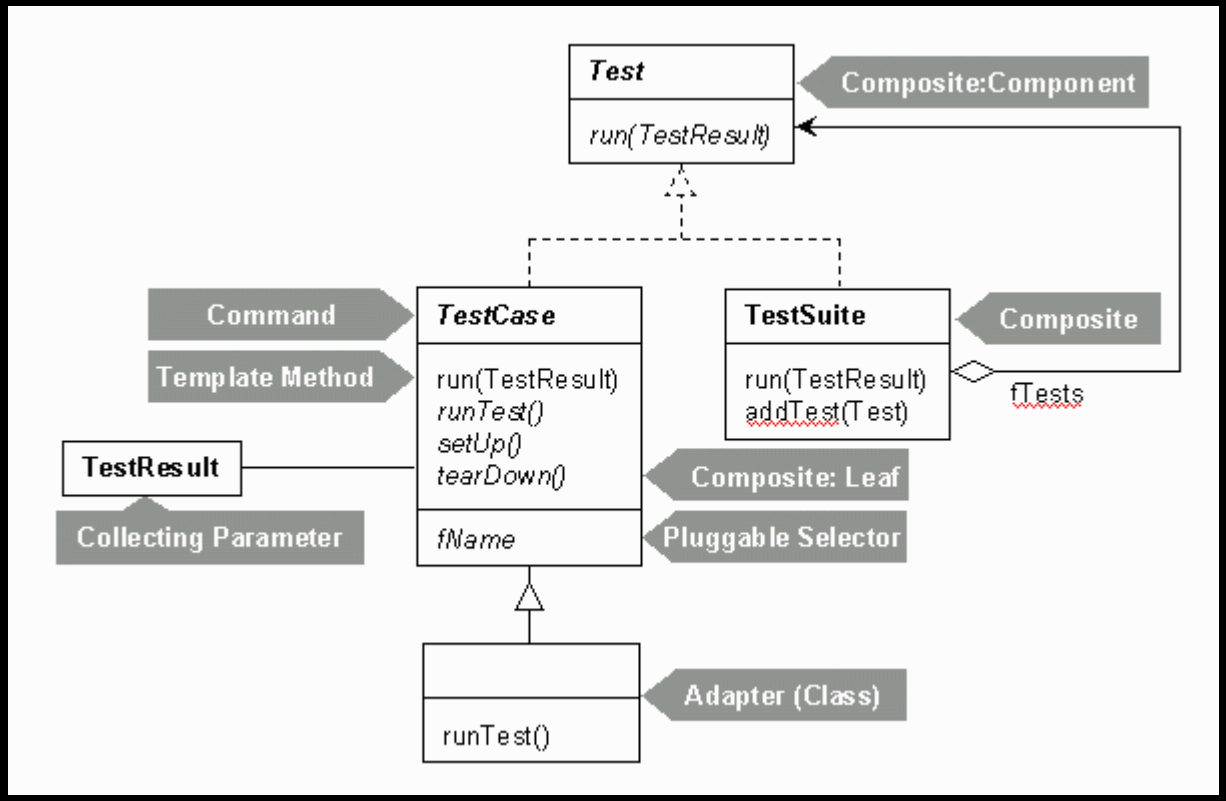
```
public class MainExample {


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


}
```

x-Unit

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



Casos de Teste

- Convenções

```
public void testMetodoDeTeste() { ... }
```

- Anotações

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

- Configurações
  - xml...

Verificação

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

- Mensagens amigáveis

- Hamcrest: Matchers + AssertThat

Demonstrações

- C
- Java
- Scala
- Groovy

Set Up e Tear Down

- Os casos de testes devem ser independentes:
  - Uns dos outros
  - Do número de vezes que é executado
  - De fatores externos

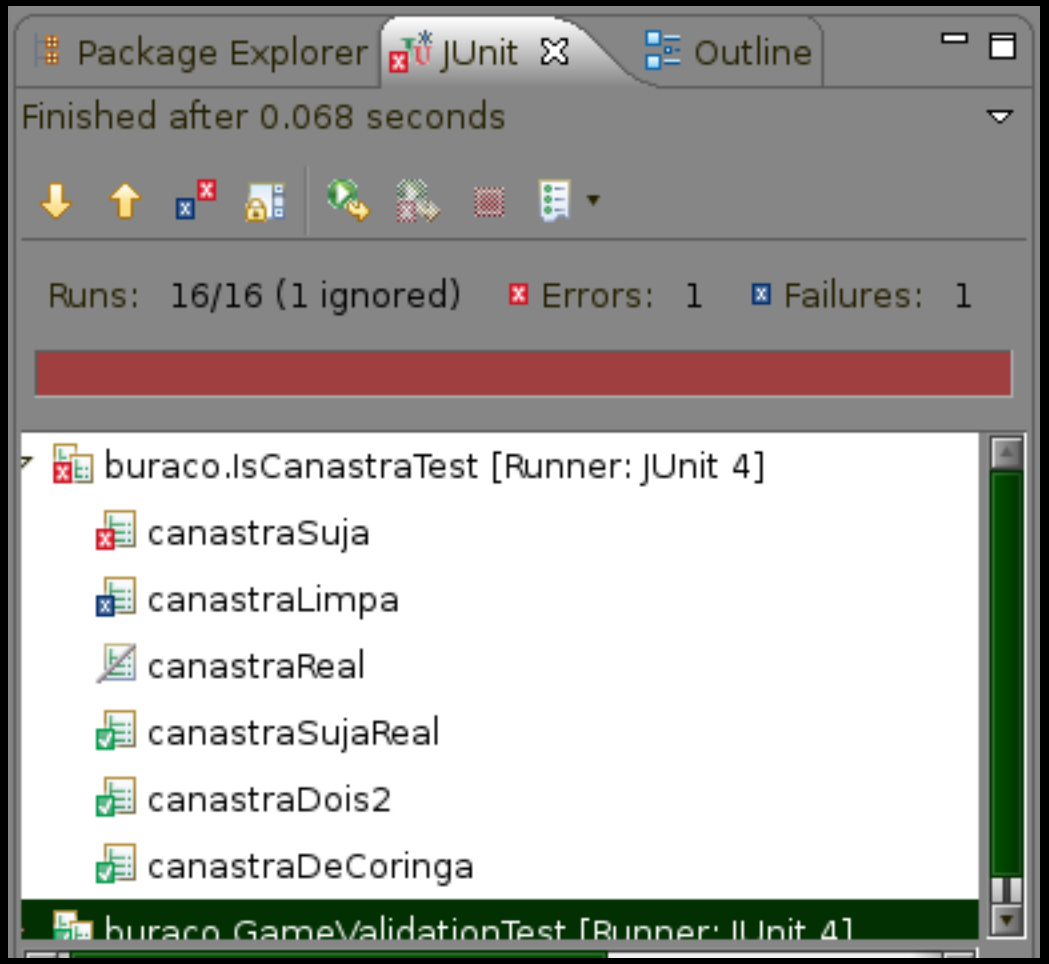
EXEMPLO

- Set Up: Prepara o ambiente para a execução do teste
- Tear Down: Limpa o ambiente para próximos testes

Resultados

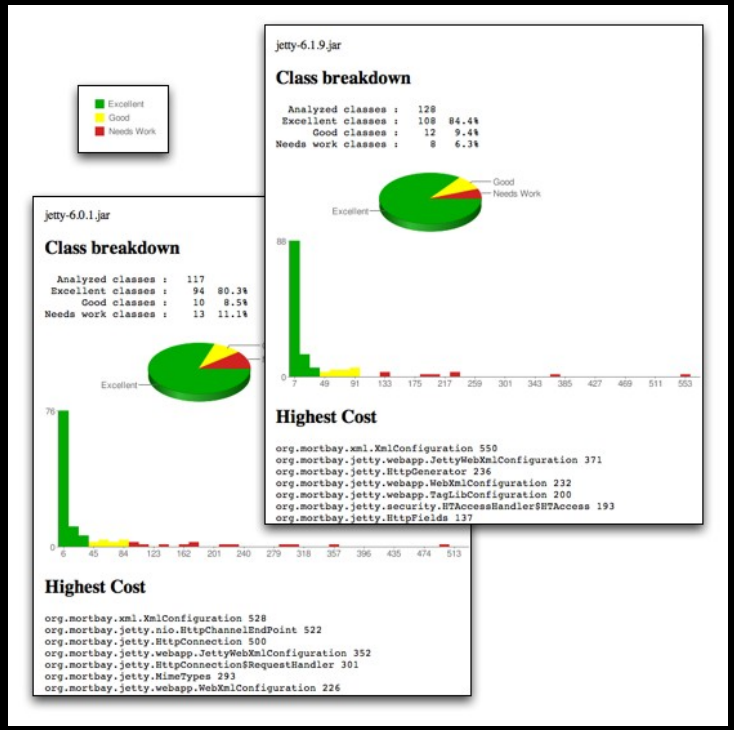
- Skip
- Fail
- Error
- Success

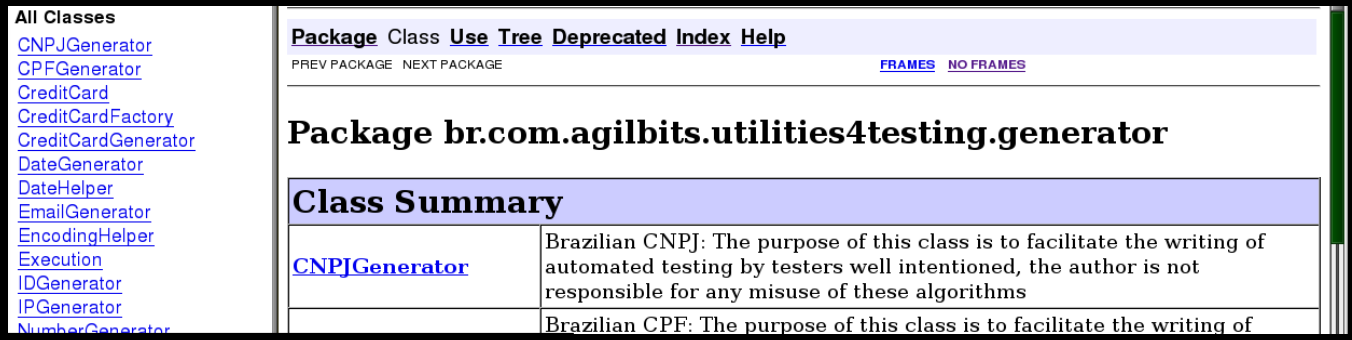
EXEMPLO



Relatórios

- Linguagem
  - API-Docs
- Ferramentas
  - StackTrace
  - Logs
  - Screenshots
- Métricas





Algumas Ferramentas

- Testes de Unidade:
  - 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

+Algumas Ferramentas

- 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

+Algumas Ferramentas

- Testes de Interface Gráfica Desktop:
  - Fest: http://fest.easytesting.org/swing
  - Jemmy: http://jemmy.netbeans.org
  - Marathon: http://www.marathontesting.com
- Testes de Interface Web:
  - Selenium: http://www.openqa.org/selenium
  - Watir: http://wtr.rubyforge.org
- Testes de Aceitação:
  - Fit: http://fit.c2.com

+Algumas Ferramentas

- Testes de Mutação:
  - Jabuti (USP São Carlos): http://jabuti.incubadora.fapesp.br
  - Jester: http://jester.sourceforge.net
  - Heckle: https://rubyforge.org/projects/seattlerb
- Testes de Desempenho / Estresse:
  - JMeter: http://jakarta.apache.org/jmeter
- Outros:
  - JPDFUnit: http://jpdfunit.sourceforge.net

Alguns 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

Alguns Livros

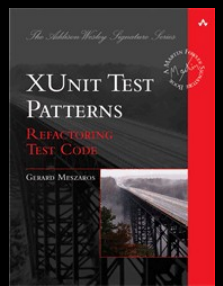
Gerard Meszaros, “xUnit Test Patterns, Refactoring Test Code”, Addison-Wesley, 2007

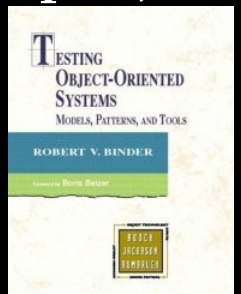
Robert V. Binder, “Testing Object-Oriented Systems”, Addison-Wesley Professional, 1999

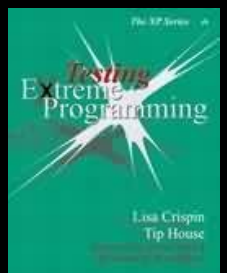
L. Crispin, T. House, “Testing Extreme Programming”, Addison-Wesley, 2005

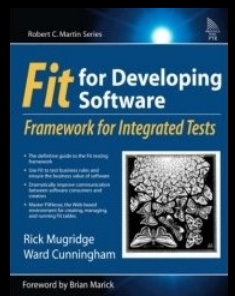
R. Mugridge, W. Cunningham, “Fit for Developing Software”, Prentice Hall, 2006

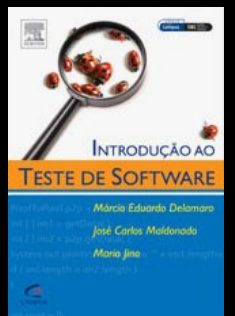
M. Delamaro, J. Maldonado, M. Jino, “Introdução ao Teste de Software”, Campus, 2007











Contato

- http://www.agilcoop.org.br

- agilcoop@agilcoop.org.br
- paulocheque@agilcoop.org.br