Residência em Software

Residência em Tecnologia da Informação e Comunicação

Introdução aos princípios de TDD

Professor:
Alvaro Degas Coelho

Java

INSTITUIÇÃO EXECUTORA
cepedi UESC

COORDENADORA
MCTI FUTURO Softex

APOIO
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

A quem cabe a tarefa de testar?

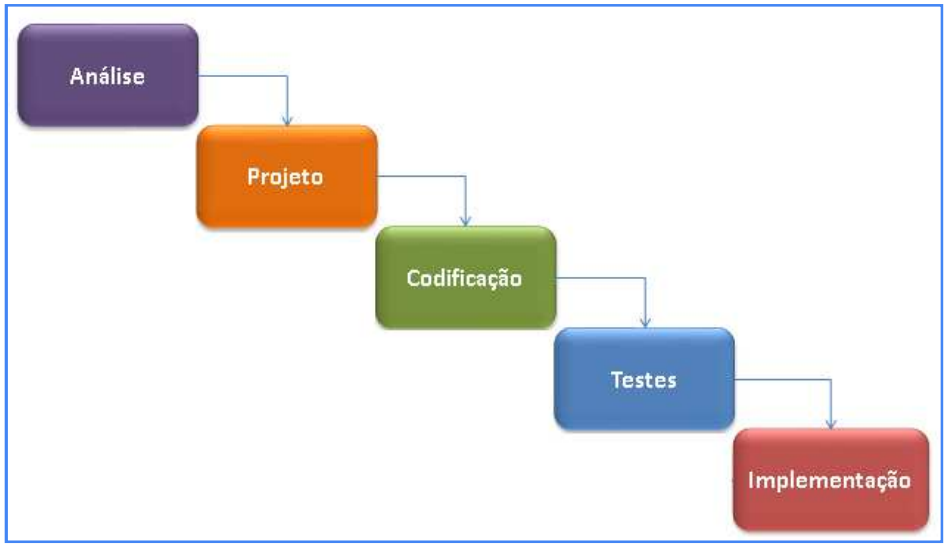
- Fábula “A Coruja e a Águia”
- O programador tem seu olhar (provavelmente) enviesado
  - Entendimento pessoal
  - Decisões de programação
  - Ansiedade para entregar
  - Confiança em si mesmo
- Mundo perfeito: um *“Tester”*
  - Responsável por verificar se o código está correto

Porque não testamos

- Planejamento
  - Pouco (às vezes nenhum) espaço reservado aos testes
  - Não seleciona pessoa especificamente para isto
  - Cultura
  - “Confiança” no trabalho do programador

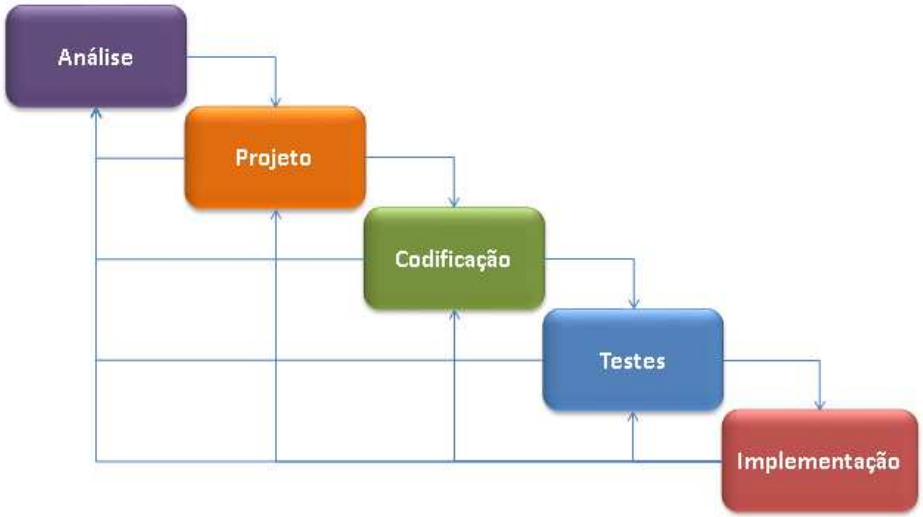
Teste SEMPRE foi entendido como importante

- Modelo em Cascata



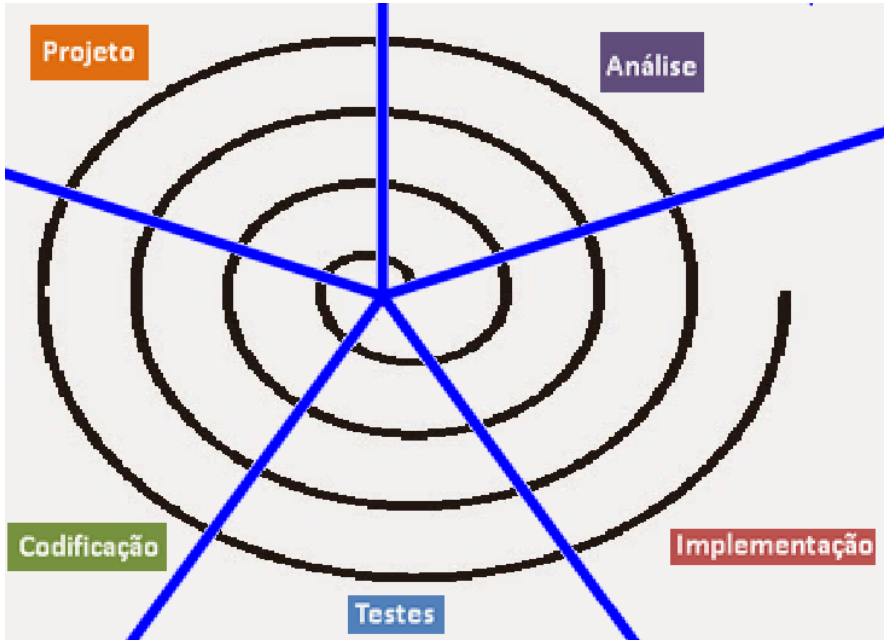
Teste SEMPRE foi entendido como importante

- Modelo Cascata com *FeedBack*



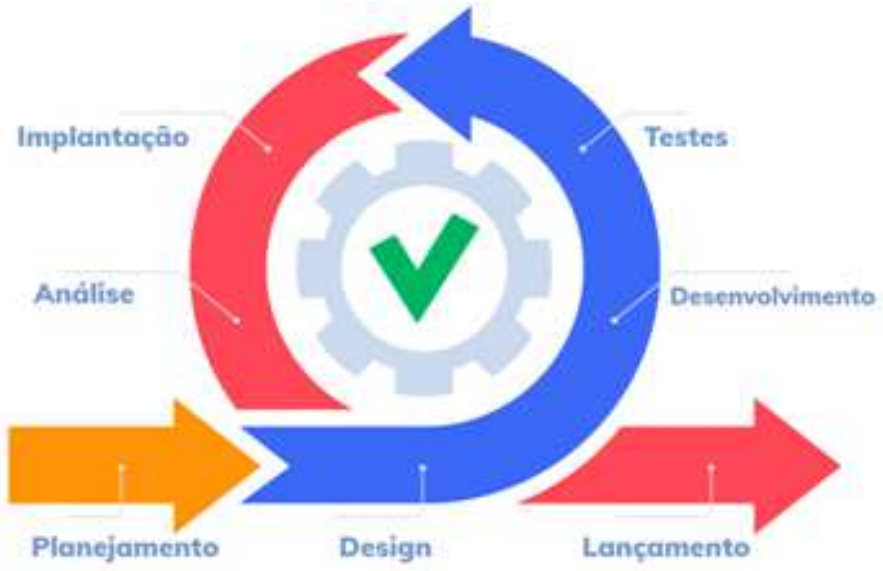
Teste SEMPRE foi entendido como importante

- Modelo em Espiral



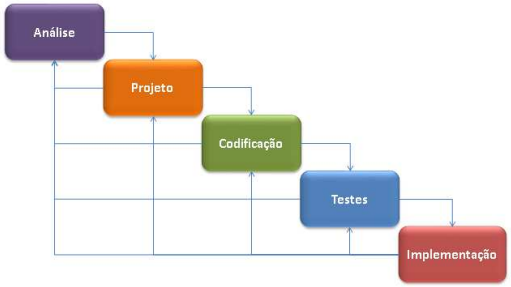
Teste SEMPRE foi entendido como importante

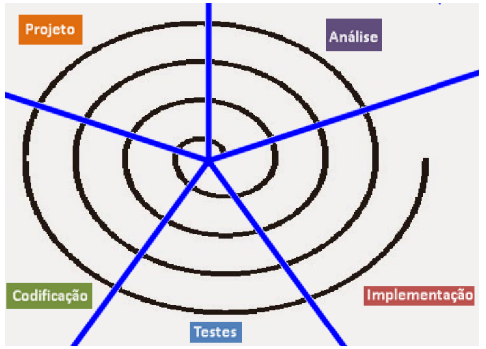
- Metodologias Ágeis (Scrum, XP)

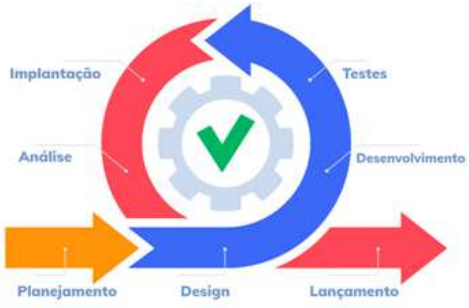


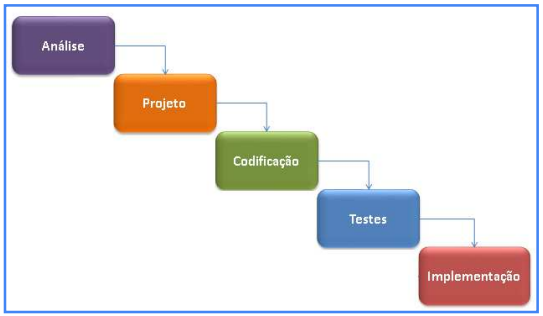
Teste SEMPRE foi entendido como importante

- Metodologias Ágeis (Scrum, XP)



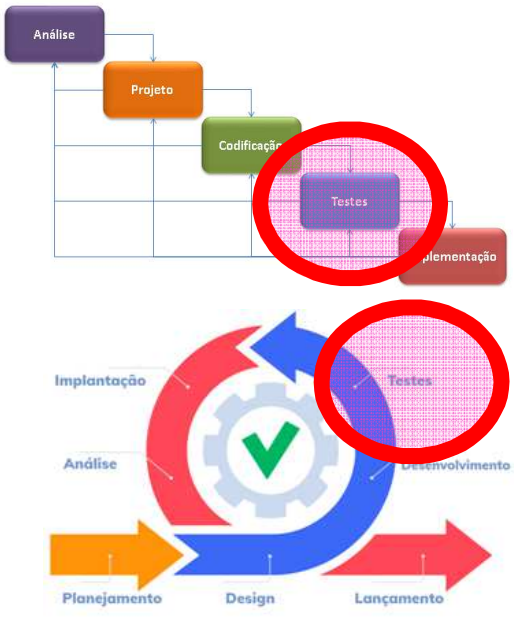


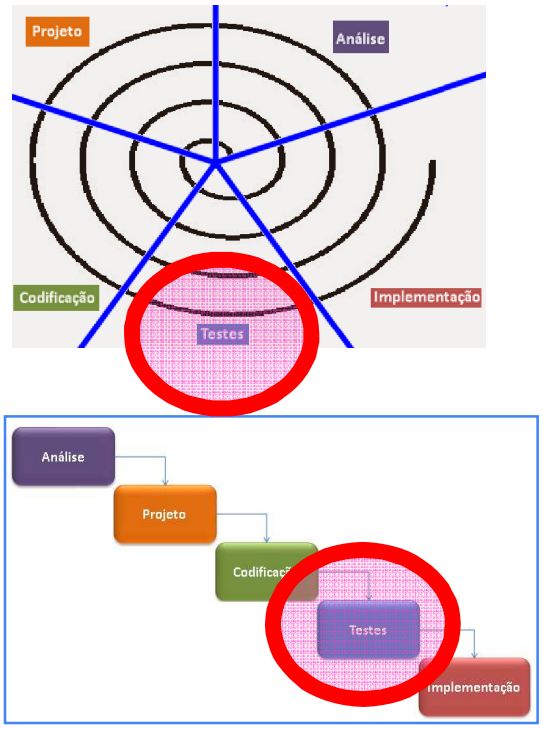




Teste SEMPRE foi entendido como importante

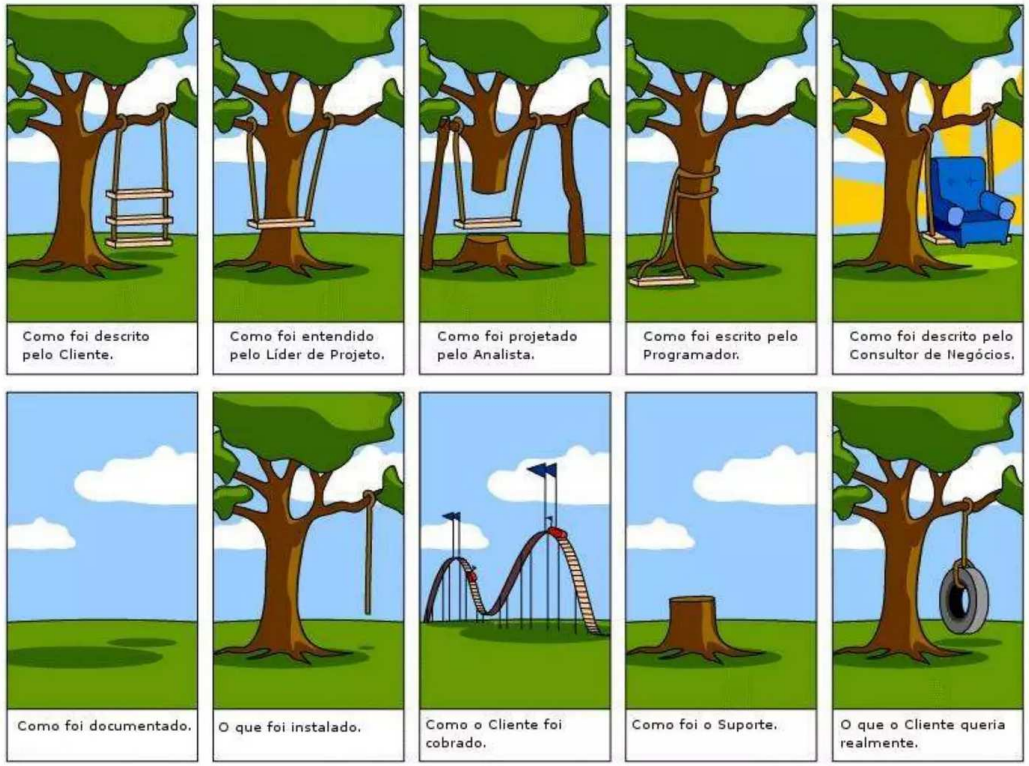
- Metodologias Ágeis (Scrum, XP)





Testes: melhorar a comunicação

- O problema da comunicação



Testes: definir o escopo

- Quando seu trabalho acaba?

Porque testes

- Entregar o que foi combinado
  - Estabelecer bem o que está sendo combinado
- Entregar algo que realmente funciona
  - Com alta probabilidade de sucesso
- Viabilizar a evolução do desenvolvimento
  - Detecção e correção precoce de falhas
  - Construção incremental de software com qualidade

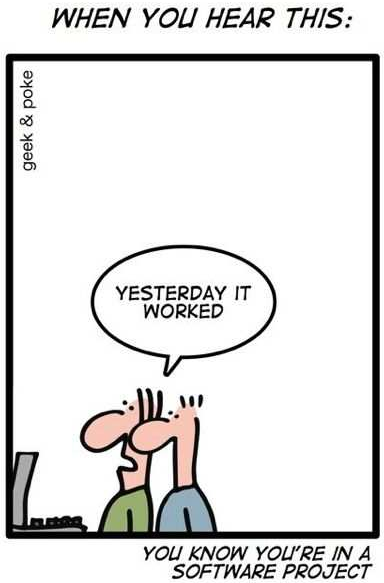
Tipos de Testes

- Testes de Unidade
  - Testar cada componente trivial de software para ver se ele realmente funciona
    - Tipicamente: métodos de classes
    - Às vezes: funções internas de classes
      - Protected
- Testes de Integração
  - Testar se a excução de processos que envolvem vários componentes funcionam
  - Importante, mas difícil de conceber e implementar
- Testes de Sistema (Funcionalidade)
  - Testar se os casos de uso estão adequadamente atendidos
  - Muito importante, muito mais difícil de conceber e implementar

Engenharia de Software x Engenharia Civil

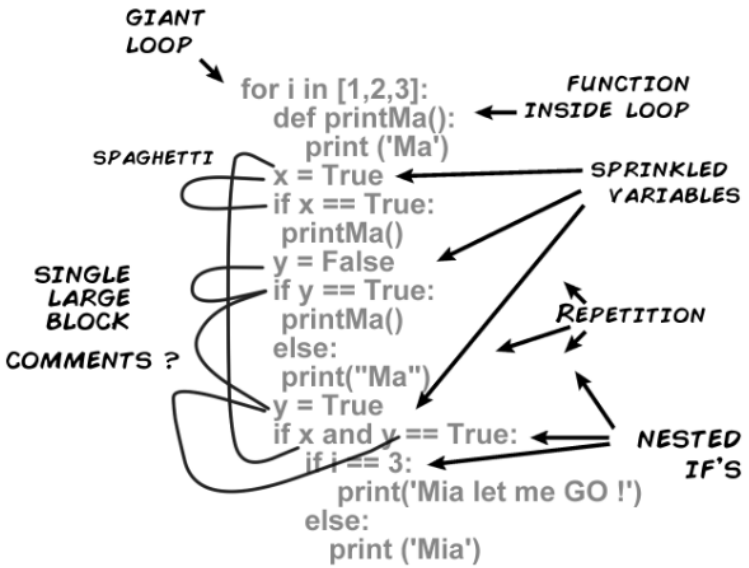
- Engenharia Civil

- Engenharia de Software



Testes de Software: apoio ao desenvolvimento

- Software spaghetti é o software muito confuso de entender
  - Fenômeno muito comum
  - Amenizado com boas práticas de programação
    - Padrões de projeto
- Bugs “Spaghetti”?
  - Um erro na classe XPTO causado pelo parâmetro K2 do método mimi() da instância construída com o construtor PTXK() da classe HJVK
- Quanto custa descobrir?



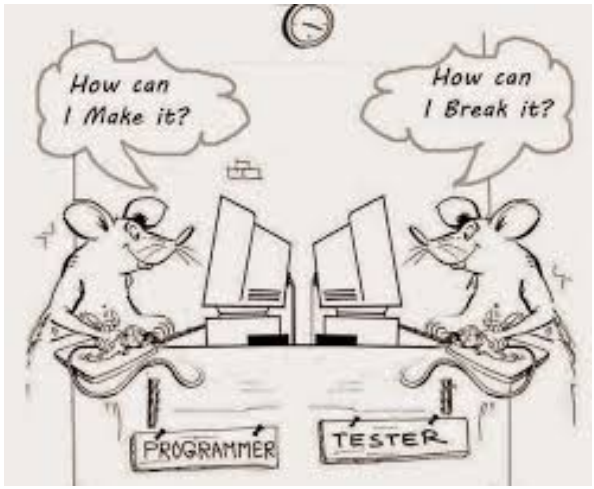
Como garantir que um artefato seja sólido?

- Engenharia civil: se o 1º andar está pronto, pode construir o 2º andar sobre ele
  - Assumindo especificações e construções padronizadas
- Engenharia de Software: se um pedaço do software está pronto, pode construir o pedaço seguinte usando ele
  - Mas... e se ele não está pronto?
  - Como eu sei se não está pronto?
  - Como dar ao artefato a confiança de que esteja de fato funcional?

Residência em Software

Como testar?

- Usando um testador
- Especialista em testes
- Idealmente NÃO PARTICIPOU DO DESENVOLVIMENTO
- Possui conhecimentos de programação
- Possui profundo conhecimento das funcionalidades



Como testar?

- Testes automáticos
- Programar o teste
  - Um (trecho de) programa para testar um (trecho de) programa
- Paradoxo?
- Sim! Um erro descoberto é do programa ou do teste?
  - Indecidível (a princípio)
- Premissa: o código do teste é extremamente simples
  - Tipicamente instanciar (ou não) objetos, executá-los e verificar se respondem corretamente às requisições que são feitas

Residência em Software

TDD – Test Driven Development