Introdução a Testes Automatizados

AgilCoop – Cursos de Verão 2010

Mariana Bravo
IME/USP

AgilCoop

Introdução

- Testes são uma forma de *feedback*:
  a funcionalidade que fizemos funciona?

- Testes como uma medida de qualidade:
  quantos *bugs* consigo encontrar testando esta funcionalidade pronta?
  - Manuais
  - Tardios

Testes automatizados

- Testes executados por um computador, seguindo um roteiro e fazendo verificações especificadas pelo testador

- Exemplos:
  - Um gravador de interações com a interface gráfica de um programa
  - Um programinha que acessa e exercita pedaços do sistema, informando caso encontre algum erro

“Inspecionar para prevenir defeitos é bom;
inspecionar para encontrar defeitos
é desperdício" – Shigeo Shingo

Objetivos – Criar valor

- Melhorar a qualidade
- Entender o sistema em teste
- Diminuir (e não introduzir) risco

Melhorar a qualidade

Testes automatizados devem...

- Atuar como especificação das funcionalidades

Melhorar a qualidade

Testes automatizados devem...

- Atuar como repelente de *bugs*

Melhorar a qualidade

Testes automatizados devem...

- Ajudar a achar defeitos no sistema em teste

Entender o sistema em teste

Testes automatizados devem...

- Atuar como documentação executável

Diminuir (e não introduzir) risco

Testes automatizados devem...

- Atuar como rede de segurança para mudanças

Diminuir (e não introduzir) risco

Testes automatizados devem...

- Não fazer mal ao sistema em teste

Objetivos – Criar valor

Testes automatizados devem...

- Atuar como especificação das funcionalidades
- Atuar como repelente de *bugs*
- Ajudar a achar defeitos no sistema em teste
- Atuar como documentação executável
- Atuar como rede de segurança para mudanças
- Não fazer mal ao sistema em teste

Conceitos e Exemplos

- O teste mais simples
- Caso de teste
- Verificação
- Preparar o ambiente
- Limpar o ambiente
- Exceções
- Resultados
- Relatórios

Tipos de testes

- Unidade
  - Verificam comportamento de unidades do sistema separadamente

Tipos de testes

- Unidade
- Integração, interface do usuário, aceitação
  - Verificam comportamento de partes do sistema, inclusive interação entre elas.
  - Os testes de aceitação verificam resultados da lógica de negócios do ponto de vista do cliente; não necessariamente interagem com a interface do usuário

Tipos de testes

- Unidade
- Integração, interface do usuário, aceitação
- Mutação
  - Testes que verificam os próprios testes, fazendo pequenas mudanças no código para ver que ele é coberto pelos testes

Tipos de testes

- Unidade
- Integração, interface do usuário, aceitação
- Mutação
- Desempenho, carga, estresse, segurança
  - Testes que verificam aspectos não-funcionais do programa, em geral através de ferramentas específicas

Características desejáveis

- Fáceis de executar
  - Totalmente automatizados
  - Verificar o próprio resultado
  - Poder repetir
  - Serem independentes

Características desejáveis

- Fáceis de executar
- Fáceis de escrever e manter
  - Simples
  - Expressivo
  - Separação de interesses

Características desejáveis

- Fáceis de executar
- Fáceis de escrever e manter
- Precisar de pouca manutenção
  - Robustos

Alguns princípios

- Escreva os testes primeiro

Alguns princípios

- Escreva os testes primeiro
- Projete para testabilidade

Alguns princípios

- Escreva os testes primeiro
- Projete para testabilidade
- Comunique a intenção

Alguns princípios

- Escreva os testes primeiro
- Projete para testabilidade
- Comunique a intenção
- Não modifique o sistema em teste

Alguns princípios

- Escreva os testes primeiro
- Projete para testabilidade
- Comunique a intenção
- Não modifique o sistema em teste
- Verifique uma condição por teste

Alguns princípios

- Escreva os testes primeiro
- Projete para testabilidade
- Comunique a intenção
- Não modifique o sistema em teste
- Verifique uma condição por teste
- Promova equilibro entre testes e produção

Perguntas?

Mariana Bravo
marivb@agilcoop.org.br

Mais princípios

- Use primeiro a porta da frente
- Isole o sistema em teste
- Mantenha os testes independentes
- Minimize a sobreposição de testes
- Minimize código não-testável
- Mantenha a lógica de teste longe do código de produção
- Teste assuntos separadamente

Referências – Livros

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

Referências – Ferramentas

- Testes de unidade
  - Java:
    - JUnit em http://www.junit.org
    - TestNG em http://testng.org
  - Ruby: RUnit em http://www.ruby-doc.org/stdlib/libdoc/test/unit/rdoc/classes/Test/Unit.html
  - Python: PyUnit em http://pyunit.sourceforge.net/
  - C++: CppUnit2 em https://launchpad.net/cppunit2
  - C: CUnit em http://cunit.sourceforge.net/

Referências – Ferramentas

- Testes de interface gráfica
  - Marathon em http://www.marathontesting.com/
  - SWTBot em http://swtbot.sourceforge.net/
- Testes de interface web
  - Selenium em http://seleniumhq.org/
  - Webrat em http://wiki.github.com/brynary/webrat/
- Testes de aceitação
  - Fit em http://fit.c2.com/
  - Cucumber em http://cukes.info/

Referências – Ferramentas

- Testes de mutação:
  - Jester em http://jester.sourceforge.net/
  - Heckle em http://ruby.sadi.st/Heckle.html
- Testes de desempenho/estresse:
  - JMeter em http://jakarta.apache.org/jmeter

Referências – Links úteis

- http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks
- http://www.opensourcetesting.org/
- http://xunitpatterns.com/
- http://mockobjects.com/
- http://java-source.net/open-source/testing-tools