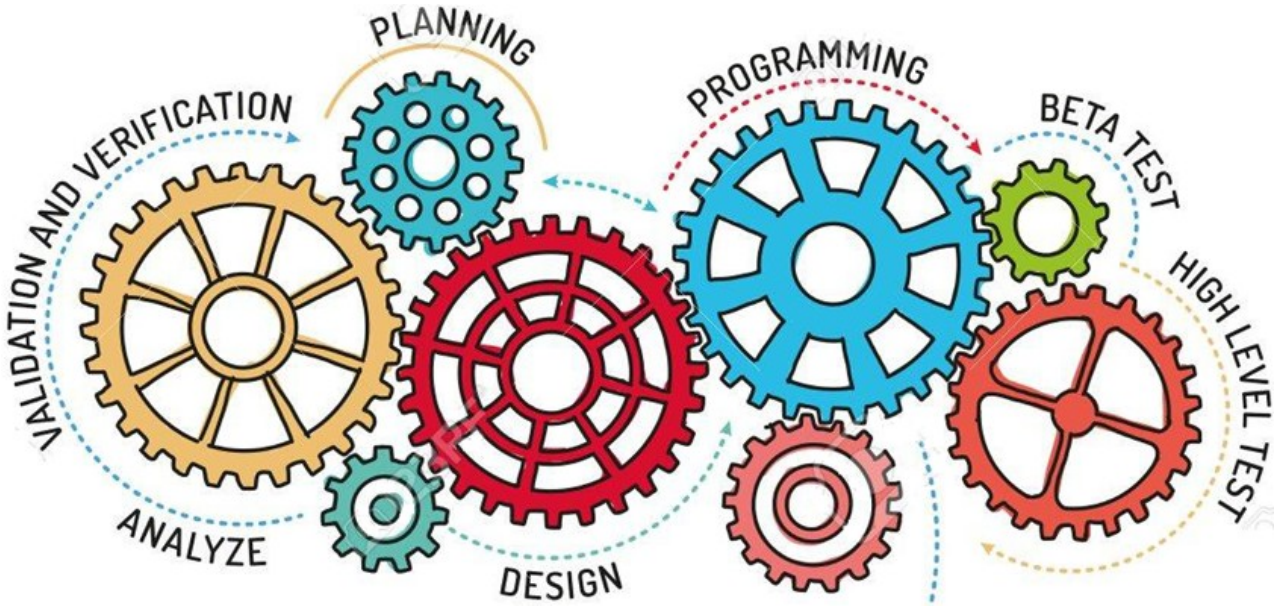
Engenharia de Software



Prof. Me. Clênio Silva
e-mail: clenio.silva@uniube.br

Uniube

Testes de software

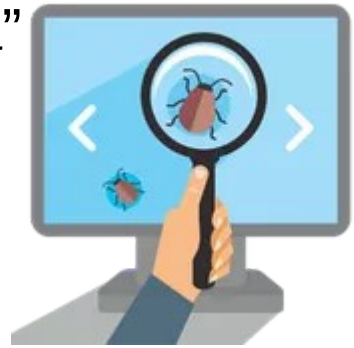
- O objetivo do teste é encontrar erros, e um bom teste é aquele que tem alta probabilidade de encontrar um erro;
- Portanto, um engenheiro de software deve projetar e implementar um sistema tendo em mente a "**testabilidade**";
- Os testes devem ter uma série de características que permitam atingir o objetivo de encontrar o maior número de erros com o mínimo de esforço.

Testes de software

- **Testabilidade**:
  - **James Bach** da a seguinte definição para testabilidade:
    - Testabilidade de software é simplesmente a facilidade com que um programa de computador pode ser testado".

  Além disso, as seguintes características levam a um software testável:

  - **Operabilidade:** "Quanto melhor funcionar, mais eficientemente pode ser testado"
  - **Observabilidade:** " O que você vê é o que testa"



Testes de software

- características levam a um software testável:
  - **Controlabilidade:** “Quanto melhor pudermos controlar o software, mais o teste pode ser automatizado e otimizado”.
  - **Decomponibilidade:** “Controlando o escopo de teste, podemos isolar problemas mais rapidamente e testar de forma mais racional”.
  - **Simplicidade:** “Quanto menos tivermos que testar, mais rapidamente podemos testá-lo”.
  - **Estabilidade:** “Quanto menos alterações, menos interrupções no teste”.

Testes de software

- características levam a um software testável:
  - **Compreensibilidade:** “Quanto mais informações tivermos, mais inteligente será o teste”.

Características do teste

- *Um bom teste tem alta probabilidade de encontrar um erro*.
  - Para atingir esse objetivo, o testador deve entender o software e tentar desenvolver uma imagem mental de como ele pode falhar.
- *Um bom teste não é redundante.*
  - O tempo e os recursos de teste são limitados. Não faz sentido realizar um teste que tenha a mesma finalidade de outro teste. Cada teste deve ter uma finalidade diferente .
- *Um bom deve teste dever o melhor da raça.*
  - Em grupo de testes com finalidades similares, as limitações de tempo e recursos podem induzir à execução de apenas um subconjunto dos testes que tenha a maior probabilidade de revelar uma classe inteira de erros.

Características do teste

- *Um bom teste não deve ser nem muito simples nem muito complexo*.
  - Embora algumas vezes seja possível combinar uma série de testes em um caso de teste, os possíveis efeitos colaterais associados a essa abordagem podem mascarar erros. Em geral, cada teste deve ser executado separadamente.

Testes Funcionais

- Testa os requisitos funcionais da aplicação e software. A ideia é verificar se a aplicação está apta a realizar as funções na qual foi desenvolvida para fazer.
- O teste funcional pode ser manual, realizado de forma automatizada ou uma mistura dos dois.
- Existem diversas maneiras de testar um software. A seguir será apresentado as principais técnicas informadas na literatura.

Técnicas de teste funcional

- **Teste de Caixa Branca:**
  - É a técnica que avalia o comportamento interno do software. Este tipo de técnica avalia o comportamento trabalhando diretamente sobre o código fonte do software para avaliar aspectos tais como: teste de condição, teste de fluxo de dados, teste de ciclos e testes de caminhos lógicos.
- **Teste de Caixa Preta:**
  - Esta técnica aborda o software a ser testado como uma caixa preta, ou seja, não se considera o comportamento interno do mesmo. Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido.
  - O componente de software a ser testado pode ser um método, uma função interna, um programa, um componente, um conjunto de programas ou uma funcionalidade.

Técnicas de teste funcional

- **Teste de Caixa Cinza:**
  - É uma abordagem híbrida das técnicas de testes de caixa branca e caixa preta, onde o testador possui algum conhecimento da estrutura/ código do sistema que esta sendo testado.
- **Teste de Unidade:**
  - É o teste básico que busca testar cada funcionalidade de software individualmente, fornecendo valores válidos ou inválidos e verificando se o retorno foi de acordo com o esperado. Ex. Teste de métodos de classes e/ou funções.

Técnicas de teste funcional

- **Teste de Integração:**
  - Envolvem testes de diferente módulos de um software. Uma aplicação é composta de diferentes submódulos que trabalham juntos para diferente funcionalidades. O objetivo dos testes de integração é validar a integração de diferentes módulos juntos e identificar os bugs e problemas relacionados a eles. Um exemplo seria validar módulos que acessam um banco de dados ou fazem uma chamada externa a outros sistemas.
- **Teste de Regressão:**
  - É uma técnica que consiste na aplicação de versões mais recentes do software, para garantir que não surgirão novos defeitos em componentes já analisados. Se adicionado novos módulos ou alterações, o teste é executado para validar todo o sistema e garantir que esta funcionado perfeitamente. Caso surjam novos defeitos em alguns módulos inalterados então se considera que o sistema regrediu.

Técnicas de teste não funcional

- **Teste de carga:**
  - Nesse tipo de teste o time de desenvolvimento ajuda a ter *insights* de como melhorar algum desempenho do software, com por exemplo o tempo de repostas de cada fluxo de uma aplicação. A principal ideia por trás do teste de carga é capturar e analisar as principais métricas de um software quando submetido a diferentes cargas de usuários.
- **Teste de capacidade:**
  - Identifica o limite do software. Com esse teste é possível metrificar a quantidade de usuários e operações que a ferramente suporta sem perder a qualidade e funcionalidade.

Técnicas de teste não funcional

- **Teste de stress:**
  - O teste de stress submete o software a uma carga excessiva de usuários e operações. A ideia é verificar o desempenho do software quando ele é submetido a cargas acima do limite estipulado. Além disso, identificar quais erros podem surgir caso essa situação ocorra de fato. Com isso é possível identificar quais partes do sistema apresentam mais falhas.

Referências

- PRESSMAN, Roger S. Engenharia de software: Uma Abordagem Profissional. 8.ed. Porto Alegre: AMGH, 2016. 968p.