

□ Unidade 11 - Introdução aos testes de software

Apresentação

No desenvolvimento de *software*, a última etapa é a que contempla os testes e ela é de suma importância. Por meio dessa etapa ocorre a detecção e a solução de erros no *software*.

Dentre as fases dos testes são executadas várias atividades que têm a finalidade de buscar no *software* qualquer tipo de erro ou falha na codificação que possa interromper ou fazer com que seu funcionamento seja prejudicado.

Nesta Unidade de Aprendizagem, você estudará o conceito, as causas e os defeitos em *software*, assim como o papel do analista de testes e saberá verificar e como obter a qualidade em um sistema desenvolvido.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Apresentar as definições de teste de *software*.
- Apontar as causas e os defeitos em *software* e o papel do analista de teste de *software*.
- Explicar o que é qualidade de *software* e o papel do teste de *software* na obtenção da qualidade.

Infográfico

O teste de *software* tem como finalidade verificar se o produto alcançou as especificações de funcionamento de acordo com o que foi projetado.

No Infográfico a seguir, você conhecerá tipos de teste de *software* e o que cada um deles verifica.

TIPOS DE TESTE DE SOFTWARE

Dentre os tipos de teste de software, pode-se citar:

- ▶ Teste de instalação e configuração
- ▶ Teste de segurança
- ▶ Teste de integridade
- ▶ Teste funcional
- ▶ Teste de unidade
- ▶ Teste de integração
- ▶ Teste de volume
- ▶ Teste de performance
- ▶ Teste de usabilidade
- ▶ Teste de regressão



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.



TESTE DE INSTALAÇÃO E CONFIGURAÇÃO

Essas verificações são baseadas no comportamento do software para diferentes configurações de sistema e de hardware, ou seja, nesses testes é avaliado o correto funcionamento para as configurações de softwares e hardware especificadas.

TESTE DE SEGURANÇA

Este teste tem como premissa identificar falhas de segurança atreladas a um software ou ambiente no qual será executado. Dessa forma, é possível verificar se há vulnerabilidades referentes a diferentes tipos de ataques e prever melhorias ou correções, como, por exemplo, verificar se é possível acessar a área administrativa sem que haja uma solicitação de senha ou, ainda, se é possível inserir códigos maliciosos nos arquivos.

TESTE DE INTEGRIDADE

Verifica se, mesmo a partir da utilização de um grande volume de dados, os componentes do software permanecerão íntegros, como, por exemplo, fazer a verificação de como se comporta uma tabela de dados com milhões de registros.

TESTE FUNCIONAL

Este teste tem como premissa verificar se os requisitos funcionais da aplicação estão conforme o solicitado. Dessa forma, as funções serão executadas corretamente, como, por exemplo, fazer teste de condição (se for digitado X, o resultado será o esperado?).



TESTE DE UNIDADE

Este teste tem como finalidade verificar se o retorno está de acordo com o esperado após a inserção de valores válidos ou não, como, por exemplo, verificar se um campo de um formulário, que deveria receber o número de um telefone, está aceitando letras.

TESTE DE INTEGRAÇÃO

Neste tipo de teste, módulos são integrados e testados em grupo, como, por exemplo, verificar se o sistema está acessando o banco de dados ou chamando externamente a outros softwares.

TESTE DE VOLUME

Este teste tem como premissa submeter uma quantidade de dados considerável ao sistema e verificar como será o comportamento, como, por exemplo, criar uma consulta que apresente todo o conteúdo do banco de dados.

TESTE DE PERFORMANCE

Este teste tem como finalidade avaliar a capacidade de retorno, robustez, confiabilidade, escalabilidade e disponibilidade do software de acordo com a quantidade de conexões realizadas ao mesmo tempo, verificando o desempenho, como, por exemplo, verificar se o software não apresenta problemas de processamento ou espaço em disco.



TESTE DE USABILIDADE

Este teste tem como premissa verificar a facilidade que o usuário terá em compreender e manipular o software. Tem foco em melhorar experiências do usuário, como, por exemplo, verificar se o site tem fundo ruim (seja cor ou imagem), fontes de difícil leitura ou tamanho de imagens inadequado.

TESTE DE REGRESSÃO

Este teste tem como finalidade garantir que não surgirão defeitos em componentes já analisados, como, por exemplo, aplicar versões mais recentes do software e verificar se surgirão defeitos em componentes já avaliados em versões anteriores.

Conteúdo do Livro

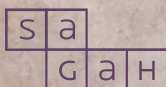
A qualidade de *software* apresenta uma tendência que cresce na indústria de software. E, para que a qualidade seja uma realidade, é necessário que os testes de *software* e os profissionais dessa área sejam cada vez mais qualificados.

No capítulo Introdução aos testes de *software*, da obra Testes de Software e Gerência de Configuração, você verá os conceitos, as causas, os defeitos de software e o papel do analista de teste de *software* para a obtenção da qualidade de *software*.

Boa leitura.

TESTES DE SOFTWARE E GERÊNCIA DE CONFIGURAÇÃO

Priscila Gonçalves



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS



Introdução aos testes de software

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Definir teste de software.
- Apontar as causas de defeitos em software e o papel do analista de teste de software.
- Explicar a qualidade de software e o papel do teste de software na obtenção dela.

Introdução

O teste de software mostra as falhas do sistema antes que o desenvolvimento seja concluído. A partir desse teste, é possível assegurar que as funcionalidades solicitadas estejam presentes e de acordo com o esperado.

Quando as falhas são corrigidas nas fases finais do projeto, o custo é maior do que se fossem solucionadas no início. Por isso, muitas empresas, profissionais e equipes preferem fazer um desenvolvimento orientado aos testes.

Neste capítulo, você vai conhecer a definição de teste de software. Você também vai ver quais são as causas de defeitos em softwares e qual é o papel do analista de teste. Além disso, você vai aprender mais sobre a qualidade dos softwares e o papel do teste na obtenção dela.

Teste de software

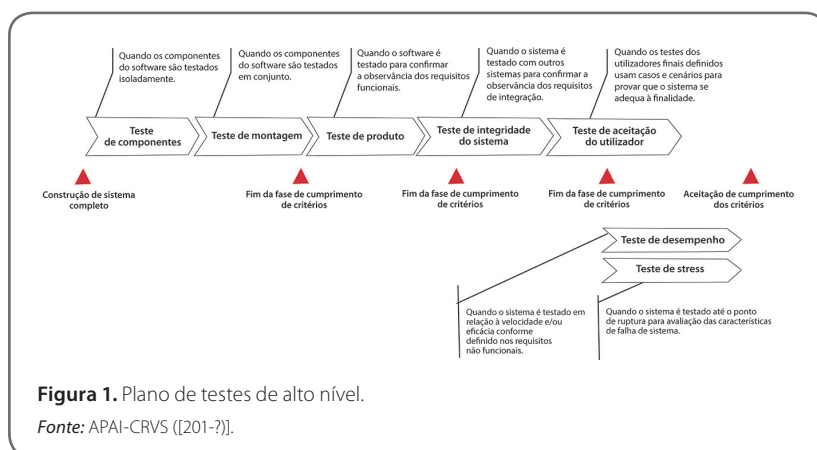
Cada vez mais pessoas e organizações utilizam softwares para automatizar e facilitar trabalhos no dia a dia. Por isso, é necessário que o funcionamento dos softwares seja correto, ou seja, que eles executem as suas funções sem produzir erros.

A verificação dos softwares ocorre por meio de testes realizados antes da entrega do produto aos consumidores. O teste é a última etapa do desenvolvimento de um software. Como você deve imaginar, é uma fase de suma importância. Por meio dela, é possível verificar e resolver problemas e *bugs* — caso ambos sejam encontrados. O teste de software tem como objetivo encontrar erros e produzir softwares com a maior qualidade possível, tornando-os seguros e confiáveis para utilização.

A qualidade de software, segundo Pressman e Maxim (2016), deve ser implementada por meio de atividades que se concentram na gestão de qualidade, tais como padrões IEEE e ISO, revisões e auditorias, testes, coleta e análise de erros e defeitos, gerenciamento de mudanças, educação, gerência de fornecedores, administração de segurança, proteção e gestão de riscos. Afinal, um programa de computador é um produto que utiliza diversas linguagens de programação, tendo disponível uma rigorosa abordagem para a especificação dos requisitos, a análise, o projeto, o desenvolvimento e os testes de software (PRESSMAN; MAXIM, 2016).

Para a realização dos testes, utiliza-se um plano de teste, que é um documento de planejamento do projeto de teste. O plano de teste deve conter todas as etapas de validação e verificação do software a serem observadas. A validação compreende o processo de examinar se o software satisfaz ou não a necessidade do usuário. Valida-se o software que é compatível com os requisitos solicitados. Por sua vez, a verificação de software é o processo que confirma que o programa satisfaz todos os requisitos.

Na Figura 1, a seguir, veja que tipos de teste devem fazer parte do plano de teste.



Os módulos de software são testados de forma individual para que os erros sejam identificados e eliminados. Caso o produto seja um software, a questão dos testes torna-se muito mais complexa. Os requisitos de software podem variar entre os diferentes ambientes de software básico nos quais os programas vão ser instalados, pois pode haver incompatibilidade. Por exemplo, um software que foi desenvolvido para funcionar em um sistema operacional do Windows pode não ter o mesmo funcionamento em um sistema operacional da Apple. O teste de configuração ou instalação é responsável por emitir o resultado pertinente a essa questão.

Os testes são classificados de diferentes formas. Veja a seguir.

- **Teste de componentes:** componentes do software são testados isoladamente.
- **Teste de montagem:** componentes do software são testados em conjunto.
- **Teste de produto:** o software é testado para confirmar que os requisitos funcionais estão presentes.
- **Teste de integridade de sistema:** testa a robustez do software, ou seja, a resistência a falhas.
- **Teste de aceitação do utilizador:** usuários finais utilizam casos e cenários para provar que o sistema se adequa à sua finalidade.
- **Teste de desempenho:** o sistema é testado em relação à velocidade ou à eficácia, conforme definição nos requisitos não funcionais.
- **Teste de performance:** avalia a capacidade de resposta, a disponibilidade, a confiabilidade e a robustez do software diante de determinada carga de trabalho, em condições específicas e por determinado tempo. O objetivo é verificar comportamentos diferentes que condições diversas podem gerar.
- **Teste de estresse:** o sistema é testado até o ponto de ruptura para avaliar características de falhas.
- **Teste de integração:** verifica se um ou mais componentes combinados funcionam de maneira satisfatória.
- **Teste de usabilidade:** é realizado com foco na experiência do usuário, analisando a consistência da interface, o *layout*, o acesso a funcionalidades, a facilidade de utilização e a viabilidade da manipulação do sistema pelo usuário.
- **Teste de configuração ou instalação:** verifica como o software se comporta ao ser instalado em diferentes configurações de software e hardware.

- **Teste de segurança:** verifica se o sistema e os dados são acessados de forma segura somente por quem executa as ações.
- **Teste funcional:** verifica os requisitos funcionais, as funções e os casos de uso, ou seja, analisa se a aplicação faz o que deveria fazer.
- **Teste de unidade:** testa um componente de forma isolada.
- **Teste de volume:** verifica o comportamento do sistema funcionando com o volume “normal” de dados e transações, envolvendo o banco de dados durante um longo período.

Geralmente, os testes são realizados por um grupo de profissionais. Entre os testes realizados, estão o teste de caixa-branca, que verifica se o código-fonte foi implementado corretamente, e o teste de caixa-preta, que verifica se as principais funções do sistema estão de acordo com os requisitos solicitados pelo cliente. Há também os testes de caixa-cinza, que são testes funcionais ou estruturais, também chamados de testes de regressão. À medida que são liberadas novas versões, novos *bugs* podem ser incluídos.

O teste de caixa-branca utiliza a perspectiva interna do sistema para realizar a modelagem dos casos de teste; no software, isso se refere ao código-fonte; no hardware, a cada nó do circuito. No teste da caixa-preta, essa perspectiva interna é desconhecida e são testadas e mensuradas somente interfaces do software. Porém, as duas técnicas podem ser utilizadas em conjunto, dando origem à técnica da caixa-cinza, em que é feita a modelagem de teste conhecendo-se a estrutura interna do sistema; porém, a execução ignora esse aspecto, assim como ocorre no teste de caixa-preta.



Exemplo

Um erro simples pode ocasionar danos graves, inclusive à vida. O Patriot era um software de balística e orientação de mísseis utilizado na Guerra do Golfo. No momento do seu uso, ocorreu um erro de arredondamento. Assim, o software calculou incorretamente o tempo, ignorando os mísseis Scud vindos do Iraque. O erro ocasionou a morte de 28 soldados e deixou outros 100 feridos.

Uma questão importante que você deve considerar em testes de softwares é o custo da verificação dos softwares. Além disso, existem ferramentas de suporte. Veja os exemplos a seguir.

- **qTest:** ferramenta de teste de desempenho.
- **Testlink:** ferramenta *open source* de gerenciamento de testes de software que possibilita que equipes de teste trabalhem de forma sincronizada.
- **Loadrunner:** ferramenta de teste de software da Micro Focus utilizada para testar aplicativos, medir o comportamento do sistema e o desempenho sob a carga.

Como você pode notar, é um grande desafio escolher e planejar a combinação adequada de técnicas que serão aplicadas nos testes de software e ao mesmo tempo satisfazer os custos previstos para o desenvolvimento do projeto.



Link

Acesse o *link* a seguir para ler a respeito de defeitos, erros e falhas relacionados a testes de software.

<https://qr.go.page.link/znWEQ>

Causas e defeitos em software e o papel do analista de teste

Defeitos (*bugs*) são imperfeições ou problemas que podem ser encontrados no software ou em seus processos de levantamento de requisitos, análise e projeto. Os defeitos também podem decorrer de algo que foi implementado de maneira errada no código-fonte. Porém, os *bugs* não são causados somente por erros de programação. Segundo o International Software Testing Qualifications Board (2016) (selo de qualidade para testadores de software, criado em 2002), há diferenças entre um *bug*, um erro e uma falha. Veja a seguir.

- **Bug:** trata-se do resultado de um erro de código. Uma anomalia é gerada no funcionamento do software por meio de uma instrução errada ou um comando incorreto.
- **Erro:** é decorrente da ação humana. Um resultado incorreto é produzido, como uma falha de escrita em um código-fonte.
- **Falha:** é o resultado da execução de um defeito gerado no código.

Durante as fases de análise de requisitos, especificação, desenho e codificação do sistema, podem ocorrer erros ou defeitos. Eles geralmente são causados por:

- especificações de requisitos realizadas de maneira errada;
- requisitos interpretados de forma incorreta pelos analistas;
- especificações funcionais e especificações técnicas mal formuladas;
- códigos-fonte implementados de forma incorreta;
- dados de entrada ou de saída errados ou inconsistentes;
- *bugs* corrigidos de forma errada.

A correlação existente entre as causas e os defeitos gerados está representada no Quadro 1, a seguir.

Quadro 1. Correlação entre causas e defeitos gerados

Defeito	Causa
Não tratamento de erros	Falta de precaução contra travamentos e comportamentos inesperados do sistema em relação às funcionalidades, bem como não antecipação de falhas e possíveis erros.
Erro de cálculo	Cálculos realizados de forma incorreta, como estouro de tamanho de campos e algoritmos implementados de forma errada. Por exemplo, no caso de implementação em formulários, o tamanho de um campo pode não ser o mesmo esperado no banco de dados; nesse caso, os dados não serão salvos corretamente.
Manipulação de dados	Dados tratados de forma errada, como datas e campos nulos.
Código-fonte	Não realização do controle correto do código-fonte do software; recursos que não tenham sido previstos e que estejam implementados no código.
Documentação	Não manutenção da documentação atualizada de acordo com o que foi solicitado, assim como das mudanças realizadas ao longo do projeto (todas devem ser devidamente registradas).
Testes	Ausência de planejamento, execução e definição de escopos de desenvolvimento adequados; falta de uma política de testes.
Usabilidade	Dificuldades de usabilidade no que diz respeito à facilidade de uso, à forma como são dispostas as informações na tela, a interface agradável, etc.

As causas dos defeitos geralmente se relacionam à especificação. Entre elas, você pode considerar erros no levantamento dos requisitos, questões de design e erros na implementação do código-fonte. Entre essas causas, a que se destaca é o levantamento dos requisitos. Nesse sentido, o analista de sistemas deve estar atento às mudanças, aos requisitos e à implementação.

O profissional de teste de software tem como premissa realizar a análise do sistema sob a visão dos testes para que possa modelar e construir os casos de testes. Os casos de testes são um conjunto de condições, valores de entrada, pré-condições de execução, entre outros elementos desenvolvidos para determinado objetivo ou condição.

O analista de teste deve analisar o software de forma criteriosa, prestar atenção aos detalhes, entender as falhas de software, conhecer o sistema ou aplicativo de teste e ter experiência em testes. Além disso, ele deve conhecer técnicas, por exemplo:

- UML — linguagem que define artefatos que ajudam nas tarefas de modelar e documentar os sistemas orientados a objetos;
- modelo V — modelo de verificação e validação, em que se observa se o software está sendo desenvolvido da maneira correta e se ele atende ao que foi solicitado;
- normas — normas e padrões de qualidade, tais como ISO 9000 e ISO 9126/IEC;
- banco de dados — estrutura do banco de dados;
- ferramentas de teste — ferramentas que auxiliam nos testes de software.

Entre as funções do analista de teste, você pode considerar:

- estabelecer estratégias, planejar, executar e monitorar testes de software;
- identificar os itens que deverão ser avaliados pelo teste;
- definir tipos de testes de acordo com os dados de entrada;
- coletar e gerenciar os dados de testes;
- avaliar o resultado de cada ciclo de teste.

A qualidade do software e o papel do teste

Um software de qualidade é aquele que satisfaz as necessidades dos usuários. Pressman e Maxim (2016) definem a qualidade de software como uma gestão efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam.

Existem fatores de qualidade internos e externos. Em relação aos fatores internos, tratando-se da estrutura do software, a avaliação é relacionada sob a visão do desenvolvedor. Geralmente, quem realiza é o responsável por toda a manutenção do software, inclusive pela modularidade, pela portabilidade, etc.

Os sistemas de software devem ser rápidos, confiáveis, fáceis de usar, legíveis, modulares, estruturados e assim por diante. Mas esses adjetivos descrevem dois tipos diferentes de qualidades. Há, por exemplo, qualidades como velocidade ou facilidade de uso, cuja presença ou ausência em um produto de software pode ser detectada pelos usuários. Essas propriedades podem ser chamadas de fatores de qualidade externos. A chave para alcançar esses fatores externos está nos internos: para que os usuários aproveitem qualidades visíveis, os designers e implementadores devem ter aplicado técnicas internas, o que garante as qualidades ocultas (MEYER, 1997).

Entre os fatores externos, você pode considerar:

- modularidade — software constituído por módulos;
- portabilidade — facilidade de utilizar o mesmo software em ambientes diferentes;
- correção — realização de tarefas de acordo com a especificação de requisitos.

Pressman e Maxim (2016) mencionam os fatores de qualidade ISO 9126, que definem os atributos fundamentais de qualidade de um software para computador. Esse padrão ISO identifica seis atributos fundamentais de qualidade de software:

- funcionalidade — implica a satisfação das necessidades declaradas com adequabilidade, exatidão, conformidade e segurança;
- confiabilidade — é medida pela probabilidade de ocorrência de falhas;
- usabilidade — é medida por fatores não quantificáveis, entre eles os itens de interface;

- eficiência — implica a boa utilização de recursos como memória e processadores;
- extensibilidade — é a possibilidade de adaptação a novas inclusões;
- facilidade de manutenção — significa que a correção pode ser realizada no software com base nos atributos de facilidade de análise, realização de mudanças, estabilidade e testabilidade.

A seguir, veja os fatores definidos pela norma ISO 9126 (2001) para que o software tenha um nível de qualidade adequado.

- **Correção:** implica a capacidade de o software realizar as tarefas de forma precisa, ou seja, de acordo com os requisitos especificados pelo cliente.
- **Extensibilidade:** é a forma como se podem inserir modificações no software; ou seja, ele deve ser flexível o suficiente para que modificações sejam realizadas de forma fácil.
- **Reusabilidade:** é a facilidade com que os softwares podem ser reutilizados totalmente ou em partes para novas aplicações. Isso faz com que haja economia e níveis de qualidade satisfatórios durante a produção de novos softwares, pois ocorrerá menor esforço na escrita e menor risco de erros.
- **Robustez (confiabilidade):** essa capacidade mostra que o software funciona mesmo em condições não validadas nas especificações dos requisitos.
- **Compatibilidade:** refere-se à facilidade de combinar softwares com outros componentes. Ou seja, ao ser utilizado, o software deve funcionar plenamente sem interferir em outras aplicações. Dessa forma, não deve haver problemas devido à execução de dois ou mais tipos de softwares ao mesmo tempo. Como exemplo, considere a estrutura de dados padronizada, as interfaces homem-máquina padronizadas ou ainda a padronização de formatos de arquivos.
- **Portabilidade:** é a facilidade com que um software pode ser transposto de um ambiente para outro, de acordo com os subatributos: adaptabilidade, facilidade de instalação, conformidade e facilidade de substituição (PRESSMAN; MAXIM, 2016). É um dos fatores difíceis de se obter, pois nem sempre é possível alinhar o software às diferentes plataformas, sistemas operacionais e periféricos.

- **Eficiência:** esse fator está diretamente relacionado à utilização racional dos recursos de hardware e do sistema operacional em que o software será instalado. Entre esses recursos, estão: memória, recursos gráficos, bibliotecas, entre outros.



Fique atento

A garantia da qualidade de um software (*Software Quality Assurance* — SQA) envolve vários aspectos. Entre eles: a adoção de normas e padrões internacionais (ISO e IEEE), as revisões técnicas e os testes de software utilizados para encontrar erros, a análise e a coleta de erros, as auditorias para assegurar que as diretrizes sejam seguidas, o gerenciamento de mudanças, a educação continuada dos engenheiros de software, a gerência dos fornecedores e a adoção de políticas de segurança e gestão de riscos.



Referências

APAI-CRVS. *Planeamento de implementação*: 6. definir o plano de abordagem e testes. [S. l.: s. n., 201-?]. Disponível em: <http://www.crvs-dgb.org/pt/activities/planeamento-de-implementac%cc%a7a%cc%83o/6-definir-o-plano-de-abordagem-e-testes/>. Acesso em: 27 jun. 2019.

INTERNATIONAL SOFTWARE TESTING QUALIFICATIONS BOARD. Home. [S. l.: s. n.], 2016. Disponível em: <https://www.istqb.org/>. Acesso em: 27 jun. 2019.

ISO. *ISO/IEC 9126-1: Software engineering — Product quality*. Rio de Janeiro: ISO, 2001.

MEYER, B. *Object-oriented software construction*. New Jersey: Prentice Hall, 1997.

PRESSMAN, R.; MAXIM, B. *Engenharia de software: uma abordagem profissional*. 8. ed. Porto Alegre: AMGH, 2016.

Leituras recomendadas

ALMEIDA, C. *Introdução ao teste de software*. [S. l.: s. n., 201-?]. Disponível em: <http://www.linhadecodigo.com.br/artigo/2775/introducao-ao-teste-de-software.aspx>. Acesso em: 27 jun. 2019.

DEVMEDIA. *Testes de software: entendendo defeitos, erros e falhas*. Rio de Janeiro: DEVMEDIA, 2019. Disponível em: <https://www.devmedia.com.br/testes-de-software-entendendo-defeitos-erros-e-falhas/22280>. Acesso em: 27 jun. 2019.

KOSCIANSKI, A.; SOARES, M. *Qualidade de software*. 2. ed. São Paulo: Novatec, 2007.

PEZZÊ, M.; YOUNG, M. *Teste e análise de software: processos, princípios e técnicas*. Porto Alegre: Bookman, 2008.

SCHACH, S. R. *Engenharia de software: os paradigmas clássicos & orientado a objetos*. Porto Alegre: AMGH, 2010.

SOFTWARE & SYSTEMS ENGINEERING STANDARDS COMMITTEE. *IEEE standard glossary of software engineering terminology*. [S. l.]: IEEE, 2019. Disponível em: <http://standards.ieee.org/findstds/standard/610.12-1990.html>. Acesso em: 27 jun. 2019.

SOUZA, K.; GASPAROTTO, A. A importância da atividade de teste no desenvolvimento de software. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 33., 2013. *Anais* [...]. Salvador, 2013. Disponível em: http://www.abepro.org.br/biblioteca/enegep2013_TN_STO_177_007_23030.pdf. Acesso em: 27 jun. 2019.

TARGETTRUST. Os principais tipos de testes de software. In: TARGETTRUST. [S. l.: s. n.], 2015. Disponível em: <https://targettrust.com.br/blog/os-13-principais-tipos-de-testes-de-software/>. Acesso em: 27 jun. 2019.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS

Dica do Professor

Para a realização de testes de *software* é fundamental que você entenda as diferenças entre os tipos de testes.

Na Dica do Professor, você aprenderá a diferença entre testes alfa e teste beta.



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

- 1) Supondo que um novo *software* voltado para redes sociais para um público específico esteja sendo desenvolvido, após o término do desenvolvimento das funcionalidades, e depois de ocorrer os testes funcionais e estruturais, o que mais seria pertinente realizar?
 - A) Teste de performance, pois seria somente um usuário realizando o teste, que tem como foco a usabilidade.
 - B) Revisões técnicas eficientes, especificações funcionais e não funcionais corretas e elaboração de um plano de teste.
 - C) Acompanhamento de usuário realizando testes, especificações menos detalhadas e elaboração de um plano de teste não funcional.
 - D) Verificação de linguagem de programação utilizada, *hardware* e conhecimento do desenvolvedor.
 - E) Realização de testes de desempenho, de *stress* e preocupação somente com a implementação do *software*.
- 2) O teste do *software* tem como objetivo fornecer informações sobre a qualidade do *software* em relação ao contexto no qual ele deverá funcionar. Dentre os testes de *software*, pode-se incluir:
 - I. Caixa-preta.
 - II. Caixa-cinza.
 - III. Caixa-branca.
 - IV. Teste de performance.
 - V. Teste de integração.
 - A) Alternativa I.
 - B) Alternativas I e II.
 - C) Alternativas II, III e V.
 - D) Alternativas I, III e IV.

E) Alternativas I, II, III, IV e V.

3) Técnicas de verificação e validação são de grande importância para identificar se um *software* tem defeitos e se está de acordo com o especificado. O que diferencia validação e verificação em um teste de *software*?

A) O tipo de revisões técnicas na implementação e na maneira de debugar o código-fonte.

B) Na validação ocorre uma verificação estática e na verificação ela é realizada de uma forma dinâmica.

C) Na validação ocorre uma verificação dinâmica e na verificação ela é realizada de uma forma estática.

D) Na verificação é assegurado que o *software* seja robusto e, na validação, que ele tenha portabilidade.

E) A validação assegura que o *software* foi criado de acordo com os requisitos, enquanto a verificação deve ter um plano de testes.

4) Em testes de *software*, há os fatores ISO de qualidade que definem parâmetros, cujo objetivo é padronizar a avaliação da qualidade de *software*. A partir disso, é correto afirmar que os fatores:

A) Compatibilidade mostra que o *software* será capaz de funcionar mesmo em condições que não tenham sido validadas nas especificações.

B) Portabilidade mostra que o *software* será capaz de funcionar mesmo em condições que não tenham sido validadas nas especificações.

C) Eficiência mostra que o *software* será capaz de funcionar mesmo em condições que não tenham sido validadas nas especificações.

D) Não levam, necessariamente, à medição direta, porém fornecem base para medidas indiretas e uma lista de verificação que norteia a avaliação de qualidade.

E) Facilidade de uso mostra que o *software* será capaz de funcionar mesmo em condições que não tenham sido validadas nas especificações.

5) Dentre as técnicas empregadas no projeto de interfaces de sistemas, uma delas é a condução de testes de usabilidade, em que os resultados fornecem importantes indicadores

para melhorar a qualidade da interface. Em que consistem os testes de usabilidade?

- A) Mostrar o sistema para um grupo de pessoas e coletar a opinião dos participantes sobre a interface e os conceitos.
- B) Mostrar o sistema para um usuário, solicitar que ele realize algum tipo de tarefa, observando as suas reações à interface, como, por exemplo, erros cometidos, dificuldades e eficiência no cumprimento da tarefa.
- C) Entregar um questionário para os usuários do sistema com perguntas sobre a interface e mapear as respostas.
- D) Submeter o sistema a um *software* robô de teste e verificar os tempos de resposta a cada padrão de navegação, verificando se estão de acordo com os requisitos não funcionais do sistema.
- E) Utilizar um *software* robô de teste para fazer acessos aleatórios ao sistema, que tenta utilizar elementos da interface e registra erros encontrados.

Na prática

De acordo com a Norma Brasileira NBR 9050/2004, acessibilidade é a possibilidade e a condição de alcance, percepção e entendimento para a utilização, com segurança e autonomia, de edificações, espaço, mobiliário, equipamento urbano e elementos.

Acessibilidade é a possibilidade de que qualquer pessoa possa usufruir de benefícios de uma vida em sociedade, entre eles, a utilização de *softwares*, sistemas e Internet, chamada, neste caso, também de inclusão digital.

Jorge, que trabalha com testes de sites para uma empresa, será responsável por fazer um *checklist* de testes em relação à acessibilidade dos produtos desenvolvidos.

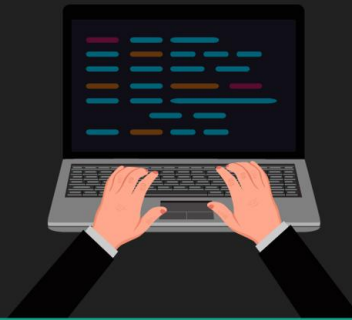
Acompanhe o Na Prática!

CHECKLIST DE ACESSIBILIDADE

O **checklist** é uma lista de itens que deverão ser verificados, tornando o trabalho padronizado, fazendo com que se tenha **maior agilidade e se mantenha as verificações sob controle dentro do processo de teste de software.**

Dentre as vantagens em utilizar **checklist** estão a garantia de que as **tarefas serão executadas de uma forma objetiva, padronizada e eficiente** por qualquer profissional da equipe.

O processo de **teste de software**, realizado a partir dos itens que estejam no **checklist**, agregará qualidade e tornará mais fácil a realização dos testes.



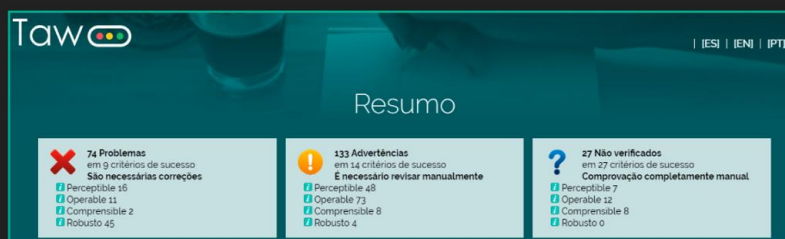
Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Dentre os itens do **checklist** direcionados à acessibilidade e à inclusão digital estão os seguintes itens que deverão constar no processo de teste de software:

- ▶ Textos alternativos para as imagens, de forma que transmitam informação.
- ▶ Textos alternativos para imagens decorativas.
- ▶ Verificar a possibilidade de escutar conteúdos de vídeo e áudio sem volume.
- ▶ Acessibilidade nos formulários.
- ▶ Possibilidade de aumentar o tamanho dos textos.
- ▶ Verificar se o site funciona corretamente em um **browser Lynx** (trata-se de um **browser** de texto).
- ▶ Verificar a navegabilidade sem o uso do **mouse**.
- ▶ Verificar se existe um mapa do **site** com fácil localização.
- ▶ Verificar se textos dos **links** são descritivos para o que eles remetem.
- ▶ Verificar o **site** em programas automatizados.

Para realizar testes de acessibilidade na web, Jorge acessou o **site TAW, ferramenta automática on-line para analisar a acessibilidade de websites**, e selecionou o nível de análise e tecnologias que deveriam ser avaliadas para o **site** desenvolvido. O **site** a avaliar pode ser um portal de notícias ou jornais, pois este apresenta preocupação com a acessibilidade.

Após analisar o **site** indicado, será apresentado como resultado um **resumo** contendo os seguintes itens:



Também é possível acessar um relatório detalhado que mostra os elementos do teste.

Ao final do teste, verificou-se que dependendo do nível de análise, há muitas alterações a serem realizadas em relação à acessibilidade no que se refere à **perceptividade, operabilidade, compreensibilidade e robustez**.

Essas correções deverão ser aplicadas aos itens do **checklist** para que o processo de teste de software ajude a obter a **acessibilidade desejada, possibilitando o alcance, mantendo a percepção, a interação para a utilização, trazendo igualdade de oportunidades com segurança para qualquer indivíduo**, independentemente do local onde esteja e do computador que esteja acessando.

Assim, garantirá a acessibilidade, utilizando qualquer tipo de tecnologia de navegação.