Introdução ao

TDD – TEST-DRIVEN DEVELOPMENT

Desenvolvimento Guiado por Testes

POR QUE DEVEMOS TESTAR?

Gastos com bugs

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

Consequências de um bug [1]

O foguete Ariane 5 explodiu por um erro de software.

Consequências de um bug [2]

Um hospital panamenho matou pacientes pois seu software para dosagem de remédios errou.

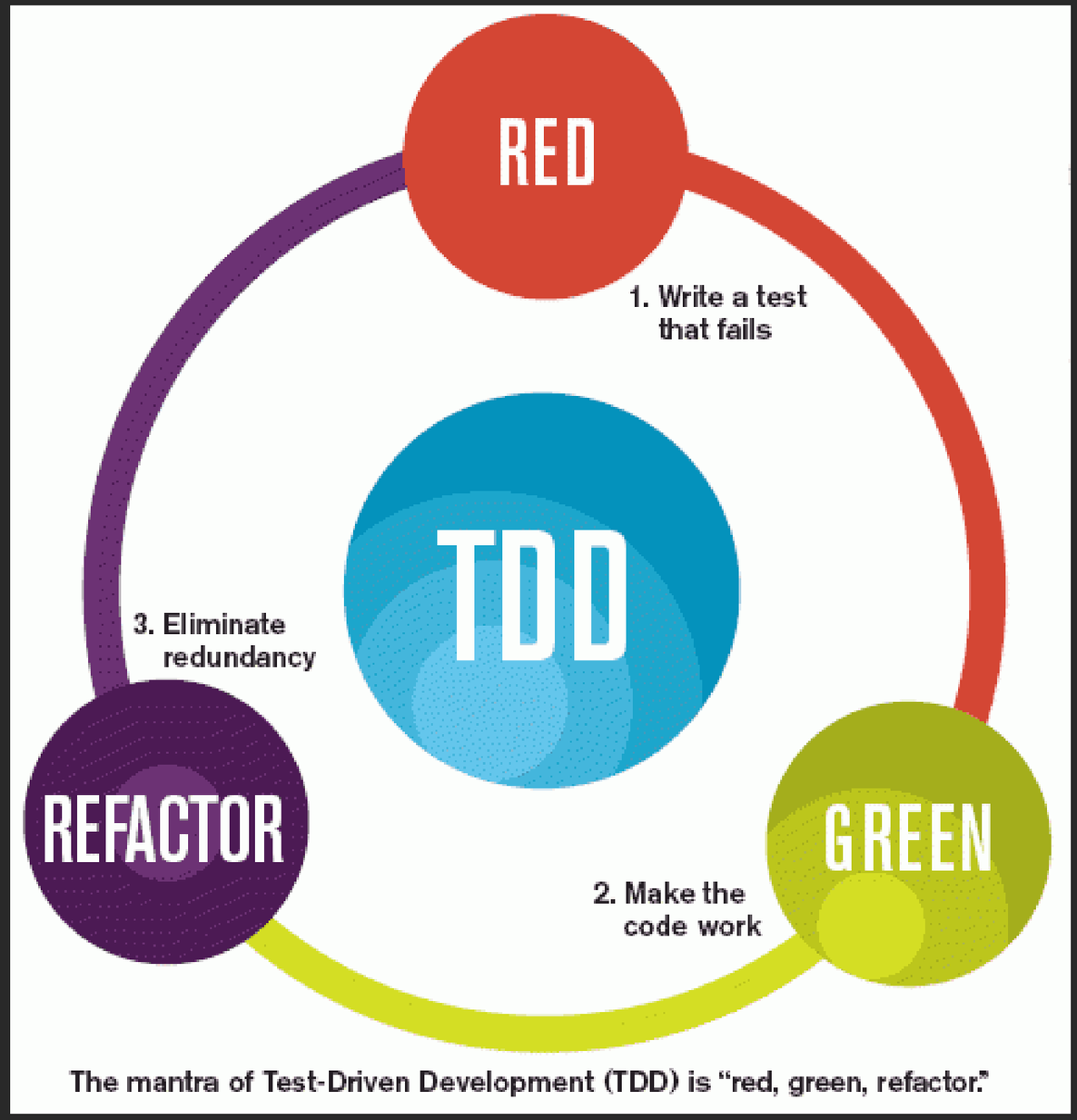
UNIT TEST (TESTE DE UNIDADE / TESTE UNITÁRIO)

A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. In most programming languages, that is a function, a subroutine, a method or property.

COMO FUNCIONA O TDD ?

O ciclo vermelho-verde-refatora

- Escrevemos um teste de unidade para uma nova funcionalidade;
- Vemos o teste falhar;
- Implementamos o código mais simples para resolver o problema;
- Vemos o teste passar;
- Melhoramos (refatoramos) nosso código quando necessário.



Passos de Bebê (ou Baby Steps)

A ideia de sempre tomar o passo mais simples que resolva o problema naquele momento (e faça o teste passar) é conhecido pelos praticantes de TDD como baby steps. A vantagem desses passos de bebê é tentar levar o desenvolvedor sempre ao código mais simples e, por consequência, mais fácil de ser compreendido e mantido posteriormente.

O desenvolvedor deve buscar pela solução mais simples, e não pela modificação mais simples. Veja que a modificação mais simples não é necessariamente a solução mais simples.

QUAIS AS VANTAGENS ?

Foco no teste e não na implementação

Ao começar pelo teste, o programador consegue pensar somente no que a classe deve fazer, e esquece por um momento da implementação. Isso o ajuda a pensar em melhores cenários de teste para a classe sob desenvolvimento

Código nasce testado

Se o programador pratica o ciclo corretamente, isso então implica em que todo o código de produção escrito possui ao menos um teste de unidade verificando que ele funciona corretamente.

Simplicidade

Ao buscar pelo código mais simples constantemente, o desenvolvedor acaba por fugir de soluções complexas, comuns em todos os sistemas.

Melhor reflexão sobre o design da classe

O teste atua como o primeiro cliente da classe que está sendo escrita. Nele, o desenvolvedor toma decisões como o nome da classe, os seus métodos, parâmetros, tipos de retorno, e etc. No fim, todas elas são decisões de design e, quando o desenvolvedor consegue observar com atenção o código do teste, seu design de classes pode crescer muito em qualidade.

O PROBLEMA: NÚMEROS ROMANOS

Dado um numeral romano, o programa deve convertê-lo para o número inteiro correspondente.

7 símbolos

- I, unus, 1, (um)
- V, quinque, 5 (cinco)
- X, decem, 10 (dez)
- L, quinquaginta, 50 (cinquenta)
- C, centum, 100 (cem)
- D, quingenti, 500 (quinhentos)
- M, mille, 1.000 (mil)

Somar

Algarismos de menor ou igual valor à direita são somados ao algarismo de maior valor;

Subtrair

Algarismos de menor valor à esquerda são subtraídos do algarismo de maior valor.

O PROBLEMA: NÚMEROS ROMANOS

Continuação...

Sem repetir mais de 3x

nenhum símbolo pode ser repetido lado a lado por mais de 3 vezes. Por exemplo, o número 4 é representado pelo número IV (5 - 1) e não pelo número IIII.

Mais regras?

Para não tornar o exercício muito complexo vamos ignorar as demais regras existentes.