

DESENVOLVIMENTO DE SISTEMAS

Testes de Software



SUMÁRIO

Apresentação	3
Testes de Software	5
Visão Geral Sobre os Testes de Software.....	5
Verificação e Validação.....	5
Defeitos, Erros e Falhas	5
Defeitos no Desenvolvimento de Software.....	8
Principais Níveis de Teste de Software (Quando Testar)	10
Teste de Unidade (Testes Unitários)	11
Teste de Integração.....	12
Teste de Validação	13
Teste de Sistema	14
Técnicas de Teste de Software (Como Testar)	15
Técnica de Caixa Branca (Estrutural, White Box ou Caixa de Vidro)	15
Técnica de Caixa Preta (Funcional, Comportamental ou Black Box).....	15
Técnica de Caixa Cinza	16
Tipos de Teste de Software (O Que Testar)	16
Teste de Desempenho.....	17
Teste de Carga	17
Teste de Stress.....	17
Teste de Vulnerabilidades.....	17
Teste de Segurança	17
Exemplos de Artefatos Produzidos no Processo de Teste.....	18
Resumo	19
Questões Comentadas em Aula.....	20
Questões de Concurso	21
Gabarito	24
Gabarito Comentado	25
Referências	32

APRESENTAÇÃO

Olá, querido (a) amigo(a), tudo bem?



Quer conquistar a sua vitória? Então, tenha uma postura grandiosa! Todos que venceram grandes desafios se alimentavam com pensamentos **positivos, engrandecedores**, de **esperança** e de **coragem**.

Mesmo diante das adversidades, conserve a elegância e uma autoimagem positiva. Essa autoimagem é como se fosse um combustível que impulsionará as suas ações. #FicaaDica #OSegredo

Rumo então à aula sobre **Testes de Software**.

Em caso de dúvidas, acesse o fórum do curso ou entre em contato.

Um forte abraço,



TESTES DE SOFTWARE

VISÃO GERAL SOBRE OS TESTES DE SOFTWARE

Antes de iniciarmos uma discussão sobre **teste de software** precisamos esclarecer alguns conceitos relacionados a essa atividade.

VERIFICAÇÃO E VALIDAÇÃO

Para alguns estamos diante de conceitos similares: **verificação** e **validação**, porém dentro do contexto de software são coisas distintas que se complementam a fim de garantir a **qualidade do software**.

- **Verificação** é analisar se a forma com a qual estamos construindo um produto (software) é a correta, e
- **validação** é analisar se o produto (software) é o correto.

As perguntas que diferenciam verificação de validação, são:

1. Verificação: **Será que estamos construindo o produto corretamente.**
2. Validação: **Será que estamos construindo o produto correto.**

Podemos então concluir que a **verificação tem o objetivo de analisar a especificação do software**, enquanto a **validação** tem o objetivo de analisar se o que o cliente solicitou é o que de fato vai receber.

Na verificação temos então uma análise principalmente da **especificação dos requisitos funcionais e não-funcionais** e na validação **temos a expectativa do cliente em relação ao software sendo analisada**. Podemos ter verificação tanto estática como dinâmica.

Verificação	Objetivo
Estática	Na verificação estática , tem-se uma inspeção do software com a análise de documentação de requisitos, projeto, análise de código, etc.
Dinâmica	Na verificação dinâmica , tem-se os famosos testes , ou seja, na verificação dinâmica são analisados os dados de entrada e saída, e verifica-se se o resultado obtido está conforme o esperado.

Nenhum dos dois tipos de verificação, garante que não haja erros ou defeitos no software em todos os casos possíveis, ou seja: **nada aqui é 100% de garantia**.

DEFEITOS, ERROS E FALHAS

Vamos à diferença entre **Defeitos**, **Erros** e **Falhas**, conforme visto na figura seguinte.

- **Defeito** é um **ato inconsistente cometido por um indivíduo** ao tentar entender uma determinada informação, resolver um problema ou utilizar um método requisitos 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**. Uma falha pode ter sido causada por diversos erros e alguns erros podem nunca causar uma falha.

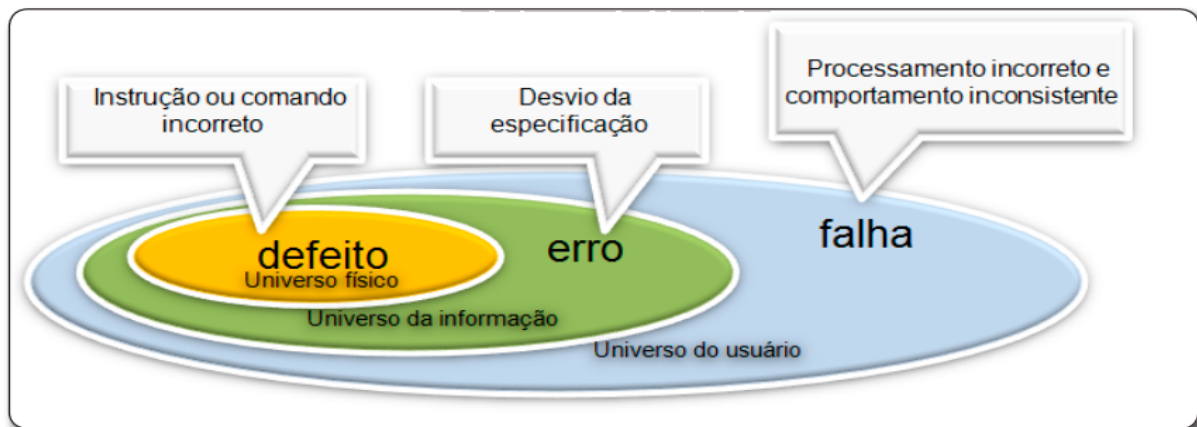


Figura. Defeito, erro e falha

Observe pela figura que **defeitos** fazem parte do universo físico (a aplicação propriamente dita) e são causados por pessoas, por exemplo, através do mal uso de uma tecnologia. **Defeitos** podem ocasionar a manifestação de erros em um produto, ou seja, a construção de um software de forma diferente ao que foi especificado (universo de informação).

Por fim, **os erros geram falhas**, que são comportamentos inesperados em um software que afetam diretamente o usuário final da aplicação (universo do usuário) e pode inviabilizar a utilização de um software.

Dessa forma, ressaltamos que **teste de software revela simplesmente falhas em um produto**.

Após a execução dos testes é necessária a execução de um processo de **depuração** para a **identificação** e **correção** dos **defeitos** que originaram essa **falha**, ou seja, **depurar não é testar!**

Falhas são esperadas nos testes, ou seja, quando testamos é normal que tenhamos comportamentos incorretos, identificando estes, executamos os ajustes necessários.

Antes de continuar, gostaria de reforçar o entendimento de que o **teste NÃO é uma fase que deve ser executada ao final do projeto, mas uma atividade que deve ser exercida ao longo do processo de desenvolvimento do software**.

A **atividade de teste é composta por alguns elementos essenciais** que auxiliam na formalização desta atividade. São eles:

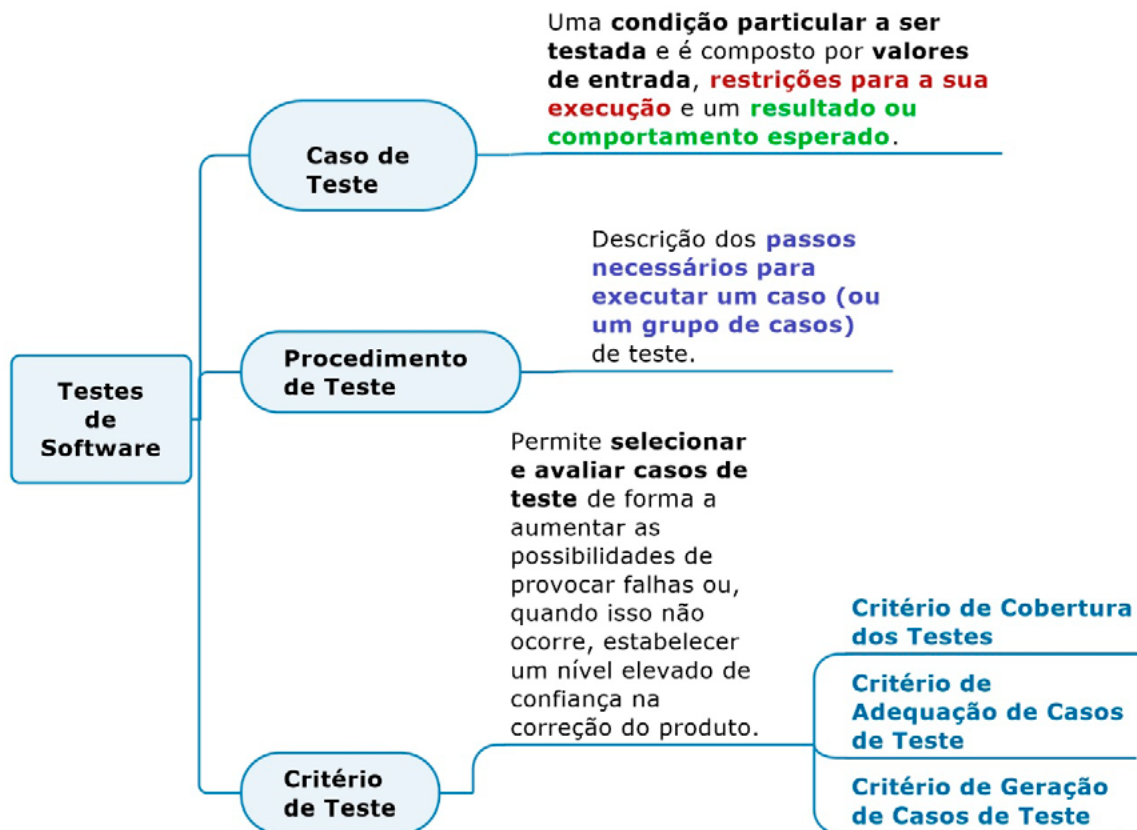


Figura. Testes de Software (QUINTÃO, 2022)

- Os **critérios de teste** podem ser utilizados como:

Critério de Cobertura dos Testes	<p>Permitem a identificação de partes do programa que devem ser executadas para garantir a qualidade do software e indicar quando o mesmo foi suficientemente testado.</p> <p>Assim, busca determinar o percentual de elementos necessários por um critério de teste que foram executados pelo conjunto de casos de teste.</p>
Critério de Adequação de Casos de Teste	<p>Quando, a partir de um conjunto de casos de teste T qualquer, ele é utilizado para verificar se T satisfaz os requisitos de teste estabelecidos pelo critério.</p> <p>Avalia se os casos de teste definidos são suficientes ou não para avaliação de um produto ou uma função.</p>
Critério de Geração de Casos de Teste	<p>Quando o critério é utilizado para gerar um conjunto de casos de teste T adequado para um produto ou função, ou seja, este critério define as regras e diretrizes para geração dos casos de teste de um produto que esteja de acordo com o critério de adequação definido anteriormente.</p>

DEFEITOS NO DESENVOLVIMENTO DE SOFTWARE

No processo de desenvolvimento de software, todos os defeitos são humanos e, apesar do uso dos melhores métodos de desenvolvimento, ferramentas ou profissionais, permanecem presentes nos produtos, o que torna a **atividade de teste fundamental** durante o desenvolvimento de um software. E essa atividade corresponde ao **último recurso para avaliação do produto antes da sua entrega ao usuário final**.

O tamanho do projeto a ser desenvolvido e a quantidade de pessoas envolvidas no processo são dois possíveis fatores que aumentam a complexidade dessa tarefa, e consequentemente aumentam a probabilidade de defeitos. Assim, a ocorrência de **falhas** é inevitável.

Mas o que significa dizer que um programa **falhou**? Basicamente significa que o **funcionamento do programa não está de acordo com o esperado pelo usuário**. Por exemplo, quando um usuário da linha de produção efetua consultas no sistema das quais só a gerência deveria ter acesso.

Esse tipo de falha pode ser originado por diversos motivos:

- A **especificação pode estar errada** ou incompleta;
- A **especificação pode conter requisitos impossíveis de serem implementados** devido às limitações de hardware ou software;
- A **base de dados pode estar organizada de forma que não seja permitido distinguir os tipos de usuário**;
- Pode ser que haja um **defeito no algoritmo** de controle dos usuários.

Os **defeitos** normalmente são introduzidos na transformação de informações entre as diferentes fases do ciclo de desenvolvimento de um software.

Vamos seguir um exemplo simples de ciclo de vida de desenvolvimento de software:

EXEMPLO

Os **requisitos** expressos pelo cliente são relatados textualmente em um documento de especificação de requisitos. Esse documento é então transformado em **casos de uso**, que por sua vez foi o artefato de entrada para o projeto do software e definição de sua arquitetura utilizando diagramas de classes da UML. Em seguida, esses modelos de projetos foram usados para a construção do software em uma linguagem que não segue o paradigma orientado a objetos.

Observe que durante esse período uma série de transformações foi realizada até chegarmos ao produto final. Nesse meio tempo, defeitos podem ter sido inseridos.

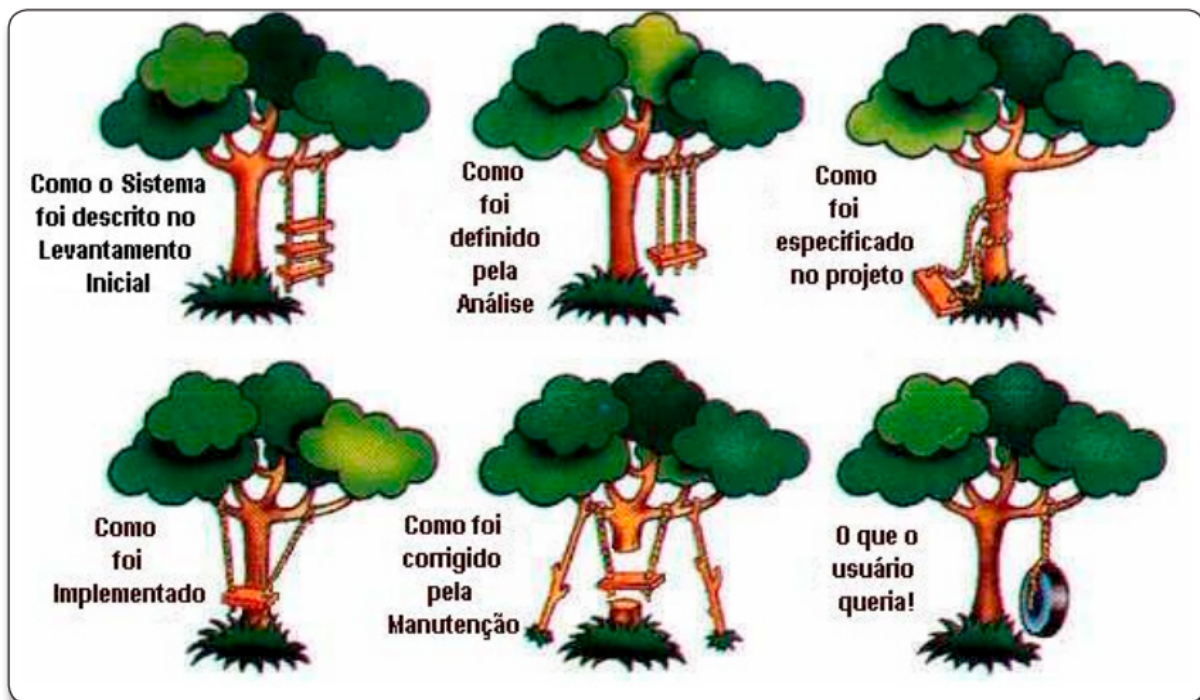


Figura. Diferentes interpretações ao longo do ciclo de desenvolvimento de um software

A figura anterior expressa exatamente a metáfora discutida nesse parágrafo. **Essa série de transformações resultou na necessidade de realizar testes em diferentes níveis**, visando avaliar o software em diferentes perspectivas de acordo com o produto gerado em cada fase do ciclo de vida de desenvolvimento de um software.

Quando o **teste** se inicia há um conflito de interesses:

- **desenvolvedores**: interesse em demonstrar que o programa é isento de erros;
- **responsáveis** pelos testes: interesse em mostrar que o programa tem erros.

Diversas atividades de **teste** são realizadas para **verificação do sistema construído**, levando-se em conta a especificação feita na fase de projeto.

O **teste de software** como o nome já diz significa **submeter um software a ser testado, com o objetivo de encontrar anomalias e problemas**, para de fato adequar o software ao que o cliente deseja, verificando assim o trabalho (código) implementado pelo programador.

Teste de software é uma atividade cara que exige tempo, planejamento, conhecimento técnico, infraestrutura e comprometimento.

Por maior e mais organizado que seja o esforço para a realização das atividades de testes, é **impossível garantir 100% de cobertura de todos os requisitos ou linhas de código existentes no sistema.**

Edsger Dijkstra sabiamente resumiu este fato com a seguinte frase: “**Testes podem mostrar a presença de erros, mas não a sua ausência**”. Dessa forma, está em suas mãos a responsabilidade de conhecer e aplicar as melhores práticas na gestão de testes para garantir a maior cobertura com o mínimo de esforço.




Uma importante razão para testar é identificar e excluir defeitos do software, mas **testar não garante que o software esteja isento de defeitos**.

Temos na literatura a conceituação de processo de teste que são as etapas implementadas a fim de orientar os testes que serão realizados.

Vamos aprofundar mais nesse assunto, destacando tipos de testes, níveis de teste e as técnicas de testes:

- os tipos de testes se referem **ao que vamos testar**;
- os níveis de testes se referem ao momento no qual vamos testar; e
- as técnicas de testes são de fato a forma pela qual nós vamos testar.

Veja o quadro seguinte com as três definições:

Técnica	Nível	Dimensões da Qualidade				
		Funcionalidade	Confiabilidade	Usabilidade	Desempenho	Suportabilidade
		Tipos de Teste (alguns exemplos)				
Caixa Branca	Teste de Unidade					
Caixa Cinza	Teste de Integração	Segurança	Integridade		Carga	Configuração
Caixa Preta	Teste de Sistema	Funcional	Regressão	Usabilidade		
	Teste de Aceitação	Volume	Maturidade		Estresse	Instalação
 Como Testar		 Quando Testar				
		 O que Testar				

PRINCIPAIS NÍVEIS DE TESTE DE SOFTWARE (QUANDO TESTAR)

Temos quatro níveis de testes: teste de unidade, teste de integração, teste de validação e teste de sistema, que vão delimitar QUANDO iremos testar.

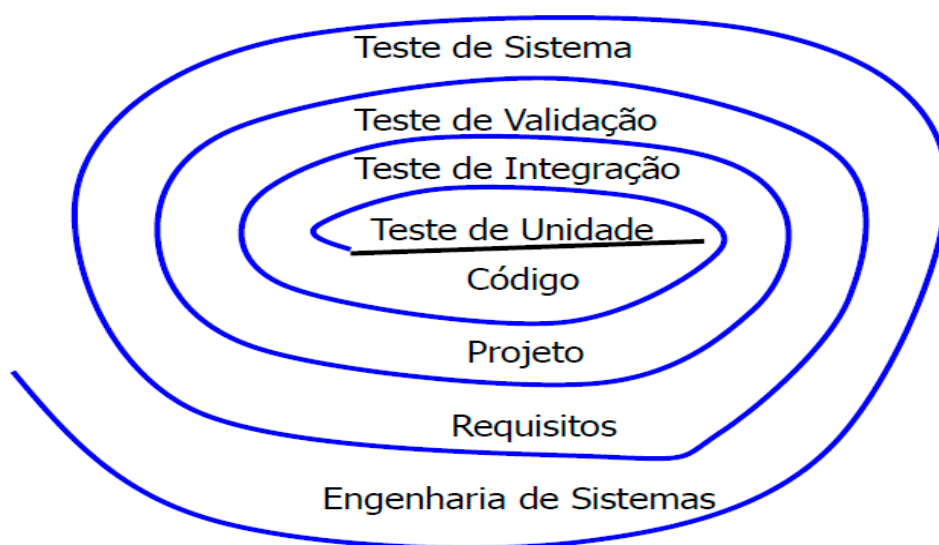


Figura. Estratégia de Teste (PRESSMAN, 2011)

Os **níveis de teste** são apresentados na figura conforme o contexto de cada um dos testes. Conforme visto, Pressman (2011) propõe esse **modelo espiral** para demonstrar a estratégia de teste.

- O **teste de unidade** começa no centro da espiral e se concentra em cada unidade (por exemplo: componente, classe, ou objeto de conteúdo de WebApp) do software conforme implementado no código-fonte.
- O teste prossegue movendo-se em direção ao exterior da espiral, passando pelo **teste de integração**, em que o foco está no projeto e construção da arquitetura de software.
- Continuando na mesma direção da espiral, encontramos o **teste de validação**, em que requisitos estabelecidos como parte dos requisitos de modelagem são validados em relação ao software criado.
- Finalmente, chegamos ao **teste do sistema**, no qual o software e outros elementos são testados como um todo. Para testar um software de computador, percorre-se a espiral em direção ao seu exterior, no sentido horário, ao longo de linhas que indicam o escopo do teste a cada volta.

Os principais **níveis de teste de software** são:

TESTE DE UNIDADE (TESTES UNITÁRIOS)

O primeiro **tipo de teste** que temos é o **teste de unidade** que trabalha com técnicas de teste para detecção de erros em componentes menores (**unidades**) que serão integradas já testadas.

Tem por objetivo **explorar a menor unidade do projeto**, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente.

O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenos trechos de código.

TESTE DE INTEGRAÇÃO

Tem o objetivo de analisar problemas na **construção (integração) dos componentes dentro de um software**. Visa provocar falhas associadas às interfaces entre os módulos quando esses são integrados para construir a estrutura do software que foi estabelecida na fase de projeto.

O teste de integração pode se basear nas seguintes abordagens convencionais:

a) Big-Bang

Integração **não** incremental. Todos os componentes são combinados com antecedência. O programa é testado como um todo.

b) Incremental

O programa é construído e integrado em pequenos incrementos.

i. **Descendente (top-down)**: os módulos são integrados deslocando-se para baixo através da hierarquia de controle, começando com o módulo de controle principal. Verifica os principais pontos de controle ou decisão antecipada no processo. Pode ser realizada: (1) primeiro-em-profundidade; (2) primeiro-em-largura.

ii. **Ascendente (bottom-up)**: começa a construção e o teste com módulos atômicos.

A seguir destacamos algumas abordagens especiais de teste de integração são:

- **Teste de regressão: não** corresponde a um nível de teste, mas é uma estratégia importante para redução de “efeitos colaterais”.

Consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema. Pode ser aplicado em qualquer nível de teste.

- **Teste fumaça**: séries de testes criada com o objetivo de expor erros que impedem uma construção (*build*) de executar exatamente sua função. A finalidade deverá ser descobrir erros “bloqueadores” que apresentam a mais alta probabilidade de atrasar o cronograma do software. A construção é integrada com outras construções, e o produto inteiro passa diariamente pelo teste fumaça

DIRETO DO CONCURSO

001. (CESPE/INPI/ANALISTA DE DESENVOLVIMENTO DE SISTEMAS/2013) No teste de integração, verifica-se se o funcionamento em conjunto dos componentes do sistema, se são chamados corretamente e se a transferência de dados acontece no tempo correto, por meio de suas interfaces.



O teste de integração visa provocar **falhas associadas às interfaces entre os módulos** quando esses são integrados para construir a estrutura do software que foi estabelecida na fase de projeto.
Certo.

TESTE DE VALIDAÇÃO

Nesse contexto os requisitos estabelecidos como parte dos requisitos de modelagem são validados em relação ao software criado.

O **teste de validação** começa quando termina o teste de integração, quando os componentes individuais já foram exercitados, o software está completamente montado como um pacote e os erros de interface já foram descobertos e corrigidos. O teste focaliza ações visíveis ao usuário e saídas do sistema reconhecidas por ele.

Fornece a garantia final de que o **software satisfaz aos requisitos informativos, funcionais, comportamentais e de desempenho**.

Os **testes de validação** incluem:

a) Aceitação Informal (Teste Alfa)

O **teste Alfa** é realizado por usuários e ocorre no ambiente de desenvolvimento, logo temos um **ambiente controlado** para os usuários realizarem os testes.

Alguns **benefícios** dessa forma de teste são (IBM, 1987):

- As funções e os recursos a serem testados são conhecidos.
- O progresso dos testes pode ser medido e monitorado.
- Os critérios de aceitabilidade são conhecidos.
- Serão revelados defeitos mais subjetivos do que no teste de aceitação formal.
- As **desvantagens** incluem (IBM, 1987):
- São necessários recursos, planejamento e recursos de gerenciamento.
- Você não tem controle sobre os casos de teste que são utilizados.
- os usuários podem se adaptar à forma como o sistema funciona e não encontrar defeitos.
- os usuários podem se concentrar na comparação do novo sistema com um sistema legado, em vez de procurar defeitos.
- Os recursos do teste de aceitação não estão mais sob o controle do projeto e podem ficar limitados.

b) Teste Beta

O **teste Beta** é realizado pelos usuários também, mas em **ambiente real**, no qual o software será utilizado. O cliente registra todos os problemas encontrados durante o teste e relata esses problemas para o desenvolvedor em intervalos regulares.

O **teste Beta** é o menos controlado das três estratégias de teste de aceitação. No **teste Beta**, a quantidade de detalhes, os dados e a abordagem adotadas são de inteira escolha do testador individual. Cada testador é responsável por criar o próprio ambiente, selecionar os dados correspondentes e determinar as funções, os recursos ou as tarefas a serem exploradas. Cada um deles é responsável por identificar seus próprios critérios para aceitar, ou não, o sistema em seu estado atual.

O **teste Beta** é implementado por usuários, geralmente com pouco ou nenhum gerenciamento por parte da organização de desenvolvimento (ou outra que não seja do usuário final). O **teste Beta** é o mais subjetivo de todas as estratégias de teste de aceitação.

Alguns **benefícios** dessa forma de teste são (IBM, 1987):

- O teste é **implementado por usuários**.
- Há grandes volumes de potenciais recursos de teste.
- Há uma **maior satisfação do cliente para aqueles que participam**.
- **São revelados defeitos mais subjetivos** que o teste de aceitação formal ou informal.

As **desvantagens** incluem (IBM, 1987):

- Talvez você não teste todas as funções ou os recursos.
- **É difícil medir o progresso do teste**.
- Os usuários podem se adaptar à forma como o sistema funciona e não encontrar ou relatar defeitos.
- Os usuários podem se concentrar na comparação do novo sistema com um sistema legado, em vez de procurar defeitos.
- Os recursos do teste de aceitação não estão mais sob o controle do projeto e podem ficar limitados.
- Os critérios de aceitabilidade não são conhecidos.
- São necessários recursos com suporte adicional para gerenciar os testadores beta.

c) Teste de Aceitação do Cliente

Uma variação do teste beta, chamada de **teste de aceitação do cliente**, às vezes é executada quando é fornecido software personalizado a um cliente sob contrato.

O cliente executa uma série de testes específicos na tentativa de descobrir erros antes de aceitar o software do desenvolvedor. Em alguns casos (Ex.: Em um grande sistema corporativo ou governamental) o teste de aceitação pode ser muito formal e levar vários dias ou mesmo semanas.



No **teste alfa** temos o usuário utilizando o software em **ambiente controlado** e no **teste beta** temos o usuário utilizando software em ambiente real.

A estratégia a ser selecionada baseia-se geralmente nos requisitos contratuais, nos padrões organizacionais e corporativos, bem como no domínio do aplicativo (IBM, 1987).

TESTE DE SISTEMA

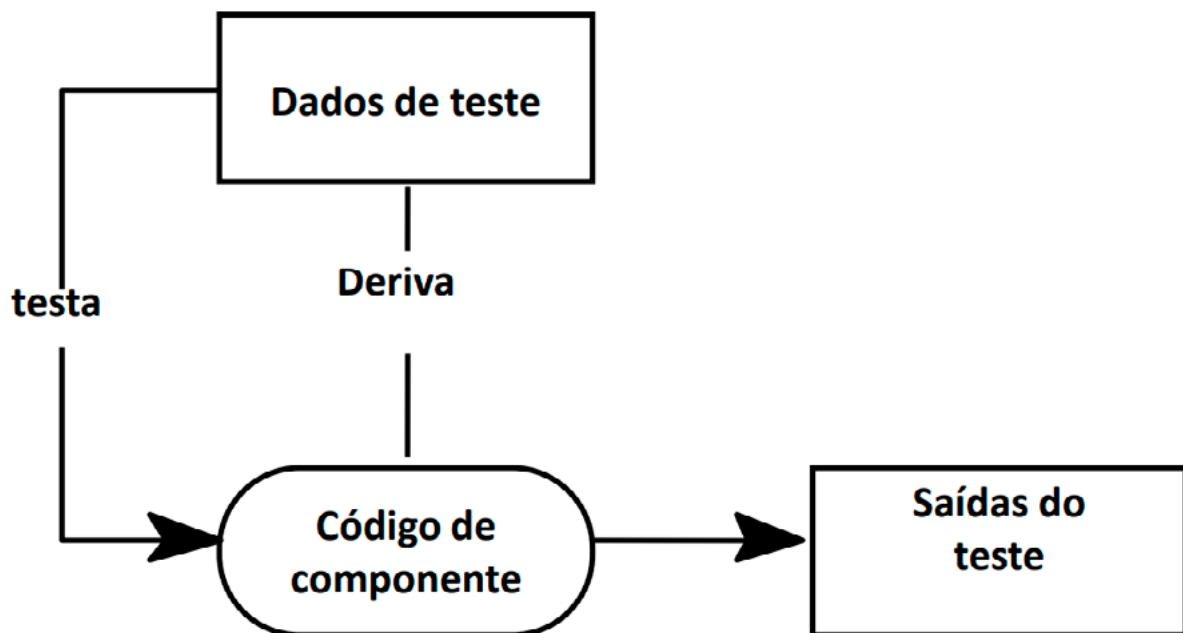
Verifica se os elementos estão coerentes na dimensão de sistema. Avalia o software em busca de **falhas** por meio da utilização do mesmo, como se fosse um usuário final. Dessa maneira, os testes são executados nos mesmos ambientes, com as mesmas condições e com os mesmos dados de entrada que um usuário utilizaria no seu dia-a-dia de manipulação do software. Verifica se o produto satisfaz seus requisitos.

TÉCNICAS DE TESTE DE SOFTWARE (COMO TESTAR)

Por fim temos as **técnicas de teste**: **caixa-branca**, **caixa-cinza** e **caixa-preta**.

TÉCNICA DE CAIXA BRANCA (ESTRUTURAL, WHITE BOX OU CAIXA DE VIDRO)

É responsável por testar os caminhos lógicos possíveis com foco na parte interna do software.



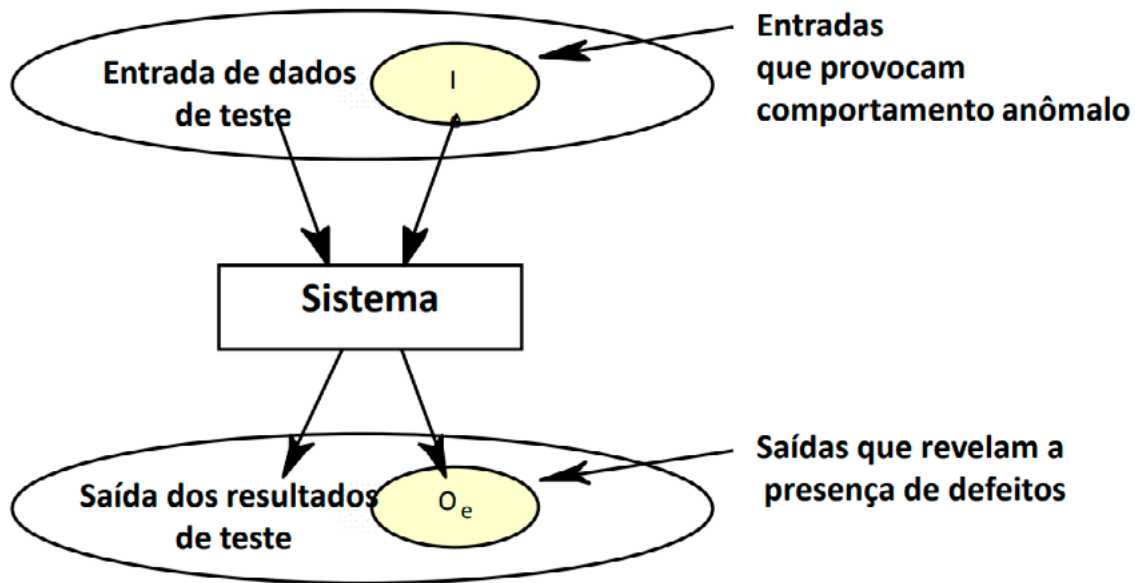
Fonte:
PRESSMAN, ROGER - Engenharia de Software - 6ª Edição
SOMMERVILLE - Engenharia de Software - 8ª Edição

TÉCNICA DE CAIXA PRETA (FUNCIONAL, COMPORTAMENTAL OU BLACK BOX)

Testa pré-condições avaliando o comportamento das funcionalidades conforme os requisitos **sem se preocupar com a lógica interna.**

Envolve dois **passos principais**:

- Identificar as funções que o software deve realizar (especificação dos requisitos);
- Criar casos de teste capazes de checar se essas funções estão sendo executadas corretamente.



DIRETO DO CONCURSO

002. (CESPE/MPU/ANALISTA JUDICIÁRIO – SISTEMAS/2013) Testes funcionais são aplicados para identificar não conformidades entre o programa e seus requisitos.



A técnica de **caixa preta (funcional)** testa pré-condições avaliando o comportamento das funcionalidades conforme os requisitos **sem se preocupar com a lógica interna**.

Logo eu estou buscando nos testes funcionais não conformidades entre as funcionalidades do software e os requisitos.

Certo.

TÉCNICA DE CAIXA CINZA

Utiliza conjuntamente as técnicas de caixa branca e caixa preta, em que conhecemos a estrutura interna do software e verificamos as estradas e saídas das funcionalidades, exemplo disso é o teste de integração.

TIPOS DE TESTE DE SOFTWARE (O QUE TESTAR)

Nos tipos de testes, estamos preocupados com O QUE vamos testar, logo temos vários exemplos dependendo daquilo que desejamos testar (Segurança, Integridade, **teste de carga**, teste de usabilidade, **Desempenho**, etc.).

Vamos detalhar alguns desses tipos principais:

TESTE DE DESEMPENHO

Trata-se de um tipo de teste de software que **ajuda a determinar o desempenho de um software** em termos de **velocidade**, **tempo de resposta**, **escalabilidade**, **uso de recursos** e **estabilidade** sob uma determinada carga de trabalho.

Exemplo de pergunta para esse teste: a aplicação suporta 1.000 transações por minuto com 1.000 usuários simultâneos?

TESTE DE CARGA

É a prática de simular o uso do mundo real, ou carregar, em qualquer software, site, aplicativo web, API ou sistema para analisar e identificar fatores como capacidade de resposta, degradação e escalabilidade.

Exemplo de pergunta para esse teste: quantas transações serão suportadas por minuto quando aumentarmos os usuários simultâneos para 2.000, 3.000, 4.000?

TESTE DE STRESS

Usado para determinar o limite da infraestrutura de TI. Procura estabelecer até que ponto a infraestrutura, hardware, software, etc. podem ser estressados e, a partir daí, fornecer um caderno de recomendações e melhorias.

O stress test na TI ajuda as empresas a saberem quais servidores, software e outras tecnologias precisam de manutenção para aguentar picos de requisições, bem como definir as tecnologias que devem ser alocadas para garantir o melhor desempenho da sua TI.

Exemplo de pergunta para esse teste: Quantas transações por minuto solicitadas por 5.000, 6.000, 7.000 usuários simultâneos, serão suportadas pela aplicação sob condições não especificadas do software e até mesmo do próprio hardware?

TESTE DE VULNERABILIDADES

Permitem descobrir quais **vulnerabilidades** estão presentes no ambiente através de revisões abrangentes e sistemáticas, buscando identificar no sistema quaisquer pontos fracos que o torne suscetível a ataques ou tentativas de *hacker*.

TESTE DE SEGURANÇA

- Tenta verificar **se os mecanismos de proteção do sistema irão de fato proteger o sistema contra invasão**.
- Tenta verificar **se todos os mecanismos de proteção embutidos num sistema o protegerão, de fato, de acessos indevidos**.
- Essa estratégia de teste deve ser feita por programadores que **não** desenvolveram o software.
- Os testadores devem tentar invadir o sistema de várias formas: obtenção ilegal de senhas, desarme do sistema, etc.

EXEMPLOS DE ARTEFATOS PRODUZIDOS NO PROCESSO DE TESTE

Ao fim do processo de teste, podemos ter artefatos como o **plano de testes** e os **casos de testes**.

O **plano de testes** representa o **planejamento para execução do teste**, os **casos de testes** apresentam as funcionalidades testadas.



O principal **produto** da **fase de Testes** é o **relatório de testes**, que **contém informações sobre erros detectados no software**. Após a atividade de testes, os diversos módulos do sistema são integrados, resultando finalmente no produto de software (BEZERRA, 2007).

RESUMO

- O **teste de software** é uma das atividades mais custosas do processo de desenvolvimento de software, pois pode envolver uma quantidade significativa dos recursos de um projeto.
- Diferentes tipos de aplicações possuem diferentes técnicas de teste a serem aplicadas, ou seja, testar uma aplicação web envolve passos diferenciados em comparação aos testes de um sistema embarcado.
- Cada tipo de aplicação possui características específicas que devem ser consideradas no momento da realização dos testes
 - **Teste de Unidade:** os pequenos elementos passíveis de teste são testados individualmente, geralmente ao mesmo tempo em que são implementados (KRUCHTEN, 2003).
 - **Teste “Caixa-Preta” (Black Box):** analisa a funcionalidade e a aderência aos requisitos, em uma ótica externa ou do usuário, sem se basear em qualquer conhecimento do código e da lógica interna do componente testado.
 - **Teste de Caixa Branca:** é uma abordagem para projetar casos de teste na qual os testes são derivados do conhecimento da estrutura e da implementação do software. O entendimento do algoritmo usado em um componente pode ajudar a identificar partições e casos de teste adicionais (SOMMERVILLE, 2007). **O teste “caixa-branca” avalia a lógica interna do sistema!**
 - **O Técnica de Caixa Cinza** utiliza conjuntamente as técnicas de caixa branca e caixa preta, em que conhecemos a estrutura interna do software e verificamos as estradas e saídas das funcionalidades, exemplo disso é o teste de integração.

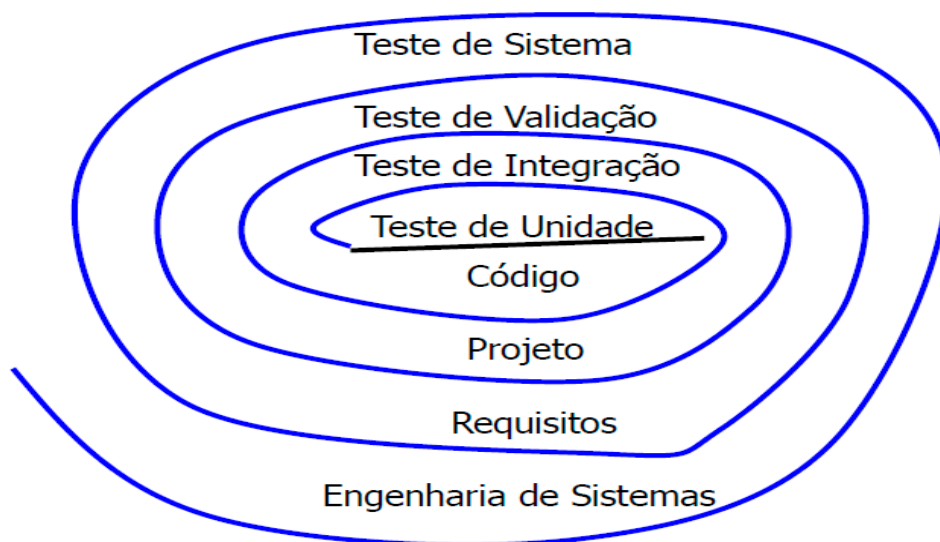


Figura. Estratégia de Teste (PRESSMAN, 2011)

QUESTÕES COMENTADAS EM AULA

001. (CESPE/INPI/ANALISTA DE DESENVOLVIMENTO DE SISTEMAS/2013) No teste de integração, verifica-se se o funcionamento em conjunto dos componentes do sistema, se são chamados corretamente e se a transferência de dados acontece no tempo correto, por meio de suas interfaces.

002. (CESPE/MPU/ANALISTA JUDICIÁRIO – SISTEMAS/2013) Testes funcionais são aplicados para identificar não conformidades entre o programa e seus requisitos.

QUESTÕES DE CONCURSO

003. (INSTITUTO AOCP/CÂMARA DE TERESINA – PI/ANALISTA DE INFORMÁTICA/2021)

Cleiton está realizando testes em um sistema e revisa o código procurando por problemas. Nesse caso, ele está realizando testes de qual tipo?

- a) Unitário.
- b) Integração.
- c) Caixa Branca.
- d) Caixa Preta.
- e) A. B.

004. (INSTITUTO AOCP/PREFEITURA DE NOVO HAMBURGO – RS – TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS/2020) Qual das seguintes alternativas apresenta uma característica de um teste de caixa preta?

- a) Ignora os mecanismos internos de um sistema e focaliza as saídas geradas em resposta às entradas e condições de execução selecionadas.
- b) É um sinônimo de teste de caixa transparente.
- c) Leva em consideração os mecanismos internos de um sistema componente. É semelhante a um teste estrutural.
- d) Nesse tipo de teste, um exemplo de resultado é a quantidade de erros de acesso da aplicação ao banco de dados.
- e) Um teste de caixa preta é utilizado para determinar se todas as bibliotecas necessárias para a execução do sistema como um todo estão disponíveis para a aplicação.

005. (INSTITUTO AOCP/PREFEITURA DE NOVO HAMBURGO-RS/TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS/2020) Assinale a alternativa que apresenta o objetivo de um teste funcional.

- a) Avaliar os aspectos de integridade do banco de dados.
- b) Verificar requisitos de desempenho, como o número de terminais suportados.
- c) Determinar os requisitos de guarda e retenção de dados.
- d) Mapear os aspectos de instabilidade do sistema.
- e) Verificar a consistência entre o produto implementado e respectivos requisitos.

006. (CESPE (CEBRASPE)/SERPRO/ANALISTA – ESPECIALIZAÇÃO: DESENVOLVIMENTO DE SISTEMAS/2021) Acerca dos fundamentos e dos princípios da qualidade de software e da gestão da configuração, julgue o item que se segue. Realizado o teste unitário de um módulo, o teste de integração contribuirá para a avaliação da existência de erros associados às interfaces do sistema.

007. (CESPE/TJ-PA/ANALISTA JUDICIÁRIO/ANÁLISE DE SISTEMAS (DESENVOLVIMENTO)/2020) No processo de teste de software, o beta teste é

- a) realizado por uma equipe de teste independente.
- b) realizado pelos clientes no local de trabalho do desenvolvedor de software.
- c) realizado por clientes em seu próprio local de trabalho.
- d) útil para testar software sob medida.
- e) executado o mais cedo possível no ciclo de vida.

008. (FGV/BANESTES/ANALISTA EM TECNOLOGIA DA INFORMAÇÃO – SUPORTE E INFRAESTRUTURA/2018) No contexto de teste de software, o termo “Beta teste” caracteriza testes que:

- a) empregam primordialmente técnicas conhecidas como “White box”;
- b) são equivalentes aos testes conhecidos pelo termo “Alfa teste”;
- c) focam em pontos críticos, cujas correções são providenciadas de imediato pelos desenvolvedores;
- d) são realizados num ambiente de laboratório do desenvolvedor;
- e) são realizados por usuários externos, em condições de uso semelhantes às de produção.

009. (CESPE/BNB/ESPECIALISTA TÉCNICO – ANALISTA DE SISTEMA/2018) O item a seguir apresenta uma situação hipotética, seguida de uma assertiva a ser julgada, com relação a testes de software. Uma equipe de desenvolvimento de softwares pretendia realizar testes de forma incremental durante o desenvolvimento de um programa, a fim de verificar se mudanças no programa não haviam nele introduzido novos bugs; para isso, foram sugeridos os testes unitários e de regressão. Nessa situação, será correto utilizar os testes unitários, mas não os testes de regressão, pois esses últimos não visam verificar novos bugs, mas sim, tão somente, avaliar as funcionalidades do sistema.

010. (FGV/DPE-RJ/TÉCNICO SUPERIOR ESPECIALIZADO – DESENVOLVIMENTO DE SISTEMAS/2014) Testes unitários são amplamente empregados no desenvolvimento de software. Sua função principal é:

- a) testar o desempenho do software e de seus componentes.
- b) testar o menor bloco de software desenvolvido, avaliando os resultados obtidos com entradas de dados pré-definidos.
- c) avaliar o comportamento do sistema como um todo e como a integração dos componentes de software se comportam.
- d) avaliar a adequação do software desenvolvido com os requisitos unitários estruturais definidos pelos usuários.
- e) testar os cenários alternativos dos casos de uso, garantindo o comportamento esperado para o sistema.

011. (CESPE/MPE-PI/ANALISTA DE INFORMÁTICA/2012) Em teste funcional, o conjunto de valores de entrada válidos pode ser reduzido por meio de partição em classes de equivalência, o que torna a quantidade de dados de entrada finita.

012. (CESPE/MEC/ANALISTA DE SISTEMAS/2011) Os testes de usabilidade avaliam a facilidade de uso do software testado e são bastante utilizados em aplicações web.

013. (CESPE/TRT 17 REGIÃO/ANALISTA JUDICIÁRIO – SISTEMAS/2013) O teste de carga, que verifica o funcionamento do software, utiliza uma grande quantidade de usuários simultaneamente.

014. (CESPE/TRE-BA/TÉCNICO JUDICIÁRIO – PROGRAMAÇÃO/2010) Teste funcional é uma técnica para se projetar casos de teste na qual o programa ou sistema é considerado uma caixa-preta e, para testá-lo, são fornecidas entradas e avaliadas as saídas geradas.

015. (CESPE/ANAC/ANALISTA ADMINISTRATIVO/TECNOLOGIA DA INFORMAÇÃO/2012) Os testes funcionais são caracterizados pelo uso do sistema conforme o seu usuário regular o faria.

016. (CESPE/ANATEL/ARQUITETURA DE SOLUÇÕES/2012) Considere as informações abaixo em relação ao desenvolvimento de sistemas:

I – Executar um software com o objetivo de revelar falhas, mas que não prova a exatidão do software.

II – Correta construção do produto.

III – Construção do produto certo.

Correspondem corretamente a I, II e III, respectivamente,

a) Validação, verificação e teste.

b) Verificação, teste e validação.

c) Teste, verificação e validação.

d) Validação, teste e verificação.

e) Teste, validação e verificação.

017. (FCC/TRT 14 REGIÃO/TÉCNICO JUDICIÁRIO – PROGRAMAÇÃO/2011) Garantir o funcionamento correto do software para atender as expectativas do cliente é o objetivo da homologação de sistemas. Nessa fase, que precede à implantação, os testes mais comuns são os testes:

a) Funcionais, de usabilidade e de aceitação.

b) De unidade, de iteração e de integração.

c) De volume, de integridade e de aceitação.

d) Da caixa-branca, de carga e de configuração.

e) De unidade, de carga e de integridade.

GABARITO

1. C
2. C
3. c
4. a
5. e
6. C
7. c
8. e
9. E
10. b
11. C
12. C
13. C
14. C
15. E
16. c
17. a

GABARITO COMENTADO

003. (INSTITUTO AOCP/CÂMARA DE TERESINA – PI/ANALISTA DE INFORMÁTICA/2021)

Cleiton está realizando testes em um sistema e revisa o código procurando por problemas. Nesse caso, ele está realizando testes de qual tipo?

- a) Unitário.
- b) Integração.
- c) Caixa Branca.
- d) Caixa Preta.
- e) A. B.



Se Cleiton está acessando código, portanto, **avalia a lógica interna do sistema!** Nesse contexto, está usando um teste de caixa branca.

O teste “caixa-branca” é uma abordagem para projetar casos de teste na qual os **testes são derivados do conhecimento da estrutura e da implementação do software**. O entendimento do algoritmo usado em um componente pode ajudar a identificar partições e casos de teste adicionais (SOMMERVILLE, 2007).

O **Teste “Caixa-Preta” (Black Box)** analisa a funcionalidade e a aderência aos requisitos, em uma **ótica externa** ou do usuário, sem se basear em qualquer conhecimento do código e da **lógica interna do componente testado**.

O **Técnica de Caixa Cinza** utiliza **conjuntamente as técnicas de caixa branca e caixa preta**, em que conhecemos a estrutura interna do software e verificamos as estradas e saídas das funcionalidades, exemplo disso é o teste de integração.

Letra c.

004. (INSTITUTO AOCP/PREFEITURA DE NOVO HAMBURGO – RS – TÉCNICO EM

DESENVOLVIMENTO DE SISTEMAS/2020) Qual das seguintes alternativas apresenta uma característica de um teste de caixa preta?

- a) Ignora os mecanismos internos de um sistema e focaliza as saídas geradas em resposta às entradas e condições de execução selecionadas.
- b) É um sinônimo de teste de caixa transparente.
- c) Leva em consideração os mecanismos internos de um sistema componente. É semelhante a um teste estrutural.
- d) Nesse tipo de teste, um exemplo de resultado é a quantidade de erros de acesso da aplicação ao banco de dados.
- e) Um teste de caixa preta é utilizado para determinar se todas as bibliotecas necessárias para a execução do sistema como um todo estão disponíveis para a aplicação.



O **Teste “Caixa-Preta” (Black Box, comportamental ou funcional)** analisa a funcionalidade e a aderência aos requisitos, em uma ótica externa ou do usuário, sem se basear em qualquer conhecimento do código e da lógica interna do componente testado. **Ele ignora os detalhes internos** e tem o objetivo de verificar a consistência entre o produto implementado e respectivos requisitos.

Letra a.

005. (INSTITUTO AOCP/PREFEITURA DE NOVO HAMBURGO-RS/TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS/2020) Assinale a alternativa que apresenta o objetivo de um teste funcional.

- a) Avaliar os aspectos de integridade do banco de dados.
- b) Verificar requisitos de desempenho, como o número de terminais suportados.
- c) Determinar os requisitos de guarda e retenção de dados.
- d) Mapear os aspectos de instabilidade do sistema.
- e) Verificar a consistência entre o produto implementado e respectivos requisitos.



O **teste caixa-preta (comportamental ou funcional)** ignora os mecanismos internos de um sistema e focaliza as saídas geradas em resposta às entradas e condições de execução selecionadas. Verifica, portanto, a consistência entre o produto implementado e respectivos requisitos.

Letra e.

006. (CESPE (CEBRASPE)/SERPRO/ANALISTA – ESPECIALIZAÇÃO: DESENVOLVIMENTO DE SISTEMAS/2021) Acerca dos fundamentos e dos princípios da qualidade de software e da gestão da configuração, julgue o item que se segue. Realizado o teste unitário de um módulo, o teste de integração contribuirá para a avaliação da existência de erros associados às interfaces do sistema.



O primeiro tipo de teste que temos é o **teste de unidade** que trabalha com técnicas de teste para detecção de erros em componentes menores (**unidades**) que serão integradas já testadas. Em seguida temos o **teste de integração** que tem o objetivo de analisar problemas na construção (**integração**) dos componentes dentro de um software.

Em suma, o teste unitário testa o módulo e o de integração as interfaces!

Certo.

007. (CESPE/TJ-PA/ANALISTA JUDICIÁRIO/ANÁLISE DE SISTEMAS (DESENVOLVIMENTO)/2020) No processo de teste de software, o beta teste é

- a) realizado por uma equipe de teste independente.
- b) realizado pelos clientes no local de trabalho do desenvolvedor de software.
- c) realizado por clientes em seu próprio local de trabalho.
- d) útil para testar software sob medida.
- e) executado o mais cedo possível no ciclo de vida.



No **teste alfa** temos o usuário utilizando o software em **ambiente controlado** e no **teste beta** temos o usuário utilizando software em ambiente real.

Assim, no processo de teste de *software*, o beta teste é realizado por clientes em seu próprio local de trabalho.

Letra c.

008. (FGV/BANESTES/ANALISTA EM TECNOLOGIA DA INFORMAÇÃO – SUPORTE E INFRAESTRUTURA/2018) No contexto de teste de software, o termo “Beta teste” caracteriza testes que:

- a) empregam primordialmente técnicas conhecidas como “White box”;
- b) são equivalentes aos testes conhecidos pelo termo “Alfa teste”;
- c) focam em pontos críticos, cujas correções são providenciadas de imediato pelos desenvolvedores;
- d) são realizados num ambiente de laboratório do desenvolvedor;
- e) são realizados por usuários externos, em condições de uso semelhantes às de produção.



No contexto de teste de software, o termo “Beta teste” caracteriza testes que são realizados por usuários externos, em condições de uso semelhantes às de produção.

No **teste alfa** temos o usuário utilizando o software em **ambiente controlado** e no **teste beta** temos o usuário utilizando software em ambiente real.

Letra e.

009. (CESPE/BNB/ESPECIALISTA TÉCNICO – ANALISTA DE SISTEMA/2018) O item a seguir apresenta uma situação hipotética, seguida de uma assertiva a ser julgada, com relação a testes de software. Uma equipe de desenvolvimento de softwares pretendia realizar testes de forma incremental durante o desenvolvimento de um programa, a fim de verificar se mudanças no programa não haviam nele introduzido novos bugs; para isso, foram sugeridos os testes unitários e de regressão. Nessa situação, será correto utilizar os testes unitários, mas não os testes de regressão, pois esses últimos não visam verificar novos bugs, mas sim, tão somente, avaliar as funcionalidades do sistema.



O **teste de regressão** consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema para assegurar que as alterações não tenham propagado efeitos colaterais indesejados. Pode ser aplicado em qualquer nível de teste.

Errado.

010. (FGV/DPE-RJ/TÉCNICO SUPERIOR ESPECIALIZADO – DESENVOLVIMENTO DE SISTEMAS/2014) Testes unitários são amplamente empregados no desenvolvimento de software. Sua função principal é:

- a) testar o desempenho do software e de seus componentes.
- b) testar o menor bloco de software desenvolvido, avaliando os resultados obtidos com entradas de dados pré-definidos.
- c) avaliar o comportamento do sistema como um todo e como a integração dos componentes de software se comportam.
- d) avaliar a adequação do software desenvolvido com os requisitos unitários estruturais definidos pelos usuários.
- e) testar os cenários alternativos dos casos de uso, garantindo o comportamento esperado para o sistema.



Os **testes de unidade** (ou **testes unitários**) têm por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente. O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenos trechos de código.

Letra b.

011. (CESPE/MPE-PI/ANALISTA DE INFORMÁTICA/2012) Em teste funcional, o conjunto de valores de entrada válidos pode ser reduzido por meio de partição em classes de equivalência, o que torna a quantidade de dados de entrada finita.



Em um **teste funcional (caixa preta)** tem-se valores de entrada que podem ser reduzidos através de partição em classes, limitando assim a quantidade de dados.

Certo.

012. (CESPE/MEC/ANALISTA DE SISTEMAS/2011) Os testes de usabilidade avaliam a facilidade de uso do software testado e são bastante utilizados em aplicações web.



Teste de usabilidade é um tipo de teste utilizado para avaliar o uso do software.

Certo.

013. (CESPE/TRT 17 REGIÃO/ANALISTA JUDICIÁRIO – SISTEMAS/2013) O teste de carga, que verifica o funcionamento do software, utiliza uma grande quantidade de usuários simultaneamente.



No **teste de carga** verificamos o funcionamento do software, em situações extremas, como uma grande quantidade de usuários acessando uma funcionalidade ao mesmo tempo.

Certo.

014. (CESPE/TRE-BA/TÉCNICO JUDICIÁRIO – PROGRAMAÇÃO/2010) Teste funcional é uma técnica para se projetar casos de teste na qual o programa ou sistema é considerado uma caixa-preta e, para testá-lo, são fornecidas entradas e avaliadas as saídas geradas.



No teste caixa preta eu me preocupo com o que entra e o que sai do meu software, o que acontece no meio do caminho, na estrutura interna não é importante aqui, logo o teste caixa preta também é chamado de teste funcional, porque o importante aqui é o que as funcionalidades do software entregam na ponta, ou seja, quais são as saídas geradas.

Relembrando: A técnica de **caixa preta (funcional)** testa pré-condições avaliando o comportamento das funcionalidades conforme os requisitos **sem se preocupar com a lógica interna**.

Certo.

015. (CESPE/ANAC/ANALISTA ADMINISTRATIVO/TECNOLOGIA DA INFORMAÇÃO/2012) Os testes funcionais são caracterizados pelo uso do sistema conforme o seu usuário regular o faria.



Isso não faz o menor sentido quando estamos tratando de testes funcionais ou testes caixa preta, lembrando:

A técnica de **caixa preta (funcional)** testa pré-condições avaliando o comportamento das funcionalidades conforme os requisitos **sem se preocupar com a lógica interna**.

Não existe a necessidade de simular como o usuário faria, existe sim a necessidade de avaliar como as funcionalidades deveriam fazer.

Errado.

016. (CESPE/ANATEL/ARQUITETURA DE SOLUÇÕES/2012) Considere as informações abaixo em relação ao desenvolvimento de sistemas:

I – Executar um software com o objetivo de revelar falhas, mas que não prova a exatidão do software.

II – Correta construção do produto.

III – Construção do produto certo.

Correspondem corretamente a I, II e III, respectivamente,

- a) Validação, verificação e teste.
- b) Verificação, teste e validação.
- c) Teste, verificação e validação.
- d) Validação, teste e verificação.
- e) Teste, validação e verificação.



Quando um software é executado com o objetivo de revelar falhas (comportamento não esperado) faz-se um **teste**, que obviamente não prova a exatidão do software como um todo.

Verificação é analisar se a forma com a qual estamos construindo um produto (software) é a correta, e **validação** é analisar se o produto (software) é o correto. As perguntas que diferenciam verificação de validação, são:

- Verificação: **Será que estamos construindo o produto corretamente.**
- Validação: **Será que estamos construindo o produto correto.**

Assim tem-se:

Teste	I – Executar um software com o objetivo de revelar falhas, mas que não prova a exatidão do software.
Verificação	II – Correta construção do produto.
Validação	III – Construção do produto certo.

Letra c.

017. (FCC/TRT 14 REGIÃO/TÉCNICO JUDICIÁRIO – PROGRAMAÇÃO/2011) Garantir o funcionamento correto do software para atender as expectativas do cliente é o objetivo da homologação de sistemas. Nessa fase, que precede à implantação, os testes mais comuns são os testes:

- a) Funcionais, de usabilidade e de aceitação.
- b) De unidade, de iteração e de integração.
- c) De volume, de integridade e de aceitação.
- d) Da caixa-branca, de carga e de configuração.
- e) De unidade, de carga e de integridade.



Para avaliar as expectativas do cliente vamos utilizar, na fase de homologação, os testes **funcionais** (analisamos se o que está saindo na ponta é de fato o que deveria estar saindo), teste de **usabilidade** (verificamos a utilização do software pelo usuário) e por fim o teste de **aceitação** (em que o usuário valida se o que foi apresentado enquanto software está em conformidade com os requisitos).

Letra a.

REFERÊNCIAS

- KOSCIANSKI, A.; SOARES, M. S. **Qualidade de Software**. São Paulo: Editora Novatec, 2007.
- KRUCHTEN, P., **Introdução ao RUP Rational Unified Process**, 2. ed., Rio de Janeiro: Ciência Moderna, 2003.
- PRESSMAN, R. S. **Engenharia de Software: Uma Abordagem Profissional**, 7. ed. Porto Alegre: Editora Mc GrawHill, 2011.
- PFLEEGER, Shari Lawrence. **Engenharia de Software: Teoria e Prática**. 2.ed., São Paulo: Prentice Hall, 2004.
- QUINTÃO, P. L. **Notas de aula da disciplina “Tecnologia da Informação”**. 2023.
- SOMMERVILLE, I., **Engenharia de Software**, 8. ed., São Paulo: Pearson Addison – Wesley, 2007.
- _____. **Engenharia de Software**, 9. ed., São Paulo: Pearson Addison – Wesley, 2011.
- CÉSAR, B. **Modelos de desenvolvimento de software: a Catedral e o Bazar**. Disponível em: <<http://www.brod.com.br/node/536>>. Acesso em: 11 jul. 2021.
- IBM. Disponível em: <https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/evolucao_do_open_source?lang=en>. Acesso em: 11 jul. 2021.
- IBM. **Teste de Aceitação**. 1987. Disponível em: <https://www.cin.ufpe.br/~gta/rup-vc/core.base_rup/guidances/concepts/acceptance_testing_12A0F152.html>. Acesso em: dez. de 2022.

Patrícia Quintão



Mestre em Engenharia de Sistemas e computação pela COPPE/UFRJ, Especialista em Gerência de Informática e Bacharel em Informática pela UFV. Atualmente é professora no Gran Cursos Online; Analista Legislativo (Área de Governança de TI), na Assembleia Legislativa de MG; Escritora e Personal & Professional Coach.

Atua como professora de Cursinhos e Faculdades, na área de Tecnologia da Informação, desde 2008. É membro: da Sociedade Brasileira de Coaching, do PMI, da ISACA, da Comissão de Estudo de Técnicas de Segurança (CE-21:027.00) da ABNT, responsável pela elaboração das normas brasileiras sobre gestão da Segurança da Informação.

Autora dos livros: Informática FCC - Questões comentadas e organizadas por assunto, 3ª. edição e 1001 questões comentadas de informática (Cespe/UnB), 2ª. edição, pela Editora Gen/Método.

Foi aprovada nos seguintes concursos: Analista Legislativo, na especialidade de Administração de Rede, na Assembleia Legislativa do Estado de MG; Professora titular do Departamento de Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia; Professora substituta do DCC da UFJF; Analista de TI/Suporte, PRODABEL; Analista do Ministério Público MG; Analista de Sistemas, DATAPREV, Segurança da Informação; Analista de Sistemas, INFRAERO; Analista - TIC, PRODEMGE; Analista de Sistemas, Prefeitura de Juiz de Fora; Analista de Sistemas, SERPRO; Analista Judiciário (Informática), TRF 2ª Região RJ/ES, etc.

@coachpatriciaquintao

/profapatriciaquintao

@plquintao

t.me/coachpatriciaquintao

**NÃO SE ESQUEÇA DE
AVALIAR ESTA AULA!**

**SUA OPINIÃO É MUITO IMPORTANTE
PARA MELHORARMOS AINDA MAIS
NOSSOS MATERIAIS.**

**ESPERAMOS QUE TENHA GOSTADO
DESTA AULA!**

**PARA AVALIAR, BASTA CLICAR EM LER
A AULA E, DEPOIS, EM AVALIAR AULA.**

AVALIAR 