

Teste de Software

Material desenvolvido para
Mini Aula Edital 2019 / 2

Marcelo Josué Telles

IENH - Instituição Evangélica de Novo Hamburgo - Unidade Fundação Evangélica
Rua Frederico Mentz, 526 - CEP: 93525-360
Bairro Hamburgo Velho- Novo Hamburgo / RS

marcelojtelles@gmail.com

30 de julho de 2019



- 1** Introdução
 - Objetivos

- 2** Desenvolvimento orientado a Teste
 - Conceitos básicos
 - Tipos de Teste
 - Exemplos Práticos

- 3** Material Extra
 - Exercícios

- 4** Finalizando
 - Revisão

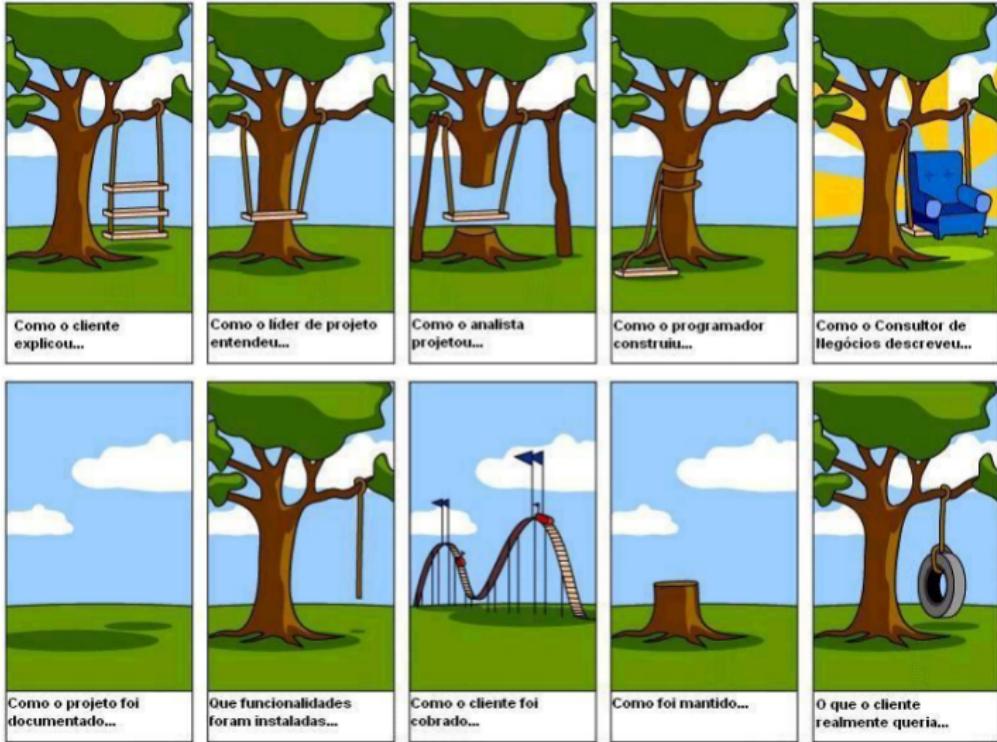


The main content area features three large colored boxes (green, orange, and teal) overlaid on the background image. Each box contains an icon and text: the green box has an icon of a notepad and pen, labeled 'EDUCAÇÃO INFANTIL ENSINO FUNDAMENTAL ENSINO MÉDIO'; the orange box has an icon of a person at a computer, labeled 'CURSOS TÉCNICOS'; and the teal box has an icon of a graduation cap, labeled 'ENSINO SUPERIOR'.


Edição Básica
29/07/2019
IENH participa do XV Congresso do Ensino Privado Gaúcho


Cursos Técnicos
09/07/2019
Alunos do Curso Técnico em Informática da IENH organizam evento com Project Model Canvas


Faculdade
29/07/2019
Professor da Faculdade IENH participa de evento internacional José Menna esteve na XIX ICPIC



Aprenda a escutar o cliente.... Entenda o mundo do cliente...



Como o cliente
explicou...



O que o cliente
realmente queria...

Objetivos desta aula – Teste de Software –

Ao final desta aula:

- Identificar a importância do desenvolvimento orientado a testes
- Ter uma visão conceitual sobre diferentes tipos de testes
- Compreender conceitos básicos sobre Engenharia de Software

Situações onde podemos aplicar os conhecimentos desta aula

- Especificar quais testes são necessários
- Caracterizar elementos e classes de uma unidade de testes
- Testar software de forma automatizada



A importância dos testes

Por que perder tempo testando?

- Confiança
- Funcionalidade
- Performance

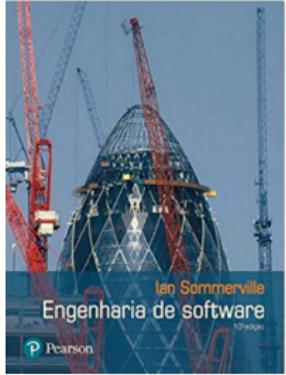
Aderson B. de Souza



Paulo Silveira



Ian Sommerville



História

- 1848 – Thomas Edison encontra um inseto em uma maquinaria da indústria e nomeia o problema como BUG.
- 1947 – Primeiro Bug é encontrado no computador Harvard Mark e este foi registrado como o primeiro erro encontrado em um computador.
- 1979 – Publicado o Livro “The Art of Software Testing” de Glendford Myers. Neste livro é mencionada a arte de encontrar erros em sistemas para ter um controle de qualidade de software.
- 1980 – Surgem os modelos prescritivos de desenvolvimento de sistemas e com eles a criação de ferramentas para efetuar os testes de software.

Primeiro Bug

9/9

0800 Autan started
 1000 stopped - autan ✓
 13' uc (03) MP-MC 1.2700 9.037 847 025
 (03) PRO-2 2.130476415 9.037 846 795 contact
 contact 2.130676415
 Relays 6-2 in 033 failed special speed test
 in relay " 10.000 test.

Relay
 2142
 Relay 3372

1100 Started Cosine Tape (Sine check)
 1525 Started Multi. Adder Test.

1545



Relay #70 Panel F
 (moth) in relay.

1600 First actual case of bug being found.
 1700 Autan started.
 closed down.

Desenvolvimento Orientado a Testes

TDD do inglês *Test Driven Development*

- Segundo Roger, você não tem nada a perder, a não ser os seus bugs.
- Primeiro escrever os testes, depois implementar o sistema.
- Os componentes individuais são testados para garantir que operem corretamente em conjunto

Roger S. Pressman,
Bruce R. Maxim,
Reginaldo Arakaki,
Julio Arakaki e Renato
Manzan de Andrade



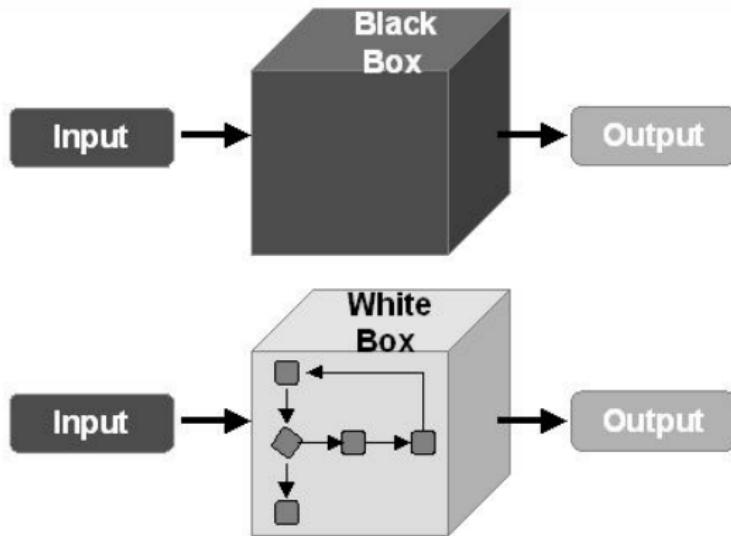
Alexandre Bartie



Tipos de Teste

- Funcional ou caixa preta
- Estrutural ou caixa branca
- Baseado em erros

Comparison among Black-Box & White-Box Tests



Funcional ou caixa preta

Ele garante que os requisitos funcionem conforme o especificado. Não se preocupa COMO foi implementado.

Classificação dos Testes Funcionais

- **Requisitos:** Testa se o sistema faz o que deve fazer
- **Régressão:** Testa se o sistema foi afetado por alguma atualização
- **Tratamento de erros:** Testa o tratamento dos erros, ou seja, alertar o usuário do que deve ser feito
- **Interfaces de integração** Testa se o sistema troca informações
- **Controle:** Testa se o sistema tem algum controle de dados (validações)
- **Paralelismo:** Testa se a versão nova e antiga geram os mesmos resultados

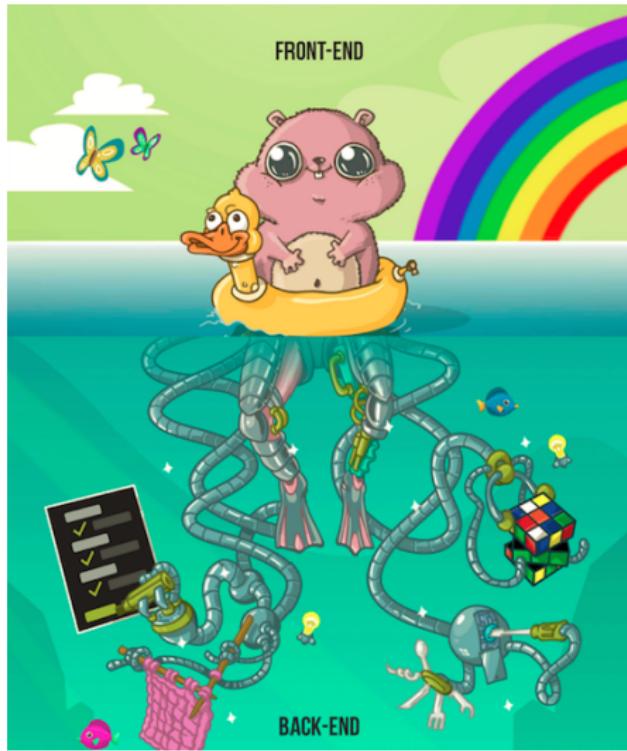
Estrutural ou caixa branca

Esse teste tem por objetivo testar o código fonte, testar cada linha de código e os fluxos básicos e os alternativos.

Classificação dos Testes Estruturais

- **Stress:** Verifica como o sistema é executado com dados volumosos
- **Execução:** Testa se o sistema atinge o nível desejado de eficiência
- **Recuperação Contingência:** Testa se o sistema opera conforme sua documentação
- **Operação** Testa se o sistema troca informações com outros
- **Compliance:** Testa se o sistema foi desenvolvido conforme procedimentos
- **Segurança:** Testa se o sistema está protegido conforme políticas da organização

Front End, Back End



Baseado em erros

Classificação do Teste Baseado em erros

- **Chamadas das operações:** resultado inesperado, mensagem errada e invocação incorreta.
- **Integração:** Testa as chamadas, não o código chamado (caixas)

```
@Test
public void deveriaSomarDoisValoresPassados() throws Exception {
    int valorA = 1;
    int valorB = 2;
    Calculadora calculadora = new Calculadora();
    int soma = calculadora.soma(valorA, valorB);

    assertEquals(3, soma);
}
```

```
/*
 *
 * @author marcelojtelles
 */
public class Calculadora {

    public int soma(int valorA, int valorB) {
        return valorA+valorB;
    }
}
```

```
@Test  
public void deveriaSubtrairDoisValores() throws Exception {  
    Calculadora calculadora = new Calculadora();  
    int valorA = 6;  
    int valorB = 2;  
    int soma = calculadora.subtrai(valorA, valorB);  
  
    assertEquals(4, soma);  
}
```

```
public int subtrai(int valorA, int valorB) {  
    return valorA - valorB;  
}
```

```
@Test(expected = ArithmeticException.class)
public void deveriaExcecaoFalhaAoDividirPorZero() throws Exception {
    int valorA = 6;
    int valorB = 0;
    Calculadora calculadora = new Calculadora();
    int divisao = calculadora.divide(valorA, valorB);

    assertEquals(0, divisao);
}
```

```
public int divide(int valorA, int valorB) {
    return valorA / valorB;
}
```

Exercício em duplas e individual

Em Duplas:

- 1) Completar a classe Calculadora e desenvolver o método multiplicação, raiz e potência.
- 2) Desenvolver um caso de teste para cada operação (multiplicação, raiz e potência).
- 3) Adicionar na classe Calculadora um método para obter a média de dois valores Float.
- 4) Desenvolver um caso de teste para média, utilize números Float.

Individual:

- 1) Quais os objetivos de teste de Software?
- 2) No desenvolvimento dirigido a testes qual primeiro passo a ser desenvolvido
- 3) Você considera suficiente o teste feito no método da soma? Este método pode operar com valores monetários?

Sites

Importância de testar

<https://imasters.com.br/devsecops/por-que-e-importante-testar-um-software>

Qualidade de software

<https://www.devmedia.com.br/a-importancia-dos-testes-para-a-qualidade-do-software/28439>

Classificação de testes Funcional e estrutural

<https://www.devmedia.com.br/testes-de-software-tecnicas/22283>

Testes baseado em erro

<http://testwarequality.blogspot.com/p/technicas-de-teste.html>

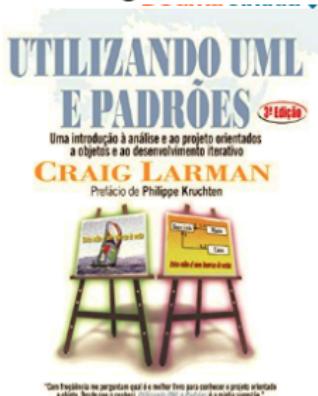
Finalizando a aula

Síntese final da aula

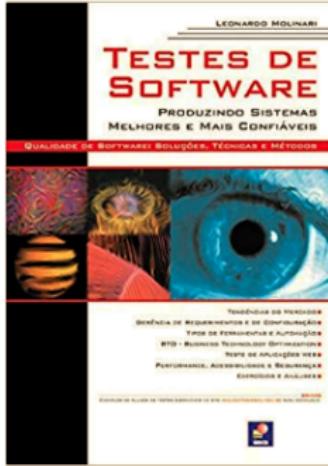
- Nesta aula vimos uma série de conceitos relacionados ao teste de software
- O teste de software está diretamente relacionado com a qualidade do software
- Exploramos um exemplo básico em Java para teste de métodos
- No Desenvolvimento Dirigido a Teste (TDD), vimos que: primeiro realizamos testes, depois desenvolvemos o código principal.
- O Teste de Software garante Confiança, Funcionalidade e Performance

Livros recomendados

Craig Larman



Leonardo Molinari



Sergio Luiz Tonsig



FACULDADE IENH

The End ...

Obrigado!

Cristiano IENH

LATEX

Google

marcelojtelles@gmail.com

