Universidade Federal Fluminense

Teste de Unidade e Funcional

Wellington Silva

Universidade Federal Fluminense

Sumário

- Introdução
- Justificativa
- Objetivo
- Tipos de Testes
- Teste de Unidade
  - Junit
- Teste de Performance
  - JUnitPerf
- Funcional
  - Selenium
- Conclusão

Universidade Federal Fluminense

Introdução

Universidade Federal Fluminense

Introdução

Teste

- Procurar erro despercebido.
- Garantia de funcionalidade.

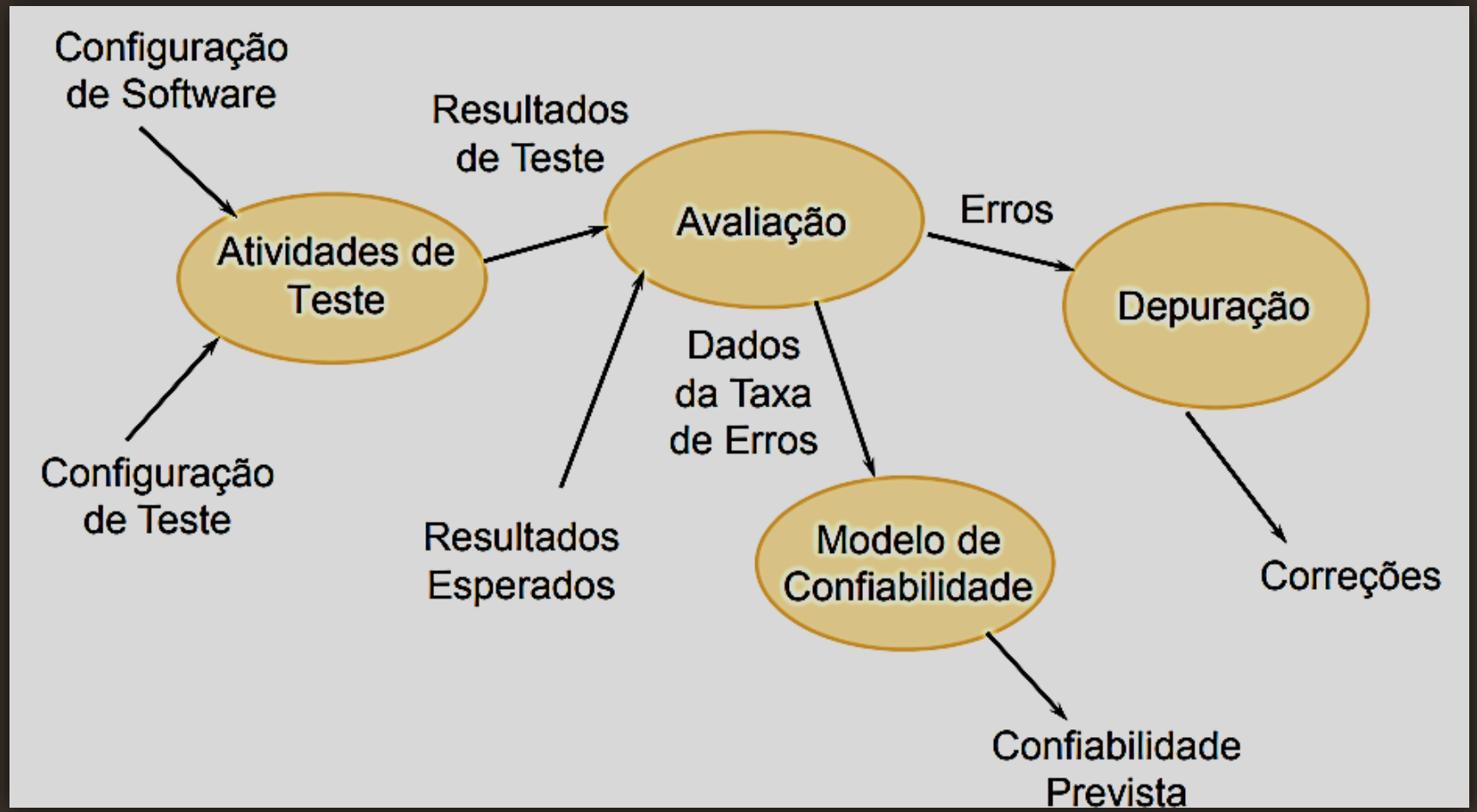
Depuração

- Não prevista no teste.

Normas

- Nível 3 do Modelo CMM/SEI
- Relevante para avaliação da característica funcionalidade (ISO 9126,14598-5)

Introdução



[9]

Universidade Federal Fluminense

Introdução

DDT		
Significado: • Data Driven Testing	Tradução: • Teste Dirigido por Dados	Funcionalidade • Testar o programa

Onde está presente?

Onde está presente?

Comunicação → Planejamento → Modelagem → Construção → Implantação

Onde está presente?

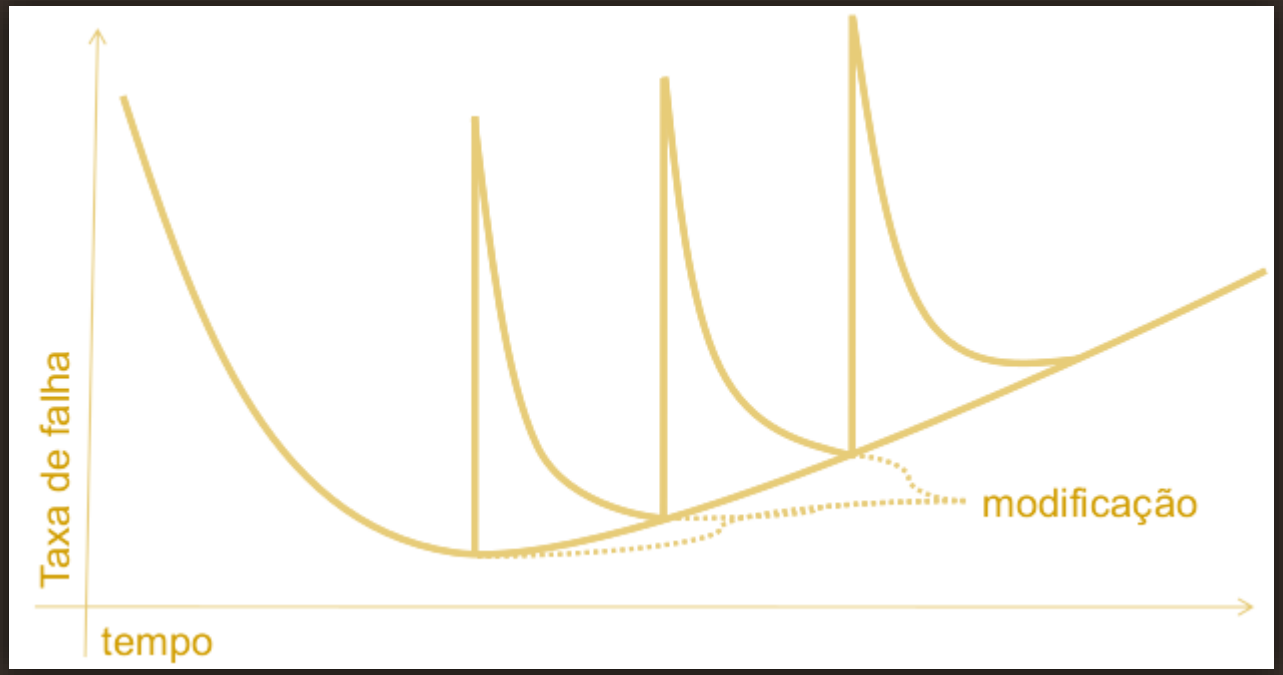
É além do ciclo de vida...

Onde está presente?

É além do ciclo de vida...

Release & Manutenção.

Com a finalidade de minimizar a curva de falha de software



[7]

Justificativa

Universidade Federal Fluminense

Justificativa

Alocação de equipe para teste;

Universidade Federal Fluminense

Justificativa

Alocação de equipe para teste;

Paralisação do desenvolvimento;

Universidade Federal Fluminense

Justificativa

Alocação de equipe para teste;

Paralisação do desenvolvimento;

Realizar um lista de teste manual;

Universidade Federal Fluminense

Justificativa

Alocação de equipe para teste;

Paralisação do desenvolvimento;

Realizar um lista de teste manual;

Gerenciamento;

Universidade Federal Fluminense

Justificativa

- Alocação de equipe para teste;
- Paralisação do desenvolvimento;
- Realizar um lista de teste manual;
- Gerenciamento;
- **Exaustivo & Repetitivo;**

Universidade Federal Fluminense

Justificativa

- Alocação de equipe para teste;
- Paralisação do desenvolvimento;
- Realizar um lista de teste manual;
- Gerenciamento;
- Exaustivo & Repetitivo;
- **Custo elevado;**

Os Estados Unidos estimam que bugs de software lhes custam aproximadamente **60 bilhões de dólares por ano.** [1]

Universidade Federal Fluminense

Justificativa

- Alocação de equipe para teste;
- Paralisação do desenvolvimento;
- Realizar um lista de teste manual;
- Gerenciamento;
- Exaustivo & Repetitivo;
- Custo elevado;
- **Risco de vida**

Foguete Ariane 5 explodiu por um erro de software;

Hospital panamenho matou pacientes pois seu software para dosagem de remédios errou

Universidade Federal Fluminense

Justificativa

- Alocação de equipe para teste;
- Paralisação do desenvolvimento;
- Realizar um lista de teste manual;
- Gerenciamento;
- Exaustivo & Repetitivo;
- Custo elevado;
- Risco de vida
- **Não é viável para nem um ciclo de vida.**

Objetivo

Universidade Federal Fluminense

Objetivo

Tornar o processo de teste automatizado;

Universidade Federal Fluminense

Objetivo

Tornar o processo de teste automatizado;

Desenvolvimento visando o teste;

Universidade Federal Fluminense

Objetivo

Tornar o processo de teste automatizado;

Desenvolvimento visando o teste;

Reduzir custo & tempo;

Universidade Federal Fluminense

Objetivo

Tornar o processo de teste automatizado;

Desenvolvimento visando o teste;

Reduzir custo & tempo;

Coletar estatística;

Universidade Federal Fluminense

Objetivo

Tornar o processo de teste automatizado;

Desenvolvimento visando o teste;

Reduzir custo & tempo;

Coletar estatística;

Manutenção;

Universidade Federal Fluminense

Objetivo

- Tornar o processo de teste automatizado;
- Desenvolvimento visando o teste;
- Reduzir custo & tempo;
- Coletar estatística;
- Manutenção;
- **Aumento da qualidade;**

Universidade Federal Fluminense

Objetivo

- Tornar o processo de teste automatizado;
- Desenvolvimento visando o teste;
- Reduzir custo & tempo;
- Coletar estatística;
- Manutenção;
- Aumento da qualidade;
- **Evitar o erro humano;**

Universidade Federal Fluminense

Objetivo

- Tornar o processo de teste automatizado;
- Desenvolvimento visando o teste;
- Reduzir custo & tempo;
- Coletar estatística;
- Manutenção;
- Aumento da qualidade;
- Evitar o erro humano;
- **Evitar estresse da equipe & pessoal.**

Universidade Federal Fluminense

Tipos de Teste

Universidade Federal Fluminense

Tipos de testes

Unidade	• Classes individuais • Responsabilidade do programador
Funcionais (integração)	• Unidade funcional coerente - intermediário • Responsabilidade do time
Sistema	• Total System Environment • Equipe independente de testes
Regressão	• Simulação – antes do lançamento • Subconjunto dos testes de sistema
Alfa & Beta	• Produto de instrução • Como realmente é usando
Mutante	• Verificação, validação e teste • Estuda o comportamento dos testes

Universidade Federal Fluminense

Tipos de testes – Foco do Trabalho

Unidade

- Importante: **Não se preocupa com funcionalidade do sistema. Apenas com uma funcionalidade especifica**
- Ferramenta: **Tunit, TunitPerf**

Funcional (integração)

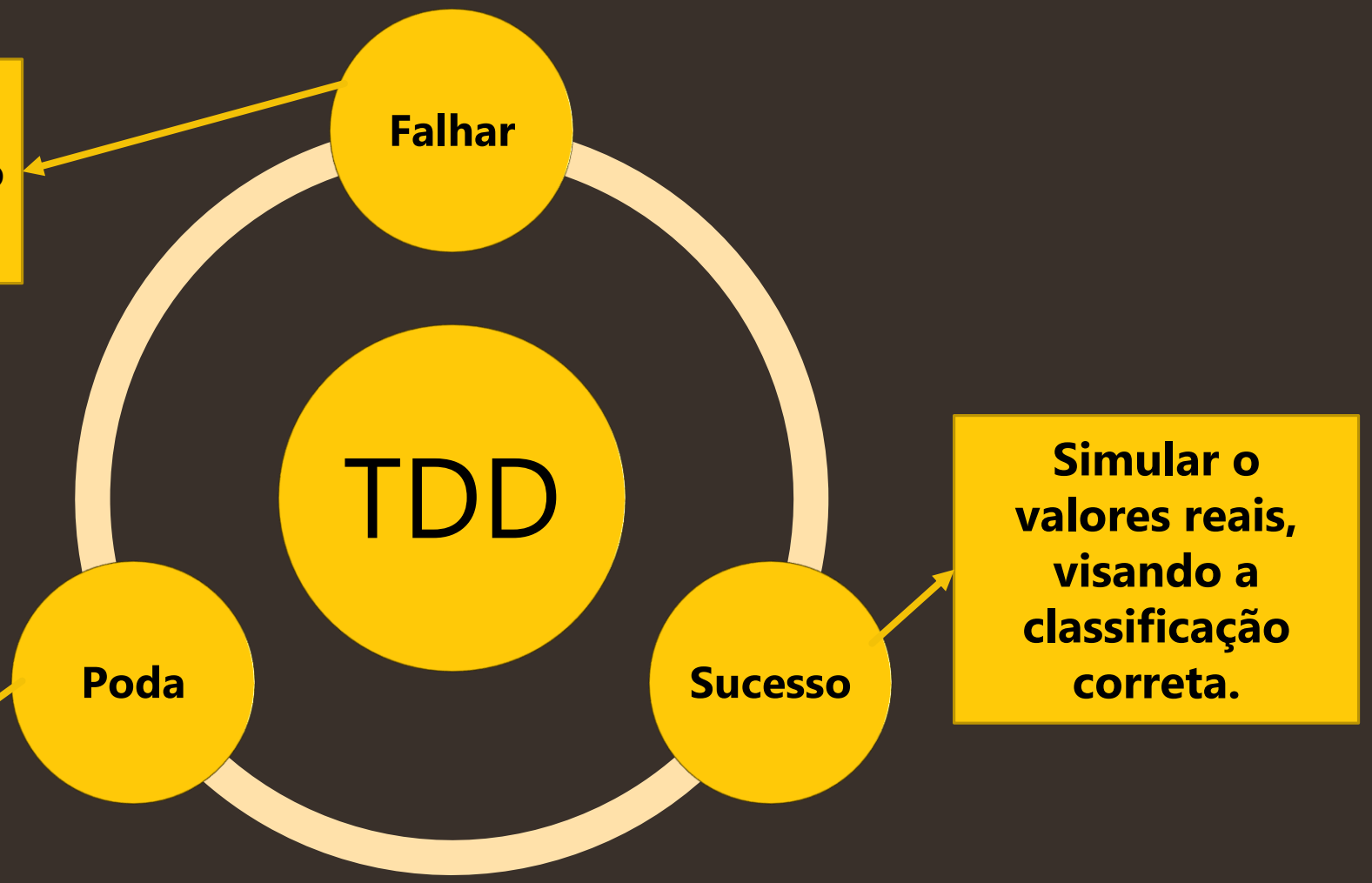
- Teste Exaustivo: Reproduz uma funcionalidade inúmeras vezes.
- Ferramenta: **Selenium**

Universidade Federal Fluminense

Teste de Unidade

Universidade Federal Fluminense

Teste de Unidade



Verificar se o teste funciona. Quebrando o código

Remover testes triviais.

Simular o valores reais, visando a classificação correta.

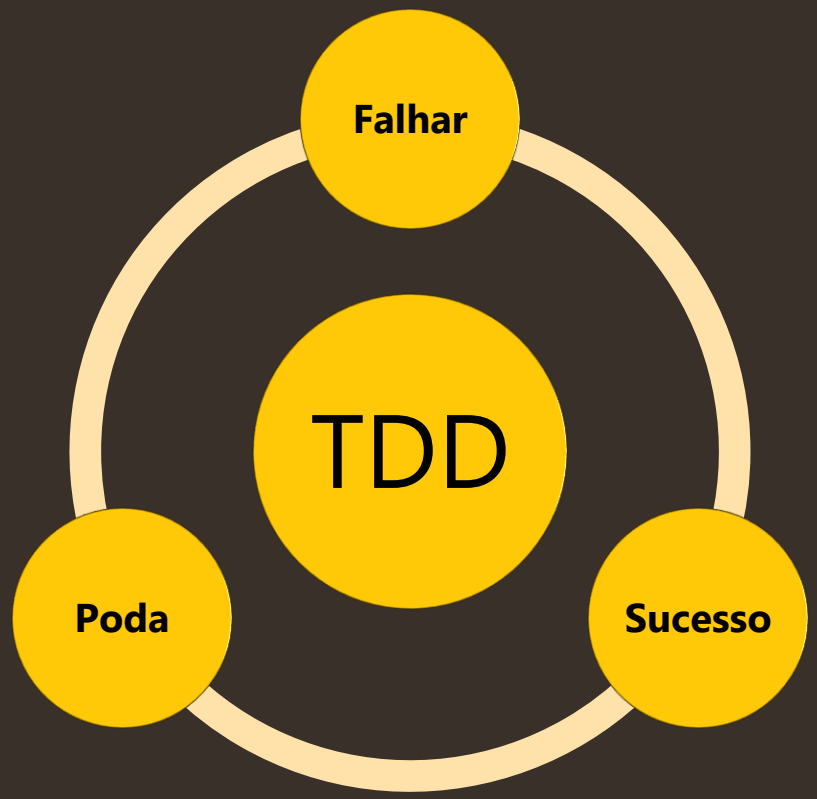
Teste de Unidade

Pode ser usar também da seguinte forma:

1 – Escrever os Testes
2 – Executar os testes e verificar as falhas

3 – Escrever o código
4 – Rodar os testes para identificar sucesso

5 – Refatorar o código para corrigir defeitos e efetuar melhorias



Universidade Federal Fluminense

Teste de unidade - Escrever os Testes

O que o programa faz.

Analisar o que deve ser testado.

Objetivo do teste.

Universidade Federal Fluminense

Teste de unidade - Escrever os Testes

Em nosso caso a classe deverá responder os tipos de triângulos, de acordo com seus lados.

Portanto a classe de teste deve exaustivamente todas as possibilidades e reportar os erro existentes.

Teste de unidade - Escrever os Testes

Logo devemos prever algumas respostas, como:

- Se for isósceles
- Se for equilátero
- Se for Escaleno
- Se a soma de dois lados for igual a do terceiro
- Se a soma de dois lados for menor a do terceiro

Podendo ser listado também

- Se algum lado for negativo ;
- Se todos os comprimentos do triângulo forem zero;
- Se algum comprimento do triângulo for zero;

Teste de unidade - Escrever os Testes

Em nosso teste, vamos verificar 14 itens do triângulo , sendo eles:

- Escaleno;
- Equilátero;
- Isósceles;
- Isósceles;
- Isósceles;
- Com Lado Nulo;
- Com Lado Negativo;

Cuja a soma dos lados:

- A e B é igual a C;
- A e C igual a B;
- C e B igual ao lado A;
- Do 1º e do 2º seja menor que a 2º;
- Sejam Nulos;
- A e B é menor que C;
- B e C é menor que B

Universidade Federal Fluminense

Teste de unidade - Escrever os Testes

Sintaxe de entrada (int, int, int, String):

{ valorLadoA, valorLadoB, valorLadoC, resultadoTeste }.

É fundamental escrever os casos de teste antes mesmo que a classe.

Teste de unidade - Escrever os Testes

Preparação dos itens

{2, 9, 10,"Escaleno"}
{20, 20, 20, "Equilátero"}
{20, 20, 30, "Isósceles"}
{20, 30, 20, "Isósceles"}
{30, 20, 20, "Isósceles"}
{0, 2, 9, "Lado Nulo"}
{3, -2, 9, "Lado Negativo"}
{5, 6, 11, "Soma dos dois lados igual a terceiro"}
{5, 11, 6, "Soma dos dois lados igual a terceiro"}
{11, 6, 5, "Soma dos dois lados igual a terceiro"}
{5, 6, 12, "Soma dos dois lados menor que a terceiro"}
{0, 0, 0, "Todos os lados Nulos"}
{5, 12, 6, "Soma dos dois lados menor que a terceiro"}
{12, 5, 6, "Soma dos dois lados menor que a terceiro"}

Universidade Federal Fluminense

Teste de Unidade

As etapas seguintes serão implementadas no Eclipse

2 – Executar os testes e verificar as falhas

3 – Escrever o código

4 – Rodar os testes para identificar sucesso

5 – Refatorar o código para corrigir defeitos e efetuar melhorias

Universidade Federal Fluminense

JUnit

Universidade Federal Fluminense

JUnit

- O que é ?
  - É um framework para linguagem Java para TDD.
- Será usado junto ao Eclipse IDLE.

Universidade Federal Fluminense

Junit

2 – Executar os testes e verificar as falhas

Package Explorer | JUnit

Finished after 0,118 seconds

Runs: 14/14 Errors: 14 Failures: 0

- test[0] (0,027 s)
- [1] (0,004 s)
- [2] (0,001 s)
- [3] (0,001 s)
- [4] (0,000 s)
- [5] (0,001 s)
- [6] (0,001 s)
- [7] (0,000 s)
- [8] (0,001 s)
- [9] (0,000 s)
- [10] (0,001 s)
- [11] (0,001 s)

Failure Trace

```
java.lang.Error: Unresolved compilation pro
Triangulo cannot be resolved to a type
Triangulo cannot be resolved to a type

at Triangulo.TDD.test(TDD.java:156)
```

TDD.java

```
133                 {11, 6, 5, "Soma dos dois lados igual a terceiro"},
134
135                 //Test10
136                 {5, 6, 12, "Soma dos dois lados menor que a terceiro"},
137
138                 //Test11
139                 {0, 0, 0, "Todos os lados Nulos"},
140
141                 //Test12
142                 {5, 12, 6, "Soma dos dois lados menor que a terceiro"},
143
144                 //Test13
145                 {12, 5, 6, "Soma dos dois lados menor que a terceiro"},
146             }
147         );
148     }
149
150     // Método que executa o teste a cada instanciação do objeto da classe teste
151
152     @Test
153     public void test() {
154
155         // Vamos criar um objeto do tipo Trianglulo (nossa futura classe que ainda
156         Triangulo escalenoValido = new Triangulo(a, b, c);
157
158         // Realizamos a comparação entre o valor que foi retornado e o valor que é
159         assertEquals(escalenoValido.retornarTipo(), tipo);
160     }
161
162 }
```

Outline

- getB() : int
- setB(int) : void
- getC() : int
- setC(int) : void
- getTipo() : String
- setTipo(String) : void
- TDD(int, int, int, String)
- carregaTriangulosDeTeste()
- test() : void

Javadoc | Declaration

Triangulo

É importante que todos os testes falhem. Pois a classe ainda não implementada.

Universidade Federal Fluminense

Junit

3 – Escrever o código

Package Explorer | JUnit

Finished after 0,06 seconds

Runs: 14/14 Errors: 14 Failures: 0

- test[0] (0,003 s)
- [1] (0,001 s)
- [2] (0,001 s)
- [3] (0,001 s)
- [4] (0,001 s)
- [5] (0,001 s)
- [6] (0,002 s)
- [7] (0,001 s)
- [8] (0,000 s)
- [9] (0,001 s)
- [10] (0,000 s)
- [11] (0,001 s)

Failure Trace

```
java.lang.Error: Unresolved compilation pro
Syntax error, insert ") Statement" to comple
amp cannot be resolved to a variable
amp cannot be resolved to a type
Syntax error on token ";", delete this token
amp cannot be resolved to a type
Syntax error on token ";", delete this token
Duplicate local variable amp
amp cannot be resolved to a type
Syntax error on token ";", delete this token
Duplicate local variable amp
```

TDD.java | Triangulo.java

```
42  public String retornarTipo() {
43
44      // erro de lógica
45      if((getA() == 0) || (getB() == 0) || (getC() == 0))
46      return "Todos os lados Nulos";
47
48      if((getA() == 0) || (getB() == 0) || (getC() == 0))
49      // Erro no retorno
50      return "Lado Núlo";
51
52      if((getA() < 0) || (getB() < 0) || (this.c < 0))
53      return "Lado Negatívo";
54
55      if((getA() == getB()) &amp;amp;amp;amp;amp;amp;amp;amp;&amp;amp;amp;amp;amp;
56      return "Equilátero";
57
58      if(
59      ((getA() != getB()) &amp;amp;amp;amp;amp;amp;amp;amp;&amp;amp;amp;amp;amp;am
60      ((getA() == getB()) &amp;amp;amp;amp;amp;amp;amp;amp;&amp;amp;amp;amp;amp;am
61      ((getA() == getC()) &amp;amp;amp;amp;amp;amp;amp;amp;&amp;amp;amp;amp;amp;am
62      )
63      // Erro no retorno
64      return "Isóceles";
65
66      if( (getA() + getB() == this.c)||
67      (getB() + getC() == getA())||
68      (getC() + getA() == getB())
69      )
70      return "Soma dos dois lados igual a terceiro";
71
72      if( (getA() + getB()< getC())||
```

Task List

Find | All | Activat...

Outline

- c : int
- getA() : int
- setA(int) : void
- getB() : int
- setB(int) : void
- getC() : int
- setC(int) : void
- Triangulo(int, int, int)
- retornarTipo() : String

Javadoc | Declaration

int Triangulo.Triangulo.getC()

```
public int getC() {
    return c;
}
```

Writable | Smart Insert | 53 : 25

Os erros foram inseridos de propósitos .

Universidade Federal Fluminense

Junit

4 – Rodar os testes para identificar sucesso

JUnit

Finished after 0,087 seconds

Runs: 14/14 Errors: 0 Failures: 4

- Triangulo.TDD [Runner: JUnit 4] (0,045 s)
  - [2] (0,035 s)
    - test[2] (0,035 s)
  - [3] (0,001 s)
  - [4] (0,001 s)
  - [5] (0,002 s)

Failure Trace

org.junit.ComparisonFailure: expected:<Isó[

at Triangulo.TDD.test(TDD.java:159)

Triangulo.java | TDD.java

```
134                 //Test10
135                 {5, 6, 12, "Soma dos dois lados menor que a terceiro"},
136
137                 //Test11
138                 {0, 0, 0, "Todos os lados Nulos"},
139
140                 //Test12
141                 {5, 12, 6, "Soma dos dois lados menor que a terceiro"},
142
143                 //Test13
144                 {12, 5, 6, "Soma dos dois lados menor que a terceiro"},
145             }
146         );
147     }
148
149     // Método que executa o teste a cada instanciação do objeto da classe teste
150
151     @Test
152     public void test() {
153
154         // Vamos criar um objeto do tipo Trianglulo (nossa futura classe que ainda
155         Triangulo escalenoValido = new Triangulo(a, b, c);
156
157         // Realizamos a comparação entre o valor que foi retornado e o valor que é
158         assertEquals(escalenoValido.retornarTipo(), tipo);
159     }
160
161 }
162
```

Outline

- getB() : int
- setB(int) : void
- getC() : int
- setC(int) : void
- getTipo() : String
- setTipo(String) : void
- TDD(int, int, int, String)
- carregaTriangulosDeTeste()
- test() : void

Javadoc | Declaration | Console

<terminated> TDD [JUnit] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (9 de nov de 2017 09:36:59)

Writable | Smart Insert | 147 : 21

Junit

4 – Rodar os testes para identificar sucesso

Quantidade de Testes

JUnit

Finished after 0,045 seconds

Runs: 14/14 Errors: 0 Failures: 4

Triangulo.TDD [Runner: JUnit 4] (0,004 s)
- [2] (0,000 s)
  - test[2] (0,000 s)
- [3] (0,001 s)
  - test[3] (0,001 s)
- [4] (0,000 s)
  - test[4] (0,000 s)
- [5] (0,000 s)
  - test[5] (0,000 s)

Failure Trace

org.junit.ComparisonFailure: expected:<[Todos os lados Nulos]> but was:<[Lado Nulo]>

at Triangulo.TDD.test(TDD.java:161)

Qual a diferença entre o resultado esperado e o recebido

Junit

5 – Refatorar o código para corrigir defeitos e efetuar melhorias

JUnit

Finished after 0,036 seconds

Runs: 13/13 Errors: 0 Failures: 0

Failure Trace

Triangulo.java | TDD.java

```
27          return c;
28      }
29
30      public void setC(int c) {
31          this.c = c;
32      }
33
34      //construtor
35      public Triangulo(int a, int b, int c) {
36
37          setA(a);
38          setB(b);
39          setC(c);
40      }
41
42      public String retornarTipo() {
43
44
45          if((getA() == 0) || (getB() == 0) || (getC() == 0))
46          return "Todos os lados são Nulos";
47
48          //if((getA() == 0) || (getB() == 0) || (getC() == 0))
49          //return "Lado Nulo";
50
51          if((getA() < 0) || (getB() < 0) || (getC() < 0))
52          return "Lado Negativo";
53
54          if((getA() == getB()) && (getB() == getC() ))
55          return "Equilátero";
56
57          if(
58          ((getA() != getB()) && (getB() == getC() ))||
59          ((getA() == getB()) && (getB() != getC() ))||
60          ((getA() == getC()) && (getB() != getC() ))
61          )
62          return "Isósceles";
63
64          if( (getA() + getB() == getC() )||
65          (getB() + getC() == getA())||
66          (getC() + getA() == getB())
67          )
68          return "Soma dos dois lados igual a terceiro";
```

Teste redundante

Writable | Smart Insert | 35 : 44

Universidade Federal Fluminense

Teste de Performance

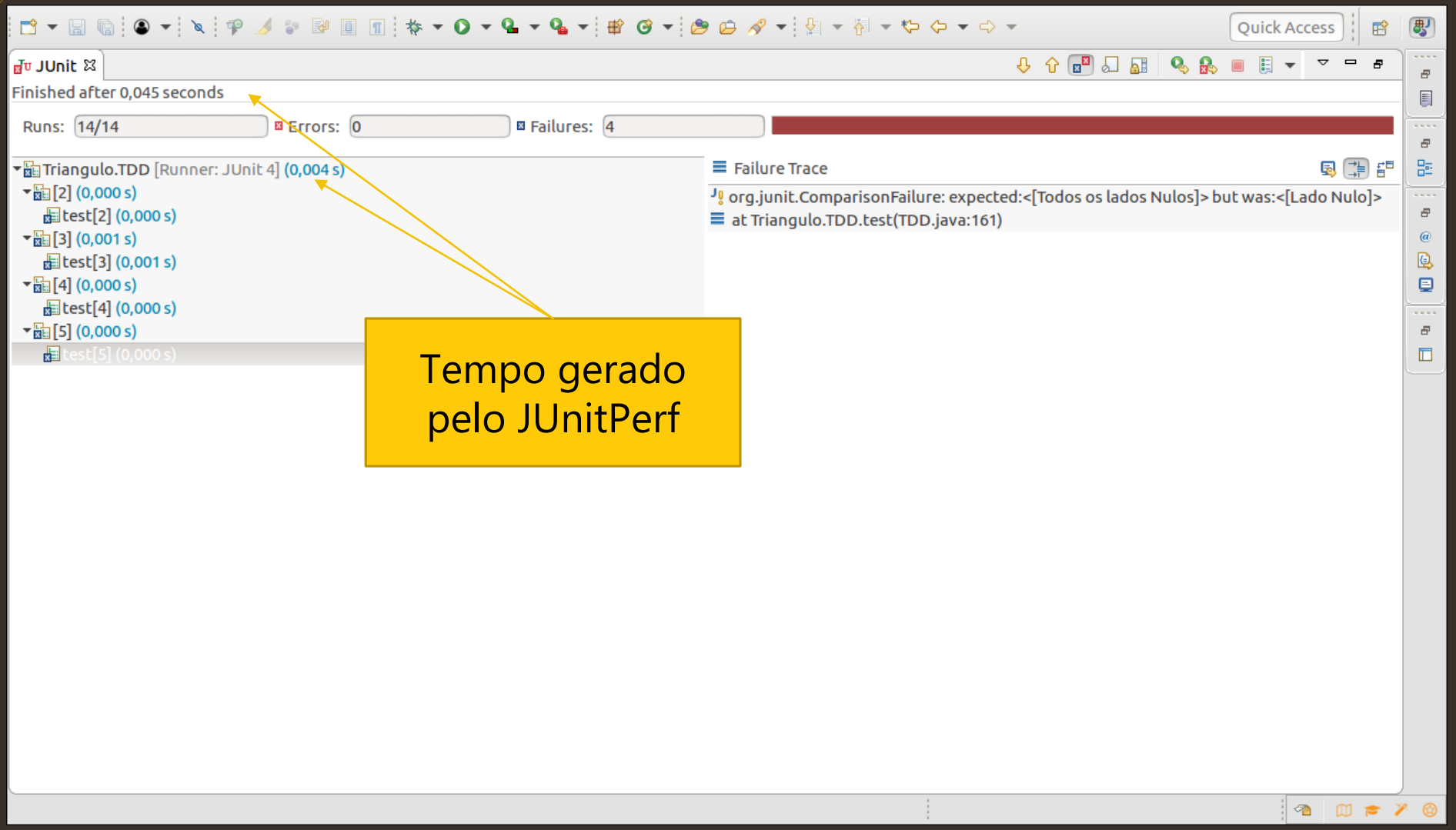
Universidade Federal Fluminense

JUnitPerf

- Um adicional no Junit
- Marcar a performance
- Tempo de cada método de forma individual

Universidade Federal Fluminense

JUnitPerf



Tempo gerado pelo JUnitPerf

Universidade Federal Fluminense

Teste Funcional

Universidade Federal Fluminense

Selenium

Universidade Federal Fluminense

Selenium

Objetivo

- rodar testes num sistema finalizado

Funcionalidade

- WEB
- Multi-plataformas

Universidade Federal Fluminense

Selenium

Selenium RC

- Servidor escrito em Java
- Recebe chamadas http e executa os testes
- TDD realiza as chamadas

Selenium IDE

- Extensão do Firefox
- Grava e recorda as ações do usuário
- As ações podem ser transformadas em código em várias linguagens entre elas java.

Universidade Federal Fluminense

Selenium

Integração

Uma aplicação web

Selenium Server

Selenium Test client com JUnit

Conclusão

Universidade Federal Fluminense

Conclusão

- Teste são essencial para garantir o funcionamento de uma aplicação;

Universidade Federal Fluminense

Conclusão

- Teste são essencial para garantir o funcionamento de uma aplicação;
- **Realizar esses testes manuais é desgastantes;**

Universidade Federal Fluminense

Conclusão

- Teste são essencial para garantir o funcionamento de uma aplicação;
- Realizar esses testes manuais é desgastantes;
- **Automatizar o processo, resulta em vários benefícios;**

Universidade Federal Fluminense

Conclusão

- Teste são essencial para garantir o funcionamento de uma aplicação;
- Realizar esses testes manuais é desgastantes;
- Automatizar o processo, resulta em vários benefícios;
- **Existe diversas ferramentas para auxiliar o desenvolvimento de teste automatizado;**

Universidade Federal Fluminense

Conclusão

- Teste são essencial para garantir o funcionamento de uma aplicação;
- Realizar esses testes manuais é desgastantes;
- Automatizar o processo, resulta em vários benefícios;
- Existe diversas ferramentas para auxiliar o desenvolvimento de teste automatizado;
- **A possível testes inúmeras funcionalidades e simulações;**

Universidade Federal Fluminense

Conclusão

- Teste são essencial para garantir o funcionamento de uma aplicação;
- Realizar esses testes manuais é desgastantes;
- Automatizar o processo, resulta em vários benefícios;
- Existe diversas ferramentas para auxiliar o desenvolvimento de teste automatizado;
- A possível testes inúmeras funcionalidades e simulações;
- **Os testes são limitados a imaginação do desenvolvedor.**

Referências

Universidade Federal Fluminense

Referências

[1] ANICHE, Maurício. Test-Driven Development - Teste e Design no Mundo Real. Casa do Codigo. Casa do Código, 2012. ISBN: 978-85-66250-04-6

[2] ANICHE, Maurício. Unidade, integração ou sistema? Qual teste fazer?. 2014, http://blog.caelum.com.br/unidade-integracao-ou-sistema-qual-teste-fazer/.

[3] FABBRI et al . "APLICAÇÃO DO CRITÉRIO - ANÁLISE DE MUTANTES NA VALIDAÇÃO DE ESPECIFICAÇÕES BASEADAS EM STATECHARTS". XI-Simpósio Brasileiro de Engenharia de Software. 1997, p. 429-444.

[4] DA ROCHA, helder. Testes de performance com JUnitPerf. 2002, https://pt.slideshare.net/denistuning/j820-10-performance.

Universidade Federal Fluminense

Referências

[6] MARC PHILIPP, Sam Brannen. The new major version of the programmer-friendly testing framework for Java 8. 5 ed. 2017, http://junit.org/junit5/.

[7] MURTA, Leonardo. Processo de Software. instituto de computação - Universidade Federal Fluminense, 2010, http://www2.ic.uff.br/~leomurta/courses/es2/aula2.pdf

[8] NOGUEIRA, Elias. Data Driven com Selenium IDE? Sim, é possível!. 2011, http://sembugs.blogspot.com.br/2011/01/selenium-ide-data-driven-massa-de-dados.html.

[9] RAMOS, Ricardo. Introdução ao Teste de Software. 2010, http://www.univasf.edu.br/ ricardo.aramos/disciplinas/ES_I_2010_2/TesteSoftware.pdf.

Universidade Federal Fluminense

Referências

[10] RIBEIRO, Camilo. Teste de unidade para quem não sabe nada de programação. 2011, http://www.bugbang.com.br/ teste-de-unidade-para-quem-nao-sabe-nada-de-programacao/.

[11] RIBEIRO, Camilo. Uma Introdução a TDD com JUnit. 2010, http://www.bugbang.com.br/uma-introducao-a-tdd-com-junit/.

[12] SAUVÉ, Jacques. Testes de Unidade. 2001, http://www.dsc.ufcg.edu.br/ jacques/cursos/apoo/html/impl/impl3.htm.

[13] SOARES, Diogo. Testes de unidade com JUnit. 2007, https://www.devmedia.com.br/testes-de-unidade-com-junit/4637.

[14] STEPPAT, Nico. Testes de aceitação com o Selenium. 2007, http://blog.caelum.com.br/testes-de-aceitacao-com-o-selenium/.

Universidade Federal Fluminense

Obrigado

Teste de Unidade e Funcional

Wellington Silva