TDD (*Test-Driven Development* - Desenvolvimento Guiado por Testes)

FAPESC – DESENVOLVEDORES PARA TECNOLOGIA DA INFORMAÇÃO

Herculano De Biasi
herculano.debiasi@unoesc.edu.br

Tópicos

- Bibliografia
- Testes de *software*
- Tipos de testes
- Definições
- TDD (*Test Driven Development*)

Segunda Lei de Weinberg

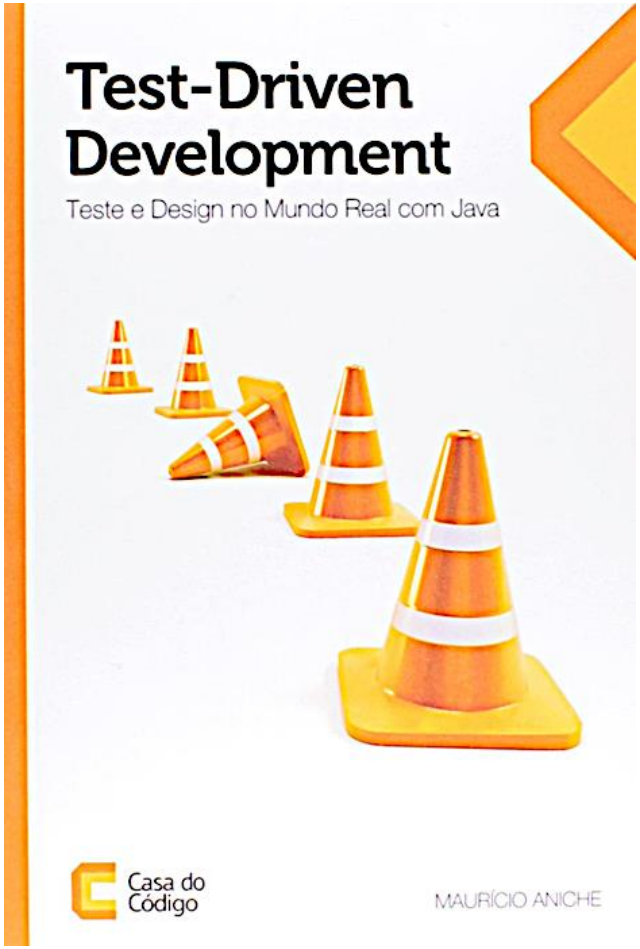
"Se os engenheiros construíssem prédios como os programadores escrevem programas, o primeiro pica-pau destruiria toda a civilização"

Gerald Weinberg

Apoiadores:
UNOESC CITEOSC INOVALE CENTRO DE INOVAÇÃO VIDEIRA DEV-TI SC TEC fapesc SANTA CATARINA

BIBLIOGRAFIA

- ANICHE, Maurício. ***Test-Driven Development***: Teste e *Design* no Mundo Real com Java. São Paulo: Casa do Código, 2012.

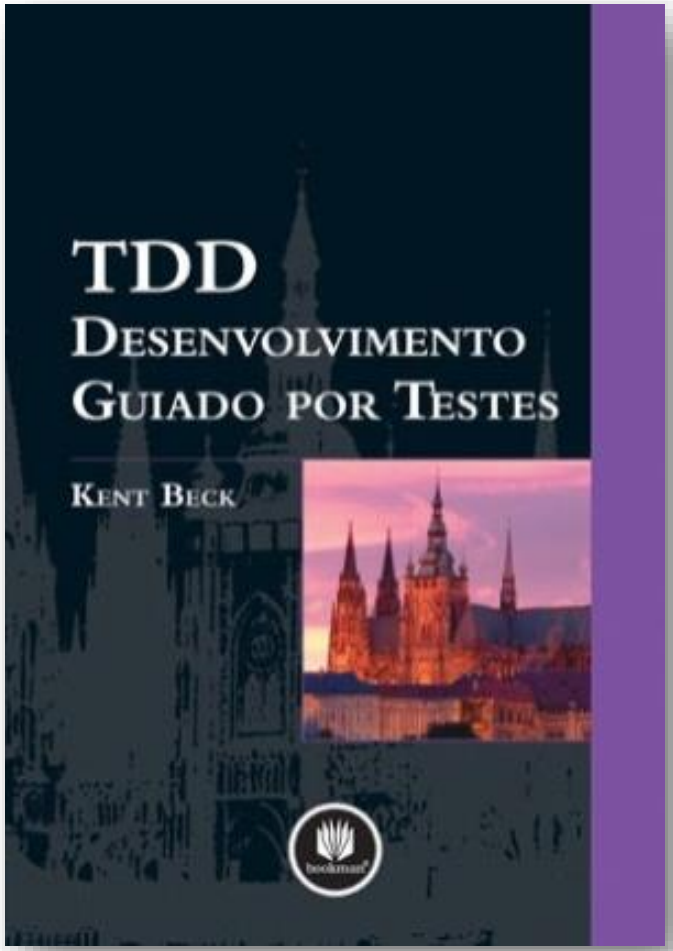


UNOESC

Apoiadores:
UNOESC CITEOSC INOVALE CENTRO DE INOVAÇÃO VIDEIRA
DEV-TI SC TEC fapesc SANTA CATARINA

BIBLIOGRAFIA

- ANICHE, Maurício. ***Test-Driven Development***: Teste e *Design* no Mundo Real com Java. São Paulo: Casa do Código, 2012.

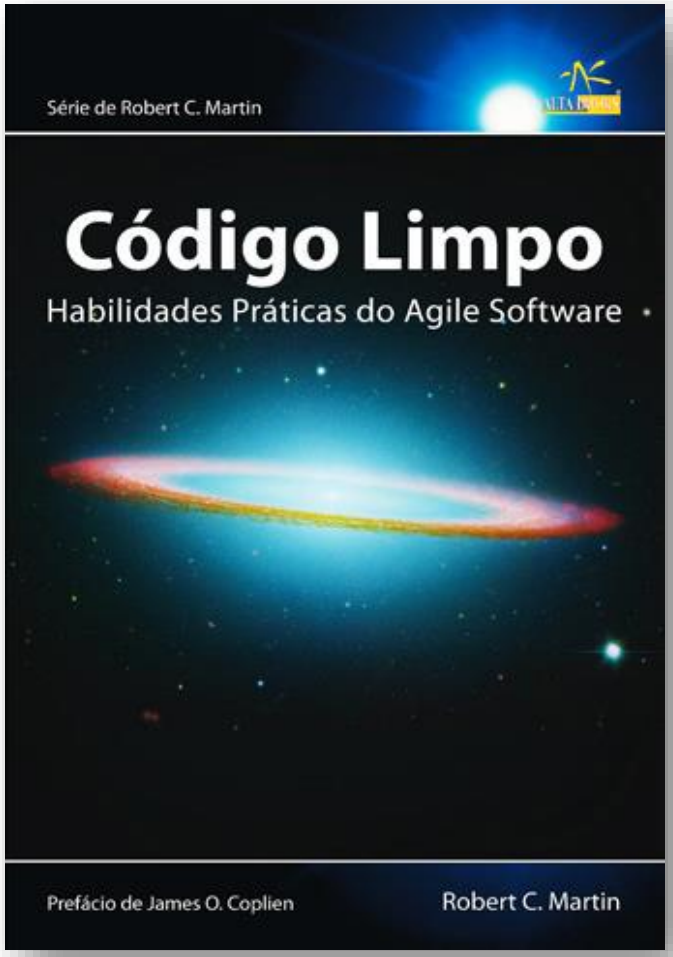


Apoiadores:

UNOESC

BIBLIOGRAFIA

- ANICHE, Maurício. ***Test-Driven Development***: Teste e *Design* no Mundo Real com Java. São Paulo: Casa do Código, 2012.



Apoiadores:

UNOESC

O cliente queria isso

Isso foi como ele explicou para o líder de projeto

O líder de projeto entendeu assim

O analista especificou assim

O programador entendeu assim

E desenvolveu o aplicativo assim

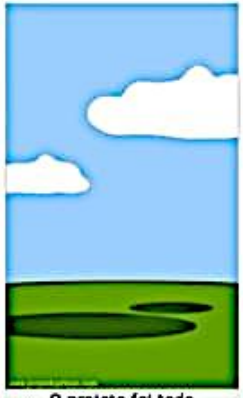
Resultado do teste de carga

Os beta testers receberam isso

O suporte instalou isso no cliente

E cobrou isso

Como os patches devem ser aplicados



O projeto foi todo documentado assim

Os consultores em marketing descreveram assim

E o software foi anunciado assim

Quando ele foi entregue

Solução do suporte para alguns problemas

Resultado do efeito Digg no site do aplicativo

A versão Open Source

TESTES DE *SOFTWARE*

- Parte do processo de desenvolvimento de software
- Procura encontrar falhas/*bugs*
- Verifica a integração adequada dos componentes
- Tenta assegurar que a regra de negócio seja atendida
- Garante o fluxo e a sequência de todo o processo
- Certifica que melhorias não degradem o desempenho da aplicação
- Procura minimizar a possibilidade que os defeitos encontrados sejam corrigidos antes da implantação do *software*
- Analisa se todos os requisitos foram implementados corretamente
- Reduz custos de manutenção corretiva e retrabalho

Tipos de Testes

- Tipos de testes

Tipo de Teste	Descrição
Teste de unidade (unitário)	Teste do menor componente possível de um sistema – normalmente em orientação a objetos é uma classe ou método.
Teste de integração	Garante que um ou mais componentes combinados (unidades) funcionam. Um teste de integração é composto por diversos testes de unidade.
Teste operacional	Garante que a aplicação pode rodar por muito tempo sem falhar
Teste positivo-negativo	Garante que a aplicação vai funcionar no seu 'caminho feliz' de seu fluxo de execução.
Teste de regressão	Toda vez que algo for modificado, toda a aplicação deve ser testada novamente, evitando que ela 'regrida' para um estado anterior.
Teste funcional	Testa as funcionalidades, requisitos, regras de negócio presentes na documentação. Valida as funcionalidades descritas na documentação (ou detecta problemas nela).

Tipos de Testes

- Tipos de testes

Tipo de Teste	Descrição
Teste de caixa-preta	Testa todas as entradas e saídas desejadas. Não se está preocupado com o código, cada saída indesejada é considerada um erro.
Teste de caixa-branca	O objetivo é testar o código. Pode acontecer de partes do código nunca serem testadas (cobertura de testes não é de 100% completa).
Teste de *interface*	Verifica se a navegabilidade e os objetivos da tela funcionam como especificados e se atendem da melhor forma ao usuário.
Teste de performance	Verifica se o tempo de resposta é o desejado para o momento de utilização da aplicação.
Teste de carga	Verifica o funcionamento da aplicação com a utilização de uma grande quantidade simultânea de usuários.
Testes de aceitação	Testa se a solução está sendo bem vista/aceita pelo usuário.

Tipos de Testes

- Tipos de testes

Tipo de Teste	Descrição
Teste de volume	Testa a quantidade de dados envolvidos.
Teste de *stress*	Submete o *software* a situações extremas.
Teste de configuração	Testa se a aplicação funciona corretamente em diferentes ambientes de *hardware* ou *software*.
Teste de instalação	Testa se a instalação da aplicação foi bem sucedida.
Testes de segurança e penetração	Testa a segurança da aplicação das mais diversas formas. Utiliza os diversos papéis, perfis e permissões para navegar no sistema.
Teste A/B	Realiza comparações entre duas versões diferentes (de uma *interface* por exemplo), procurando identificar qual delas gera as melhores respostas.
Teste e2e (*end-to-end*)	Testar um fluxo da aplicação desde o começo até o fim.

DEFINIÇÕES

- Erro, defeito, falha segundo padrões do IEEE
  - Engano (*mistake*): Ação humana que introduz um erro
  - Erro (*error*): Problema de qualidade interno do *software* causada por um engano, é a diferença entre o resultado obtido e o resultado esperado, podendo gerar um defeito
  - Defeito/falta (*bug*/*fault*): Manifestação do erro, sendo uma imperfeição em um produto (percepção externa que algo não se comportou como era esperado) que pode ou não acarretar em falha
  - Falha (*failure*): Incapacidade do *software* em cumprir seu objetivo

UNOESC
Apoiadores:
UNOESC
CITEOSC
INOVALE
CENTRO DE INOVAÇÃO VIDEIRA
DEV-TI
SC TEC
fapesc
SANTA CATARINA

DEFINIÇÕES

- Exemplo

modulo2 - teste.py

```
1  def soma(a, b):
2      return a / b
3
4  print( soma(10, 0) )
```

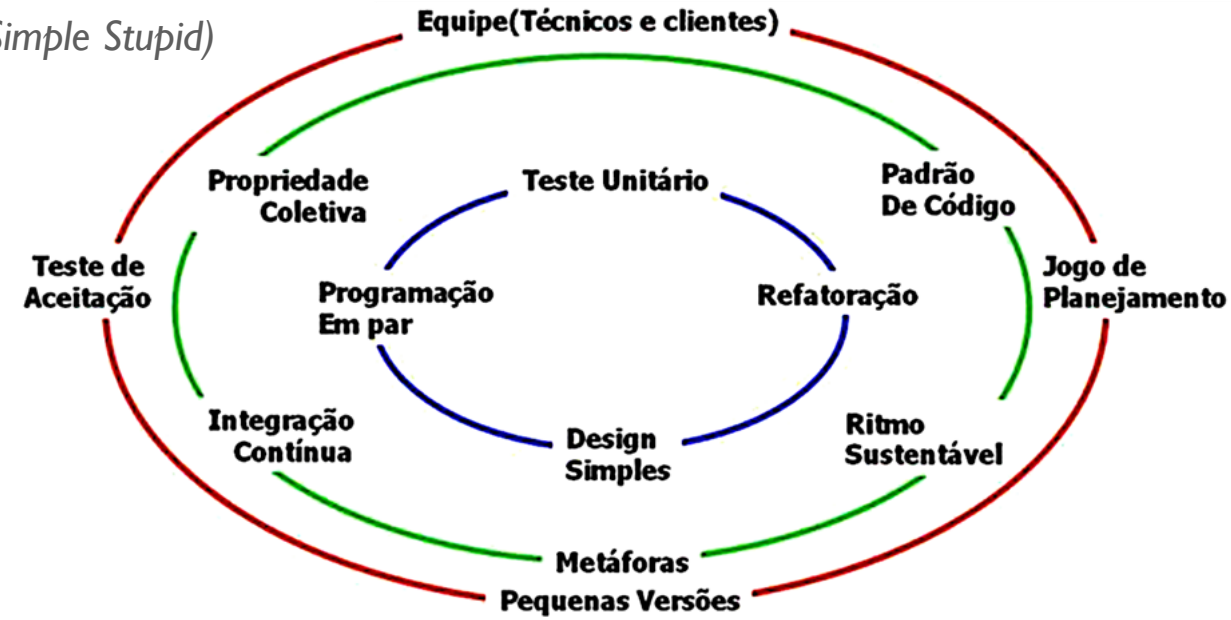
- Em resumo
  - Engano: O programador trocou o símbolo $+$ por $/$
  - Erro: O programa executará a / b em vez de $a + b$
  - Defeito: O programa divide as entradas em vez de somá-las
  - Falha: O programa irá abortar (*crash*)

Apoiadores:
UNOESC CITEOSC INOVALE CENTRO DE INOVAÇÃO VIDEIRA DEV-TI SC TEC fapesc GOVERNO DE SANTA CATARINA

TDD (*Test-Driven Development*)

- TDD não é uma forma de escrever testes e sim de desenvolver código
  - TDD: ***T**est-**D**riven **D**evelopment* (Desenvolvimento Guiado por Testes)
  - TDD: **T**este primeiro, **D**epois **D**esenvolva ☺

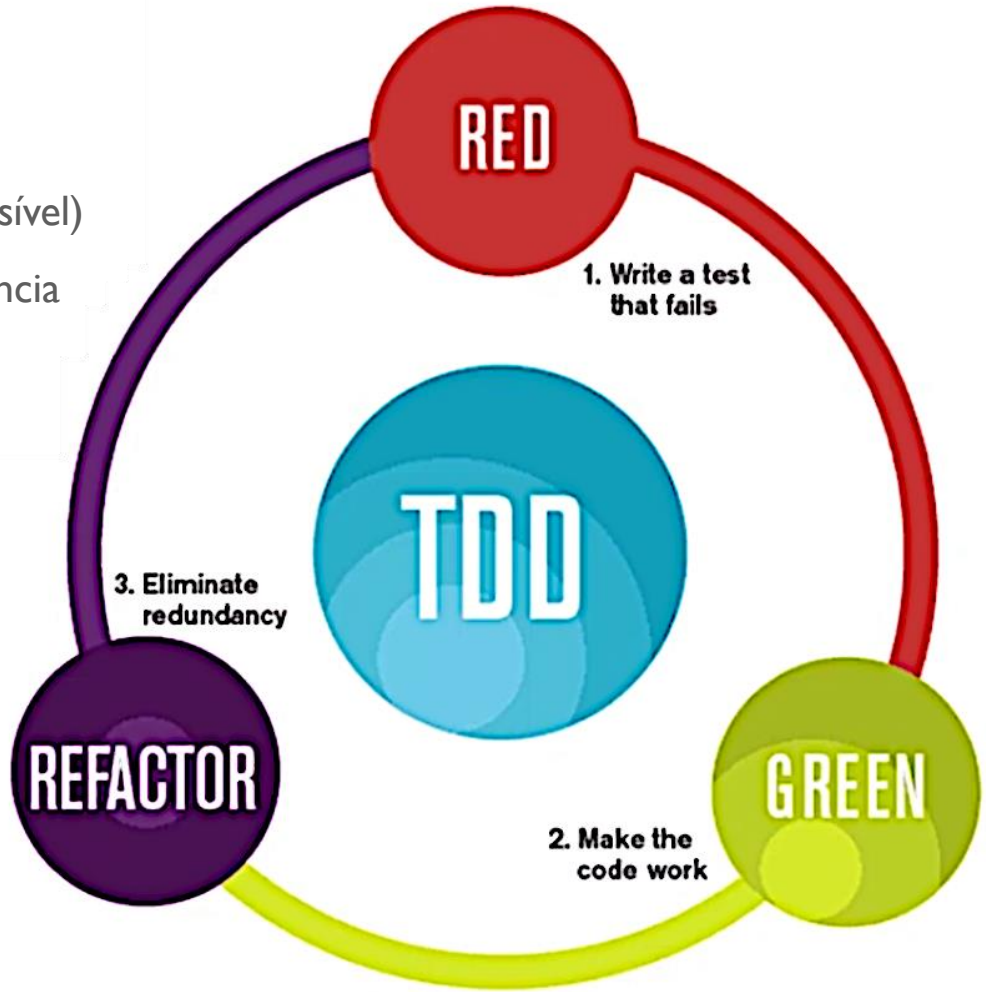
- Prática de desenvolvimento indicada pelo XP (e*Xtreme Programming*)
  - Criado em 2003 por Kent Beck (um dos criadores do XP)
  - XP envolve quatro atividades: Planejamento, projeto (*design*), codificação e testes
  - Princípio KISS (*Keep It Simple Stupid*)



UNOESC

TDD (*Test-Driven Development*)

- ‘Mantra’ do TDD
  1. Escreva um teste que falhe
  2. Crie um código que funcione (da maneira mais simples possível)
  3. Refatore para evitar redundância
  - Repita o ciclo (primeiro faça, depois faça certo e depois faça melhor)



The mantra of Test-Driven Development (TDD) is “red, green, refactor.”

UNOESC