**Fundamentos e níveis de teste**

Testes de software, segundo Neto (Dias Neto, 2010 *apud* Felix, 2016), são definidos como todo e qualquer procedimento que contribui na hora de determinar se um software cumpre com o objetivo de sua criação e justamente por ter características preventivas, Neto descreve que testes de software são de natureza destrutiva, jamais construtiva.

Em conformidade com Félix (2016), testar um software significa verificar se seu funcionamento é limpo, amigável e se ele se enquadra no ambiente que foi projetado, fazendo com que sejam baixas as probabilidades de que o usuário final seja surpreendido negativamente no futuro, para tal, os projetos são submetidos a testes como: invasão por vírus devido o mau uso do usuário final ou teste para performance do software e usabilidade.

A figura 1 mostra a relação entre níveis, tipos e técnicas de testes, e qual a ordem que cada categoria deve ser feita para que os testes se tornem menos complicados, pois, segundo Crespo et al. (2004), cada teste deve seguir uma ordem de execução já que deixar testes para depois pode acarretar em complicações devido a incapacidade de alguns testes de serem executados em certas etapas na fase do desenvolvimento do software.

Figura 1 – Relação entre níveis, tipo e técnicas de testes.

Diagrama

Descrição gerada automaticamente

Fonte: Crespo et al. (2004, p.4)

De acordo com Bourque e Fairlay(2004) a definição dos principais tipos e níveis de testes utilizados para garantir a qualidade dos softwares são:

* Teste de unidade:
  + Verifica o funcionamento isolado de elementos de software que são testáveis separadamente, podendo ser subprogramas individuais ou componentes maiores. Geralmente, ocorre com acesso ao código e suporte de ferramentas de depuração, frequentemente realizado pelos programadores que escreveram o código.
* Teste de integração:
  + Verifica as interações entre componentes de software, nele estratégias clássicas, como *top-down* e *bottom-up*, são frequentemente utilizadas. As estratégias modernas são geralmente impulsionadas pela arquitetura, integrando componentes ou subsistemas com base em funcionalidades identificadas, preferindo a integração incremental ao teste de ‘*big bang*'.
* Teste de sistema:
  + Se preocupa com o comportamento de um sistema completo. Testes eficazes de unidade e integração devem ter identificado muitos defeitos. Este nível é apropriado para avaliar requisitos não funcionais, como segurança, velocidade e confiabilidade, além de interfaces externas. Wazlawick (2011) afirma que testes de sistema é avaliado do ponto de vista do usuário e, também diz que, seu principal objetivo é verificar a funcionalidade geral do programa.
* Teste de Aceitação / Qualificação:
  + Esse teste determina se um sistema satisfaz os critérios de aceitação, verificando comportamentos desejados em relação aos requisitos do cliente. Pode ser realizado pelo cliente ou um representante, podendo ou não envolver os desenvolvedores.
* Teste de Instalação:
  + Após os testes de sistema e aceitação, o software é verificado ao ser instalado no ambiente alvo. O teste de instalação pode ser visto como um teste de sistema realizado em condições operacionais específicas.
* Teste de Regressão:
  + Teste de regressão é o reteste seletivo de um sistema ou componente para verificar se modificações não causaram efeitos indesejados e que o sistema ainda cumpre os requisitos especificados.
* Teste de Performance:
  + Verifica se o software atende aos requisitos de desempenho especificados, avaliando características como capacidade e tempo de resposta.
* Teste de Segurança:
  + Foca na verificação de que o software está protegido contra ataques externos, garantindo a confidencialidade, integridade e disponibilidade do sistema e seus dados.
* Teste de Stress:
  + Exercita o software sob a carga máxima designada e além dela, visando determinar os limites de comportamento e testar mecanismos de defesa.
* Teste Back-to-Back:
  + Envolve executar duas ou mais variantes de um programa com as mesmas entradas, comparando as saídas e analisando erros em caso de discrepâncias.
* Teste de Usabilidade e Interação Humano-Computador:
  + Avalia a facilidade com que os usuários finais aprendem e utilizam o software.

**Defeito, falha e erro**

Segundo Bourque e Fairlay (2004), a literatura de engenharia de software usa termos como falha, erro e defeito para descrever mal funcionamentos, o termo "fault" (falha) refere-se à causa do mal funcionamento, enquanto "failure" (erro) refere-se ao efeito indesejado observado e o termo mais genérico "defect" pode ser usado quando a distinção entre falha e erro não é crucial.

Figura x – Defeito *versus* erro *versus* falha

Diagrama

Descrição gerada automaticamente

Fonte: Félix (2016)

Na figura 1 é possível entender um pouco melhor o significado de cada termo, como descrito no *International Software Testing Qualifications Board* (ISTQB) – selo de qualidade para testadores de software--, os significados de erro, defeito e falha são:

* Erro:
  + Falha humana; produz resultado incorreto,
  + É, por exemplo, falha na escrita de um código;
* Defeito:
  + Resultado de um código mal escrito, ou seja, um erro.
  + Causa anomalia no funcionamento do sistema. O usuário final normalmente não vê o defeito propriamente dito.
  + Também é conhecido como *bug.*
* Falha:
  + Quando um código que apresenta defeito é executado, temos uma falha.
  + Funcionamento inesperado das funções do software.
  + É a camada que chega aos olhos do usuário.

Um exemplo para melhor entendimento dos termos citados anteriormente seria o seguinte: “Em um caixa eletrônico, você seleciona a tecla “saque”, o software reconhece a entrada e chama uma função, que deveria ser a entrega do valor solicitado. Essa função depende de um código. Uma vez que este código está mal escrito, a função executada não corresponde à que deveria, de fato, ser executada, apresentando, assim, um resultado inesperado, uma falha: a entrega de cheques, e não de dinheiro. (Felix, 2016)”.

No quadro 1 abaixo é possível visualizar os dados de forma mais organizada, separando pelo que seriam os erros, defeitos e falhas.

Quadro 1 – Aplicação dos conceitos

Tabela

Descrição gerada automaticamente

Fonte: Félix (2016).

CRESPO, A. N. et al. **Uma metodologia para teste de software no contexto da melhoria de processo**. Campinas, SP: CenPRA - Centro de Pesquisas Renato Archer, DMPS - Divisão de Melhoria de Processos de Software, 2004. Disponível em: <www.lbd.dcc. ufmg.br/colecoes/sbqs/2004/024.pdf>

DIAS NETO, A. C**. Introdução a testes de software.** Engenharia de Software *Magazine*, n. 1, 14 jun. 2010. Disponível em: <www. devmedia.com.br/artigo-engenharia-de-software-introducao-a- teste-de-software/8035>

FÉLIX, Rafael (org.). **Teste de software**. São Paulo: Pearson, 2016. *E-book*. Disponível em: https://plataforma.bvirtual.com.br.

BOURQUE, Pierre; FAIRLEY, RJNICS. Swebok. **Nd: IEEE Computer society**, 2004.

WAZLAWICK, R.S. **Análise e projeto de sistemas de informação orientados a objetos.** 2. ed. Rio de Janeiro: Campus, 2011.