Testes automatizados com práticas ágeis

Roteiro

- Introdução
- Técnicas e Metodologias
- Tipos de Testes
- Conclusão

Roteiro

- Introdução
- Técnicas e Metodologias
- Tipos de Testes
- Conclusão

Introdução

- Cenários comuns de desenvolvimento
- Problemas de testes manuais
- Automação de testes

Cenário Comum de Desenvolvimento

- <Implementar> → <Testar Manualmente>
- <Implementar> → <Equipe de testes>

Problemas do Cenário Comum de Desenvolvimento

- Atraso na entrega
- Dificuldade de manutenção
- Dificuldade de Evolução
- Grande quantidade de erros

Problemas do Cenário Comum de Desenvolvimento

- Processo custoso, repetitivo e tedioso
- Testar um caso de teste é rápido, testar vários não é!
- É comum que casos de testes já testados não sejam testados novamente após uma alteração no código.
- Erros, erros e mais erros..

Automação de Testes

Definição

"Usar software para controlar a execução de testes de software"

Software testando software!

Vantagens da Automação de Testes

- Exercício das funcionalidades do sistema com validação automática dos resultados e efeitos colaterais obtidos
- Todos os casos de testes podem ser repetidos facilmente e sem esforço
- Soluções mais elaboradas e complexas

Vantagens da Automação de Testes

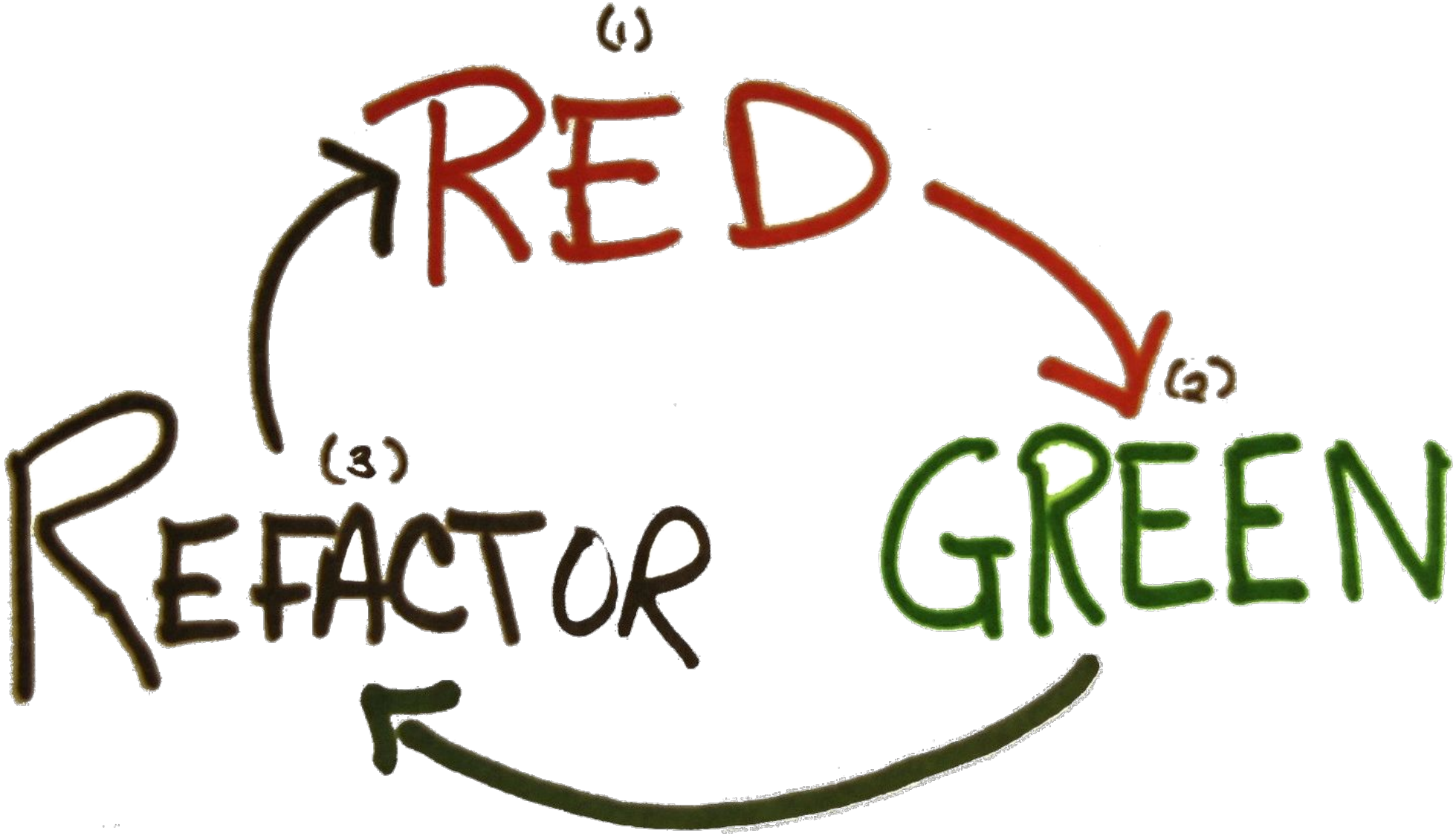
- Testes de maior magnitude
  - Ex: Simular milhares de usuário utilizando o sistema
- Situações específicas podem ser simuladas identicamente e inúmeras vezes

Roteiro

- Introdução
- Técnicas e Metodologias
- Tipos de Testes
- Conclusão

Test Driven Development - TDD

Test Driven Development - TDD



Simplicidade e Baby Steps

1. Para cada parte de uma funcionalidade da aplicação, escreva testes para validar essa funcionalidade (Baby Steps)

2. Escreva a quantidade exata de código para fazer os testes passarem

3. Refatore o código e os testes

TDD como prática de testes

Prática de Testes:

- Código nasce testado

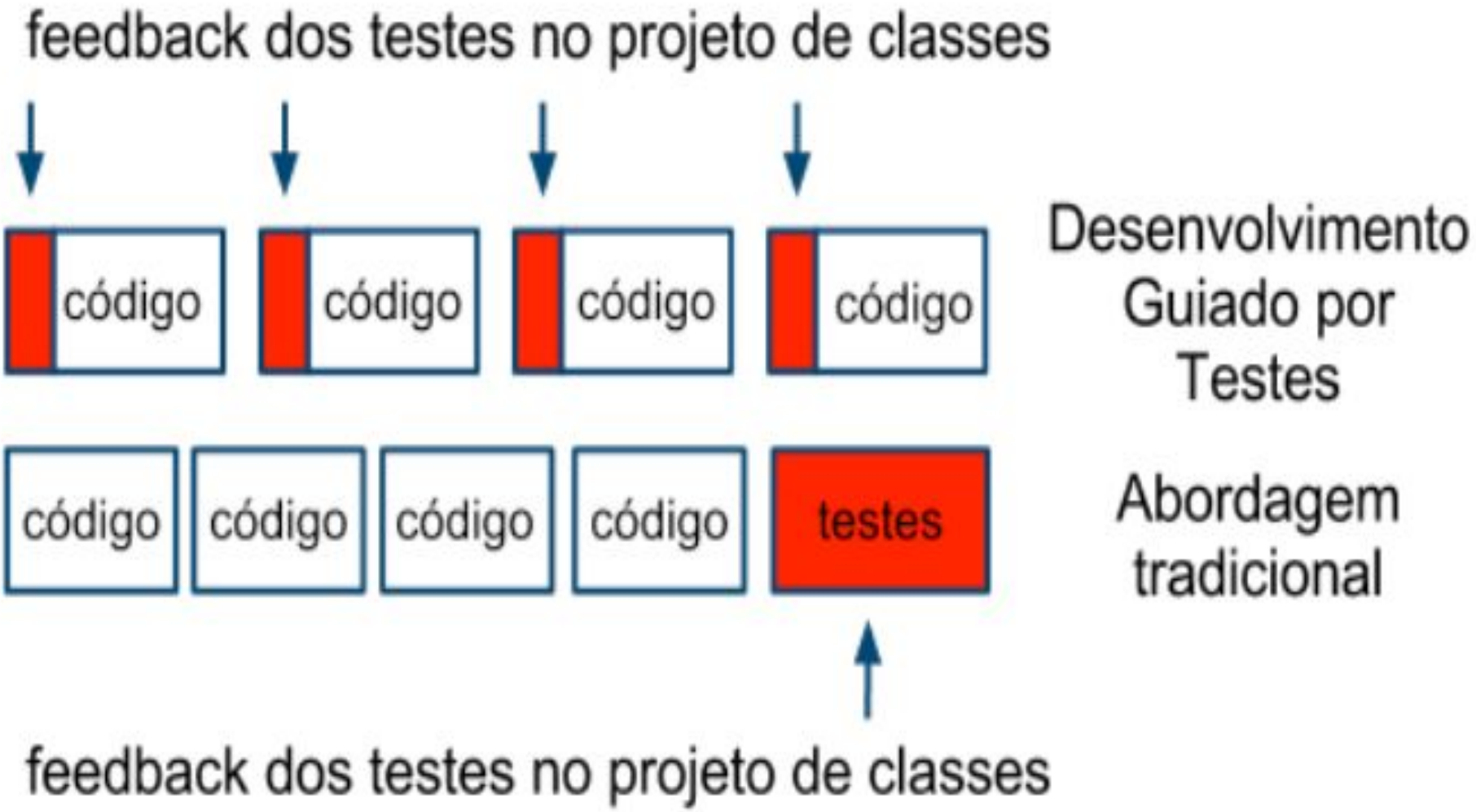
Prática de design:

- Tem como objetivo guiar o projeto e implementação

- Com a prática do TDD, seu projeto de classes fica melhor e mais simples*

Erros Comuns

- Esquecer de refatorar constantemente
- Refatorar outro trecho de código
- Não começar pelo teste mais simples possível
- Não fazer baby steps


feedback dos testes no projeto de classes
código
código
código
código
Desenvolvimento Guiado por Testes
código
código
código
código
testes
Abordagem tradicional
feedback dos testes no projeto de classes

Roteiro

- Introdução
- Técnicas e Metodologias
- Tipos de Testes
- Conclusão

Tipos de Testes

- Testes Unitários
- Testes de Integração
- Testes de Sistema
- Outros...

Testes Unitários


VIDADEPROGRAMADOR
.COM.BR
real historia;
string sender;
sender= "RASCHADECK";
BY NC SA
#1069
ENTÃO, P.A., TE ENCAMINHEI ESSE CÓDIGO PARA FAZER O TESTE UNITÁRIO DO PROJETO...
Ask Google
VEJO QUE OS ASSASSINOS FALHARAM
PROGRAMADOR, ESSE AÍ É AQUELE PROJETO QUE A GENTE ESTAVA VENDO?
AHAN...
Ask Google
VEJO QUE OS ASSASSINOS
ESSE PROJETO É MUITO COMPLEXO PARA APENAS UM TESTE... PEDE LOGO UNS 3...
PLOFT!

Testes Unitários

- Tipo mais “básico” de teste
- Uma unidade é um trecho de código
  - Método, Função, Módulo, Componente,...
- Foca um trecho específico de código, diminuindo tempo com debug
  - Dados de entrada e saída
  - Dados armazenados em estrutura de dados
  - Tratamento de erros
- Já escreveu um main() pra testar um método?

Testes Unitários

- Segundo Vicent Massol, autor do livro JUnit in Action, “Um teste unitário examina o comportamento de uma unidade distinta de trabalho.”
- Nesta definição, dois termos merecem ser destacados: **comportamento e unidades de trabalho.**
- Um teste unitário deve ser capaz de examinar o comportamento do código sob as mais variadas condições.
- Como o código deve se comportar se determinado parâmetro for passado (ou não), o que ele retorna se determinada condição for verdadeira, os efeitos que ele causa durante a sua execução, etc.

Testes Unitários

- Como o nome sugere, devem testar unidades de trabalho isoladas.
- E geralmente, em termos de códigos orientados a objeto, estas unidades de trabalho **são métodos de uma classe**.
- Isoladas, é realmente isoladas! Um teste unitário, tipicamente, testa aquele e somente aquele método, evitando acesso à outros recursos como sistema de arquivos, banco de dados, rede etc.
- Para testar métodos que fazem uso desses recursos em suas implementações, devemos fazer uso **de stubs e/ou mocks**.

Vantagens dos Testes Unitários

- Testes unitários permitem maior cobertura de teste
- Testes unitários previnem regressão
- Testes unitários incentivam o refactoring
- Testes unitários evitam longas sessões de debug
- Testes de unidade servem como documentação

- ***Dizem que: "O teste é o primeiro cliente do seu código e o ditado diz que o cliente sempre tem a razão."***

Frameworks

Java:

Junit

...

...

.Net:

Nunit

...

...

Mocks

Mocks

- Simulam o comportamento de objetos reais
  - Dependências em métodos testados unitariamente
- Objetos difíceis de serem criados ou incorporados nos testes.
  - Ex: Serviços Externos

Quando usar mocks?

- Objetos que geram resultados não determinísticos
- Objetos com estados difíceis de serem criados ou reproduzidos
- Serviços Externos
- Componentes lentos
- Objetos que ainda não existem ou não podem ser incorporados aos testes

Evitar uso excessivo

- Se o código for muito alterado pode ser necessário alterar os testes, mesmo que a lógica do código seja a mesma

- Pode mascarar algum erro

Framework Java:

Mockito

EasyMock

MockCreator

HibernateMock

Jmock

Jmockit

...

Framework .NET:

NMockLib

Rhino Mocks

NMock

NMock 2

TypeMock

DOJO

Coding DOJO

.**Motivação:** Programador não treina

.**Origem:** França

.**Objetivo:** treinar e não derrubar o adversário

.**Princípios:**

- aprendizado contínuo, ambiente seguro (não competitivo, colaborativo, inclusivo), passos de bebê

.**Retrospectiva:** sugestões/comentários/...

Formato Randori - Coding Dojo

- Programação em pares

- Turnos "time-boxed"

- Após o turno (5-7 min):
  - O co-piloto vira piloto
  - O piloto volta para a plateia
  - Um novo co-piloto é convidado da plateia

- Comentários e críticas somente após o **verde**

- **Silêncio no vermelho**

Hands On

JUNIT

import org.junit.*	Import das declaração para usar anotações.
@Test	Identifica um método como um método de teste.
@Before	Executado antes de cada teste. Ele é usado para preparar o ambiente de teste (por exemplo, ler dados de entrada, inicializar a classe, etc.).
@After	Executado após cada teste. Ele é usado para limpar o ambiente de teste (por exemplo, excluir dados temporários, restaurar padrões). Também pode economizar memória limpando estruturas de memória caras.
@BeforeClass	Executado uma vez, antes do início de todos os testes. Ele é usado para executar atividades intensivas de tempo, por exemplo, para conectar-se a um banco de dados. Os métodos marcados com esta anotação precisam ser definidos como estáticos para funcionar com o JUnit.
@AfterClass	Executado uma vez, depois de todos os testes terem sido concluídos. Ele é usado para executar atividades de limpeza, por exemplo, para desconectar de um banco de dados. Os métodos anotados com essa anotação precisam ser definidos como estáticos para funcionar com o JUnit.

JUNIT

@Ignore or @Ignore("Why disabled")	Marca que o teste deve ser desativado. Isso é útil quando o código foi alterado e o caso de teste ainda não foi adaptado. Ou se o tempo de execução deste teste for muito longo para ser incluído. É uma prática recomendada fornecer a descrição opcional, informando o porquê do teste está desativado.
@Test(timeout=100)	Falha se o método demorar mais de 100 milissegundos.

Biblioteca Virtual– BIVA

- **Crud de Livro**
  - salvar, editar, inativar, atualizarNumeroDeExemplares
- **Crud de Aluno**
  - salvar, editar, inativar.
- **Crud de Empréstimo de Livro**
  - Alugar até 3 livros;
  - Reservar livro caso esteja alugado;
  - Bloquear emprestimo quando:
    - Atingir o limite máx.
    - Estiver com algum livro atrasado;
  - Renovar empréstimo:
    - Se não existir uma reserva para esse livro;

Biblioteca Virtual– BIVA

- **Multa***
  - Enviar e-mail quando faltar 1 e 2 dias para o prazo da entrega;
  - Calcular a multa considerando os dias da semana (excluir sábados e domingos);
    - Cada dia 0,5 centavos de multa por livro
  - Conceder desconto de 20% quando atingir 10-15 dias de atraso (Notificar estudante via e-mail quando o desconto estiver disponível);
  - Conceder desconto de 25% quando atingir 16-29 dias de atraso (Notificar estudante via e-mail quando o desconto estiver disponível);
  - Conceder desconto de 30% os dias de atraso for superior a 30 dias (Notificar estudante via e-mail quando o desconto estiver disponível)