Testes Automatizados

Paulo Cheque

AgilCoop

27/01/2009 Verão 2009

Erros de Software

Causam prejuízos de aproximadamente $59.5 bi na economia dos Estados Unidos

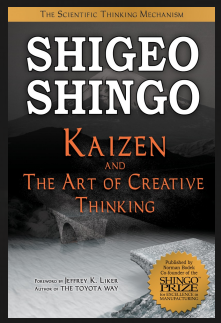
Fonte: NIST/2002 - http://www.nist.gov

Estratégias

- Desenvolvimento com qualidade
- Setor de Homologação
- Controle de Qualidade
- Análises formais
- Feedback de usuários:
  - Versões alfa e beta
  - Produção

Como utilizar o tempo?

"Inspecionar para previnir defeitos é bom; Inspecionar para encontrar defeitos é desperdício" - Shigeo Shingo



Objetivos:

- Diminuir tempo gasto com depuração
- Aumentar o tempo gasto com verificação

Práticas de Verificação

- Revisões de código
- Análises Formais
- Programação Pareada
- Testes (unidade, integração, ...)
  - Manuais
  - Automatizados

EXEMPLO

Teste Manual

- Difícil repetição, demorado, cansativo
- Executado poucas vezes, poucos casos e casos simples
- Sem documentação ou documentação adicional e obsoleta

- Software/Ambiente exige manutenção?

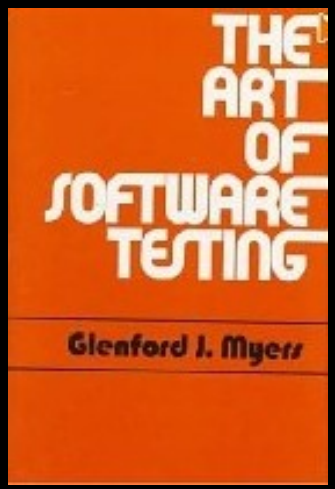
  ⟶ Regressão

Exemplo Testes Automatizados

- TODOS os testes podem (devem) ser executados a qualquer momento
- Reprodutibilidade
- Casos complexos
- Certificação do que foi testado
- Mais segurança na manutenção

História

- Até 1956 – Orientado a depuração
- 1957-1978 – Orientado a demonstração
- 1979-1982 – Orientado a destruição
- 1983-1987 – Orientado a avaliação
- 1988-???? – Orientado a prevenção



História...

1959/1963: NASA - Projeto Espacial Mercúrio

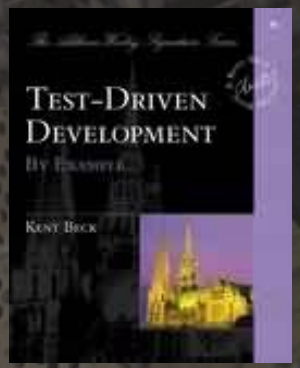
1989: FIT-like Framework

1991: Taligent Framework

1994: SUnit

1998: JUnit

2002: Test-Driven Development by Example



http://www.sscnet.ucla.edu/geog/gessler/

Hoje...


Robert C. Martin Series
Fit for Developing Software
Framework for Integrated Tests
XUnit Test Patterns
Refactoring Test Code
Gerard Meszaros
Refactoring
Improving the Design of Existing Code
Testing Extreme Programming
Lisa Crispin
Tip House
test-driven development
A Practical Guide
Refactoring to Patterns
Testing Object-Oriented Systems
Models, Patterns, and Tools
Robert V. Binder
Test-Driven Development
By Example
Kent Beck
Software Testing
Introdução ao Teste de Software
Márcio Eduardo Delamaro
José Carlos Maldonado
Mario Jino

Hoje...

- Muito software livre de qualidade
- Condição para novos tipos de testes
- DSL
- Otimização:
  - JUnitMax
  - Selenium-Grid
  - Automated Continuous Testing
- Novas métricas: Cobertura, Testabilidade..
- Estratégias, Padrões e Anti-Padrões

Academia e Indústria

- Estudos de Métodos Ágeis
- Estudos específicos de Testes Automatizados
- Cursos
- Internet: blogs, wikis, listas, forums

Pesquisa: AgilCoop - CNPq

Região	Visitas previstas	V
Grande São Paulo	8	10
Campinas/Hortolândia	3	4
São Carlos	3	3
Rio de Janeiro	4	5

Número de funcionários	Número de empresas
De 1 a 10	1
De 11 a 100	9
De 101 a 1000	7
Mais de 1000	5

Tabela 2: Resultados acumulados do número de funcionários

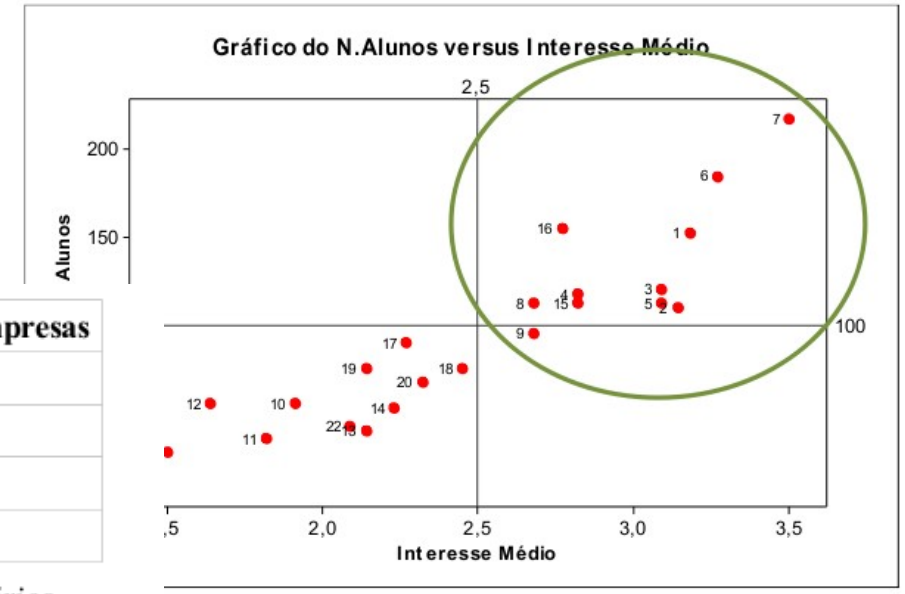


Gráfico 01. Gráfico do Interesse versus Número de Alunos.

Tempo no merca
Menos de 5 anos
De 5 a 10 anos
De 10 a 15 anos
De 15 a 20 anos
De 20 a 25 anos
De 25 a 30 anos
Mais de 30 anos

Tabela 3: Resultados a

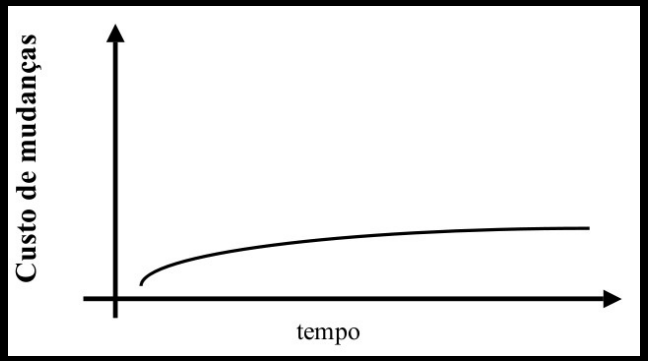
Nº curso	Curso	Média aritmética	Posição
7	Desenvolvimento de Software de Qualidade através de Testes Automatizados	1,0	1
6	Práticas de Métodos Ágeis para o Dia-a-dia dos Programadores	2,0	2
1	Introdução a Métodos Ágeis de Desenvolvimento de Software	3,5	3
3	Liderança Ágil de Projetos de Software	5,0	4
16	Padrões de Projeto (Design Patterns) e Princípios de Orientação a Objetos	6,0	5
4	Gestão Ágil de Projetos com Scrum	6,5	6
5	Planejamento e Estimativas Ágeis em Projetos de TI	6,5	7
2	Desenvolvimento Ágil de Software para Gerentes e Administradores	7,0	8
15	Desenvolvimento Dirigido por Testes	8,5	9
8	Laboratório Prático de Desenvolvimento Ágil de Software	9,0	10

Experimentos

- Produtividade e Qualidade
- Difícil isolação de outras variáveis
  - Experiência técnica
  - Tecnologias
  - Metodologias
- Resultados conflitantes

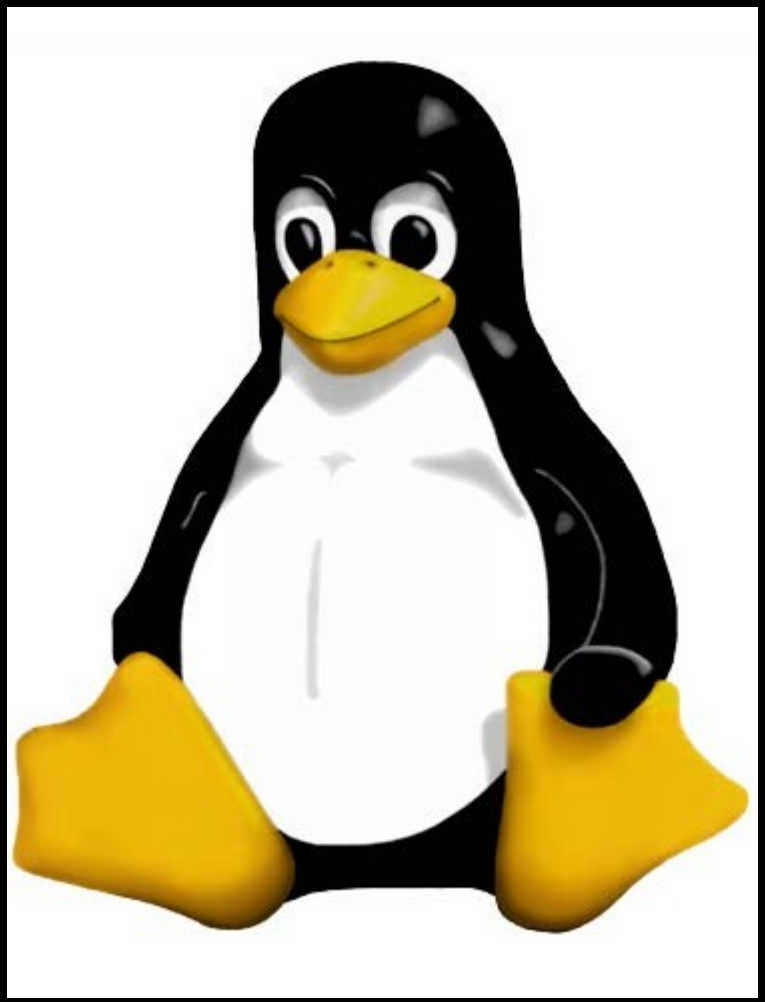
Métodos Ágeis

- Desenvolvimento Incremental
- Constante manutenção
  - Refatoração
  - Design Incremental
  - Entregas frequentes
- Foco no time
  - Código compartilhado
  - Integração contínua
- Envolvimento real com o cliente



Software Livre

- Equipe:
  - Mantenedores
  - Colaboradores
- Contribuições:
  - Distantes
  - Desconhecidas
  - Heterogêneas
- É seguro sem testes?

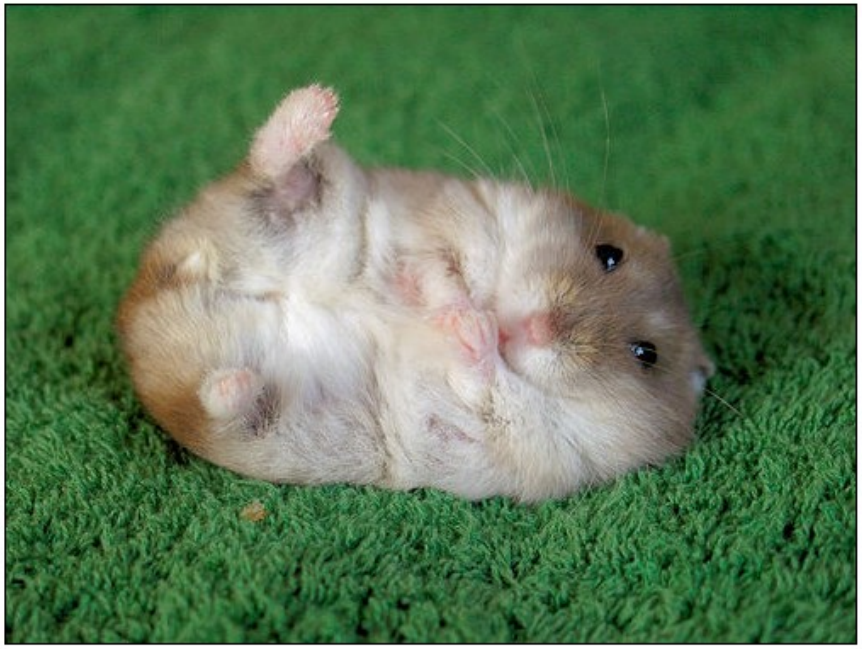


Qualidade?

ENQUETE

Ocorreu um erro interno no processamento da sua requisição.

Por favor tente novamente mais tarde.



Our hamster powered servers are too busy and begging for few seconds of rest.

```
-e:3:in `load': /usr/bin/rails:4: syntax error, unexpected '=', expecting ']' (SyntaxError)
if [ "x$1" = "x" ]; then
           ^
/usr/bin/rails:4: syntax error, unexpected ']', expecting $end
if [ "x$1" = "x" ]; then
                ^     from -e:3
```

Software com Qualidade

Correção
Eficiência
Segurança
Durabilidade
Usabilidade
Portabilidade

Flexibilidade
Robustez
Manutenibilidade
Acessibilidade
Beleza
...

O que testes automatizados tem a ver com tudo isso?

Correção

```
public aspect AspectScreenshotByAnnotation {
    // Any method annotated or any class annotated
    pointcut annotationHandler():
        if(System.getProperty("selenium.screenshot") != null &&
            System.getProperty("selenium.screenshot").equals("true")) &&

        // Method
        (execution(@br.com.agilbits.util4selenium.annotations.Screenshot * *(..)) ||

        // Class
        (execution(* (@br.com.agilbits.util4selenium.annotations.Screenshot *).*(..)) &&
        !execution(public * selenium())));

    after() returning(): annotationHandler() {
        // ...
    }
```

Evitando StackOverFlow

Qualquer erro é desastroso

Eficiência

- Desempenho/Estresse/Carga
- Como simular manualmente grande quantidade de dados/usuários?
- Como medir o tempo?
- Quando buscar gargalos?

- A maioria das ferramentas de testes automatizados fornecem o tempo de execução.

Segurança

- Atualização de servidores
- Mudanças de *queries*, interface...

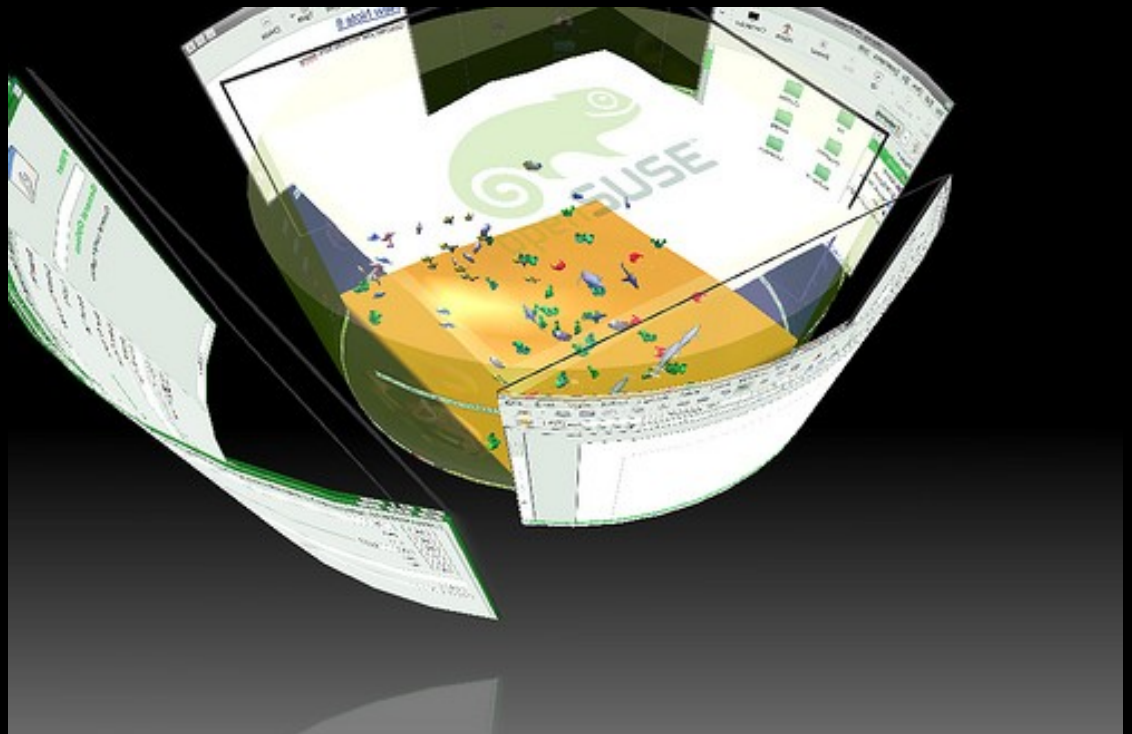
Erros de regressão são testados?

Durabilidade

- Falta de manutenção
- Erros inibem os usuários
- Ciclo de erros de regressão
- Sistemas legados

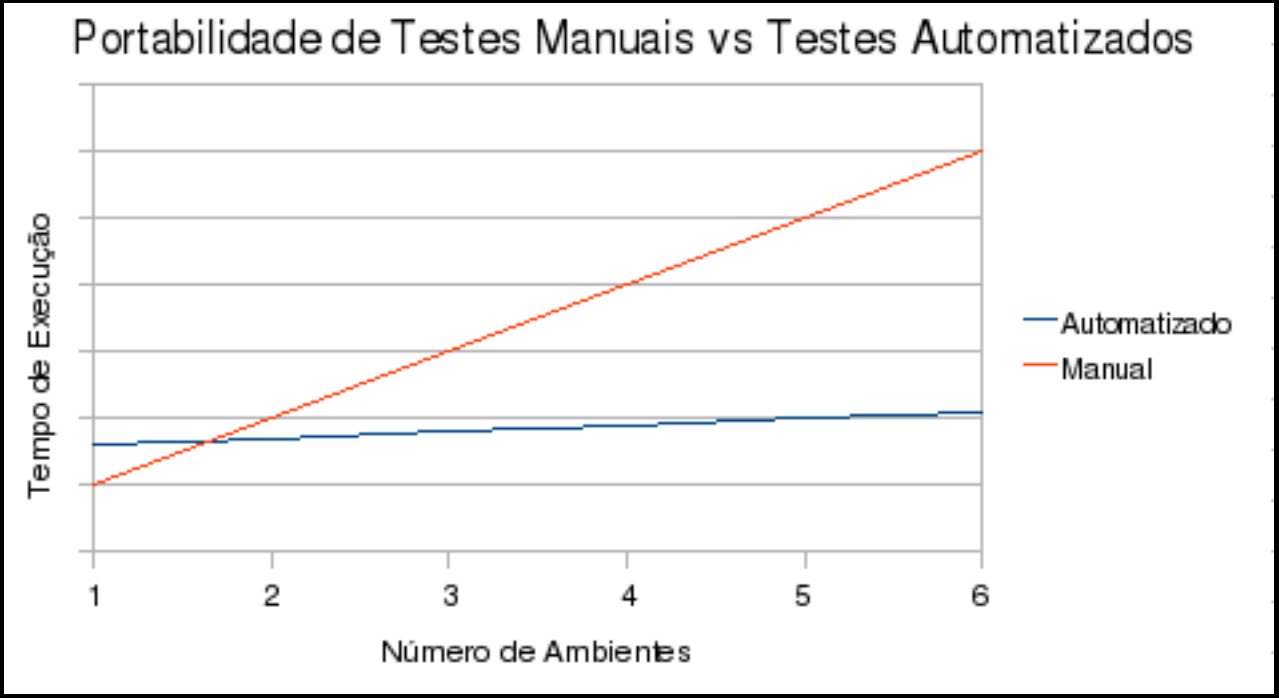
Usabilidade

- Teste de interface
- Necessidade de ferramentas
- Heurísticas de Interação Humano-Computador



Portabilidade

Sistemas Operacionais, Navegadores...



Flexibilidade

TDD: Test-Driven Development/Design

Single Responsibility Principle (SRP)

Open Closed Principle (OCP)

Liskov Substitution Principle (LSP)

Interface Segregation Principle (ISP)

Dependency Inversion Principle (DIP)

Robustez

Confiabilidade para mudanças

Testes em diversas plataformas

Maior Flexibilidade

Manutebilidade

- Refatoração / Otimização
- Correção
- Adição de novas funcionalidades

Não encosta no que está funcionando!

É só colocar um if(obj != null)!

Acessibilidade

- Testes de interface
  - Teclado
  - Mouse
- Testes de layout
  - Tamanho das letras

Beleza

- Designers trabalham com a interface
- Layout


```
@Screenshot(policy = ScreenshotPolicy.ALWAYS)
public class SomeClassAnnotatedAlways implements SeleniumClass {
    private Selenium selenium;
    public Selenium selenium() { return selenium; }
}
```

Princípios Básicos

- Código dos testes merecem a mesma atenção do código do sistema
- Precisam de Manutenção
- Testes podem conter erros
- O código **precisa** ser simples
- Devem ser o mais legível quanto possível
  - Documentação
- Não devem exigir intervenção humana

Os Primeiros Testes de Unidade

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

Main

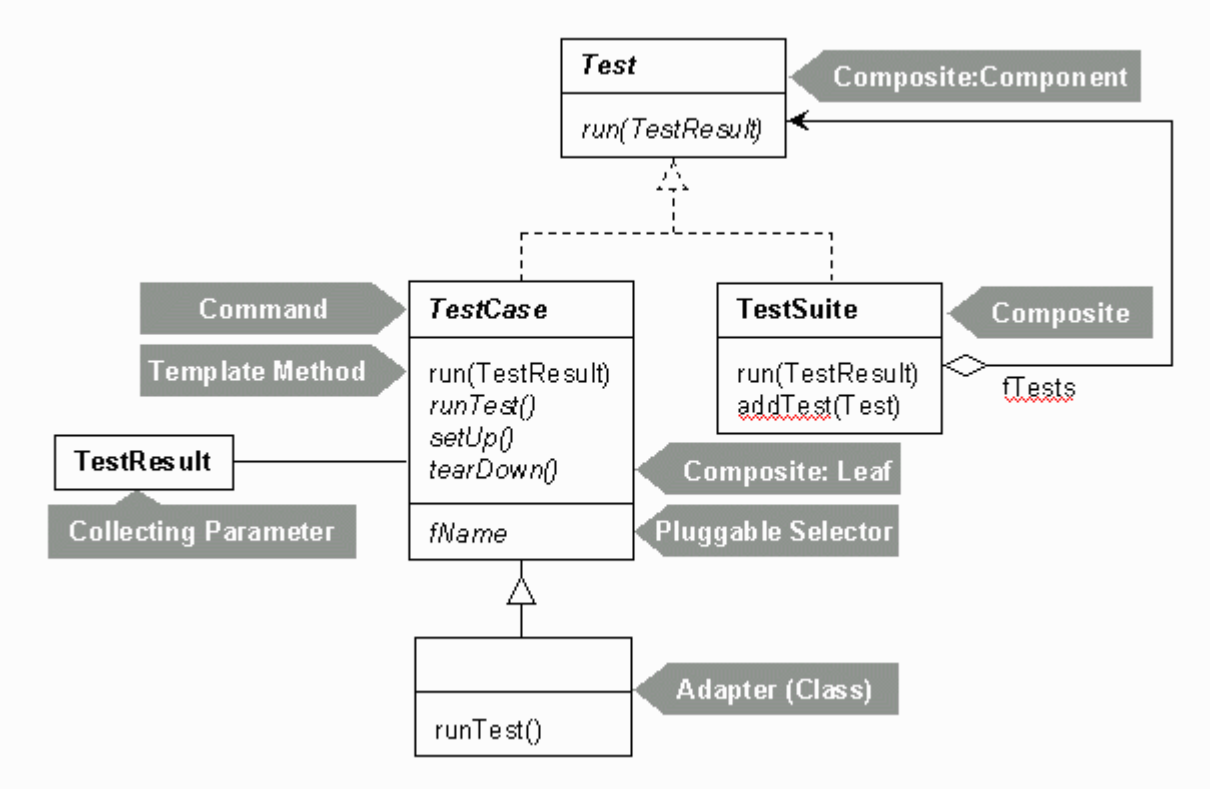
```
public class MainExample {


    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

EXEMPLO

- Mensagens amigáveis

- Hamcrest: Matchers + AssertThat

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

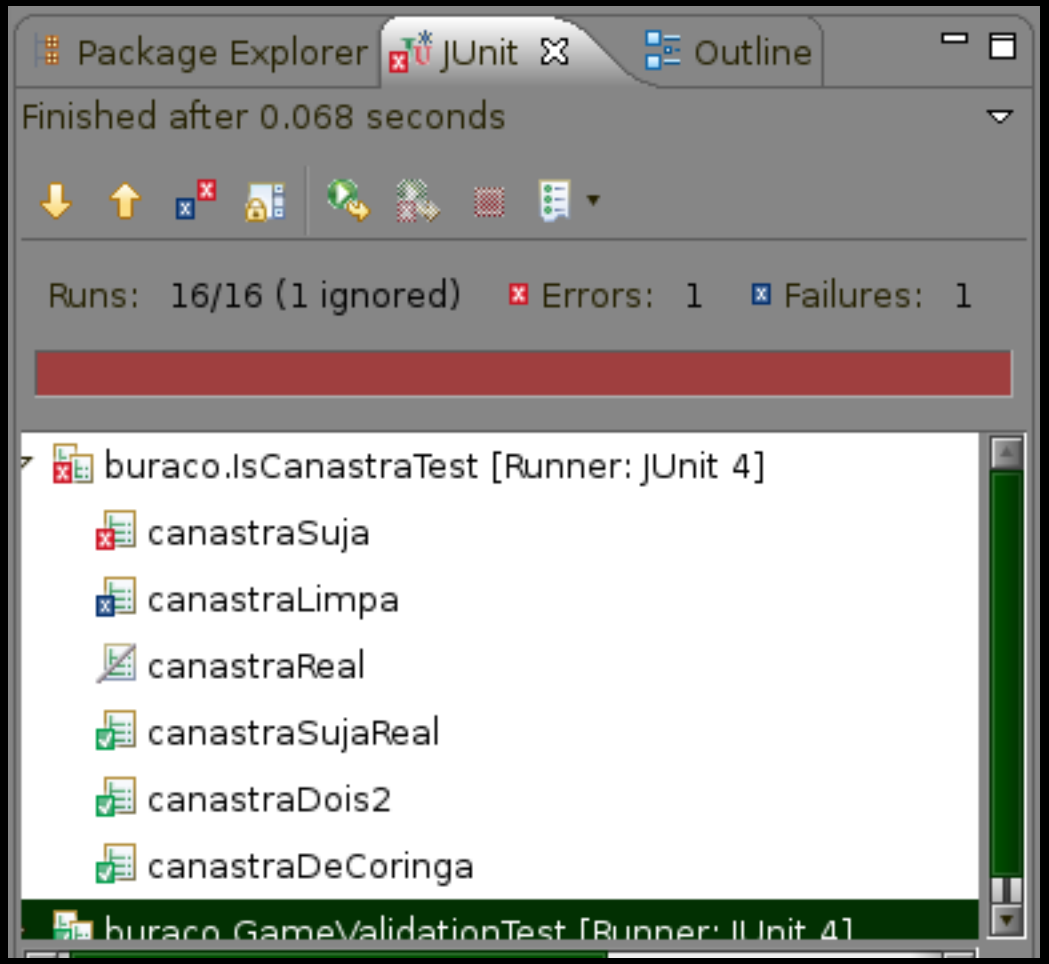
EXEMPLO

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

Resultados

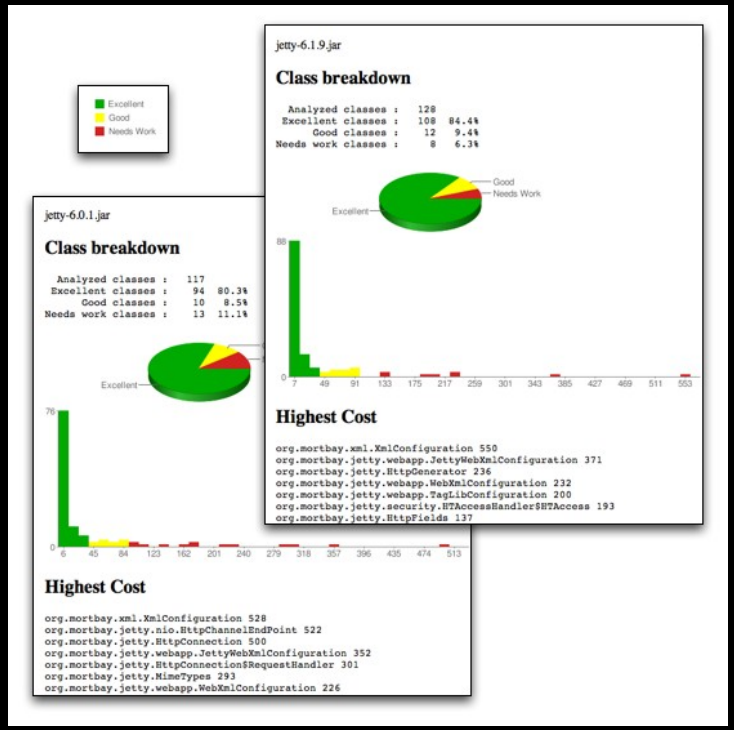
- Skip
- Fail
- Error
- Success

EXEMPLO



Relatórios

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



...

- Testes Manuais
- Testes Automatizados
- História
- Métodos Ágeis
- Software Livre
- Qualidade
- Exemplos

...

“Qualquer funcionalidade que não possui testes automatizados simplesmente não existe” - Kent Beck

Contato

http://www.agilcoop.org.br

agilcoop@agilcoop.org.br

paulocheque@agilcoop.org.br