

# Desenvolvimento de Software de Qualidade através de Testes Automatizados

Paulo Cheque ([paulocheque@agilcoop.org.br](mailto:paulocheque@agilcoop.org.br))

Cursos de Verão 2010

Licença:

Creative Commons: Attribution-Share Alike 3.0 Unported

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



# Erros de Software

- Erros de software custam cerca de US\$59,5 bi à economia dos Estados Unidos
  - Fonte: NIST/2002 - <http://www.nist.gov>



[Advanced Client Use \(SSH, The Secure Shell: The Definitive Guide\)](#) - [ [Traduzir esta página](#) ]

[Este site pode danificar seu computador.](#)

Several client features are controlled by **environment variables**. .... Client **configuration files** come in two flavors. A single, global client **configuration ...**  
[docstore.mik.ua/oreilly/networking\\_2ndEd/ssh/ch07\\_01.htm](http://docstore.mik.ua/oreilly/networking_2ndEd/ssh/ch07_01.htm) - [Páginas Semelhantes](#)

[Passing Environment Variables In A Config File: msg#00027 gnu ...](#) - [ [Traduzir esta página](#) ]

[Este site pode danificar seu computador.](#)

Passing **Environment** Varia - Find Help in our gnu.screen.user Forum.  
[osdir.com/ml/gnu.screen.user/2006-08/msg00027.html](http://osdir.com/ml/gnu.screen.user/2006-08/msg00027.html) - [Páginas Semelhantes](#)

[Re: \[CFG\] Environment variables vs. configuration files](#) - [ [Traduzir esta página](#) ]

[Este site pode danificar seu computador.](#)

Re: [CFG] **Environment variables vs. configuration files**. Denis Barbier Thu, 27 Jan 2000 05:53:28 -0800. On Thu, 27 Jan 2000, Denis B. Roegel wrote: > `Denis ...  
[www.mail-archive.com/tetex@informatik.uni-hannover.de/msg00104.html](http://www.mail-archive.com/tetex@informatik.uni-hannover.de/msg00104.html) - [Páginas Semelhantes](#)

[\[CFG\] Environment variables vs. configuration files](#) - [ [Traduzir esta página](#) ]

[Este site pode danificar seu computador.](#)

[CFG] ...  
200

[www](#)

[Pági](#)

[Mais](#)

**Aviso- acessar este site pode danificar o seu computador!**

**Sugestões:**

- [Volte à página anterior](#) e selecione outro resultado.
- Tente pesquisar outros termos para encontrar o que você está procurando.

Ou continue a <http://www.mail-archive.com/tetex@informatik.uni-hannover.de/msg00104.html> por sua conta e risco. Para obter informações detalhadas sobre os problemas que encontramos, visite a [página de diagnóstico Navegação segura](#) do Google para este site.

Para obter mais informações sobre como se proteger contra softwares prejudiciais on-line, visite [StopBadware.org](http://StopBadware.org).

Se você é o proprietário deste site, pode solicitar uma revisão do site usando as [Ferramentas para webmaster](#) do Google. Informações adicionais sobre o processo de revisão estão disponíveis na [Central de Ajuda para webmasters](#) do Google.

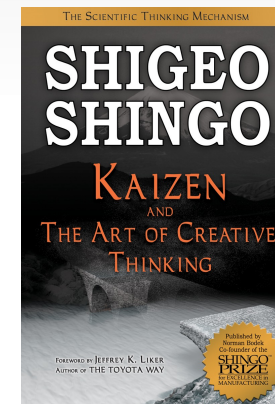
Aviso gerado por [Google](#)

# Como Melhorar a Qualidade?

- Identificação de erros:
  - Departamentos de Homologação
  - Controle de Qualidade
  - Análises Formais
  - *Feedback* de usuários:
    - Versões alfa e beta
    - Produção
- **Prevenção de erros**
  - **Desenvolvimento com qualidade**

# Como utilizar o tempo?

- *“Inspeccionar para procurar defeitos é desperdício, inspeccionar para prevenir defeitos é essencial”*
  - Shigeo Shingo
- Objetivos:
  - Reduzir tempo gasto com depuração
  - Aumentar tempo gasto com verificação



# Práticas de Verificação

- Abordagens tradicionais
  - Revisão de código
  - Inspeção de documentos
  - Testes manuais
- Abordagens caras e complexas
  - Análises formais
- **Abordagens ágeis**
  - **Programação em pares**
  - **Testes Automatizados**



# Testes Manuais

- Difícil de repetir, lentos e chatos
- Poucos casos de testes simples
- Executado poucas vezes
- Documentação inexistente ou obsoleta
- Erros de regressão

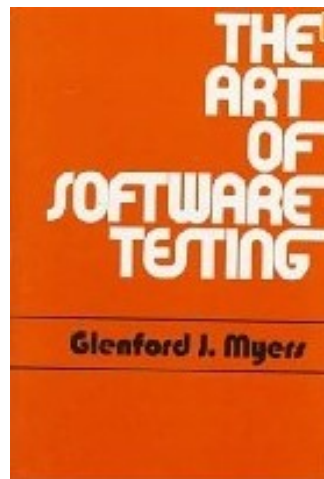
# Testes Automatizados

- TODOS testes podem (devem) ser executados a qualquer momento
- Certificação do que foi testado
- Seguro para manutenção
- Reprodutibilidade: facilita identificação de erros
- Casos complexos

# História (1/2)

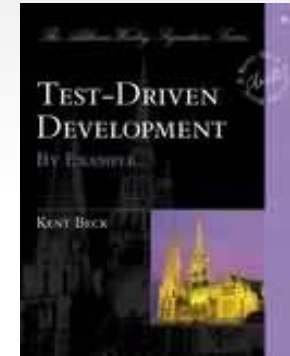
- Até 1956 – Orientado a depuração
- 1957-1978 – Orientado a demonstração
- 1979-1982 – Orientado a destruição
- 1983-1987 – Orientado a avaliação
- 1988-???? - Orientado a prevenção

(1988)



# História (2/2)

- 1959/1963 – NASA: Projeto espacial Mercúrio
- 1989 – FIT-Like Framework
- 1991 – Taligent Framework
- 1994 – SUnit
- 1998 – JUnit
- 2002 – Test Driven Development by Example








# Hoje























# Hoje...

- Java/Maven
- Ruby/Rails
- Groovy/Grails
- Scala/Lift
- .....

- ▷  src/main/java
-  src/main/resources
- ▷  src/test/java
-  src/test/resources

-  src/main/resources
- ▷  src/main/scala
-  src/test/resources
- ▷  src/test/scala

-  src/java
-  src/groovy
- ▷  grails-app/conf
-  grails-app/controllers
- ▷  grails-app/domain
-  grails-app/services
-  grails-app/taglib
-  test/integration
- ▷  test/unit

- ▽  rails
  - ▷  app
  -  components
  - ▷  config
  -  db
  - ▷  doc
  - ▷  lib
  - ▷  log
  - ▷  public
  - ▷  script
  - ▽  test
    -  fixtures
    - ▷  functional
    -  integration
    - ▷  mocks
    - ▷  unit
    -  test\_helper.rb 57!

# Tendências

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

# Academia e Indústria

- Pesquisas em métodos ágeis
- Pesquisa em automação de testes
- Cursos
- Internet: Blogs, wikies, listas, forums



# Pesquisa: AgilCoop - CNPq

Região	Visitas previstas	Visitas realizadas
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

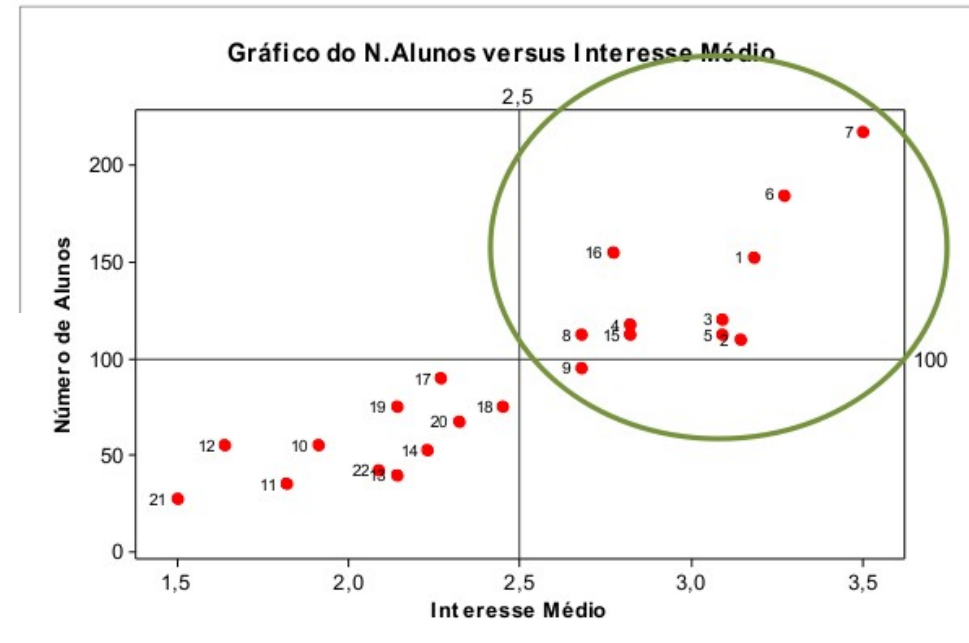


Gráfico 01. Gráfico do Interesse versus Número de Alunos.

Tempo no mercado	Nº curso	Curso	Média aritmética	Posição
Menos de 5 anos	7	Desenvolvimento de Software de Qualidade através de Testes Automatizados	1,0	1
De 5 a 10 anos	6	Práticas de Métodos Ágeis para o Dia-a-dia dos Programadores	2,0	2
De 10 a 15 anos	1	Introdução a Métodos Ágeis de Desenvolvimento de Software	3,5	3
De 15 a 20 anos	3	Liderança Ágil de Projetos de Software	5,0	4
De 20 a 25 anos	16	Padrões de Projeto (Design Patterns) e Princípios de Orientação a Objetos	6,0	5
De 25 a 30 anos	4	Gestão Ágil de Projetos com Scrum	6,5	6
De 30 a 35 anos	5	Planejamento e Estimativas Ágeis em Projetos de TI	6,5	7
Mais de 30 anos	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

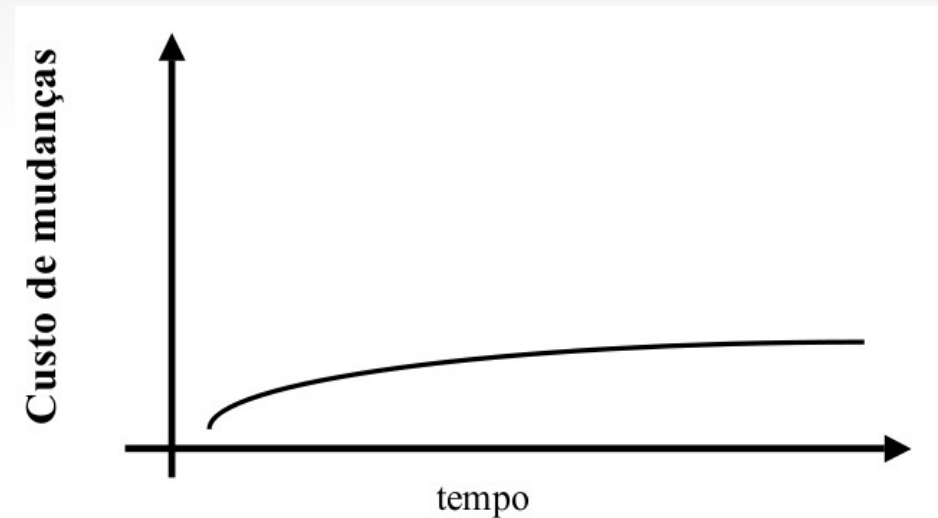
Tabela 3: Resultados a

# Resistência

- Mudanças culturais são difíceis de serem realizadas
  - Pessoas não aceitam mudanças com tranquilidade, mesmo quando há falhas recorrentes com o trabalho
- Falta de evidência que novos métodos funcionam servem de desculpa
- É necessário testar (ágil, testes) na prática!

# Métodos Ágeis e Testes Automatizados

- Desenvolvimento incremental
- Manutenção constante
  - Refatorações
  - Design incremental
  - Pequenos *releases*
- Foco no time
  - Código compartilhado
  - Integração contínua
- Involvimento real com clientes e usuários



# Software Livre

- Time
  - Donos de projetos
  - Colaboradores
- Contribuições
  - Distantes
  - Pessoas desconhecidas
  - Heterogêneas
- É seguro sem testes automatizados?



# Qualidade?




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

ENQUETE

Ocorreu um erro interno no processamento da sua requisição.

Por favor tente novamente mais tarde.



12X R\$149,92 CHLEBR

Frete Grátis shop time

fecha

```
-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 de Qualidade

- Correção
- Robustez
- Eficiência
- Segurança
- Durabilidade
- Flexibilidade
- Portabilidade
- Manutenibilidade
- Beleza
- Usabilidade
- Acessibilidade
- ...



# 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
  - precisa executar todos os testes novamente
- Mudanças de queries, interface...
  - precisa executar todos os testes novamente
- Erros de regressão são testados?
  - precisam ser!

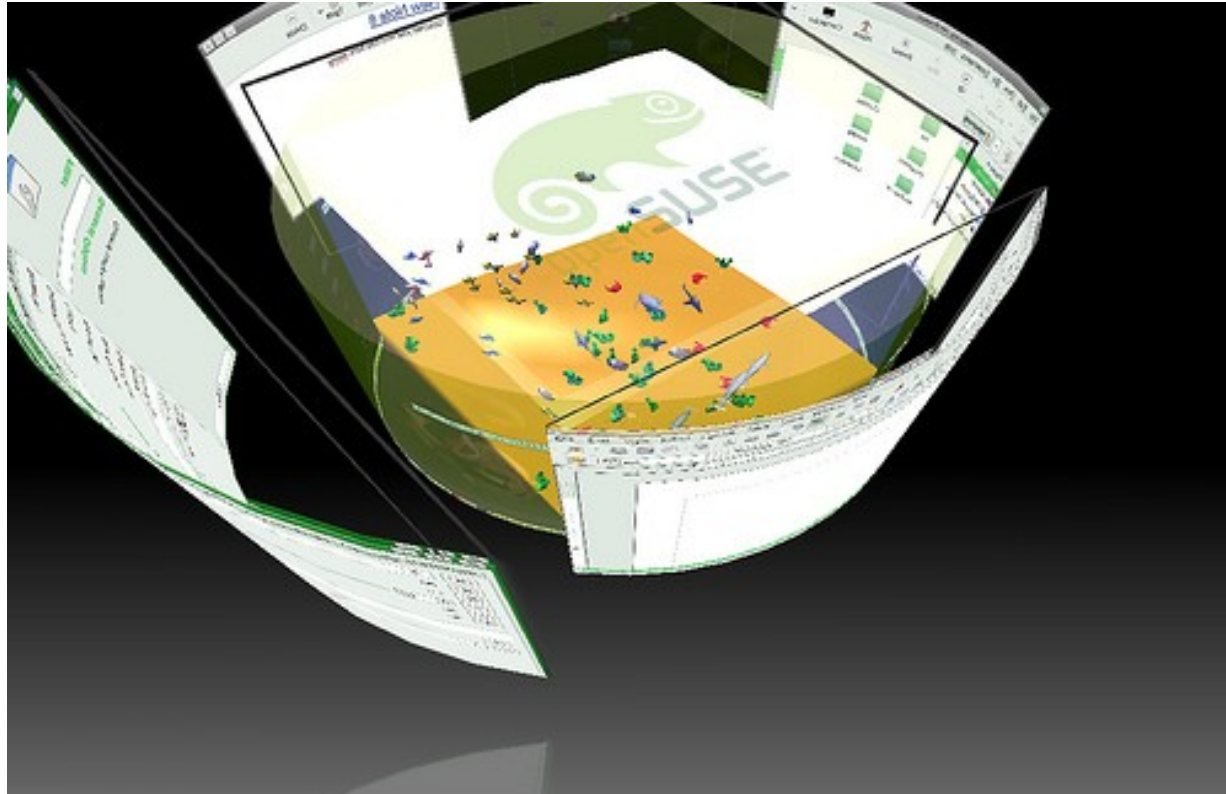


# Durabilidade

- Software sem manutenção morre
  - “Não mexe porque está funcionando”
  - “Não funciona mas acho melhor deixar assim pois se formos mexer piora”
- Erros inibem os usuários
- Ciclo de erros de regressão
  - corrige um erro e adiciona mais três

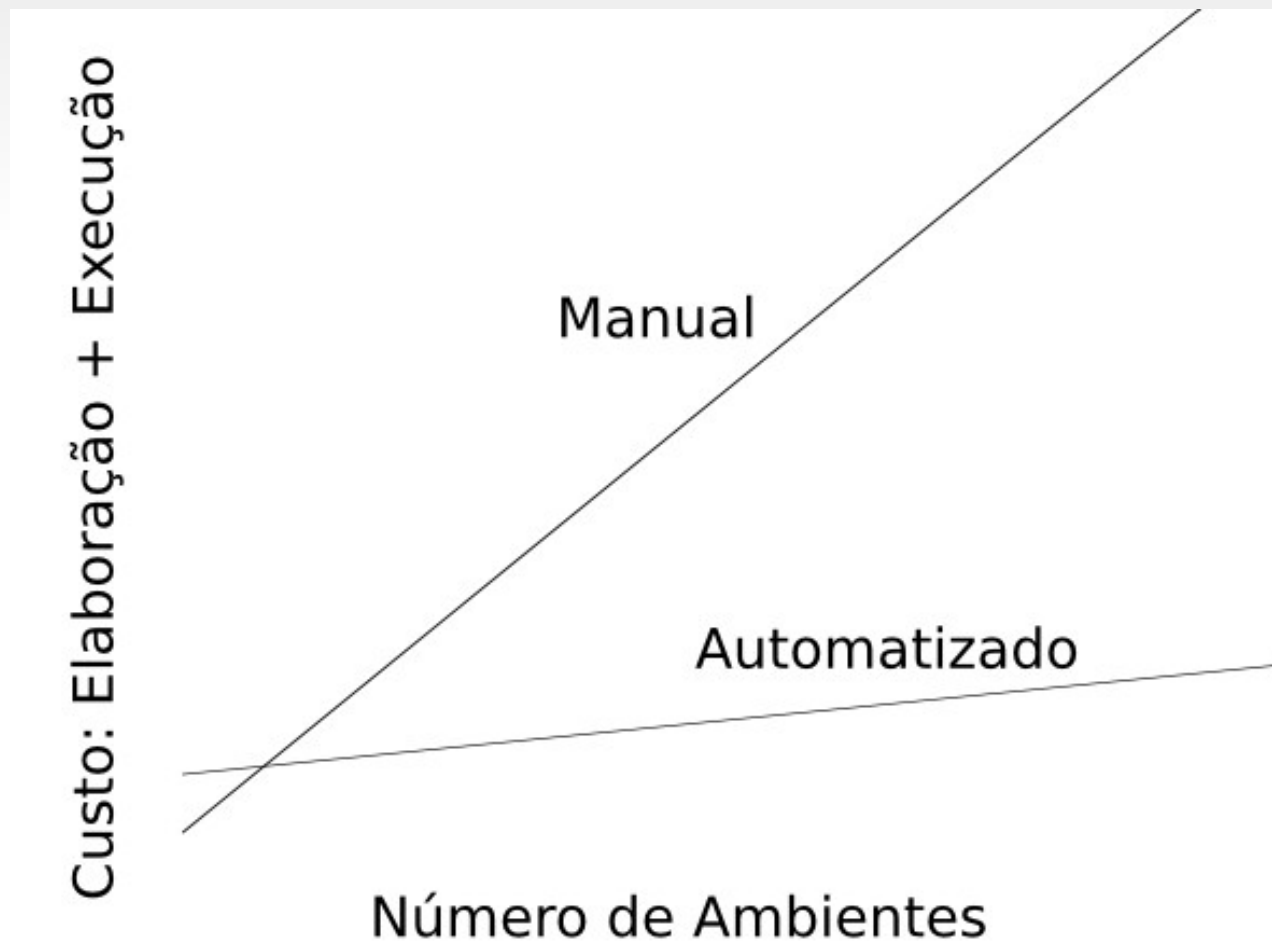
# Usabilidade

- Teste de interface
- Ainda há poucas 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



# Manutenibilidade

- 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



Google Chrome

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

# Recapitulando

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

# Palavras Finais

- *“Qualquer funcionalidade sem testes automatizados não existe”*
  - Kent Beck

# Contato

<http://www.agilcoop.org.br>

<http://ccsl.ime.usp.br>

[paulocheque@agilcoop.org.br](mailto:paulocheque@agilcoop.org.br)

Licença:

Creative Commons: Attribution-Share Alike 3.0 Unported

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

