Testability-Explorer

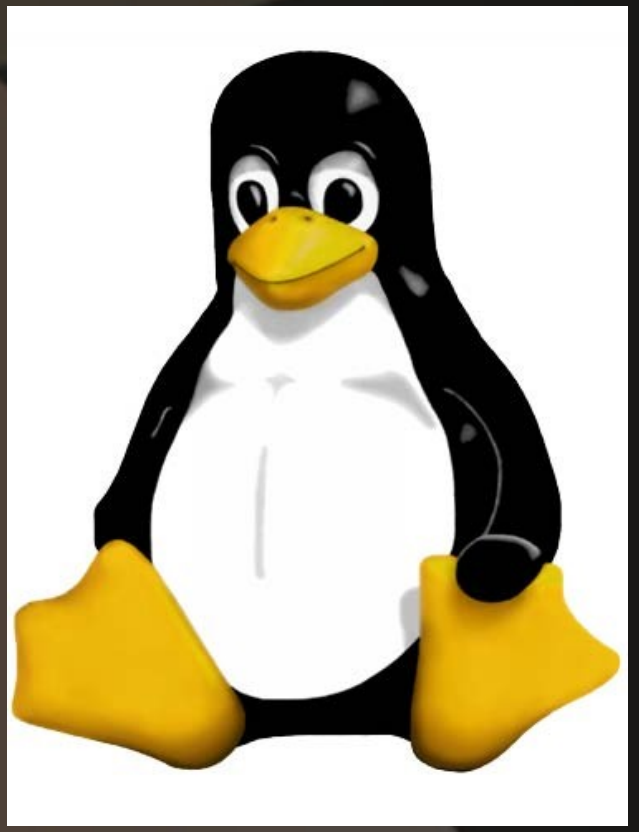
Paulo Cheque (paulocheque@gmail.com)

QualiPSo Workshop
2009

Licença:
Creative Commons: Attribution-Share Alike 3.0 Unported
http://creativecommons.org/licenses/by-sa/3.0/

Software Livre

- Time
  - Donos de projetos
  - Colaboradores
- Contribuições
  - Distantes
  - Pessoas desconhecidas
  - Heterogêneas



- É ágil e seguro sem testes automatizados?

Tendências

- Muitos software livres de qualidade
  - Diversidade de ferramentas
- DSL para testes
- Otimização dos testes
  - Grades Computacionais (Selenium-Grid)
  - Testes Contínuos (JUnitMax)
- Estratégias, padrões e anti-padrões
- Geração de casos de testes
- Novas métricas: cobertura, testabilidade …

Testabilidade (1/2)

- Mede a facilidade de escrita de testes (manual ou automatizado) dentro de um contexto
- Testar nem sempre é simples
  - É preciso controlar o software e o ambiente apropriadamente
- Sistemas bem modularizados são fáceis de isolar

Testabilidade (2/2)

- Não é uma métrica intrínsica do sistema
- É necessário várias medidas para criar uma métrica de testabilidade
- É necessário cautela ao se comparar a testabilidade de sistemas
  - Contextos diferentes
  - Linguagens diferentes
  - Complexidade diferente

Testability-Explorer

- http://code.google.com/p/testability-explorer
- OOPSLA 2008
- Verifica testabilidade de programas Java
  - Usa ASM para análise de *bytecodes*
- Relatórios:
  - http://www.testabilityexplorer.org/report

Testability-Explorer

- Opções:
  - maxAcceptableCost, maxEcellentCost, minCost
- Maven:

```
<reporting><plugins><plugin>
    <groupId>com.google.testability-explorer</groupId>
    <artifactId>maven-testability-plugin</artifactId>
    <version>1.3.2-SNAPSHOT</version>
</plugin></plugins></reporting>
```

Testability-Explorer

- Demonstração

O quê a Testability-explorer verifica

- Non-Mockable Total Recursive Cyclomatic Complexity
- Global Mutable State
- Law of Demeter

Indícios para melhorias: Construtores

- Instanciar na declaração de um campo ou no construtor
- Chamada de métodos estáticos nos contrutores ou na declaração dos campos
- Qualquer coisa além de instanciação de variáveis no construtor
- Objeto não inicializado apropriadamente após o a execução do construtor
- Fluxo de controle no construtor
  - Use padrões de criação: Builder, Factory ...

Indícios para melhorias: Colaboradores

- Colaboradores não usados diretamente
- Violação da Lei de Demeter
  - a.getX().getY().getZ()...
- Nomes suspeitos:
  - Manager, context, environment, principal, container, principal, runner

Indícios para melhorias: Estados globais

- Singletons
- Métodos ou campos estáticos
- Blocos de inicialização estáticos
- Uso de registros e localizador de serviços

- Testes não isolados
- Testes não paralelizáveis

Indícios para melhorias: Responsabilidades

- Classe com mais de uma responsabilidade
- Classe difícil de entender a responsabilidade
- Classe com campos raros de serem usados
- Com métodos estáticos que só operam os parâmetros
  - Math.abs(-1)

Boa Testabilidade

- Orientação a Objeto
  - Programar modularmente com linguagens OO não resolve
- Não pergunte, diga!
  - Injeção de dependência
- Evite variáveis globais
  - Evite Singletons

OO

- Single Responsibility Principle (SRP)
- Open Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

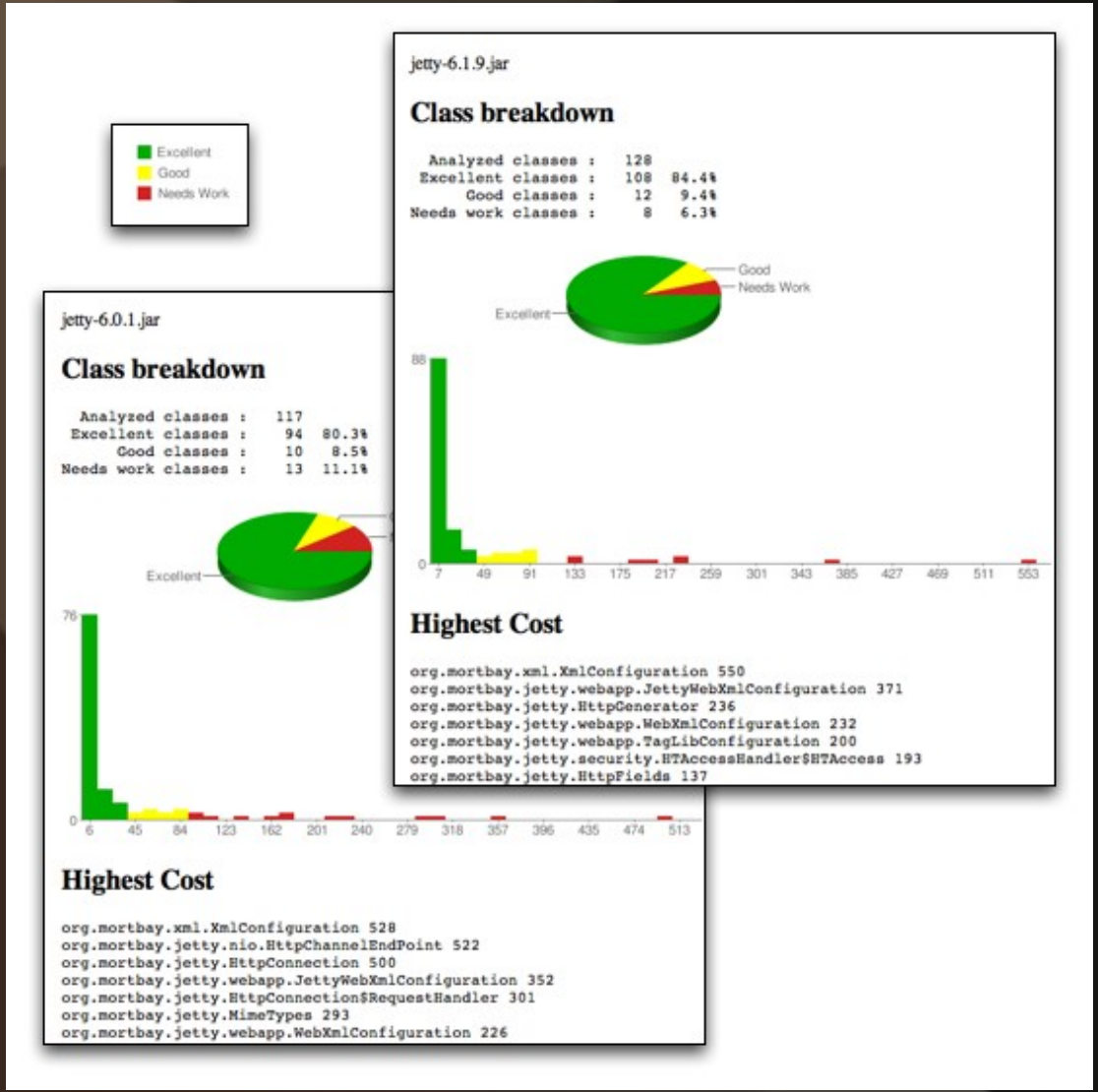
TDD e TFD

- Test Driven Development
- Test First Development

- Testes não são deixados de lado
- Design emerge dos testes

Relatórios

- Custos
  - Excelente
  - Bom
  - Precisa melhorar
- Customizável



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:
  - Mockito (Java): http://code.google.com/p/mockito
  - 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

Contato

paulocheque@gmail.com

Licença:

Creative Commons: Attribution-Share Alike 3.0 Unported

http://creativecommons.org/licenses/by-sa/3.0/