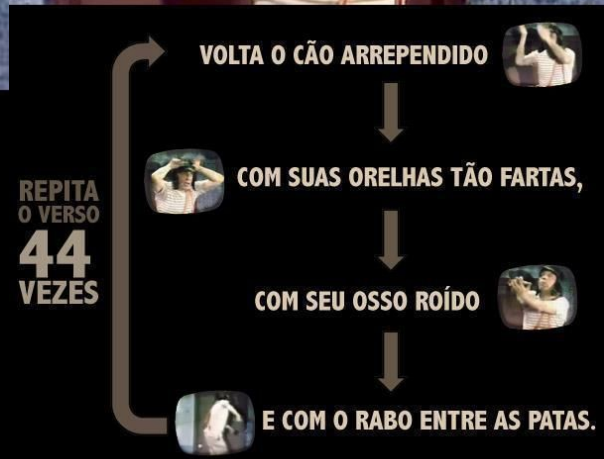
UniCesumar

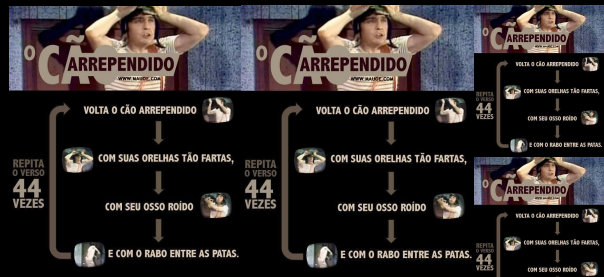
Projeto, implementação e Teste de Software

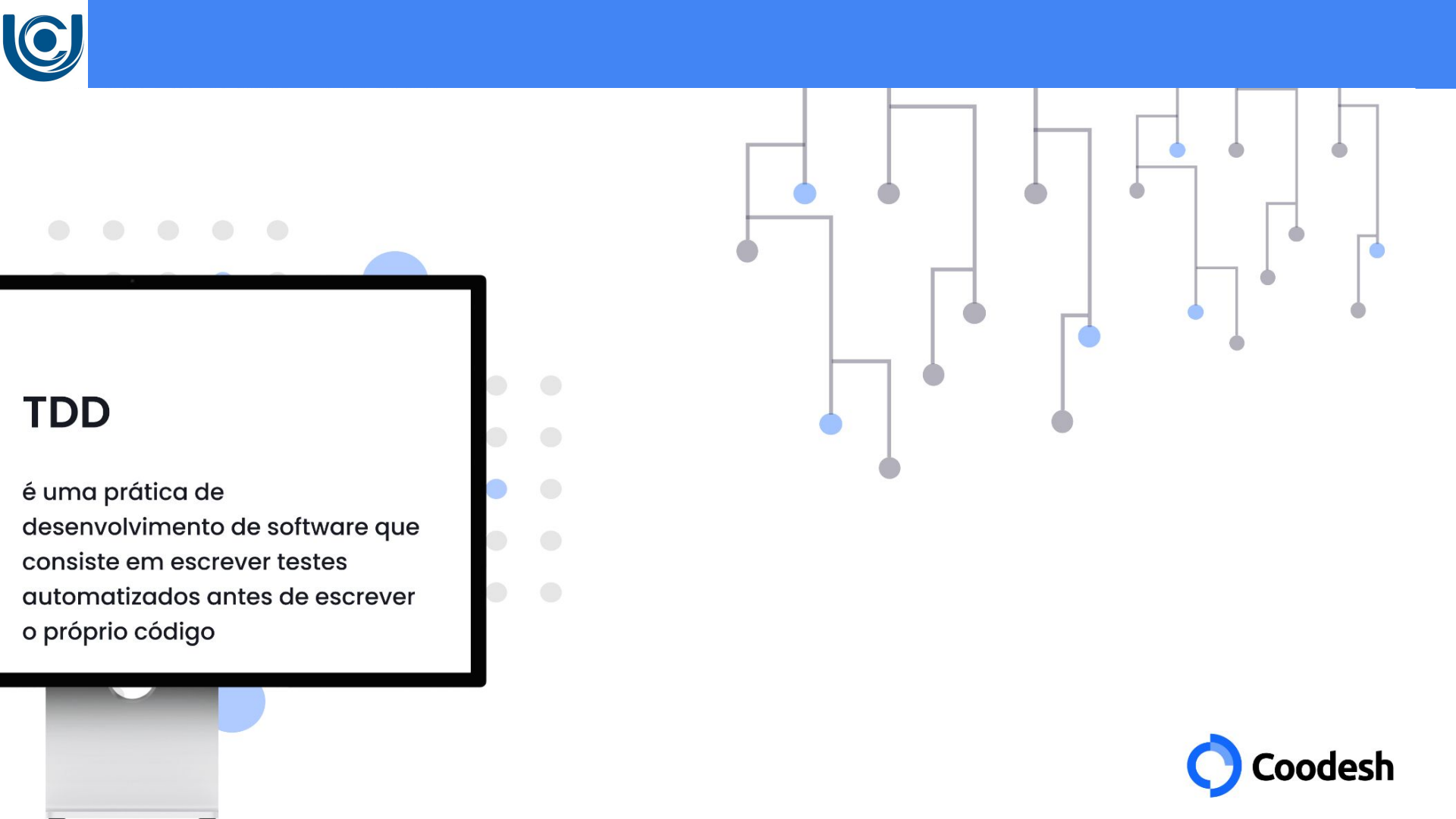
TDD

Prof. Esp. Dacio F. Machado

TDD
Test-Driven Development





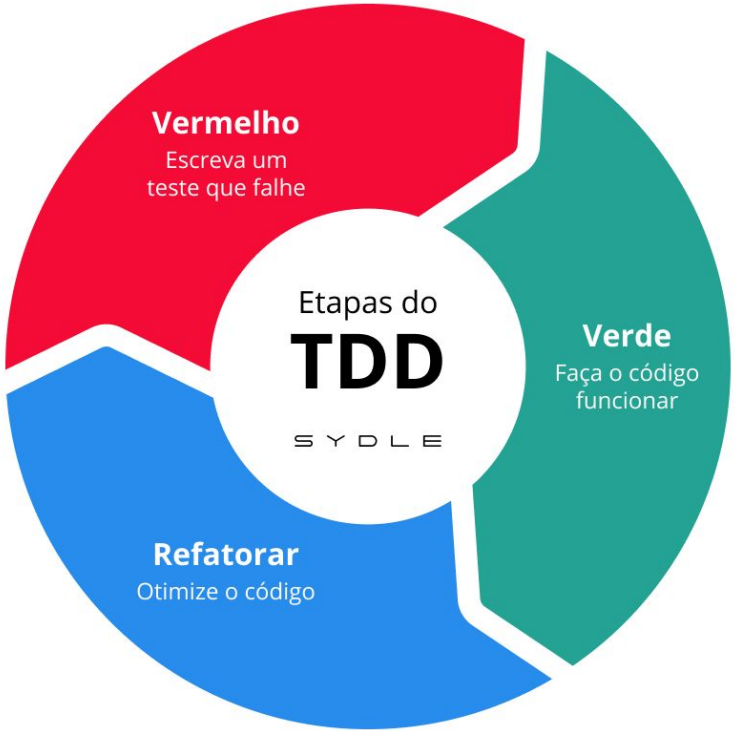

TDD
é uma prática de desenvolvimento de software que consiste em escrever testes automatizados antes de escrever o próprio código

Coodesh

Test-Driven Development

TDD significa Test-Driven Development, que é uma abordagem de desenvolvimento de software que coloca um forte foco na escrita de testes antes de escrever o código real.

Ciclo de desenvolvimento Red,Green, Refactor.



Test (RED)_

O desenvolvedor escreve um teste que descreve a funcionalidade desejada do software.

Este teste inicial geralmente falha, já que a funcionalidade ainda não foi implementada.

Develop (Green)_

O desenvolvedor escreve o código necessário para fazer o teste passar.
O objetivo é fazer com que o teste inicialmente falhado seja bem-sucedido.
Nenhum código é escrito além do necessário para fazer o teste passar.

REFECTOR ()

O desenvolvedor pode refatorar o código para torná-lo mais limpo, eficiente e legível, sem alterar o comportamento observável do software.

Isso ajuda a manter a qualidade do código.

Novo Teste

Se temos uma nova funcionalidade do sistema e fazemos o processo inverso ao tradicional:

Testamos e Codificamos e não Codificamos e Testamos.

No primeiro momento isto parece estranho, esquisito ou feio, mas não é

Teste Falhando

Neste momento, acabamos de escrever o teste e não temos a implementação.

Óbvio que o teste falhará, pois ele espera uma resposta que ainda não temos implementada em lugar algum.

Com um Teste falhando na nossa frente, temos um único objetivo na vida:

Fazê-lo passar!

Nova funcionalidade

Nesse momento vamos codificar a nossa nova funcionalidade da forma mais simples possível para fazer o nosso Teste passar.

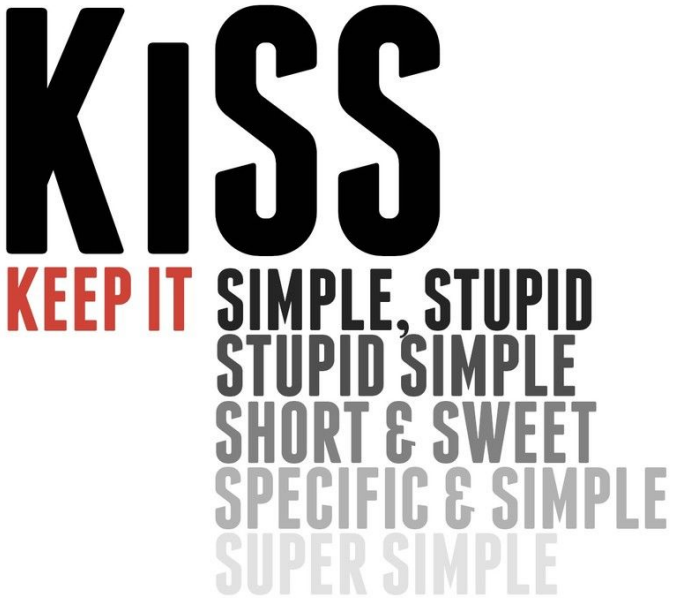
Já ouviu falar no KISS


vevo

Keep It Simple, Stupid

é um princípio de design que defende a simplicidade e a evitação de complexidades desnecessárias em sistemas e projetos.

A ideia central é que os sistemas funcionam melhor quando são mantidos simples, e essa filosofia se originou com a Marinha dos EUA em 1960




Japanese
German

Refatoração

Neste momento é que vamos analisar melhor aquele código que fizemos simplesmente para o nosso Teste passar.

"Single Responsibility Principle". Este princípio nos diz que devemos ter somente um motivo para modificar uma classe. Ou seja, ele fala sobre termos uma classe com somente uma responsabilidade.

Não podemos refatorar um trecho de código e quebrar vários Testes

Testar algo que não existe

Criar algo que passe no teste.

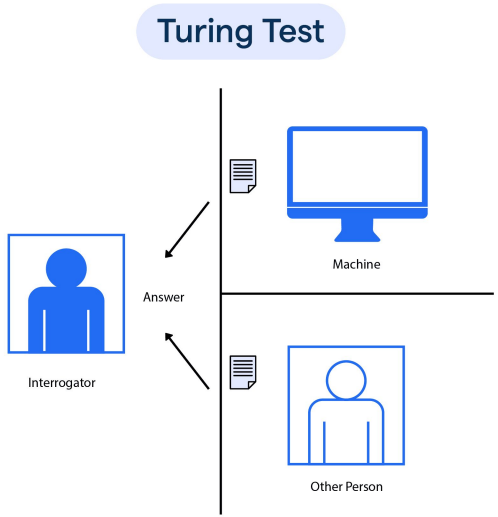
Melhorar o que foi criado

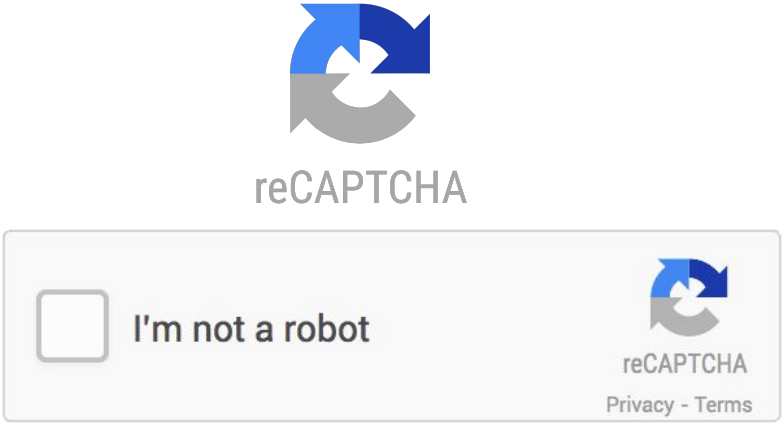
T **D** **D**

Testar algo que não existe

Criar algo que passe no teste.

Melhorar o que foi criado





Aplicando o TDD

- Implementar uma calculadora que realiza operações de soma, subtração, multiplicação e divisão. Pelo método TDD

Aplicando TDD

Criar um projeto no Replit, ou local na máquina para iniciar.

https://replit.com/@daciofrancisco/TDD-Calc#main.py

OBRIGADO

dacio.francisco@unicesumar.edu.br