VOLTAR

Testes de Software - Definições e Conceitos

Estudar as definições, fundamentos e princípios sobre testes de software.

NESTE TÓPICO

Introdução

O processo de testes consiste em executar o software de uma maneira controlada com o objetivo de avaliar se ele se comporta conforme o especificado. É uma das atividades que compõem a GQS (Garantia de Qualidade do Software).

A realização de testes, além de verificar o correto funcionamento do produto, tem como foco encontrar defeitos. Pressman (2011) afirma que “o objetivo do teste é encontrar erros, e um bom teste é aquele que tem alta probabilidade de encontrar um erro”.

Para Koscianski e Soares (2007), “o objetivo do teste é econtrar defeitos, revelando que o funcionamento do software em uma determinada situação não está de acordo com o esperado”. Podemos dizer que um teste bem sucedido é aquele que descobre os defeitos ainda não encontrados. Por mais que se deseje construir um produto de engenharia de maneira perfeita na primeira vez, as atividades de produção de software são conduzidas por humanos e estão suscetíveis a falhas. Por exemplo, problemas de comunicação, na maioria das vezes, estão propensas a gerar erros e defeitos em produtos de software.

Para muitos analistas e programadores, a realização dos testes de software é uma tarefa que, além de ser considerada maçante, é pouco compreendida. Com efeito, passar horas e horas procurando "defeitos" em algo que foi criado pela própria pessoa, ou equipe, remete o indivíduo a ter uma

desconfiança de sua própria capacidade. Esse motivo leva, muitas vezes, à recomendação de que outro analista, diferente daquele que desenvolveu o(s) artefato(s) a ser(em) testado(s), realize testes com o objetivo de encontrar erros. Em muitos casos, há uma equipe de testes, independente, que executa algumas das atividades de testes. Muitas vezes essas equipes são conhecidas como "fábricas de testes", sendo que há empresas especializadas nesse nicho de mercado.

Fundamentos do Teste de Software

De maneira genérica, podemos afirmar que o teste de software consiste na execução de um produto, com o intuito de determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. É necessário salientar que, o objetivo principal de testar o produto é, antes de qualquer coisa, validar se o que foi produzido atende às expectativas e necessidades dos usuários.

Como vimos, um bom teste é aquele que tem boas chances de encontrar os erros no componente que está sendo averiguado. Para que o analista exerça esse fundamento, deve desenvolver um pensamento do tipo: "como esse software pode falhar".

Outro fundamento é que os projetos de software têm recursos limitados, por isso devem-se planejar os testes de modo que não haja redundância. Cada teste deve ter finalidade diferente, mesmo que essa diferença seja sutil. Pressman (2011) salienta que "em um grupo de testes com finalidades similares, as limitações de tempo e recursos podem induzir à execução de apenas um subconjunto desses testes". Para esses casos, continua o autor, "deverá ser usado o teste que tenha a maior probabilidade de revelar uma classe inteira de erros".

Por outro lado, combinar uma série de testes em um único caso, pode deixá-lo complexo demais, gerando o risco de que erros sejam mascarados. Nessa situação, é importante segregar a execução do teste para torná-lo mais compreensível. Em outras palavras, "um bom teste não deve ser nem muito simples nem muito complexo".

Conceitos do Testes de Software

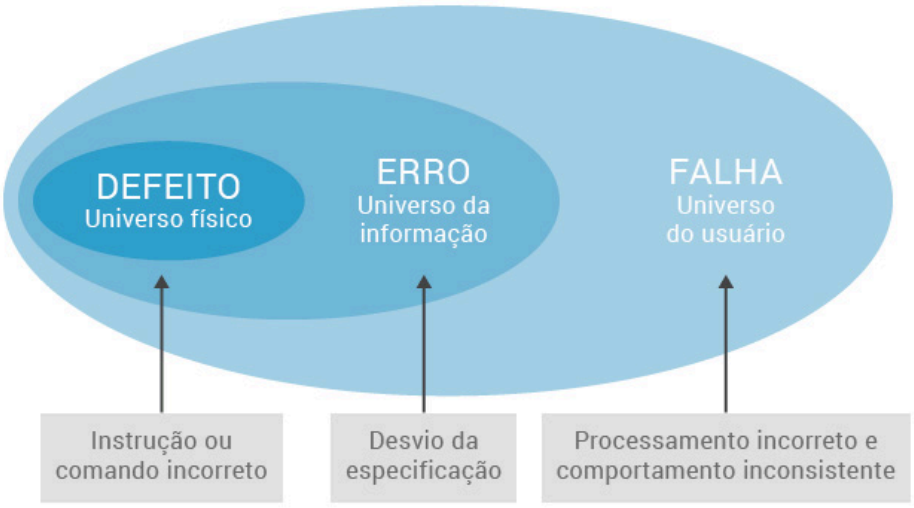
Primeiramente vamos compreender alguns importantes conceitos sobre testes, distinguindo o que é um defeito, um erro e uma falha no software. Segundo a IEEE Standards Association (1990), podemos definir esses conceitos como:

- **Defeito** é um ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta. Por exemplo, uma *instrução ou comando incorreto*.
- **Erro** é uma manifestação concreta de um defeito num artefato de software. *Diferença entre o valor obtido e o valor esperado*, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa constitui um erro.
- **Falha** é o comportamento operacional do software diferente do esperado pelo usuário.

AS FALHAS E OS ERROS

Uma falha pode ter sido causada por diversos erros e alguns erros podem nunca causar uma falha.

A figura a seguir ilustra as diferenças entre esses conceitos:



Defeito, erro e falha.

Para realizar as atividades de testes, a engenharia de software possui alguns elementos básicos. Os autores Craig e Jaskiel (2002) conceitua três deles:

- **Caso de Teste**: descreve uma condição particular a ser testada e é composto por valores de entrada, restrições para a sua execução e um resultado ou comportamento esperado.
- **Procedimento de Teste**: é uma descrição dos passos necessários para executar um caso (ou um grupo de casos) de teste.
- **Critério de Teste**: serve para selecionar e avaliar casos de teste de forma a aumentar as possibilidades de provocar falhas ou, quando isso não ocorre, estabelecer um nível elevado de confiança na correção do produto.

O Processo de Testes de Software

Os preparativos para a realização de testes de software devem considerar alguns fatores:

- Erros nem sempre são óbvios.
- Erros diferentes podem ter a mesma origem e manifestação.

- Erros podem ser propagados entre processos, ou seja, deve existir uma rastreabilidade do erro e da sua origem.
- Testar um processo não indica que, ao ser testado em conjunto com outros processos, ele se comporte de maneira semelhante.
- Saber que um processo não está correto não é, necessariamente, saber como o erro deve ser corrigido.

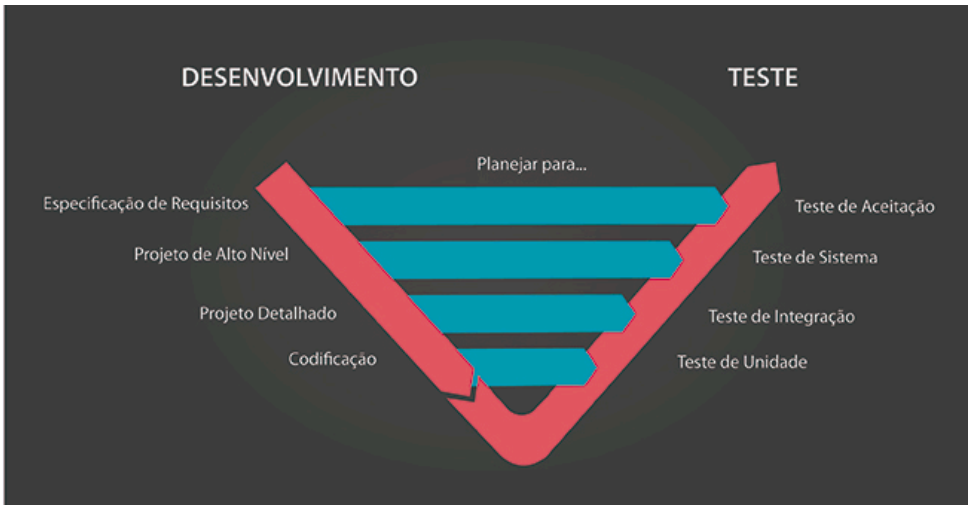
Dessa forma, não basta efetuar somente testes unitários (programa a programa, método a método, etc.) ou somente o teste de sistema (testar os programas em conjunto), mas gerar testes para cada fase do desenvolvimento que ampliem cada vez mais a abrangência dos processos.

Estratégias de Teste de Software

As estratégias de teste são divididas nos seguintes estágios:

1. Teste de Unidade
2. Teste de Integração
3. Teste do Sistema
4. Teste de Aceitação

Conforme a figura abaixo, os planejamentos dos testes ocorrem desde as fases iniciais do projeto de software. As especificações de requisitos são base para o planejamento dos testes de aceitação, o teste de sistema é planejado a partir do projeto de alto nível, o projeto detalhado é fonte para os planos de teste de integração e, por fim, a codificação subsidiará os planos de teste unitários.



O paralelismo entre as atividades de desenvolvimento e teste de software.

Vamos estudar agora os quatro estágios que compõem as estratégias de testes.

Teste de Unidade (Componente):

O objetivo é testar os componentes individualmente. Utilizado para garantir que um programa, método ou função atenda às especificações e produza os resultados esperados.

Nesse estágio, são elaborados os caminhos para descobrir erros nos limites dos módulos.

Emprega-se foco na lógica interna e estrutura de dados do componente, onde cada componente tem suas interfaces testadas para garantir que as informações fluam corretamente para dentro e para fora da unidade.

E por fim, assegura que todas as instruções de um módulo tenham sido executadas pelo menos uma vez.

Teste de Integração:

Utilizado para verificar se os requisitos explícitos e implícitos do sistema estão sendo atendidos. Deve servir, também, para analisar a estrutura hierárquica de execução dos módulos.

Permite que os componentes que já passaram pelo teste da unidade sejam testados de forma integrada, verificando assim se suas interfaces estão funcionando como deveriam.

Uma boa prática é a aplicação de uma abordagem incremental de testes para a construção da arquitetura do sistema. O programa é construído e testado em pequenos incrementos, onde os erros são mais fáceis de serem isolados e corrigidos, na contramão de se construir e testar unitariamente todos os componentes e só integrá-los no final, de uma vez só. Essa última opção normalmente traz um grande volume de erros difíceis de serem administrados e resolvidos.

Durante essa fase, a equipe de testes tem acesso ao código-fonte do sistema. Quando um problema é encontrado, o time de integração investiga para determinar a causa-raiz e identificar os componentes que estão causando os defeitos no sistema para resolvê-los.

Estratégias de Teste de Integração:

- Integração Descendente (top-down): início dos testes no programa principal e depois nos módulos subordinados;
- Integração Ascendente (bottom-up): inicio dos testes nos módulos atômicos (componentes nos níveis mais baixos do programa);
- Teste de Regressão: tem o objetivo de garantir que alterações efetuadas no sistema não estejam gerando erros. Para isso, executam-se novamente os testes dos incrementos já testados anteriormente para assegurar que as alterações não causaram problemas;
- Teste Fumaça: estratégia normalmente utilizada para projetos com prazo crítico. São testes abrangentes, de ponta a ponta, com o objetivo de verificar se passa sem erros para, só depois, realizar os testes rigorosos.

Teste de Sistema

Aqui a equipe de testes testa uma versão a ser liberada ao usuário. O foco nessa etapa é validar se o sistema está atendendo aos requisitos e se é confiável. Geralmente é um teste tipo “caixa preta”, onde a equipe permanece concentrada em confirmar o correto funcionamento do sistema.

> Teste de sistema é na realidade uma série de diferentes testes cuja finalidade primária é exercitar totalmente o sistema. Embora cada um dos testes tenha uma finalidade diferente, todos funcionam no sentido de verificar se os elementos do sistema foram integrados adequadamente e executam as funções a eles alocadas.
>
> ((PRESSMAN, 2011))

Tipos de Testes de Sistemas:

- **Teste de Recuperação**: Recuperação de falhas e retorno ao processamento em pouco ou nenhum tempo de parada. O teste força o software a falhar de várias formas e verifica se a recuperação é executada corretamente.
- **Teste de Segurança**: Verifica se os mecanismos de proteção de fato protegem contra acesso indevido. Com tempo e recursos, um bom teste de segurança conseguirá invadir o sistema. O objetivo é tornar o custo da invasão maior do que o valor das informações a serem obtidas.
- **Teste por Esforço** (*estresse*): O sistema é forçado a operar em limites fora do projeto do software. Colocam os programas em situações anormais, ou seja, demanda recursos em quantidade, frequência ou volumes anormais.
- **Teste de Desempenho**: Testa o desempenho em tempo de execução. O objetivo é de descobrir situações que levam à degradação e possível falha do sistema.
- **Teste de Disponibilização** (ou teste de configuração): Exercita o software em cada ambiente que ele deva operar e examina todos os procedimentos de instalação.

Teste de Aceitação

Nessa etapa, há a participação de usuários e, se a confirmação (aceitação) do sistema for feita, o mesmo pode ser liberado para uso. Caso contrário, os erros são relatados à equipe de desenvolvimento que deverá prosseguir com as devidas correções.

Utilizado para garantir que o cliente vistoriou o produto entregue e está de acordo com os requisitos.

Tipos de Testes de Aceitação:

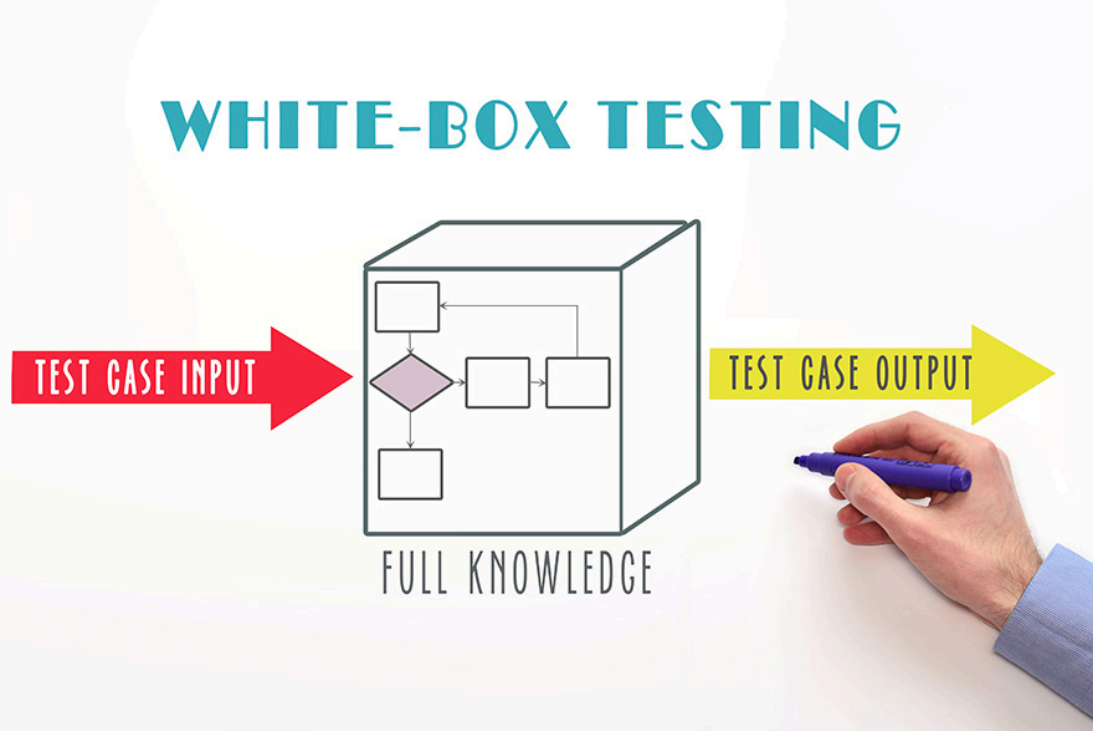
- **Teste Alfa**: conduzido em um ambiente controlado (ambiente do desenvolvedor), com a presença do desenvolvedor.

- **Teste Beta**: conduzido nas instalações do usuário (ambiente do cliente), sem a presença do desenvolvedor (o usuário registra os problemas e encaminha ao desenvolvedor).

Principais Abordagens de Testes

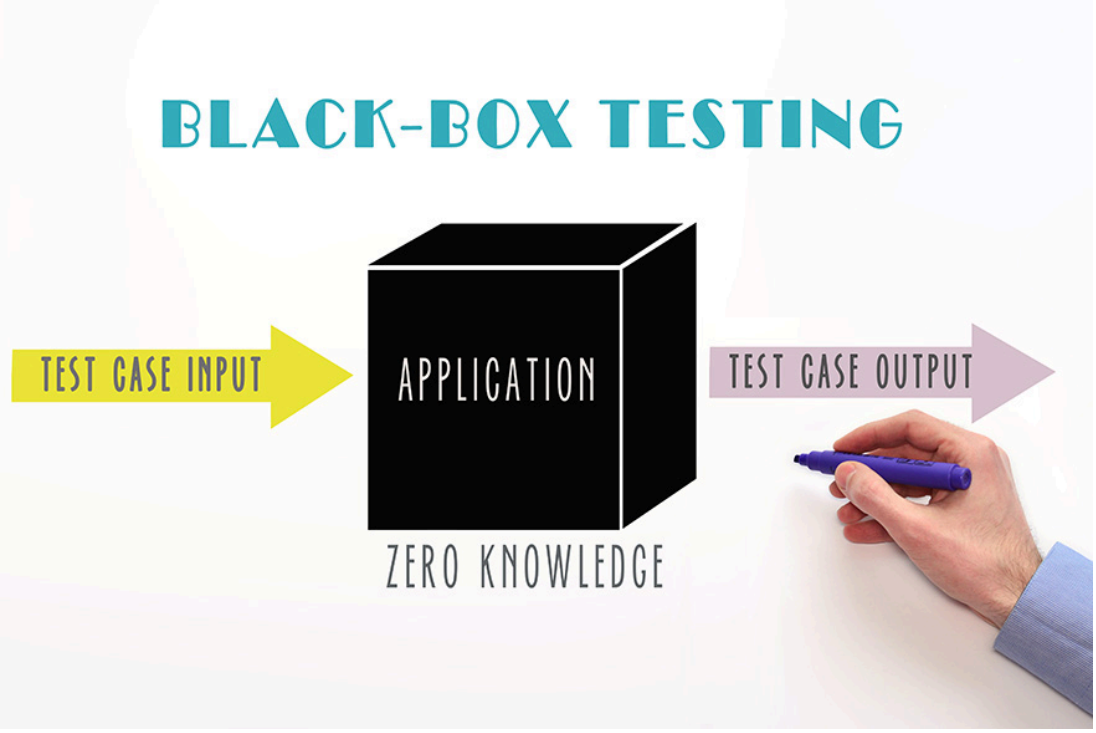
As abordagens para testes de software estão divididas em duas visões:

1) Visão Interna: Há o conhecimento dos componentes internos do produto. Nessa visão é utilizada a abordagem conhecida como Teste Caixa-Branca (também conhecida como teste caixa-de-vidro);



Teste Caixa-Branca

2) Visão Externa: Há o conhecimento das funções que o sistema deve desempenhar, mas não há conhecimento do funcionamento interno do produto. Essa abordagem é conhecida como Teste Caixa-Preta.



Teste Caixa-Preta

Teste Caixa-Branca (ou estrutural)

Utilizado para garantir a fidelidade dos conjuntos de instruções e da estrutura interna dos dados. Os testes são gerados a partir de uma análise dos caminhos lógicos possíveis de serem executados. É necessário conhecimento do funcionamento interno dos componentes do software.

Objetivos:

- Garantir que todos os caminhos independentes de um módulo sejam executados pelo menos uma vez;
- Realizar todas as decisões lógicas para valores falsos e verdadeiros;
- Executar todos os ciclos em seus limites e dentro de suas fronteiras operacionais;
- Avaliar as estruturas de dados internas.

Principais técnicas:

- Testes de caminhos;
- Testes de estruturas de controle (laços, estruturas condicionais, etc.);
- Complexidade ciclomática (métrica).

Teste Caixa-Preta (ou funcional)

Utilizado para verificar se as funções estão sendo executadas de forma correta, ou seja, se os resultados obtidos são aqueles esperados. Tem foco nos requisitos funcionais do software. Permitem derivar séries de condições de entrada que utilizarão completamente todos os requisitos.

Objetivos:

- Encontrar funções incorretas ou ausentes;
- Encontrar erros de interface;
- Encontrar erros em estrutura de dados ou acesso a base de dados externas;
- Encontrar erros de comportamento;
- Encontrar erros de inicialização e término.

Os testes são feitos para responder às seguintes questões:

- Como a validade funcional é testada?
- Como o comportamento e o desempenho do sistema são testados?
- Que classes de entrada farão bons casos de teste?
- O sistema é particularmente sensível a certos valores de entrada?
- Como as fronteiras de uma classe de dados são isoladas?
- Que taxas e volume de dados o sistema pode tolerar?
- Que efeito combinações específicas de dados terão sobre a operação do sistema?

Princípios do Teste de Software

A Comissão Internacional para Qualificação de Teste de Software (ISTQB – sigla em inglês para International Software Testing Qualifications Board) traz na sua publicação sobre Fundamento de Software, sete princípios os quais oferecem um guia geral para o processo de teste como um todo (ISTQB, 2011):

Princípio 1 – Teste demonstra a presença de defeitos

O teste pode demonstrar a presença de defeitos, mas não pode provar que eles não existem. O Teste reduz a probabilidade que os defeitos permaneçam em um software, mas mesmo se nenhum defeito for encontrado, não prova que ele esteja perfeito.

Princípio 2 – Teste exaustivo é impossível

Testar tudo (todas as combinações de entradas e pré-condições) não é viável, exceto para casos triviais. Em vez do teste exaustivo, riscos e prioridades são levados em consideração para dar foco aos esforços de teste.

Princípio 3 – Teste antecipado

A atividade de teste deve começar o mais breve possível no ciclo de desenvolvimento do software ou sistema e deve ser focado em objetivos definidos.

Princípio 4 – Agrupamento de defeitos

Um número pequeno de módulos contém a maioria dos defeitos descobertos durante o teste antes de sua entrega ou exibe a maioria das falhas operacionais.

Princípio 5 – Paradoxo do Pesticida

Pode ocorrer de um mesmo conjunto de testes que são repetidos várias vezes não encontrarem novos defeitos após um determinado momento. Para superar este "paradoxo do pesticida", os casos de testes necessitam ser frequentemente revisado e atualizado. Um conjunto de testes novo e diferente precisa ser escrito para exercitar diferentes partes do software ou sistema com objetivo de aumentar a possibilidade de encontrar mais erros.

Princípio 6 – Teste depende do contexto

Testes são realizados de forma diferente conforme o contexto. Por exemplo, softwares de segurança crítica são testados diferentemente de um software de comércio eletrônico.

Princípio 7 – A ilusão da ausência de erros

Encontrar e consertar defeitos não ajuda se o sistema construído não atende às expectativas e necessidades dos usuários.

Quiz

Exercício Final

Testes de Software - Definições e Conceitos

INICIAR

Referências

CRAIG, R. D.; JASKIEL, S. P. *Systematic Software Testing*. Boston: Artech House Publishers, 2002.

IEEE STANDARDS ASSOCIATION. *Standard Glossary of SoftwareEngineering Terminology*. [S.l.]: Std 610.12-1990, 1990.

ISTQB - INTERNATIONAL SOFTWARE TESTING QUALIFICATIONS BOARD. *Certified Tester Foundation Level Syllabus*. [S.l.]: ISTQB, 2011.

KOSCIANSKI, A.; SOARES, M. D. S. *Qualidade de software*: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2. ed. São Paulo: Novatec, 2007.

PRESSMAN, R. S. *Engenharia de Software*: Uma Abordagem Profissional. 7. ed. Porto Alegre: McGraw Hill, 2011.

SOMMERVILLE, I. *Engenharia de Software*. 8. ed. São Paulo: Pearson Addison-Wesley, 2007.

Avalie este tópico

ANTERIOR
Melhoria de Processo de Software Brasileiro - Modelo MPS.BR

Índice

PRÓXIMO
Testes de Software - Verificação e Validação