Desenvolvimento Dirigido por Testes

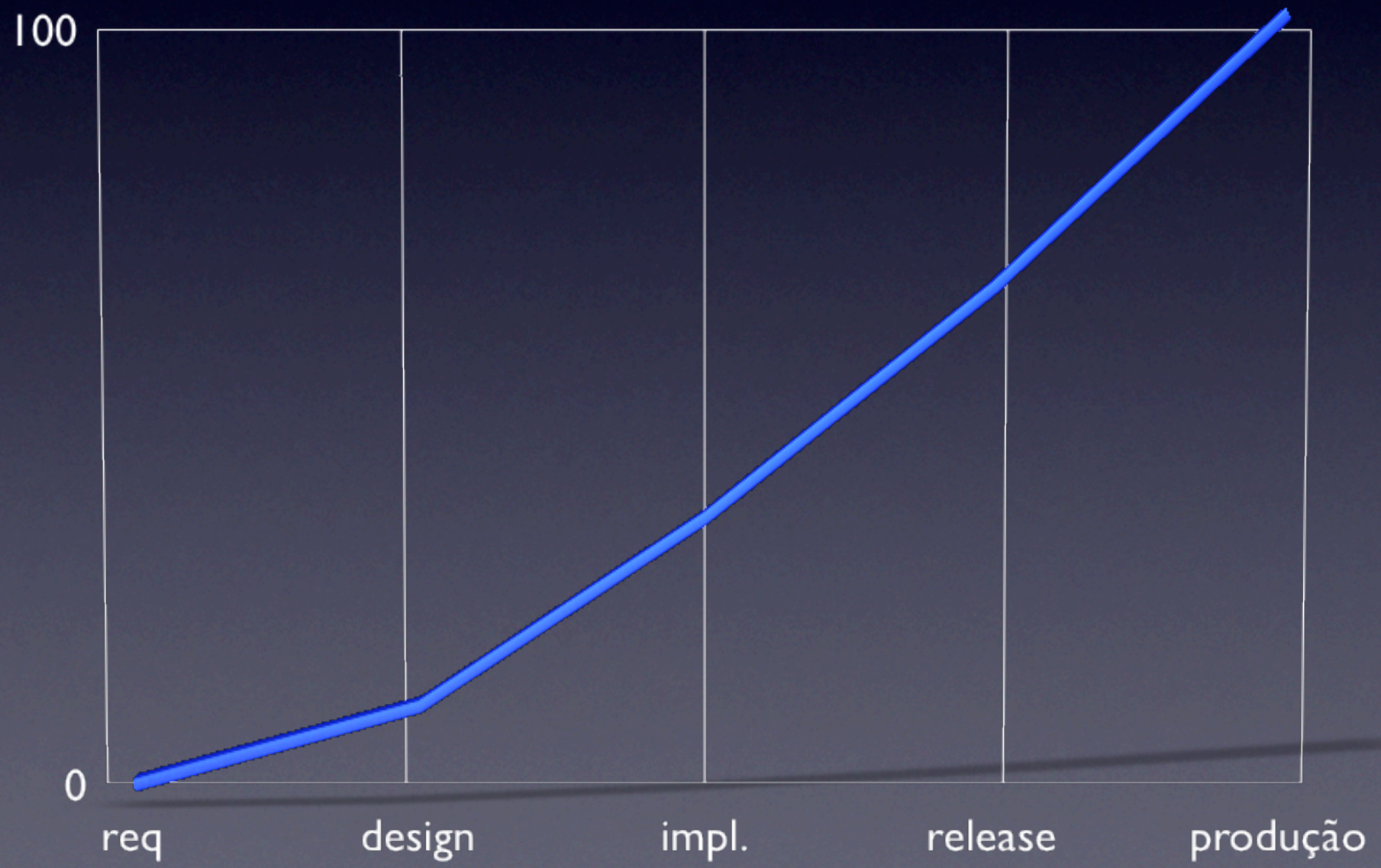
Test-Driven Development - TDD

Dairton Bassi
dbassi@gmail.com

O que é TDD?

"Encontrar e corrigir um defeito em um software depois de entregue é 100 vezes mais caro do que encontrá-lo e corrigi-lo durante as requisições ou na fase de design."

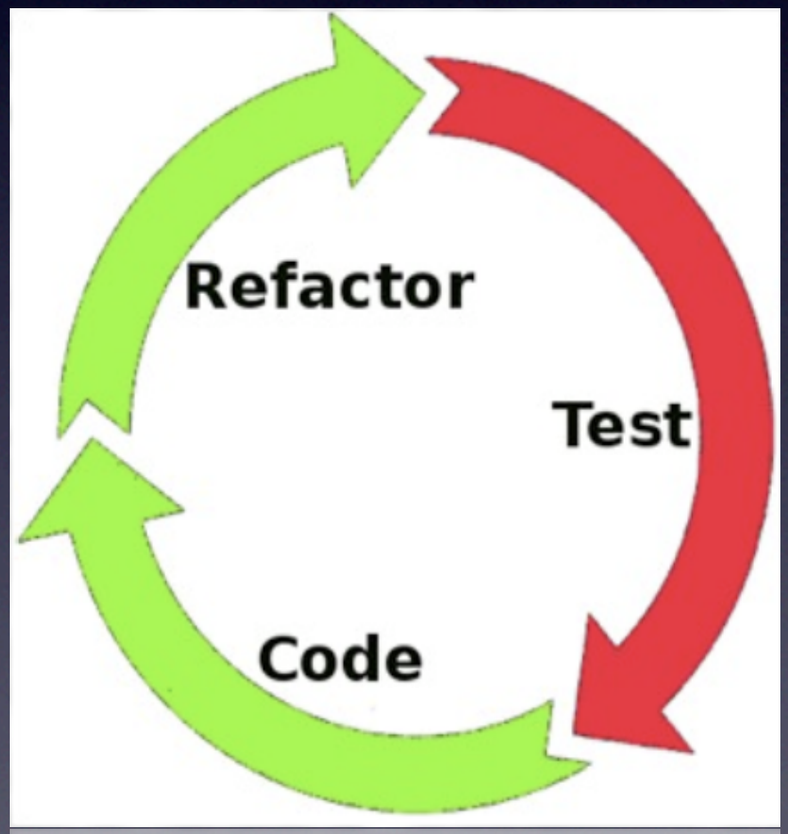
Shull, Basili, Boehm e outros (2002)



MAS EU FAÇO TESTES!

- E a cobertura do código?
- Você tem coragem para refatorar?
- O design é adequado?
- O que isso deveria fazer?

TDD =
TestFirst + Refactoring



Testes Automaticos

```java
public boolean ePrimo(int numero) {

    for (int i = 2; i < numero; i++) {

        int resto = numero % i;

        if (resto == 0)
            return false;
    }
    return true;
}
```

```java
@Test
public void testa2() {
    Primo primo = new Primo();

    assertTrue(primo.ePrimo(2));
}
```

```java
@Test
public void testa3() {
    Primo primo = new Primo();

    assertTrue(primo.ePrimo(3));
}
```

```java
@Test
public void testa4() {
    Primo primo = new Primo();
```

```java
@Test
public void testa21() {
    Primo primo = new Primo();
```

```java
@Test
public void testa23() {
    Primo primo = new Primo();

    assertEquals(true, primo.ePrimo(23));
}
```

Teste cedo, sempre e automaticamente

- Escreva uma vez, rode sempre
- 1 click
- Sem entrada humana
- Saída analisada automaticamente

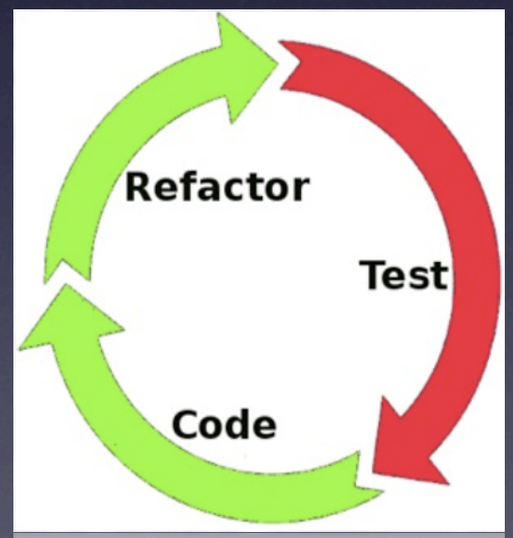
Passou

Falhou

Refatoração

Uma modificação no sistema que não altera o seu comportamento funcional, mas melhora sua estrutura interna.

- Simplicidade
- Flexibilidade
- Clareza
- Desempenho*

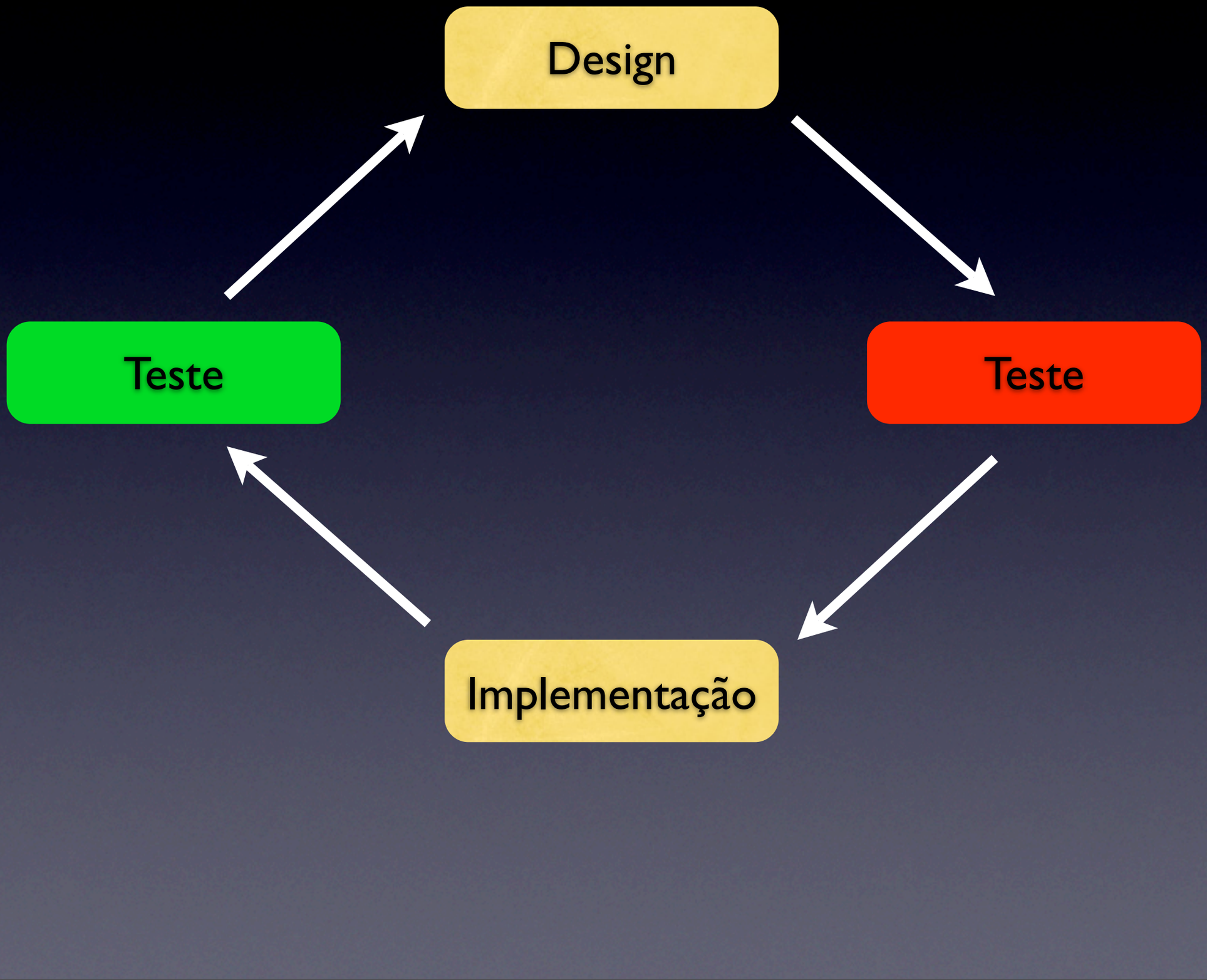


Entendendo melhor...

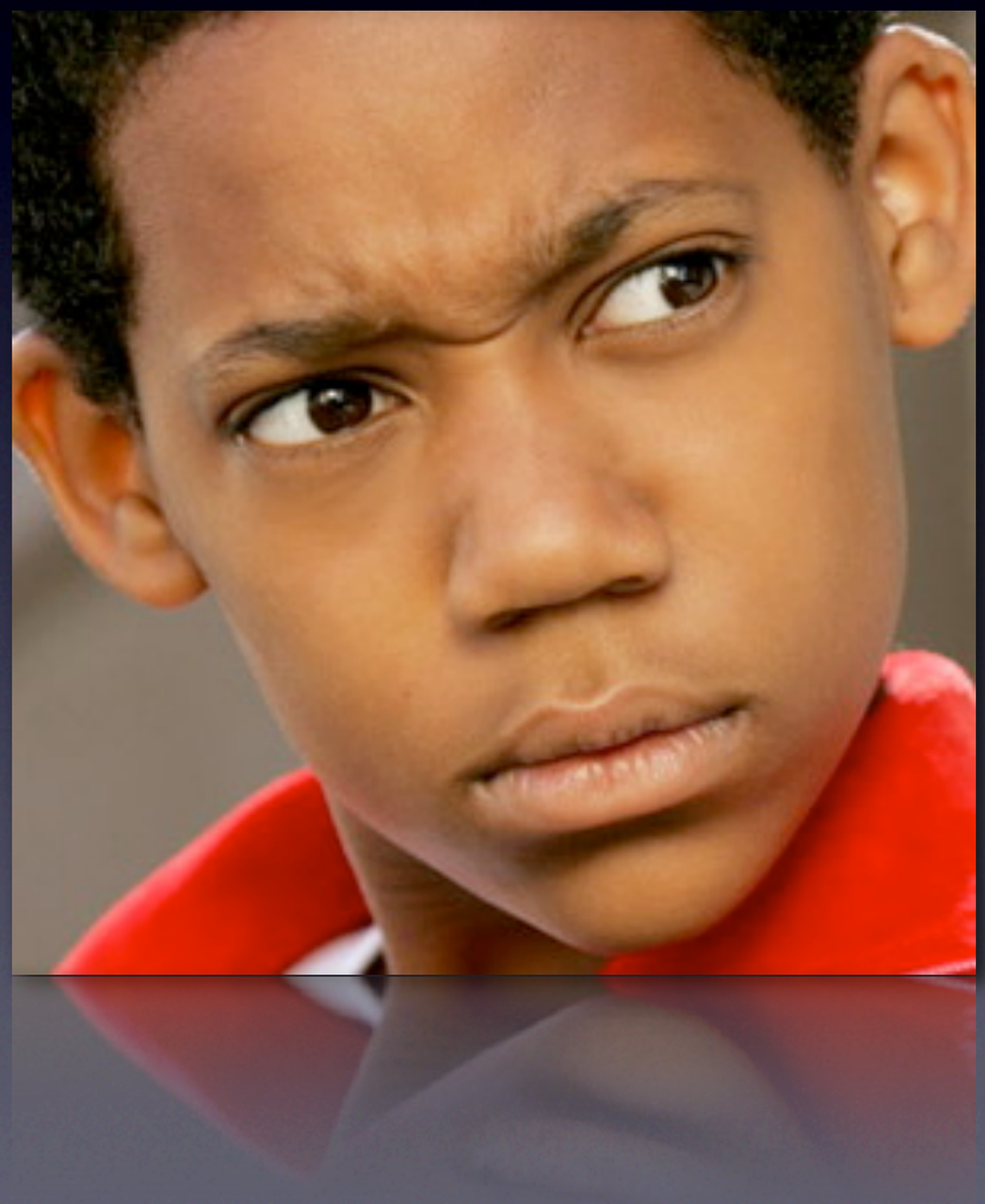
Olhando a interface

Conhecendo o código

Código refatorado


Design
Teste
Teste
Implementação

Como assim?

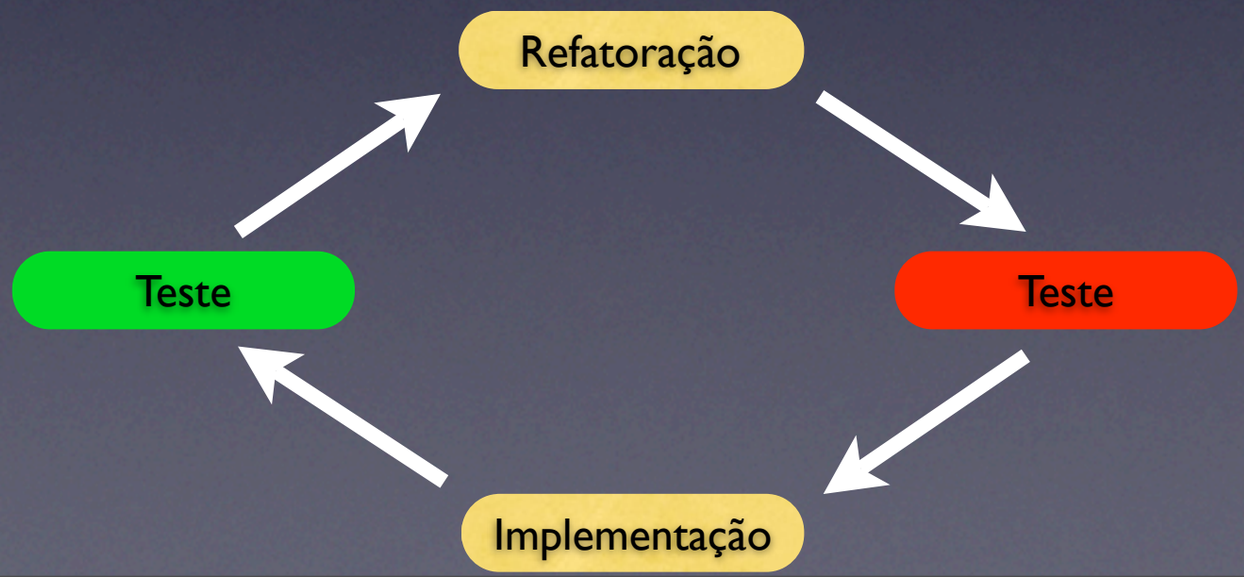


1. ~~***Design:*** entenda o que você precisa fazer~~
   Refatoração: melhore a qualidade do código

2. ***Teste:*** escreva um teste que expresse o seu design - deve **falhar**.

3. ***Implementação:*** escreva o código.

4. Teste novamente - deve **passar**.



3 regras de TDD

1. Não escreva código se não for para fazer algum teste passar.

2. Não escreva mais testes do que o suficiente para falhar.

3. Escreva só o código necessário para fazer os testes passarem.

Problema / Demo

Dado um número **x**, determine dois números primos: p_1 e p_2, tal que $max(p_1) < \mathbf{x} < min(p_2)$.

Na entrega

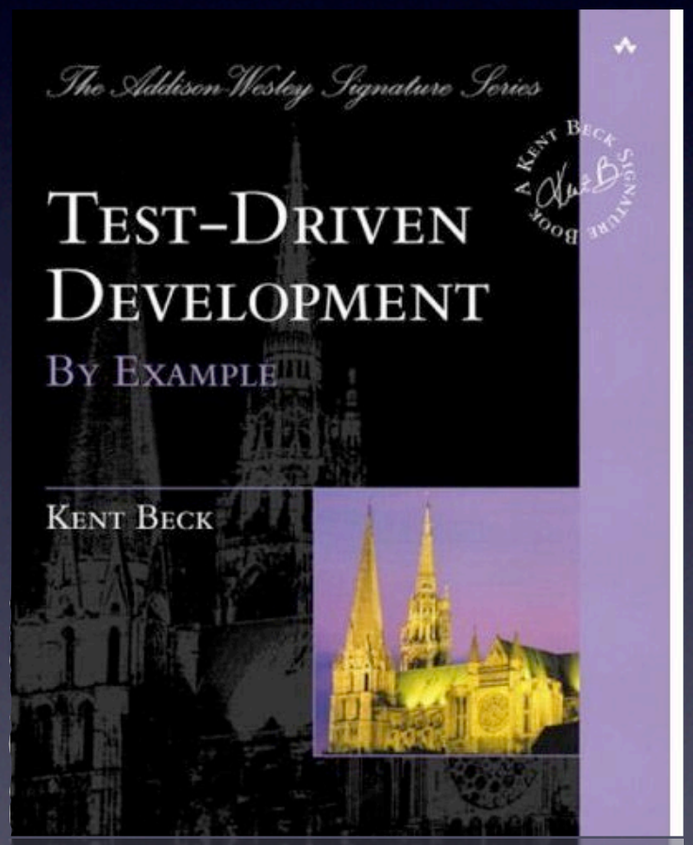
- Todas as funcionalidades requeridas.
- Somente as funcionalidades requeridas.
- Cobertura total do código.
- Evidências de qualidade.
- Design na medida certa.

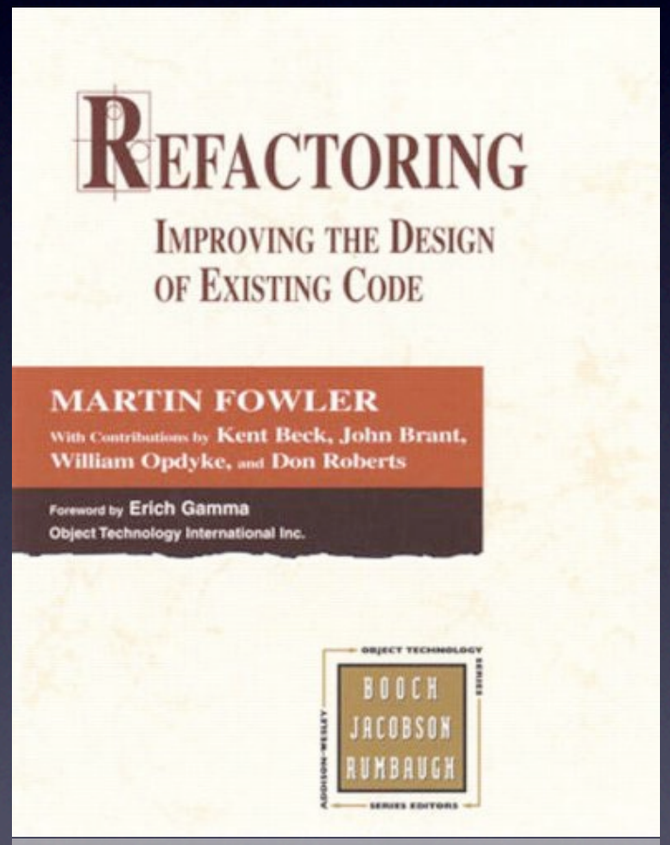
Desenvolvimento voltado para o desenvolvimento

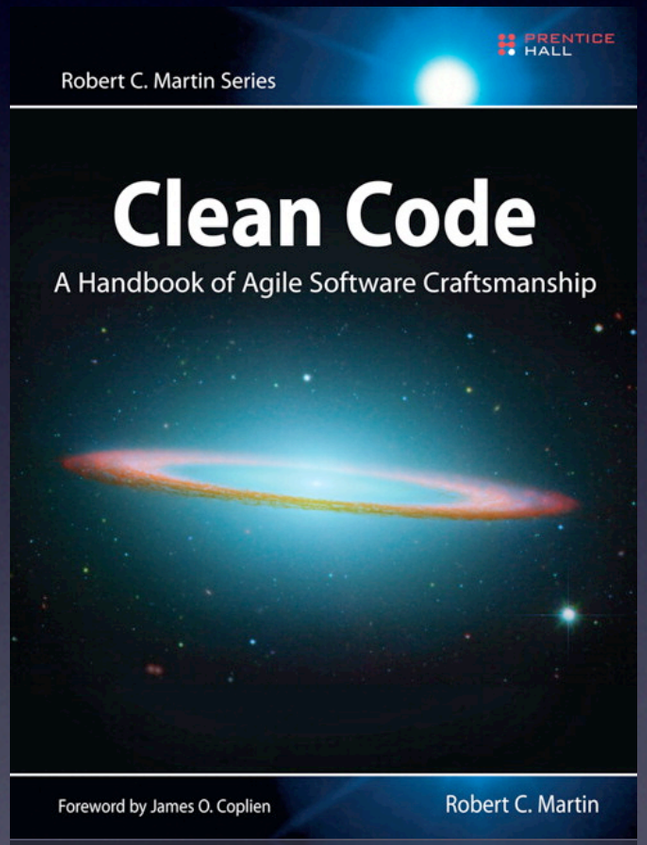
- Testes não ficam para depois.
- Mais confiança no código.
- Facilidade para mudanças.
- Aumento da qualidade do código.
- Não há complexidade desnecessária.
- Alta coesão e baixo acoplamento.

Teoria das Janelas Quebradas

Leitura Recomendada







Obrigado!

NEUROBOX

Dairton Bassi
dbassi@gmail.com
twitter: dbassi

?