Estratégia Concursos

Aula 00 - Prof. Diego Carvalho e Fernando Pedrosa

CAGEPA (Analista de Sistemas - Sistemas de TI) Engenharia de software - 2024 (Pós-Edital)

Autor:

Diego Carvalho, Renato da Costa

10 de Novembro de 2024

Índice

APRESENTAÇÃO DO PROFESSOR

PROF. DIEGO CARVALHO

FORMADO EM CIÊNCIA DA COMPUTAÇÃO PELA UNIVERSIDADE DE BRASÍLIA (UNB), PÓS-GRADUADO EM GESTÃO DE TECNOLOGIA DA INFORMAÇÃO NA ADMINISTRAÇÃO PÚBLICA E, ATUALMENTE, AUDITOR FEDERAL DE FINANÇAS E CONTROLE DA SECRETARIA DO TESOURO NACIONAL.

ESTRATÉGIA CONCURSOS

PROFESSOR DIEGO CARVALHO - WWW.INSTAGRAM.COM/PROFESSORDIEGOCARVALHO

Sobre o curso: galera, todos os tópicos da aula possuem Faixas de Incidência, que indicam se o assunto cai muito ou pouco em prova. Diego, se cai pouco para que colocar em aula? Cair pouco não significa que não cairá justamente na sua prova! A ideia aqui é: se você está com pouco tempo e precisa ver somente aquilo que cai mais, você pode filtrar pelas incidências média, alta e altíssima; se você tem tempo sobrando e quer ver tudo, vejam também as incidências baixas e baixíssimas. *Fechado?*

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

INCIDÊNCIA EM PROVA: BAIXA

INCIDÊNCIA EM PROVA: MÉDIA

INCIDÊNCIA EM PROVA: ALTA

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Além disso, essas faixas não são por banca – é baseado tanto na quantidade de vezes que caiu em prova independentemente da banca quanto nas minhas próprias avaliações sobre cada assunto.

#ATENÇÃO

Avisos Importantes

O curso abrange todos os níveis de conhecimento...

Esse curso foi desenvolvido para ser acessível a **alunos com diversos níveis de conhecimento diferentes.** Temos alunos mais avançados que têm conhecimento prévio ou têm facilidade com o assunto. Por outro lado, temos alunos iniciantes, que nunca tiveram contato com a matéria ou até mesmo que têm trauma dessa disciplina. A ideia aqui é tentar atingir ambos os públicos - iniciantes e avançados - da melhor maneira possível..

Por que estou enfatizando isso?

O **material completo** é composto de muitas histórias pessoais, exemplos, metáforas, piadas, memes, questões, desafios, esquemas, diagramas, imagens, entre outros. Já o **material simplificado** possui exatamente o mesmo núcleo do material completo, mas ele é menor e mais objetivo. *Professor, eu devo estudar por qual material?* Se você quiser se aprofundar nos assuntos ou tem dificuldade com a matéria, necessitando de um material mais passo-a-passo, utilize o material completo. Se você não quer se aprofundar nos assuntos ou tem facilidade com a matéria, necessitando de um material mais direto ao ponto, utilize o material simplificado.

Por fim...

O curso contém diversas questões espalhadas em meio à teoria. Essas questões possuem um comentário mais simplificado porque **têm o único objetivo de apresentar ao aluno como bancas de concurso cobram o assunto previamente administrado**. A imensa maioria das questões para que o aluno avalie seus conhecimentos sobre a matéria estão dispostas ao final da aula na lista de exercícios e **possuem comentários bem mais abrangentes**.

TEST DRIVEN DEVELOPMENT (TDD)

Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

O Test-Driven Development (TDD) é uma abordagem de desenvolvimento de software em que se intercalam testes e desenvolvimento de código. Essencialmente, você desenvolve um código de forma incremental, em conjunto com um teste para esse incremento. Você não caminha para o próximo incremento até que o código desenvolvido passe no teste. O desenvolvimento dirigido a testes foi apresentado como parte dos métodos ágeis, como o XP.

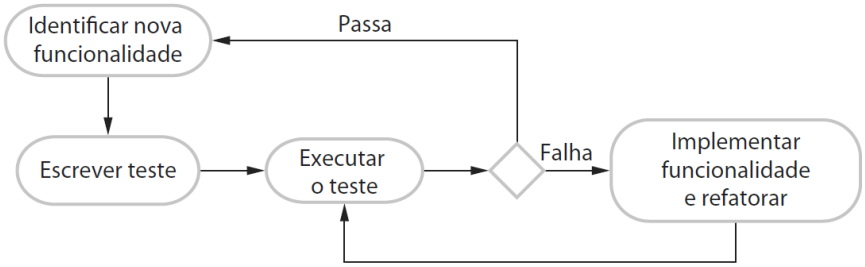
Por outro lado, ele também pode ser utilizado em processos de desenvolvimento dirigido a planos. Trata-se de uma abordagem que **se baseia na repetição de um ciclo de desenvolvimento curto focado em testes unitários**. A ideia fundamental dessa abordagem consiste em escrever o teste, encontrar uma falha nesse mesmo teste e depois refatorá-lo. Vamos agora ver as etapas do processo de desenvolvimento dirigido a testes:

ETAPA	DESCRIÇÃO
1	Você começa identificando o incremento de funcionalidade necessário. Este, normalmente, deve ser pequeno e implementável em poucas linhas de código.
2	Você escreve um teste para essa funcionalidade e o implementa como um teste automatizado. Isso significa que o teste pode ser executado e relatará se passou ou falhou.
3	Você, então, executa o teste, junto com todos os outros testes implementados. Inicialmente, você não terá implementado a funcionalidade, logo o novo teste falhará. Isso é proposital, pois mostra que o teste acrescenta algo ao conjunto de testes.
4	Você, então, implementa a funcionalidade e executa novamente o teste. Isso pode envolver a refatoração do código existente para melhorá-lo e adicionar um novo código sobre o que já está lá.
5	Depois que todos os testes forem executados com sucesso, você caminha para implementar a próxima parte da funcionalidade.

Como o código é desenvolvido em incrementos muito pequenos, você precisa ser capaz de executar todos os testes cada vez que adicionar funcionalidade ou refatorar o programa. Dessa forma, os testes são embutidos em um programa separado que os executa e invoca o sistema que está sendo testado. Usando essa abordagem, é possível rodar centenas e centenas de testes separados em poucos segundos.

Um argumento forte a favor do desenvolvimento dirigido a testes é que ele ajuda os programadores a clarear suas ideias sobre o que um segmento de código supostamente deve fazer**. Para escrever um teste, você precisa entender a que ele se destina, e como esse entendimento faz que seja**

mais fácil escrever o código necessário. Certamente, se você tem conhecimento ou compreensão incompleta, o desenvolvimento dirigido a testes não ajudará.



Se você não sabe o suficiente para escrever os testes, não vai desenvolver o código necessário. Por exemplo: se seu cálculo envolve divisão, você deve verificar se não está dividindo o número por zero. Se você se esquecer de escrever um teste para isso, então o código para essa verificação nunca será incluído no programa. Além de um melhor entendimento do problema, outros benefícios do desenvolvimento dirigido a testes são:

BENEFÍCIOS	DESCRIÇÃO
COBERTURA DE CÓDIGO	Em princípio, todo segmento de código que você escreve deve ter pelo menos um teste associado. Assim, você pode ter certeza de que todo o código no sistema foi realmente executado. Cada código é testado enquanto está sendo escrito; assim, os defeitos são descobertos no início do processo de desenvolvimento.
TESTE DE REGRESSÃO	Um conjunto de testes é desenvolvido de forma incremental enquanto um programa é desenvolvido. Você sempre pode executar testes de regressão para verificar se as mudanças no programa não introduziram novos bugs.
DEPURAÇÃO SIMPLIFICADA	Quando um teste falha, a localização do problema deve ser óbvia. O código recém-escrito precisa ser verificado e modificado. Você não precisa usar as ferramentas de depuração para localizar o problema. Alguns relatos de uso de desenvolvimento dirigido a testes sugerem que, em desenvolvimento dirigido a testes, quase nunca é necessário usar um sistema automatizado de depuração.
DOCUMENTAÇÃO DE SISTEMA	Os testes em si mesmos agem como uma forma de documentação que descreve o que o código deve estar fazendo. Ler os testes pode tornar mais fácil a compreensão do código.

Um dos benefícios mais importantes de desenvolvimento dirigido a testes é que ele reduz os custos dos testes de regressão. O teste de regressão envolve a execução de conjuntos de testes que tenham sido executados com sucesso, após as alterações serem feitas em um sistema. Ele também verifica se essas mudanças não introduziram novos bugs no sistema e se o novo código interage com o código existente conforme o esperado.

O teste de regressão é muito caro e geralmente impraticável quando um sistema é testado manualmente, pois os custos com tempo e esforço são muito altos. Em tais situações, você

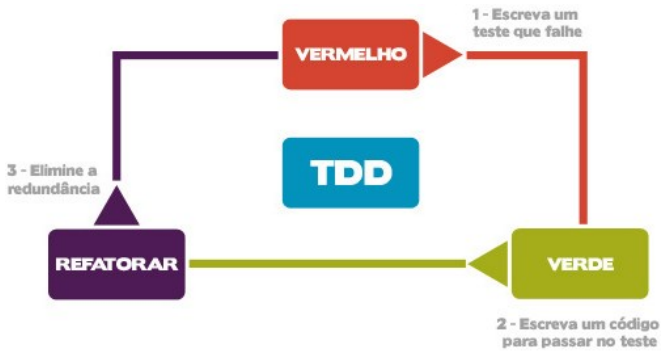
precisa tentar escolher os testes mais relevantes para executar novamente, e é fácil perder testes importantes. No entanto, testes automatizados – fundamentais para o desenvolvimento *test-first* – reduzem drasticamente os custos com testes de regressão.

Os testes existentes podem ser executados novamente de forma rápida e barata. **Após se fazer uma mudança para um sistema em desenvolvimento *test-first*, todos os testes existentes devem ser executados com êxito antes de qualquer funcionalidade ser adicionada.** Como um programador, você precisa ter certeza de que a nova funcionalidade não tenha causado ou revelado problemas com o código existente.

O desenvolvimento dirigido a testes é de maior utilidade no desenvolvimento de softwares novos, em que a funcionalidade seja implementada no novo código ou usando bibliotecas-padrão já testadas. **Se você estiver reusando componentes de código ou sistemas legados grandes, você precisa escrever testes para esses sistemas como um todo.** O desenvolvimento dirigido a testes também pode ser ineficaz em sistemas multi-threaded.

As *threads* diferentes podem ser intercalados em tempos diferentes, em execuções diferentes, e isso pode produzir resultados diferentes. Se você usa o desenvolvimento dirigido a testes, ainda precisa de um processo de teste de sistema para validar o sistema, isto é, verificar se atende aos requisitos dos stakeholders. O teste de sistema também testa o desempenho, a confiabilidade, e verifica se o sistema não faz coisas que não deveria, como produzir resultados indesejados etc.

O desenvolvimento dirigido a testes revelou-se uma abordagem de sucesso para projetos de pequenas e médias empresas. Geralmente, os programadores que adotaram essa abordagem estão satisfeitos com ela e acham que é uma maneira mais produtiva de desenvolver softwares. Em alguns experimentos, foi mostrado que essa abordagem gera melhorias na qualidade do código. Bem, agora vamos falar sobre o ciclo vermelho, verde e refatoração.



ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.

REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

Vamos lá! Nós sabemos que – para cada parte da aplicação – adiciona-se um teste escrito antes mesmo do desenvolvimento do código em si. *Por que?* **Porque eles podem ajudar a reduzir riscos de possíveis problemas no código.** Executamos o teste e ele... falha! Ele deve necessariamente falhar! *Por que?* Ora, porque ele é o primeiro teste e você nem criou a funcionalidade ainda, logo ele não irá funcionar!

Então nós adicionamos uma nova funcionalidade ao sistema apenas para que ele passe no teste e execute novamente (agora ele deve passar no teste). **Então, nós adicionamos um novo teste e rodamos o teste anterior e esse novo teste.** Se algum deles falhar, modifica-se o código da funcionalidade e rodam-se todos os testes novamente, e assim por diante – nós já vimos aquela imagem anterior que mostra como tudo isso funciona...

Galera, vocês percebem que o feedback sobre a nova funcionalidade ocorre de maneira bem rápido? **Além disso, cria-se um código mais limpo, visto que o código para passar nos testes deve ser bastante simples.** Há mais segurança na correção de eventuais bugs; aumenta-se a produtividade, visto que se perde menos tempo com depuradores; e o código se torna mais flexível, menos acoplado e mais coeso.

Nós podemos afirmar que, em geral, utilizam-se testes unitários, testes de integração ou testes de aceitação – sendo os dois primeiros os mais comuns. **Algumas ferramentas que podem ser utilizadas para implementar o processo de desenvolvimento orientado a testes:** JUnit, TesteNG, PHPUnit, SimpleTest, NUnit, Jasmine, CUnit, PyUnit, etc. Pessoal, agora um detalhe que nós passamos direto sobre a abordagem de desenvolvimento...

O TEST DRIVEN DEVELOPMENT (TDD) NÃO É UMA ABORDAGEM PARA REALIZAR TESTES - TRATA-SE DE UMA ABORDAGEM PARA DESENVOLVER SOFTWARES. ELA PODE EVENTUALMENTE SER CONSIDERADA TAMBÉM UMA TÉCNICA DE PROGRAMAÇÃO!

Professor, uma curiosidade: isso já caiu em alguma prova discursiva? **Sim, galera... o enunciado dessa prova requisitava ao aluno informar as vantagens do emprego do TDD em relação a outras metodologias ágeis.** Poderíamos responder essa pergunta afirmando que o software desenvolvido, em geral, apresenta maior qualidade, na medida em que é implementado direcionado às expectativas do cliente.

Poderíamos dizer também que há a possibilidade de se testar todo o código desenvolvido, o que oferece maior confiabilidade ao sistema. Por fim, em geral, o código é mais modularizado, flexível e extensível, visto que a metodologia requer que os desenvolvedores imaginem o software como pequenas unidades[1] que podem ser reescritas, desenvolvidas e testadas de forma independente e integradas em momento posterior.

Essa mesma prova perguntava também quais são os princípios da Metodologia Extreme Programming (XP) apoiados pelo TDD. Uma resposta adequada poderia afirmar que o XP apresenta diversas práticas que podem ser relacionadas com o TDD. *Qual é a mais óbvia?* **A mais óbvia é o *Test-First* (Teste Primeiro), ratificando a característica básica recomendada veementemente pelo desenvolvimento orientado a testes.**

O TDD pode apoiar esse princípio por fornecer detalhes para a realização dos testes de unidade e de funcionalidade, que são importantes e necessários. Ademais, o desenvolvimento orientado a testes apresenta relação intrínseca com a refatoração, tendo em vista que confere ao programador maior segurança para identificar e remover o código duplicado, e permite, assim, a melhoria contínua do programa. Vamos resumir as principais características do TDD:

CARACTERÍSTICAS DO TDD	DESCRIÇÃO
ORIENTADO A TESTES	O TDD coloca o teste no centro do processo de desenvolvimento, guiando a escrita do código. Cada nova funcionalidade é conduzida por um ou mais testes que definem o comportamento esperado.
PEQUENOS PASSOS	TDD promove a implementação em pequenos passos iterativos. Cada ciclo de desenvolvimento é curto, envolvendo a escrita de um pequeno teste, o código correspondente e a refatoração.
ALTA COBERTURA	Como os testes são escritos antes do código, TDD naturalmente resulta em uma alta cobertura de testes automatizados, o que ajuda a detectar regressões e garantir que o código funcione conforme esperado.
FEEDBACK IMEDIATO	O TDD oferece feedback imediato sobre a correção do código. Se um teste falha, o desenvolvedor pode corrigir o problema imediatamente, minimizando o tempo entre a introdução de um erro e sua correção.
DOCUMENTAÇÃO VIVA	Os testes servem como documentação executável do comportamento do sistema. Eles descrevem o que o código deve fazer de forma precisa e concisa, e essa "documentação" é sempre atualizada porque os testes são executados continuamente.
FACILITA REFATORAÇÃO SEGURA	A prática de TDD promove um design de código mais limpo e modular, já que o desenvolvedor refatora o código regularmente com a segurança de que os testes irão capturar qualquer erro introduzido.
PROJETO GUIADO POR TESTES	TDD influencia positivamente o design do software, encorajando a criação de código coeso e de baixo acoplamento. Escrever testes primeiro ajuda os desenvolvedores a pensar na interface e na responsabilidade das classes antes de implementá-las.

[1] Atenção: apesar de serem tipicamente realizados testes de unidade, não é obrigatória a utilização desse tipo de teste.

MAIOR QUALIDADE DE CÓDIGO	A prática constante de escrever testes e refatorar resulta em um código de maior qualidade, com menos bugs e mais fácil de manter e evoluir.
REDUÇÃO DE BUGS DE PRODUÇÃO	Com o TDD, muitos erros são capturados e corrigidos durante o desenvolvimento, o que resulta em uma menor probabilidade de bugs aparecerem em produção.
MELHORIA NA CONFIANÇA	Como os testes automatizados garantem que o código funciona conforme esperado, os desenvolvedores têm mais confiança em modificar, refatorar ou estender o código sem introduzir novos erros.
AUMENTO DA MANUTENI-BILIDADE	O código desenvolvido com TDD tende a ser mais modular e de fácil manutenção, uma vez que os testes forçam o desenvolvedor a pensar em como dividir o código em partes bem definidas e testáveis.
ASSERÇÕES DE VERIFICAÇÃO	Cada caso de teste deve incluir asserções que verifiquem se o comportamento do código está conforme o esperado. As asserções são fundamentais para garantir que o teste valide corretamente o comportamento do código.
TESTES DE INTEGRAÇÃO	Testes de integração são utilizados para validar recursos complexos, como acesso a dados, onde múltiplos componentes ou sistemas interagem. Esses testes garantem que as partes integradas funcionem corretamente juntas.
INDEPENDÊNCIA DOS TESTES	A independência dos testes é um princípio fundamental. Cada teste deve ser executado isoladamente, sem dependências entre eles. Comentários sobre quais testes foram criados não substituem a necessidade de garantir que os testes não se afetam mutuamente.

Galera, nós temos também uma variação chamada Acceptance Test-Driven Development (ATDD). **Ele é método ágil de desenvolvimento de software que se baseia na comunicação entre clientes do negócio, desenvolvedores e testadores.** Diferente do Test-Driven Development (TDD), que se concentra em testes unitários, o ATDD se concentra em testes de aceitação que validam se o sistema atende aos critérios de aceitação definidos pelo cliente ou pelo usuário final.

Ele promove uma estreita colaboração entre todas as partes interessadas (stakeholders). Isso inclui desenvolvedores, testadores, analistas de negócios e clientes que trabalham juntos para definir e escrever os testes de aceitação antes que o desenvolvimento comece. Essa colaboração garante que todos tenham um entendimento comum do que precisa ser construído. Os testes de aceitação servem como uma forma de documentação viva e executável.

Eles descrevem o comportamento esperado do sistema e podem ser usados para verificar automaticamente se o software atende aos requisitos acordados. Além disso, são diretamente derivados dos requisitos funcionais e não funcionais do sistema. Isso garante que o software entregue esteja em conformidade com as especificações e expectativas do cliente. Outra característica importante é o seu feedback imediato.

Como os testes são escritos antes do desenvolvimento, o ATDD oferece feedback imediato sobre a conformidade do software com os requisitos do cliente durante o processo de desenvolvimento. Isso permite que problemas e mal-entendidos sejam identificados e corrigidos mais cedo. Ao validar os critérios de aceitação antes de começar a codificar, o ATDD ajuda a reduzir a quantidade de retrabalho causado por mal-entendidos sobre os requisitos.

Embora similar ao TDD, o ciclo do ATDD adapta o ciclo Red-Green-Refactor para se concentrar em testes de aceitação. O ciclo pode ser descrito como:

ETAPA	DESCRIÇÃO
RED	Escrever um teste de aceitação que inicialmente falha, porque a funcionalidade ainda não foi implementada.
GREEN	Implementar o código necessário para fazer o teste passar.
REFACTOR	Melhorar o código enquanto se certifica de que todos os testes de aceitação continuam a passar.

O uso do ATDD resulta em um software de maior qualidade, pois os requisitos do cliente são continuamente validados ao longo do processo de desenvolvimento. A confiança na qualidade do software também é aumentada, pois os testes de aceitação garantem que o sistema se comporta conforme o esperado. A tabela exibida a seguir apresenta as principais características do ATDD e a tabela seguinte nos mostra as diferenças para o TDD:

CARACTERÍSTICAS DO ATDD	DESCRIÇÃO
FOCO EM TESTES DE ACEITAÇÃO	ATDD envolve a escrita de testes de aceitação antes do desenvolvimento, garantindo que os critérios do cliente sejam atendidos.
COLABORAÇÃO ENTRE EQUIPES	Promove a colaboração entre desenvolvedores, testadores e stakeholders para definir os critérios de aceitação.
TESTES BASEADOS EM REQUISITOS	Os testes de aceitação são diretamente derivados dos requisitos funcionais, garantindo que o software entregue esteja alinhado com as expectativas do cliente.
DOCUMENTAÇÃO EXECUTÁVEL	Os testes de aceitação servem como documentação viva e executável, descrevendo o comportamento esperado do sistema.
FEEDBACK IMEDIATO	Oferece feedback imediato sobre a conformidade do software com os requisitos do cliente durante o desenvolvimento.
CICLO ADAPTADO	Similar ao TDD, mas com foco em garantir que o software atenda aos critérios de aceitação antes de refatorar o código.
ENGAJAMENTO DE STAKEHOLDERS	Incentiva o envolvimento dos stakeholders no processo de desenvolvimento, garantindo que o produto final atenda às suas necessidades.
QUALIDADE DO PRODUTO	Ao focar nos testes de aceitação desde o início, ATDD ajuda a detectar problemas cedo, resultando em um produto de maior qualidade.
REDUÇÃO DE RETRABALHO	Ao validar os critérios de aceitação antes do desenvolvimento, minimiza a necessidade de retrabalho causado por mal-entendidos nos requisitos.
VALIDAÇÃO DE REQUISITOS	Ajuda a validar e clarificar os requisitos antes de iniciar o desenvolvimento, garantindo que todos tenham o mesmo entendimento sobre as expectativas.

TDD (TEST-DRIVEN DEVELOPMENT)	ATDD (ACCEPTANCE TEST-DRIVEN DEVELOPMENT)
O foco principal do TDD está nos testes unitários. O desenvolvedor escreve pequenos testes automatizados antes de escrever o código correspondente. O objetivo é garantir que cada unidade de código (geralmente uma função ou método) funcione corretamente em um nível granular.	O ATDD se concentra nos testes de aceitação, que validam o comportamento do sistema em um nível mais alto, assegurando que o software atende aos requisitos e critérios de aceitação definidos pelos stakeholders, incluindo clientes, analistas de negócios e desenvolvedores.
Os testes em TDD são geralmente limitados a uma unidade de código individual, como um método ou classe. Eles são escritos pelos desenvolvedores para verificar a correção funcional de pequenas partes do sistema.	Os testes de ATDD abrangem o sistema como um todo ou grandes partes dele, focando no comportamento e na interação entre os componentes do sistema. Esses testes validam se a aplicação atende às necessidades e expectativas do usuário final.
A prática de TDD é geralmente realizada exclusivamente pelos desenvolvedores. O ciclo de escrita de testes, codificação e refatoração é iterado pelos programadores durante o desenvolvimento.	ATDD envolve uma colaboração mais ampla entre desenvolvedores, testadores, analistas de negócios e clientes. Todos trabalham juntos para definir os critérios de aceitação e os testes que garantem que o sistema está alinhado com as expectativas do cliente.
O objetivo do TDD é criar um código que funcione corretamente desde o nível mais baixo, permitindo um design de software mais limpo, modular e testável. Ele também visa capturar erros no nível mais granular possível.	O objetivo do ATDD é garantir que o sistema como um todo funcione conforme esperado pelos usuários finais e que ele atenda aos requisitos de negócios. ATDD valida que o software satisfaz as necessidades e critérios de aceitação antes de ser considerado "pronto".
Os testes criados durante o TDD atuam como uma forma de documentação técnica do comportamento das unidades de código. Eles ajudam outros desenvolvedores a entender como o código foi projetado para funcionar.	Os testes de aceitação no ATDD servem como documentação executável do sistema. Eles descrevem o comportamento esperado do sistema de uma maneira que é compreensível tanto para técnicos quanto para stakeholders não técnicos.
O ciclo de TDD (Red-Green-Refactor) é rápido e focado, proporcionando feedback imediato sobre pequenas porções de código. Isso ajuda a manter o desenvolvimento alinhado com o design desejado desde o início.	O ATDD também oferece feedback, mas em um nível mais alto, garantindo que o software em desenvolvimento está de acordo com os requisitos do usuário. O feedback no ATDD é obtido antes mesmo do início do desenvolvimento real, durante a fase de especificação dos requisitos.
Os testes são escritos pelos desenvolvedores, que também são responsáveis por escrever o código para passar esses testes.	Os testes são definidos em colaboração entre desenvolvedores, testadores e stakeholders (como clientes ou analistas de negócios), e podem ser escritos por desenvolvedores ou testadores.

Em suma: TDD é focado em garantir que as pequenas partes do código (unidades) funcionem corretamente, através da escrita de testes unitários antes do desenvolvimento do código correspondente. Ele é mais técnico e orientado para a implementação. ATDD é focado em garantir que o software como um todo atende aos requisitos do cliente, através da escrita de testes de aceitação antes do desenvolvimento. Ele é mais colaborativo e orientado para o negócio.

QUESTÕES COMENTADAS – DIVERSAS BANCAS

1. **(CESGRANRIO / IPEA – 2024)** Uma gerente de testes de software propôs a seu time de desenvolvimento que começasse a aplicar a abordagem Test Driven Development (TDD). É uma das características principais dessa abordagem iniciar o desenvolvimento de testes:

   a) antes de implementar alguma funcionalidade em si.
   b) durante o período de homologação.
   c) após as funcionalidades serem construídas.
   d) quando a primeira leva de funcionalidades planejadas forem codificadas em algum sprint.
   e) pelos testes de interface automatizado, seguidos pelos testes unitários.

Comentários:

(a) Correto. No Test Driven Development (TDD), os testes são escritos antes da implementação das funcionalidades. Isso garante que o código seja desenvolvido para passar nos testes desde o início;

(b) Errado. O período de homologação ocorre após a implementação e testes iniciais, e não é o foco do TDD, que deve iniciar antes da implementação;

(c) Errado. Escrever testes após a construção das funcionalidades é a abordagem tradicional de desenvolvimento, não a abordagem TDD;

(d) Errado. No TDD, os testes são criados antes de qualquer funcionalidade ser codificada, não após uma leva de funcionalidades;

(e) Errado. A abordagem TDD prioriza a escrita de testes unitários antes da implementação das funcionalidades, não necessariamente começando pelos testes de interface automatizados.

Gabarito: Letra A

---

2. **(CESPE / INPI – 2024)** O desenvolvimento dirigido por testes (TDD) é modelado em três estados: vermelho, verde e refatorar. Um exemplo da ação de refatoração é a simulação do comportamento dos componentes que interagem com a unidade de teste que está falhando.

Comentários:

No desenvolvimento dirigido por testes (TDD), os três estados são: vermelho (escrever um teste que falha), verde (escrever o código mínimo necessário para fazer o teste passar) e refatorar (melhorar o código mantendo os testes passando). A simulação do comportamento dos componentes que interagem com a unidade de teste que está falhando é uma prática de criação de

mocks ou stubs, usada para isolar a unidade de teste, e não um exemplo de refatoração. Refatoração envolve melhorar a estrutura do código sem alterar seu comportamento externo.

Gabarito: Errado

---

3. **(CESPE / CNPq – 2024)** O TDD é uma tendência que enfatiza o projeto de casos de teste antes da criação do código fonte e se caracteriza como parte do modelo ágil de desenvolvimento de software.

Comentários:

Test-Driven Development (TDD) é uma prática que enfatiza a criação de casos de teste antes do desenvolvimento do código-fonte. É uma abordagem iterativa que faz parte das metodologias ágeis de desenvolvimento de software, promovendo um ciclo de feedback rápido e garantindo que o código seja continuamente testado e refinado.

Gabarito: Correto

---

4. **(CESPE / CNPq – 2024)** No TDD, o teste deve ser criado com o objetivo de fazer o segmento de código falhar, gerando-se um processo iterativo que permite a submissão de muitas subfunções simultaneamente, o que confere uma agilidade significativa ao processo.

Comentários:

Test-Driven Development (TDD) é uma prática que enfatiza a criação de casos de teste antes do desenvolvimento do código-fonte. É uma abordagem iterativa que faz parte das metodologias ágeis de desenvolvimento de software, promovendo um ciclo de feedback rápido e garantindo que o código seja continuamente testado e refinado.

Gabarito: Errado

---

5. **(CESPE / AGER-MT – 2023)** Assinale a opção que corresponde ao método de teste de software adotado, sob a perspectiva do desenvolvedor, a partir de casos de teste do código, escritos em linguagem técnica, para testar as funcionalidades antes da implementação da solução desenvolvida:

a) unit testing
b) TDD
c) BDD
d) ATDD
e) teste de caixa preta

Comentários:

(a) Errado. Unit testing (Teste de Unidade) é o teste de unidades individuais do código (como funções ou métodos), mas não necessariamente envolve escrever testes antes da implementação da funcionalidade.

(b) Correto. TDD (Test-Driven Development) é uma prática de desenvolvimento onde os desenvolvedores escrevem casos de teste antes de implementar a funcionalidade correspondente. Esses testes são escritos em linguagem técnica e guiam a implementação do código.

(c) Errado. BDD (Behavior-Driven Development) é uma abordagem de desenvolvimento que estende o TDD, focando na colaboração entre desenvolvedores, QA e não técnicos, utilizando uma linguagem natural para descrever o comportamento esperado.

(d) Errado. ATDD (Acceptance Test-Driven Development) é semelhante ao TDD, mas envolve stakeholders na criação dos testes de aceitação antes do desenvolvimento, focando na validação do comportamento do sistema do ponto de vista do usuário.

(e) Errado. Teste de caixa preta (Blackbox Testing) é uma técnica de teste que verifica a funcionalidade do software sem se preocupar com a estrutura interna do código, focando apenas nas entradas e saídas.

Gabarito: Letra B

---

6. **(CESPE / TC-DF – 2023)** Na etapa de refactor do processo TDD, parte-se do pressuposto de que os testes tenham passado nas fases anteriores, o que permite que o código seja aprimorado sem a preocupação de duplicações de código.

Comentários:

A questão é polêmica, mas a banda justificou o gabarito:

"É neste momento que retiramos duplicidade, renomeamos variáveis, extraímos métodos, extraímos classes, extraímos interfaces, usamos algum padrão conhecido, etc. É neste momento que podemos deixar o nosso código simples e claro e o melhor de tudo: Funcional. Temos um teste que indicará qualquer passo errado que podemos dar ao melhorar o código"o.

Gabarito: Errado

---

7. **(CESPE / EMPREL – 2023)** Ao adotar uma prática ágil para a criação de um software, seu desenvolvedor optou pela implementação com qualidade de uma funcionalidade do sistema; para isso, escreveu um caso de teste automatizado, com base nos requisitos especificados, e realizou testes de unidade em uma linguagem similar à usada no desenvolvimento da funcionalidade. Da situação hipotética precedente infere-se que a prática adotada pelo desenvolvedor está associada ao:

a) desenvolvimento orientado por comportamento (BDD).
b) gerenciamento de produtos com Scrum.
c) desenvolvimento guiado por testes (TDD).
d) desenvolvimento guiado por testes de aceitação (ATDD).
e) gerenciamento de produtos com Kanban.

Comentários:

(a) Errado. O Desenvolvimento Orientado por Comportamento (BDD) foca em descrever o comportamento esperado do software em uma linguagem que todos os stakeholders possam entender, geralmente com cenários de aceitação claros e exemplos. A questão não menciona o uso de cenários compreensíveis por não técnicos;

(b) Errado. O Gerenciamento de Produtos com Scrum é uma metodologia ágil para gerenciamento de projetos que utiliza sprints e eventos do Scrum, como reuniões diárias e retrospectivas. Não foca especificamente na criação de testes automatizados antes do desenvolvimento;

(c) Correto. O Desenvolvimento Guiado por Testes (TDD) é uma prática de desenvolvimento ágil onde os desenvolvedores escrevem testes de unidade antes da implementação do código funcional. O processo inclui a escrita de um teste que inicialmente falha, a implementação do código para passar o teste e a refatoração do código, se necessário. Isso está alinhado com a descrição fornecida;

(d) Errado. O Desenvolvimento Guiado por Testes de Aceitação (ATDD) envolve escrever testes de aceitação com base em requisitos, geralmente com a participação de clientes e stakeholders. Foca em garantir que a funcionalidade atenda aos requisitos de aceitação. A questão menciona testes de unidade, não de aceitação;

e) Errado: O Gerenciamento de Produtos com Kanban é uma abordagem ágil que visa melhorar o fluxo de trabalho visualizando e gerenciando tarefas através de um quadro Kanban. Não envolve diretamente a prática de escrita de testes automatizados antes da implementação.

Gabarito: Letra C

---

8. **(FGV / SEFAZ-MG – 2023)** Você entrou para um projeto novo, já em andamento, no qual a metodologia que a equipe do projeto segue é a de definir e escrever testes de software a partir das regras de negócio antes mesmo de implementar as funcionalidades propostas. Assinale a opção que indica o nome desse processo de desenvolvimento de software:

a) DDD
b) TDD
c) BDD
d) XP

e) Scrum

Comentários:

(a) Errado. DDD (Domain-Driven Design) é uma abordagem de desenvolvimento que foca em modelar o software de acordo com o domínio do problema, não necessariamente relacionada à escrita de testes antes do desenvolvimento;

(b) Correto. TDD (Test-Driven Development) é a prática de escrever testes de software antes da implementação das funcionalidades, garantindo que o código seja desenvolvido para passar nos testes;

(c) Errado. BDD (Behavior-Driven Development) é uma prática semelhante ao TDD, mas foca em descrever o comportamento esperado do software em linguagem natural, geralmente envolvendo colaboração entre desenvolvedores e não-desenvolvedores;

(d) Errado. XP (Extreme Programming) é uma metodologia ágil que inclui práticas como TDD, mas não é exclusivamente sobre escrever testes antes do desenvolvimento;

(e) Errado. Scrum é uma metodologia ágil para gerenciamento de projetos, não específica para a prática de escrever testes antes do desenvolvimento.

Gabarito: Letra B

---

9. **(FGV / BB – 2023)** O desenvolvimento orientado a testes (TDD) é um processo que se baseia na repetição em ciclos de desenvolvimento curtos. Ele é baseado no conceito test-first oriundo da programação extrema (XP) que incentiva o design simples com alto nível de confiança. O procedimento que conduz este ciclo é denominado:

a) refatoração vermelho-verde.
b) documentação executável.
c) testes da caixa-branca.
d) testes da caixa-preta.
e) mocking up.

Comentários:

(a) Correto. Trata-se de uma técnica de desenvolvimento ágil em que um teste falha inicialmente (red), é feito o mínimo de código para passar no teste (green), e, em seguida, o código é refatorado para melhorar a qualidade sem alterar o comportamento externo;

(b) Errado. Documentação Executável é a prática de criar documentação que também serve como testes automatizados, garantindo que a descrição funcional do software seja verificável e alinhada com o comportamento real do sistema;

(c) Errado. Testes de Caixa Branca são testes de software que verificam a lógica interna, a estrutura e o funcionamento do código, permitindo que os testadores utilizem seu conhecimento sobre a implementação para criar casos de teste;

(d) Errado. Testes de Caixa Preta são testes de software que avaliam a funcionalidade do aplicativo sem examinar seu código-fonte, focando nas entradas e saídas de acordo com as especificações de requisitos;

(e) Errado. Mocking Up é a prática de criar versões simuladas de componentes ou sistemas para testar funcionalidades isoladamente, permitindo que os desenvolvedores avaliem o comportamento do sistema sem depender de integrações externas.

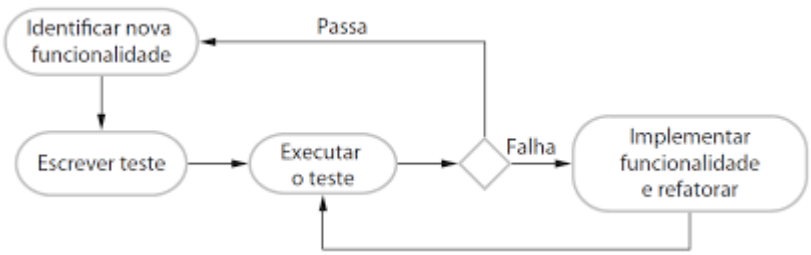
Gabarito: Letra A

---

10. (FGV / Senado Federal – 2022) Você foi contratado para liderar uma equipe de DevOps. Um dos objetivos da sua liderança é aumentar a velocidade das entregas e a qualidade de novos recursos das aplicações utilizando o desenvolvimento orientado a testes. Assinale a opção que apresenta a ordem que descreve o ciclo de desenvolvimento orientado a testes.

a) Refatorar - > Escrever um código funcional
b) Escrever um caso de teste -> Refatorar
c) Refatorar - > Escrever um código funcional - > Escrever um caso de teste
d) Escrever um caso de teste -> Escrever um código funcional -> Refatorar
e) Escrever um código funcional -> Escrever um caso de teste -> Refatorar

Comentários:



O ciclo de desenvolvimento orientado a testes segue o seguinte ciclo: (1) Identificar nova funcionalidade; (2) **Escrever um teste**; (3) Executar o teste; (4) **Implementar funcionalidade e refatorar**.

Gabarito: Letra D

---

11. (CESPE / BANRISUL – 2022) No processo de TDD, o código é desenvolvido em grandes blocos de requisitos do usuário. Cada iteração resulta em um novo teste, que faz parte um conjunto de testes de regressão executado no final do processo de integração.

Comentários:

Durante o TDD, o código é desenvolvido em incrementos **pequenos** e nenhum código é escrito enquanto não houver um teste para experimentá-lo. Cada iteração resulta em um ou mais novos testes, os quais são acrescentados a um conjunto de testes de regressão que são executados a cada mudança. Isso é feito para garantir que o novo código não tenha gerado efeitos colaterais que causem erros no código anterior.

Gabarito: Errado

---

12. **(CESPE / BANRISUL – 2022)** O TDD (Test-Driven Development) é uma metodologia que, ao longo do tempo, implica que o aplicativo em desenvolvimento tenha um conjunto abrangente de testes que ofereça confiança no que foi desenvolvido até então.

Comentários:

O TDD é um processo de desenvolvimento de software que incentiva os desenvolvedores a escrever testes antes de escrever código. A ideia por trás deste processo é que os desenvolvedores escrevam testes para cada pequena parte do código que eles escrevem, garantindo que o código funcione corretamente. Estes testes verificam se o código está funcionando como o esperado. O TDD também incentiva o uso de design incremental, onde pequenas partes do código são adicionadas gradualmente, permitindo que os problemas sejam detectados rapidamente e corrigidos. Ao longo do tempo, o TDD cria um conjunto abrangente de testes que fornecem confiança aos desenvolvedores de que o código é robusto e está funcionando como o esperado.

Gabarito: Correto

---

13. **(CESPE / BANRISUL – 2022)** O TDD (Test-Driven Development), como atividade da XP, é uma forma disciplinada de organizar o código, alterando-o de modo a aprimorar sua estrutura interna, sem que se altere o comportamento externo do software.

Comentários:

Essa não é a definição de TDD e, sim, Refactoring. O TDD é uma atividade do XP que envolve a escrita de testes antes mesmo de o código ser escrito, para que o desenvolvedor possa ter certeza de que o código que está escrevendo passará nos testes, e portanto que as funcionalidades desejadas estão sendo implementadas. Refatoração, por outro lado, é o processo de reescrever o código existente, para melhorar a sua qualidade, sem alterar a funcionalidade existente. Refatoração inclui a reestruturação do código para torná-lo mais limpo e legível, a remoção de código desnecessário, a simplificação da lógica usada e a correção de erros.

Gabarito: Errado

---

14. (FGV / SEFAZ-MG – 2023) Você entrou para um projeto novo, já em andamento, no qual a metodologia que a equipe do projeto segue é a de definir e escrever testes de software a partir das regras de negócio antes mesmo de implementar as funcionalidades propostas. Assinale a opção que indica o nome desse processo de desenvolvimento de software.

a) DDD
b) TDD
c) BDD
d) XP
e) Scrum

Comentários:

O TDD é um processo de desenvolvimento de software que incentiva a escrita de testes de software antes da implementação de qualquer código. O processo começa com a escrita de um teste para uma regra de negócio específica. Então, o código é escrito e, finalmente, os testes são executados para garantir que a regra de negócio esteja funcionando corretamente. O TDD é amplamente utilizado porque ajuda a garantir que o código seja testado de forma adequada, permitindo que qualquer problema seja detectado e corrigido antes que o código seja liberado para produção.

Gabarito: Letra B

---

15. (FGV / Senado Federal – Análise de Sistemas – 2022) Durante o processo de construção de software, a metodologia de Desenvolvimento Orientado a Testes é muito aplicada.

A ordem utilizada na prática do TDD é:

a) escrever a funcionalidade após escrever os testes unitários e, por fim, refatorar o código implementado.

b) escrever os testes unitários após escrever a funcionalidade e, por fim, refatorar o código implementado.

c) escrever a funcionalidade após refatorar o código implementado e por fim escrever os testes unitários.

d) escrever os testes unitários após escrever a funcionalidade e, por fim, escrever os testes de integração.

e) escrever os testes de integração após escrever a funcionalidade e, por fim, refatorar o código.

Comentários:

Essa é uma questão muito clássica! Basta saber a ordem correta: primeiro, escrevemos os testes unitários; depois, escrevemos a funcionalidade; por fim, refatoramos o código. Em outras palavras, a ordem correta é escrever a funcionalidade após escrever os testes unitários e, por fim, refatora o código implementado.

Gabarito: Letra A

---

16. (CESPE / SERPRO – 2021) Em TDD, os testes de um sistema devem ocorrer antes da implementação e ser oportunos, isolados e autoverificáveis.

Comentários:

Os testes devem seguir o modelo FIRST: F (Fast): devem ser rápidos, pois testam apenas uma unidade; I (Isolated): são isolados, testando individualmente as unidades e não sua integração; R (Repeateble): repetição nos testes, com resultados de comportamento constante; S (Self-verifying): autoverificação deve verificar se passou ou se deu como falha o teste; T (Timely): deve ser oportuno, sendo um teste por unidade.

Gabarito: Correto

---

17. (CESPE / INMETRO – 2009) A rotina diária dos desenvolvedores, ao empregar processos baseados no TDD (Test-Driven Development), é concentrada na elaboração de testes de homologação.

Comentários:

A rotina dos desenvolvedores é concentrada, na verdade, na elaboração de testes unitários e, não, de homologação. Lembrem-se: constrói o teste, constrói a funcionalidade e refatora a funcionalidade.

Gabarito: Errado

---

18. (CESPE / INPI – 2013) Usando-se o TDD, as funcionalidades devem estar completas e da forma como serão apresentadas aos seus usuários para que possam ser testadas e consideradas corretas.

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.

REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

Pelo contrário, primeiro são feitos os testes e depois desenvolvem-se as funcionalidades.

Gabarito: Errado

---

19. (CESPE / ANCINE – 2013) No desenvolvimento de software conforme as diretivas do TDD (Test-Driven Development), deve-se elaborar primeiramente os testes e, em seguida, escrever o código necessário para passar pelos testes.

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

Perfeito! Primeiro, criam-se os testes, depois cria-se o código.

▪

Gabarito: Correto

---

20. (CESPE / INMETRO – 2009) Considerando uma organização na qual a abordagem de Test Driven Development (TDD) esteja implementada, assinale a opção correta.

a) Nessa organização, ocorre a execução de iterações com ciclo longo, isto é, com duração de alguns meses.

b) No início de cada iteração, a primeira atividade realizada pela equipe de desenvolvimento é produzir o código que será validado através de testes.

c) O refactoring é uma das primeiras atividades realizada no início de cada iteração.

d) Entre as atividades finais de cada iteração, o desenvolvedor escreve casos de teste automatizados, cuja execução verifica se houve a melhoria desejada ou se uma nova funcionalidade foi implementada.

e) Há coerência e inter-relação com os princípios promovidos pela prática da extreme programming (XP).

Comentários:

(a) Errado, são ciclos curtos e, não, longos; (b) Errado, são produzidos primeiramente os testes e, depois, os códigos; (c) Errado, a refatoração é a última atividade realizada em uma iteração; (d) Errado, escrever casos de testes é uma das atividades iniciais; (e) Correto, trata-se inclusive de uma das práticas recomendadas pelo XP.

Gabarito: Letra E

---

21. (CESPE / MPOG – 2013) Ao realizar o TDD (test-driven development), o programador é conduzido a pensar em decisões de design antes de pensar em código de implementação, o que cria um maior acoplamento, uma vez que seu objetivo é pensar na lógica e nas responsabilidades de cada classe.

Comentários:

Em Engenharia de Software, há dois conceitos importantíssimos: coesão e acoplamento. Quando eu estudava, eu decorava uma frase pequena para entender: "*Coesão é a divisão de responsabilidades e Acoplamento é a dependência entre componentes*". Há outra que dizia assim: "*Uma boa arquitetura de software deve ter componentes de projeto com baixo acoplamento e alta coesão*".

O Acoplamento trata do nível de dependência entre módulos ou componentes de um software. *Por que é bom ter baixo acoplamento?* Porque se os módulos pouco dependem um do outro, modificações de um não afetam os outros, além de não prejudicar o reúso. Se esse princípio não for observado durante a construção da arquitetura de um sistema de software, pode haver problemas sérios de manutenção futura!

Voltando à questão: se o programador pensa em decisões de design antes de pensar em código de implementação, isso diminui o acoplamento - os componentes ficam menos dependentes, tornando a arquitetura bem mais flexível.

Gabarito: Errado

---

22. (CESPE / MPU – 2013) Na metodologia TDD, ou desenvolvimento orientado a testes, cada nova funcionalidade inicia com a criação de um teste, cujo planejamento permite a identificação dos itens e funcionalidades que deverão ser testados, quem são os responsáveis e quais os riscos envolvidos.

Comentários:

Perfeito! Essa metodologia permite um aprendizado maior sobre o problema a ser resolvido, permitindo (não obrigatoriamente) a identificação de itens, funcionalidades, responsáveis e riscos.

Gabarito: Correto

---

23. (CESPE / STF – 2013) No TDD, o primeiro passo do desenvolvedor é criar o teste, denominado teste falho, que retornará um erro, para, posteriormente, desenvolver o código e aprimorar a codificação do sistema.

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

Perfeito! Cria-se o teste falho, desenvolve-se um código e só então ocorre a refatoração.

Gabarito: Correto

---

24. (CESPE / TRT17 – 2013) TDD consiste em uma técnica de desenvolvimento de software com abordagem embasada em perspectiva evolutiva de seu desenvolvimento. Essa abordagem envolve a produção de versões iniciais de um sistema a partir das quais é possível realizar verificações de suas qualidades antes que ele seja construído.

Comentários:

A questão diz que versões iniciais são produzidas e, a partir dessas versões, é possível realizar verificações de suas qualidades antes que ele seja construído. Na verdade, testes são criados inicialmente para verificar sua qualidade e, a partir daí, versões são produzidas. Logo, a questão inverteu os conceitos!

Gabarito: Correto

---

25. (CESPE / AL-RN – 2013) Um típico ciclo de vida de um projeto em TDD consiste em:

I. Executar os testes novamente e garantir que estes continuem tendo sucesso.
II. Executar os testes para ver se todos estes testes obtiveram êxito.
III. Escrever a aplicação a ser testada.
IV. Refatorar (refactoring).
V. Executar todos os possíveis testes e ver a aplicação falhar.
VI. Criar o teste.

A ordem correta e cronológica que deve ser seguida para o ciclo de vida do TDD está expressa em:

a) IV – III – II – V – I – VI.
b) V – VI – II – I – III – IV.
c) VI – V – III – II – IV – I.
d) III – IV – V – VI – I – II.
e) III – IV – VI – V – I – II.

Comentários:

A ordem correta é: (VI) Criar o teste; (V) Executar todos os possíveis testes e ver a aplicação falhar; (III) Escrever a aplicação a ser testada; (II) Executar os testes para ver se todos estes testes obtiveram êxito; (IV) Refatorar (refactoring); (I) Executar os testes novamente e garantir que estes continuem tendo sucesso.

Gabarito: Letra C

---

26. (FGV / ALMT – 2013) Com relação ao desenvolvimento orientado (dirigido) a testes (do Inglês Test Driven Development – TDD), analise as afirmativas a seguir.

I. TDD é uma técnica de desenvolvimento de software iterativa e incremental.

II. TDD implica escrever o código de teste antes do código de produção, um teste de cada vez, tendo certeza de que o teste falha antes de escrever o código que irá fazê-lo passar.

III. TDD é uma técnica específica do processo XP (Extreme Programming), portanto, só pode ser utilizada em modelos de processos ágeis de desenvolvimento de software.

Assinale:

a) Se somente as afirmativas I e II estiverem corretas.
b) Se somente as afirmativas I e III estiverem corretas.
c) Se somente as afirmativas II e III estiverem corretas.
d) Se somente a afirmativa III estiver correta.
e) Se somente a afirmativa I estiver correta.

Comentários:

(I) Correto, é iterativo e incremental (lembrando que, em sua imensa maioria, metodologias ágeis são iterativas/incrementais); (II) Correto, escrevem-se os testes antes, fá-lo falhar e só depois escreve-se o código da aplicação; (III) Errado, ele é uma abordagem independente que pode ser utilizado com metodologias ágeis ou tradicionais.

Gabarito: Letra A

---

27. (FCC / TRT-MG – 2015) Um analista de TI está participando do desenvolvimento de um software orientado a objetos utilizando a plataforma Java. Na abordagem de desenvolvimento adotada, o código é desenvolvido de forma incremental, em conjunto com o teste para esse incremento, de forma que só se passa para o próximo incremento quando o atual passar no teste. Como o código é desenvolvido em incrementos muito pequenos e são executados testes a cada vez que uma funcionalidade é adicionada ou que o programa é refatorado, foi necessário definir um ambiente de testes automatizados utilizando um framework popular que suporta o teste de programas Java.

A abordagem de desenvolvimento adotada e o framework de suporte à criação de testes automatizados são, respectivamente,

a) Behavior-Driven Development e JTest.
b) Extreme Programming e Selenium.
c) Test-Driven Development e Jenkins.
d) Data-Driven Development and Test e JUnit.
e) Test-Driven Development e JUnit.

Comentários:

A abordagem claramente é o TDD e o framework evidentemente é o JUnit (ferramenta de suporte à criação de testes unitários automatizados). Lembrando que: DDD é um modelo de programação em que os próprios dados controlam o fluxo do programa e não a lógica do programa; XP é uma metodologia ágil de gerenciamento de projetos que suporta lançamentos frequentes em curtos ciclos de desenvolvimento para melhorar a qualidade do software e permitir que os desenvolvedores respondam às mudanças nos requisitos dos clientes; BDD é um método de desenvolvimento ágil que encoraja a colaboração entre desenvolvedores; Selenium é uma ferramenta usada para testes funcionais em aplicações web; e JTest é um ramework de análise estática que usa a linguagem Java.

Gabarito: Letra E

---

28. (CESPE / TRE-PE – 2017) O desenvolvimento orientado a testes (TDD):

a) é um conjunto de técnicas que se associam ao XP (extreme programming) para o desenvolvimento incremental do código que se inicia com os testes.

b) agrega um conjunto de testes de integração para avaliar a interconexão dos componentes do software com as aplicações a ele relacionadas.

c) avalia o desempenho do desenvolvimento de sistemas verificando se o volume de acessos/transações está acima da média esperada.

d) averigua se o sistema atende aos requisitos de desempenho verificando se o volume de acessos/transações mantém-se dentro do esperado.

e) testa o sistema para verificar se ele foi desenvolvido conforme os padrões e a metodologia estabelecidos nos requisitos do projeto.

Comentários:

O único item que faz algum sentido em relação ao TDD é o primeiro, isto é, conjunto de técnicas intimamente ligadas ao XP para o desenvolvimento de software que se inicia com os testes.

Gabarito: Correto

---

29. (CESPE / STM – 2018) O TDD (test driven development) parte de um caso de teste que caracteriza uma melhoria desejada ou nova funcionalidade a ser desenvolvida, de modo a confirmar o comportamento correto e possibilitar a evolução ou refatoração do código.

Comentários:

TDD é um método para construir software que enfatiza a criação de testes antes da criação do código-fonte. Logo, faz-se o teste - se não passou, refatora!

Gabarito: Correto

---

30. (CESPE / TRE/PI – 2016) O TDD (test driven development):

a) apresenta como vantagem a leitura das regras de negócio a partir dos testes, e, como desvantagem, a necessidade de mais linhas de códigos que a abordagem tradicional, o que gera um código adicional.

b) impede que seja aplicada a prática de programação em pares, que é substituída pela interação entre analista de teste, testador e programador.

c) é um conjunto de técnicas associadas ao eXtremme Programing e a métodos ágeis, sendo, contudo, incompatível com o Refactoring, haja vista o teste ser escrito antes da codificação.

d) refere-se a uma técnica de programação cujo principal objetivo é escrever um código funcional limpo, a partir de um teste que tenha falhado.

e) refere-se a uma metodologia de testes em que se devem testar condições, loops e operações; no entanto, por questão de simplicidade, não devem ser testados polimorfismos.

Comentários:

(a) Errado, esse item não faz nenhum sentido em relação ao TDD; (b) Errado, não impede a programação em pares; (c) Errado, ela é totalmente compatível e dependente da refatoração; (d) Correto, ela pode ser vista como uma técnica de programação cujo objetivo é escrever um código funcional limpo a partir de um teste falho; (e) Errado, não se trata de uma metodologia de testes.

Gabarito: Letra D

---

31. (UFRRJ / UFRRJ – 2015) Os testes de unidade têm papel central na metodologia de implementação dirigida por testes, popularizada pelo processo XP e adotada em outros métodos. Esses testes são criados primeiro, exercitando o contrato de cada operação implementada pelos métodos. Em seguida, o código dos métodos é escrito para cumprir os contratos e, portanto, passar nos testes de unidade. Esse cenário corresponde à abordagem

a) TDD.
b) MDD.
c) DDC.
d) MDE.
e) FDD.

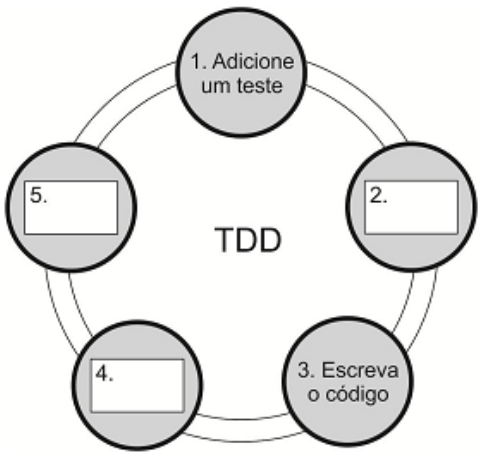
Comentários:

Testes de unidade têm papel central? Metodologia dirigida a testes? Popularizada pelo XP? Testes são criados primeiro? Tudo isso nos remente a... TDD!

Gabarito: Letra A

---

32. (FCC / TRE-PR – 2017) Considere o ciclo do Test-Driven Development – TDD.



A Caixa:

a) 2. corresponde a "Execute os testes automatizados".
b) 4. corresponde a "Refatore o código".
c) 5. corresponde a "Execute os testes novamente e observe os resultados".
d) 4. corresponde a "Execute os testes automatizados".
e) 5. corresponde a "Faça todos os testes passarem".

Comentários:

(a) Errado, corresponde a "Execute o teste e observe o resultado"; (b) Errado, corresponde a "Execute os testes automatizados"; (c) Errado, corresponde a "Refatore os códigos"; (d) Correto, corresponde realmente a "Execute os testes automatizados"; (e) Errado, corresponde a "Refatore os códigos".

Gabarito: Letra D

---

33. (IESES / TRE-MA – 2015) A respeito da técnica de testes TDD é correto afirmar que:

a) Testa o software com base no comportamento esperado.
b) É uma prática para desenvolvimento de testes unitário que pode utilizar o processo Red-Green-Refactor.
c) Utiliza-se da estrutura Dado, Quando e Então para montar os testes.
d) Prega que os testes devem ser realizados sempre após a implementação ser concluída.

Comentários:

(a) Errado, não vejo nada de errado nesse item – ele pode testar o software com base no comportamento esperado! No entanto, a banca considerou o item errado; (b) Correto, ele realmente utiliza o processo RED/GREEN/REFACTOR; (c) Errado, esse item não faz qualquer sentido; (d) Errado, testes são realizados antes de a implementação ser concluída.

Gabarito: Letra B

---

34. (CESPE / TER-RS – 2015) Projeto para o desenvolvimento de software que utilize TDD deve:

a) realizar sprints a cada quinzena.
b) desenvolver pequenos releases.
c) apresentar grande quantidade de testes unitários de código-fonte previamente desenvolvidos.
d) apresentar linguagem de programação estruturada.
e) recomendar a preparação dos testes para que, posteriormente, seja desenvolvido o código.

Comentários:

(a) Errado, não há nenhuma relação entre TDD e Sprints; (b) Errado, são desenvolvidas pequenas unidades e, não, releases; (c) Errado, os testes unitários são escritos iterativamente a medida que o desenvolvimento ocorre; (d) Errado, não há nenhuma relação ou dependência com linguagens de programação; (e) Correto, recomenda-se preparar testes antes do desenvolvimento do código.

Gabarito: Letra E

---

35. (FCC / TRE-AP – 2015) O TDD – Test Driven Development (Desenvolvimento orientado a teste):

a) é parte das metodologias ágeis UP – Unified Process e XP – Extreme Programming, tendo sido criado para ser usado em metodologias que respeitam os 4 princípios do Manifesto Ágil.

b) transforma o desenvolvimento, pois deve-se primeiro implementar o sistema antes de escrever os testes. Os testes são utilizados para facilitar no entendimento do projeto e para clarear o que se deseja em relação ao código.

c) baseia-se em um ciclo simples: escreve-se um código -> cria-se um teste para passar no código -> refatora-se.

d) propõe a criação de testes que validem o código como um todo para reduzir o tempo de desenvolvimento.

e) beneficia-se de testes que seguem o modelo FIRST: F (Fast) I (Isolated) R (Repeatable) S (Self-verifying) T (Timely).

Comentários:

ETAPA	DESCRIÇÃO

VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

(a) Errado, Unified Process (UP) não é uma metodologia ágil; (b) Errado, devem ser implementados os testes antes do sistema; (c) Errado, cria-se um teste falho > escreve-se o código que passa > refatora-se; (d) Errado, criam-se testes que validam unidades e, não, o código como um todo; (e) Correto, vamos falar um pouquinho sobre isso agora:

Existe uma representação chamada Modelo FIRST: **F (Fast):** devem ser rápidos, pois testam apenas uma unidade; **I (Isolated):** testes unitários são isolados, testando individualmente as unidades e não sua integração; **R (Repeateble):** repetição nos testes, com resultados de comportamento constante; **S (Self-verifying):** a auto verificação deve verificar se passou ou se deu como falha o teste; **T (Timely):** o teste deve ser oportuno, sendo um teste por unidade.

Gabarito: Letra E

---

36. (CESPE / TRE-TO – 2017) O TDD (Test-Driven Development), que vem sendo adotado para testar os projetos de software,

a) utiliza os testes de caixa preta antes da entrega do software.
b) agiliza os testes por amostragem sem compatibilidade retroativa.
c) cobre amplamente os testes unitários.
d) escreve o teste antes da codificação do software.
e) realiza refactoring antes de escrever a aplicação a ser testada.

Comentários:

Eu já começo discordando do enunciado – ele vem sendo utilizado para desenvolver software utilizando testes, mas não para testar projetos de software. Ignorando a redação da questão, vamos aos comentários: (a) Errado, ele tipicamente utiliza testes de unidade antes da entrega do software; (b) Errado, esse item não faz qualquer sentido; (c) Errado, ele não tem o intuito principal de cobrir amplamente testes unitários; (d) Correto, o teste é escrito antes da codificação do software; (e) Errado, não faz sentido fazer *refactoring* antes de escrever a aplicação.

Gabarito: Letra D

---

37. (FGV / IBGE – 2016) O Desenvolvimento Orientado a Testes (TDD) é um método de desenvolvimento criado e disseminado por Kent Beck em seu livro "Test-driven development". O método define regras, boas práticas e um ciclo de tarefas com 3 etapas: a etapa vermelha, a etapa verde e a etapa de refatoração, ilustrado na imagem abaixo:

Com relação às regras e boas práticas de TDD e ao seu ciclo, é correto afirmar que:

a) pode-se escrever testes que não compilam na etapa vermelha;
b) na etapa verde deve-se escrever código que testa uma funcionalidade a fundo de forma criteriosa e detalhada;
c) código novo só é escrito se um teste automatizado passar;
d) a duplicação é tolerada na etapa de refatoração;
e) é uma boa prática de TDD iniciar o desenvolvimento do código de uma funcionalidade e, logo em seguida, testá-la.

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

(a) Correto. Nessa etapa, o objetivo do desenvolvedor é escrever um pequeno teste que não funcione e que talvez nem mesmo compile inicialmente; (b) Errado. Nessa etapa, escreve-se o código que apenas passa no teste, sem necessidade de aprofundamento em detalhes de implementação; (c) Errado. O código novo é escrito para o teste automatizado passar; (d) Errado. Não é tolerada a duplicação, uma vez que o objetivo é eliminar redundâncias e melhorar o código; (e) Errado. Vimos centenas de vezes que o primeiro passo é criar o teste e depois escrever.

Gabarito: Letra A

---

38. (IADES / EBSERH – 2013) Assinale a alternativa que não corresponde a uma das fases do processo de desenvolvimento, dirigido a testes (TDD).

a) Executar o teste, com os outros testes implementados, que rodarão e fornecerão o resultado de que o software está sem problemas.

b) Escrever o teste para a funcionalidade e implementação.
c) Realizar a identificação do incremento de funcionalidade.
d) Implementar a funcionalidade e executar novamente o teste.
e) Implementar a próxima parte da funcionalidade, após todos os testes terem sido executados, com sucesso

Comentários:

(a) Errado. Na terceira fase, executa-se realmente o teste junto com todos os outros testes implementados. No entanto, inicialmente você não terá implementado a funcionalidade, logo o novo teste falhará; (b) Correto, essa é a segunda fase; (c) Correto, essa é a primeira fase; (d) Correto, essa é a quarta fase; (e) Correto, essa é a quinta fase.

Gabarito: Letra A

---

39. (PR-4 / UFRJ – 2018) O ciclo do TDD - Test Driven Development, ou, em português, Desenvolvimento Guiado por Testes consiste em:

a) implementar teste unitário falho, tornar o teste bem-sucedido e refatorar.
b) implementar a funcionalidade, executar teste unitário e refatorar.
c) implementar teste unitário falho, refatorar e tornar o teste bem-sucedido.
d) implementar a funcionalidade, refatorar e tornar o teste bem-sucedido.
e) refatorar, executar teste unitário e implementar a funcionalidade.

Comentários:

(a) Correto, as atividades são exatamente essas e nessa ordem; (b) Errado, o teste vem antes da implementação da funcionalidade; (c) Errado, primeiro você torna o teste bem sucedido e depois refatora; (d) Errado, não faz sentido refatorar antes do teste; (e) Errado, não faz sentido refatorar antes do teste.

Gabarito: Letra A

---

40. (CESPE / STJ – 2015) Um dos passos executados no ciclo de atividades do processo TDD é a criação de novos testes para as falhas encontradas no código original, sem alteração deste.

Comentários:

Nooooope... testes devem realmente encontrar falhas – o que se altera é o código para passar no teste e, não, o teste em si.

Gabarito: Errado

---

41. (CS-UFG / AL-GO – 2015) O desenvolvimento dirigido a testes (TDD, do Inglês Test-Driven Development) é uma abordagem de desenvolvimento de software na qual se intercalam testes e desenvolvimento de código. Uma das características da abordagem TDD é:

a) a sua utilidade no desenvolvimento de softwares novos.
b) o maior custo associado aos testes de regressão.
c) a redução da importância da automatização dos testes.
d) a sua adequação a processos de software sequenciais.

Comentários:

(a) Correto. Ela é mais ideal para o desenvolvimento de softwares novos; (b) Errado, ela reduz custos de testes de regressão; (c) Errado, ela aumenta a importância da automatização de testes; (d) Errado, ela é adequada a processos iterativos – lembrem-se do ciclo de testes, falhas e refatorações.

Gabarito: Letra A

---

42. (UECE-CEV / FUNCEME – 2018) Test-driven Development (TDD) é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código
(Sommerville, I. Engenharia de Software, 9a edição, 2011).

A respeito do TDD, é correto afirmar que:

a) consiste em um processo iterativo que se inicia escrevendo um código de uma funcionalidade do sistema e, logo em seguida, testa-o para saber se a implementação foi correta.
b) apesar de útil, não diminui o custo de testes de regressão do sistema.
c) sua utilização elimina a necessidade de testes de validação do sistema, uma vez que ele já foi testado incrementalmente.
d) apesar de ter sido apresentado como parte dos métodos ágeis, também pode ser usado em outros processos de desenvolvimento de software.

Comentários:

(a) Errado, inicia-se escrevendo o teste; (b) Errado, diminui – sim – os custos de testes de regressão; (c) Errado, não elimina em nenhuma hipótese testes de validação/aceitação; (d) Correto, ele pode ser utilizado com métodos ágeis ou tradicionais.

Gabarito: Letra D

---

43. (FAUGRS / UFRGS – 2018) _____ é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código. Essencialmente, desenvolve-se um código de forma incremental em conjunto com um teste para este incremento. Não se avança para o próximo incremento até que o código desenvolvido passe no teste. Essa abordagem foi introduzida como parte de métodos ágeis, mas pode ser também

usada em processos de desenvolvimento dirigido a planos. Assinale a alternativa que preenche corretamente a lacuna do texto acima.

a) Desenvolvimento Guiado por Testes (TDD)
b) Desenvolvimento em Espiral
c) Engenharia Dirigida a Modelos (MDD)
d) Rational Unified Process (RUP)
e) Teste de Sistema

Comentários:

TDD é uma **abordagem para o desenvolvimento** de programas em que se **intercalam testes e desenvolvimento de código**. Essencialmente, desenvolve-se um código de forma incremental em conjunto com um teste para este incremento. Não se avança para o próximo incremento até que o código desenvolvido passe no teste. Essa abordagem foi introduzida como parte de métodos ágeis, mas pode ser também usada em processos de desenvolvimento dirigido a planos.

Gabarito: Letra A

---

44. (VUNESP / TCE-SP – 2015) No Desenvolvimento Orientado a Testes (TDD), os casos de teste que definem o recurso a ser implementado devem ser elaborados:

a) assim que o código do teste estiver pronto.
b) antes de o código do recurso ser desenvolvido.
c) após o código do recurso ter sido completamente documentado.
d) simultaneamente com o desenvolvimento do código do recurso.
e) somente se o código do recurso apresentar erros.

Comentários:

(a) Errado, os casos de testes já devem estar prontos antes do código do teste ser feito; (b) Correto, devem ser elaborados antes de o código do recurso ser desenvolvido; (c) Errado, não existe o conceito de documentação formal, os próprios testes agem como uma forma de documentação que descreve o que o código deve fazer; (d) Errado, as etapas de desenvolvimento do código do recurso e a implementação dos casos de testes ocorrem em etapas distintas e não concomitantes; (e) Errado, casos de testes sempre são implementados porque eles representam a funcionalidade a ser desenvolvida.

Gabarito: Letra B

---

45. (IBFC / EMBASA – 2017) No Ciclo de Desenvolvimento do TDD (Test-Driven Development), utiliza-se a estratégia que aplica três palavras-chaves (em inglês), que é denominada:

a) Red, Green, Refactor
b) White, Gray, Black
c) White, Black, Refactor
d) Green, Yellow, Red

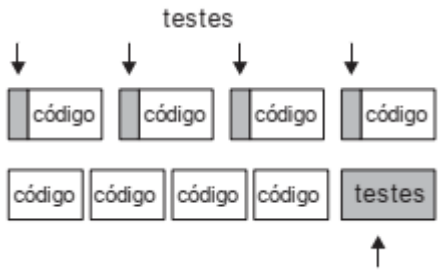
Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

Trata-se do RED > GREEN > REFACTOR.

Gabarito: Letra A

46.(FCC / CREMESP – 2016) Considere a figura abaixo que apresenta duas abordagens de teste.



A figura:

a) ilustra as duas fases do TDD, que correspondem a escrever pequenos testes e testá-los no final.
b) mostra o ciclo conhecido como Vermelho-Verde-Refatora.
c) apresenta a diferença entre testes automatizados e testes manuais no XP.
d) mostra que um desenvolvedor que pratica TDD tem mais feedbacks do que um que escreve testes ao final.
e) evidencia que TDD é impraticável, pois o desenvolvedor gasta muito tempo escrevendo código de testes.

Comentários:

Qual é a diferença entre a abordagem superior e inferior? Na parte superior, testes são aplicados individualmente em cada unidade de código. Na parte inferior, todos os códigos foram escritos e,

somente depois, foram testados. Logo, existem mais feedbacks na abordagem superior (TDD) do que na abordagem inferior.

Gabarito: Letra D

---

47. (CESPE / ANATEL – 2014) Em se tratando de desenvolvimento de softwares dirigidos a testes (TDD), a execução dos testes é realizada antes da implementação da funcionalidade.

Comentários:

Perfeito! A execução dos testes de realmente ocorrem antes da implementação da funcionalidade.

Gabarito: Correto

---

48. (CESPE / TC-DF – 2014) No TDD, o refatoramento do código deve ser realizado antes de se escrever a aplicação que deve ser testada.

Comentários:

Opaaaaa... como você vai refatorar um código que ainda não foi escrito? Não faz sentido!

Gabarito: Errado

---

49. (CESPE / STJ – 2015) No método de desenvolvimento TDD (Test Driven Development), o desenvolvedor escreve primeiro um caso de teste e, posteriormente, o código.

Comentários:

Perfeito! Esse é o conceito do *test-first*, isto é, o desenvolvedor escreve primeiro um caso de teste e, posteriormente, o código.

Gabarito: Correto

---

50. (CESPE / MPE-PI – 2018) O TDD possibilita o desenvolvimento de softwares fundamentado em testes. O ciclo de desenvolvimento do TDD segue os seguintes passos:

- escrever um teste que inicialmente não passa;
- adicionar uma nova funcionalidade do sistema;
- fazer o teste passar;
- realizar a integração contínua do código;
- escrever o próximo teste.

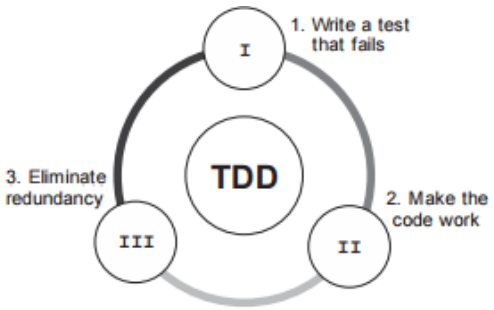
Comentários:

A ordem realmente está correta, no entanto há uma pegadinha: na quarta etapa, realiza-se a refatoração do código e, não, a integração contínua.

Gabarito: Errado

---

51. (FCC / Prefeitura de Teresina-PI – 2016) O Test Driven Development – TDD é uma das práticas sugeridas na eXtreme Programming – XP, onde o programador escreve o teste antes de escrever o código. O ciclo de desenvolvimento utilizando TDD é mostrado abaixo.



Considere:

I. Etapa inicial, onde se escreve um teste que falha, para alguma funcionalidade que ainda será Escrita.
II. Já com o teste criado, é o momento de executar o teste.
III. Eliminar códigos redundantes, remover acoplamentos, enfim, identificar pontos de melhoria no código.

As etapas I, II e III são, respectivamente,

a) Iniciação, Execução e Controle.
b) Red, Green e Refactor.
c) Iniciação, Atuação e Otimização.
d) Plan, Do e Check.
e) Planejamento, Execução e Melhoria.

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

(RED) Etapa inicial, onde se escreve um teste que falha, para alguma funcionalidade que ainda será Escrita; (GREEN) Já com o teste criado, é o momento de executar o teste; (REFACTOR) Eliminar códigos redundantes, remover acoplamentos, enfim, identificar pontos de melhoria no código.

Gabarito: Letra B

---

52. (FAUGRS / BANRISUL – 2018) Considere as ações abaixo, executadas em desenvolvimento orientado a testes, Test-Driven Design (TDD).

I - Escrever código de teste.
II - Verificar se o teste falha.
III - Escrever código de produção.
IV - Executar teste até passar (reescrevendo o código de produção, se for necessário, até que o teste passe).
V - Refatorar código de produção e/ou de teste para melhorá-lo.

Considerando que se deseja incluir um novo caso de teste, assinale a alternativa que apresenta a sequência de ações que devem obrigatoriamente ocorrer para essa inclusão, segundo o TDD.

a) I, III e IV.
b) III, I e IV.
c) I, II, III e IV.
d) I, III, IV e V.
e) I, II, III, IV e V.

Comentários:

A ordem correta é: (I) Escrever código de teste; (II) Verificar se o teste falha; (III) Escrever código de produção; (IV) Executar teste até passar (reescrevendo o código de produção, se for necessário, até que o teste passe); (V) Errado, essa não é uma ação obrigatória – como apresenta o enunciado.

Gabarito: Letra C

---

53. (FGV / IBGE – 2017) Test Driven Development (TDD) é uma prática muito utilizada no processo de desenvolvimento de sistemas computacionais. Analise as afirmativas a seguir sobre o uso da prática de TDD:

I. Tornam os testes de regressão mais demorados porque o desenvolvedor precisará fazer testes manuais várias vezes por dia.

II. Garante que os requisitos do sistema sejam atendidos porque o desenvolvedor escreverá o código de testes sempre que acabar a implementação do código do sistema.

III. Ajuda o desenvolvedor a escrever código de qualidade porque ele gastará parte do seu tempo escrevendo código de testes.

Está correto o que se afirma em:

a) somente I;
b) somente II;
c) somente III;
d) somente II e III;
e) I, II e III.

Comentários:

(I) Errado. De acordo com Ian Sommerville, um dos benefícios mais importantes de desenvolvimento dirigido a testes é que ele reduz os custos dos testes de regressão, uma vez que testes automatizados – fundamentais para o desenvolvimento *test-first* – reduzem drasticamente os custos com testes de regressão; (II) Errado. O desenvolver escreverá o código de testes antes de acabar a implementação do código do sistema; (III) Correto. O TDD realmente ajuda o desenvolvedor a escrever código de qualidade porque ele gastará parte do seu tempo escrevendo código de testes.

Gabarito: Letra C

---

54. (CESPE / ANATEL – 2014) Na atividade de TDD (test-driven development), a escrita de teste primeiro define implicitamente tanto uma interface quanto uma especificação do comportamento para a funcionalidade que está sendo desenvolvida, estando, entretanto, a viabilidade do uso dessa abordagem limitada aos processos de desenvolvimento de software que seguem as práticas ágeis.

Comentários:

Opa... a interface deve ser definida explicitamente! Além disso, conforme afirma Ian Sommerville, ele também pode ser utilizado em processos de desenvolvimento dirigido a planos (tradicionais) e, não só, aos ágeis.

Gabarito: Errado

---

55. (CESPE / TRE-MT – 2015) Considere as seguintes etapas de um processo do tipo desenvolvimento orientado a testes (TDD).

I Implementar funcionalidade e refatorar.
II Identificar nova funcionalidade.
III Executar o teste.

IV Escrever o teste.
V Implementar a próxima parte da funcionalidade.

Assinale a opção que apresenta a sequência correta em que essas etapas devem ser realizadas.

a) I; IV; III; II; V
b) IV; III; II; I; V
c) I; IV; II; III; V
d) II; IV; III; I; V
e) IV; II; III; I; V

Comentários:

A ordem correta é: (II) Identificar nova funcionalidade; (IV) Escrever o teste; (III) Executar o teste; (I) Implementar funcionalidade e refatorar; (V) Implementar a próxima parte da funcionalidade.

Gabarito: Letra D

---

56. **(FCC / SEFAZ-SC – 2018)** O Test-Driven Development (TDD) é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código. As etapas do processo fundamental de TDD são mostradas abaixo em ordem alfabética:

I. Escrever um teste para a funcionalidade identificada e implementá-lo como um teste automatizado.

II. Executar o teste, junto com os demais testes já implementados, sem implementar a nova funcionalidade no código.

III. Identificar e implementar uma outra funcionalidade, após todos os testes serem executados com sucesso.

IV. Identificar uma nova funcionalidade pequena para ser incrementada com poucas linhas em um código.

V. Implementar a nova funcionalidade no código e reexecutar o teste.

VI. Refatorar o código com melhorias incrementais até que o teste execute sem erros.

VII. Revisar a funcionalidade e o teste, caso o código execute sem falhar.

Considerando o item IV a primeira etapa e o item III a última etapa, a sequência intermediária correta das etapas do processo é:

a) I – II – VII – V e VI.
b) I – V – II – VII e VI.
c) I – VI – V – VII e II.
d) V – I – II – VII e VI.
e) V – I – VI – VII e II.

Comentários:

São somente cinco etapas: I. Escrever um teste para a funcionalidade identificada e implementá-lo como um teste automatizado; II. Executar o teste, junto com os demais testes já implementados, sem implementar a nova funcionalidade no código; VII. Revisar a funcionalidade e o teste, caso o código execute sem falhar; V. Implementar a nova funcionalidade no código e reexecutar o teste; VI. Refatorar o código com melhorias incrementais até que o teste execute sem erros.

Gabarito: Letra A

LISTA DE QUESTÕES – DIVERSAS BANCAS

1. **(CESGRANRIO / IPEA – 2024)** Uma gerente de testes de software propôs a seu time de desenvolvimento que começasse a aplicar a abordagem Test Driven Development (TDD). É uma das características principais dessa abordagem iniciar o desenvolvimento de testes:

   a) antes de implementar alguma funcionalidade em si.
   b) durante o período de homologação.
   c) após as funcionalidades serem construídas.
   d) quando a primeira leva de funcionalidades planejadas forem codificadas em algum sprint.
   e) pelos testes de interface automatizado, seguidos pelos testes unitários.

2. **(CESPE / INPI – 2024)** O desenvolvimento dirigido por testes (TDD) é modelado em três estados: vermelho, verde e refatorar. Um exemplo da ação de refatoração é a simulação do comportamento dos componentes que interagem com a unidade de teste que está falhando.

3. **(CESPE / CNPq – 2024)** O TDD é uma tendência que enfatiza o projeto de casos de teste antes da criação do código fonte e se caracteriza como parte do modelo ágil de desenvolvimento de software.

4. **(CESPE / CNPq – 2024)** No TDD, o teste deve ser criado com o objetivo de fazer o segmento de código falhar, gerando-se um processo iterativo que permite a submissão de muitas subfunções simultaneamente, o que confere uma agilidade significativa ao processo.

5. **(CESPE / AGER-MT – 2023)** Assinale a opção que corresponde ao método de teste de software adotado, sob a perspectiva do desenvolvedor, a partir de casos de teste do código, escritos em linguagem técnica, para testar as funcionalidades antes da implementação da solução desenvolvida:

   a) unit testing
   b) TDD
   c) BDD
   d) ATDD
   e) teste de caixa preta

6. **(CESPE / TC-DF – 2023)** Na etapa de refactor do processo TDD, parte-se do pressuposto de que os testes tenham passado nas fases anteriores, o que permite que o código seja aprimorado sem a preocupação de duplicações de código.

7. **(CESPE / EMPREL – 2023)** Ao adotar uma prática ágil para a criação de um software, seu desenvolvedor optou pela implementação com qualidade de uma funcionalidade do sistema; para isso, escreveu um caso de teste automatizado, com base nos requisitos especificados, e realizou testes de unidade em uma linguagem similar à usada no desenvolvimento da

funcionalidade. Da situação hipotética precedente infere-se que a prática adotada pelo desenvolvedor está associada ao:

a) desenvolvimento orientado por comportamento (BDD).
b) gerenciamento de produtos com Scrum.
c) desenvolvimento guiado por testes (TDD).
d) desenvolvimento guiado por testes de aceitação (ATDD).
e) gerenciamento de produtos com Kanban.

8. **(FGV / SEFAZ-MG – 2023)** Você entrou para um projeto novo, já em andamento, no qual a metodologia que a equipe do projeto segue é a de definir e escrever testes de software a partir das regras de negócio antes mesmo de implementar as funcionalidades propostas. Assinale a opção que indica o nome desse processo de desenvolvimento de software:

a) DDD
b) TDD
c) BDD
d) XP
e) Scrum

9. **(FGV / BB – 2023)** O desenvolvimento orientado a testes (TDD) é um processo que se baseia na repetição em ciclos de desenvolvimento curtos. Ele é baseado no conceito test-first oriundo da programação extrema (XP) que incentiva o design simples com alto nível de confiança. O procedimento que conduz este ciclo é denominado:

a) refatoração vermelho-verde.
b) documentação executável.
c) testes da caixa-branca.
d) testes da caixa-preta.
e) mocking up.

10. **(FGV / Senado Federal – 2022)** Você foi contratado para liderar uma equipe de DevOps. Um dos objetivos da sua liderança é aumentar a velocidade das entregas e a qualidade de novos recursos das aplicações utilizando o desenvolvimento orientado a testes. Assinale a opção que apresenta a ordem que descreve o ciclo de desenvolvimento orientado a testes.

a) Refatorar - > Escrever um código funcional
b) Escrever um caso de teste -> Refatorar
c) Refatorar - > Escrever um código funcional - > Escrever um caso de teste
d) Escrever um caso de teste -> Escrever um código funcional -> Refatorar
e) Escrever um código funcional -> Escrever um caso de teste -> Refatorar

11. **(CESPE / BANRISUL – 2022)** No processo de TDD, o código é desenvolvido em grandes blocos de requisitos do usuário. Cada iteração resulta em um novo teste, que faz parte um conjunto de testes de regressão executado no final do processo de integração.

12. **(CESPE / BANRISUL – 2022)** O TDD (Test-Driven Development) é uma metodologia que, ao longo do tempo, implica que o aplicativo em desenvolvimento tenha um conjunto abrangente de testes que ofereça confiança no que foi desenvolvido até então.

13. **(CESPE / BANRISUL – 2022)** O TDD (Test-Driven Development), como atividade da XP, é uma forma disciplinada de organizar o código, alterando-o de modo a aprimorar sua estrutura interna, sem que se altere o comportamento externo do software.

14. **(FGV / SEFAZ-MG – 2023)** Você entrou para um projeto novo, já em andamento, no qual a metodologia que a equipe do projeto segue é a de definir e escrever testes de software a partir das regras de negócio antes mesmo de implementar as funcionalidades propostas. Assinale a opção que indica o nome desse processo de desenvolvimento de software.

a) DDD
b) TDD
c) BDD
d) XP
e) Scrum

15. **(FGV / Senado Federal – Análise de Sistemas – 2022)** Durante o processo de construção de software, a metodologia de Desenvolvimento Orientado a Testes é muito aplicada.

A ordem utilizada na prática do TDD é:

a) escrever a funcionalidade após escrever os testes unitários e, por fim, refatorar o código implementado.

b) escrever os testes unitários após escrever a funcionalidade e, por fim, refatorar o código implementado.

c) escrever a funcionalidade após refatorar o código implementado e por fim escrever os testes unitários.

d) escrever os testes unitários após escrever a funcionalidade e, por fim, escrever os testes de integração.

e) escrever os testes de integração após escrever a funcionalidade e, por fim, refatorar o código.

16. **(CESPE / SERPRO – 2021)** Em TDD, os testes de um sistema devem ocorrer antes da implementação e ser oportunos, isolados e autoverificáveis.

17. **(CESPE / INMETRO / 2009)** A rotina diária dos desenvolvedores, ao empregar processos baseados no TDD (Test-Driven Development), é concentrada na elaboração de testes de homologação.

18. **(CESPE / INPI / 2013)** Usando-se o TDD, as funcionalidades devem estar completas e da forma como serão apresentadas aos seus usuários para que possam ser testadas e consideradas corretas.

19. **(CESPE / ANCINE / 2013)** No desenvolvimento de software conforme as diretivas do TDD (Test-Driven Development), deve-se elaborar primeiramente os testes e, em seguida, escrever o código necessário para passar pelos testes.

20. **(CESPE / INMETRO / 2009)** Considerando uma organização na qual a abordagem de Test Driven Development (TDD) esteja implementada, assinale a opção correta.

a) Nessa organização, ocorre a execução de iterações com ciclo longo, isto é, com duração de alguns meses.

b) No início de cada iteração, a primeira atividade realizada pela equipe de desenvolvimento é produzir o código que será validado através de testes.

c) O refactoring é uma das primeiras atividades realizada no início de cada iteração.

d) Entre as atividades finais de cada iteração, o desenvolvedor escreve casos de teste automatizados, cuja execução verifica se houve a melhoria desejada ou se uma nova funcionalidade foi implementada.

e) Há coerência e inter-relação com os princípios promovidos pela prática da extreme programming (XP).

21. **(CESPE / MPOG / 2013)** Ao realizar o TDD (test-driven development), o programador é conduzido a pensar em decisões de design antes de pensar em código de implementação, o que cria um maior acoplamento, uma vez que seu objetivo é pensar na lógica e nas responsabilidades de cada classe.

22. **(CESPE / MPU – 2013)** Na metodologia TDD, ou desenvolvimento orientado a testes, cada nova funcionalidade inicia com a criação de um teste, cujo planejamento permite a identificação dos itens e funcionalidades que deverão ser testados, quem são os responsáveis e quais os riscos envolvidos.

23. **(CESPE / STF – 2013)** No TDD, o primeiro passo do desenvolvedor é criar o teste, denominado teste falho, que retornará um erro, para, posteriormente, desenvolver o código e aprimorar a codificação do sistema.

24. **(CESPE / TRT-17 – 2013)** TDD consiste em uma técnica de desenvolvimento de software com abordagem embasada em perspectiva evolutiva de seu desenvolvimento. Essa abordagem envolve a produção de versões iniciais de um sistema a partir das quais é possível realizar verificações de suas qualidades antes que ele seja construído.

25. **(CESPE / ALRN – 2013)** Um típico ciclo de vida de um projeto em TDD consiste em:

I. Executar os testes novamente e garantir que estes continuem tendo sucesso.
II. Executar os testes para ver se todos estes testes obtiveram êxito.
III. Escrever a aplicação a ser testada.
IV. Refatorar (refactoring).
V. Executar todos os possíveis testes e ver a aplicação falhar.
VI. Criar o teste.

A ordem correta e cronológica que deve ser seguida para o ciclo de vida do TDD está expressa em:

a) IV – III – II – V – I – VI.
b) V – VI – II – I – III – IV.
c) VI – V – III – II – IV – I.
d) III – IV – V – VI – I – II.
e) III – IV – VI – V – I – II.

26. **(FGV / ALMT – 2013)** Com relação ao desenvolvimento orientado (dirigido) a testes (do Inglês Test Driven Development – TDD), analise as afirmativas a seguir.

I. TDD é uma técnica de desenvolvimento de software iterativa e incremental.

II. TDD implica escrever o código de teste antes do código de produção, um teste de cada vez, tendo certeza de que o teste falha antes de escrever o código que irá fazê-lo passar.

III. TDD é uma técnica específica do processo XP (Extreme Programming), portanto, só pode ser utilizada em modelos de processos ágeis de desenvolvimento de software.

Assinale:

a) Se somente as afirmativas I e II estiverem corretas.
b) Se somente as afirmativas I e III estiverem corretas.
c) Se somente as afirmativas II e III estiverem corretas.
d) Se somente a afirmativa III estiver correta.
e) Se somente a afirmativa I estiver correta.

27. **(FCC / TRT-MG – 2015)** Um analista de TI está participando do desenvolvimento de um software orientado a objetos utilizando a plataforma Java. Na abordagem de desenvolvimento adotada, o código é desenvolvido de forma incremental, em conjunto com o teste para esse incremento, de forma que só se passa para o próximo incremento quando o atual passar no teste. Como o código é desenvolvido em incrementos muito pequenos e são executados testes a cada vez que uma funcionalidade é adicionada ou que o programa é refatorado, foi necessário definir um ambiente de testes automatizados utilizando um framework popular que suporta o teste de programas Java.

A abordagem de desenvolvimento adotada e o framework de suporte à criação de testes automatizados são, respectivamente,

a) Behavior-Driven Development e JTest.
b) Extreme Programming e Selenium.
c) Test-Driven Development e Jenkins.
d) Data-Driven Development and Test e JUnit.
e) Test-Driven Development e JUnit.

28. **(CESPE / TRE-PE – 2017)** O desenvolvimento orientado a testes (TDD):

a) é um conjunto de técnicas que se associam ao XP (extreme programming) para o desenvolvimento incremental do código que se inicia com os testes.

b) agrega um conjunto de testes de integração para avaliar a interconexão dos componentes do software com as aplicações a ele relacionadas.

c) avalia o desempenho do desenvolvimento de sistemas verificando se o volume de acessos/transações está acima da média esperada.

d) averigua se o sistema atende aos requisitos de desempenho verificando se o volume de acessos/transações mantém-se dentro do esperado.

e) testa o sistema para verificar se ele foi desenvolvido conforme os padrões e a metodologia estabelecidos nos requisitos do projeto.

29. **(CESPE / STM – 2018)** O TDD (test driven development) parte de um caso de teste que caracteriza uma melhoria desejada ou nova funcionalidade a ser desenvolvida, de modo a confirmar o comportamento correto e possibilitar a evolução ou refatoração do código.

30. **(CESPE / TRE/PI – 2016)** O TDD (test driven development):

a) apresenta como vantagem a leitura das regras de negócio a partir dos testes, e, como desvantagem, a necessidade de mais linhas de códigos que a abordagem tradicional, o que gera um código adicional.

b) impede que seja aplicada a prática de programação em pares, que é substituída pela interação entre analista de teste, testador e programador.

c) é um conjunto de técnicas associadas ao eXtremme Programing e a métodos ágeis, sendo, contudo, incompatível com o Refactoring, haja vista o teste ser escrito antes da codificação.

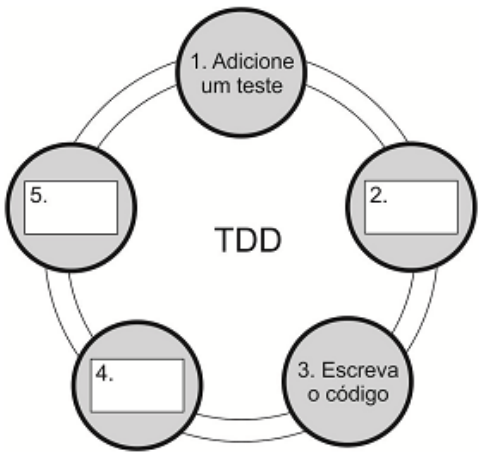
d) refere-se a uma técnica de programação cujo principal objetivo é escrever um código funcional limpo, a partir de um teste que tenha falhado.

e) refere-se a uma metodologia de testes em que se devem testar condições, loops e operações; no entanto, por questão de simplicidade, não devem ser testados polimorfismos.

31. **(UFRRJ / UFRRJ – 2015)** Os testes de unidade têm papel central na metodologia de implementação dirigida por testes, popularizada pelo processo XP e adotada em outros métodos. Esses testes são criados primeiro, exercitando o contrato de cada operação implementada pelos métodos. Em seguida, o código dos métodos é escrito para cumprir os contratos e, portanto, passar nos testes de unidade. Esse cenário corresponde à abordagem

a) TDD.
b) MDD.
c) DDC.
d) MDE.
e) FDD.

32. **(FCC / TRE-PR – 2017)** Considere o ciclo do Test-Driven Development – TDD.



A Caixa:

a) 2. corresponde a "Execute os testes automatizados".
b) 4. corresponde a "Refatore o código".
c) 5. corresponde a "Execute os testes novamente e observe os resultados".

d) 4. corresponde a "Execute os testes automatizados".
e) 5. corresponde a "Faça todos os testes passarem".

33. **(IESES / TRE/MA – 2015)** A respeito da técnica de testes TDD é correto afirmar que:

a) Testa o software com base no comportamento esperado.
b) É uma prática para desenvolvimento de testes unitário que pode utilizar o processo Red-Green-Refactor.
c) Utiliza-se da estrutura Dado, Quando e Então para montar os testes.
d) Prega que os testes devem ser realizados sempre após a implementação ser concluída.

34. **(CESPE / TRE/RS – 2015)** Projeto para o desenvolvimento de software que utilize TDD deve:

a) realizar sprints a cada quinzena.
b) desenvolver pequenos releases.
c) apresentar grande quantidade de testes unitários de código-fonte previamente desenvolvidos.
d) apresentar linguagem de programação estruturada.
e) recomendar a preparação dos testes para que, posteriormente, seja desenvolvido o código.

35. **(FCC / TRE/AP – 2015)** O TDD – Test Driven Development (Desenvolvimento orientado a teste):

a) é parte das metodologias ágeis UP – Unified Process e XP – Extreme Programming, tendo sido criado para ser usado em metodologias que respeitam os 4 princípios do Manifesto Ágil.

b) transforma o desenvolvimento, pois deve-se primeiro implementar o sistema antes de escrever os testes. Os testes são utilizados para facilitar no entendimento do projeto e para clarear o que se deseja em relação ao código.

c) baseia-se em um ciclo simples: escreve-se um código -> cria-se um teste para passar no código -> refatora-se.

d) propõe a criação de testes que validem o código como um todo para reduzir o tempo de desenvolvimento.

e) beneficia-se de testes que seguem o modelo FIRST: F (Fast) I (Isolated) R (Repeatable) S (Self-verifying) T (Timely).

36. **(CESPE / TRE/TO – 2017)** O TDD (Test-Driven Development), que vem sendo adotado para testar os projetos de software,

a) utiliza os testes de caixa preta antes da entrega do software.
b) agiliza os testes por amostragem sem compatibilidade retroativa.
c) cobre amplamente os testes unitários.

d) escreve o teste antes da codificação do software.
e) realiza refactoring antes de escrever a aplicação a ser testada.

37. **(FGV / IBGE – 2016)** O Desenvolvimento Orientado a Testes (TDD) é um método de desenvolvimento criado e disseminado por Kent Beck em seu livro "Test-driven development". O método define regras, boas práticas e um ciclo de tarefas com 3 etapas: a etapa vermelha, a etapa verde e a etapa de refatoração, ilustrado na imagem abaixo:

Com relação às regras e boas práticas de TDD e ao seu ciclo, é correto afirmar que:

a) pode-se escrever testes que não compilam na etapa vermelha;
b) na etapa verde deve-se escrever código que testa uma funcionalidade a fundo de forma criteriosa e detalhada;
c) código novo só é escrito se um teste automatizado passar;
d) a duplicação é tolerada na etapa de refatoração;
e) é uma boa prática de TDD iniciar o desenvolvimento do código de uma funcionalidade e, logo em seguida, testá-la.

38. **(IADES / EBSERH – 2013)** Assinale a alternativa que não corresponde a uma das fases do processo de desenvolvimento, dirigido a testes (TDD).

a) Executar o teste, com os outros testes implementados, que rodarão e fornecerão o resultado de que o software está sem problemas.
b) Escrever o teste para a funcionalidade e implementação.
c) Realizar a identificação do incremento de funcionalidade.
d) Implementar a funcionalidade e executar novamente o teste.
e) Implementar a próxima parte da funcionalidade, após todos os testes terem sido executados, com sucesso

39. **(PR-4 UFRJ / UFRJ – 2018)** O ciclo do TDD - Test Driven Development, ou, em português, Desenvolvimento Guiado por Testes consiste em:

a) implementar teste unitário falho, tornar o teste bem-sucedido e refatorar.
b) implementar a funcionalidade, executar teste unitário e refatorar.

c) implementar teste unitário falho, refatorar e tornar o teste bem-sucedido.
d) implementar a funcionalidade, refatorar e tornar o teste bem-sucedido.
e) refatorar, executar teste unitário e implementar a funcionalidade.

40. **(CESPE / STJ – 2015)** Um dos passos executados no ciclo de atividades do processo TDD é a criação de novos testes para as falhas encontradas no código original, sem alteração deste.

41. **(CS-UFG / AL-GO – 2015)** O desenvolvimento dirigido a testes (TDD, do Inglês Test-Driven Development) é uma abordagem de desenvolvimento de software na qual se intercalam testes e desenvolvimento de código. Uma das características da abordagem TDD é:

a) a sua utilidade no desenvolvimento de softwares novos.
b) o maior custo associado aos testes de regressão.
c) a redução da importância da automatização dos testes.
d) a sua adequação a processos de software sequenciais.

42. **(UECE-CEV / FUNCEME – 2018)** Test-driven Development (TDD) é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código (Sommerville, I. Engenharia de Software, 9a edição, 2011).

A respeito do TDD, é correto afirmar que:

- a) consiste em um processo iterativo que se inicia escrevendo um código de uma funcionalidade do sistema e, logo em seguida, testa-o para saber se a implementação foi correta.
b) apesar de útil, não diminui o custo de testes de regressão do sistema.
c) sua utilização elimina a necessidade de testes de validação do sistema, uma vez que ele já foi testado incrementalmente.
d) apesar de ter sido apresentado como parte dos métodos ágeis, também pode ser usado em outros processos de desenvolvimento de software.

43. **(FAUGRS / UFRGS – 2018)** _____ é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código. Essencialmente, desenvolve-se um código de forma incremental em conjunto com um teste para este incremento. Não se avança para o próximo incremento até que o código desenvolvido passe no teste. Essa abordagem foi introduzida como parte de métodos ágeis, mas pode ser também usada em processos de desenvolvimento dirigido a planos.

Assinale a alternativa que preenche corretamente a lacuna do texto acima.

a) Desenvolvimento Guiado por Testes (TDD)
b) Desenvolvimento em Espiral
c) Engenharia Dirigida a Modelos (MDD)
d) Rational Unified Process (RUP)
e) Teste de Sistema

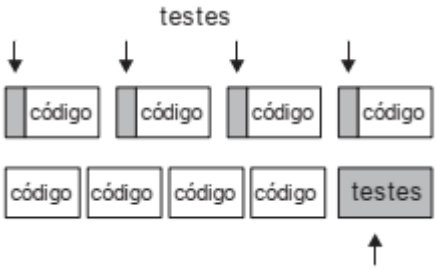
44. (VUNESP / TCE-SP – 2015) No Desenvolvimento Orientado a Testes (TDD), os casos de teste que definem o recurso a ser implementado devem ser elaborados:

a) assim que o código do teste estiver pronto.
b) antes de o código do recurso ser desenvolvido.
c) após o código do recurso ter sido completamente documentado.
d) simultaneamente com o desenvolvimento do código do recurso.
e) somente se o código do recurso apresentar erros.

45. (IBFC / EMBASA – 2017) No Ciclo de Desenvolvimento do TDD (Test-Driven Development), utiliza-se a estratégia que aplica três palavras-chaves (em inglês), que é denominada:

a) Red, Green, Refactor
b) White, Gray, Black
c) White, Black, Refactor
d) Green, Yellow, Red

46. (FCC / CREMESP – 2016) Considere a figura abaixo que apresenta duas abordagens de teste.



A figura:

a) ilustra as duas fases do TDD, que correspondem a escrever pequenos testes e testá-los no final.
b) mostra o ciclo conhecido como Vermelho-Verde-Refatora.
c) apresenta a diferença entre testes automatizados e testes manuais no XP.
d) mostra que um desenvolvedor que pratica TDD tem mais feedbacks do que um que escreve testes ao final.
e) evidencia que TDD é impraticável, pois o desenvolvedor gasta muito tempo escrevendo código de testes.

47. (CESPE / ANATEL – 2014) Em se tratando de desenvolvimento de softwares dirigidos a testes (TDD), a execução dos testes é realizada antes da implementação da funcionalidade.

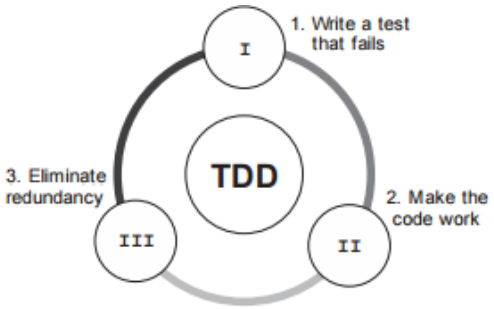
48. (CESPE / TC/DF – 2014) No TDD, o refatoramento do código deve ser realizado antes de se escrever a aplicação que deve ser testada.

49. **(CESPE / STJ – 2015)** No método de desenvolvimento TDD (Test Driven Development), o desenvolvedor escreve primeiro um caso de teste e, posteriormente, o código.

50. **(CESPE / MPE-PI – 2018)** O TDD possibilita o desenvolvimento de softwares fundamentado em testes. O ciclo de desenvolvimento do TDD segue os seguintes passos:

- escrever um teste que inicialmente não passa;
- adicionar uma nova funcionalidade do sistema;
- fazer o teste passar;
- realizar a integração contínua do código;
- escrever o próximo teste.

51. **(FCC / Pref. de Teresina/PI – 2016)** O Test Driven Development – TDD é uma das práticas sugeridas na eXtreme Programming – XP, onde o programador escreve o teste antes de escrever o código. O ciclo de desenvolvimento utilizando TDD é mostrado abaixo.



Considere:

I. Etapa inicial, onde se escreve um teste que falha, para alguma funcionalidade que ainda será Escrita.
II. Já com o teste criado, é o momento de executar o teste.
III. Eliminar códigos redundantes, remover acoplamentos, enfim, identificar pontos de melhoria no código.

As etapas I, II e III são, respectivamente,

a) Iniciação, Execução e Controle.
b) Red, Green e Refactor.
c) Iniciação, Atuação e Otimização.
d) Plan, Do e Check.
e) Planejamento, Execução e Melhoria.

52. **(FAUGRS / BANRISUL – 2018)** Considere as ações abaixo, executadas em desenvolvimento orientado a testes, Test-Driven Design (TDD).

I - Escrever código de teste.

II - Verificar se o teste falha.
III - Escrever código de produção.
IV - Executar teste até passar (reescrevendo o código de produção, se for necessário, até que o teste passe).
V - Refatorar código de produção e/ou de teste para melhorá-lo.

Considerando que se deseja incluir um novo caso de teste, assinale a alternativa que apresenta a sequência de ações que devem obrigatoriamente ocorrer para essa inclusão, segundo o TDD.

a) I, III e IV.
b) III, I e IV.
c) I, II, III e IV.
d) I, III, IV e V.
e) I, II, III, IV e V.

53. **(FGV / IBGE – 2017)** Test Driven Development (TDD) é uma prática muito utilizada no processo de desenvolvimento de sistemas computacionais. Analise as afirmativas a seguir sobre o uso da prática de TDD:

I. Tornam os testes de regressão mais demorados porque o desenvolvedor precisará fazer testes manuais várias vezes por dia.

II. Garante que os requisitos do sistema sejam atendidos porque o desenvolvedor escreverá o código de testes sempre que acabar a implementação do código do sistema.

III. Ajuda o desenvolvedor a escrever código de qualidade porque ele gastará parte do seu tempo escrevendo código de testes.

Está correto o que se afirma em:

a) somente I;
b) somente II;
c) somente III;
d) somente II e III;
e) I, II e III.

54. **(CESPE / ANATEL – 2014)** Na atividade de TDD (test-driven development), a escrita de teste primeiro define implicitamente tanto uma interface quanto uma especificação do comportamento para a funcionalidade que está sendo desenvolvida, estando, entretanto, a viabilidade do uso dessa abordagem limitada aos processos de desenvolvimento de software que seguem as práticas ágeis.

55. **(CESPE / TRE/MT – 2015)** Considere as seguintes etapas de um processo do tipo desenvolvimento orientado a testes (TDD).

I Implementar funcionalidade e refatorar.
II Identificar nova funcionalidade.
III Executar o teste.
IV Escrever o teste.
V Implementar a próxima parte da funcionalidade.

Assinale a opção que apresenta a sequência correta em que essas etapas devem ser realizadas.

a) I; IV; III; II; V
b) IV; III; II; I; V
c) I; IV; II; III; V
d) II; IV; III; I; V
e) IV; II; III; I; V

56. **(FCC / SEFAZ-SC – 2018)** O Test-Driven Development (TDD) é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código. As etapas do processo fundamental de TDD são mostradas abaixo em ordem alfabética:

I. Escrever um teste para a funcionalidade identificada e implementá-lo como um teste automatizado.

II. Executar o teste, junto com os demais testes já implementados, sem implementar a nova funcionalidade no código.

III. Identificar e implementar uma outra funcionalidade, após todos os testes serem executados com sucesso.

IV. Identificar uma nova funcionalidade pequena para ser incrementada com poucas linhas em um código.

V. Implementar a nova funcionalidade no código e reexecutar o teste.

VI. Refatorar o código com melhorias incrementais até que o teste execute sem erros.

VII. Revisar a funcionalidade e o teste, caso o código execute sem falhar.

Considerando o item IV a primeira etapa e o item III a última etapa, a sequência intermediária correta das etapas do processo é:

a) I – II – VII – V e VI.
b) I – V – II – VII e VI.
c) I – VI – V – VII e II.
d) V – I – II – VII e VI.

e) V – I – VI – VII e II.

GABARITO – DIVERSAS BANCAS

1. LETRA A
2. ERRADO
3. CORRETO
4. ERRADO
5. LETRA B
6. ERRADO
7. LETRA C
8. LETRA B
9. LETRA A
10. LETRA D
11. ERRADO
12. CORRETO
13. ERRADO
14. LETRA B
15. LETRA A
16. CORRETO
17. ERRADO
18. ERRADO
19. CORRETO
20. LETRA E
21. ERRADO
22. CORRETO
23. CORRETO
24. CORRETO
25. LETRA C
26. LETRA A
27. LETRA E
28. CORRETO
29. CORRETO
30. LETRA D
31. LETRA A
32. LETRA D
33. LETRA B
34. LETRA E
35. LETRA E
36. LETRA D
37. LETRA A
38. LETRA A
39. LETRA A
40. ERRADO
41. LETRA A
42. LETRA D
43. LETRA A
44. LETRA B
45. LETRA A
46. LETRA D
47. CORRETO
48. ERRADO
49. CORRETO
50. ERRADO
51. LETRA B
52. LETRA C
53. LETRA C
54. ERRADO
55. LETRA D
56. LETRA A

ESTRATÉGIA FLASHCARDS

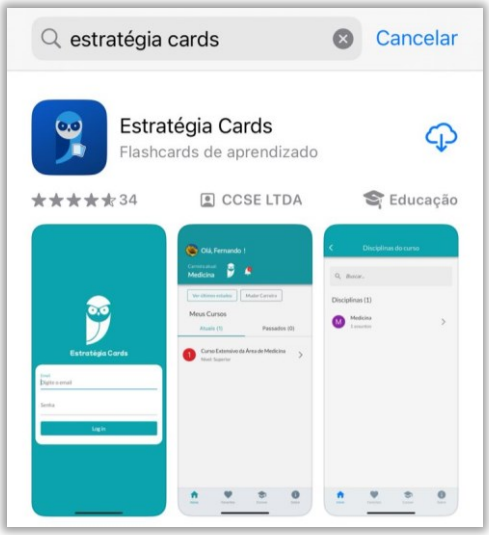
📚 Você tem dificuldade de estudar, memorizar e revisar os conteúdos que estuda em nossas aulas? Então nós temos a ferramenta perfeita para você!

Apresentamos o **Estratégia Cards**: app de flashcards que vai revolucionar sua forma de **estudar** e **revisar** conteúdos de provas de concurso público. Com nossa tecnologia inovadora e interface amigável, você dominará os tópicos mais complexos de maneira eficiente e divertida.

✨ Recursos do Estratégia Cards:

Curadoria de Flashcards	Flashcards criados e revisados por professores especializados em cada área, com qualidade e voltados para concursos públicos.
Flashcards Personalizados	Crie seus próprios flashcards, cobrindo os principais tópicos e matérias dos concursos públicos.
Repetição Espaçada	Técnica de aprendizagem que envolve revisar informações em intervalos crescentes para melhorar a retenção de longo prazo e combater o esquecimento.
Estatísticas Personalizadas	Visualize graficamente o percentual de acertos, erros ou dúvidas dos decks estudados.
Modo Offline	Estude em qualquer lugar, mesmo sem conexão à internet, fazendo o download dos decks.
Estudo por Áudio	*Está dirigindo ou fazendo esteira e quer continuar estudando?* Basta utilizar a opção "Escutar".
Decks Favoritos	Você pode escolher decks específicos como favoritos e visualizá-los em uma aba separada do app.
Opções de Estudo	Você poderá estudar todos os cards de um deck; ou apenas os que você errou; ou apenas os que você não estudou ainda; entre outras opções.

E como eu consigo baixar?



É muito fácil! Basta pesquisar por "Estratégia Cards" na loja oficial do seu smartphone.

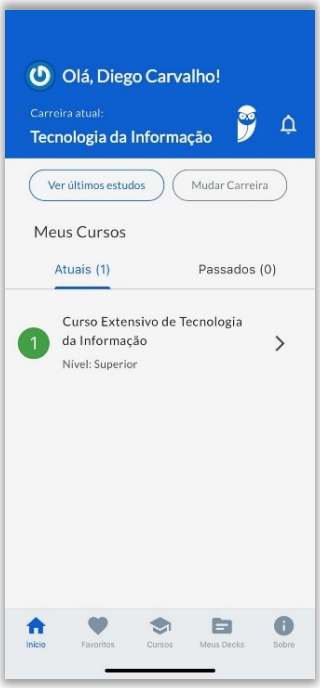
Se você tiver um Android, basta acessar a **Google Play**;

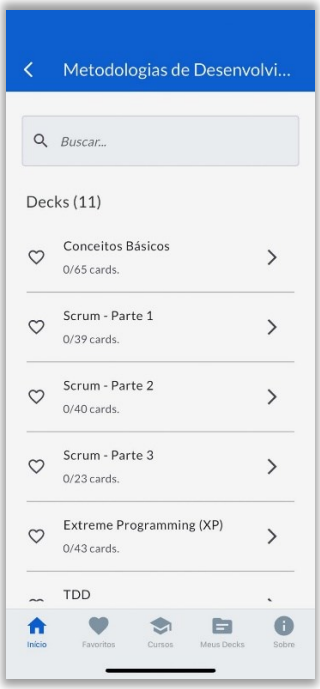
Se for tiver um iPhone, basta acessar a **App Store (iOS)**.

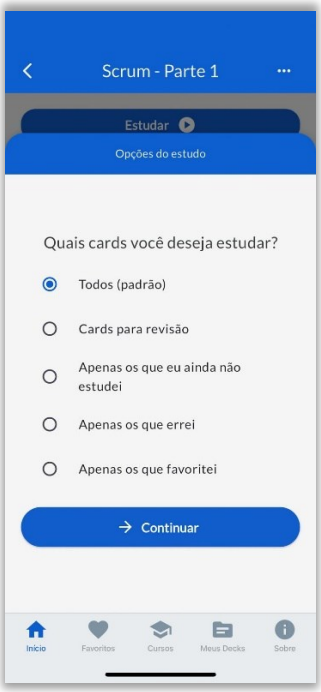
E para acessar?

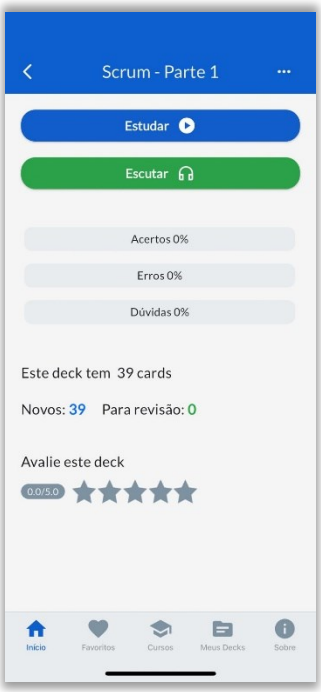
Para acessar, basta ter uma conta no Estratégia Concursos. Em seguida, utilize suas credenciais de login e senha para acessar o aplicativo. Por fim, acessa a carreira de Tecnologia da Informação.

Como utilizar o app:









TESTES DE SOFTWARE

Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

Galera, existe uma frase que eu gosto muito que diz: "***A melhor maneira de construir confiança é tentar destruí-la***". Calma que eu não vou filosofar, isso é apenas um subterfúgio para que nós entendamos a importância de um teste. As imagens abaixo mostram testes de impacto automobilístico, que consistem no impacto de veículos automotores contra barreiras indeformáveis ou deformáveis.

Ela tem por objetivo avaliar a segurança automotiva para verificar se cumprem determinadas normas de segurança de proteção à colisão em situações de acidente de trânsito. Ora, quanto mais testes que buscam destruir o carro eu faço, mais confiança eu construo sobre o carro. *Por que?* **Porque, a cada teste realizado, eu identifico as falhas e as corrijo – sempre com o intuito de construir o carro mais seguro possível para os passageiros.**

Com testes de software, o processo é bastante similar! **O teste de software é o processo de executar um software com dois objetivos principais**: primeiro, demonstrar ao desenvolvedor e ao cliente que o software atende aos requisitos especificados; segundo, descobrir falhas ou defeitos no software que apresente comportamento incorreto, não desejável ou em não conformidade com sua especificação.

A primeira meta conduz ao Teste de Validação, no qual você espera que o sistema seja executado corretamente em um dado conjunto de casos de teste que refletem o uso esperado do sistema. A segunda meta conduz ao teste de defeitos, no qual são projetados casos de teste para expor defeitos. Os casos de teste podem ser obscuros e não precisam refletir como o sistema é usado normalmente.

(FCC – TRF4 – Analista Judiciário – Item III) A única meta do teste de software é descobrir falhas ou defeitos no software que apresenta comportamento incorreto, não desejável ou em não conformidade com sua especificação.

_____

Comentários: são duas metas: demonstrar ao desenvolvedor e ao cliente que o software atende aos requisitos; e descobrir falhas ou defeitos no software que apresenta comportamento incorreto, não desejável ou em não conformidade com sua especificação (Errado).

Professor, testes resolvem todos os problemas de qualidade? Não! **Testes de Software não podem demonstrar – por exemplo – que um software é livre de defeitos ou que ele se comportará conforme especificado em todas as circunstâncias existentes**. É sempre possível que um teste importante tenha sido ignorado. Já dizia Edsger Dijkstra: "*Os testes podem somente mostrar a presença de erros, não sua ausência*". *Captaram?*

A grande lance dos testes é convencer os desenvolvedores e os clientes do sistema de que o software é bom o suficiente para o uso operacional. O teste de software é um processo voltado a atingir a confiabilidade do software. Se um programa falhar frequentemente e repetidas vezes, pouco importa se outros fatores de qualidade de software são ou não aceitáveis. *Legal, mas qual é a definição de confiabilidade de software?*

A confiabilidade de software é definida em termos estatísticos como **a probabilidade de operação sem falhas de um programa de computador em um dado ambiente por um determinado tempo.** Diferentemente de outros fatores de qualidade, pode ser medida diretamente e estimada usando-se dados históricos e de desenvolvimento. Exemplo: estima-se que o programa X tenha uma confiabilidade de 0,999 depois de decorridas oito horas de processamento.

Em outras palavras, se o programa X tiver de que ser executado 1.000 vezes e precisar de um total de oito horas de tempo de processamento (tempo de execução), é provável que ele opere corretamente 999 vezes e falhe apenas uma vez! *Bacana?* Bem, para o Teste de Validação, um teste bem-sucedido é aquele em que o sistema funciona corretamente; para o Teste de Defeitos, **um teste bem-sucedido é o que expõe um defeito que causa o funcionamento incorreto.**

Já falamos tanto de testes de software, mas qual é a sua definição? Como é possível de prever, há diversas definições diferentes de diversos autores. **Myers, por exemplo, diz que é o *processo de executar um determinado software com a intenção de encontrar defeitos***. Já a IEEE 729 define como o *processo formal de avaliar um sistema ou componente por meios manuais ou automáticos para verificar se ele satisfaz os requisitos especificados*.

O famoso Glossário ISTQB (International Software Testing Qualifications Board) conceitua atividades do ciclo de vida, estáticas ou dinâmicas, voltadas para o planejamento, preparação e avaliação de produtos de software e produtos de trabalho relacionados **a fim de determinar se eles satisfazem os requisitos especificados e demonstrar que estão aptos para sua finalidade e detecção de defeitos**.

Interessante! *No entanto, o que define um bom teste?* Bem, existem quatro características que nos ajudam a classificar um teste como bom. São elas:

CARACTERÍSTICAS	DESCRIÇÃO
UM BOM TESTE TEM ALTA PROBABILIDADE DE ENCONTRAR DEFEITOS	Para atingir esse objetivo de encontrar defeitos, o testador deve entender o software e tentar desenvolver uma imagem mental de como o software pode falhar. O ideal é que as classes de falhas sejam investigadas.
UM BOM TESTE NÃO É REDUNDANTE	O tempo e os recursos de teste são limitados. Não tem sentido realizar um teste que tenha a mesma finalidade de outro teste. Cada teste deve ter uma finalidade diferente (mesmo que seja sutilmente diferente).
UM BOM TESTE DEVERÁ SER "O MELHOR DA RAÇA"	Em um grupo de testes com finalidades similares, as limitações de tempo e recursos podem induzir à execução de apenas um subconjunto desses testes. Nesses casos, deverá ser usado o teste que tenha a maior probabilidade de revelar uma classe inteira de erros.
UM BOM TESTE NÃO DEVE SER NEM MUITO SIMPLES NEM MUITO COMPLEXO	Embora seja possível combinar algumas vezes uma série de testes em um caso de teste, os possíveis efeitos colaterais associados com essa abordagem podem mascarar erros. Em geral, cada teste deve ser executado separadamente.

(DPE/SP – 2013) O teste de software constitui-se em uma etapa importante no ciclo de desenvolvimento de software. Uma das características mais importantes de um conjunto de testes de software, adequadamente planejados, é:

a) provar a correção integral no programa sob teste.

b) ter alta probabilidade de detectar erros no programa sob teste.

c) ter grande redundância, a fim de testar mais de uma vez cada linha do programa sob teste.

d) ser de alta complexidade, pois assim pode-se cobrir todo o programa sob teste com apenas um teste.

e) ser ocultado da equipe de desenvolvimento do software, pois esta pode querer impedir sua aplicação.

Comentários: (a) Errado, testes exaustivos são impossíveis; (b) Correto, deve-se agrupar os defeitos mais sensíveis para aumentar a probabilidade de detectar erros no programa sob teste; (c) Errado, caso os mesmos testes sejam aplicados repetidamente, em determinado momento eles deixam de ser úteis; (d) Errado, testes de software não devem ser muito simples nem muito complexos; (e) Errado, esse item não faz qualquer sentido lógico (Letra B).

Ao longo de diversos anos, a Engenharia de Software evoluiu bastante, de modo a sugerir alguns princípios que guiam os Testes de Software. Ao todo, são sete princípios fundamentais:

PRINCÍPIOS FUNDAMENTAIS	DESCRIÇÃO
TESTES DEMONSTRAM A PRESENÇA DE DEFEITOS...	Um teste pode demonstrar a presença de defeitos, mas não pode provar que eles não existem. Ele reduz a probabilidade de que os defeitos permaneçam, mas mesmo se nenhum defeito for encontrado não quer dizer que ele não os tenha.
TESTES EXAUSTIVOS SÃO IMPOSSÍVEIS...	Testar todas as combinações de entradas e pré-condições é inviável, exceto para casos triviais. Em vez de realizar testes exaustivos, os riscos e prioridades são levados em consideração para dar foco aos esforços de teste.
TESTE O MAIS BREVE POSSÍVEL (ANTECIPADO)...	Os defeitos encontrados nas fases iniciais do processo de desenvolvimento de software são mais baratos de serem corrigidos do que aqueles encontrados já em fase produção. Há, inclusive, técnicas de testes antes mesmo da implementação.
AGRUPEM OS DEFEITOS MAIS SENSÍVEIS...	Seguindo o Princípio de Pareto, 80% dos defeitos são causados por 20% do código. Ao identificar essas áreas sensíveis, os testes podem priorizá-las, de forma a ter alta probabilidade de encontrar defeitos.
PARADOXO DO PESTICIDA...	Caso os mesmos testes sejam aplicados repetidamente, em determinado momento eles deixam de ser úteis, ou seja, não conseguem encontrar nenhum novo defeito. Por isso, os testes precisam ser revisitados com frequência.
TESTES DEPENDEM DO CONTEXTO...	Os testes devem ser elaborados de acordo com o contexto de utilização do software. Ex: um sistema bancário deve ser testado de maneira diferente de uma rede social. Assim como testes de aplicação web têm foco diferente do desktop.
AUSÊNCIA DE DEFEITOS É UMA ILUSÃO	Identificar e corrigir os problemas de um software não garantem que ele está pronto. Os testes foram elaborados para identificar todas as possíveis falhas? O sistema atende às necessidades e expectativas dos usuários? Logo, há outros fatores!

(TCU – 2015) Os casos de testes são especificações acerca das entradas para o teste e da saída esperada e englobam, também, uma declaração do que está sendo testado. Devido ao tamanho do espaço de possibilidades de teste, a geração automática exaustiva de casos de testes que exploram todas as entradas e saídas para qualquer configuração de teste é impossível ou computacionalmente intratável.

_____

Comentários: casos de teste realmente são especificações das entradas para o teste e da saída esperada e englobam, também, uma declaração do que está sendo testado. A geração automática ou não de casos de teste que exploram todas as entradas e saídas para qualquer configuração de teste é impossível ou computacionalmente intratável, visto que um dos princípios fundamentais é o de que testes exaustivos são impossíveis, isto é, testar todas as combinações de entradas e pré-condições é inviável, exceto para casos triviais. Assim, é realmente inviável testar todo o espaço amostral de testes de software, exceto para casos triviais – mesmo por meio de ferramentas automatizadas (Correto).

(TRT/MG – 2015) Um técnico de TI do Tribunal pretende prestar exame de certificação de teste de software e necessita conhecer os sete princípios do Teste CFTL. Após um tempo de estudo ele observou o seguinte:

Pode ocorrer o fato de um mesmo conjunto de testes que são repetidos várias vezes não encontrar novos defeitos após um determinado momento. Para superar esta condição, os casos de testes necessitam ser frequentemente revisados e atualizados. 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.

Este princípio é corretamente denominado:

a) Ilusão da ausência de erros.
b) Agrupamento de defeitos.
c) Teste antecipado.
d) Dependência de contexto.
e) Paradoxo do pesticida.

Comentários: trata-se do Paradoxo do Pesticida, isto é, caso os mesmos testes sejam aplicados repetidamente, em determinado momento eles deixam de ser úteis, ou seja, não conseguem encontrar nenhum novo defeito – por isso, os testes precisam ser revisitados com frequência (Letra E).

(PROCEM/PA – 2014) O teste é parte fundamental no ciclo de vida de um software. Seus princípios devem servir como um guia geral, tanto para testadores quanto para desenvolvedores. Afinal, ambos participam efetivamente do processo de amadurecimento do sistema. Assinale a opção que apresenta corretamente três dos sete princípios fundamentais do teste de software.

a) Testes apontam a presença de falhas – Testes dependem do contexto – Testes apontam os custos

b) Depende dos requisitos – Testes apontam a presença de falhas – Paradoxo do pesticida

c) Teste antecipado – Testes apontam os custos – Ausência de erros é uma ilusão

d) Paradoxo do pesticida – Teste antecipado – Testes dependem do contexto

e) Ausência de erros é uma ilusão – Agrupamento de defeitos – Dependem dos requisitos

---

Comentários: (a) Errado, o terceiro item não é um dos princípios fundamentais; (b) Errado, o primeiro item não é um dos princípios fundamentais; (c) Errado, o segundo item não é um dos princípios fundamentais; (d) Correto, todos são princípios fundamentais; (e) Errado, o terceiro item não é um dos princípios fundamentais (Letra D).

O objetivo do teste é encontrar erros, e um bom teste é aquele que tem alta probabilidade de encontrar um erro. Ao mesmo tempo, os próprios testes devem ter uma série de características que permitam atingir o objetivo de encontrar o maior número de erros com o mínimo de esforço. Dessa forma, um engenheiro de software deve projetar e implementar um sistema ou produto tendo em mente a *testabilidade. O que seria isso, Diego?*

Cara, a testabilidade é definida como a facilidade com que um programa de computador pode ser testado. As seguintes características levam a um software testável:

CARACTERÍSTICA	DESCRIÇÃO
OPERABILIDADE	Grosso modo, podemos dizer que "*quanto melhor funciona, mais eficientemente pode ser testado*".
OBSERVABILIDADE	Grosso modo, podemos dizer que "*o que você vê é o que você testa*".
CONTROLABILIDADE	Grosso modo, podemos dizer que "*quanto melhor você pode controlar o software, mais o teste pode ser automatizado e otimizado*".
DECOMPONIBILIDADE	Grosso modo, podemos dizer que "*controlando o escopo, podemos isolar problemas mais rapidamente e realizar testes mais inteligentes*".
SIMPLICIDADE	Grosso modo, podemos dizer que "*quanto menos houver a testar, mais rapidamente podemos testá-lo*".
ESTABILIDADE	Grosso modo, podemos dizer que "*quanto menos modificações, menos interrupções no teste*".
COMPREENSIBILIDADE	Grosso modo, podemos dizer que "*quanto mais informações temos, de forma mais inteligente vamos testar*".

(FIOCRUZ – 2010) No que diz respeito aos sistemas de software, o objetivo do teste é encontrar erros, sendo um teste aquele que tem alta probabilidade de encontrar um erro. Assim, um engenheiro de software deve projetar e implementar um sistema ou um produto baseado em computador com "testabilidade" em mente. Ao mesmo tempo, os testes devem exibir um conjunto de características que atinge o objetivo de encontrar a maioria dos erros com um mínimo de esforço. Dentre as características que levam a um software testável, uma pode ser resumida pela frase "Quanto melhor funciona, mas eficientemente pode ser testado".

Se um sistema é projetado e implementado com qualidade em mente, poucos defeitos vão bloquear a execução dos testes, permitindo que o teste progrida sem problemas. Essa característica é definida como:

a) estabilidade.
b) simplicidade.
c) operabilidade.
d) controlabilidade.
e) observabilidade.

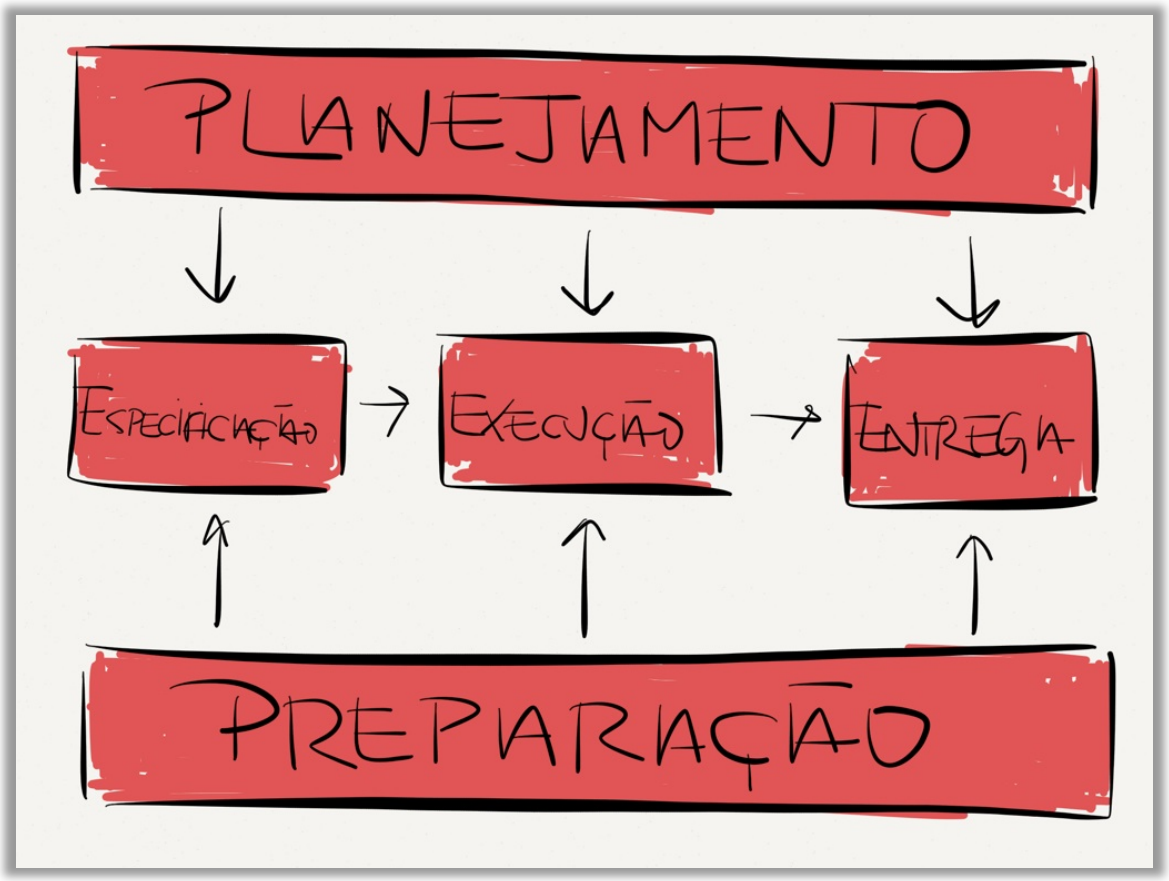
_____

Comentários: a questão trata da operabilidade - "*quanto melhor funciona, mais eficientemente pode ser testado*" (Letra C).

Processo de Teste

INCIDÊNCIA EM PROVA: BAIXA

O processo de testes é caracterizado pela execução das principais etapas da atividade de testes e de suas subetapas. O objetivo dos processos de teste é direcionar e indicar a próxima atividade a ser feita. **Esses processos são conjuntos de boas práticas que orientam para chegar ao objetivo da melhor maneira e servem para minimizar os riscos causados por defeitos originados no processo de desenvolvimento.** Vejamos como é esse processo de testes...



ETAPAS DO PROCESSO	DESCRIÇÃO
PLANEJAMENTO	Nesta etapa, elaboram-se o Projeto de Testes e o Plano de Testes. Ela acompanha todo o processo de teste, por meio de atividades como captação de requisitos, planejamento do projeto, análise de riscos e preparação de ambiente de testes.
PREPARAÇÃO	nesta etapa, organiza-se o ambiente de testes (infraestrutura, equipamentos, hardware, software, pessoal capacitado, ferramentas e massa de testes adequadas) para que os testes sejam executados conforme planejados.
ESPECIFICAÇÃO	nesta etapa, temos as atividades de elaborar e revisar casos de testes e roteiros de testes (scripts). Esse último descreve a relação dos casos de testes e a previsão de execução dos testes.

EXECUÇÃO	Nesta etapa, testes são executados conforme roteiros estabelecidos para os testes. Executa-se sempre que ocorrem mudanças na aplicação e analisam-se os testes executados com sucesso e com defeito – os resultados obtidos são registrados.
ENTREGA	Nesta etapa, o projeto é finalizado, registra-se toda a documentação e relatam-se todas as incidências relevantes à melhoria do processo em um relatório de conformidades e não-conformidades – por fim, a documentação gerada é arquivada.

(MEC – 2011) Em um projeto de teste, após ser concluída a etapa de execução, inicia-se a etapa de entrega, cujos produtos incluem o relatório de não conformidades.

_____

Comentários: a etapa de entrega realmente se inicia após a etapa de execução, gerando um relatório de conformidades e não-conformidades (Correto).

(MEC – 2011) A elaboração de scripts para teste é um dos insumos necessários na etapa de planejamento.

_____

Comentários: a elaboração de scripts para teste é um dos insumos necessários na etapa de execução (Errado).

(MEC – 2011) Na etapa de especificação, ocorrem a elaboração e a revisão dos casos de testes.

_____

Comentários: realmente temos as atividades de elaborar e revisar casos de testes e roteiros de testes na etapa de especificação (Correto).

(MEC – 2011) Na etapa de execução, os roteiros dos testes são insumos necessários, que descrevem a relação dos casos de testes e a previsão de execução.

_____

Comentários: os roteiros de testes (scripts) realmente são insumos da etapa de execução e – de fato – descrevem a relação dos casos de teste e a previsão de execução (Correto).

(MEC – 2011) A implementação do ambiente de teste deve ser feita durante a preparação do processo de teste, devendo o responsável pela execução ser identificado na matriz de responsabilidades.

_____

Comentários: organiza-se o ambiente de testes (infraestrutura, equipamentos, hardware, software, pessoal capacitado, ferramentas e massa de testes adequadas) justamente na etapa de preparação de testes (Correto).

Plano de Testes

INCIDÊNCIA EM PROVA: BAIXA

O processo de teste de software pode produzir diversos artefatos, dentre eles dois muito importantes: Plano de Testes e Casos de Testes. O Plano de Testes apresenta o planejamento para execução das atividades de testes, apresentando seu escopo, métodos empregados, prazo estimado, nível de qualidade esperado, expectativa de capacidade, recursos que serão utilizados como ferramenta de apoio, e métricas e formas de acompanhamento do processo.

Como se trata de um documento gerencial, ele contempla a elaboração de um cronograma contendo todas as atividades e responsáveis por sua execução, podendo abranger casos de testes desde as primeiras entregas até o sistema completo. **Ele busca definir e comunicar a intenção do esforço de teste em determinada programação.** Como em outros documentos de planejamento, o principal objetivo é ganhar a aceitação e aprovação dos envolvidos no esforço de teste.

Para isso, o documento deve evitar informações que não serão compreendidas ou que serão consideradas irrelevantes pelos envolvidos. Planos de teste podem variar de organização para organização, mas em geral podem apresentar atributos apresentados na tabela a seguir. **Não é necessário decorar essas informações, basta entender que o Plano de Testes busca descrever toda informação que possa ser necessária para a execução das atividades de teste.**

SEÇÃO	CONTEÚDO
INTRODUÇÃO	Contém uma identificação do projeto, descrição dos objetivos do documento, o público ao qual ele se destina e escopo do projeto a ser desenvolvido.
REQUISITOS	Descreve em linhas gerais o conjunto de requisitos a serem testados no projeto a ser desenvolvido, comunicando o que deve ser verificado.
ESTRATÉGIAS E FERRAMENTAS	Apresenta um conjunto de tipos de testes a serem realizados, respectivas técnicas empregadas e critério de finalização de teste. Além disso, é listado o conjunto de ferramentas utilizadas.
EQUIPE E INFRAESTRUTURA	Contém descrição da equipe e da infraestrutura utilizada para o desenvolvimento das atividades de testes, incluindo: pessoal, equipamentos, software de apoio, materiais, etc.
CRONOGRAMA DE ATIVIDADES	Contém uma descrição de marcos importantes (*milestones*) das atividades (incluindo as datas de início e fim da atividade).
DOCUMENTAÇÃO COMPLEMENTAR	Apresenta-se uma relação dos documentos pertinentes ao projeto.

(MPE/MA – 2011) O propósito de um plano de testes é descrever o escopo, os recursos, a abordagem e o tempo alocado para as atividades de teste. Identifica os itens e funcionalidades a serem testados, os responsáveis e os riscos.

_____

Comentários: todos esses realmente são atributos que podem ser contemplados em um plano de teste (Correto).

Casos de Testes

INCIDÊNCIA EM PROVA: MÉDIA

O Caso de Teste é um artefato que contém um conjunto de condições e entradas utilizadas para testar um software. Em geral, possui os seguintes atributos: identificador, itens constantes no teste, especificação de entrada, especificação de saída, definição do ambiente necessário, necessidades especiais e dependências de outros casos de testes. Eu sei que esse assunto está muito abstrato para vocês, mas vamos tentar simplificar...

Imagine que uma fabricante de automóveis resolva revolucionar o mercado e construir um protótipo de carro voador. **Durante a construção, é necessário realizar diversos testes para que – ao final – o carro realmente possa voar sem problemas.** Logo, é chamada uma equipe de testadores. Esses caras vão construir um plano de testes que vai descrever em linhas gerais toda informação que possa ser necessária para a execução das atividades de teste.

Após isso, é necessário descrever casos de teste, ou seja, um conjunto de condições e entradas utilizadas para testar o carro. Logo, esses casos de teste deverão apresentar um conjunto de ações e resultados esperados para elas, sendo que as ações são passos que serão executados pelo testador. Vejam abaixo possíveis exemplos – bastante simplificados – de casos de testes para o nosso carro voador hipotético.

ENTRADA/CONDIÇÃO	RESULTADO ESPERADO
- Pressionar o botão de ligar	- Motor é acionado e o carro é ligado
- Desativar o freio de mão	- Desativar o sistema de frenagem
- Pressionar o pedal do acelerador	- Carro deverá começar sua aceleração vertical

É claro que isso é só um exemplo bobo, mas é legal para que vocês entendam que o testador realiza um conjunto de ações que devem gerar um resultado. **Caso o resultado não esteja de acordo com o esperado, significa que o testador encontrou uma falha – que é o principal objetivo de um teste** – por exemplo: o testador pressionou o botão de ligar do carro voador e o motor simplesmente não foi acionado.

No mundo do software, é bastante semelhante! Imagine um sistema de cadastro que exige que o usuário insira seu CPF. Um caso de teste óbvio é verificar se o campo de CPF obedece ao formato **###.###.###-##**. *Por que?* Porque esse é o formato padrão de um CPF! Logo, um caso de teste interessante seria tentar inserir o número **1234.56.7-7890** – se você conseguir, significa que o sistema está falhando, porque está aceitando um CPF em formato diferente do esperado.

ENTRADA/CONDIÇÃO	RESULTADO ESPERADO
- Validar a máscara do campo CPF	- Exibir máscara no formato ###.###.###-##
- Preencher CPF e clicar em salvar	- Registro salvo na tabela de cadastro
- Clicar no botão Voltar	- Sistema deve retornar à página inicial

(INFRAERO – 2011) A especificação de um caso de teste (*test case specification*) deve conter, entre outros, identificador, itens constantes no teste, especificação de entrada, especificação de saída, definição do ambiente necessário, necessidades especiais e dependências de outros casos de testes.

_____

Comentários: são exatamente esses itens que devem constar em um caso de teste (Correto).

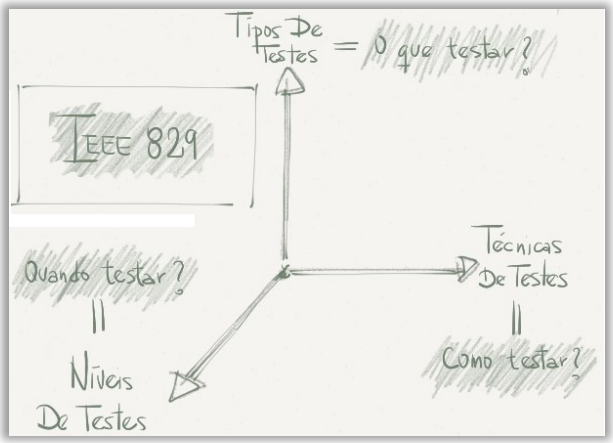
(EBSERH – 2013) Os testes de software são executados, usando os procedimentos e documentos de script de teste. Para que a fase de execução de teste, seja realizada com sucesso deve(m) serexecutado(s):

a) os casos de uso
b) os diagramas de atividade
c) os casos de teste
d) os testes de Turing
e) o teste de COMA.

_____

Comentários: na fase de especificação os casos de teste são elaborados e revisados, mas eles são efetivamente executados na fase de execução (Letra C).

Galera, nós podemos olhar para os testes por meio de três dimensões ou perspectivas diferentes: **Técnicas de Testes, Níveis de Testes e Tipos de Testes**. A imagem à esquerda as apresenta como um plano cartesiano de três dimensões – nós vamos ver nos próximos tópicos cada uma delas em detalhes. Na imagem à direita, podemos ver como essas dimensões de subdividem e qual é a função de cada uma delas: *como, quando e o que testar?*



Estratégia/Níveis de Testes

INCIDÊNCIA EM PROVA: MÉDIA

O software é testado para revelar erros cometidos inadvertidamente quando projetado e construído. *Mas como devemos conduzir os testes? Devemos estabelecer um plano formal para nossos testes? Devemos testar o programa como um todo ou executar testes somente em uma parte dele? Devemos refazer os testes quando acrescentamos novos componentes ao sistema? Quando devemos envolver o cliente?*

Essas e muitas outras questões são respondidas quando desenvolvemos uma estratégia de teste de software. *Por que ela é importante?* **O teste muitas vezes requer mais trabalho de projeto do que qualquer outra ação da engenharia de software.** Se for feito casualmente, perde-se tempo, fazem-se esforços desnecessários, e, ainda pior, erros passam sem ser detectados, portanto é razoável estabelecer uma estratégia sistemática e formal para teste de software.

De maneira genérica, podemos dizer que o teste começa pelo "pequeno" e passa para o "grande". Em outras palavras, os testes iniciais focalizam um único componente ou um pequeno grupo de componentes relacionados e aplicam-se testes para descobrir erros nos dados e na lógica de processamento que foram encapsulados pelo(s) componente(s). **Depois de testados, os componentes devem ser integrados até que o sistema completo esteja pronto.**

Nesse ponto, são executados muitos testes de ordem superior para descobrir erros ao atender aos requisitos do cliente. À medida que os erros forem descobertos, devem ser diagnosticados e corrigidos usando um processo chamado de depuração. Imaginem o motor de um carro: os testes começam nos componentes menores (Ex: vela de ignição, biela, virabrequim, pistão, válvula, etc) até terminar testando o funcionamento do carro como um todo.

Bem, muitas estratégias de teste de software já foram propostas na literatura. **Todas elas fornecem um modelo para o teste e todas têm as seguintes características genéricas:**

- Para executar um teste eficaz, proceder a revisões técnicas eficazes. Fazendo isso, muitos erros serão eliminados antes do começo do teste.

- O teste começa no nível de componente e progride em direção à integração do sistema computacional como um todo.

- Diferentes técnicas de teste são apropriadas para diferentes abordagens de engenharia de software e em diferentes pontos no tempo.

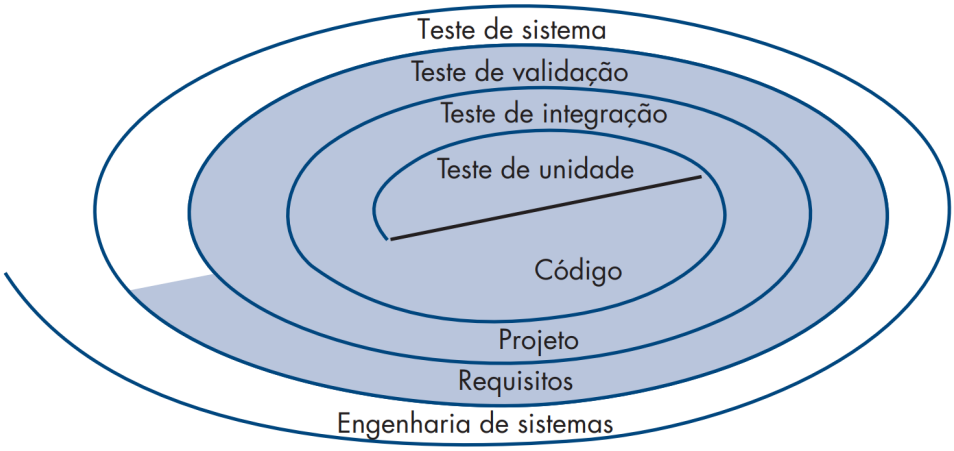
- O teste é feito pelo desenvolvedor do software e (para grandes projetos) por um grupo independente de teste.

- O teste e a depuração são atividades diferentes, mas a depuração deve ser associada com alguma estratégia de teste.

Nós sabemos que uma estratégia de teste de software deve acomodar testes de baixo nível, necessários para verificar se um pequeno segmento de código fonte foi implementado corretamente, bem como testes de alto nível, que validam as funções principais do sistema de acordo com os requisitos do cliente – como no exemplo do motor. **No entanto, existem abordagens diferentes no mundo do teste de software para executar uma estratégia.**

Em um dos extremos, pode-se esperar até que o sistema esteja totalmente construído e, então, executar os testes apenas no sistema completo esperando encontrar os erros. **Essa abordagem, embora atraente, simplesmente não funciona – resultará em um software defeituoso que desagrada todos os que investiram nele**. No outro extremo, você pode executar testes diariamente, sempre que uma parte do sistema for construída.

Essa abordagem, embora menos atraente, pode ser muito eficaz! Uma estratégia que é preferida pela maioria das equipes de software está entre os dois extremos. **Ela assume uma visão incremental do teste, conforme é apresentado na espiral**. Ela pode ser compreendida sob dois pontos de vista diferentes: pela visão do Processo de Software, deve ser lida no sentido anti-horário; pela visão de Testes de Software, deve ser lida no sentido horário. *Como assim, Diego?*



Bem, se nós quisermos analisar sob o ponto de vista do Processo de Software, inicialmente realizam-se atividades de **Engenharia de Sistemas**, passamos à análise de **Requisitos**, em seguida desenvolve-se um **Projeto** e – finalmente – chegamos à **Codificação**. Logo, para desenvolver softwares, percorre-se a espiral no sentido anti-horário ao longo de linhas que indicam a diminuição do nível de abstração em cada volta, isto é, iniciamos em um nível alto de abstração (entendendo o sistema e o domínio de informação) e terminamos com o código em si.

(UFG – 2017) Considere os diferentes níveis de teste de funcionalidade de um software. Os testes de sistema estão para a Engenharia de Sistemas, assim como:

a) os testes de integração e de validação estão, respectivamente, para o código e o projeto.

b) os testes de unidades e de integração estão, respectivamente, para os requisitos e o projeto.

c) os testes de unidades e de validação estão, respectivamente, para o código e o projeto.

d) os testes de validação e de integração estão, respectivamente, para os requisitos e o projeto.

_____

Comentários: bastava lembrar da nossa espiral. Logo, os testes de sistema estão para engenharia de sistemas, assim como os testes de validação e de integração estão, respectivamente, para requisitos e projeto (Letra D).

Por outro lado, se nós quisermos analisar sob o ponto de vista de Estratégia de Testes, inicialmente realizam-se **Testes de Unidade**, que se concentram em cada unidade do software (Ex: componente, classe, entre outros) 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.

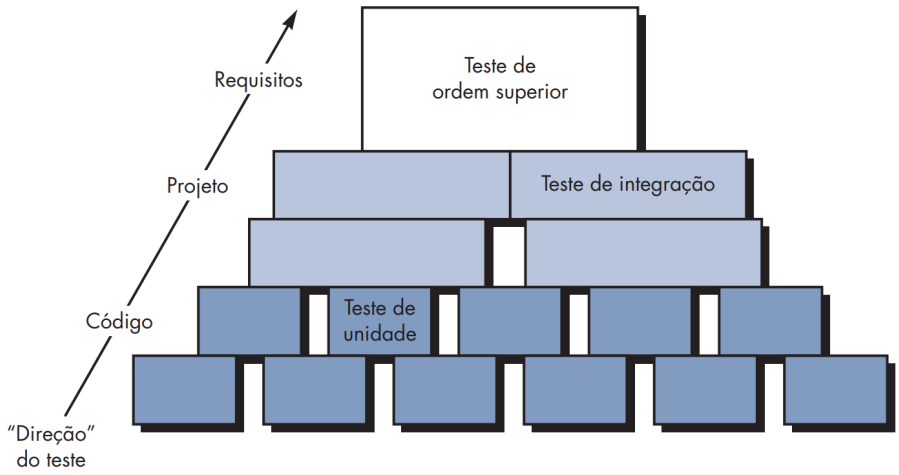
Na mesma direção, encontramos o **Teste de Validação**, em que requisitos estabelecidos como parte dos requisitos de modelagem são validados em relação ao software criado. Finalmente, chegamos ao **Teste do Sistema**, no qual o software e outros elementos são testados como um todo. Para testar um software 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.

Considerando o processo de um ponto de vista procedimental, **o teste dentro do contexto de engenharia de software é na realidade uma série de quatro etapas que são implementadas sequencialmente – as etapas ilustradas na figura da página anterior.** Inicialmente, os testes focalizam cada componente individualmente, garantindo que ele funcione adequadamente como uma unidade, daí o nome teste de unidade.

O teste de unidade usa intensamente técnicas de teste com caminhos específicos na estrutura de controle de um componente para garantir a cobertura completa e a máxima detecção de erro. Em seguida, o componente deve ser montado ou integrado para formar o pacote completo de software. **O teste de integração cuida de problemas associados com aspectos de verificação e construção de programa.**

Técnicas de projeto de casos de teste que focalizam em entradas e saídas são mais predominantes durante a integração, embora técnicas que usam caminhos específicos de programa possam ser utilizadas para segurança dos principais caminhos de controle. **Depois que o software foi integrado (construído e montado), é executada uma série de testes de ordem superior.** Os critérios de validação – estabelecidos durante a análise de requisitos – devem ser avaliados.

O teste de validação proporciona a garantia final de que o software satisfaz a todos os requisitos informativos, funcionais, comportamentais e de desempenho. A última etapa de teste de ordem superior extrapola os limites da engenharia de software, entrando em um contexto mais amplo de engenharia de sistemas de computadores. O software, uma vez validado, deve ser combinado com outros elementos do sistema (por exemplo: hardware, pessoas, base de dados).



O teste de sistema basicamente verifica se todos os elementos se combinam corretamente e se a função/desempenho global do sistema é conseguida. Há uma classificação que divide os testes em Testes de Baixo Nível (1º Nível) e Testes de Alto Nível (2º Nível). **No primeiro caso, o profissional deve ter um profundo conhecimento da estrutura interna do software – nesse caso, específico é comum que os testes sejam realizados pelo próprio desenvolvedor.** *Por que?*

Porque ele possui toda a carga de conhecimento que é necessária para realizar essas atividades. O primeiro nível é composto pelos Testes Unitários e Testes de Integração. **Já no segundo nível, não é necessário conhecimento da estrutura interna do software.** Os testes são guiados pelas especificações de negócio e pela lista de requisitos do software. O segundo nível é composto pelos Testes de Validação e Testes de Sistema.

(TCE/AM – 2011) Uma estratégia de teste que é escolhida por grande parte das equipes de software adota uma visão incremental do teste, começando com o teste de unidades individuais de programa, avançando para testes projetados a fim de facilitar a integração das unidades e culmina com testes que exercitam o sistema construído.

_____

Comentários: a questão mostra exatamente o que vimos de uma abordagem incremental começando com testes de unidade e terminando com testes de sistema (Correto).

(UFC – 2016) Qual das opções abaixo contém apenas níveis de teste de software?

a) Teste de unidade, Teste de estresse, Teste de aceitação.
b) Teste de integração, Teste de estresse, Teste de sistema.

c) Teste de aceitação, Teste Chinês, Teste de caixa branca.
d) Teste de unidade, Teste de integração, Teste de aceitação.
e) Teste de sistema, Teste de caixa branca, Teste de caixa preta.

_____

Comentários: (a) Errado, Teste de Estresse não é nível de teste; (b) Errado, Teste de Estresse não é nível de teste; (c) Errado, Teste Chinês e Caixa Branca não são níveis de teste; (d) Correto, todos são níveis de teste; (e) Errado, Teste Caixa Branca e Preta não são níveis de teste (Letra D).

Teste de Unidade

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Também chamado de Teste de Componente/Módulo, focaliza o esforço de verificação na menor unidade de projeto do software: o componente ou módulo. A ideia aqui é testar caminhos importantes para descobrir erros dentro dos limites do módulo. Além disso, ele enfoca a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente, sendo possível ser conduzido em paralelo para diversos componentes.

A interface de um módulo é testada para assegurar que as informações fluam corretamente para dentro ou para fora da unidade de programa que está sendo testada. **Caminhos independentes da estrutura de controle são usados para assegurar que todas as instruções em um módulo tenham sido executadas pelo menos uma vez**. As condições-limite são testadas para garantir que o módulo opere adequadamente nas fronteiras estabelecidas para limitar o processamento.

Finalmente, são testados todos os caminhos de manipulação de erro. *Como assim, Diego?* Este é um processo de teste de defeitos e, portanto, sua meta é expor defeitos nesses componentes. *Professor, o que você quer dizer com componente ou módulo?* No código-fonte, seria uma função ou método individual de um objeto; classes de objeto com vários atributos e métodos; componentes compostos que constituem diferentes objetos ou funções.

Esses componentes compostos têm uma interface definida usada para acessar sua funcionalidade. **As funções ou métodos individuais são os tipos mais simples de componentes e seus testes são um conjunto de chamadas dessas rotinas com diferentes parâmetros de entrada**. Eles verificam o funcionamento de um pedaço do software isoladamente ou que possam ser testados separadamente.

O Teste de Unidade pode ser realizado antes ou depois que o código-fonte tenha sido escrito. **Geralmente são feitos pelos próprios desenvolvedores de maneira mais informal e, não, por especialistas em testes**. Por meio de Testes de Unidade, é possível encontrar problemas mais cedo; facilita mudanças; simplifica integrações; auxilia a documentação; melhora o projeto do software; entre outros. Vamos para as nossas tradicionais metáforas agora...

Vocês se lembram do nosso exemplo? Os componentes de um motor de carro seriam basicamente suas peças – elas são a menor unidade de um motor e, portanto, precisam ser testadas individualmente! **Além disso, é necessário testar as interfaces dessas peças, uma vez que elas serão integradas com outras peças futuramente como a biela e os pistões**. Com software, é bastante similar, mas são testadas unidades de código-fonte.


VIRABREQUIM


BIELA


PISTÃO

DEFINIÇÕES DE PROVA - TESTES DE UNIDADE
Testes de Unidade são aqueles realizados sobre as menores estruturas de código-fonte, como métodos e classes.
Testes de Unidade consistem em testar individualmente, componentes ou módulos de *software* que, posteriormente devem ser testados de maneira integrada.
Testes de Unidade focalizam cada componente de um software de forma individual, garantindo que o componente funciona adequadamente.
Testes de Unidade focalizam o esforço de verificação na menor unidade de projeto de software, isto é, no componente ou no módulo de software.
Testes de Unidade têm por objetivo explorar a menor unidade do projeto, procurando identificar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo separadamente.
Testes de Unidade enfocam a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente.
Testes de Unidade concentram o esforço de verificação na menor unidade de design de software.
Testes de Unidade concentram-se na lógica de processamento interno e nas estruturas de dados dentro dos limites de um componente.
Testes de Unidade têm por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente.
Testes de Unidade têm como foco as menores unidades de um programa, que podem ser funções, procedimentos, métodos ou classes.

Existem duas estratégias que podem ser eficazes para ajudar você a escolher casos de teste. São elas:

1. **Teste de Partição**, em que você identifica os grupos de entradas que possuem características comuns e devem ser tratados da mesma maneira. Você deve escolher os testes dentro de cada um desses grupos.

2. **Testes baseados em Diretrizes**, em que você usa as diretrizes de testes para escolher casos de teste. Essas diretrizes refletem a experiência anterior dos tipos de erros que os programadores cometem frequentemente no desenvolvimento de componentes.

(BRB – 2011) Os testes de unidade, normalmente feitos pelos próprios desenvolvedores, sem necessidade de processos muito formais, são tratados dentro do próprio fluxo de implementação por meio de métodos simplificados.

_____

Comentários: realmente são feitos pelos próprios desenvolvedores e de forma mais simplificada (Correto).

(TRT20 – 2010) No contexto da estratégia para o teste de um projeto, os estágios de teste desempenham um papel importante. O teste que é aplicado a componentes do modelo de implementação para verificar se os fluxos de controle e de dados estão cobertos e funcionam conforme o esperado, é o teste:

a) do desenvolvedor.
b) independente.
c) de integração.
d) de sistema.
e) unitário.

_____

Comentários: o teste de unidade – de fato – é aplicado a componentes e busca verificar se os fluxos de controle (ordem das instruções de um algoritmo) e de dados estão cobertos e conforme o esperado (Letra E).

Teste de Integração

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Um novato no mundo do software pode levantar uma questão aparentemente legítima quando todos os módulos tiverem passado pelo teste de unidade: *ora, se todos os módulos já foram testados e estão funcionando perfeitamente de forma individual, porque eles não funcionariam em conjunto?* Cara, o problema é justamente colocá-los todos juntos porque pode haver diversos problemas de interfaces.

No contexto de softwares, dados podem ser perdidos através de uma interface; um componente pode ter um efeito inesperado ou adverso sobre outro; subfunções, quando combinadas, podem não produzir a função principal desejada; imprecisão aceitável individualmente pode ser amplificada em níveis não aceitáveis; estruturas de dados globais podem apresentar problemas. Infelizmente, essa lista não tem fim...

O Teste de Integração é uma técnica sistemática para construir a arquitetura de software ao mesmo tempo que conduz testes para descobrir erros associados com as interfaces. O objetivo é construir uma estrutura de programa determinada pelo projeto a partir de componentes testados em nível de unidade. **Muitas vezes, há uma tendência de tentar integração não incremental** – também chamado de abordagem Big Bang.

Nesse caso, todos os componentes são combinados com antecedência. **O programa inteiro é testado como um todo**. *Essa é uma boa abordagem?* Não, usualmente o resultado é o caos, porque muitos erros podem ser encontrados de uma só vez e a correção se torna complexa, visto que há muito espaço para procurar e isolar a causa do erro. Uma vez corrigidos esses erros, novos erros aparecem e o processo parece não ter fim.

Temos também a integração incremental, que é o oposto da abordagem Big Bang. O programa é construído e testado em pequenos incrementos ainda no ambiente de desenvolvimento, em que os erros são mais fáceis de isolar e corrigir; as interfaces têm maior probabilidade de serem testadas completamente; e uma abordagem sistemática de teste pode ser aplicada. *Professor, você está falando com uma linguagem muito complexa – você pode simplificar?* Claro...

Vamos pensar nas peças do nosso motor! Nós precisamos integrar o virabrequim, a biela e o pistão porque eles – juntos – ajudam a transformar o deslocamento em movimento de rotação. Em outras palavras, esse é o conjunto responsável por transmitir a energia gerada pelo propulsor para a transmissão, que a distribui para as rodas do veículo. **As interfaces de cada módulo foram testadas na etapa anterior, agora é necessário testar se elas se comunicam corretamente.** Vejam como as três peças mencionadas nos testes de unidade se integram.

DEFINIÇÕES DE PROVA - TESTES DE INTEGRAÇÃO
Testes de Integração são caracterizados por testar as interfaces entre os componentes ou interações de diferentes partes de um sistema.
Testes de Integração visam testar as falhas decorrentes da integração dos módulos do sistema.
Testes de Integração são uma técnica sistemática para construir a arquitetura do software, enquanto, ao mesmo tempo, conduz testes para descobrir erros associados às interfaces.
Testes de Integração têm por objetivo construir uma estrutura de programa determinada pelo projeto a partir de componentes já testados.
Testes de Integração são uma técnica utilizada para descobrir erros associados às interfaces na qual, a partir de componentes testados individualmente, se constrói uma estrutura de programa determinada pelo projeto.
Testes de Integração verificam 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.
Testes de Integração verificam se os componentes do sistema, juntos, trabalham conforme descrito nas especificações do sistema e do projeto do programa.
Testes de Integração são uma técnica sistemática para construir a arquitetura do *software* enquanto conduz testes para descobrir erros associados às interfaces.

(INPI – 2013) No teste de integração, verificam-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.

_____

Comentários: ele realmente verifica o funcionamento do **conjunto** de componentes por meio de suas interfaces (Correto).

(CET – 2011) Em um teste de integração, é possível detectar possíveis falhas provenientes da integração interna dos componentes de um sistema. O teste de integração sucede o teste de unidade, no qual os módulos são testados individualmente, e antecede o teste de sistema, em que o sistema completo é testado.

_____

Comentários: ele realmente detecta falhas de provenientes da integração dos componentes. Além disso, é verdade que o teste de integridade sucede o teste de unidade e antecede o teste de sistema (Correto).

(TCU – 2015) O teste de integração consiste em construir gradualmente o sistema, por integração de seus componentes, e testar o sistema resultante, buscando identificar e analisar problemas originados a partir das interações entre esses componentes, em um ambiente de execução com características próximas àquelas a serem utilizadas no ambiente operacional real.

_____

Comentários: testes de integração ocorrem ainda no ambiente de desenvolvimento e, não, execução (Errado).

Teste de Validação

INCIDÊNCIA EM PROVA: MÉDIA

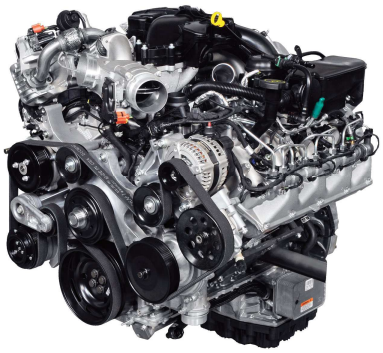
Também chamado de Teste de Aceitação, ele começa quando termina o teste de integração, quando os componentes individuais já foram exercitados, o software está completamente montado como um pacote e os erros de interface já foram descobertos e corrigidos. **Esse teste focaliza simplesmente em ações visíveis ao usuário e também em saídas do sistema reconhecíveis pelo usuário**. Se o usuário não vê ou reconhece, não é testado aqui!

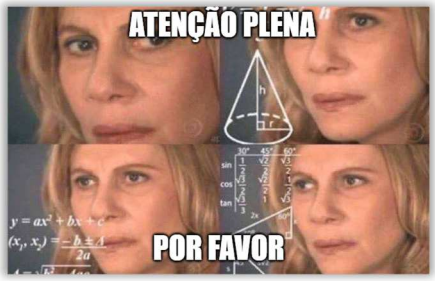
A validação pode ser definida de várias maneiras, mas uma definição simples (embora rigorosa) afirma que **a validação tem sucesso quando o software funciona de uma maneira que pode ser razoavelmente esperada pelo cliente**. Nesse ponto, um desenvolvedor de software veterano pode dizer: *quem ou o que é o árbitro para decidir o que são expectativas razoáveis? Que critério é esse?* Difícil decidir isso...

Bem, se durante o processo de desenvolvimento foi desenvolvido um documento de requisitos de software, ela provavelmente descreverá todos os atributos do software visíveis ao usuário e conterá uma seção denominada Critérios de Validação, que ajuda a avaliar se os requisitos estão de acordo com as expectativas do usuário. **Para tal, existem os Testes Alfa e Testes Beta, que serão vistos posteriormente com mais detalhes**.

Para dar uma pequena noção sobre esses dois testes: o primeiro é conduzido por uma equipe de testadores no local em que ocorre o desenvolvimento; já o segundo é conduzido pelos próprios usuários no local em que ele realmente será utilizado. **De todo modo, a ideia por trás dos testes de validação/aceitação é funcionar como um teste formal sobre as necessidades dos usuários para determinar se o software satisfaz ou não os critérios de validação predefinidos.**

Vocês se lembram do nosso exemplo do motor? Pois é, vamos supor que aquele motor tenha sido encomendado para BMW para ser instalado em um carro da Citröen. Esse é o momento de chamar os funcionários da Citröen e testar o motor na presença deles, e também é o momento de entregar o motor para os próprios funcionários da Citröen testarem se está tudo certo. **O teste é também realizado em relação ao que estava especificado nos critérios de validação.**





Galera, nós sabemos que a verificação ocorre em relação à especificação de requisitos e a validação ocorre em relação às expectativas dos clientes. No entanto, a expectativa dos clientes é arbitrada em relação aos critérios de validação, que também estão presentes no documento de requisitos. **Logo, eu preciso da plena atenção de vocês agora porque as provas possuem algumas nuances em relação a esse assunto.**

Como os critérios de validação também pertencem ao documento de requisitos, **caso alguma questão afirme que a validação ocorre em relação aos requisitos não está errado!** *Diego, mas isso não faz o menor sentido!* Pois é, eu sei que isso é realmente confuso, mas nós temos que nos adaptar às bancas e, não, o contrário. Além disso, Roger Pressman também dá margem a esse entendimento em seu livro:

> *Como todas as outras etapas de teste, a validação tenta descobrir erros, mas o foco está no nível de requisitos — em coisas que ficarão imediatamente aparentes para o usuário final. A validação de software é conseguida por meio de uma série de testes que demonstram conformidade com os requisitos.*

Vamos adotar um entendimento padrão? O teste de verificação tem por finalidade encontrar defeitos e inconsistências com relação a sua especificação de requisitos. Já o teste de validação ocorre em relação às expectativas do usuário e às funcionalidades percebíveis por ele. **Por outro lado, se uma questão afirmar que o teste de validação pode ocorrer em relação aos requisitos especificados, sem estar comparando com testes de verificação, recomendo marcar correto.**

DEFINIÇÕES DE PROVA - TESTES DE VALIDAÇÃO/ACEITAÇÃO
Testes de Validação focalizam ações e saídas, tais como percebidas pelo usuário final.
Testes de Validação são executados logo após montagem do pacote de software, quando os erros de interface já foram descobertos e corrigidos.
Testes de Validação têm como principal característica verificar o sistema em relação aos seus requisitos originais e às necessidades atuais do usuário.
Testes de Validação avaliam o software com respeito aos seus requisitos e detecta falhas nos requisitos e na interface com o usuário.

(UFRJ – 2008) O teste de validação focaliza ações e saídas tais como percebidas pelo usuário final.

_____

Comentários: ele realmente só enfoca aquilo que é visto ou reconhecido pelo usuário (Correto).

(MPE/MA – 2013) O teste de aceitação normalmente é realizado utilizando-se a interface final do sistema. Sobre esse tipo de teste pode-se afirmar que tem como objetivo principal a validação do software quanto aos requisitos.

_____

Comentários: ele faz a validação quanto aos requisitos e seus critérios de validação descritos no documento de requisitos (Correto).

(ANAC – 2009) Requisitos descrevem um acordo ou contrato entre duas partes, especificando, entre outros aspectos, o que o sistema de software deve fazer para ser aprovado em um teste de aceitação.

_____

Comentários: a questão trata dos critérios de validação/aceitação (Correto).

(TRE/ES – 2011) O teste de validação tem por finalidade encontrar defeitos e inconsistências no programa com relação a sua especificação.

_____

Comentários: observem que o enunciado evidentemente trata de uma comparação, logo quem tem por finalidade encontrar defeitos e inconsistências no programa em relação a sua especificação é o teste de verificação (Errado).

Teste de Sistema

INCIDÊNCIA EM PROVA: MÉDIA

Galera, um software é apenas um elemento de um grande sistema de computador. Ao final do desenvolvimento, ele precisará ser incorporado aos outros elementos do sistema (por exemplo: hardware, pessoas, informações). Um problema clássico de teste de sistema é a "procura do culpado". **Isso ocorre quando um erro é descoberto e os desenvolvedores de diversos elementos do sistema começam a acusar um ao outro pelo problema.**

Em vez de adotar essa postura sem sentido, você deve se antecipar aos problemas potenciais de interface e realizar as seguintes atividades:

- Criar caminhos de manipulação de erro que testem todas as informações vindas de outros elementos do sistema;
- Executar uma série de testes que simulem dados incorretos ou outros erros potenciais na interface de software;
- Deve registrar os resultados dos testes para utilizar como evidência se ocorrer a caça ao culpado;
- Participar do planejamento e projeto de testes do sistema para assegurar que o software seja testado adequadamente.

O 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. Galera, vamos finalizar as estratégias de testes pensando novamente em nosso exemplo.

Poxa, legal... eu testei os componentes do motor; depois eu testei como esses componentes se integravam; em seguida testei se aquilo satisfazia as necessidades dos clientes e os critérios de validação; e agora eu tenho que testar se aquele motor funciona corretamente integrado aos outros sistemas que integram o carro: sistema hidráulico, sistema elétrico, sistema de transmissão, sistema de arrefecimento, sistema de lubrificação, sistema de freios, entre outros.

Com software, é bastante similar! Um software é apenas parte de um sistema maior que envia e recebe dados e deve se integrar da melhor maneira possível. **O teste de integração verifica interfaces entre componentes do mesmo software, já o teste de sistema verifica interfaces**

entre componentes diferentes de um mesmo sistema – incluindo softwares, hardwares, pessoas e informações, além dos requisitos funcionais e não-funcionais. *Capiche?*

DEFINIÇÕES DE PROVA - TESTES DE SISTEMA
Testes de Sistema incluem diversas modalidades de teste, cujo objetivo é testar o sistema computacional como um todo.
Testes de Sistema testam se o sistema cumpre seus requisitos funcionais e não funcionais.
Testes de Sistema avaliam o software com respeito ao seu projeto arquitetural e detecta falhas de especificação, desempenho, robustez e segurança.
Testes de Sistema visam a verificar o sistema, baseado em computador, não se limitando ao software, mas incluindo o processo como um todo, como hardware, pessoal e informação.

(BANRISUL – 2018) _____ é uma verificação de consistência entre o sistema de software e sua especificação e, portanto, é uma atividade de verificação feita depois que se tem o sistema completo, com todas suas partes integradas para verificar se as funcionalidades especificadas nos documentos de requisitos estão todas corretamente implementadas. Este tipo de teste é focado principalmente na descoberta de falhas e executado pelo grupo de desenvolvimento de testes, tendo também um papel importante para avaliar se o produto pode ser liberado para os consumidores, o que é diferente do seu papel de expor falhas que são removidas para melhorar o produto. Assinale a alternativa que preenche corretamente a lacuna do texto acima.

a) Teste de sistema
b) Teste de unidade
c) Inspeção
d) Teste de regressão
e) Teste de integração

_____

Comentários: a questão trata dos testes de sistema. Eles realmente verificam a consistência entre sistema e especificação, sendo uma atividade de verificação e, não, validação. Item impecável (Letra A).

(FACEPE – 2015) Assinale a alternativa que corresponde ao estágio de processo de teste de software, no qual os componentes são integrados para compor o sistema, com vistas, entre outros, à busca de erros que resultam das interações não previstas entre os componentes, problemas de interface de componentes, validação de que o sistema atende aos requisitos funcionais e não funcionais.

a) Teste de sistema
b) Teste de componente
c) Teste de aceitação
d) Teste de especificação
e) Teste de operação

_____

Comentários: a questão trata dos testes de sistema – eles buscam erros resultantes das interações e integrações do sistema como um todo e também dos problemas de interface. A questão foi tão genérica que, se houvesse a opção Teste de Integração, teríamos duas respostas (Letra A).

Técnicas de Testes

As Técnicas se dividem em: Testes Caixa-Branca, Testes Caixa-Preta e Testes Caixa-Cinza. A primeira se foca nas estruturas internas dos procedimentos do sistema. A segunda se foca nas entradas e saídas especificadas nos requisitos funcionais. Por fim, a terceira se foca tanto em estruturas internas quanto nas entradas e saídas especificadas nos requisitos. Vejamos em detalhes cada uma delas. Venham comigo...

Teste Caixa-Branca

INCIDÊNCIA EM PROVA: ALTÍSSIMA

A Técnica Caixa-Branca (também conhecida como Estrutural, Procedimental, Orientada à Lógica, Caixa-de-Vidro ou Caixa-Clara) analisa caminhos lógicos possíveis de serem executados, portanto é necessário ter conhecimento sobre o funcionamento interno dos componentes. Ela busca garantir que todos os caminhos independentes de um módulo sejam executados pelo menos uma vez.

Além disso, ele trata de todas as decisões lógicas para valores verdadeiros e falsos, além de executar laços dentro dos valores limites e avaliar as estruturas de dados internas do software – esse bando de termo técnica apenas mostra algumas formas de se avaliar a estrutura interna dos componentes. As técnicas principais são: Testes de Caminho Básico e Testes de Estruturas de Controle – não vamos entrar em detalhes sobre cada uma delas.

- **Caminho Básico:** permite derivar uma medida de complexidade lógica de um procedimento e a utiliza para definir um conjunto de caminhos de execução[1].

- **Estruturas de Controle:** permite validar estruturas de controle como estruturas de condição, estruturas de fluxo de dados ou estruturas de laços.

Galera, vocês se lembram que a estratégia de teste nos diz quando testar e a técnica de testes nos diz como testar? Se não, voltem algumas páginas! **Pois bem... a ideia da técnica de caixa-branca é testar a estrutura interna do software, isto é, seu código-fonte em si.** Ele é conhecido como caixa de vidro por uma razão: em uma caixa de vidro, eu consigo ver o que tem dentro; em uma caixa-preta, eu já não consigo fazer isso.

Então, a ideia aqui é: eu consigo ver as partes internas dos componentes, logo eu posso testá-las. Outro nome para essa técnica é teste estrutural! *Por que?* Porque eu consigo analisar a estrutura interna do código. Outro nome é teste procedimental! *Por que?* Porque eu consigo analisar como

[1] Geralmente é utilizado a Complexidade Ciclomática, que permite medir quantitativamente a complexidade lógica de um código por meio do número de caminhos independentes.

os procedimentos internos do componente. Outro nome é teste orientado à lógica! *Por que?* Porque eu consigo analisar sua lógica interna.

Vamos voltar ao nosso famoso motor de carro? **Fazendo uma comparação, o teste caixa-branca seria um teste em que o testador tem acesso às estruturas internas do motor.** Se ele quiser, ele pode abrir o motor para descobrir se o problema é, por exemplo, no virabrequim, biela ou pistão. Ele pode testar cada componente porque ele tem acesso a esses componentes e sabe como eles deveriam funcionar. Nós vamos ver no próximo tópico que nem sempre é assim...

(DATAPREV – 2012) Nesse tipo de teste, o analista tem total acesso à estrutura interna da entidade sendo analisada. Permite, por exemplo, que o analista possa escolher partes específicas de um componente para serem testadas. O tipo de teste citado é conhecido como teste:

a) Funcional
b) Caixa-preta.
c) Caixa-branca.
d) De estresse.
e) De integração.

_____

Comentários: a questão trata do teste caixa-branca – aquele em que o analista tem total acesso à estrutura interna analisada.

(Prefeitura de Teresina – 2016) Um Analista de Sistemas da PRODATER realizou testes diretamente sobre o código fonte de um componente de software para avaliar o seu comportamento interno usando testes de condição, de fluxo de dados, de ciclos e de caminhos lógicos. O Analista aplicou técnica de teste de:

a) caixa preta.
b) caixa branca.
c) sistema.
d) integração.
e) aceitação.

_____

Comentários: trata-se do teste caixa-branca – ele possui acesso ao código fonte dos componentes e consegue avaliar o seu comportamento interno por meio de vários métodos diferentes (Letra B).

(UFBA – 2009) Os testes de software Caixa-Branca examinam o comportamento interno do componente de software.

_____

Comentários: a questão está perfeita – definição pura e simples de teste caixa-branca (Correto).

(TRE/PI – 2009) Também conhecido por teste estrutural ou orientado à lógica, é uma técnica de teste de software que trabalha diretamente sobre o código fonte do componente de software para avaliar aspectos, tais como, teste de condição, teste de fluxo de dados, teste de ciclos e teste de caminhos lógicos. Trata-se da técnica de teste:

a) da Caixa-branca.
b) da Caixa-cinza.
c) da Caixa-preta.
d) de Integração.
e) de Regressão.

_____

Comentários: trata-se do teste caixa-branca – aquele que trabalha diretamente sobre o código-fonte por meio de diversos métodos, como os mencionados no enunciado (Letra A).

(UFRJ – 2009) Um teste de software que está relacionado a um exame minucioso de sua estrutura interna e detalhes procedimentais e que trabalha diretamente sobre o código fonte do software é conhecido como teste:

a) de validação;
b) de caixa-branca;
c) funcional;
d) de caixa-preta;
e) orientado a dado.

_____

Comentários: a técnica que trata do exame minucioso da estrutura interna e detalhes procedimentais é a técnica de teste caixa-branca (Letra B).

Teste Caixa-Preta

INCIDÊNCIA EM PROVA: ALTÍSSIMA

A Técnica Caixa-Preta (também conhecida como Comportamental, Funcional, Orientada a Dado, Orientada à Entrada/Saída ou Caixa-Escura) se baseia em pré-condições e pós-condições, geralmente sendo utilizada nas etapas posteriores da disciplina de testes. Ela busca funções incorretas ou inexistentes, erros de comportamento ou desempenho, erros de inicialização e interface, entre outros.

Deriva casos de teste a partir da especificação de requisitos, ignorando detalhes de implementação e se focando nas saídas geradas em resposta a entradas escolhidas e condições especificadas. Essa técnica tem o objetivo de verificar a funcionalidade e aderência aos requisitos, em uma ótica externa ou do usuário, baseado apenas em suas interfaces, sem se basear em qualquer conhecimento do código ou lógica interna do componente de software.

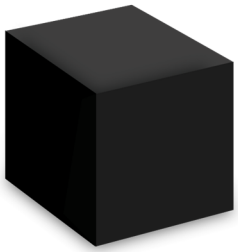
As técnicas principais são: Testes Baseados em Grafos, Particionamento de Equivalências, Análise de Valor Limite e Teste de Matriz Ortogonal.

- **Baseado em Grafos:** permite identificar os objetos e gera grafos para representá-los, testando-os e seus relacionamentos com o intuito de descobrir erros.

- **Partição de Equivalência:** permite agrupar os valores de entrada em categorias de dados para evitar redundância e aumentar a cobertura de testes do sistema.

- **Análise de Valor Limite:** permite exercitar os limites do domínio de entrada, tendo em vista que a maioria dos erros se encontram nas extremidades da entrada.

- **Matriz Ortogonal:** utilizado com entradas relativamente pequenas, casos de testes são espalhados uniformemente pelo domínio do teste para detectar falhas.



Galera, uma caixa-preta não permite que eu veja seus componentes internos. Logo, de forma diferente do teste caixa-branca, não é possível testar suas partes internas. *Ué, professor... então como eu vou testar?* Bem, nós temos um conjunto de pré-condições e pós-condições que também nos permitem encontrar erros e inconsistências. **Eu sei que, dadas condições específicas, eu devo alcançar resultados específicos.**

Outro nome para essa técnica é teste comportamental! *Por que?* Porque eu não consigo analisar a estrutura, mas eu consigo analisar seu comportamento esperado. Outro nome é teste Funcional! *Por que?* Porque eu consigo analisar como o software deveria funcionar. Outro nome é Orientado a Dado! *Por que?* **Porque eu não consigo analisar sua lógica, mas eu sei quais dados devem ser gerados como resultado** – além disso, dada uma entrada, temos uma saída específica.

Vamos voltar novamente ao nosso motor de carro? **Fazendo uma comparação, o teste caixa-preta seria um teste em que o testador não possui acesso às estruturas internas do motor.** Ele não pode abri-lo e verificar seu funcionamento interno. No entanto, ele sabe o comportamento que o motor deve ter quando se pressiona o acelerador, freio ou embreagem; quando se troca o tipo de combustível; quando se inicia a ignição; quando falta lubrificação; entre outros.

Para cada entrada ou pré-condição, há uma saída esperada. Dessa forma, é possível descobrir qual é o problema do motor. Galera, isso serve para tudo! Você não tem acesso às partes internas dos componentes que te fornecem internet em casa. **No entanto, você pode fazer testes caixa-preta! Se sua internet cair, você pode testar se o cabo de força está conectado na tomada; se o cabo de internet está conectado ao computador; entre outros.**

Como você sabe que, dadas entradas específicas, você deve ter saídas específicas, você pode realizar diversos testes para identificar problemas mesmo sem ter acesso a componentes internos. **Você não sabe como uma calculadora faz internamente para realizar um cálculo, mas você sabe que – ao calcular 3x7 – o resultado necessariamente terá que ser 21. Qualquer outro resultado implicaria em um defeito da calculadora!** *Bacana?* Fechou...

(TJ/AP – 2014) No Tribunal de Justiça do Estado do Amapá, um software está passando por um teste no qual são verificadas as suas funcionalidades sem preocupação com os detalhes de implementação. Nesse processo de teste estão sendo realizadas a identificação das funcionalidades que o software deve realizar e a criação dos casos de testes capazes de checar se essas funcionalidades estão sendo realizadas adequadamente. Trata-se do teste:

a) alfa.
b) de integração.
c) de caixa-branca.
d) de unidade.
e) de caixa-preta.

_____

Comentários: a questão afirma que se trata de um teste no qual são verificadas as suas funcionalidades sem preocupação com detalhes de implementação, logo se trata dos testes caixa-preta (Letra E).

(TRT/PR – 2010) A técnica de teste de software, também chamada de comportamental, é a técnica de:

a) caixa-preta.
b) caixa-branca.
c) ciclo.
d) condição.
e) fluxo de dados.

_____

Comentários: teste comportamental é um teste caixa-preta (Letra A).

(MPE/RO – 2012) Considerando a arquitetura de software convencional, a técnica de teste que avalia o comportamento externo do componente de software, sem considerar o seu comportamento interno, denomina-se:

a) Caixa-branca.
b) Caixa-cinza.
c) Caixa-preta.
d) Regressão.
e) Técnicas não funcionais

_____

Comentários: quem avalia o comportamento externo de um componente de software sem considerar seu comportamento interno é a técnica de teste caixa-preta (Letra C).

(CPDEMIG – 2013) Qual o tipo de teste de software contempla os requisitos comportamentais, e que permitem testar plenamente os requisitos funcionais de um programa?

a) Teste de ciclo.
b) Teste de condição.
c) Teste caixa-preta
d) Teste de fluxo.

_____

Comentários: as palavras-chave já dão uma dica: requisitos **comportamentais** que permitem testar plenamente os requisitos **funcionais** de um programa são o teste caixa-preta – lembrando que essa técnica também chamada de teste funcional ou comportamental (Letra C).

(COBRA – 2015) Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido. Haverá sucesso no teste, se o resultado obtido for igual ao resultado esperado. Interessa a função executada, e não a característica interna do componente. O componente de software a ser testado pode ser um método, uma função interna, um programa, um componente, um conjunto de programas e/ou componentes ou mesmo uma funcionalidade. Trata-se de um teste:

a) Não funcional.
b) Caixa Preta.
c) Caixa Branca.
d) Ponto de Função.
e) Não estrutural.

_____

Comentários: trata-se do teste caixa-preta ou orientado à entrada/saída (Letra B).

Teste Caixa-Cinza

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

O Teste Caixa-Cinza é uma versão híbrida entre os Testes Caixa-Branca e os Testes Caixa-Preta. Nesse sentido, essa técnica analisa a parte lógica mais a funcionalidade do sistema, fazendo uma comparação do que foi especificado com o que está sendo realizado. Usando esse método, o testador comunica-se com o desenvolvedor para entender melhor o sistema e otimizar os casos de teste que serão realizados.

Isso envolve ter acesso a estruturas de dados e algoritmos do componente a fim de desenvolver os casos de teste, que são executados como na técnica da caixa-preta. Manipular entradas de dados e formatar a saída não é considerado caixa-cinza, pois a entrada e a saída estão claramente fora da caixa-preta. Alguns autores definem o Teste de Integração como um Teste Caixa-Cinza – esse raramente cai em prova, galera...

(SERPRO – 2010) A engenharia de software é um a disciplina do conhecimento humano que aplica princípios da engenharia ao desenvolvimento de software. N o entanto, existem diferenças significativas entre as engenharias clássicas — mecânica, civil, elétrica— e a engenharia de software, muitas delas decorrentes das diferenças de natureza entre o produto ou sistema resultante da atividade das engenharias clássicas e a engenharia de software. A engenharia de software desdobrou-se em várias áreas especializadas, como as áreas de requisitos de software, de análise e projeto de software, de implementação de software, de testes de software, de gestão de configuração, entre outras. N a interface entre as engenharias clássicas e a de software insere-se a engenharia de sistemas, abordada colateralmente na ISO/IEC 12207.

Com relação ao emprego de diferentes técnicas para a realização de testes de software, é correto afirmar que haverá maior diminuição da dependência de acesso às especificações arquiteturais de um sistema se o testador empregar a técnica de caixa-branca (white-box), em vez das técnicas de caixa-cinza (gray-box) e de caixa-preta (black-box).

Comentários: o teste caixa-branca depende completamente do acesso às especificações arquiteturais de um sistema, testes caixa cinza menos e testes caixa-preta necessitam apenas de pré-condições e pós-condições. Dessa forma, a questão está incorreta, tendo em vista que haverá ~~maior~~ menor diminuição da dependência de acesso às especificações arquiteturais de um sistema se o testador empregar a técnica de caixa-branca, em vez das técnicas de caixa-cinza e de caixa-preta (Errado).

Tipos de Testes

Galera, agora veremos vários e vários tipos de testes. É importante que vocês saibam que cada autor pode criar seu tipo de teste e eles – por vezes – se contradizem. **A ideia aqui é apresentar aqueles testes que possuem maior destaque entre os autores mais consagrados de engenharia de software, mas deixando claro que não se trata de uma lista exaustiva.** Eventualmente, vocês podem encontrar um teste que não foi mencionado em aula. *Animados?*

Teste de Desempenho

INCIDÊNCIA EM PROVA: ALTA

Para alguns tipos de software, é inaceitável que um software execute suas funções, mas não esteja em conformidade com seus requisitos de desempenho. **O teste de desempenho (ou performance) é projetado para testar o desempenho em tempo de execução do software dentro do contexto de um sistema integrado.** O teste de desempenho é feito em todas as etapas no processo de teste e deve ser realizado, se possível, o quanto antes.

Até mesmo em nível de unidade, o desempenho de um módulo individual pode ser avaliado durante o teste. No entanto, o verdadeiro desempenho de um sistema só pode ser avaliado depois que todos os elementos do sistema estiverem totalmente integrados. **Os testes de desempenho muitas vezes são acoplados ao teste de esforço e usualmente requerem instrumentação de hardware e software.**

Em outras palavras, frequentemente é necessário medir a utilização dos recursos (por exemplo, ciclos de processador) de forma precisa. Instrumentação externa pode monitorar intervalos de execução, log de eventos (por exemplo, interrupções) à medida que ocorrem, e verificar os estados da máquina regularmente. **Monitorando o sistema com instrumentos, o testador pode descobrir situações que levam à degradação e possível falha do sistema.**

Eu sei, linguagem muito técnica! Vamos falar de maneira bem clara: teste de desempenho trata do esforço em assegurar que o sistema computacional pode operar sob a carga de operação especificada. Imaginem o site do Estratégia Concursos em dia de Black Friday! **Se eu não faço testes para avaliar qual a carga de acessos simultâneos ele é capaz de suportar, o site pode cair, perde clientes e perder dinheiro.** Logo, é um teste importantíssimo!

Esse teste pode ser usado também para identificar gargalos, determinar conformidades com os requisitos não-funcionais de desempenho e coletar outras informações, como hardware necessário para a operação da aplicação. Em geral, devem ser projetados para assegurar que o sistema pode operar na carga especificada. **Isso envolve o planejamento de uma série de testes em que a carga é constantemente aumentada até que o desempenho se torne inaceitável.**

(SERPRO – 2015) Um dos tipos de teste de software é o teste de desempenho. Um teste desse tipo:

a) corresponde a medir os tempos de processamento com falhas no hardware do sistema computacional.
b) não pode ser realizado com o sistema já em operação.
c) não se aplica a programas escritos com linguagens de programação orientadas a objetos.
d) visa apenas verificar se o software sob teste não contém erros lógicos.
e) visa assegurar se o sistema computacional pode operar sob a carga de operação especificada.

_____

Comentários: (a) Errado, não faz nenhum sentido; (b) Errado, claro que pode ser realizado com o sistema em operação; (c) Errado, pode ser aplicado em programas que utilizem quaisquer linguagens de programação; (d) Errado, esse seria um teste caixa-branca; (e) Correto, ele realmente visa assegurar que o sistema computacional pode operar sob a carga de operação especificada (Letra E).

(MEC – 2015) A realização de testes de desempenho deve ser realizada o quanto antes, ainda durante a etapa de arquitetura e implementação da solução. Para este fim, deve-se adotar estratégia que identifique as partes de uma funcionalidade que estejam demorando mais tempo para serem executadas, no nível de teste de componente ou no nível de integração de componente.

_____

Comentários: testes de desempenho podem ocorrer em todas as etapas do processo de testes e deve começar o quando antes. Os resultados desses testes permitem a realização de análises que podem ajudar a estimar a configuração do hardware necessária para suportar o software em questão (Correto).

(MEC – 2015) Um teste de desempenho deve ser realizado para mensurar se a infraestrutura tecnológica de produção é suficiente para suportar a quantidade de acessos simultâneos à aplicação web. Uma forma de realizar essa verificação é utilizar ferramenta com suporte à medição de teste de cobertura.

Comentários: ele realmente é realizado para mensurar se a infraestrutura tecnológica em produção (isto é, aquela que será utilizada para executar o software) é capaz de suportar a quantidade de acessos simultâneos. De fato, uma forma de realizar essa verificação é por meio de ferramentas de teste de cobertura, que conseguem analisar a abrangência do teste realizado. Se você testa de forma mais abrangente possível, você descobre quanto o software pode suporte (Correto).

(TCE/PR – 2011) Segundo Sommerville, após um sistema ser completamente integrado, é possível testar propriedades como a de desempenho do sistema. Neste contexto, considere:

I. Testes de desempenho devem ser produzidos de forma a garantir que o sistema possa processar a sua carga prevista, sendo que tais testes geralmente são planejados para que a carga seja continuamente aumentada até que o sistema apresente desempenho fora do aceitável.

II. Os testes de desempenho devem determinar se um sistema corresponde às suas exigências, sendo que a descoberta de defeitos ou problemas no sistema não é enfoque desta etapa.

III. Para determinar se o desempenho está sendo atingido, pode ser necessário a construção de um perfil operacional, que é a listagem de todo o grupo de operadores/usuários que farão uso deste sistema.

Está correto o que se afirma em:

a) I, apenas.
b) I, II, III.
c) III, apenas.
d) I e II, apenas.
e) II e III, apenas.

Comentários: (I) Correto. Testes de Desempenho devem – de fato - ser projetados para assegurar que o sistema pode operar na carga necessária, envolvendo, geralmente, o planejamento de uma série de testes em que a carga é constantemente aumentada até que o desempenho se torne inaceitável; (II) Errado. Um dos objetivos primários do teste de desempenho é demonstrar que o sistema atende aos requisitos e descobrir problemas e defeitos no sistema; (III) Errado. Na verdade, um perfil operacional é um conjunto de testes que refletem uma combinação real de trabalho a que o sistema será submetido (Letra A).

Galera, alguns autores consideram Testes de Stress e Testes de Carga como subgêneros dos Testes de Desempenho. Vejamos em detalhes...

Teste de Estresse

INCIDÊNCIA EM PROVA: ALTA

Também chamado de teste de esforço, serve para colocar os programas em situações anormais. Essencialmente, o testador que executa teste por esforço pergunta: *até onde podemos*

forçar o sistema até que ele falhe? Esse teste usa um sistema de maneira que demande recursos em quantidade, frequência ou volumes anormais. Podem ser realizados testes em condições de alta taxa de entrada de dados, máximo de memória ou processamento, entre outros.

Em suma, esse tipo de teste é uma forma de teste deliberadamente intenso e completo utilizado para determinar a estabilidade de um determinado sistema ou entidade. Envolve o teste para além da capacidade operacional normal, muitas vezes até um ponto de ruptura, a fim de observar os resultados. **Assim como o teste de desempenho – que está em uma hierarquia superior – o teste de estresse testa basicamente requisitos não-funcionais.**

(IF/SC – 2014) O teste de stress tem por objetivo avaliar como o sistema responde em condições anormais. Pode abranger cargas de trabalho extremas, memória insuficiente, hardware e serviços indisponíveis ou recursos compartilhados limitados.

_____

Comentários: esse tipo de teste realmente busca avaliar a resposta do sistema em condições anormais de utilização pelo usuário (Correto).

(MEC – 2015) Após um sistema ter sido completamente integrado, é possível testá-lo em relação ao desempenho e à confiabilidade. O teste de desempenho significa estressar o sistema por meio de demandas fora dos limites do projeto de software.

_____

Comentários: a questão trata do teste de estresse, que é um tipo de teste de desempenho (Correto).

(COPASA – 2014) Sobre os testes de estresse, assinale a alternativa incorreta.

a) Ocorrem quando módulos ou subsistemas são integrados para criar sistemas maiores.
b) Testa o comportamento de falha do sistema.
c) Causa estresse no sistema e pode acarretar a detecção de defeitos que, normalmente, não se manifestariam.
d) São relevantes para sistemas distribuídos, com base em uma rede de computadores.

_____

Comentários: (a) Errado, a questão trata de testes de integração; (b) Correto, testes de estresse realmente testam o comportamento de falha do sistema; (c) Correto, testes de estresse causam estresse no sistema e podem acarretar a detecção de defeitos que normalmente não se manifestariam; (d) Correto, isso permite detectar gargalos e anomalias (Letra A).

(MPE/SE – 2009) A execução de um sistema com o objetivo de encontrar falhas sob condições que demandam recursos em quantidade, frequência ou volume anormais é definida como:

a) payload.
b) teste de estresse.
c) teste de desempenho.

d) latência da falha.
e) workload.

_____
Comentários: a questão nos remete claramente a teste de estresse (Letra B).

(INFRAERO – 2011) Executa um sistema de forma a demandar recursos em volume ou frequência acima do normal. Trata-se de:

a) validação de unidade de implementação.
b) teste beta.
c) teste de estresse.
d) validação de integração.
e) teste de desempenho.

_____
Comentários: o teste que executa o sistema de forma a demandar recursos em volume ou frequência acima do normal é o teste de estresse (Letra C).

(BRDE – 2015) Qual tipo de teste é usado para testar especificamente o comportamento de um sistema sob demanda de recursos em quantidade, frequência ou volume anormais?

a) Teste de Recuperação.
b) Teste de Segurança.
c) Teste de Fluxo de Dados.
d) Teste de Integração.
e) Teste de Estresse.

_____
Comentários: o teste utilizado para testar especificamente o comportamento de um sistema sob demanda de recursos em quantidade, frequência ou volume anormais é o teste de estresse (Letra E).

(IF/SE – 2016) O teste de estresse deve ser feito com o objetivo de avaliar os efeitos da submissão do sistema a situações com alta demanda por recursos.

_____
Comentários: trata-se do teste de estresse (Correto).

Teste de Carga

INCIDÊNCIA EM PROVA: ALTA

Testes de Carga são a forma mais simples de testes para compreender o comportamento de um sistema sob uma carga específica. Ele é capaz de determinar o comportamento de um sistema sob condições especificadas ou acordadas. A carga pode ser, por exemplo, o número de usuários

simultâneos esperados na aplicação. Em suma: testes de desempenho podem se dividir em teste de stress e teste de carga.

Os testes de carga procuram determinar como o software responderá a várias condições de carga (Ex: número de usuários, número de transações por usuário por unidade de tempo, carga de dados processados por transação) de acordo com os limites de operação do sistema. Já o teste de estresse ultrapassa os limites operacionais do sistema para determinar com que capacidade o ambiente pode lidar. *Fechou?*

(TRT/ES – 2013) O teste de carga, que verifica o funcionamento do software, utiliza uma grande quantidade de usuários simultaneamente.

_____

Comentários: o teste de carga verifica o funcionamento do software utilizando a quantidade de usuários acordados ou especificados – esse valor não é necessariamente grande (Errado).

(ANTAQ – 2014) A realização de testes de carga é importante para os sistemas distribuídos, pois permite a identificação do ponto de degradação desses sistemas, o que possibilita a criação de controles de rejeição de operações a partir desse ponto.

_____

Comentários: quem possibilita identificar o ponto de degradação ou crash do sistema é teste de estresse (Errado).

(TCE/SC – 2016) Para se assegurar que o sistema opere com a carga necessária, são realizados testes de desempenho em que se aumenta progressivamente a carga até que se possa definir se o desempenho do sistema está aceitável.

_____

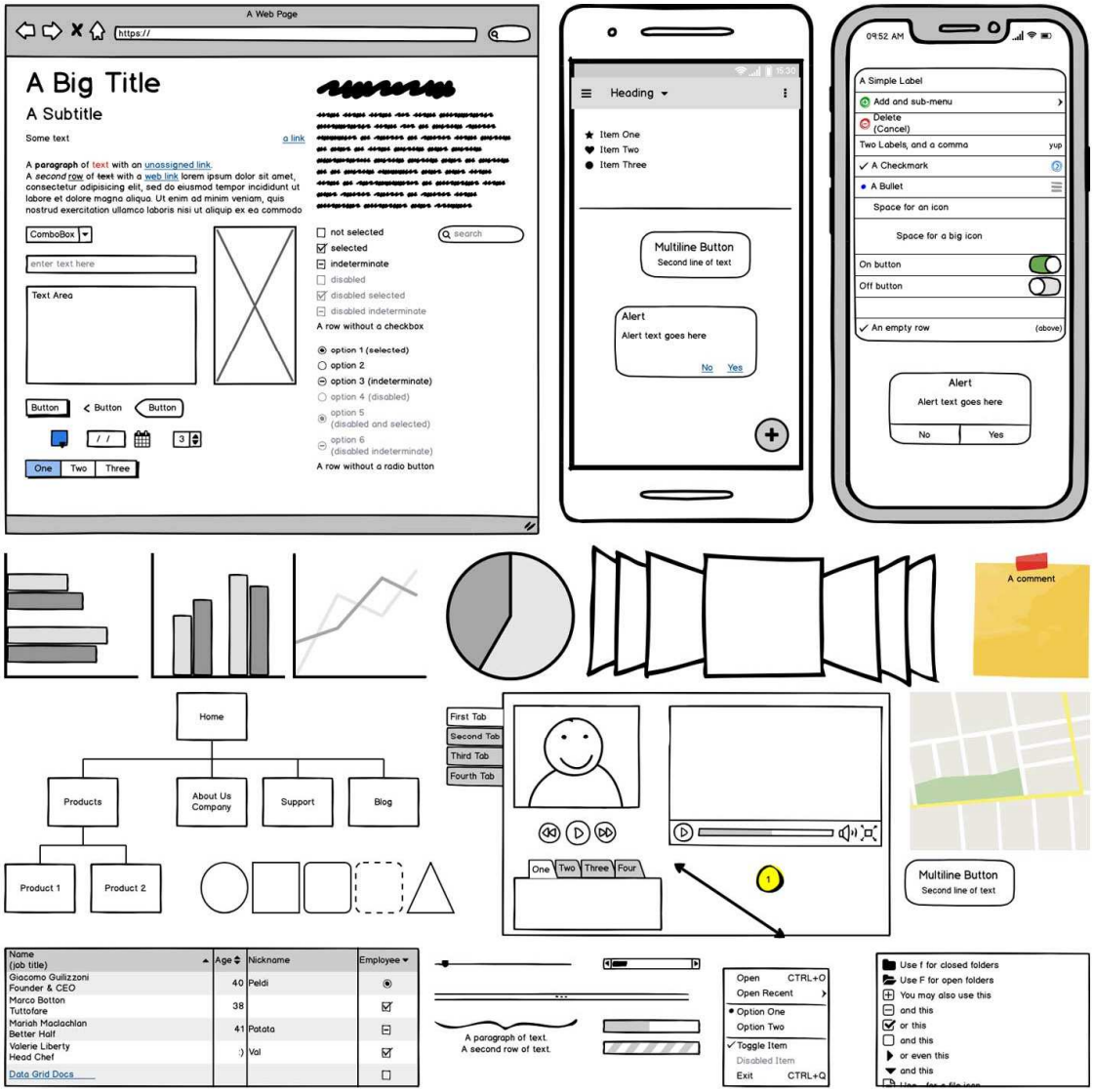
Comentários: essa é uma questão polêmica porque aumentar progressivamente a carga até que se possa definir se o desempenho do sistema está aceitável é a definição de teste de carga, no entanto o teste de carga é um tipo de teste de desempenho – logo, questão perfeita (Correto).

Teste de Usabilidade

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Testes de Usabilidade avaliam o grau com o qual os usuários podem interagir efetivamente com o software e o grau em que o software dirige as ações do usuário, proporciona uma boa realimentação e reforça uma abordagem de interação consistente. **Basicamente, os testes de usabilidade são projetados para determinar o grau com o qual a interface de um software facilita a vida do usuário.**

Dito de outra forma, esse teste trata do esforço em demonstrar falhas na facilidade de uso do software pelos usuários finais. Evidentemente, ele enfatiza fatores humanos, interface gráfica, ajuda online, estética/layout, wizards, documentação do usuário, material de treinamento, acesso às funcionalidades, entre outros. **Costuma-se dizer que uma boa interface com o usuário deve ser fácil de usar e de entender.**



EXEMPLO DE SOFTWARE QUE PERMITE A CONSTRUÇÃO DE PROTÓTIPOS DE TELA (BALSAMIQ)

Em suma, trata-se de um teste focado na experiência do usuário que avalia a facilidade de uso e aprendizado de um software e o desempenho da interação homem-computador com o intuito de obter indícios do nível de satisfação do usuário. Para tal, esse teste se utiliza de diversas técnicas, como a utilização de protótipos para abordar a interatividade, layout, clareza, estética, características da tela, sensibilidade ao tempo, personalização, acessibilidade, etc.

(MEC – 2011) Os testes de usabilidade avaliam a facilidade de uso do software testado e são bastante utilizados em aplicações web.

_____

Comentários: a questão está impecável – eles realmente avaliam a facilidade de uso do software e são muito aplicados em aplicações web (Correto).

(TCE/PA – 2016) É desnecessária a elaboração de protótipos para desenvolvimento de interface web.

_____

Comentários: é fundamental a elaboração de protótipos (Errado).

(MPE/PB – 2015) Considere os seguintes testes de software:

I. Avaliar o desempenho da interação homem-computador.
II. Obter indícios do nível de satisfação do usuário.
III. Avaliar a integridade dos dados registrados no sistema.

Pode ser considerado teste de usabilidade o que consta APENAS em:

a) I.
b) II.
c) III.
d) I e II.
e) II e III.

_____

Comentários: (I) Correto, ele de fato avalia o desempenho da interação homem-computador; (II) Correto, ele realmente é capaz de obter indícios do nível de satisfação do usuário; (III) Errado, ele não tem nenhuma intenção de avaliar a integridade dos dados registrados no sistema (Letra D).

Teste de Regressão

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Galera, pensem lá no teste de integração. *Como ele funciona?* Ele testa como os componentes se comportam quando estão integrados com outros componentes. **Cada vez que um novo módulo é acrescentado como parte do teste de integração, o software muda.** Novos caminhos de fluxo de dados são estabelecidos, podem ocorrer novas entradas e saídas, e uma nova lógica de controle pode ser invocada.

Essas alterações podem causar problemas com funções que antes funcionavam corretamente. **No contexto de uma estratégia de teste de integração, o teste de regressão é a reexecução do mesmo subconjunto de testes que já foram executados para assegurar que as alterações não tenham propagado efeitos colaterais indesejados.** Sim, pessoal... por vezes, você vai corrigir um problema, obtém êxito, mas acaba inserindo defeitos em outro local.

Em um contexto mais amplo, testes bem-sucedidos (de qualquer tipo) resultam na descoberta de erros, e os erros devem ser corrigidos. Sempre que o software é corrigido, algum aspecto da configuração do software (o programa, sua documentação, ou os dados que o suportam) é alterado. **O teste de regressão ajuda a garantir que as alterações (devido ao teste ou por outras razões) não introduzam comportamento indesejado ou erros adicionais.**

O teste de regressão pode ser executado manualmente, reexecutando um subconjunto de todos os casos de teste ou usando ferramentas automáticas de captura/reexecução. Ferramentas de captura/reexecução permitem que o engenheiro de software capture casos de teste e resultados para reexecução e comparação subsequente. À medida que o teste de integração progride, o número de testes de regressão pode crescer muito.

Dessa forma, o conjunto de testes de regressão deve ser projetado de forma a incluir somente aqueles testes que tratam de uma ou mais classes de erros em cada uma das funções principais do programa. **É impraticável e ineficiente reexecutar todos os testes para todas as funções do programa quando ocorre uma alteração.** *Entendido?* Galera, esse é talvez o tipo de teste mais recorrente em prova, então vamos resumir tudo que vimos...

O teste de regressão busca verificar a existência de defeitos após alterações em um sistema já testado. A ideia é descobrir erros por meio da reaplicação dos testes a um programa por conta de modificações ou novas funcionalidades para garantir que não foram introduzidos novos defeitos em componentes já testados anteriormente. **Eles são realizados geralmente durante a fase de manutenção, para mostrar que as modificações efetuadas estão corretas.** Como é na prática...

Imagine que eu acabo de desenvolver o novo site do Estratégia Concursos! Antes de disponibilizar para os alunos, eu faço vinte testes diferentes. Todos são exitosos, então eu finalmente lanço a versão 1.0 e disponibilizo para os alunos. **Só que aí o Prof. Ricardo Vale me pede para inserir uma nova funcionalidade que os alunos estão pedindo desesperadamente.** Eu implemento essa nova funcionalidade e lanço a versão 1.1!

Maaaaaaaaaaaas... antes de lançar essa nova versão, eu faço um teste de regressão. *Por que, Diego?* **Porque essa nova funcionalidade que eu implementei pode ter causado algum problema naquilo que já estava funcionando perfeitamente na versão anterior.** *Então, o que eu faço?* Eu reaplico os vinte testes diferentes da versão anterior para garantir que essa nova funcionalidade não introduziu novos defeitos que comprometam as funcionalidades da versão anterior.

(TJ/PI – 2015) Um sistema matemático, já em utilização, vem sofrendo diversas manutenções evolutivas. Após uma das novas funcionalidades ter sido implementada, a equipe responsável percebeu que algumas funcionalidades antigas começaram a apresentar falhas em seus resultados. Os membros da equipe, ainda inexperientes, definiram algumas medidas para verificar a possível causa do problema. A medida inicial mais adequada a ser adotada é:

a) novas entrevistas com o cliente;
b) reavaliação do escopo;
c) novos testes unitários;
d) testes de regressão;
e) revisão da modelagem.

Comentários: após uma das novas funcionalidades ter sido implementada, a equipe responsável percebeu que algumas funcionalidades antigas começaram a apresentar falhas em seus resultados. Logo, a medida inicial mais adequada a ser adotada é o teste de regressão (Letra D).

(FUNPRESP/JUD – 2016) O teste de regressão visa garantir a integridade de um software já testado que tenha recebido uma nova implementação.

Comentários: testes de regressão são desenvolvidos de forma incremental para verificar se as mudanças no programa não introduziram novos bugs (Correto).

(TRE/BA – 2010) Falha é o resultado de um ou mais defeitos em algum aspecto do sistema. No teste de regressão, caso um novo componente ou as suas alterações, quando acrescentados aos componentes restantes do sistema, resultem em novos defeitos em componentes inalterados, então considera-se que o sistema regrediu.

Comentários: perfeito, perfeito, perfeito! Galera, quando um sistema não passa nos testes de regressão, nós dizemos que o sistema regrediu. *Por que?* Porque antes ele funcionava corretamente e, após a inserção de novas funcionalidades, ele parou de funcionar. Esse item é coisa linda! (Correto).

Teste de Fumaça

INCIDÊNCIA EM PROVA: BAIXA

Teste fumaça é uma abordagem de teste de integração usada frequentemente quando produtos de software são desenvolvidos. É projetado como um mecanismo de marcapasso para projetos com prazo crítico, permitindo que a equipe de software avalie o projeto frequentemente. Uma série de testes é criada para expor erros que impedem a execução correta de uma função básica do sistema.

A finalidade deverá ser descobrir erros bloqueadores ou impeditivos (*showstopper*) que apresentam a mais alta probabilidade de atrasar o cronograma do software. O teste fumaça pode ser caracterizado como uma estratégia de integração rolante. O software é recriado (com novos componentes acrescentados) e o teste fumaça é realizado todos os dias. O teste fumaça deve usar o sistema inteiro de ponta a ponta.

Ele não precisa ser exaustivo, mas deve ser capaz de expor os principais problemas. O teste fumaça deve ser bastante rigoroso de forma que, se a construção passar, você pode assumir que ele é estável o suficiente para ser testado mais rigorosamente. *Diegão, essa linguagem está muito técnica, manda aquele exemplo maroto?* Claro, seus lindos! Deixe-me contar uma coisa muito comum no mundo dos programadores...

Por vezes, você passa meses desenvolvendo um novo software. Como você está seguro e confiante de tudo que tem desenvolvido, você nem sequer testa o que você fez e já chama o cliente para o qual você está desenvolvendo o software para apresentar sua obra de arte. O cliente traz toda a equipe dele para ver o novo software e você começa a sua apresentação. **São quinze pessoas esperando que você mostre o novo sistema e você mal começa a usar e... bug!**

Pior que isso! É um bug impeditivo, que te impossibilita de apresentar o restante do sistema. *Dá um exemplo, Diego?* Claro, você vai mostrar o sistema que você fez e ele dá um erro logo na hora de logar no sistema, ou seja, como você não consegue sequer logar, você não consegue apresentar nada para o seu cliente e a equipe de quinze pessoas. **Galera, juro para vocês, eu já vi isso acontecer tantas vezes... é frustrante, decepcionante e constrangedor.**

O teste de fumaça (*smoke testing*) é aquele teste que busca verificar se o software roda e provê suas funcionalidades e características básicas. **Prestem atenção: a ideia do teste de fumaça nem é ser detalhista como vários outros que nós vimos – é simplesmente ver se o software roda e se ele faz o básico, fundamental, o caminho feliz do software.** Caso ele passe nesse teste, ele estará habilitado a receber testes mais detalhados posteriormente. *Bacana?*

(MPE/SC – 2014) Nos termos de Engenharia de Software, assinale a alternativa que contém o tipo de teste de software correspondente àquele realizado imediatamente após a conclusão da fase de desenvolvimento, e que visa testar se o software roda e provê suas funcionalidades e características básicas, de forma a estar habilitado para

receber testes mais detalhados, ou se ele possui algum problema estrutural básico (ex. o software não roda) que deve ser corrigido antes de realizar outros tipos de testes:

a) Teste de carga (stress testing)
b) Teste de fumaça (smoke testing)
c) Teste do sistema (system testing)
d) Teste de aceitação (acceptance testing)
e) Teste de desempenho (performance testing)

_____

Comentários: trata-se do teste de fumaça. Há apenas uma observação – ele não ocorre apenas após a fase de desenvolvimento, ele ocorre sempre que há uma nova integração, isto é, uma nova funcionalidade (Letra B).

Teste de Comparação

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Também conhecidos como testes back-to-back, eles são bem comuns em sistemas críticos. Realizam-se testes em versões diferentes do software e os resultados são comparados – não só o conteúdo, mas também o tempo de resposta. Em outras palavras, versões diferentes recebem a mesma entrada e suas saídas são comparadas. Imagine a situação em que uma empresa já possua um sistema legado/antigo e encomendou o desenvolvimento de uma nova versão do sistema.

Essa nova versão é mais moderna e conta – por exemplo – com tecnologias mais novas, linguagens de programação diferentes ou novos sistemas de armazenamento. **Nesse contexto, o teste de comparação é útil para verificar se a nova versão do sistema implementa as mesmas funcionalidades e com os mesmos resultados do sistema antigo.** Ao final, é interessante gerar um relatório que deixem expostas possíveis diferenças.

Galera, é muito comum haver uma confusão entre teste de regressão e teste de comparação. O primeiro teste verifica se uma alteração no software não inseriu bugs em partes já previamente testadas. O segundo teste executa um mesmo teste em duas versões diferentes do sistema e compara os resultados obtidos. *Entendido?* Vamos ver agora o único exercício que eu encontrei sobre esse tipo de teste...

(ABIN – 2010) Para a verificação de resultados de um protótipo de sistema, podem-se utilizar testes back-to-back, nos quais os mesmos casos de teste são submetidos ao protótipo e ao sistema em teste a fim de se produzir um relatório de diferenças.

_____

Comentários: realmente é possível utilizar o teste back-to-back (ou comparação) para os mesmos casos de teste de entrada em sistemas diferentes a fim de gerar um relatório de diferenças (Correto).

Testes Alfa e Beta

INCIDÊNCIA EM PROVA: MÉDIA

É praticamente impossível para um desenvolvedor de software prever como o cliente realmente usará um programa. As instruções de uso podem ser mal interpretadas, combinações estranhas de dados podem ser usadas regularmente; resultados que pareciam claros para o testador podem ser confusos para um usuário. **Pois bem, existem alguns tipos de testes de aceitação que são utilizados para permitir ao cliente validar todos os requisitos: testes alfa e beta.**

Ambos são conduzidos pelo usuário final e, não, por testadores, programadores ou engenheiros de software – sendo que um teste de aceitação pode variar desde um informal *test drive* até uma série de testes planejados e sistematicamente executados. Na verdade, um teste de aceitação pode ser executado por um período de semanas ou meses, descobrindo assim erros cumulativos que poderiam degradar o sistema ao longo do tempo.

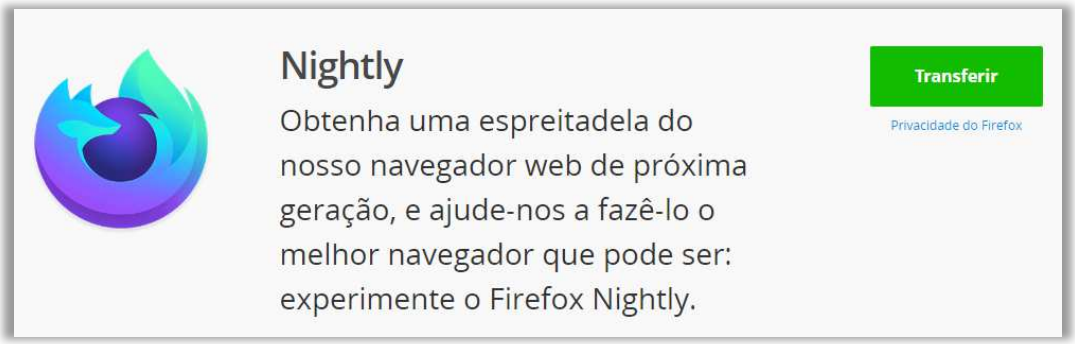
Se um software for desenvolvido como um produto para ser usado por muitos clientes, é impraticável executar testes formais de aceitação para cada cliente. **Muitos construtores de software usam os testes alfa e beta para descobrir erros que somente o usuário final parece ser capaz de encontrar.** Pois bem, então vamos descobrir agora a grande diferença entre os testes alfa e os testes beta.

Testes Alfa são testes de aceitação executados quando o desenvolvimento está próximo a sua conclusão. Este tipo de teste ocorre no ambiente do desenvolvedor e é geralmente realizado por um grupo representativo de clientes e usuários finais e, não, por programadores ou testadores. Em suma, eles são executados por usuários nas instalações do desenvolvedor do software, isto é, em um ambiente controlado. **Na prática, o programador fica olhando e registrando erros.**

Testes Beta são testes de aceitação executados quando o desenvolvimento está praticamente concluído e quando o maior número possível de defeitos precisa ser encontrado antes do lançamento do produto. Este tipo de teste também é realizado por clientes e usuários finais e, não, por programadores ou testadores. **Ademais, eles ocorrem nas instalações do usuário, isto é, no local real de utilização/trabalho – em um ambiente não controlado.**

Diferentemente do teste alfa, o desenvolvedor geralmente não está presente. Logo, o teste beta é uma aplicação "ao vivo" do software em um ambiente que não pode ser controlado pelo desenvolvedor. O cliente registra todos os problemas (reais ou imaginários) encontrados e relata para o desenvolvedor em intervalos regulares. **Os engenheiros de software, em geral, fazem modificações e então preparam a liberação do software para todos os clientes.**

Galera, uma dica! **Quando eu estudava para concursos, eu tinha grande dificuldade de memorizar qual era o Teste Alfa e o Beta.** *Como eu fiz para memorizar?* Bem, eu me lembrava dos aplicativos beta que eu instalava no meu computador. *Como é, Diegão?* Por exemplo: Firefox Nighly! Para quem não sabe, o Navegador Firefox possui várias versões (entrem no site e vejam vocês mesmos...).



Uma dessas versões é chamada Firefox Nightly e ela é disponibilizada antes da versão final para quem quiser testar, descobrir as próximas novidades e reportar eventuais erros de funcionalidades, compatibilidade, estabilidade, etc. Percebam: quem testa o software é o usuário final em seu próprio computador. **No caso, eu no meu quarto – um ambiente não controlado pelos programadores.** Vejam abaixo o link para enviar um feedback...

(TCE/RO – 2007) O teste alfa (alpha test) é conduzido pelo:

a) cliente, no seu próprio ambiente.
b) cliente, no ambiente do desenvolvedor.
c) analista de teste, no ambiente do desenvolvedor.
d) analista de teste, no seu próprio ambiente.
e) desenvolvedor, no seu próprio ambiente.

_____
Comentários: teste alfa é conduzido pelo cliente no ambiente do desenvolvedor (Letra B).

(COPASA – 2018) Teste realizado na instalação do desenvolvedor com os usuários finais, em um ambiente controlado, para identificar erros e problemas de uso durante a operação do sistema pelos usuários é denominado:

a) Teste Alfa.
b) Teste Beta.
c) Teste de Regressão.
d) Teste Fumaça.

_____

Comentários: realizado na instalação do desenvolvedor com os usuários finais em um ambiente controlado é o teste alfa (Letra A).

(COPASA – 2018) Teste realizado em ambiente de produção por um grupo de usuários finais para identificar problemas e realizar as devidas correções antes de liberar o software para toda a base de clientes:

a) Teste Alfa.
b) Teste Beta.
c) Teste de Regressão.
d) Teste Fumaça.

_____

Comentários: realizado no ambiente de produção (ambiente operacional) é o teste beta (Letra B).

(DATAPREV – 2011) Com relação aos testes de software, convenciona-se dizer que:

a) Teste Alfa é realizado com o software em fase de codificação e o Teste Beta é feito com o software já pronto.

b) Teste Alfa é feito em ambiente controlado e o Teste Beta, no ambiente do usuário.

c) Teste Alfa é feito pelos usuários e o Teste Beta, pelos desenvolvedores, simultaneamente.

d) Teste Alfa é feito na fase de criação e o Teste Beta, na fase de produção

e) Teste Alfa é feito sem os dados e o Teste Beta é feito com dados fictícios.

_____

Comentários: testes alfa são realizados em ambiente controlado e testes beta são realizados no ambiente do usuário (Letra B).

Testes de Recuperação

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Muitos sistemas de computador devem se recuperar de falhas e retomar o processamento em pouco ou nenhum tempo de parada. Em alguns casos, um sistema tem de ser tolerante a falhas; ou seja, falhas no processamento não devem causar a paralisação total do sistema. Em outros casos, uma falha no sistema deve ser corrigida dentro de um determinado período de tempo, caso contrário, poderão ocorrer sérios prejuízos financeiros.

O teste de recuperação é um teste do sistema que força o software a falhar de várias formas e verifica se a recuperação é executada corretamente. Se a recuperação for automática (executada pelo próprio sistema), a reinicialização, os mecanismos de verificação, recuperação de dados e reinício são avaliados quanto à correção. Se a recuperação requer intervenção humana, o tempo médio de reparo (MTTR[1]) é avaliado para determinar se está dentro dos limites aceitáveis.

Esses testes validam a capacidade e qualidade da recuperação do software após crashes, falhas de hardware ou outros problemas catastróficos. Em outras palavras, os testes de recuperação forçam o software a falhar de várias formas e verificam se a recuperação foi adequada, podendo ocorrer manual ou automática. Galera, aqui não tem muito segredo... vamos partir direto para as perguntas sobre esse tema.

(EPE – 2007) Devido ao aumento da demanda por aplicações que funcionem 24 horas por dia na Internet, um software deve ser capaz de manter-se em operação após uma determinada falha. A estratégia de teste que melhor garante essa característica é o(a):

a) teste de estresse.
b) teste de recuperação.
c) teste unitário.
d) debugging.
e) simulação combinatória.

_____

Comentários: o termo estratégia de teste não é o mais adequado; de todo modo, o teste que melhor garante a característica de ser capaz de manter um software em operação após uma determinada falha é o teste de recuperação (Letra B).

(MPE/ES – 2013) Há uma técnica de teste de software chamada de teste de recuperação, que tem como premissa:

a) fazer com que o software falhe e verificar se ocorre, corretamente, sua recuperação.

[1] Mean Time To Repair

b) rodar um teste de grande tempo de duração, esperando que não haja falhas registradas no software.

c) verificar se apenas usuários autorizados conseguem acessar o sistema.

d) verificar se o software atende adequadamente aos limites máximos de utilização estabelecidos.

e) verificar se são atendidos os tempos máximos e médios especificados para a realização das tarefas.

_____

Comentários: mais uma vez, o termo não foi adequado; de todo modo, vamos avaliar: (a) Correto, essa é a premissa fundamental do teste de recuperação; (b) Errado, está mais próximo a um teste operacional; (c) Errado, isso não tem nenhuma relação com teste de recuperação; (d) Errado, está mais próximo a um teste de carga; (e) Errado, está mais próximo a um teste de carga (Letra A).

(COPASA – 2018) O "Teste de Recuperação" força o software a falhar de diversos modos e verifica se a recuperação é adequadamente realizada.

_____

Comentários: a questão está impecável (Correto).

(PETROBRÁS – 2013) O teste de recuperação é um teste de sistema que força o software a falhar de diversos modos e verifica se a recuperação é adequadamente realizada, seja ela feita de forma automática (realizada pelo próprio sistema) ou requerendo intervenção humana.

_____

Comentários: a questão está impecável – ademais, realmente pode ocorrer de forma automática ou manual (Correto).

Testes de Compatibilidade

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Diferentes computadores, dispositivos de imagem, sistemas operacionais, navegadores e velocidades de conexão de rede podem ter influência significativa na operação de software. Cada configuração de computador pode resultar em diferenças nas velocidades de processamento no lado do cliente, resoluções de tela e velocidades de conexão. Excentricidades de sistemas operacionais podem causar problemas no processamento da aplicação de software.

Diferentes navegadores, às vezes, produzem resultados ligeiramente diferentes. **Em alguns casos, pequenos problemas de compatibilidade não apresentam dificuldades significativas, mas – em outros – podem ser encontrados erros graves.** Por exemplo, velocidades de download podem se tornar inaceitáveis, a falta de um plug-in necessário pode tornar o conteúdo indisponível, diferenças entre navegadores podem mudar significativamente o layout da página, etc.

O teste de compatibilidade procura descobrir esses problemas antes que a aplicação de software fique disponível. A finalidade desses testes é descobrir erros ou problemas de execução que podem ser atribuídos a diferenças em configuração. Em suma: eles validam a capacidade do software de ser executado em um ambiente particular de hardware, software, sistema operacional, rede, entre outros – podendo ser automático ou manual.

(MEC – 2011) O teste de compatibilidade serve para verificar se um software pode ser executado no sistema operacional Solaris.

_____

Comentários: eles realmente servem para testar um software em ambientes diferentes de hardware, software, sistema operacional, redes, entre outros (Correto).

Testes de Segurança

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Qualquer sistema de computador que trabalhe com informações sensíveis ou que cause ações que podem inadequadamente prejudicar (ou beneficiar) indivíduos, é um alvo para acesso impróprio ou ilegal. **As invasões abrangem uma ampla gama de atividades: hackers que tentam invadir sistemas por diversão, funcionários desgostosos que tentam invadir por vingança, indivíduos desonestos que tentam invadir para obter ganhos pessoais ilícitos.**

O teste de segurança – teste de ameaça ou invasão – tenta verificar se os mecanismos de proteção incorporados ao sistema vão de fato protegê-lo contra acesso indevido. Com tempo e recursos suficientes, um bom teste de segurança finalmente conseguirá invadir o sistema. **O papel do criador do sistema é tornar o custo da invasão maior do que o valor das informações que poderiam ser obtidas.** Durante o teste, o testador faz o papel do indivíduo que quer invadir!

O testador pode tentar obter senhas por meios externos; pode atacar o sistema com software personalizado projetado para romper defesas; pode sobrecarregar o sistema, o sistema pode assim recusar serviço a outros; pode causar erros no sistema propositadamente, esperando poder invadir durante a recuperação; pode examinar dados que não estão em segurança, tentando encontrar a chave para a entrada no sistema.

(FUB – 2018) Para que um teste de invasão leve informações úteis à segurança de uma aplicação web, é importante que tal aplicação esteja em um estágio avançado no seu ciclo de desenvolvimento.

_____

Comentários: aplicações em estágios iniciais de desenvolvimento geralmente sequer possuem funcionalidades implementadas, logo não possuem informações úteis e não são de interesse de testes de invasão (Correto).

(COPASA – 2018) O "Teste de Segurança" verifica se os mecanismos de proteção incorporados a um sistema vão de fato protegê-lo de invasão imprópria.

_____

Comentários: a questão está perfeita – essa é a função de um teste de segurança (Correto).

Testes de Vulnerabilidade

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Testes de Vulnerabilidade são processos essenciais no âmbito da segurança da informação. Eles têm como objetivo identificar, quantificar e classificar vulnerabilidades em sistemas, redes e aplicações – a ideia é descobrir falhas de segurança antes que um invasor o faça. Realizar testes de vulnerabilidade regularmente é fundamental para qualquer organização que deseje proteger suas informações e sistemas de tecnologia da informação contra ameaças cibernéticas.

Esses testes fornecem uma visão essencial da postura de segurança da organização e ajudam a orientar as estratégias de mitigação de riscos, podendo ser realizados de forma manual ou automatizada (com ferramentas como Nessus, OpenVAS, Qualys e Rapid7). **As vulnerabilidades mais comuns incluem configurações incorretas, software desatualizado, falhas em protocolos de rede, e problemas em códigos de aplicações.**

As principais etapas dos testes de vulnerabilidade incluem: definir o escopo e os objetivos do teste; utilizar ferramentas para identificar sistemas ativos e aberto, e potenciais pontos de entrada; determinar a criticidade das vulnerabilidades encontradas; em alguns casos, pode-se tentar explorar vulnerabilidades para entender o impacto potencial; documentar as vulnerabilidades encontradas e fornecer recomendações para mitigá-las; e corrigir as vulnerabilidades identificadas.

Os testes de vulnerabilidade são cruciais para manter a segurança cibernética e proteger dados sensíveis. Eles ajudam a evitar ataques cibernéticos, como hacking e vazamento de dados. Frequentemente, ade são exigidos para conformidade regulatória em várias indústrias, como parte das melhores práticas de segurança. **Idealmente, eles devem ser feitos regularmente devido ao surgimento constante de novas vulnerabilidades e à evolução das ameaças cibernéticas.**

A área de testes de vulnerabilidade está em constante evolução, com novas ferramentas e técnicas sendo desenvolvidas para enfrentar as ameaças emergentes.

(PROF. DIEGO / INÉDITA – 2024) É recomendável executar os testes de vulnerabilidade em ambientes de produção a fim de garantir resultados mais precisos e realistas.

_____

Comentários: embora testar em ambientes de produção possa fornecer dados realistas, isso pode apresentar riscos significativos de interrupção dos serviços e outros impactos operacionais. Normalmente, os testes são realizados em ambientes de teste ou desenvolvimento para evitar tais riscos (Errado).

(PROF. DIEGO / INÉDITA – 2024) Testes de Vulnerabilidade, ao identificarem falhas de segurança, fornecem soluções específicas para corrigir cada vulnerabilidade encontrada.

_____

Comentários: parte do processo de teste de vulnerabilidade inclui não apenas a identificação de vulnerabilidades, mas também a recomendação de medidas corretivas. Essas recomendações são geralmente detalhadas e específicas, orientando os responsáveis pela segurança da informação sobre como remediar as falhas identificadas (Correto).

Testes Automatizados

INCIDÊNCIA EM PROVA: MÉDIA

Imagine que você é o proprietário de uma fábrica de bolo e precisa garantir que seus bolos sempre saiam perfeitos. Para isso, você decide implementar um sistema de testes automatizados, que irá testar cada bolo que sair da linha de produção.

Assim como os testes automatizados, o sistema que você implementou irá seguir um conjunto de regras para determinar se o bolo está bom ou não. Por exemplo, o sistema pode medir o peso e a altura do bolo para garantir que ele atenda a determinados padrões. Se o bolo não atender a esses padrões, o sistema irá automaticamente marcá-lo como rejeitado e enviar um alerta para um funcionário inspecionar e corrigir o problema.

Da mesma forma, os testes automatizados em software seguem um conjunto de regras predefinidas para determinar se o software está funcionando corretamente ou não. Eles são executados automaticamente pelo sistema de integração contínua, que verifica se todas as funcionalidades estão de acordo com o esperado, identificando erros e avisando o desenvolvedor para que ele possa corrigi-los antes de enviar o código para produção.

Os testes automatizados podem ser comparados a um sistema de segurança em uma casa. Assim como um sistema de segurança verifica continuamente se há alguma violação, os testes automatizados são projetados para verificar continuamente se o software está funcionando corretamente. **Os testes automatizados são uma forma de validar se o software está funcionando corretamente sem a necessidade de intervenção manual.**

Eles são executados automaticamente e ajudam a identificar erros e falhas de forma rápida e eficiente. **Isso significa que as equipes de desenvolvimento podem identificar e corrigir problemas mais rapidamente, reduzindo o tempo necessário para lançar novas versões de software.** Existem diferentes tipos de testes automatizados, incluindo testes de unidade, testes de integração, testes de sistema e testes de aceitação.

Cada tipo de teste é projetado para verificar **diferentes aspectos do software**. Testes de unidade, por exemplo, são usados para testar pequenas partes individuais do software, enquanto os testes de sistema são usados para verificar se o software está funcionando corretamente como um todo. Os benefícios dos testes automatizados são muitos, incluindo redução de erros e falhas, a melhoria da qualidade do software, a economia de tempo e recursos, e a redução de custos a longo prazo.

Além disso, os testes automatizados podem ser executados de forma **consistente e repetitiva**, o que garante a validação contínua do software. *E sobre as ferramentas de automação?* Bem, elas são programas que permitem a criação, execução e gerenciamento de testes automatizados. Essas ferramentas são usadas para ajudar a garantir a qualidade do software, automatizando processos repetitivos de teste e permitindo que os testes sejam executados de forma mais rápida e eficiente.

Há uma série de critérios que você precisa levar em conta ao escolher a ferramenta de automação de teste perfeita. Devem ser considerados critérios, tais como facilidade de utilização e criação de testes, suporte a múltiplas plataformas e navegadores, facilidade de análise,

flexibilidade, custo e o poder dos recursos do software. Algumas das ferramentas de automação de testes mais populares incluem:

FERRAMENTAS	DESCRIÇÃO
SELENIUM	Usado para testar aplicativos web, o Selenium é uma ferramenta de código aberto que oferece uma variedade de recursos para automatizar testes, incluindo a gravação de ações do usuário e a execução de testes em diferentes navegadores.
APPIUM	Utilizado para testar aplicativos móveis, o Appium é uma ferramenta de automação de testes móveis que permite a execução de testes em diferentes dispositivos móveis, como smartphones e tablets.
JMETER	Uma ferramenta de código aberto usada para testar o desempenho de aplicativos web, o JMeter permite a criação de cenários de teste para simular diferentes volumes de tráfego e medir a resposta do aplicativo.
ROBOT	Uma ferramenta de automação de testes de código aberto que permite a criação de testes automatizados usando palavras-chave em inglês simples. É utilizado para testar aplicativos web, móveis e de desktop.
JUNIT	Uma ferramenta de teste de unidade para a linguagem Java. Ele fornece uma série de anotações e assertivas para facilitar a criação de testes automatizados e o desenvolvimento orientado a testes.

É importante lembrar que a escolha da ferramenta de automação de testes deve levar em consideração as necessidades específicas do projeto e da equipe de testes. Cada ferramenta tem suas próprias vantagens e desvantagens e pode ser mais adequada para diferentes tipos de testes e ambientes. *Agora por que é tão recomendável a utilização de testes automatizados em alguns cenários?* Vejamos...

BENEFÍCIOS	DESCRIÇÃO
AUMENTO DE ESCALA	Automatizar seus testes transforma a escala na qual sua equipe de teste opera. Isso ocorre porque os computadores podem executar testes 24 horas por dia, 7 dias por semana. Mesmo seus engenheiros de qualidade mais atentos só podem gerenciar 60 horas por semana! O resultado é que você pode executar muito mais testes com os mesmos recursos. Isso é importante, já que os engenheiros de teste são um recurso relativamente escasso.
VELOCIDADE DE ENTREGA	É comum haver uma constante pressão sobre equipes de desenvolvimento para lançar novos recursos. No entanto, lançar um aplicativo com bugs pode acabar com todos esses ganhos em minutos. Testes automatizados aceleram significativamente os testes de regressão. Por sua vez, isso reduz o tempo entre a integração de um novo recurso e o lançamento. Ao final, quanto mais rápido você conseguir subir novos recursos, mais competitivo você será.
LANÇAMENTOS	A abordagem tradicional para o desenvolvimento de software vê todos os testes feitos depois que o produto é desenvolvido. Mas se você usar a automação de teste, poderá testar seu aplicativo constantemente durante o desenvolvimento. Cada vez que um novo código é enviado, você pode executar testes de fumaça. Qualquer novo recurso pode ser testado assim que estiver estável. Isso significa que todo o seu processo de lançamento se torna mais eficiente e simplificado.

Vejamos na tabela seguinte uma **comparação entre a realização de testes manuais e testes automatizados**:

CRITÉRIO	TESTE MANUAL	TESTE AUTOMATIZADO
VELOCIDADE	Lento, requer esforço manual e execução de casos de teste.	Mais rápido, permite a execução simultânea de casos de teste usando ferramentas de automação.
CONFIABILIDADE	Mais propenso a erros humanos.	Menos propenso a erros humanos, pois os casos de teste são executados usando ferramentas de automação e podem ser repetidos de forma consistente.
MANUTENÇÃO	Requer mais esforço para manter grandes conjuntos de testes.	Requer mais esforço para configurar inicialmente, mas requer um esforço mínimo para manter depois que os scripts de automação são criados.
REUSABILIDADE	Os testes podem ser reutilizados, mas exigem execução manual, limitando a escalabilidade dos testes.	Os testes podem ser reutilizados várias vezes com o mínimo de esforço, permitindo testes mais abrangentes e escaláveis.
ESCOPO	Limitado no escopo, requer esforço manual e tempo, dificultando a obtenção de cobertura total de testes.	Pode abranger um escopo mais amplo de testes, incluindo testes de regressão, facilitando a obtenção de cobertura total de testes.
CUSTO	Menor custo inicial, pois não requer ferramentas especializadas ou tecnologia, mas pode se tornar mais caro ao longo do tempo devido ao aumento dos custos de mão de obra.	Custo inicial mais alto, pois requer ferramentas e tecnologia especializadas, mas pode se tornar mais econômico ao longo do tempo devido ao aumento da eficiência.
HABILIDADES	Requer um testador com habilidades de teste manual, incluindo uma compreensão da aplicação que está sendo testada e a capacidade de identificar e relatar problemas.	Requer um testador com habilidades de teste de automação, incluindo linguagens de programação e ferramentas de automação, bem como a capacidade de escrever scripts de teste automatizados.

Existem alguns fatores a serem considerados ao decidir se um teste pode e deve ser automatizado. Aqui estão os mais importantes:

FATORES	DESCRIÇÃO
REPETÍVEL	O teste deve ser aquele que pode (e será) repetido regularmente. Por exemplo, não adianta automatizar um teste para um recurso que está prestes a ser preterido.
DETERMINANTE	Tem que haver um resultado claro certo e errado para o teste. Em outras palavras, deve ser fácil para um computador decidir se o teste falhou ou não.
TEDIOSO	Via de regra, os seres humanos são muito pobres em tarefas repetitivas. Nossas mentes vagam ou nos distraímos. Qualquer teste que envolva fazer repetidamente a mesma ação é melhor deixar para um computador automatizando-o.

CRÍTICO	Se um teste é absolutamente crítico, você deve tentar o seu melhor para automatizá-lo e programá-lo para ser executado regularmente. Dessa forma, você pode ter certeza de que esse teste está sempre sendo realizado.

Esses foram os critérios a serem considerados para se utilizar testes automatizados. *E para se utilizar testes manuais?* Vejamos:

FATORES	DESCRIÇÃO
MUDANÇAS CONSTANTES	Testes em que o resultado correto muda com frequência não podem ser automatizados. Da mesma forma, testes em que o resultado nem sempre é claro.
TESTES AD-HOC	Às vezes, você precisa testar para verificar uma condição específica ou para procurar um bug relatado. Esses testes ad-hoc não são adequados para automação. No entanto, se você encontrar etapas para recriar o bug, convém automatizar o teste.
EVOLUÇÃO DE FUNCIONALIDADES	À medida que um novo recurso é desenvolvido, você precisa desenvolver seus testes em paralelo. Normalmente, não vale a pena automatizar o teste enquanto o recurso ainda está em constante evolução.

Nós podemos dizer que existem quatro etapas para criar um teste automatizado, conforme vemos a seguir:

ETAPAS	DESCRIÇÃO
ESCOLHA UMA FERRAMENTA OU ESTRUTURA ADEQUADA PARA EXECUTAR OS TESTES	Isso dependerá do tipo de testes que você está automatizando – existem dezenas de ferramentas.
DEFINA COM PRECISÃO SEU CASO DE TESTE	Isso significa anotar cada passo e o resultado necessário. É importante não fazer suposições e não perder nenhuma etapa que um testador manual possa fazer sem pensar nisso. Por exemplo, aceitar um pop-up de banner de cookie.
CONVERTA SEU CASO DE TESTE EM UM TESTE QUE PODE SER EXECUTADO NA ESTRUTURA ESCOLHIDA	Isso geralmente pode envolver a escrita de um script personalizado. Você precisa verificar se seu teste realmente funciona corretamente e se ele funciona em todos os casos que você precisa testar. Por exemplo, se seu aplicativo precisar ser executado em navegadores diferentes, você precisará garantir que seu teste funcione em todos os navegadores.
EXECUTE O TESTE E AVALIE O RESULTADO	Essa etapa às vezes pode ser difícil. Muitas vezes, as falhas de teste não aparecem imediatamente e pode ser necessário algum trabalho de detetive para descobrir o que realmente deu errado. Além disso, muitas falhas de teste são "falsos positivos" desencadeados por alguma pequena alteração no aplicativo. Nesses casos, você precisará atualizar seu teste e executá-lo novamente.

Depois de escolher sua ferramenta, você precisa aplicar a automação de testes da maneira correta. As práticas recomendadas a seguir se aplicam ao teste de nível do sistema:

- **Planeje seus testes cuidadosamente.** Certifique-se de que seus casos de teste estejam claramente definidos e bem escritos. Os casos de teste devem ser independentes e fáceis de entender.

- **Teste o mais cedo e com a maior frequência possível durante o desenvolvimento.** Quanto mais cedo você identificar um bug, mais fácil será corrigi-lo. Por outro lado, bugs encontrados tardiamente exigem muito mais esforço para serem corrigidos.

- **Planeje a ordem em que os testes são executados.** Muitas vezes, você pode usar um teste para criar o estado necessário para um segundo teste. Por exemplo, criar um usuário antes de testar a página da conta.

- Se possível, **use ferramentas que possam agendar testes automaticamente**, seja quando um novo código for enviado por push ou de acordo com uma agenda. Isso garante que seu código seja sempre bem testado.

- **Configure alertas para que você seja informado assim que um teste falhar.** Em seguida, decida se você deve abortar a execução de teste atual ou deixá-la completar. Por exemplo, se for um bug sério, provavelmente é melhor abortar e corrigir o bug.

- Lembre-se de que você precisa **reavaliar constantemente seus planos de teste à medida que seu aplicativo se desenvolve e muda**. Não adianta testar recursos herdados que não fazem mais parte dos aplicativos.

Por fim, vamos falar sobre os Princípios FIRST! *O que seria isso, Diego?* **Trata-se de um conjunto de diretrizes concebidas para melhorar a qualidade e a eficácia dos testes unitários no desenvolvimento de software.** Cada letra do acrônimo FIRST representa uma característica fundamental que os testes unitários devem possuir: [F]ast, [I]solated/[I]ndependent, [R]epeatable, [S]elf-Validating e [T]imely. Vamos detalhar:

PRINCÍPIO	DESCRIÇÃO
FAST [RÁPIDOS]	Os testes unitários devem ser rápidos para executar. A rapidez é crucial porque permite que os desenvolvedores os executem frequentemente, o que, por sua vez, facilita a detecção e correção de erros precocemente no ciclo de desenvolvimento. Testes lentos podem se tornar um gargalo, desencorajando sua execução regular e comprometendo a agilidade do desenvolvimento. Testes rápidos são cruciais em um ambiente de testes automatizados, permitindo que a suíte de testes seja executada frequentemente sem atrasar o processo de desenvolvimento. Isso facilita a integração contínua e a entrega contínua (CI/CD).
ISOLATED/ INDEPENDENT [ISOLADOS/ INDEPENDENTES]	Cada teste unitário deve ser isolado e independente dos outros. Isso significa que a execução de um teste não deve depender do resultado de outro teste, nem alterar o ambiente de forma a afetar outros testes. Essa característica garante que falhas em um teste não mascarem ou causem falhas em outros testes, facilitando a identificação de problemas específicos no código.

	A independência dos testes garante que os testes automatizados possam ser executados em qualquer ordem ou em paralelo, sem interferências, aumentando a confiabilidade dos resultados e a eficiência do processo de teste.
REPEATABLE [REPETÍVEIS]	Os testes unitários devem ser repetíveis em qualquer ambiente, e seus resultados devem ser consistentes, independentemente de onde ou quando são executados. Isso assegura que os testes sejam confiáveis e que suas falhas indiquem problemas reais no código, e não inconsistências no ambiente de teste. A capacidade de repetir os testes em diferentes ambientes assegura que os testes automatizados sejam robustos e confiáveis, indicando que falhas nos testes refletem problemas reais no código.
SELF-VALIDATING [AUTOVALIDÁVEIS]	Os testes unitários devem ser autovalidáveis, o que significa que eles devem automaticamente indicar se passaram ou falharam, sem a necessidade de interpretação manual dos resultados. Isso aumenta a eficiência do processo de teste, permitindo que os desenvolvedores identifiquem rapidamente se o código atende aos critérios de correção especificados. Testes que automaticamente indicam se passaram ou falharam são fundamentais para a automação, permitindo que sistemas de integração contínua processem e relatem os resultados dos testes sem intervenção humana.
TIMELY [OPORTUNOS]	Os testes unitários devem ser escritos de forma oportuna, o que geralmente significa escrevê-los antes ou simultaneamente ao desenvolvimento do código que eles testam. Esse princípio está alinhado com a prática de Desenvolvimento Guiado por Testes (TDD - Test-Driven Development), onde os testes ajudam a guiar o design do código e garantem que o software seja construído com testabilidade em mente desde o início. Escrever testes no momento certo, especialmente antes do código de produção, como promovido pelo TDD, é essencial para garantir que a base de testes automatizados seja relevante e abrangente.

Como pudemos ver, esses princípios estão profundamente interligados com testes automatizados e com a prática de TDD, servindo como um guia para criar testes unitários eficazes e de alta qualidade dentro desses contextos. Seguir esses princípios ajuda a maximizar os benefícios dos testes automatizados, incluindo a melhoria da qualidade do código, a eficiência do desenvolvimento e a capacidade de manter altos padrões de qualidade de software.

(CESPE / TCE-RJ – 2022) Quando aplicados os princípios FIRST em testes de unidade, o S representa a autovalidação, que gera um resultado a ser interpretado pelo responsável.

_____

Comentários: a afirmação erra ao associar o "S" de Self-Validating (Autovalidação) a algo que necessita interpretação externa; na verdade, testes autovalidáveis devem produzir um resultado claro (aprovado ou reprovado) sem necessidade de interpretação adicional (Errado).

(FCC / TRE-AP – 2015) O TDD – Test Driven Development (Desenvolvimento orientado a teste):

a) é parte das metodologias ágeis UP – Unified Process e XP – Extreme Programming, tendo sido criado para ser usado em metodologias que respeitam os 4 princípios do Manifesto Ágil.

b) transforma o desenvolvimento, pois deve-se primeiro implementar o sistema antes de escrever os testes. Os testes são utilizados para facilitar no entendimento do projeto e para clarear o que se deseja em relação ao código.

c) baseia-se em um ciclo simples: escreve-se um código -> cria-se um teste para passar no código -> refatora-se.

d) propõe a criação de testes que validem o código como um todo para reduzir o tempo de desenvolvimento.

e) beneficia-se de testes que seguem o modelo FIRST: F (Fast) I (Isolated) R (Repeatable) S (Self-verifying) T (Timely).

Comentários:

(a) Errado. O TDD é uma prática comum em metodologias ágeis como o XP (Extreme Programming), baseando-se nos princípios do Manifesto Ágil, que promove desenvolvimento iterativo, colaboração, e resposta a mudanças. No entanto, UP (Unified Process) não é uma metodologia ágil – ele é baseado no RUP (Rational Unified Process);

(b) Errado. O TDD, ao contrário, enfatiza a escrita dos testes antes do desenvolvimento do sistema propriamente dito. Nesse sentido, os testes orientam o desenvolvimento do código, assegurando que o código atenda aos requisitos definidos desde o início;

(c) Errado. O ciclo do TDD é mais específico e estruturado do que o descrito: primeiro escreve-se um teste que falha, depois implementa-se o código necessário para passar no teste e, por fim, refatora-se o código para melhorar a qualidade sem alterar seu comportamento;

(d) Errado. A ideia do TDD não é validar o código como um todo de imediato, mas sim promover um desenvolvimento incremental e contínuo por meio de testes pequenos e específicos que validam partes do código à medida que são desenvolvidas;

(e) Correto. Os testes no TDD são benéficos quando seguem o modelo FIRST, que destaca características importantes que os testes devem ter para serem eficazes no desenvolvimento orientado a testes, como serem rápidos, isolados, repetíveis, autovalidáveis e oportunos (Letra E).

Testes Ágeis

Nos processos ágeis, os testes são integrados ao ciclo de desenvolvimento de software de maneira contínua e iterativa, com ênfase na colaboração entre desenvolvedores, testadores e clientes para garantir a entrega rápida de software funcional e de alta qualidade. **Testes em processos ágeis são caracterizados por sua integração contínua ao desenvolvimento, foco na colaboração, práticas como TDD/BDD, e uma abordagem adaptativa que busca a melhoria contínua dos processos:**

CARACTERÍSTICAS	DESCRIÇÃO
TESTES DESDE O INÍCIO	Em ambientes ágeis, os testes começam no início do ciclo de vida do desenvolvimento de software e são realizados de forma contínua até o final do projeto. Isso significa que os testes são conduzidos simultaneamente com o desenvolvimento, permitindo a detecção e correção imediata de defeitos.
COLABORAÇÃO E COMUNICAÇÃO	Há uma forte ênfase na comunicação e colaboração entre desenvolvedores, testadores e clientes. Os testadores participam das reuniões de planejamento, revisão e retrospectiva para garantir que os critérios de aceitação estejam claros e sejam atendidos.
DESENVOLVIMENTO ORIENTADO POR TESTES	TDD é uma prática comum em ambientes ágeis, onde os testes são escritos antes do código de produção. Isso ajuda a garantir que o código atenda aos requisitos desde o início e facilita a refatoração e a manutenção do código ao longo do tempo.
DESENVOLVIMENTO ORIENTADO A COMPORTAMENTO	BDD expande o TDD ao escrever testes de aceitação em uma linguagem próxima da natural, com foco no comportamento do software do ponto de vista do usuário. Isso promove a compreensão compartilhada dos requisitos entre a equipe e os stakeholders.
AUTOMAÇÃO DE TESTES	A automação é crucial em ambientes ágeis devido à necessidade de testes frequentes em várias versões do software. A automação ajuda a acelerar o processo de teste, permitindo que as equipes se concentrem em testes mais complexos e exploratórios.
INTEGRAÇÃO CONTÍNUA	Os testes são uma parte integrante da CI, onde o código é construído, testado e integrado várias vezes ao dia. Isso permite a detecção rápida de defeitos e a entrega contínua de software funcional.
TESTES EXPLORATÓRIOS	Além dos testes planejados, os testes exploratórios são encorajados para identificar problemas não cobertos pelos testes automatizados ou de aceitação. Essa abordagem depende da criatividade e da experiência do testador para explorar o software e encontrar falhas.
ADAPTAÇÃO E MELHORIA CONTÍNUA	As práticas de teste em ambientes ágeis são continuamente avaliadas e adaptadas com base no feedback dos clientes e da equipe. A retrospectiva da sprint é uma oportunidade para refletir sobre os processos de teste e fazer ajustes conforme necessário.

Doubles

Vamos falar sobre Doubles! *O que é isso, professor?* **É uma categoria de dublês de teste formada por vários conceitos, tais como: Dummy, Mocks, Stubs, Spy e Fakes.** Galera, às vezes, não é possível fazer testes de unidade em um pedaço de código devido à indisponibilidade de objetos colaboradores ou o custo de implementação para o colaborador. Dublês de Teste aliviam a necessidade de um colaborador.

O que é um colaborador, professor? É algum objeto que ajuda a realizar testes de unidade. *Sabe os dublês de filmes?* É aquele cara que substitui um ator em uma cena de ação, um ator em uma luta ou uma atriz tocando piano. Eles são utilizados para proteger atores reais ou substituir atores quando eles não estão disponíveis ou não são capazes de realizar alguma cena. Agora imaginem que estamos fazendo um sistema de pagamentos que faz uma interface com um banco.

Ora, você vai realizar pagamentos reais para testar seu código? Não, você utiliza algum dublê! Eles são substitutos qualificados para objetos do colaborador. Dessa forma, podemos dizer que **doubles são objetos ou componentes utilizados para substituir outras dependências do sistema durante a execução dos testes** – eles são usados para simular o comportamento de objetos reais, como bancos de dados, serviços web ou componentes externos.

Dito de outra maneira, ele pode isolar o código em teste e garantir que ele funcione corretamente mesmo quando essas dependências externas não estão disponíveis ou precisam ser controladas de forma previsível. Os doubles podem ser classificados em diferentes tipos, dependendo do nível de interação e comportamento que eles reproduzem. Alguns dos tipos comuns de doubles são apresentados a seguir:

DOUBLES	DESCRIÇÃO
DUMMY	Trata-se de um objeto que é passado como argumento, mas não é usado no teste em si. Geralmente, é usado apenas para preencher a assinatura do método. Imagine que você está montando um quebra-cabeça e precisa preencher um espaço vazio com uma peça específica. No entanto, você percebe que a peça não é realmente necessária para a conclusão do quebra-cabeça, mas é exigida pela regra de colocar todas as peças disponíveis. Essa peça extra e não utilizada seria um exemplo de um dummy, pois está presente apenas para cumprir uma exigência, mas não desempenha um papel significativo no resultado final.
STUB	Trata-se de um objeto que fornece respostas pré-programadas para as chamadas de método durante o teste – eles são usados para simular o comportamento básico de uma dependência. Vamos supor que você queira testar uma nova receita de bolo de chocolate, mas você não tem o ingrediente real para o glacê de chocolate. Em vez disso, você usa uma versão pré-fabricada de glacê em pó que você já sabe que dá um sabor semelhante. O glacê em pó pré-fabricado seria um exemplo de um stub, pois fornece uma resposta simulada para substituir o glacê de chocolate real.
MOCK	Trata-se de um objeto que permite verificar o comportamento das chamadas de método, bem como definir expectativas e verificar se elas foram atendidas durante o teste. Os mocks são

	usados para verificar interações específicas e garantir que o código em teste esteja se comunicando corretamente com a dependência simulada. Imagine que você está ensaiando uma peça de teatro e precisa verificar se um determinado ator segue corretamente as suas instruções em uma determinada cena. Para tal, você contrata um "ator falso" que tem a função de representar o personagem com base nas instruções que você deu. Esse "ator falso" seria um exemplo de um mock, pois você pode verificar se o ator segue as instruções corretamente e se comporta como esperado durante o ensaio.
SPY	Trata-se de um objeto que registra informações sobre as chamadas de método que foram feitas a ele. Isso permite verificar posteriormente como o objeto foi usado durante o teste. Suponha que você seja um detetive investigando um caso e deseja saber quem está acessando um determinado local à noite. Você instala câmeras de vigilância ocultas para observar as pessoas que passam pelo local e registrar suas atividades. As câmeras de vigilância seriam um exemplo de um spy, pois registram informações sobre as atividades das pessoas, permitindo que você as analise posteriormente.

Existem também os fakes, que são objetos que fornecem implementações simplificadas de componentes reais do sistema, mas com comportamento semelhante. Eles são criados para substituir as dependências reais durante os testes de software. Ao contrário dos dummies, stubs, mocks e spies, que são tipos específicos de doubles com finalidades distintas, os fakes são uma categoria mais ampla que abrange implementações simuladas de componentes reais.

Os fakes geralmente são usados quando é necessário um comportamento mais complexo do que um dummy, mas menos sofisticado do que um componente real. Eles são úteis em cenários em que a implementação real é difícil de usar, lenta, complexa demais ou simplesmente não está disponível durante os testes. **Por exemplo, em um sistema que se comunica com um banco de dados, um fake de banco de dados pode ser usado para substituir o banco de dados real durante os testes.**

Esse fake pode armazenar dados em memória, em vez de fazer chamadas ao banco de dados real, para melhorar a eficiência e a simplicidade dos testes. Eles são projetados para fornecer um comportamento simulado suficiente para que os testes possam ser executados sem a necessidade de acessar recursos externos ou dependências complexas e permitem que testes sejam executados de forma mais rápida e controlada, facilitando a detecção de problemas e o isolamento de falhas.

Vamos detalhar um pouco mais os mocks: **essa técnica é fundamental para a validação do layout do código sem afetar ou depender do ambiente em questão**. Esse termo vem do inglês e significa imitação, zombaria, falsidade, simulação, entre outros. Sabemos que os testes de unidade procuram testar classes de um sistema isoladamente, porém classes em um sistema normalmente alcançam seus objetivos com a ajuda de outras (por mais desacoplada que ela seja).

Um dos principais desafios no teste de unidade é isolar a classe que está sendo testada para que nenhuma outra classe do sistema seja envolvida no teste. Nesse momento, entram os mocks... **eles são objetos que simulam o comportamento de objetos reais de forma controlada.** *Como "controlada"?* Como eu os crio, eu tenho total controle sobre ele e sua implementação. Um objeto sob teste pode ter dependências sobre outros objetos complexos.

Os mocks devem ter a mesma interface que os objetos reais, caso contrário não conseguirão se comunicar. Eles são utilizados quando temos objetos que geram resultados variáveis; objetos com estados difíceis de serem reproduzidos; objetos lentos (como banco de dados que precisam ser inicializados antes do teste), objetos que ainda não existem ou podem ter comportamentos alterados, entre outros.

Um exemplo: estamos fazendo um programa de despertador que toca uma sirene em um horário marcado. Para testar este comportamento, o teste de unidade deve esperar chegar a hora programada para testar a sirene corretamente. Se um mock for usado no lugar do objeto real, ele pode ser programado para simular a hora de programação do despertador. Assim, o código do despertador pode ser testado corretamente e sem maior impacto no tempo de desenvolvimento.

Os mocks possuem bibliotecas que ajudam em sua criação, tais como: EasyMock, JMock, Mockito (para Java); NMock, TypeMock (para .NET), entre outros.

RESUMO

TESTE DE SOFTWARE

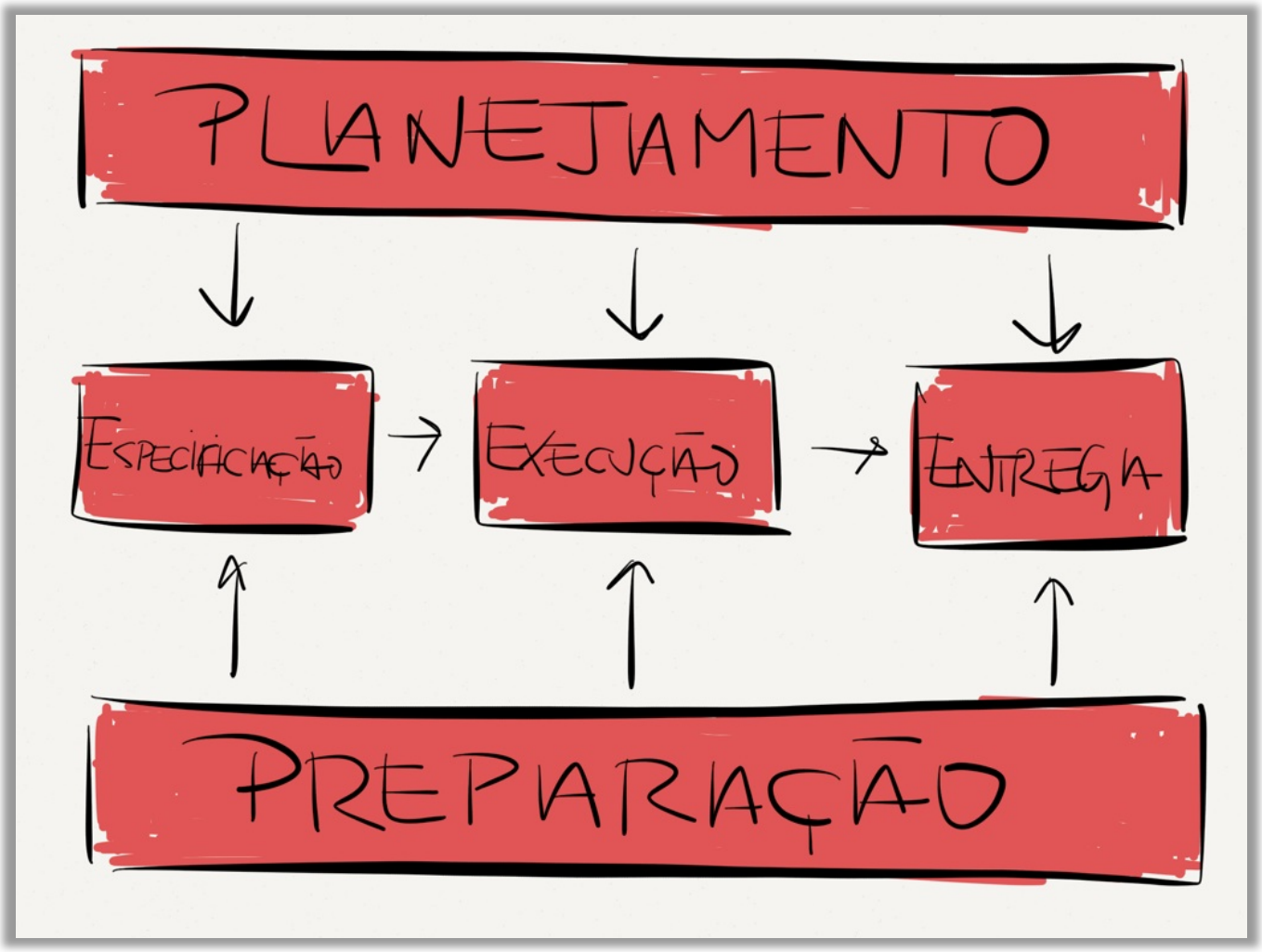
O teste de software é o processo de executar um software com dois objetivos principais: primeiro, demonstrar ao desenvolvedor e ao cliente que o software atende aos requisitos especificados; segundo, descobrir falhas ou defeitos no software que apresente comportamento incorreto, não desejável ou em não conformidade com sua especificação.

CARACTERÍSTICAS	DESCRIÇÃO
UM BOM TESTE TEM ALTA PROBABILIDADE DE ENCONTRAR DEFEITOS	Para atingir esse objetivo de encontrar defeitos, o testador deve entender o software e tentar desenvolver uma imagem mental de como o software pode falhar. O ideal é que as classes de falhas sejam investigadas.
UM BOM TESTE NÃO É REDUNDANTE	O tempo e os recursos de teste são limitados. Não tem sentido realizar um teste que tenha a mesma finalidade de outro teste. Cada teste deve ter uma finalidade diferente (mesmo que seja sutilmente diferente).
UM BOM TESTE DEVERÁ SER "O MELHOR DA RAÇA"	Em um grupo de testes com finalidades similares, as limitações de tempo e recursos podem induzir à execução de apenas um subconjunto desses testes. Nesses casos, deverá ser usado o teste que tenha a maior probabilidade de revelar uma classe inteira de erros.
UM BOM TESTE NÃO DEVE SER NEM MUITO SIMPLES NEM MUITO COMPLEXO	Embora seja possível combinar algumas vezes uma série de testes em um caso de teste, os possíveis efeitos colaterais associados com essa abordagem podem mascarar erros. Em geral, cada teste deve ser executado separadamente.

PRINCÍPIOS FUNDAMENTAIS	DESCRIÇÃO
TESTES DEMONSTRAM A PRESENÇA DE DEFEITOS...	Um teste pode demonstrar a presença de defeitos, mas não pode provar que eles não existem. Ele reduz a probabilidade de que os defeitos permaneçam, mas mesmo se nenhum defeito for encontrado não quer dizer que ele não os tenha.
TESTES EXAUSTIVOS SÃO IMPOSSÍVEIS...	Testar todas as combinações de entradas e pré-condições é inviável, exceto para casos triviais. Em vez de realizar testes exaustivos, os riscos e prioridades são levados em consideração para dar foco aos esforços de teste.
TESTE O MAIS BREVE POSSÍVEL (ANTECIPADO)...	Os defeitos encontrados nas fases iniciais do processo de desenvolvimento de software são mais baratos de serem corrigidos do que aqueles encontrados já em fase produção. Há, inclusive, técnicas de testes antes mesmo da implementação.
AGRUPEM OS DEFEITOS MAIS SENSÍVEIS...	Seguindo o Princípio de Pareto, 80% dos defeitos são causados por 20% do código. Ao identificar essas áreas sensíveis, os testes podem priorizá-las, de forma a ter alta probabilidade de encontrar defeitos.
PARADOXO DO PESTICIDA...	Caso os mesmos testes sejam aplicados repetidamente, em determinado momento eles deixam de ser úteis, ou seja, não conseguem encontrar nenhum novo defeito. Por isso, os testes precisam ser revisitados com frequência.
TESTES DEPENDEM DO CONTEXTO...	Os testes devem ser elaborados de acordo com o contexto de utilização do software. Ex: um sistema bancário deve ser testado de maneira diferente de uma rede social. Assim como testes de aplicação web têm foco diferente do desktop.

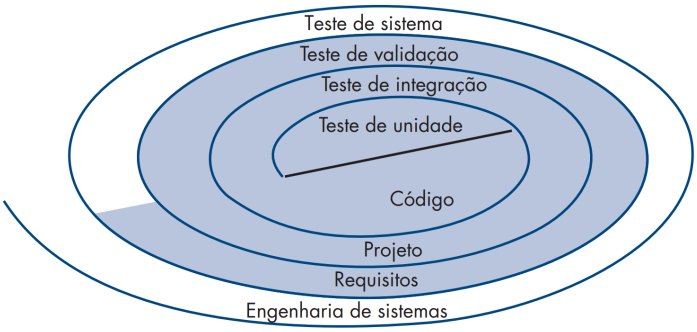
AUSÊNCIA DE DEFEITOS É UMA ILUSÃO	Identificar e corrigir os problemas de um software não garantem que ele está pronto. Os testes foram elaborados para identificar todas as possíveis falhas? O sistema atende às necessidades e expectativas dos usuários? Logo, há outros fatores!

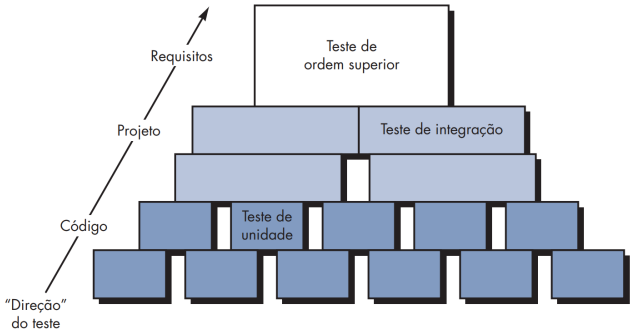
CARACTERÍSTICA	DESCRIÇÃO
OPERABILIDADE	Grosso modo, podemos dizer que "*quanto melhor funciona, mais eficientemente pode ser testado*".
OBSERVABILIDADE	Grosso modo, podemos dizer que "*o que você vê é o que você testa*".
CONTROLABILIDADE	Grosso modo, podemos dizer que "*quanto melhor você pode controlar o software, mais o teste pode ser automatizado e otimizado*".
DECOMPONIBILIDADE	Grosso modo, podemos dizer que "*controlando o escopo, podemos isolar problemas mais rapidamente e realizar testes mais inteligentes*".
SIMPLICIDADE	Grosso modo, podemos dizer que "*quanto menos houver a testar, mais rapidamente podemos testá-lo*".
ESTABILIDADE	Grosso modo, podemos dizer que "*quanto menos modificações, menos interrupções no teste*".
COMPREENSIBILIDADE	Grosso modo, podemos dizer que "*quanto mais informações temos, de forma mais inteligente vamos testar*".



ETAPAS DO PROCESSO	DESCRIÇÃO
PLANEJAMENTO	Nesta etapa, elaboram-se o Projeto de Testes e o Plano de Testes. Ela acompanha todo o processo de teste, por meio de atividades como captação de requisitos, planejamento do projeto, análise de riscos e preparação de ambiente de testes.
PREPARAÇÃO	nesta etapa, organiza-se o ambiente de testes (infraestrutura, equipamentos, hardware, software, pessoal capacitado, ferramentas e massa de testes adequadas) para que os testes sejam executados conforme planejados.
ESPECIFICAÇÃO	nesta etapa, temos as atividades de elaborar e revisar casos de testes e roteiros de testes (scripts). Esse último descreve a relação dos casos de testes e a previsão de execução dos testes.
EXECUÇÃO	Nesta etapa, testes são executados conforme roteiros estabelecidos para os testes. Executa-se sempre que ocorrem mudanças na aplicação e analisam-se os testes executados com sucesso e com defeito – os resultados obtidos são registrados.
ENTREGA	Nesta etapa, o projeto é finalizado, registra-se toda a documentação e relatam-se todas as incidências relevantes à melhoria do processo em um relatório de conformidades e não-conformidades – por fim, a documentação gerada é arquivada.

PLANO DE TESTE	Trata-se de um artefato que apresenta o planejamento para execução das atividades de testes, apresentando seu escopo, métodos empregados, prazo estimado, nível de qualidade esperado, expectativa de capacidade, recursos que serão utilizados como ferramenta de apoio, e métricas e formas de acompanhamento do processo.
CASOS DE TESTE	Trata-se de um artefato que contém um conjunto de condições e entradas utilizadas para testar um software. Em geral, possui os seguintes atributos: identificador, itens constantes no teste, especificação de entrada, especificação de saída, definição do ambiente necessário, necessidades especiais e dependências de outros casos de testes.





ESTRATÉGIAS DE TESTES	DESCRIÇÃO
TESTE DE UNIDADE	Também chamado de Teste de Componente/Módulo, focaliza o esforço de verificação na menor unidade de projeto do software: o componente ou módulo.
TESTE DE INTEGRAÇÃO	Executado para assegurar que os componentes no modelo de implementação operem adequadamente quando combinado para executar um caso de uso.
TESTE DE VALIDAÇÃO	Também chamado de Teste de Aceitação, é executado quando o software está todo montado como um pacote e os erros de interface já foram descobertos e corrigidos.

TESTE DE SISTEMA	O Teste de Sistema é na realidade uma série de diferentes testes cuja finalidade primária é exercitar o sistema como um todo.

TÉCNICAS DE TESTES	DESCRIÇÃO
TESTE CAIXA-BRANCA	Também conhecido como teste estrutural, procedimental, orientado à lógica, caixa-de-vidro ou caixa-clara, ele analisa caminhos lógicos possíveis de serem executados, portanto é necessário ter conhecimento sobre o funcionamento interno dos componentes. Ela busca garantir que todos os caminhos independentes de um módulo sejam executados pelo menos uma vez.
TESTE CAIXA-PRETA	Também conhecido como teste comportamental, funcional, orientado a dado, orientada à entrada/saída ou caixa-escura, baseia-se em pré-condições e pós-condições, geralmente sendo utilizada nas etapas posteriores da disciplina de testes. Ela busca funções incorretas ou inexistentes, erros de comportamento ou desempenho, erros de inicialização e interface, entre outros.
TESTE CAIXA-CINZA	Trata-se de uma versão híbrida entre os testes caixa-branca e os testes caixa-preta. Nesse sentido, essa técnica analisa a parte lógica mais a funcionalidade do sistema, fazendo uma comparação do que foi especificado com o que está sendo realizado. Usando esse método, o testador comunica-se com o desenvolvedor para entender melhor o sistema e otimizar os casos de teste que serão realizados.

TIPOS DE TESTES	DESCRIÇÃO
TESTE DE DESEMPENHO	Trata-se de um tipo de teste projetado para testar o desempenho em tempo de execução do software dentro do contexto de um sistema integrado. Ele é feito em todas as etapas no processo de teste e deve ser realizado, se possível, o quanto antes.
TESTE DE ESTRESSE	Trata-se de um tipo de teste de desempenho deliberadamente intenso, utilizado para determinar a estabilidade de um determinado sistema para além da capacidade operacional normal, às vezes até um ponto de ruptura, a fim de observar os resultados.
TESTE DE CARGA	Trata-se de um tipo de teste de desempenho que procura determinar como o software responderá a várias condições de carga (Ex: número de usuários, número de transações por usuário, etc) de acordo com os limites de operação do sistema.
TESTE DE USABILIDADE	Trata-se de um tipo de teste que avalia o grau com o qual os usuários podem interagir efetivamente com o software e o grau em que o software dirige as ações do usuário, proporciona uma boa realimentação e reforça uma interação consistente.
TESTE DE REGRESSÃO	Trata-se de um tipo de teste que reexecuta o mesmo subconjunto de testes que já foram executados para assegurar que as alterações não tenham propagado efeitos colaterais indesejados.
TESTE DE FUMAÇA	Trata-se de um tipo de teste utilizado para expor erros que impedem a execução corretamente de uma função básica em um software.
TESTE DE COMPARAÇÃO	Trata-se de um tipo de teste que executados em versões diferentes do software e cujos resultados são comparados – não só o conteúdo, mas também o tempo de resposta. São também conhecidos como Testes Back-to-Back.
TESTE ALFA	Trata-se de um tipo de teste de aceitação executados em ambiente controlado que disponibiliza o sistema para um pequeno grupo de usuários, que dão feedbacks sobre o estado atual do sistema.
TESTE BETA	Trata-se de um tipo de teste de aceitação executados em ambiente não controlado quando o desenvolvimento está praticamente concluído e quando o maior número possível de defeitos precisa ser encontrado antes do lançamento do produto.

TESTE DE RECUPERAÇÃO	Trata-se de um tipo de teste que força o software a falhar de várias formas e verifica se a recuperação é executada corretamente.
TESTE DE COMPATIBILIDADE	Trata-se de um tipo de software que valida a capacidade de um sistema de ser executado em um ambiente particular de hardware, software, sistema operacional, rede, entre outros – podendo ser automático ou manual.
TESTE DE SEGURANÇA	Trata-se de um tipo de software que tenta verificar se os mecanismos de proteção incorporados ao sistema vão de fato protegê-lo contra acesso indevido – é também conhecido como teste de ameaça ou teste de invasão.

QUESTÕES COMENTADAS – CESPE

1. **(CESPE / CNPq – 2024)** Os testes de carga e os testes de esforço são testes de desempenho que exigem instrumentação de hardware e software, uma vez que frequentemente é necessário medir a utilização dos recursos de forma precisa.

Comentários:

Os testes de carga e os testes de esforço são tipos de testes de desempenho que requerem instrumentação de hardware e software para medir com precisão a utilização de recursos como CPU, memória, rede e disco. Esses testes ajudam a identificar gargalos e a capacidade máxima do sistema sob diferentes condições de carga.

Gabarito: Correto

---

2. **(CESPE / CAU-BR – 2024)** Considerada uma técnica sistemática para construir a arquitetura de software concomitantemente à realização de testes para descobrir erros associados às interfaces, o teste de integração realiza testes a partir de componentes testados em unidade.

Comentários:

O teste de integração é uma técnica sistemática usada para verificar a interação entre diferentes componentes ou unidades de software. Após a realização de testes de unidade em cada componente individualmente, o teste de integração combina esses componentes e testa suas interfaces para identificar erros na comunicação e interação entre eles.

Gabarito: Correto

---

3. **(CESPE / CAU-BR – 2024)** Considerando que o teste funcional objetiva determinar se um recurso funciona corretamente sem problemas, é possível automatizar esse tipo de teste mesmo que o sistema seja web, em que é possível simular os retornos esperados.

Comentários:

Testes funcionais podem ser automatizados, inclusive em sistemas web. Ferramentas como Selenium, Cypress e QTP permitem simular interações do usuário e validar se o sistema funciona corretamente conforme especificado, verificando os retornos esperados de funcionalidades específicas.

Gabarito: Correto

---

4. **(CESPE / CAU-BR – 2024)** Os testes de regressão são realizados por ocasião da ocorrência de mudanças no software.

Comentários:

Os testes de regressão são realizados quando ocorrem mudanças no software para garantir que as alterações feitas não introduzam novos defeitos em partes previamente funcionais do sistema. Esses testes reexecutam os casos de teste existentes para verificar se o software continua a operar corretamente após as modificações, como correções de bugs, adição de novas funcionalidades ou melhorias.

Gabarito: Correto

---

5. **(CESPE / TST – 2024)** Assinale a opção em que é apresentada uma técnica de desenvolvimento de software orientada a testes que é voltada para o atendimento dos requisitos do sistema com base no negócio, que utiliza exemplos e dublês de teste e que descreve funcionalidades por meio da sintaxe dado que, quando e então:

a) teste unitário de software
b) desenvolvimento orientado por comportamento (BDD)
c) desenvolvimento guiado por testes (TDD)
d) desenvolvimento guiado por testes de aceitação (ATDD)
e) testes de aceitação de usuário (UAT)

Comentários:

(a) Errado. Teste unitário de software é uma técnica que testa unidades individuais de código, como funções ou métodos, de forma isolada;

(b) Correto. Desenvolvimento orientado por comportamento (BDD) é uma técnica que se concentra no comportamento do sistema de acordo com os requisitos de negócios, utilizando exemplos e dublês de teste, e descrevendo funcionalidades com a sintaxe dado que, quando, e então;

(c) Errado. Desenvolvimento guiado por testes (TDD) é uma técnica que envolve escrever testes antes de implementar o código, mas não necessariamente usa a sintaxe dado que, quando, e então;

(d) Errado. Desenvolvimento guiado por testes de aceitação (ATDD) é uma técnica onde os testes de aceitação são escritos antes do desenvolvimento, mas não foca especificamente na sintaxe dado que, quando, e então;

(e) Errado. Testes de aceitação de usuário (UAT) são realizados pelos usuários finais para validar se o sistema atende aos requisitos de negócios, mas não utilizam a técnica e sintaxe específica do BDD.

Gabarito: Letra B

---

6. **(CESPE / TST – 2024)** O projeto de software é a primeira atividade técnica voltada à construção de software e deve ser iniciada logo após:

a) o teste de software.
b) o projeto de dados.
c) o projeto arquitetural.
d) a análise e a especificação de requisitos.
e) a codificação.

Comentários:

(a) Errado. O teste de software ocorre após a codificação e não precede o projeto de software; (b) Errado. O projeto de dados é uma etapa que pode estar incluída no projeto de software, mas não precede a análise e especificação de requisitos; (c) Errado. O projeto arquitetural é uma parte do projeto de software, não uma atividade que ocorre antes dele; (d) Correto. A análise e a especificação de requisitos são etapas que identificam e documentam o que o sistema deve fazer e servem de base para o projeto de software; (e) Errado. A codificação é a fase de implementação que ocorre após o projeto de software.

Gabarito: Letra D

---

7. **(CESPE / MPE-GO – 2024)** Caso seja necessário verificar se o software desenvolvido está funcionando conforme o esperado e garantir que suas principais funções não apresentem grandes falhas, na execução rápida de seus principais recursos, indica-se a realização do teste fumaça.

Comentários:

O teste de fumaça é uma abordagem rápida e inicial para verificar se o software desenvolvido está funcionando conforme o esperado nas suas principais funcionalidades. Esse tipo de teste visa garantir que as funções mais críticas não apresentem falhas significativas e que o sistema está estável o suficiente para que testes mais detalhados sejam realizados posteriormente. Ele serve como uma verificação preliminar para identificar problemas óbvios que podem impedir a execução completa de testes mais exaustivos.

Gabarito: Correto

---

8. **(CESPE / LNA – 2024)** Assinale a opção em que é corretamente apresentado o tipo de teste de software responsável por verificar se diferentes partes do sistema de software foram projetadas para interagir entre si e se fazem essa interação corretamente, avaliando, inclusive, como os dados são transferidos entre elas.

a) teste de desempenho
b) teste unitário

c) teste funcional
d) teste de integração
e) teste de aceitação

Comentários:

(a) Errado. Teste de desempenho verifica a performance do sistema, como tempo de resposta e utilização de recursos, não a interação entre partes do sistema;

(b) Errado. Teste unitário verifica partes individuais do código (funções ou métodos) isoladamente, sem testar a interação entre diferentes componentes do sistema;

(c) Errado. Teste funcional verifica se o sistema atende aos requisitos funcionais especificados, focando em saídas corretas para entradas dadas, não especificamente na interação entre componentes;

(d) Correto. Teste de integração verifica se diferentes partes do sistema interagem corretamente entre si, incluindo a transferência de dados entre componentes;

(e) Errado. Teste de aceitação verifica se o sistema atende aos requisitos e expectativas do usuário final, sem foco específico na interação entre componentes internos do sistema.

Gabarito: Letra D

---

9. **(CESPE / LNA – 2024)** É uma característica-chave de um bom framework de automação de teste:

a) a falta de suporte a linguagens de programação populares.
b) a escalabilidade limitada.
c) a alta complexidade.
d) a baixa manutenibilidade.
e) a flexibilidade mínima.

Comentários:

(a) Errado. Um bom framework de automação de teste deve ter suporte a linguagens de programação populares para ser amplamente adotado e integrado com diferentes sistemas e equipes;

(b) Errado. A escalabilidade limitada é uma característica negativa, pois um bom framework deve ser capaz de crescer conforme as necessidades do projeto aumentam;

(c) Errado. A alta complexidade dificulta a implementação, manutenção e uso do framework. Um bom framework deve ser fácil de usar e configurar;

(d) Correto. Um bom framework deve ser fácil de manter, logo deve exigir baixa manutenibilidade para atualizar e adaptar às mudanças;

(e) Errado. A flexibilidade mínima é uma desvantagem, pois impede o framework de ser adaptado a diferentes necessidades e cenários de teste. Um bom framework deve ser flexível para suportar vários tipos de testes e configurações;

Gabarito: Letra D

---

10. (CESPE / LNA – 2024) Os frameworks de teste de software:

a) permitem o aumento da produtividade, apesar de não serem úteis para testar softwares em diferentes plataformas.

b) permitem o aumento da produtividade, ainda que não possam ser integrados a processos de integração contínua.

c) são usados exclusivamente para testes manuais.

d) são ferramentas para criar bugs no software.

e) fornecem estruturas e funcionalidades para automatizar os testes de software.

Comentários:

(a) Errado. Os frameworks de teste são úteis para testar softwares em diferentes plataformas, o que contradiz a afirmação; (b) Errado. Muitos frameworks de teste são integráveis a processos de integração contínua, aumentando a eficiência; (c) Errado. Frameworks de teste são usados principalmente para testes automatizados, não apenas para testes manuais; (d) Errado. O objetivo dos frameworks de teste é identificar bugs, não os criar; (e) Correto. Os frameworks de teste fornecem estruturas e funcionalidades para automatizar os testes de software, o que melhora a eficiência e a eficácia do processo de teste.

Gabarito: Letra E

---

11. (CESPE / LNA – 2024) A abordagem que se concentra principalmente em examinar as estruturas internas ou os funcionamentos de uma aplicação de software é denominada teste de:

a) sistema.
b) caixa preta.

c) caixa branca.
d) aceitação.
e) caixa cinza.

Comentários:

(a) Errado. O teste de sistema verifica o sistema como um todo, focando em suas funcionalidades e comportamentos em relação aos requisitos, não nas estruturas internas;

(b) Errado. O teste de caixa preta avalia a funcionalidade externa da aplicação, sem considerar suas estruturas internas ou o código;

(c) Correto. O teste de caixa branca foca nas estruturas internas da aplicação, verificando o funcionamento do código e a lógica interna;

(d) Errado. O teste de aceitação é realizado para validar o software em relação aos requisitos de negócios e necessidades do usuário, sem examinar as estruturas internas;

(e) Errado. O teste de caixa cinza combina aspectos dos testes de caixa branca e preta, mas não se concentra exclusivamente nas estruturas internas.

Gabarito: Letra C

---

12. (CESPE / MPO – 2024) Realiza-se o teste de estresse para confrontar os programas com situações anormais, de forma a exigir recursos em maior quantidade, frequência ou volume.

Comentários:

O teste de estresse é uma técnica que avalia a estabilidade e a confiabilidade de um software submetendo-o a condições extremas e situações anormais. O objetivo é verificar como o sistema se comporta ao lidar com carga além de sua capacidade regular, demandando mais recursos, como CPU, memória e rede, para identificar pontos de falha e melhorar a resiliência do sistema.

Gabarito: Correto

---

13. (CESPE / MPO – 2024) O teste automatizado pode conter recursos de auditoria eletrônica com avaliadores e geradores automáticos de testes.

Comentários:

O teste automatizado frequentemente inclui ferramentas que permitem a criação, execução e avaliação de testes de forma automática. Esses testes podem incorporar recursos de auditoria eletrônica, como registradores de execução, geradores automáticos de casos de teste e avaliadores

que verificam a conformidade do sistema com os requisitos especificados, facilitando a identificação de erros e o monitoramento contínuo da qualidade do software.

Gabarito: Correto

---

14. (CESPE / INPI – 2024) Um teste de software de regressão estará corretamente projetado quando se considera, em cada uma das funções principais do software, apenas os testes que tratam de uma ou mais classes de erros.

Comentários:

De acordo com Roger Pressman, o conjunto de testes de regressão deve ser projetado de forma a incluir somente aqueles testes que tratam de uma ou mais classes de erros em cada uma das funções principais do programa. Lembrando que existem três tipos diferentes de casos de testes:

- Uma amostra representativa dos testes que usam todas as funções do software.
- Testes adicionais que focalizam as funções de software que podem ser afetadas pela alteração.
- Testes que focalizam os componentes do software que foram alterados.

Gabarito: Correto

---

15. (CESPE / CAU-BR – 2024) Os estágios das atividades de teste de software devem ser realizados na seguinte ordem: teste do sistema, teste de integração e teste de unidade.

Comentários:

A ordem correta das atividades de teste de software é geralmente: teste de unidade, teste de integração e teste do sistema.

- Teste de Unidade: primeiro, cada unidade ou componente individual do software é testado isoladamente para garantir que funcionam corretamente;
- Teste de Integração: em seguida, as unidades são combinadas e testadas em conjunto para verificar se interagem corretamente;
- Teste do Sistema: finalmente, o sistema completo é testado como um todo para assegurar que todos os componentes funcionam juntos como esperado e atendem aos requisitos especificados.

Gabarito: Errado

---

16. (CESPE / SEFIN de Fortaleza-CE – 2023) Acerca dos testes de software e das ferramentas para automatização de testes, bem como do desenvolvimento orientado por comportamento, julgue o item que se segue.

Na análise do valor limite, casos de teste podem ser derivados dos domínios de entrada e de saída.

Comentários:

Na análise do valor limite (Boundary Value Analysis), casos de teste são criados com base nos limites das entradas e saídas, onde os erros são mais propensos a ocorrer. Essa técnica foca nos valores que estão exatamente nos limites inferior e superior, bem como nos valores imediatamente acima e abaixo desses limites, tanto para os domínios de entrada quanto para os de saída. Isso ajuda a identificar falhas em situações críticas, garantindo que o sistema lide corretamente com os valores extremos.

Gabarito: Correto

---

17. **(CESPE / DATAPREV – 2023)** Apesar de primar pela agilidade, testes ágeis exigem processos bem definidos, sob pena de perda de qualidade do produto final.

Comentários:

O que seriam processos ágeis "bem definidos"? Não dá para fazer uma avaliação objetiva dessa forma e, por isso, a questão foi anulada.

Gabarito: Anulada

---

18. **(CESPE / SERPRO – 2023)** Os mocks são métodos utilizados para realizar testes unitários quando é impossível testar o objeto real, seja porque ele não está disponível, seja porque não é possível executá-lo durante o teste.

Comentários:

Mocks são objetos simulados que substituem objetos reais em testes unitários. Eles são usados quando testar o objeto real não é possível ou viável, por exemplo, devido a dependências externas (como bases de dados ou serviços de terceiros) ou porque o objeto real ainda não está disponível. O lance é que a questão falou apenas "impossível" e isso dificulta a avaliação. A questão foi anulada com a seguinte justificativa: *"O item possibilita mais de uma interpretação, fato que prejudicou seu julgamento objetivo".*

Gabarito: Anulado

---

19. **(CESPE / SEFIN de Fortaleza-CE – 2023)** Em um teste de integração, cada uma das unidades é testada separadamente para se observar se elas funcionam de forma adequada.

Comentários:

No teste de integração, o foco não é testar as unidades individualmente, mas sim verificar a interação entre diferentes unidades ou módulos para garantir que funcionem corretamente quando

combinados. O objetivo é identificar problemas na interface e na comunicação entre os componentes. O teste de unidades individualmente é chamado de teste unitário.

Gabarito: Errado

---

20. (CESPE / SEFIN de Fortaleza-CE – 2023) Em um teste funcional de software, os elementos de uma classe devem se comportar de maneira equivalente.

Comentários:

Em um teste funcional, a técnica de partição de equivalência é utilizada para dividir os dados de entrada em classes de equivalência, onde todos os elementos dentro de uma classe devem se comportar de maneira semelhante ou equivalente. O objetivo é reduzir o número de casos de teste, garantindo que, ao testar um elemento de cada classe, o comportamento seja representativo de todos os elementos dessa classe. Se um elemento de uma classe passa no teste, espera-se que todos os outros elementos dessa mesma classe também passem.

Gabarito: Correto

---

21. (CESPE / SEFIN de Fortaleza-CE – 2023) No particionamento de equivalências para a criação de casos de teste, devem ser consideradas apenas as partições válidas.

Comentários:

No particionamento de equivalência, tanto as partições válidas quanto as partições inválidas devem ser consideradas. As partições válidas representam entradas que estão dentro dos limites normais de operação, enquanto as partições inválidas representam entradas fora desses limites. Testar ambas é fundamental para garantir que o sistema funcione corretamente com entradas válidas e lide adequadamente com entradas inválidas, como erros ou exceções.

Gabarito: Errado

---

22. (CESPE / EMPREL – 2023) A escolha de casos de teste de unidade é fundamental para a redução dos custos dos testes e uma estratégia que pode ser adotada para a seleção de casos é a de teste de partição de equivalência. Nesse contexto, considere-se um programa que precisa aceitar as seguintes opções de tamanho de folha para impressão.

- 10,5 cm × 14,8 cm
- 14,8 cm × 21,0 cm
- 21,0 cm × 29,7 cm
- 29,7 cm × 42,0 cm

Nessa hipótese, para a aplicação correta da técnica de equivalência de partição com o número mínimo de casos de teste, é necessário realizar:

a) apenas um caso de teste, para verificar se o programa pode imprimir folhas em todos os tamanhos exigidos.

b) três casos de teste, para verificar se o programa pode imprimir folhas de 15 cm, 22 cm ou 30 cm.

c) quatro casos de teste, para verificar se o programa pode imprimir folhas em todos os tamanhos exigidos.

d) seis casos de teste, para verificar se o programa pode imprimir folhas em todos os tamanhos exigidos e para tamanhos menores que 10,5 cm e maiores de 29,7 cm.

e) cinco casos de teste, para verificar se o programa pode imprimir folhas em todos os tamanhos exigidos e para tamanhos maiores que 29,7 cm.

Comentários:

Para aplicar a técnica de partição de equivalência na escolha de casos de teste de unidade, precisamos dividir o conjunto de entradas possíveis em classes de equivalência. Essas classes agrupam entradas que devem ser tratadas de maneira semelhante pelo software, permitindo que uma entrada de cada classe seja suficiente para testar o comportamento esperado do sistema.

Como temos uma partição para folhas medindo 10,5 cm × 14,8 cm, precisaremos criar um conjunto de testes específico para essa partição. Esse procedimento será repetido para cada partição, resultando em quatro conjuntos de testes, pois há quatro conjuntos de partições.

Gabarito: Letra C

---

23. (CESPE / MPE-RO – 2023) O teste de integração de software é responsável por:

a) examinar os módulos desenvolvidos sem considerar acoplamento.
b) iniciar a recuperação de um sistema após uma falha.
c) analisar a forma como um sistema se comporta quando seus recursos são muito acessados.
d) validar, de forma conjunta, os módulos construídos individualmente.
e) verificar se os mecanismos de proteção de um software impedem acessos indevidos.

Comentários:

(a) Errado. O teste de integração de software foca precisamente no acoplamento entre módulos, testando como eles interagem entre si. Examinar módulos isoladamente sem considerar acoplamento seria o objetivo de testes unitários.

(b) Errado. Recuperação de um sistema após uma falha é um aspecto de teste de recuperação ou resiliência, não de integração. Ele verifica a capacidade do sistema de retornar a um estado operacional após uma falha.

(c) Errado. Analisar a forma como um sistema se comporta quando seus recursos são muito acessados é característico de testes de desempenho ou carga, que avaliam como o sistema lida com grandes volumes de uso.

(d) Correto. O teste de integração é responsável por validar, de forma conjunta, os módulos construídos individualmente, verificando se eles funcionam corretamente quando combinados. Este teste foca nas interfaces e interações entre módulos ou subsistemas.

(e) Errado. Verificar se os mecanismos de proteção de um software impedem acessos indevidos é tarefa de testes de segurança, que avaliam a eficácia das medidas de proteção contra ameaças e acessos não autorizados.

Gabarito: Letra D

---

24. (CESPE / DATAPREV – 2023) No teste de acompanhamento, um grupo é designado para verificar quaisquer problemas que necessitem ser resolvidos e quaisquer alterações que devam ser feitas no ambiente de recuperação de desastres.

Comentários:

No teste de acompanhamento, um grupo é designado para verificar problemas que precisam ser resolvidos e quaisquer alterações que devem ser feitas no ambiente de recuperação de desastres. Este processo assegura que as estratégias de recuperação estão atualizadas e funcionais, identificando falhas e áreas de melhoria para garantir a continuidade dos negócios em caso de desastre.

Gabarito: Correto

---

25. (CESPE / TBG – 2023) Os testes dos tipos alfa e beta são executados em um ambiente controlado e com a presença de, pelo menos, um desenvolvedor.

Comentários:

Os testes alfa e beta são tipos distintos de testes de aceitação realizados em diferentes ambientes:

1. Teste Alfa: executado em um ambiente controlado, geralmente nas instalações do desenvolvedor, e pode envolver a presença de desenvolvedores. Este teste é realizado antes do lançamento do produto para um público restrito, como a equipe interna ou um grupo seleto de usuários.

2. Teste Beta: realizado em um ambiente real pelos usuários finais, sem a presença direta dos desenvolvedores. Esse teste visa identificar problemas que não foram detectados durante o teste alfa, oferecendo uma visão mais ampla do desempenho do software no mundo real.

Gabarito: Errado

---

26. (CESPE / TBG – 2023) O teste de regressão deve ser efetuado para garantir que novos componentes não tenham causado problema nas funções que antes funcionavam corretamente.

Comentários:

O teste de regressão é essencial para assegurar que alterações recentes no software, como a adição de novos componentes, correções de bugs ou atualizações, não introduziram problemas em funcionalidades que anteriormente funcionavam corretamente. Esse tipo de teste verifica se o comportamento do sistema permanece estável após mudanças, ajudando a identificar e corrigir possíveis regressões no código.

Gabarito: Correto

---

27. (CESPE / TBG – 2023) O teste de segurança estático (SAST) trabalha diretamente com o código e é empregado de forma complementar ao teste de segurança dinâmico (DAST).

Comentários:

O teste de segurança estático (SAST) analisa o código fonte, bytecode ou binário de uma aplicação de forma estática, ou seja, sem executá-la, para identificar vulnerabilidades de segurança. O SAST é empregado de forma complementar ao teste de segurança dinâmico (DAST), que testa a aplicação em execução, simulando ataques para encontrar vulnerabilidades. Juntos, SAST e DAST fornecem uma abordagem abrangente para identificar e mitigar problemas de segurança em diferentes estágios do desenvolvimento de software.

Gabarito: Errado

---

28. (CESPE / BANRISUL – 2022) Na gestão de defeitos, o princípio de teste da regra 10 de Myers estabelece que as atividades de teste estático e dinâmico devem ser planejadas muito antes de serem iniciadas.

Comentários:

Na verdade, a Regra 10 de Myers indica que o custo da correção de um defeito tende a ser cada vez maior quanto mais tarde ele for descoberto. Defeitos encontrados nas fases iniciais da etapa de

desenvolvimento do software são mais baratos de serem corrigidos do que aqueles encontrados na produção.

Gabarito: Errado

---

29. (CESPE / BANRISUL – 2022) No teste de fumaça (smoke test), os códigos do software são integrados em componentes bloqueadores de erros com módulos reutilizáveis necessários para implementar as funções do software.

Comentários:

Vamos lá! Teste de Fumaça é uma abordagem de teste de integração comumente usada quando um produto de software é desenvolvido. Ele é projetado como um mecanismo de acompanhamento para projetos de tempo crítico, permitindo que a equipe de software avalie o projeto com frequência. Ele busca descobrir erros do tipo "show-stopper", que têm a maior probabilidade de atrasar o calendário do projeto.

Dito isso, esse item é uma viagem total! O Teste de Fumaça é um teste projetado para verificar se o software pode ser testado. Sabe aqueles erros tão básicos que impedem até o próprio teste? Pois é! Eu acredito que a banca traduziu mal "show-stopper error" como "componentes bloqueadores de erros". Enfim, nada faz sentido nesse item. Se alguém tiver alguma explicação melhor, podem mandar no nosso fórum :)

Gabarito: Errado

---

30. (CESPE / BANRISUL – 2022) É possível que um defeito que resida em código sem causar uma falha não seja encontrado em um teste dinâmico.

Comentários:

Teste dinâmico é um tipo de teste de software no qual o código sendo testado é executado. Ele é usado para testar a lógica de programação, as interfaces de usuário e as APIs, bem como para verificar a integridade de um código. Testes dinâmicos só verificam se o código está funcionando como esperado. Se um defeito estiver presente no código, mas não estiver causando nenhuma

falha, então ele não será encontrado no teste dinâmico, pois o código estará funcionando como esperado.

Gabarito: Correto

---

31. (CESPE / BANRISUL – 2022) O teste estático é uma técnica de verificação de software que revisa o código do programa para identificar se ele representa as especificações produzidas para o software.

Comentários:

Perfeito! O teste estático é uma ótima maneira de identificar possíveis erros no código antes mesmo de executar o programa. Ele pode ajudar a reduzir o tempo e os custos associados ao processo de desenvolvimento de software, pois os erros podem ser detectados e corrigidos mais rapidamente. Além disso, o teste estático também pode melhorar a qualidade do software, pois permite que os erros sejam identificados e corrigidos antes que o programa seja liberado para o usuário final.

Gabarito: Correto

---

32. (CESPE / BANRISUL – 2022) Os testes de estresse devem verificar o uso da memória ao longo do tempo para garantir que não existam perdas acumulativas.

Comentários:

Os testes de estresse são usados para verificar a capacidade de um sistema de suportar um alto volume de tráfego. Eles não são projetados para verificar o uso da memória ao longo do tempo. Na verdade, a questão trata de Testes de Resistência.

Gabarito: Errado

---

33. (CESPE / BANRISUL – 2022) O objetivo do teste de integração é verificar se os requisitos atendem a especificação e se as funcionalidades do sistema foram implementadas corretamente, sendo todo o sistema testado de modo a simular um ambiente de execução real.

Comentários:

Opa... esse é o objetivo do Teste de Sistema. O Teste de Integração é um tipo de teste de software que verifica se diferentes partes do sistema estão integradas corretamente. É um tipo de teste de sistema específico projetado para verificar se as partes do sistema funcionam corretamente juntas.

Gabarito: Errado

---

34. (CESPE / BANRISUL – 2022) Os testes unitários são realizados com o objetivo de isolar cada parte do sistema para garantir que elas estejam funcionando conforme especificado.

Comentários:

Testes unitários são testes de software que são realizados em partes individuais de um aplicativo ou sistema. Os testes unitários isolam cada parte do sistema para garantir que elas estejam funcionando como esperado, por exemplo, verificando se a entrada de dados está sendo processada corretamente, se as saídas estão de acordo com o esperado e se a lógica interna está funcionando corretamente. Ao realizar testes unitários, também é possível verificar se o código está executando como esperado em todos os níveis, o que reduz o risco de erros de código.

Gabarito: Correto

---

35. (CESPE / BANRISUL – 2022) O teste unitário é o processo de testar os componentes de programa, como métodos ou classes de objeto.

Comentários:

Teste unitário é uma técnica de teste de software que envolve a verificação de cada componente individual de um programa. Os testes são realizados para garantir que cada parte do programa funcione de forma adequada e forneça resultados corretos. Os testes unitários são executados a cada vez que uma alteração é feita no código, para garantir que todas as alterações sejam devidamente consideradas. Os testes podem ser realizados em métodos ou classes de objeto, variáveis, instruções de controle e até em funções de APIs.

Gabarito: Correto

---

36. (CESPE / BANRISUL – 2022) Ao se testarem as classes do objeto, devem-se testar as amostras de operações a ele associadas, não havendo necessidade de simular todos os eventos que causam mudança de estado.

Comentários:

O teste unitário é o processo de testar os componentes de programa, como métodos ou classes de objeto. As funções individuais ou métodos são o tipo mais simples de componente. Seus testes devem ser chamadas para essas rotinas com parâmetros diferentes de entrada. Você pode usar as abordagens para projeto de casos de teste para projetar testes de funções ou métodos. Quando você está testando as classes de objeto, deve projetar os testes para fornecer uma cobertura de **todas** as características do objeto. Isso significa que você deve:

1. Testar todas as operações associadas ao objeto;
2. Definir e verificar o valor de todos os atributos associados ao objeto;

3. Colocar o objeto em todos os estados possíveis, o que significa **simular todos os eventos que causam mudanças de estado**.

Gabarito: Errado

---

37. (CESPE / BANRISUL – 2022) Devem ser escolhidos casos efetivos de teste unitário, o que significa que os casos de teste devem mostrar que, quando usado como esperado, o componente que se está testando faz o que ele é proposto a fazer e, se houver defeitos nos componentes, estes devem ser revelados por casos de teste.

Comentários:

O teste é custoso e demorado, por isso é importante que você escolha casos efetivos de teste unitário. A efetividade, nesse caso, significa duas coisas:
1. Os casos de teste devem mostrar que, quando usado como esperado, o componente que você está testando faz o que ele é proposto a fazer.
2. Se houver defeitos nos componentes, estes devem ser revelados por casos de teste.

Gabarito: Correto

---

38. (CESPE / TRT-AP-PA – 2022) A respeito de testes automatizados, no contexto de DevOps e DevSecOps, assinale a opção correta.

a) Em um teste unitário, os métodos da classe sendo testada e suas dependências podem ter relação com recursos externos.

b) Os bugs são detectados no final do ciclo de desenvolvimento, o que pode aumentar o tempo na criação de novos produtos.

c) Os testes unitários são testes de caixa preta com cada função que compõe o software.

d) O TDD (Test Driven Development) eleva o nível dos testes unitários e tem como característica criar a classe de testes antes da classe de produção, de forma que os testes guiem o código a ser implementado.

e) Os testes de integração são caracterizados pela verificação de partes internas do sistema, que se inter-relacionam entre si, conforme definido pelos clientes.

Comentários:

(a) Errado. Na verdade, idealmente ele não pode ter relação com recursos externos para garantir que o teste seja o mais autônomo possível e não seja influenciado por mudanças nos recursos externos. Quando há dependências externas, o teste se torna mais complexo e pode falhar de forma inconsistente ao longo do tempo; (b) Errado. Idealmente, bugs devem ser detectados o quanto

antes para evitar retrabalho; (c) Errado, testes unitários não são testes caixa-preta - eles visam testar a funcionalidade de uma unidade com base em sua estrutura; (d) Correto. TDD é uma excelente abordagem para o desenvolvimento de software, pois permite que o código seja criado de acordo com a especificação desejada. Ele também ajuda a melhorar a qualidade do código, pois os testes unitários são executados antes da produção e validados; (e) Errado. Testes de integração não são caracterizados pela verificação de partes internas do sistema, mas sim pela verificação de como partes externas funcionam em conjunto. A integração testa a comunicação entre sistemas, o fluxo de dados entre sistemas e a interoperabilidade entre sistemas.

Gabarito: Letra D

---

39. (CESPE / BNB – 2022) Na seleção de casos para os testes de unidade, uma estratégia eficaz é a do teste baseado em diretriz, em que os casos são escolhidos com base nas indicações geradas a partir de erros mais comuns identificados no desenvolvimento dos programas.

Comentários:

Existem duas estratégias que podem ser eficazes para ajudar você a escolher casos de teste. São elas:

1. Teste de Partição, em que você identifica os grupos de entradas que possuem características comuns e devem ser tratados da mesma maneira. Você deve escolher os testes dentro de cada um desses grupos.

2. Testes baseados em Diretrizes, em que você usa as diretrizes de testes para escolher casos de teste. Essas diretrizes refletem a experiência anterior dos tipos de erros que os programadores cometem frequentemente no desenvolvimento de componentes.

Gabarito: Correto

---

40. (CESPE / BNB – 2022) O teste com base em casos de uso é um procedimento efetivo para se alcançar o resultado pretendido com um teste de integração do sistema.

Comentários:

O teste com base em casos de uso é um dos métodos mais comuns usados para testar a integração entre os componentes de um sistema. Ele é eficaz na verificação de se um sistema atende aos requisitos e funciona como deveria. Os casos de uso podem ser usados para descrever a função esperada de um sistema, e esta descrição pode ser usada como um guia para realizar o teste. Os testadores podem usar os casos de uso para validar o sistema, certificando-se de que todas as funcionalidades sejam executadas de acordo com o esperado e que não haja erros.

Gabarito: Correto

---

41. (CESPE / BNB – 2022) O teste automatizado usualmente é mais apropriado que o teste manual quando a interface do usuário do aplicativo muda consideravelmente em prazos curtos e a automação de teste ainda não está disponível.

Comentários:

Na verdade, ele é mais apropriado quando a interface de usuário não muda consideravelmente em prazos curtos, sendo melhor empregado em processos repetitivos. Em geral, não compensa o esforço de se criar um teste automatizado para algo que mudará com frequência.

Gabarito: Errado

---

42. (CESPE / TJ-AM – 2019) Erro e defeito são conceitos distintos: erro pode ser o resultado de uma falha; defeito é uma imperfeição ou inconsistência no produto do software ou em seu processo.

Comentários:

Exato! O erro pode ser um resultado de um defeito ou uma falha, como um retorno esperado, que por causa de uma falha teve um valor diferente do que esperado. Já o defeito é qualquer imperfeição ou inconsistência no produto do software ou em seu processo, um defeito é também uma não conformidade. O Defeito faz parte do produto, é algo que esta implementada no código de maneira errada.

Gabarito: Correto

---

43. (CESPE / TJ-AM – 2019) Validação refere-se a um conjunto de atividades destinadas a garantir que o sistema esteja de acordo com os requisitos do usuário.

Comentários:

Perfeito! Em um teste de validação os requisitos estabelecidos como parte dos requisitos de modelagem são validados em relação ao software criado. Em suma, a validação tem sucesso quando o software funciona de uma maneira que pode ser razoavelmente esperada pelo cliente

Gabarito: Correto

---

44. (CESPE / TJ-AM – 2019) O teste caixa preta trata o software como uma caixa cujo conteúdo é desconhecido e da qual só é possível visualizar os dados de entrada fornecidos e as respostas produzidas como saída.

Comentários:

Perfeita definição! O teste de caixa-preta não tem acesso às estruturas internas do software, ele se foca nas saídas geradas em resposta a entradas escolhidas e condições especificadas.

Gabarito: Correto

---

45. (CESPE / TJ-AM – 2019) O teste de esforço é uma continuidade do teste de carga, e ambos são modalidades do teste de desempenho.

Comentários:

O teste de desempenho (ou performance) é projetado para testar o desempenho em tempo de execução do software dentro do contexto de um sistema integrado. O teste de esforço e o teste de carga são modalidades do teste de desempenho, sendo que o teste de esforço (estresse) – normalmente – é realizado após o teste de carga.

Gabarito: Correto

---

46. (CESPE / TJ-AM – 2019) O teste de integração descendente da modalidade primeiro em largura (breadth-first) move-se pela hierarquia de controle e integra todos os componentes em um caminho selecionado como principal.

Comentários:

Na verdade, trata-se do teste de integração descendente da modalidade primeiro em profundidade. De acordo com Pressman, a integração primeiro-em-profundidade integra todos os componentes em um caminho de controle principal da estrutura do programa. A seleção de um caminho principal é, de certa forma, arbitrária e depende das características específicas da aplicação.

Gabarito: Errado

---

47. (CESPE / MC-PA – 2019) Consoante os princípios dos métodos ágeis, na construção de um sistema, os testes de unidade do código criado devem ser sempre executados primeiramente:

a) pela equipe de testes, somente.
b) pela equipe de testes e pelo gerente do projeto.
c) pelo gerente do projeto, somente.
d) pelo programador.
e) pelo programador com o apoio da equipe de testes.

Comentários:

Os testes de unidade geralmente são feitos pelos próprios desenvolvedores de maneira mais informal e, não, por especialistas em testes.

Gabarito: Letra D

48.(CESPE / FUB – 2018) Os testes de caixa-branca buscam verificar o comportamento interno do software, ou seja, os elementos relacionados ao código-fonte desse software.

Comentários:

Perfeito! A ideia da técnica de caixa-branca é testar a estrutura interna do software, isto é, seu código-fonte em si.

Gabarito: Correto

49.(CESPE / BNB – 2018) Determinada equipe de desenvolvimento de softwares desejava realizar testes que avaliassem o comportamento do sistema por meio do estudo das entradas e das saídas relacionadas, sem validação da implementação do software e sem acesso ao seu código-fonte. Para isso, a equipe sugeriu a utilização dos testes de caixa-preta e de caixa-branca. Nessa situação, somente o teste tipo caixa-preta é corretamente aplicável, pois o tipo caixa-branca depende de acesso ao código-fonte do sistema.

Comentários:

Perfeito! O teste em que não há acesso às estruturas internas do software é o teste de caixa-preta, dessa forma, o teste de caixa-branca não é aplicável pois ele tem acesso ao código-fonte.

Gabarito: Correto

---

50. (CESPE / BNB – 2018) Determinada equipe de desenvolvimento de softwares pretendia realizar testes que permitissem avaliar cenários com os quais os usuários reais do sistema pudessem se relacionar. Esses cenários deveriam descrever uma maneira de usar o sistema. Para isso, foram sugeridos os testes de release e de cenário. Nessa situação, será correto aplicar testes de cenários, que são incompatíveis com os de release, devendo a aplicação desses últimos ser descartada.

Comentários:

Teste de cenário é uma abordagem de teste de release em que você imagina cenários típicos de uso e os usa para para desenvolver casos de teste para o sistema. Dessa forma, não há que se falar que os testes de cenário são incompatíveis com os testes de release.

Gabarito: Errado

---

51. (CESPE / BNB – 2018) O planejamento de testes é governado pela necessidade de selecionar alguns poucos casos de teste de um grande conjunto de possíveis casos. O exame que avalia se um grupo de entrada de dados resultou nas saídas pretendidas, levando-se em consideração a especificação do programa, é denominado teste:

a) de stress.
b) da caixa preta.
c) da caixa branca.
d) da caixa cinza.
e) de integração.

Comentários:

O teste que não tem acesso às estruturas internas do software e focaliza-se nas entradas e saídas é o teste caixa-preta.

Gabarito: Letra B

---

52. (CESPE / STJ – 2018) Em um serviço de integração contínua, testes de unidade são executados automaticamente com a finalidade de detectar erros funcionais.

Comentários:

Basicamente a questão quer saber se os testes de unidade podem ser utilizados de forma automatizada, e sim eles podem.

Gabarito: Correto

---

53. (CESPE / STJ – 2018) Enquanto os testes de unidade propiciam a qualidade externa, os testes de aceitação ajudam o desenvolvedor a avaliar a qualidade interna do código, dando feedback sobre o design dos módulos e permitindo a manutenção com menor custo.

Comentários:

A questão inverteu os conceitos. Na verdade, os testes que propiciam qualidade externa são os testes de aceitação (validação), ou seja, aqueles que são testados por usuários finais. Já os testes de unidade avaliam a qualidade dos módulos ou componentes.

Gabarito: Errado

---

54. (CESPE / STJ – 2018) Teste de software pode ser definido como o processo de execução de um programa ou sistema com a intenção de se verificar se o mesmo está de acordo com o planejado nas especificações dos seus requisitos.

Comentários:

Perfeito! O teste de software é um processo voltado a atingir a confiabilidade do software. De fato, temos que o teste de software é um processo mais amplo que o teste de validação. No entanto, essa definição está mais associada ao teste de validação que focalizam ações e saídas, ou seja, se o software funciona conforme foi proposto em seus requisitos.

Gabarito: Correto

---

55. (CESPE / ABIN – 2018) As ferramentas de execução do teste são classificadas como ferramentas de suporte para execução e registro e têm, como vantagem, o fato de não requererem um grande esforço para a obtenção de resultados expressivos.

Comentários:

Não é porque o teste é automatizado que ele não exigirá grande esforço. Por exemplo, alguns testes automatizados podem levar um grande tempo para serem concluídos.

Gabarito: Errado

---

56. (CESPE / ABIN – 2018) No teste funcional, que é uma das fases do processo de validação, não é necessário o conhecimento das estruturas internas do software.

Comentários:

Exatamente isso! No teste funcional, também chamado de teste de caixa-preta, não há acesso às estruturas internas do software. Em suma, ele serve para analisar como o software deveria funcionar, ou seja, se o software está de acordo com os requisitos propostos para o mesmo.

Gabarito: Correto

---

57. (CESPE / ABIN – 2018) No teste de integração, o foco é a comunicação entre os módulos do software, não as suas funcionalidades; portanto, nessa fase, testes funcionais não podem ser utilizados.

Comentários:

Testes de Integração verificam 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. Dessa forma, não há nenhuma restrição em se utilizar os testes de caixa-preta (também chamados de testes funcionais) nessa etapa.

Gabarito: Errado

---

58. (CESPE / STM – 2018) Em um processo de cascata, testes de sistemas testam todo o sistema, enquanto, em processos de desenvolvimento iterativo, será testado apenas um incremento a ser entregue ao cliente.

Comentários:

De acordo com Sommerville: "Em um processo de desenvolvimento iterativo, o teste de sistema concentra-se no teste de um incremento que será entregue ao cliente; em um processo em cascata, o teste de sistema concentra-se no teste de todo o sistema".

Gabarito: Correto

---

59. (CESPE / CGM DE JOÃO PESSOA-PB – 2018) O particionamento de equivalência é uma técnica de teste caixa-preta caracterizada por dividir o domínio de entrada de um módulo em classes de equivalência, a partir das quais casos de teste são derivados.

Comentários:

A técnica de particionamento de equivalência é uma técnica de caixa-preta, ela permite otimizar o processo de testes por meio da separação das entradas em classes ou categorias diferentes.

Gabarito: Correto

---

60. (CESPE / STM – 2018) Testes de regressão servem ao propósito de verificar se o sistema pode operar na carga necessária, fazendo-a regredir constantemente até que o comportamento de falha do sistema seja testado ou que defeitos sejam identificados.

Comentários:

Carga? Nope! Testes de Regressão são executados para garantir que uma funcionalidade ou parte de um software já testado continue funcionando após o mesmo sofrer alguma implementação ou manutenção.

Gabarito: Errado

---

61. (CESPE / STM – 2018) Em testes de integração, a estratégia de integração bottom-up integrará componentes de infraestrutura que fornecem serviços comuns, adicionando a eles

componentes funcionais; para testar uma nova característica, pode ser necessário integrar componentes diferentes.

Comentários:

O que é Bottom-Up? Eu construo, integro e testo componentes de infraestrutura que fornecem serviços comuns a vários outros componentes; são mais genéricos, ou seja, é aquela classe mais independente que presta serviço para outras e depois componentes funcionais em que eu testo componentes que realizam funcionalidades específicas, que são mais dependentes, que precisam de outras classes – então é do mais geral para o mais específico (no sentido de funcionalidades) e do menor para o maior (no sentido de tamanho). A abordagem top-down é o contrário, eu começo testando o sistema até chegar aos módulos, ou seja, do mais específico para o mais geral (no sentido de funcionalidade). Portanto, a questão está perfeita!

Gabarito: Correto

---

62. (CESPE / BNB – 2018) 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.

Comentários:

Pelo contrário, esse é o principal teste responsável por verificar novos bugs inseridos no sistema.

Gabarito: Errado

---

63. (CESPE / TRE-BA – 2017) Durante o planejamento de um projeto e a elaboração dos casos de uso, foram incluídos diversos componentes para cálculos de tributos e da quantidade de recursos orçamentários alocados. De acordo com os níveis de testes existentes, o resultado de um dos componentes em questão poderá ser validado por meio do teste:

a) unitário.
b) de sistema.
c) de aceitação.
d) de codificação.
e) de integração.

Comentários:

Questão que você não pode errar, porque todo mundo acertará! A questão fala em diversos componentes para cálculos e fala em níveis de teste (importante saber todos os níveis de teste), logo – para testar um componente – utilizam-se testes unitários.

Gabarito: Letra A

64.(CESPE / TRE-BA – 2017) O gestor de um órgão organizador de concursos públicos pretende oferecer condições para que mais de um milhão de candidatos inscritos em determinado evento possa obter o gabarito das provas a partir do acesso ao seu sistema eletrônico. Nessa situação, para verificar se o sistema eletrônico suportará uma quantidade grande de acessos simultâneos, a equipe de TI do órgão, ao preparar o ambiente de acesso eletrônico, deverá realizar o teste:

a) de estresse.
b) unitário.
c) integrado.
d) de sistema.
e) de regressão.

Comentários:

Essa questão é bastante mal escrita, visto que uma "*grande quantidade*" é muito subjetivo! Sabemos que testes de estresse testam cargas acima das especificadas/esperadas até descobrir em que ponto o sistema começa a falhar. Logo, para mim, essa questão não tem resposta – porém, a primeira opção é a menos errada.

Gabarito: Letra A

65.(CESPE / TRT-CE – 2017) A respeito de engenharia de software, assinale a opção correta.

a) A finalidade dos testes de segurança é garantir que o sistema se recupere de uma falha e esteja apto a retomar o processamento em um prazo preestabelecido.

b) Efetuar testes de regressão consiste em reexecutar testes já finalizados para garantir que eventuais alterações não tenham impactado funções que antes funcionavam corretamente.

c) Os testes de integração ascendentes são caracterizados pelo fato de a sua realização ocorrer conforme o desenvolvimento dos módulos.

d) Na etapa de desenvolvimento de um software, os testes de validação e de integração são executados simultaneamente, para identificar inconsistências antes da entrega final.

Comentários:

(a) Errado, esse é o teste de recuperação; (b) Correto, é a definição impecável de teste de regressão; (c) Errado, testes de integração ascendentes são caracterizados pela integração dos módulos de baixo nível primeiro; (d) Errado, testes de validação são posteriores.

Gabarito: Letra B

---

66. (CESPE / TJDFT – 2015) As atividades de validação incluem os testes unitários e os de aceitação.

Comentários:

Testes Unitários não são atividades de validação, mas de verificação! À princípio, a banca deu o item como correto, mas depois o anulou sob a justificativa de que "*a redação do item prejudicou seu julgamento objetivo*". Galera, na minha opinião, não há nada de errado com a redação – o item está completamente errado.

Gabarito: ANULADO

---

67. (CESPE / MEC – 2015) A automação de testes apresenta maior impacto positivo sobre a realização de testes de regressão do que testes de usabilidade.

Comentários:

Correto! Automatizar testes de usabilidade não é simples, já testes de regressão são facilmente automatizáveis.

Gabarito: Correto

---

68. (CESPE / MPU – 2013) Para se avaliar a documentação do projeto do software, deve ser utilizado o teste de unidade.

Comentários:

Testes de Unidade, de fato, auxiliam a documentação. No entanto, não se pode dizer que ele **deve** ser utilizado. Eu acredito que o mais adequado para avaliar a documentação do projeto do software seria o teste de integração ou de sistema.

Gabarito: Errado

---

69. (CESPE / ANTT – 2013) O teste de aceitação pode utilizar um processo chamado de teste alfa e beta, sendo conduzido por desenvolvedores e podendo contar com a participação do usuário. O teste alfa é realizado em ambiente real e o beta em ambiente controlado.

Comentários:

Testes Alfa ocorrem em um ambiente controlado e Testes Beta ocorrem em um ambiente real.

Gabarito: Errado

---

70. (CESPE / TCE-RO – 2013) Os principais níveis de teste de software são os de caixa branca, os de caixa preta, os de sistema e os de aceitação.

Comentários:

Testes Caixa-Branca e Testes Caixa-Preta não são níveis de testes, mas técnicas de testes.

Gabarito: Errado

---

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

Comentários:

Observe que a questão mencionou algo bastante genérico. *Testes Funcionais podem ser aplicados para identificar não conformidades entre o programa e seus requisitos?* Sim! Na verdade, qualquer teste é feito em relação aos seus requisitos (de software ou usuário). Teste de Carga é geralmente em relação a algum requisito não-funcional do software; Teste de Aceitação é em relação a algum requisito do usuário em seu contexto; Teste de Usabilidade pode ser em relação aos dois; e assim por diante.

Gabarito: Correto

---

72. (CESPE / MPU – 2013) Um dos critérios do teste de unidade é o particionamento de equivalência, que consiste no particionamento do domínio de entrada do programa de modo que o conjunto de testes resultantes corresponda a uma representação satisfatória de todo o domínio.

Comentários:

Particionamento de Equivalência é um dos critérios de teste caixa-preta e, não, teste de unidade!

Gabarito: Errado

---

73. (CESPE / MPU – 2013) Para realizar testes de unidade ou estrutural, pode-se utilizar uma representação conhecida como grafo de fluxo de controle de um programa. A partir do grafo, executam-se todos os caminhos do programa, principalmente na presença de laços.

Comentários:

Vamos por partes: teste de unidade é um nível de teste e teste estrutural (ou caixa-branca) é uma técnica de teste - estes conceitos são diferentes e independentes! A técnica de grafos de fluxos de controle é uma técnica do tipo caixa-preta. *Por que?* Porque ela trata do relacionamento entre objetos e, não, de aspectos internos ou estruturais. Além disso, não são executados todos os caminhos do programa - são exercitados todos os objetos e relacionamentos representados nos grafos. Acredito que a questão quis fazer o candidato se confundir com a técnica de caminho básico, em que se trata de estruturas internas (Ex: Laços).

Gabarito: Errado

---

74. **(CESPE / INPI – 2013)** De modo geral, o teste de release é um processo de teste do tipo caixa-branca em que as funcionalidades são verificadas e validadas mediante a avaliação interna dos módulos.

Comentários:

Caixa-Branca? Não, Caixa Preta!

Gabarito: Errado

---

75. **(CESPE / MPE-PI – 2012)** Os testes de unidade são feitos por equipes especializadas em testes, de forma a se garantir que os módulos que compõem o sistema sob construção estejam funcionando de acordo com as especificações.

Comentários:

Eles são executados pelos próprios desenvolvedores e, não, por uma equipe especializada em testes.

Gabarito: Errado

---

76. **(CESPE / MPE-PI – 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.

Comentários:

Galera, é mais simples do que parece! Nós já vimos que é inviável (se não, impossível) testar todas as entradas possíveis de um sistema. A Técnica de Partição de Equivalência nos permite otimizar o processo de testes por meio da separação das entradas em classes ou categorias diferentes. Vamos

ver um exemplo: nós estamos fazendo um sistema de uma escola que recebe a nota final de um aluno! Como ficaria...

Se Nota < 5.0 → Aluno reprovado
Se Nota >= 5.0 e Nota < 7.0 → Aluno em dependência
Se Nota > 7.0 → Aluno aprovado

Ora, vejam que beleza! Nós reduzimos as entradas para quatro classes. Professor, só estou vendo três! Isso é porque nós sempre temos que considerar uma classe de entradas inválidas (Ex: Nota = $%@& - isso não é uma entrada válida). Dessa forma, eu não preciso testar todos os valores de 0.0 a 5.0, basta escolher um valor (Ex: 3.5), porque todos os outros valores dessa classe se comportarão da mesma maneira. Galera, aqui estou sendo bem simplista por conta do exemplo, mas é possível especificar várias outras entradas, logo teríamos mais partições. As regras para utilizar essa técnica são:

(1) A divisão das partições deve ser bem clara e definida;
(2) Não deve ser possível que um elemento se enquadre em partições diferentes;
(3) Não é permitida uma partição que não possua membros (vazia);
(4) Valores inválidos devem ser considerados.

Gabarito: Correto

---

77. (CESPE / MEC – 2011) Ao ser estabelecido, um plano de testes necessita de diversos insumos, sendo um deles a estratégia de testes.

Comentários:

O plano de testes necessita realmente – como insumo – da abordagem ou da estratégia de testes.

Gabarito: Correto

---

78. (CESPE / MEC – 2011) Na definição do documento referente ao plano de testes, devem ser incluídos os tipos e a metodologia dos testes. No entanto, critérios de aceitação e processos associados fogem ao escopo desse documento e devem ser inseridos na análise dos riscos.

Comentários:

Na verdade, critérios de aceitação e processos associados estão abrangidos pelo Plano de Testes.

Gabarito: Errado

---

79. (CESPE / TJ-ES – 2011) No plano de teste, um documento de nível gerencial, definem-se como o teste vai ser realizado, quem vai executar os testes, o prazo estimado e o nível de qualidade esperado.

Comentários:

Ele é realmente um artefato gerencial que define como o teste será realizado (abordagem); quem executará os testes (responsabilidades); prazo estimado (cronograma); e nível de qualidade (critérios de aceitação).

Gabarito: Correto

80.(CESPE / MEC – 2011) O teste denominado caixa-preta é utilizado para verificar se os requisitos do software são atendidos, sem verificar o código ou a lógica do componente testado.

Comentários:

Perfeito, ele não necessita verificar o código ou a lógica interna do componente testado.

Gabarito: Correto

---

81. (CESPE / MEC – 2011) O teste caixa-branca ou teste de caixa de vidro é um método de projeto de casos de teste que usa a estrutura de controle do projeto procedimental para derivar casos de teste. Dessa maneira garante-se que todos os caminhos independentes de um módulo tenham sido exercitados pelo menos uma vez, já que erros lógicos e pressupostos incorretos são inversamente proporcionais à probabilidade de que um caminho de programa vai ser executado.

Comentários:

Perfeito! Percebam que – quanto mais um caminho é percorrido ou executado – menor a quantidade de erros lógicos e pressupostos incorretos encontrados.

Gabarito: Correto

82.(CESPE / MEC – 2011) Quando o objetivo é testar uma funcionalidade, assegurando-se que, para todo tipo de entrada, a saída observada corresponda àquela esperada, pode-se alcançar esse objetivo fazendo-se uso de testes do tipo caixa-branca.

Comentários:

A questão trata, na verdade, de testes caixa-preta, em que um comportamento somente pode ser determinado por meio de suas entradas e saídas relacionadas.

Gabarito: Errado

83.(CESPE / BRB – 2011) O teste de regressão tem o objetivo de localizar defeitos na estrutura interna do produto, exercitando, suficientemente, os possíveis caminhos de execução do sistema.

Comentários:

Não, isso seria o Teste de Caminho Básico.

Gabarito: Errado

84.(CESPE / SAD-PE – 2010) A respeito do plano de teste, um registro do processo de planejamento de testes de software, assinale a opção correta.

a) O processo de planejamento de testes é usualmente descrito em um plano de testes.

b) Um plano de teste de software é um registro da execução de um caso de teste de software.

c) A automação de um teste de integração é mais facilmente empreendida que a de um teste de módulo.

d) A produção de scripts de teste deve preceder a eventual construção de casos de teste.

e) Ao se inspecionar o conteúdo de um plano de testes, devem-se encontrar, entre outras, as seguintes descrições: escopo de testes, abordagens de teste, recursos para realização dos testes e cronograma das atividades de teste a serem realizadas.

Comentários:

(a) Errado. O processo de planejamento de testes gera o Plano de Testes – ele não é descrito por ele; (b) Errado. Ele é um planejamento para execução do teste de software; (c) Errado. Basta raciocinar! *Quem é mais difícil de automatizar?* Ora, um teste de integração é muito mais complexo que um teste de módulo. Logo, a questão não faz sentido; (d) Errado. A elaboração de scripts de testes ocorre na etapa de especificação; a execução ocorre posteriormente na etapa de execução; (e) Correto. O escopo é representado pelas características a serem testadas ou não; e os recursos são representados pelas necessidades de equipe e de treinamento.

Gabarito: Letra E

85.(CESPE / MPE-TO – 2010) Entre os diversos níveis possíveis de testes de software, há os chamados testes de unidade (Unit Tests), que procuram testar o programa como um todo,

dentro de um contexto totalmente integrado, procurando validar todas as suas potencialidades de forma unificada.

Comentários:

A questão descreve testes de sistema e, não, testes de unidade.

Gabarito: Errado

---

86. (CESPE / TJ-ES – 2010) No teste de unidade, o software é forçado a falhar de diversos modos a fim de verificar se os requisitos funcionais foram adequadamente implementados. As unidades, sejam funções, procedimentos, métodos ou classes, são testadas duas a duas. Nesse teste, espera-se identificar erros relacionados a algoritmos incorretos ou mal implementados, estruturas de dados incorretas ou simples erros de programação.

Comentários:

Testes Unitários são feitos um-a-um! Além disso, eles não forçam o software até a falha, essa é uma característica do teste de recuperação.

Gabarito: Errado

---

87. (CESPE / ABIN – 2010) Nos testes de caixa branca, o código-fonte do programa é usado para identificar testes de defeitos potenciais, particularmente no processo de validação, o qual demonstra se um programa atende a sua especificação.

Comentários:

Para validação, utiliza-se o teste caixa-preta; para verificação, utiliza-se o teste caixa-branca, em que se tem acesso ao código-fonte do programa para identificar testes de defeitos potenciais.

Gabarito: Errado

---

88. (CESPE / INMETRO – 2010) Os testes caixa preta (Black Box) avaliam as cláusulas de código, a lógica interna do componente codificado, as configurações e outros elementos técnicos.

Comentários:

Lógica interna do componente codificado? Não, isso é Teste Caixa-Branca!

Gabarito: Errado

---

89. (CESPE / INMETRO – 2010) Testes de caixa preta são usualmente fundamentados na análise do código de um programa. Por outro lado, entre as técnicas de teste não relacionadas a testes de caixa preta, estão aquelas embasadas na intuição do testador, em especificações comportamentais e no uso.

Comentários:

Análise de código? Não, isso é Teste Caixa-Branca!

Gabarito: Errado

90.(CESPE / INMETRO – 2010) Os testes caixa branca (White Box) verificam a funcionalidade e a aderência aos requisitos, em uma ótica externa ou do usuário, sem que se tenha qualquer conhecimento do código e da lógica interna do componente testado.

Comentários:

Pelo contrário, a questão trata dos testes caixa-preta.

Gabarito: Errado

---

91. (CESPE / INMETRO – 2010) O teste de caminho básico é uma técnica que identifica as rotinas normalmente usadas, deixando de lado as rotinas eventualmente executadas.

Comentários:

O teste de caminho básico é uma técnica que identifica todas as rotinas e caminhos eventualmente executados!

Gabarito: Errado

---

92. (CESPE / TRE-ES – 2010) O teste de partições caracteriza-se por ser um projeto de caso de teste, em que o conhecimento da estrutura do programa é utilizado para projetar testes que verificam todas as partes desse programa.

Comentários:

Não, teste de partições de equivalência é uma técnica de teste caixa-preta, portanto não utiliza conhecimento da estrutura do programa!

Gabarito: Errado

---

93. **(CESPE / TRE-BA – 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.

Comentários:

Perfeito! Teste funcional é um teste caixa-preta utilizado para projetar casos de teste em que são fornecidas entradas e avaliadas as saídas geradas.

Gabarito: Correto

---

94. **(CESPE / TJ-ES – 2010)** O teste de integração, a exemplo do teste caixa-branca, focaliza o esforço de validação na menor unidade de projeto do software e, com o uso de técnicas de componentização, caminhos de controle relevantes são testados para descobrir erros dentro dos limites do componente.

Comentários:

Quem concentra esforços de validação na menor unidade de projeto do software é o teste de unidade e, não, teste de integração.

Gabarito: Errado

---

95. **(CESPE / MPU – 2010)** O teste de integração geralmente é um processo de teste de caixa-preta no qual os testes são derivados da especificação do sistema, cujo comportamento pode ser determinado por meio do estudo de suas entradas e saídas.

Comentários:

Na verdade, testes de integração são considerados testes caixa-cinza, visto que misturam testes de caixa-branca e testes de caixa-preta.

Gabarito: Errado

---

96. **(CESPE / TJ-ES – 2010)** O teste de caixa-preta é utilizado quando uma nova versão do software está sendo lançada ou quando um novo ciclo de testes for necessário em paralelo ao desenvolvimento do mesmo.

Comentários:

Na verdade, isso é um teste de regressão, que é um teste caixa-branca.

Gabarito: Errado

---

97. **(CESPE / INMETRO – 2010)** Um teste de regressão pode ser o primeiro teste a ser realizado no software.

Comentários:

Não, isso é impossível! Teste de Regressão é aplicado em novas versões, portanto ele não pode ser o primeiro teste a ser realizado.

Gabarito: Errado

---

98. **(CESPE / TRE-BA – 2010)** Se um software já testado receber modificações e, após isso, somente essas modificações forem testadas, a aplicação do teste de regressão a esse software testará inclusive as partes que não tenham sido modificadas.

Comentários:

Perfeito! O teste de regressão testará tudo...

Gabarito: Correto

---

99. **(CESPE / TRE-MT – 2010)** O teste alfa é conduzido pelo cliente em seu ambiente de uso final.

Comentários:

Não, a questão trata do teste beta.

Gabarito: Errado

---

100. **(CESPE / INMETRO – 2010)** Um teste de recuperação deve evitar que o sistema apresente falhas que interrompam o seu funcionamento.

Comentários:

Teste algum consegue evitar isso – teste de recuperação apenas buscará verificar a capacidade de recuperação de um sistema.

Gabarito: Errado

---

101. **(CESPE / TRE-PR – 2009)** Nos testes de integração, realizados antes dos testes unitários, os componentes são construídos e testados separadamente.

Comentários:

Na verdade, testes de integração ocorrem após os testes unitários.

Gabarito: Errado

---

102. (CESPE / TRE-PR – 2009) O teste de aceitação envolve a integração de dois ou mais componentes que implementam funções ou características do sistema. Existem duas fases distintas de teste do sistema: testes de integração e teste de caixa de vidro.

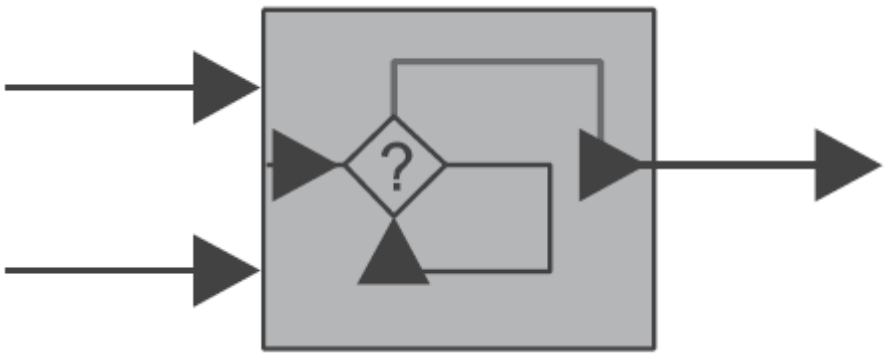
Comentários:

A questão trata de testes de integração e, não, de testes de aceitação. Além disso, não existe essa subdivisão de testes de sistema.

Gabarito: Errado

---

103. (CESPE / TRE-BA – 2009) A figura a seguir ilustra esquematicamente a técnica estrutural de teste de software (ou teste caixa-branca), que avalia o comportamento interno do componente de software, atuando diretamente sobre o código-fonte do componente para realizar testes de condição, de fluxo de dados, de ciclos e de caminhos lógicos.



Comentários:

A imagem mostra uma caixa em que é possível visualizar as estruturas internas e o fluxo de controle, logo se trata de um teste caixa-branca. Ele realmente avalia o comportamento interno do componente de software, atuando diretamente sobre o código-fonte do componente para realizar testes de condição, de fluxo de dados, de ciclos e de caminhos lógicos.

Gabarito: Correto

---

104. (CESPE / TRE-PR – 2009) Enquanto o teste caixa-preta é estrutural ou orientado à lógica, o teste caixa-branca é funcional, orientado a dado ou orientado a entrada e saída.

Comentários:

Testes Caixa-Preta são comportamentais, funcionais, orientados a dado ou à entrada/saída e os Testes Caixa-Branca são estruturais ou orientados à lógica.

Gabarito: Errado

---

105. (CESPE / TRE-PR – 2009) Entre os tipos de testes de caixa preta, encontram-se o teste baseado em grafos; o particionamento de equivalência; a análise de valor-limite; e o teste de matriz ortogonal.

Comentários:

De fato, todas essas são técnicas de teste caixa-preta.

Gabarito: Correto

---

106. (CESPE / CEHAP-PB – 2009) Aplicado ao final do processo de teste, o teste caixa-preta ou comportamental é baseado nos requisitos funcionais do software.

Comentários:

Perfeito, testes caixa-preta ou comportamentais são aqueles em que não se tem acesso ao conteúdo interno dos componentes e são baseados em requisitos funcionais do software.

Gabarito: Correto

---

107. (CESPE / CEHAP-PB – 2009) O teste gama envolve a liberação do sistema a uma série de clientes potenciais que concordam em usar esse sistema.

Comentários:

Na verdade, esse é o Teste Beta!

Gabarito: Errado

---

108. (CESPE / CEHAP-PB – 2009) O teste alfa, conhecido como teste de aceitação, encerra-se quando cliente e projetista concordam que o sistema é uma implementação aceitável dos requisitos do sistema e não se aplica a sistemas desenvolvidos para um único cliente.

Comentários:

Testes Beta somente são utilizados quando o sistema é desenvolvido para ser utilizado por múltiplos usuários, caso contrário não faria sentido fazê-los.

De acordo com Sommerville: *"O teste de aceitação é algumas vezes chamado de teste alfa. Os sistemas sob encomenda são desenvolvidos para um único cliente. O processo de teste alfa continua até que o projetista do sistema e o cliente concordem que o sistema liberado é uma implementação aceitável dos requisitos do sistema. Quando um sistema será comercializado como um produto de software, frequentemente é usado um processo de teste denominado teste beta"*.

Gabarito: Errado

---

109. (CESPE / MPE-RR – 2008) No Processo Unificado, um modelo de teste é tipicamente composto por casos de teste, os quais podem especificar como testar cenários específicos de casos de uso. Os casos de teste tipicamente especificam entradas, resultados esperados e outras condições relevantes para as verificações dos cenários.

Comentários:

Ele – em geral – possui os seguintes campos: resumo, pré-condições, entradas, ações, resultados esperados e pós-condições.

Gabarito: Correto

---

110. (CESPE / IPEA – 2008) O teste caixa-preta ou comportamental, aplicado no início do processo de teste, é embasado nos requisitos funcionais do software. Identifica, entre outros, erros de iniciação e término, erros de estrutura de dados, erros de interface e funções incorretas ou omitidas.

Comentários:

No início do processo de teste? Não, geralmente é nas etapas posteriores.

Gabarito: Errado

---

111. (CESPE / Hemobrás – 2008) Teste de usabilidade consiste na análise de um website por um grupo de experts em usabilidade.

Comentários:

Testes de Usabilidade não se restringem a websites e são realizados, em geral, pelos usuários finais.

Gabarito: Errado

---

112. (CESPE / Hemobrás – 2008) As técnicas de avaliação de usabilidade experimentais ou empíricas contam com a participação direta dos usuários e compreendem, basicamente, os testes com usuários por meio do monitoramento de sessões de uso do produto, ou protótipo, em consideração. Em geral, os testes de usabilidade com a participação dos usuários são avaliações confiáveis.

Comentários:

Realmente contam com a participação direta dos usuários e compreendem, basicamente, os testes com usuários por meio do monitoramento de sessões de uso do produto, ou protótipo, em consideração. De fato, testes de usabilidade com a participação dos usuários são avaliações confiáveis.

Gabarito: Correto

---

113. (CESPE / TSE – 2007) Entre os artefatos produzidos por um processo de teste, têm-se os casos de teste. Um caso de teste é uma situação real de uso, pois não pode ser sintetizado a partir de parâmetros predefinidos.

Comentários:

Não se trata de uma situação real! Além disso, pode sim ser sintetizado a partir de parâmetros predefinidos. Há, inclusive, uma sessão para descrição das pré-condições, resultados esperados e pós-condições.

Gabarito: Errado

---

114. (CESPE / TSE – 2006) Os testes são realizados em várias fases de um desenvolvimento. Testes de unidade são de baixo nível, testes de sistema são executados após os de integração, testes beta empregam apenas desenvolvedores.

Comentários:

Testes de Unidade são de baixo nível e Testes de Sistema são executados após os Testes de Integração. No entanto, testes beta empregam usuários e, não, desenvolvedores.

Gabarito: Errado

---

115. (CESPE / TSE – 2006) Um teste de unidade pode ser projetado usando-se uma estratégia caixa branca. Nesse caso, há um foco nos mecanismos internos da unidade sendo testada. A realização de testes caixa branca pode ser apoiada por métricas de cobertura.

Comentários:

Perfeito, foco em mecanismos internos! Ademais, pode ser apoiada por métricas de cobertura porque pode medir a quantidade de código testada por um conjunto de casos de teste.

Gabarito: Correto

---

116. (CESPE / PMV – 2005) O teste de usabilidade em um sítio da Web tem como objetivo identificar problemas de usabilidade e coletar dados relacionados ao desempenho e às preferências dos usuários.

Comentários:

Esse desempenho – de fato – é relacionado à performance dos usuários na realização de tarefas específicas.

Gabarito: Correto

---

117. (CESPE / SESPA-PA – 2004) Para efeito de validação de um software, o beta teste é realizado pelo cliente usuário do software em um ambiente controlado, normalmente nas instalações do desenvolvedor.

Comentários:

Testes Alfa são feitos em ambiente controlado e Testes Beta são feitos em ambiente real.

Gabarito: Errado

---

118. (CESPE / STJ – 2004) Um software-produto, antes de ser lançado no mercado normalmente deve ser testado por usuários reais do sistema. Nessa etapa, configura-se a realização de beta testes.

Comentários:

Perfeito! Testes Beta são normalmente realizados por usuários reais do sistema antes de serem lançados no mercado.

Gabarito: Correto

QUESTÕES COMENTADAS – FCC

1. **(FCC / SANASA CAMPINAS – 2019)** Considere que está em desenvolvimento um projeto de software na SANASA e os Analistas optaram pela reexecução de alguns subconjuntos de testes que já foram conduzidos para garantir que as modificações não tenham propagado efeitos colaterais no software. Este tipo de teste ajuda a garantir que mudanças não insiram erros e comportamentos indesejados e é denominado:

a) Regressão.
b) Fumaça.
c) Unidade.
d) Alfa.
e) Showstopper.

Comentários:

A reexecução de um conjunto de testes de forma a evitar efeitos colaterais no software trata-se do teste de regressão.

Gabarito: Letra A

---

2. **(FCC / SEMEF MANAUS-AM – 2019)** Considerando a realização de testes de caixa branca e preta de software, a equipe técnica deve considerar que o teste de caixa:

a) preta não visa testar a estrutura lógica interna do módulo de software sob teste.
b) branca deve ser feito somente com o sistema completo, com todos os módulos integrados.
c) preta é um teste que exclui do programa o código testado.
d) preta deve ser executado no modo de segurança do compilador em uso.
e) branca visa testar apenas a interface de cada módulo de software.

Comentários:

(a) Perfeito, no teste de caixa preta o testador não possui acesso às estruturas internas do software; (b) Errado, não há necessidade de o sistema estar completo; (c) Errado, não há exclusão do código testado no teste de caixa preta; (d) Errado, não há essa necessidade; (e) Errado, ele pode testar qualquer componente do software ou o sistema por completo.

Gabarito: Letra A

---

3. **(FCC / SEMEF MANAUS-AM – 2019)** Ao realizar testes de unidade de módulos de software, um técnico de TI deve atentar que:

a) um módulo pseudocontrolador é um módulo que contém apenas o número da versão do módulo sob teste.
b) não é necessária a utilização de módulos pseudocontrolados, mas apenas de pseudocontroladores.
c) um módulo pseudocontrolador substitui módulos chamados pelo módulo sob teste.
d) não é necessária a utilização de módulos pseucontroladores, mas apenas de pseudocontrolados.
e) se admite apenas um módulo pseudocontrolado para cada módulo sob teste.

Comentários:

O que são pseudocontroladores? De acordo com Pressman, os pseudocontroladores servem para substituir módulos subordinados (chamados pelo) ao componente a ser testado. Vejam que a tradução não ficou muito boa, mas é exatamente o que diz a alternativa (c). Os módulos pseudocontroladores substituem módulos chamados pelo módulo sob teste.

Gabarito: Letra C

---

4. **(FCC / SEMEF MANAUS-AM – 2019)** A equipe de teste de software deve ter bem entendido que um dos objetivos principais de um teste de software é:

a) determinar o nível de qualidade do software sob análise.
b) reduzir o tamanho do código fonte do software sob análise.
c) detectar falhas ou defeitos no software, de acordo com o estabelecido em sua especificação.
d) demonstrar que o software sob análise não é cópia de outro software.
e) verificar se o software sob análise não contém dados sigilosos.

Comentários:

Os testes de software têm dois objetivos principais primeiro, demonstrar ao desenvolvedor e ao cliente que o software atende aos requisitos especificados; segundo, descobrir falhas ou defeitos no software que apresente comportamento incorreto, não desejável ou em não conformidade com sua especificação.

Gabarito: Letra C

---

5. **(FCC / SEMEF MANAUS-AM – 2019)** Uma equipe de assistentes técnicos está encarregada de realizar os testes do software referente a um projeto. Dessa forma, essa equipe deve considerar que há um tipo de teste de software, no qual são reexecutados conjuntos de testes já realizados, de forma a garantir que a adição de novos módulos de software em um teste de integração não introduza erros até então inexistentes. Tal tipo de teste denomina-se:

a) de regressão.

b) ascendente.
c) descendente.
d) fracionado.
e) integral.

Comentários:

A reexecução de um conjunto de testes trata-se do teste de regressão, de forma a assegurar que as alterações não causem efeitos colaterais indesejados no software.

Gabarito: Letra A

---

6. **(FCC / SEMEF MANAUS-AM – 2019)** A Fazenda Municipal aplica, em seus projetos de software, as práticas de construção de software, dentre as quais está a codificação, que conta com três princípios fundamentais: de preparação, de codificação propriamente dita e de validação, sendo certo que:

a) entender a arquitetura do software é um dos princípios de preparação.
b) conduzir inspeções de código é um dos princípios de validação.
c) selecionar nomes significativos para as variáveis do software é um dos princípios de preparação.
d) entender os princípios e conceitos básicos do projeto é um dos princípios de preparação.
e) realizar testes unitários e corrigir erros do software é um dos princípios de codificação propriamente dita.

Comentários:

Essa questão trata de princípios retirados livro do Pressman. De acordo com ele, há alguns princípios fundamentais, dentre eles o princípio da preparação. Esse princípio diz que antes de escrever uma linha de código, certifique-se de que: (1) Compreendeu bem o problema a ser solucionado. (2) Compreendeu bem os princípios e conceitos básicos sobre o projeto.

Gabarito: Letra D

---

7. **(FCC / SEFAZ-BA – 2019)** Suponha que uma Auditora Fiscal da área de TI atue na etapa de testes e avaliação da qualidade de um software em desenvolvimento. Como o software sofria alterações a cada nova funcionalidade a ele incorporada, a Auditora propôs que a equipe de testes adotasse como padrão um tipo de teste que garantisse que as mudanças recentes no código deixassem o resto do código intacto, visando impedir a introdução de erros. A equipe decidiu realizar um tipo de teste para testar a parte modificada e as áreas adjacentes que podem ter sido afetadas, dentro de uma abordagem baseada em risco. Assim, os testadores destacariam as áreas de aplicação que poderiam ser afetadas pelas recentes alterações de

código e selecionariam os casos de testes relevantes para o conjunto de testes. Procedendo desta forma, seriam realizados testes:

a) de Revisão de Funcionalidade.
b) Gama.
c) de Aceite Operacional.
d) de Regressão.
e) de Caixa-preta.

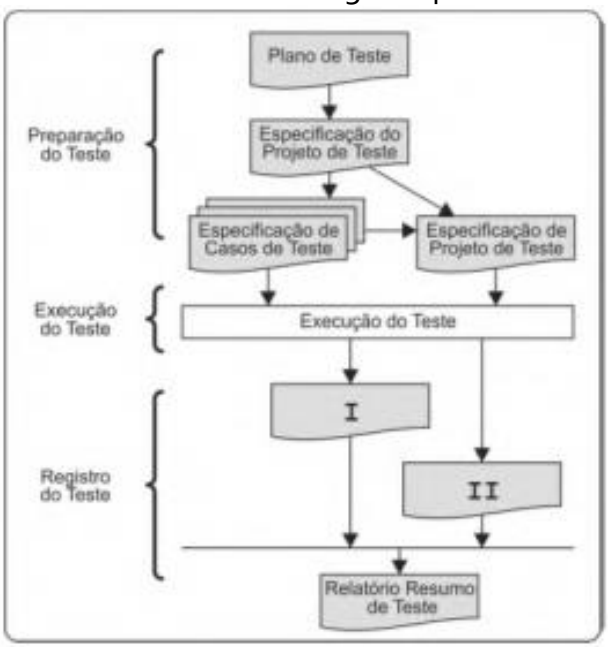
Comentários:

O teste que faz a reexecução do mesmo subconjunto de testes que já foram executados para assegurar que as alterações não tenham propagado efeitos colaterais indesejados é o teste de regressão.

Gabarito: Letra D

---

8. **(FCC / SEFAZ-BA – 2019)** Considere o procedimento apresentado na figura a seguir, no qual são utilizados documentos consistentes e adequados capazes de definir, registrar e prover condições de análise dos resultados obtidos ao longo do processo de testes de software.



Na etapa de Registro do Teste, I corresponde ao:

a) Registro de Testes Verde-Vermelho e II corresponde aos Critérios para Homologação de Teste.
b) Log de Teste e II corresponde ao Relatório de Incidentes de Teste.
c) Gap de Teste e II corresponde à Refatoração de Testes.

d) Registro de Testes Funcionais e Não Funcionais e II corresponde ao Relatório de Análise Ciclomática de Teste.
e) Relatório de Testes de Usabilidade e II corresponde ao Relatório de Testes Sincronizados.

Comentários:

De acordo com a IEEE 829, que está relacionado ao processo de testes: (I) Diário de teste ou log de teste – registram a execução dos casos de teste; (II) Relatório de incidente de teste - todos os defeitos encontrados durante o teste são registrados e passados para a equipe de desenvolvimento para as devidas correções.

Gabarito: Letra B

---

9. **(FCC / SEFAZ-SC – 2018)** Os testes unitários são aplicados em subprogramas individuais ou em componentes maiores construídos com unidades altamente coesas e são executados:

I. sempre com acesso ao código que está sendo testado.
II. normalmente com o suporte de ferramentas de depuração.
III. sempre pelos programadores que escreveram o código.
IV. para verificar o funcionamento dos elementos de software separadamente.

Está correto o que consta de:

a) I e IV, apenas.
b) I, II e III, apenas.
c) I, II e IV, apenas.
d) II, III e IV, apenas.
e) I, II, III e IV.

Comentários:

(I) Correto, deve-se ter acesso ao código do componente; (II) Correto, pode-se fazer uso de ferramentas de depuração, é importante relembrar que o teste e a depuração são atividades diferentes, mas a depuração deve ser associada com alguma estratégia de teste; (III) Errado, não necessariamente deve ser feito pelos programadores que desenvolveram o código, pode ser feito por outros atores ou – até mesmo – de forma automatizada; (IV) Correto, justamente por isso é chamado de teste de unidade.

Gabarito: Letra C

---

10. **(FCC / DP-AM – 2018)** Considere, por hipótese, que na Defensoria esteja sendo desenvolvido um projeto com prazo crítico, sendo necessário que os desenvolvedores avaliem o software

frequentemente. A equipe envolvida decidiu utilizar uma abordagem de teste de integração que trabalha da seguinte maneira:

I. Componentes necessários para implementar funções do software, como arquivos de dados, bibliotecas, módulos reutilizáveis etc são integrados em uma build (construção).

II. Diversos testes são projetados para que erros que possam impedir a build em andamento de desempenhar de forma adequada sua função, com o objetivo de descobrir showstoppers que impliquem em atrasos no cronograma.

III. A build é integrada a outras builds e todo o software passa diariamente por este tipo de teste, podendo usar abordagem ascendente ou descendente de integração.

O teste de integração descrito é denominado teste:

a) de fumaça.
b) de regressão.
c) top-down.
d) breadth-first.
e) de caixa cinza (grey box).

Comentários:

Trata-se do teste de fumaça. Pressman define o teste de fumaça como uma abordagem de teste de integração que é usada frequentemente quando produtos de software são desenvolvidos. Além disso, ele é projetado como um mecanismo de marca-passo para projetos com prazo crítico, permitindo que a equipe de software avalie o projeto frequentemente. Por fim, todas as atividades citadas nos itens (I), (II) e (III) são atividades do teste de fumaça.

Gabarito: Letra A

---

11. (FCC / TCM-GO – 2015) Um Auditor de Controle Externo do Tribunal de Contas dos Municípios do Estado de Goiás da área de TI indicou a seguinte estratégia convencional para testes de um sistema que está sendo desenvolvido:

I. Para cada componente ou módulo, testar a interface, a estrutura de dados local, os caminhos independentes ao longo da estrutura de controle e as condições-limite para garantir que a informação flui adequadamente para dentro e para fora do módulo, que todos os comandos tenham sido executados e que todos os caminhos de manipulação de erros sejam testados.

II. Aplicar uma abordagem incremental de testes para a construção da arquitetura do sistema, de forma que os módulos testados sejam integrados a partir do módulo de controle principal e os testes sejam conduzidos à medida que cada componente é inserido.

O Auditor indicou em I e II, respectivamente, os testes de:

a) caixa branca e de caixa preta, que são suficientes para validar todo o sistema.

b) unidade e de integração; na sequência, indicou os testes de validação e de sistema que são adequados para validar todo o sistema.

c) unidade e de interoperabilidade; na sequência, indicou os testes de caixa branca e de caixa preta que são adequados para validar todo o sistema.

d) carga e de desempenho; na sequência, indicou os testes de usabilidade e interoperabilidade que são adequados para validar todo o sistema.

e) caixa preta e de caixa branca, que são suficientes para validar todo o sistema.

Comentários:

Testar interface de módulos é característica do teste de unidade; e aplicar uma abordagem incremental para a construção da arquitetura do sistema é característica do teste de integração. Além disso, os testes de validação e sistema são realmente adequados para validar todo sistema.

Gabarito: Letra B

---

12. (FCC / TCM-GO – 2015) Um Auditor de Controle Externo do Tribunal de Contas dos Municípios do Estado de Goiás da Área de TI recebeu a tarefa de identificar testes que sejam capazes de verificar:

– a validade funcional do sistema;
– o comportamento e o desempenho do sistema;
– quais classes de entrada vão constituir bons casos de teste;
– se o sistema é sensível a certos valores de entrada;
– quais taxas e volumes de dados o sistema pode tolerar;
– que efeito combinações específicas de dados terão na operação do sistema.

A indicação correta do Auditor é utilizar:

a) testes de caixa branca.
b) mais de um tipo de teste, pois não há um único tipo de teste capaz de avaliar todas estas situações.
c) um tipo diferente de teste para cada uma das situações elencadas.
d) testes de caixa preta.
e) testes de desempenho para os 2 primeiros e de carga para os demais.

Comentários:

Apenas o teste caixa-preta já seria suficiente, porque ele cumpre todos os requisitos! Não é necessário conhecer a estrutura interna para realizar nenhum desses testes. Além disso, não faz sentido fazer testes de desempenho para verificar a validade funcional do sistema – testes de desempenho tratam de requisitos não-funcionais.

Gabarito: Letra D

---

13. (FCC / TRT1 – 2014) Considerando o teste de software, há o chamado teste de unidade, que consiste em testar:

a) o software completo, incluindo todos os seus componentes ou módulos, no ambiente de testes.
b) o funcionamento dos compiladores que estiverem sendo utilizados no desenvolvimento do software.
c) individualmente, componentes ou módulos de software que, posteriormente devem ser testados de maneira integrada.
d) o software completo em seu ambiente final de operação, já com o hardware base do projeto.
e) apenas componentes ou módulos de software cujo código fonte tenha mais de 100 linhas.

Comentários:

(a) *Software completo? Incluindo todos os seus componentes ou módulos? No ambiente de testes?* Isso é um teste alfa; (b) Testes de Unidade não têm nenhuma relação com funcionamento de compiladores; (c) Correto, testam-se componentes e módulos individualmente e, posteriormente, testa-se em conjunto no teste de integração; (d) *Software completo? Em seu ambiente final de operação? Com o hardware base do projeto?* Isso é um teste beta! (e) Esse item não faz o menor sentido! Não é porque é um teste de unidade que ele é pequeno – não há relação com a quantidade de linhas.

Gabarito: Letra C

---

14. (FCC / DPE-SP – 2013) Para aplicações convencionais, o software é testado a partir de duas perspectivas diferentes: a lógica interna do programa é exercitada usando técnicas de projeto de caso de teste ...I... e os requisitos de software são exercitados usando técnicas de projeto de casos de teste ...II... .

O teste ...I... fundamenta-se em um exame rigoroso do detalhe procedimental. Os caminhos lógicos do software e as colaborações entre componentes são testados exercitando conjuntos específicos de condições e/ou ciclos.

O teste ...II... faz referência a testes realizados na interface do software. Esse tipo de teste examina alguns aspectos fundamentais de um sistema, com pouca preocupação em relação à estrutura lógica interna do software.

As lacunas I e II são preenchidas correta e respectivamente, com:

a) de caminho básico - caixa-de-vidro
b) alfa - beta
c) caixa branca - caixa preta
d) de ciclo - de usabilidade
e) unitário - de interface

Comentários:

O teste caixa-branca fundamenta-se em um exame rigoroso do detalhe procedimental. Os caminhos lógicos do software e as colaborações entre componentes são testados exercitando conjuntos específicos de condições e/ou ciclos. O teste caixa-preta faz referência a testes realizados na interface do software. Esse tipo de teste examina alguns aspectos fundamentais de um sistema, com pouca preocupação em relação à estrutura lógica interna do software.

Gabarito: Letra C

---

15. (FCC / TRT-PE – 2012) No que se refere a testes de software, é correto afirmar que:

a) o teste de operação é a fase onde é testada a ergonomia da interface de uso do software.

b) o teste da caixa preta (teste funcional), baseia-se em analisar os arquivos de log do sistema procurando por mensagens de funcionamento inconsistente.

c) um teste bem sucedido é um teste que não encontra nenhum erro no software.

d) o teste da caixa branca (teste estrutural), baseia-se em testar as estruturas do código fonte, como comandos condicionais e de repetição.

e) um caso de teste é uma categoria de possíveis resultados na execução de testes.

Comentários:

(a) Errado, esse é o teste de usabilidade; (b) Errado, ela verifica a funcionalidade e aderência aos requisitos, em uma ótica externa ou do usuário, sem se basear em qualquer conhecimento do código ou lógica interna do componente de software; (c) Errado, um teste bem-sucedido é aquele que encontra erros – se ele não encontrar erros, é mais provável que o teste tenha sido mal feito;

(d) Correto, essa é a definição de Teste Caixa-Branca; (e) Errado, não são possíveis resultados, são possíveis entradas e os resultados esperados para essas possíveis entradas.

Gabarito: Letra D

---

16. (FCC / TRE-PE – 2011) Com relação aos testes de software, é correto afirmar:

a) Um princípio muitas vezes adotado ao testar um software é o de Pareto. Ele afirma que existe um forte desequilíbrio entre causas e efeitos, entre esforços e resultados e entre ações e objetivos alcançados.

b) Testes sempre podem mostrar a ausência de erros.

c) Para que o resultado de um teste de software seja confiável, é preciso garantir que os casos de teste utilizados cubram um número reduzido de possibilidades de execução.

d) Um software que produz saídas corretas deve ser aprovado, pois isso demonstra que todos os erros foram corrigidos.

e) Um programador deve testar seu próprio código porque facilmente conseguirá criar um caso de teste que rompe com a lógica de funcionamento do seu código.

Comentários:

(a) Correto, o princípio de pareto é o 80/20, i.e., há – sim – um desequilíbrio entre causas e efeitos, entre esforços e resultados e entre ações e objetivos alcançados, ou seja, 20% das causas resultam em 80% dos defeitos; 20% dos esforços provocam 80% dos resultados; 20% ações resultam em 80% dos objetivos alcançados; (b) Errado, testes de software não garantem a ausência de erros – jamais errem isso; (c) Errado, *um número reduzido de possibilidades de execução?* Não, um número amplo de possibilidades de execução; (d) Errado, testes não garantem ausência de erros – se ele produziu saídas corretas, não necessariamente ele deve ser aprovado. Muitas vezes, não foram realizados testes suficientes ou eventualmente a qualidade dos testes foi baixa; (e) Errado, em geral, o melhor caminho é que outra pessoa teste o código. O próprio programador pode construir um teste enviesado, de maneira que resulte nas saídas esperadas.

Gabarito: Letra A

---

17. (FCC / TRE-PE – 2011) Com relação aos testes de software, é correto afirmar:

a) Um princípio muitas vezes adotado ao testar um software é o de Pareto. Ele afirma que existe um forte desequilíbrio entre causas e efeitos, entre esforços e resultados e entre ações e objetivos alcançados.

b) Testes sempre podem mostrar a ausência de erros.

c) Para que o resultado de um teste de software seja confiável, é preciso garantir que os casos de teste utilizados cubram um número reduzido de possibilidades de execução.

d) Um software que produz saídas corretas deve ser aprovado, pois isso demonstra que todos os erros foram corrigidos.

e) Um programador deve testar seu próprio código porque facilmente conseguirá criar um caso de teste que rompe com a lógica de funcionamento do seu código.

Comentários:

(a) Correto. 20% dos erros causam 80% dos defeitos, 20% dos esforços gera 80% dos resultados e 20% das ações alcançam 80% dos objetivos; (b) Errado, testes nunca podem demonstrar ausência de erros; (c) Errado, ele deve ser amplo; (d) Errado, sabemos que isso é impossível; (e) Errado, não é recomendável que ele mesmo teste seu código, visto que dificilmente conseguirá criar um caso que rompa com a lógica que ele mesmo criou.

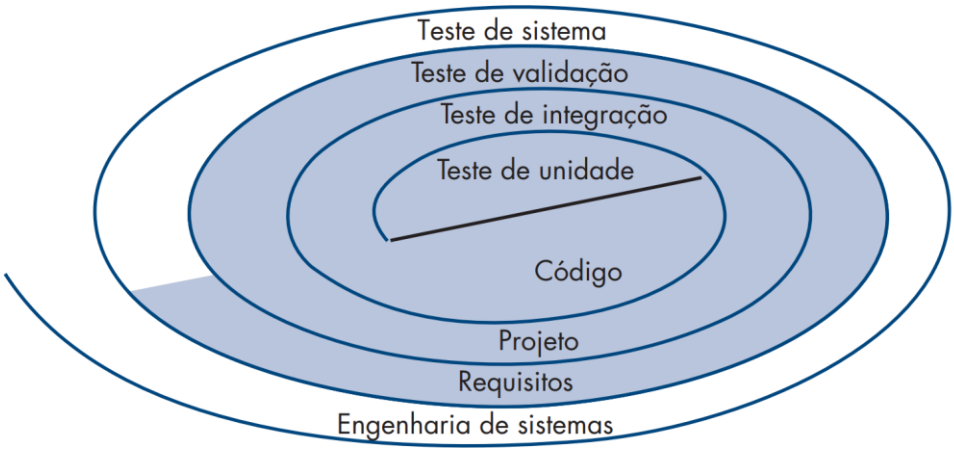
Gabarito: Letra A

---

18. (FCC / INFRAERO – 2011) Na direção dos tipos de teste focados pela engenharia de software, os testes de integração cuidam dos tópicos associados com os problemas de verificação:

a) da engenharia de sistemas.
b) do projeto do software.
c) dos códigos do programa.
d) dos requisitos funcionais.
e) dos requisitos não funcionais.

Comentários:



Bastava lembrar da espiral! Testes de Integração estão associados ao Projeto de Software.

Gabarito: Letra B

---

19. (FCC / TRT14 – 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.

Comentários:

Teste de Unidade: ocorrem antes da implementação; Teste de Iteração: desconheço, acredito que a banca inventou; Teste de Integridade: na homologação, esses testes já devem estar finalizados; Teste de Volume: poderia ser feito durante a homologação; Teste Caixa-Branca: não é recomendável fazê-los durante a homologação; Teste de Carga: poderia ser feito durante a homologação; Teste de Configuração: poderia ser feito durante a homologação. A homologação é como uma aceitação oficial e formal do produto.

Logo, trata-se dos testes funcionais, de usabilidade e de aceitação.

Gabarito: Letra A

---

20. (FCC / TRT9 – 2010) O teste de sistema que força o software a falhar de diversos modos e verifica o retorno do processamento dentro de um tempo pré-estabelecido é um tipo de teste de:

a) Integração.
b) Estresse.
c) Recuperação.
d) Desempenho.
e) Segurança.

Comentários:

O teste que busca forçar o software a falhar de diversos modos e verifica o retorno do processamento dentro de um tempo pré-estabelecido é claramente o Teste de Recuperação!

Gabarito: Letra C

---

21. (FCC / SEFAZ-SP – 2009) Garantir que um ou mais componentes de um sistema combinados funcionam corretamente é o objetivo do tipo de teste:

a) de sistema.
b) de integração.
c) de configuração.
d) operacional.
e) funcional.

Comentários:

Esse é o objetivo do teste de integração, isto é, testar se a combinação dos componentes funciona do modo correto.

Gabarito: Letra B

---

22. (FCC / TRT15 – 2009) Os testes de integração têm por objetivo verificar se:

a) os módulos testados produzem os mesmos resultados que as unidades testadas individualmente.
b) os módulos testados suportam grandes volumes de dados.
c) as funcionalidades dos módulos testados atendem aos requisitos.
d) os valores limites entre as unidades testadas individualmente são aceitáveis.
e) o tempo de resposta dos módulos testados está adequado.

Comentários:

Acredito que os únicos itens que podem gerar alguma dúvida são: A e C! Essa questão gerou uma boa polêmica! *Por que o primeiro está errado?* Bem, partamos do princípio de que módulo é um conjunto de unidades. Dito isso, não faz nenhum sentido que módulos produzam os mesmos resultados que unidades. *Por que o terceiro está certo?* Bem, esse item foi extremamente genérico, mas não está errado – as funcionalidades dos módulos testados atendem aos requisitos.

Gabarito: Letra C

---

23. (FCC / TRT-MG – 2009) NÃO se trata de uma técnica para testar software o teste de:

a) caixa preta.
b) regressão.
c) desempenho.
d) unidade.
e) carga

Comentários:

Observem que a ÚNICA opção que contém uma Técnica de Teste de software é a primeira (Caixa-Preta). Teste de unidade é nível ou estratégia de teste e o restante é tipo de teste. Dessa forma, essa questão possui quatro respostas, mas a FCC considerou que a correta é apenas a quarta opção. Maluquice!

Gabarito: Letra D

---

24. (FCC / TRT-MA – 2009) Há um tipo de teste que vislumbra a "destruição do programa" por meio de sua submissão a quantidades, frequências ou volumes anormais que é o teste:

a) de recuperação.
b) de configuração.
c) beta.
d) de desempenho.
e) de estresse.

Comentários:

Destruição do programa por meio de sua submissão a quantidades, frequências ou volumes anormais é uma característica do teste de estresse.

Gabarito: Letra E

---

25. (FCC / AFR-SP – 2009) Garantir que um ou mais componentes de um sistema combinados funcionam corretamente é o objetivo do tipo de teste:

a) funcional.
b) de sistema.
c) de integração.
d) de configuração.
e) operacional.

Comentários:

Esse é o objetivo do teste de integração, isto é, testar se a combinação dos componentes funciona do modo correto.

Gabarito: Letra C

---

26. (FCC / TRT-GO – 2008) Uma sistemática para construção da arquitetura do software enquanto, ao mesmo tempo, conduz ao descobrimento de erros associados às interfaces é a estratégia de teste de software denominada de:

a) sistema.
b) unidade.
c) validação.
d) arquitetura.
e) integração.

Comentários:

O Teste de Integração é uma técnica sistemática para construir a arquitetura de software ao mesmo tempo que conduz testes para descobrir erros associados com as interfaces. O objetivo é construir uma estrutura de programa determinada pelo projeto a partir de componentes testados em nível de unidade.

Gabarito: Letra E

---

27. (FCC / METRÔ-SP – 2008) Um critério de teste de software baseado no fluxo de dados de aplicação pode ser utilizado como uma técnica de teste baseada:

a) na especificação.
b) no código.
c) em falhas.
d) no uso da aplicação.
e) na intuição e experiência do engenheiro.

Comentários:

Resolvendo por raciocínio lógico: uma maneira de analisar o fluxo de dados da aplicação é analisar seu código-fonte! Logo, trata-se de uma técnica de teste baseada em código!

Gabarito: Letra B

QUESTÕES COMENTADAS – FGV

1. **(FGV / TJ-MS – 2024)** A testabilidade é um fator importante para o desenvolvimento e a implementação de um software. Uma característica de um software testável é que os estados do software devem ser visíveis e podem ser consultados durante a execução. Essa característica é chamada de:

a) compreensibilidade;
b) controlabilidade;
c) estabilidade;
d) observabilidade;
e) operabilidade.

Comentários:

(a) Errado. Compreensibilidade está relacionada à facilidade de entendimento do software, mas não envolve a visibilidade dos estados durante a execução;

(b) Errado. Controlabilidade refere-se à capacidade de controlar o software, por exemplo, configurando entradas, mas não se aplica diretamente à visibilidade dos estados;

(c) Errado. Estabilidade refere-se à capacidade do software de manter seu desempenho ou funcionamento ao longo do tempo, não à visibilidade dos estados;

(d) Correto. Observabilidade é a capacidade de ver os estados internos do software durante sua execução, o que é fundamental para a testabilidade;

(e) Errado. Operabilidade está relacionada à facilidade de operação do software, não à visibilidade dos estados.

Gabarito: Letra D

---

2. **(FGV / SES-MT – 2024)** Existem várias técnicas de teste de software, cada uma com seus próprios objetivos e métodos específicos. Relacione os testes de software listados a seguir, às suas respectivas definições.

1. Teste Funcional
2. Teste de Conformidade
3. Teste de Caixa Preta
4. Teste de Desempenho

( ) Avalia como o sistema se comporta em termos de velocidade, escalabilidade e estabilidade sob diferentes condições de carga.
( ) Testa o software sem conhecimento interno da lógica ou estrutura do código, focando nos requisitos e funcionalidades visíveis.
( ) Verifica se as funções do software estão operando conforme esperado. Isso pode incluir testes de casos de uso, fluxos de trabalho e requisitos funcionais.
( ) Verifica se o software atende a padrões, regulamentos e requisitos legais.

Assinale a opção que indica a relação correta, na ordem apresentada.

a) 2 – 4 – 3 – 1
b) 2 – 1 – 3 – 4.
c) 4 – 3 – 1 – 2.
d) 4 – 2 – 1 – 3.

Comentários:

(4) Este tipo de teste avalia a capacidade do sistema de operar eficientemente sob diversas condições de carga; (3) Neste teste, a funcionalidade é avaliada sem considerar o código ou a lógica interna; (1) Este teste verifica se o software realiza as funções especificadas corretamente; (2) Este teste assegura que o software cumpre com normas e regulamentações relevantes.

Gabarito: Letra C

---

3. **(FGV / SES-MT – 2024)** A excelência no desenvolvimento de software começa com testes meticulosos, assim como a precisão é crucial em uma obra de arte. Diversas técnicas de teste de software existem, cada uma com propósito e abordagem específicos. Assinale a opção que indica o tipo de teste que assegura que as funcionalidades previamente implementadas não serão afetadas pelas alterações feitas no código:

a) Teste Unitário.
b) Teste de Integração.
c) Teste de Usabilidade.
d) Teste de Regressão.

Comentários:

(a) Errado. O Teste Unitário se concentra em verificar se unidades individuais de código (como funções ou métodos) funcionam corretamente, isoladamente das demais partes do sistema. Ele não é projetado especificamente para garantir que mudanças no código não afetem funcionalidades pré-existentes;

(b) Errado. O Teste de Integração tem como objetivo verificar se diferentes módulos ou componentes de um sistema funcionam corretamente em conjunto. Embora ajude a identificar problemas quando módulos interagem, não é seu foco principal assegurar que funcionalidades antigas não sejam impactadas por novas mudanças;

(c) Errado. O Teste de Usabilidade visa avaliar a facilidade de uso e a eficiência da interface do usuário, garantindo que o sistema seja intuitivo e eficiente para os usuários finais. Ele não se preocupa com a integridade das funcionalidades após alterações no código;

(d) Correto. O Teste de Regressão é projetado especificamente para assegurar que novas mudanças no código não introduzam defeitos ou afetem negativamente funcionalidades já existentes. Ele reexecuta conjuntos de testes anteriores para garantir que o sistema continua funcionando corretamente após modificações.

Gabarito: Letra D

---

4. **(FGV / AL-PR – 2024)** No contexto da Engenharia de Software, os testes de software desempenham um papel consideravelmente importante no âmbito do processo de desenvolvimento. Nesse contexto, os testes de unidade caracterizam-se por:

a) dependerem do sucesso do teste de integração descendente (top-down).
b) representarem validações em componentes que representam programas independentes.
c) serem responsáveis pela verificação na menor unidade do componente ou módulo de software.
d) substituírem paulatinamente a abordagem de teste por fumaça na verificação do software.
e) utilizarem como metodologia de base a abordagem de testes de regressão.

Comentários:

(a) Errado. Os testes de unidade não dependem do teste de integração descendente. Eles são realizados antes da integração e testam as partes mais isoladas do software;

(b) Errado. Componentes que representam programas independentes seriam mais adequados para testes de sistema ou de integração, não de unidade;

(c) Correto. Os testes de unidade são responsáveis pela verificação da menor unidade de software, como funções, métodos ou classes, isoladamente do restante do sistema;

(d) Errado. Os testes de unidade não substituem os testes de fumaça, que são uma forma de testes de aceitação ou testes de sistema para verificar se as principais funções funcionam. Eles têm propósitos diferentes no processo de desenvolvimento de software;

(e) Errado. Testes de regressão são usados para verificar se as novas mudanças não afetam o comportamento existente do software. Embora possam incluir testes de unidade, eles não são uma metodologia base para testes de unidade.

Gabarito: Letra C

---

5. **(FGV / CGE-SC – 2023)** O tipo de teste de software que serve para garantir que todas ou algumas partes de um sistema estão dialogando e funcionando corretamente em conjunto é o teste:

a) de regressão.
b) de aceitação.
c) de integração.
d) de validação.
e) unitário.

Comentários:

(a) Errado. O teste de regressão verifica se as alterações no código não introduziram novos erros em partes previamente funcionais do sistema, mas não se concentra na interação entre diferentes partes do sistema;

(b) Errado. O teste de aceitação é realizado para garantir que o sistema atende aos requisitos e expectativas do usuário final, geralmente antes da entrega do produto, mas não foca especificamente na interação entre componentes do sistema;

(c) Correto. O teste de integração é o tipo de teste de software que visa garantir que diferentes módulos ou componentes de um sistema estão funcionando corretamente em conjunto. Ele verifica as interações entre as partes do sistema para garantir que elas dialogam corretamente;

(d) Errado. O teste de validação é utilizado para garantir que o sistema atende aos requisitos e especificações do cliente, mas não se concentra especificamente na interação entre os componentes do sistema;

(e) Errado. O teste unitário verifica o funcionamento de componentes individuais ou pequenas partes do sistema isoladamente, mas não avalia a interação entre diferentes partes do sistema.

Gabarito: Letra C

---

6. **(FGV / SEFAZ-MT - 2023)** A estratégia de teste software cujo objetivo principal é verificar como um dado software se comporta em um cenário que exige recursos computacionais em quantidades, frequência ou volumes anormais é o teste de:

a) estresse.
b) integração.

c) regressão.
d) unidade.
e) usabilidade.

Comentários:

(a) Correto. O teste de estresse verifica o comportamento do software sob condições extremas de uso, avaliando sua estabilidade e limites;

(b) Errado. O teste de integração foca na interação entre módulos ou componentes, verificando se funcionam corretamente juntos;

(c) Errado. O teste de regressão verifica se alterações no software não introduzem novos defeitos em funcionalidades já existentes;

(d) Errado. O teste de unidade avalia individualmente os menores componentes ou unidades do software, geralmente funções ou métodos;

(e) Errado. O teste de usabilidade avalia a facilidade com que um usuário consegue utilizar o software, não focando em condições extremas;

Gabarito: Letra A

---

7. **(FGV / TCE-TO – 2022)** O analista de sistemas Carlos está desenvolvendo o software CharlieApp e implementou o teste C. O teste C consiste apenas em determinar se o método A do código de CharlieApp retorna o resultado esperado C ao chamar o método B que realiza uma consulta ao banco de dados de CharlieApp.

   Portanto, o teste C implementado por Carlos é de:

   a) unidade;
   b) aceitação;
   c) ponta a ponta;
   d) exploração;
   e) integração.

Comentários:

Note que temos dois métodos: A e B. O Teste C consiste em verificar se o Método A, ao chamar o Método B, retorna o resultado esperado. Logo, o intuito é examinar se ambos os métodos estão bem integrados, logo é um teste de integração.

Gabarito: Letra E

---

8. **(FGV / IMBEL - 2021)** Com referência às metodologias de teste de software, a técnica que avalia as funcionalidades sem ter contato com o código-fonte, mas apenas com as respostas que o sistema dá a determinadas ações, é conhecida como:

a) Caixa Branca.
b) Caixa Cinza.
c) Caixa Preta.
d) Regressão.
e) Testes não funcionais.

Comentários:

(a) Errado, o teste caixa branca tem contato com o código-fonte; (b) Errado, esse teste permitiria o acesso a partes do código-fonte; (c) Correto, avaliar funcionalidades sem ter contato com o código-fonte é função da técnica caixa preta; (d) Errado, teste de regressão não é uma técnica e, sim, um tipo de teste; (e) Errado, testes não funcionais não são técnicas de software (aliás, desconheço testes não-funcionais).

Gabarito: Letra C

---

9. **(FGV / FUNSAÚDE-CE - 2021)** No contexto da testagem de software, os testes do tipo Unitário, aplicam-se normalmente:

a) à aderência a padrões.
b) às funções codificadas.
c) às interfaces de entrada de dados.
d) à integração dos componentes.
e) aos limites de carga.

Comentários:

(a) Errado, isso seria função da análise estática de código-fonte; (b) Correto, o teste unitário é bastante focado em funções/procedimentos do código-fonte; (c) Errado, isso seria função do teste caixa-preta; (d) Errado, isso seria função do teste de integração; (e) Errado, isso seria função do teste de carga.

Gabarito: Letra B

---

10. **(FGV / TCE-AM - 2021)** A Equipe de Desenvolvimento de Software (EDS) de um tribunal de contas está trabalhando na construção de componentes de um novo sistema de software.

Para verificar o funcionamento do software no nível de componente, a EDS deverá aplicar testes de caixa:

a) branca, para validar parâmetros de entrada;
b) preta, para garantir que caminhos independentes dos componentes tenham sido testados;
c) branca do tipo análise de valor-limite;
d) preta como alternativa a testes de caixa branca;
e) branca, para exercitar decisões lógicas em seus lados verdadeiro e falso.

Comentários:

Já que se deseja verificar o funcionamento do software no nível de componentes, entende-se que não se trata de suas interfaces e, sim, de seu funcionamento interno. Logo, trata-se de um teste caixa-branca. Vamos analisar as alternativas: (a) Errado, isso seria uma prática do teste caixa-preta; (b) Errado, o teste no nível de componente é um teste caixa-branca; (c) Errado, análise do valor-limite é um teste caixa-preta; (d) Errado, o teste no nível de componente é um teste caixa-branca e funciona de maneira bastante díspar dos testes caixa-preta; (e) Correto, testes caixa-branca realmente buscam exercitar decisões lógicas internas em seus lados verdadeiros e falsos.

Gabarito: Letra E

---

11. **(FGV / DPE-RJ – 2019)** Uma empresa foi contratada por um órgão governamental para modificar e adaptar um sistema para gerenciamento eletrônico de documentos, com base nas especificações criadas pelo próprio órgão. A contratada entregou ao órgão uma parte do sistema com as alterações solicitadas, e um grupo de usuários finais do sistema está simulando operações de rotina, para atestar se seu comportamento está de acordo com as expectativas da empresa.

Conclui-se que está sendo realizado o teste de:

a) unidade;
b) regressão;
c) integração;
d) aceitação;
e) cobertura.

Comentários:

Vejam que um grupo de usuários finais estão testando o sistema, portanto, trata-se do teste de aceitação, que verifica se o software está de acordo com os requisitos estabelecidos.

Gabarito: Letra D

---

12. **(FGV / DPE-RJ – 2019)** No processo de validação de software, quando os componentes individuais são avaliados para garantir que eles possam operar corretamente, sendo testados independentemente, isto é, sem a presença de outros componentes do sistema, isto é conhecido como teste de:

a) módulo.
b) aceitação.
c) subsistema.
d) unidade.
e) sistema.

Comentários:

Quando os componentes são testados individualmente, estamos realizando um teste de unidade (também chamado de teste de componente/módulo). Ou seja, é focalizado o esforço na menor unidade do projeto do software. A letra (a) poderia causar dúvida, no entanto, devemos marcar a alternativa "mais correta".

Gabarito: Letra D

---

13. **(FGV / AL-RO – 2018)** O teste de software que visa verificar que, por exemplo, a correção de uma falha (ou bug) não introduziu uma nova falha (ou bug), é o teste:

a) revisional.
b) de integração.
c) funcional.
d) de regressão.
e) de recuperação.

Comentários:

A questão pergunta pelo teste que identifica se uma correção não criou uma nova falha. Estamos falando do teste de regressão que faz a reexecução do mesmo subconjunto de testes que já foram executados para assegurar que as alterações não tenham propagado efeitos colaterais indesejados. Em suma, O teste de regressão ajuda a garantir que as alterações (devido ao teste ou por outras razões) não introduzam comportamento indesejado ou erros adicionais.

Gabarito: Letra D

---

14. **(FGV / MPE-AL – 2018)** Eduardo é o líder técnico do Sistema de Vendas de uma rede de farmácias. O sistema deve ser utilizado em mais de 40 unidades espalhadas por vários estados. O sistema entrou em produção e, já na primeira semana de uso, ficou muito lento e diversas

vezes indisponível para os operadores das lojas. Diante deste cenário, assinale a opção que indica a técnica de teste que foi negligenciada:

a) de fumaça.
b) funcional de limite.
c) de desempenho.
d) caixa-branca.
e) de análise de valor-limite.

Comentários:

Se o sistema estava lento e diversas vezes indisponível, pode-se inferir que houve negligência em relação aos testes de desempenho, que permitiriam avaliar o software com relação ao seu limite de processamento de dados, entre outros.

Gabarito: Letra C

---

15. (FGV / BANESTES – 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.

Comentários:

(a) Errado, os testes White box (caixa-branca) são testes diferentes do teste beta; (b) Errado, eles são diferentes; (c) Errado, eles não focam em pontos críticos; (d) Errado, os testes alfa é que são realizados no ambiente do desenvolvedor; (e) Correto, é como funciona o teste beta.

Gabarito: Letra E

---

16. (FGV / BANESTES – 2018) O termo "Alfa teste" caracteriza testes de software que:

a) empregam primordialmente técnicas conhecidas como "Black box";
b) são equivalentes aos testes conhecidos pelo termo "Beta teste";
c) focam em pontos como performance e confiabilidade;
d) são realizados em ambientes controlados pelo desenvolvedor;
e) são realizados por usuários externos, em condições de uso semelhantes às de produção.

Comentários:

(a) Errado, os testes de caixa-preta talvez estejam mais relacionados aos testes beta; (b) Errado, são testes diferentes; (c) Errado, os testes que focam em performance são os testes desempenho; (d) Correto, os testes alfa são realizados no ambiente do desenvolvedor; (e) Errado, os testes realizados por usuários externos são os testes beta.

Gabarito: Letra D

---

17. (FGV / COMPESA - 2018) Com relação à análise estática de código, considere as afirmativas a seguir.

I. É um tipo de teste de software.

II. Visa detectar e corrigir defeitos existentes em programas.

III. É capaz de detectar defeitos do tipo bad smell (termo que significa incorreções técnicas ou anomalias que não evitam o sistema de ser executado, mas causam efeitos inesperados durante a execução).

Está correto o que se afirma em:

- a) I, somente.
  b) II, somente.
  c) III, somente.
  d) I e II, somente.
  e) I, II e III.

Comentários:

(I) Errado. Análise estática de código-fonte – na minha visão – não é um tipo de teste. Trata-se de uma prática de verificação da qualidade do código-fonte de um software, logo eu divirjo da banca; (II) Correto; (III) Correto. Um Bad Smell ocorre quando o código foi mal projetado ou quando a implementação escolhida estava errada, apesar de não impedir a execução do software. De toda forma, eu acredito que essa questão não possui resposta correta.

Gabarito: Letra E

---

18. (FGV / IBGE - 2017) Testes devem ser realizados durante o desenvolvimento de um sistema computacional para garantir a qualidade e detectar falhas antes que ele seja disponibilizado para os seus usuários finais. Analise as afirmativas a seguir sobre tipos de teste de software:

I. Teste de caixa preta é uma técnica de teste onde o código-fonte e a estrutura interna do sistema são considerados para modelar os casos de teste.

II.Teste de regressão tem a função de verificar se alguma modificação recente causou algum efeito indesejado e certificar se o sistema ainda atende aos requisitos.

III.Teste de desempenho foca na experiência do usuário, ergonomia da interface e acesso às funcionalidades.

Está correto o que se afirma em:

a) somente I.
b) somente II.
c) somente III.
d) somente I e III.
e) I, II e III.

Comentários:

(I) Errado, isso seria um teste de caixa branca; (II) Correto; (III) Errado, isso seria um teste de usabilidade.

Gabarito: Letra B

19.(FGV / ALERJ – 2017) A atividade de teste de software contribui para revelar defeitos latentes nos programas. Em relação às técnicas de testes de software, é correto afirmar que:

a) testes de caixa branca têm por objetivo testar o código-fonte, testar cada linha de código possível, testar os fluxos básicos e os alternativos;

b) testes de regressão têm por objetivo verificar se o sistema se mantém funcionando de maneira satisfatória após longos e intensos períodos de uso;

c) todas as declarações internas do programa devem ser testadas pelo menos uma vez durante os testes funcionais;

d) testes de unidade se preocupam em exercitar o sistema além de sua carga máxima de projeto, até que ele falhe;

e) testes de usabilidade verificam se o software instala como planejado, em diferentes hardwares e sob diferentes condições.

Comentários:

(a) Correto, em testes caixa-branca, teoricamente você deve testar todo caminho possível através do código – na prática, isso é meio inviável. (b) Errado, isso é um teste de estabilidade; (c) Errado, em testes funcionais não se testam declarações internas; (d) Errado, isso é um teste de estresse; (e) Errado, isso é um teste de instalação.

Gabarito: Letra A

---

20. (FGV / IBGE – 2016) Os testes de aceitação são muitas vezes a última etapa de testes antes de implantar o software em produção. Seu objetivo maior é verificar se o software está apto para utilização por parte dos usuários finais, de acordo com os requisitos de implementação definidos. Há três estratégias de implementação de testes de aceitação: a aceitação formal, a aceitação informal (ou teste alfa) e o teste beta.

Com relação às três estratégias de implementação dos testes de aceitação, é correto afirmar que:

a) o teste de aceitação informal, ou teste alfa, é conduzido nas instalações do usuário final, geralmente sem a presença do desenvolvedor;

b) o teste beta é conduzido na instalação do desenvolvedor por um grupo representativo de usuários finais;

c) o teste de aceitação formal utiliza todo o conjunto de casos de teste aplicados durante o teste do sistema, para procurar novos problemas;

d) o teste beta é focado na busca de defeitos e seu progresso é facilmente medido;

e) o teste de aceitação formal pode ser realizado de forma automatizada.

Comentários:

(a) Errado. Teste Alfa é conduzido nas instalações do desenvolvedor; (b) Errado. Teste Beta é conduzido nas instalações do usuário final; (c) Errado. Ele costuma ser uma extensão do Teste de Sistema. No entanto, trata-se de um subconjunto dos testes que foram realizados no Teste de Sistema; (d) Errado. Teste Beta é mais um teste de aceitação de um produto de software do que um teste focado em buscar defeitos, além disso não é fácil medir seu progresso; (e) Correto. Ele pode – sim – ser realizado de forma automatizada.

Gabarito: Letra E

---

21. (FGV / IBGE – 2016) Trata-se de um teste que desconhece o conteúdo do código fonte. Nesse teste o componente testado é tratado como uma caixa preta: são fornecidos dados de entrada

e o resultado comparado com aquele esperado e previamente conhecido. Além disso, esse teste pode ser aplicado em diversas fases de teste. A questão retrata características do teste:

a) funcional;
b) de integração;
c) de desempenho;
d) de carga;
e) unitário.

Comentários:

Pela primeira frase dessa questão, já era possível descobrir: nos testes funcionais, ignoram-se detalhes de implementação – também chamados de testes caixa-preta.

Gabarito: Letra A

---

22. (FGV / Prefeitura de Paulínia - SP - 2016) A equipe de desenvolvimento de sistemas da empresa "Sistemas Unidos" está trabalhando em um software com a utilização do processo unificado. Seguindo essa metodologia, as equipes realizam diversas disciplinas ao longo do desenvolvimento, dentre as quais estão os testes. A partir deste momento, a equipe deverá avaliar como os módulos trabalham em conjunto.

A equipe estará realizando os testes do tipo:

a) aceitação.
b) unitário.
c) integração.
d) sistema.
e) regressão.

Comentários:

O teste que avalia como os módulos trabalham em conjunto é o teste de integração.

Gabarito: Letra C

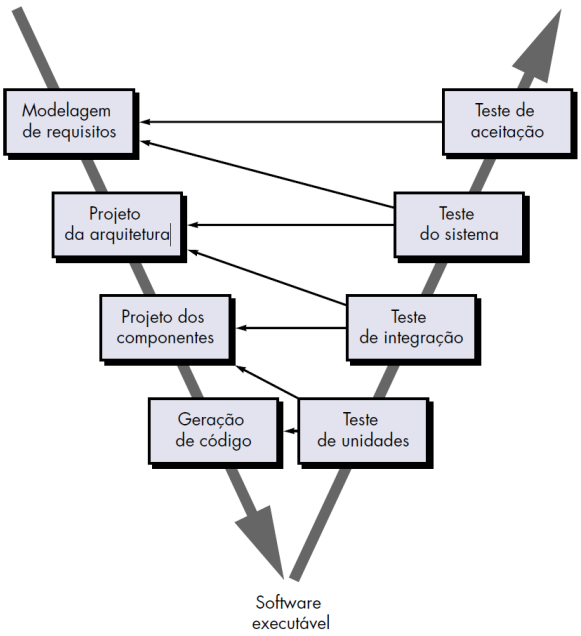
---

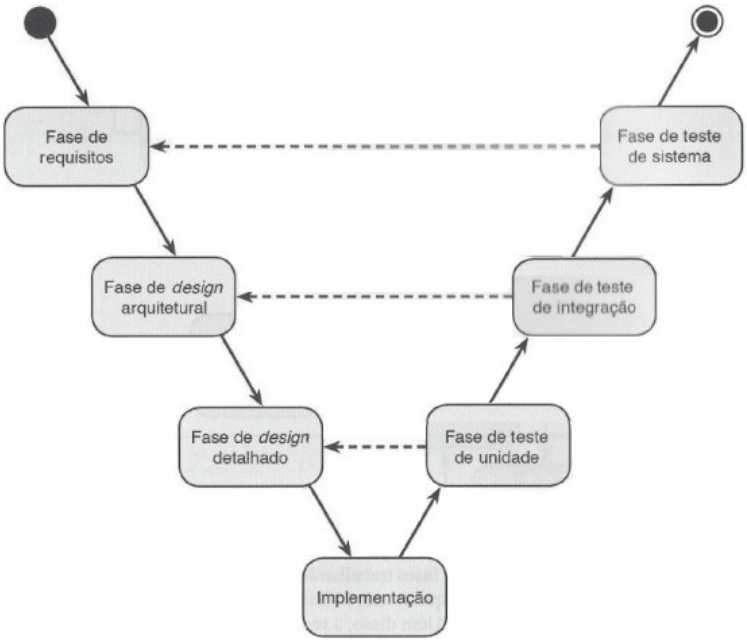
23. (FGV / TJ-PI - 2015) A equipe de desenvolvimento da empresa "Sosistemas" utiliza o modelo V para desenvolver seus sistemas de informação. Seguindo essa metodologia, as equipes realizam diversos tipos de testes ao longo do desenvolvimento. No momento atual, o funcionário José está testando um grupo de classes para avaliar seu funcionamento em conjunto. Para fazer essa avaliação, José está realizando testes do tipo:

a) unitário.

b) de integração.
c) de aceitação.
d) de segurança.
e) de carga.

Comentários:





O teste de um grupo de classes para avaliar seu funcionamento em conjunto é o Teste de Integração, isto é, como módulos (conjunto de classes que já foram testadas individualmente) funcionam quando estão integradas.

Gabarito: Letra B

---

24. (FGV / DPE-RJ - 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.

Comentários:

(a) Errado, esse seria o teste de desempenho; (b) Correto, o teste unitário realmente testa o menor bloco de software desenvolvido, avaliando os resultados obtidos com entradas de dados pré-definidos. A ideia é resolver um problema grande dividindo-o em partes menores; (c) Errado, esse seria o teste de integração; (d) Errado, essa até é uma função do teste unitário, mas não é sua função principal; (e) Errado, descrição meio estranha, mas parece ser um teste caixa-preta.

Gabarito: Letra B

---

25. (FGV / PROCEM-PA – 2014) A verificação dinâmica está baseada nas três dimensões de testes, listadas a seguir: tipos de teste, técnicas de teste e níveis de teste. Assinale a opção que apresenta somente itens da dimensão tipos de teste.

a) Teste de Aceitação – Teste de Regressão – Teste Estrutural
b) Teste de Funcionalidade – Teste de Desempenho – Teste de Unidade
c) Teste de Interface – Teste de Carga – Teste de Segurança
d) Teste Funcional – Teste de Volume – Teste de Sistema
e) Teste de Usabilidade – Teste de Funcionalidade – Teste de Integração

Comentários:

(a) Errado. Nível de Teste; Tipo de Teste; Técnica de Teste;
(b) Errado. Tipo de Teste; Tipo de Teste; Nível de Teste;
(c) Correto. Tipo de Teste; Tipo de Teste; Tipo de Teste;
(d) Errado. Técnica de Teste; Tipo de Teste; Nível de Teste;
(e) Errado. Tipo de Teste; Tipo de Teste; Nível de Teste;

Gabarito: Letra C

---

26. (FGV / FIOCRUZ – 2010) Um tipo de teste de sistemas de software é também chamado de "teste comportamental" e focaliza os requisitos funcionais do software, permitindo ao engenheiro de software derivar conjuntos de condições de entrada que vão exercitar plenamente todos os requisitos funcionais de um programa. Esse tipo de teste tende a ser aplicado durante os últimos estágios do teste e tenta encontrar erros em funções incorretas ou omitidas, de interfaces, de estrutura de dados ou de acesso à base de dados externa, de comportamento ou desempenho de iniciação e término. Além disso, é um tipo de teste que despreza, de propósito, a estrutura de controle, sendo a atenção focalizada no domínio da informação. Esse tipo é conhecido por teste:

a) caixa-preta.
b) caixa-branca.
c) de fluxo de dados.
d) de caminho básico.

e) de lógica composta.

Comentários:

Pela primeira frase dessa questão, já era possível descobrir: testes comportamentais são os testes caixa-preta.

Gabarito: Letra A

---

27. (FGV / BADESC – 2010) O teste de software que projeta casos de testes derivados do conhecimento da estrutura e da implementação do software é conhecido por:

a) teste de releases.
b) teste caixa-claro.
c) teste caixa-preta.
d) teste de aceitação.
e) teste de integração.

Comentários:

Projeto casos de testes **derivados do conhecimento da estrutura e da implementação** do software é o famoso teste caixa-branca ou teste caixa-clara.

Gabarito: Letra B

QUESTÕES COMENTADAS – DIVERSAS BANCAS

1. **(CESGRANRIO / IPEA – 2024)** Uma desenvolvedora foi contratada para a equipe de desenvolvimento de uma empresa e teve, como primeira tarefa, estudar sobre stub. Ao pesquisar sobre o assunto, ela encontrou uma boa definição para esse termo, que explicava que stub é(são):

a) a implementação real, mas não necessariamente igual à implementação que estará no ambiente de produção.

b) a implementação que permite fornecer respostas prontas, sendo usada nas situações em que se deseja validar apenas o resultado.

c) a técnica que permite criar métodos com o mesmo nome em uma mesma classe, e o que varia entre os métodos escritos são os tipos de informações que poderão receber em seus parâmetros.

d) os objetos fornecidos, mas não utilizados, sendo geralmente usados no preenchimento da lista de parâmetros.

e) os objetos pré-programados que verificam se um ou mais métodos foram ou não chamados, a ordem de chamadas, se foram chamados com os argumentos certos e quantas vezes foram chamados.

Comentários:

(a) Errado. A implementação real, mas não necessariamente igual à implementação que estará no ambiente de produção, descreve uma abordagem de desenvolvimento, mas não define um stub;

(b) Correto. Stubs são usados para fornecer respostas prontas em situações de teste, permitindo validar o comportamento de partes do sistema sem envolver suas dependências reais;

(c) Errado. Essa definição descreve o conceito de sobrecarga de métodos, não de stubs;

(d) Errado. Objetos fornecidos mas não utilizados, geralmente para preencher parâmetros, são mais bem descritos como objetos "dummy" ou falsos;

(e) Errado. Objetos que verificam chamadas de métodos, a ordem, argumentos e frequência são conhecidos como mocks, não stubs.

Gabarito: Letra B

2. **(CESGRANRIO / TRANSPETRO – 2023)** Uma equipe de desenvolvimento pretende convidar um conjunto de representantes dos usuários finais de uma solução, que aparentemente está completamente desenvolvida, para testar sua última versão antes de disponibilizá-la no mercado. Caso essa versão seja aprovada, será colocada em produção. A situação descrita faz referência aos testes:

a) unitários
b) de aceitação
c) de desempenho
d) de integração
e) de sistema

Comentários:

(a) Errado. Testes unitários são realizados pelos desenvolvedores para verificar se partes individuais do código (unidades) funcionam corretamente;

(b) Correto. Testes de aceitação envolvem representantes dos usuários finais para garantir que a solução atende aos requisitos e expectativas antes de ser colocada em produção;

(c) Errado. Testes de desempenho avaliam a resposta e estabilidade do sistema sob cargas específicas, mas não envolvem diretamente os usuários finais para validação antes da produção;

(d) Errado. Testes de integração verificam a interação entre diferentes módulos do sistema, mas não envolvem diretamente os usuários finais;

(e) Errado. Testes de sistema verificam o sistema completo como um todo, mas não necessariamente envolvem a participação dos usuários finais para aprovação antes da produção.

Gabarito: Letra B

---

3. **(CESGRANRIO / TRANSPETRO – 2023)** O principal objetivo dos testes estáticos é o de reduzir os defeitos de um software por meio da redução de defeitos na documentação a partir da qual o software foi desenvolvido. Uma das técnicas mais importantes para a execução de testes estáticos é a de:

a) regressão
b) walk-through
c) análise do valor limite
d) análise de casos de uso
e) particionamento de equivalência

Comentários:

(a) Errado. Testes de regressão são testes dinâmicos realizados para garantir que modificações no código não introduzam novos defeitos;

(b) Correto. Walk-through é uma técnica de teste estático onde o autor do documento apresenta o material para os colegas, que fazem perguntas e comentários para identificar defeitos;

(c) Errado. Análise do valor limite é uma técnica de teste dinâmico usada para identificar defeitos nas extremidades dos intervalos de entrada;

(d) Errado. Análise de casos de uso é uma técnica usada para identificar requisitos e cenários de uso, não para testes estáticos;

(e) Errado. Particionamento de equivalência é uma técnica de teste dinâmico usada para reduzir o número de casos de teste, identificando grupos de entradas que devem ser tratadas da mesma forma.

Gabarito: Letra B

---

4. **(CESGRANRIO / IPEA – 2024)** Uma nova funcionalidade acabou de ser desenvolvida para um software que tem sido usado há anos por uma empresa. A líder da equipe de desenvolvimento informou à equipe a relevância de executar um conjunto de testes a toda nova versão desse software, a fim de garantir que mudanças realizadas nas novas versões não impactem o restante do sistema. Com base no cenário descrito, qual(is) teste(s) valida(m) se o que foi criado em versões anteriores desse mesmo software continua funcionando a partir de mudanças em suas outras funcionalidades?

a) Carga
b) Stress
c) Regressão
d) Portabilidade
e) Não funcionais

Comentários:

(a) Errado. Testes de carga avaliam o desempenho do sistema sob condições normais de uso, não se focam em verificar se funcionalidades anteriores continuam funcionando;

(b) Errado. Testes de stress avaliam como o sistema se comporta sob condições extremas, não verificam a integridade de funcionalidades anteriores;

(c) Correto. Testes de regressão validam se funcionalidades previamente desenvolvidas continuam funcionando corretamente após alterações no software;

(d) Errado. Testes de portabilidade avaliam se o software pode ser executado em diferentes ambientes e plataformas, mas não verificam a continuidade das funcionalidades antigas;

(e) Errado. Testes não funcionais avaliam aspectos como desempenho, segurança e usabilidade, mas não se concentram especificamente na integridade de funcionalidades antigas.

Gabarito: Letra C

---

5. **(CESGRANRIO / IPEA – 2024)** Um desenvolvedor de sistemas, ao analisar algumas ferramentas de testes, deparou-se com o xUnit, o qual, dentre outras características, destaca-se por:

a) suportar a criação de testes unitários e de testes de interface para as linguagens .NET e Swift.
b) ser código fechado e ser voltado a criar testes unitários para as linguagens orientadas a objetos.
c) ser código aberto e permitir a criação de testes unitários para linguagens .NET.
d) ser uma biblioteca gratuita que permite a criação de testes unitários para aplicações desenvolvidas em Java e Swift.
e) ser uma biblioteca gratuita e de código fechado que permite a criação de testes unitários e funcionais para diversas linguagens, como, por exemplo, C# e F#.

Comentários:

(a) Errado. xUnit suporta principalmente testes unitários e é mais associado ao ecossistema .NET, não Swift, e não é focado em testes de interface;

(b) Errado. xUnit é de código aberto, não fechado, e é voltado principalmente para testes unitários no ecossistema .NET;

(c) Correto. xUnit é uma ferramenta de código aberto que permite a criação de testes unitários principalmente para linguagens .NET;

(d) Errado. xUnit não é usado para aplicações Java e Swift; ele é especificamente para .NET;

(e) Errado. xUnit é de código aberto e gratuito, mas não suporta diversas linguagens como C# e F#, focando mais em .NET.

Gabarito: Letra C

---

6. **(CESGRANRIO / IPEA – 2024)** Em projetos de desenvolvimento de softwares, é importante que eles sejam testados em diferentes situações comuns de ocorrerem, de modo a contribuir para que eles atinjam a qualidade esperada. Por isso, nesses projetos, quatro níveis de teste são

importantes de serem considerados, a fim de ajudar a lidar com tais situações. Esses quatro níveis de teste são os seguintes:

a) acessibilidade, usabilidade, unidade e sistema
b) desempenho, funcionais, componente e aceitação
c) funcionais, não funcionais, componente e desempenho
d) funcionais, não funcionais, usabilidade e acessibilidade
e) unidade, integração, sistema e aceitação

Comentários:

(a) Errado. Acessibilidade e usabilidade são tipos de testes não funcionais, não níveis de teste;

(b) Errado. Desempenho e funcionais são tipos de teste, enquanto componente e aceitação são níveis, mas esta combinação não cobre todos os quatro níveis principais;

(c) Errado. Funcionais e não funcionais são categorias de teste, não níveis. Componente e desempenho também são tipos de teste e não níveis completos;

(d) Errado. Funcionais e não funcionais são categorias de teste. Usabilidade e acessibilidade são tipos de teste não funcionais;

(e) Correto. Os quatro níveis de teste mais importantes em projetos de desenvolvimento de software são unidade, integração, sistema e aceitação. Esses níveis garantem uma cobertura abrangente desde os componentes menores até a validação final pelo usuário.

Gabarito: Letra E

---

7. **(CESGRANRIO / IPEA – 2024)** Considere que um novo software foi desenvolvido e está prestes a entrar no ambiente de produção de uma empresa, mas, antes disso, serão realizados testes finais. Para isso, um conjunto de representantes dos usuários finais deve participar desse estágio de testes. Caso se perceba que o software está tendo o comportamento esperado, ele será implantado em produção. Qual estágio de teste está descrito no cenário acima?

a) Aceitação
b) Componente
c) Configuração
d) Desempenho
e) Usabilidade

Comentários:

(a) Correto. O teste de aceitação envolve representantes dos usuários finais para validar se o software atende aos requisitos e tem o comportamento esperado antes de ser implantado em produção;

(b) Errado. O teste de componente verifica unidades isoladas do software, sem envolver usuários finais;

(c) Errado. O teste de configuração verifica se o software funciona corretamente em diferentes ambientes e configurações, mas não envolve diretamente a validação por usuários finais;

(d) Errado. O teste de desempenho avalia como o software se comporta sob carga e condições de estresse, mas não foca na validação funcional com usuários finais;

(e) Errado. O teste de usabilidade avalia a facilidade de uso do software, envolvendo usuários finais, mas o cenário descrito foca na validação funcional geral antes da implantação.

Gabarito: Letra A

---

8. **(CESGRANRIO / AGERIO – 2023)** Ao planejar um projeto de sistema seguindo um ciclo de vida linear, um gerente de projeto resolveu instituir uma estratégia global de teste de software.

   Considerando-se uma ordem do mais específico para o mais geral, ou seja, terminando-se com o teste de ordem superior, qual a ordem dos testes a serem realizados?

   a) Teste de integração, teste de validação, teste de sistema, teste de unidade
   b) Teste de sistema, teste de validação, teste de unidade, teste de integração
   c) Teste de validação, teste de integração, teste de unidade, teste de sistema
   d) Teste de validação, teste de sistema, teste de unidade, teste de integração
   e) Teste de unidade, teste de integração, teste de validação, teste de sistema.

Comentários:

(a) Errado. A ordem dos testes está incorreta, pois começa com o teste de integração e termina com o teste de unidade, o que não segue a sequência do mais específico para o mais geral.

(b) Errado. A ordem começa com o teste de sistema e termina com o teste de integração, o que também não segue a sequência correta.

(c) Errado. A ordem começa com o teste de validação, o que é um teste de nível superior, enquanto o teste de unidade, que é o mais específico, deveria ser o primeiro.

(d) Errado. A ordem começa com o teste de validação, que é mais geral, e inclui o teste de unidade e de integração fora de ordem.

(e) Correto. A sequência correta dos testes, do mais específico para o mais geral, é: Teste de unidade (testa componentes individuais) > Teste de integração (verifica a interação entre componentes) > Teste de validação (assegura que o sistema atende às necessidades do cliente) > Teste de Sistema (avalia o sistema completo em seu ambiente).

Gabarito: Letra E

---

9. **(CESGRANRIO / Caixa – 2024)** Em um contrato para o desenvolvimento de um software de gestão empresarial, uma cláusula específica solicita a execução, pelo fornecedor, de um "teste alfa" antes da entrega do produto ao cliente. Para atender a essa cláusula do contrato, o fornecedor deve:

a) realizar uma série de testes que visa garantir que mudanças recentes no código não afetem as funcionalidades existentes do software, mantendo a integridade do sistema após atualizações ou correções.

b) distribuir o software para um grupo externo de usuários para que estes o utilizem em condições reais e forneçam feedback sobre a experiência antes de distribuir amplamente.

c) verificar a comunicação e o funcionamento adequados entre diferentes módulos ou componentes do software, assegurando que eles trabalhem juntos conforme esperado.

d) testar individualmente os menores pedaços de código do software, como funções ou métodos, para garantir que funcionem corretamente.

e) avaliar o software com uma equipe interna que simula o comportamento do usuário final, buscando identificar falhas antes da liberação para usuários externos.

Comentários:

(a) Errado. Esse tipo de teste é conhecido como teste de regressão, que visa garantir a integridade do sistema após atualizações ou correções;

(b) Errado. Distribuir o software para um grupo externo de usuários para feedback é característico de um teste beta, não alfa;

(c) Errado. Verificar a comunicação entre diferentes módulos do software corresponde a testes de integração, não ao teste alfa;

(d) Errado. Testar individualmente menores pedaços de código refere-se a testes unitários, não ao teste alfa;

(e) Correto. O teste alfa envolve uma equipe interna simulando o comportamento do usuário final para identificar falhas antes da liberação para usuários externos.

Gabarito: Letra E

---

10. (VUNESP / TCM-SP – 2023) Dentre as várias estratégias de teste de software, há uma delas que estabelece que quando da realização de testes de integração de módulos de software, alguns testes são executados novamente de modo a verificar se a adição de novos módulos não tenha provocado erros até então inexistentes. A essa técnica atribui-se a denominação de teste:

a) adaptativo.
b) principal.
c) de regressão.
d) secundário.
e) estratégico.

Comentários:

O teste de regressão é a técnica utilizada para verificar se as adições ou modificações de novos módulos no software não introduziram novos erros ou reativaram problemas anteriores. Ele é executado repetidamente após alterações no código para garantir que o sistema ainda funciona como esperado. Quanto às outras alternativas, todas elas foram inventadas pelo examinador e não existem no contexto de testes de software.

Gabarito: Letra C

Figura 12 – Tela de entrada de dados

11. (FUNDATEC / ISS-Porto Alegre – 2022) Sabe-se que a equipe responsável pelo desenvolvimento da funcionalidade Agendar Atendimento, do software Sistema de Atendimento Agendado (SAA), realizou testes intensivos, com o objetivo de entregar tal funcionalidade estável e sem erros. A equipe de testes, antes de iniciar suas atividades, estudou os artefatos elaborados no projeto, tais como documento visão, diagramas e especificações de

casos de uso, histórias de usuário, casos de teste, regras de negócio, modelo de dados, lista de mensagens, tipos de dados e valores válidos de entrada e saída, dentre outros. Os testadores realizaram muitas simulações, inserindo, na tela de entrada de dados, dados certos e errados, de modo a observar o comportamento do software e as correspondentes saídas de dados. Não cabia a essa equipe realizar testes na arquitetura do software e nem a validação de algoritmos, linguagem de programação ou quaisquer outras estruturas de dados, dessa funcionalidade. Nesse caso, pode-se afirmar que a equipe realizou o seguinte tipo de testes de software:

a) Teste unitário.
b) Teste de stress.
c) Teste de regressão.
d) Teste de caixa preta.
e) Teste de caixa branca.

Comentários:

"Não cabia a essa equipe realizar testes na arquitetura do software e nem a validação de algoritmos, linguagem de programação ou quaisquer outras estruturas de dados, dessa funcionalidade".

Essa é a dica para matar a questão: se não se avalia o funcionamento interno do software, apenas o seu "*comportamento e as correspondentes saídas de dados*", está na cara que é um Teste Caixa Preta.

Gabarito: Letra D

---

12. (IDHTEC / PREFEITURA DE MARAGOGI-AL – 2019) Engenharia de software é uma abordagem sistemática e disciplinada para o desenvolvimento de software (PRESSMAN, 2006). Considere V para afirmativa verdadeira e F para falsa:

( ) Análise de requisito - Através da análise de requisito é o momento onde efetua a programação do código fonte para desenvolver o software (JALOTE, 2005).

( ) Design do software - Pelo design do software é o momento que o engenheiro de software realiza o planejamento da solução do problema que foi levantado no documento de requisito (JALOTE, 2005).

( ) Codificação - A codificação é o momento que criptografa e transformará em uma linguagem de programação (JALOTE, 2005).

( ) Teste - O teste de software é o processo que tem a intenção de encontrar defeitos nos artefatos de software (MYERS, 2004). O teste é uma maneira de medir o controle da qualidade do software durante o desenvolvimento de software (JALOTE, 2005).

A sequência correta, de cima para baixo, é:

a) F, V, V, V
b) V, F, V, F.
c) F, V, F, F.
d) V, F, F, F.
e) F, V, F, V.

Comentários:

(F) Errado, a programação do código ocorre na etapa de codificação, na análise de requisitos ocorre o refinamento dos requisitos obtidos na etapa de levantamento de requisitos; (V) Correto, trata-se da fase de elaboração, ela busca como satisfazer os requisitos que foram estabelecidos anteriormente; (F) Errado, a codificação é o momento em que se transforma o projeto em uma linguagem de programação; (V) Correto, trata-se de uma definição de testes de software.

Gabarito: Letra E

---

13. **(IDHTEC / PREFEITURA DE MARAGOGI-AL – 2019)** Quanto à Automação de Testes, julgue os itens a seguir:

- I. Ferramentas de automação não possuem outros usos, além da medição de performance de aplicações. Elas também não podem ser usadas para preparar um ambiente de teste com um grande volume de dados.

II. No teste de interface gráfica, uma plataforma gera os eventos de entrada na interface de utilizador do sistema e observa as mudanças na saída.

III. No teste baseado em código, a interface pública das classes, módulos ou bibliotecas são testadas com uma variedade de argumentos de entrada, observando-se a saída.

Está (estão) correto(s):

a) Somente I.
b) Somente I e II.
c) Somente II e III.
d) I, II e III.
e) Somente II.

Comentários:

(I) Errado, ferramentas de testes automatizados podem ser – sim – utilizadas para grande volume de dados; (II) Correto, para testar uma interface gráfica devemos gerar eventos de entrada e

observar as saídas; (III) Correto, o teste baseado em código testa os elementos presentes no código-fonte.

Gabarito: Letra C

---

14. (IDHTEC / PREFEITURA DE MARAGOGI-AL – 2019) Sobre testes, analise as afirmativas a seguir:

I. Teste de regressão corresponde a um nível de teste, mas não 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.

II. A técnica de teste de Estrutural é recomendada para os níveis de Teste da Unidade e Teste da Integração, cuja responsabilidade principal fica a cargo dos desenvolvedores do software, que são profissionais que conhecem bem o código-fonte desenvolvido e dessa forma conseguem planejar os casos de teste com maior facilidade.

III. Teste Funcional é a Técnica de teste em que o componente de software a ser testado é abordado como se fosse uma caixa-preta, ou seja, não se considera o comportamento interno do mesmo.

Está (estão) correta(s):

a) Somente I.
b) Somente I e II.
c) Somente II e III.
d) I, II e III.
e) Somente II.

Comentários:

(I) Errado, na verdade, os testes de regressão são uma importante estratégia para redução dos efeitos colaterais, uma vez que o objetivo deles é a reexecução dos testes que já foram executados; (II) Correto, trata-se do teste de caixa-branca; (III) Correto, trata-se do teste de caixa-preta.

Gabarito: Letra C

---

15. (IDHTEC / PREFEITURA DE MARAGOGI-AL – 2019) Com relação aos testes realizados no processo de desenvolvimento de software, julgue as afirmativas a seguir:

I. Diversas atividades de testes são executadas a fim de se validar o produto de software, testando cada funcionalidade de cada módulo, buscando, levando em consideração a especificação feita na fase de projeto.

II. Na fase de Testes de Integração as unidades do sistema são testados de forma combinada, o objetivo é detectar falhas na interação entre as unidades integradas.

III. Na fase de Testes de Verificação de Unidade serão testados apenas os módulos das funcionalidades requeridas pelo cliente durante o projeto, garantindo o pleno funcionamento. Deve ser feito, preferencialmente, pelo usuário final.

Está (estão) correta(s):

a) Apenas a afirmativa I.
b) Apenas a afirmativa II.
c) Apenas a afirmativa III.
d) Apenas as afirmativas I e II.
e) As afirmativas I, II e III.

Comentários:

(I) Correto, é como funcionam os testes de software; (II) Correto, os testes de integração verificam o funcionamento em conjunto dos componentes do sistema; (III) Errado, os testes de unidade são realizados pelos desenvolvedores, e não pelos usuários finais.

Gabarito: Letra D

---

16. (AVANÇA SP / CÂMARA MUNICIPAL DE TABOÃO DA SERRA-SP – 2019) No que se refere às técnicas de teste de software, há os testes conhecidos como "caixa preta" e "caixa branca". Sobre o tema, analise os itens a seguir e, ao final, assinale a alternativa correta:

I – Testes do tipo "caixa branca" são realizados apenas após o software estar completamente integrado.

II – Testes do tipo "caixa preta" não são aplicáveis a software de pequeno porte.

III – Testes do tipo "caixa preta" tem a finalidade de exercitar as interfaces do software sob teste.

a) Apenas o item I é verdadeiro.
b) Apenas o item II é verdadeiro.
c) Apenas o item III é verdadeiro.
d) Apenas os itens I e II são verdadeiros.
e) Todos os itens são verdadeiros.

Comentários:

(I) Errado, testes do tipo "caixa branca" podem ser realizados a qualquer momento; (II) Errado, testes do tipo "caixa preta" podem ser aplicados a qualquer tipo de software; (III) Correto, os testes do tipo "caixa preta" têm o objetivo de verificar a funcionalidade e aderência aos requisitos, em uma ótica externa ou do usuário. Ele é baseado apenas nas interfaces dos softwares, sem se basear em qualquer conhecimento do código ou lógica interna do componente de software.

Gabarito: Letra C

---

17. (FUNDEP / PREFEITURA DE LAGOA SANTA-MG – 2019) Assinale a associação correta presente na tabela ASSOCIAÇÕES que define corretamente os elementos a definir da TABELA A com as definições ou caracterizações da TABELA B.

TABELA A	
	A definir
1	Os processos de software são
2	Modelos de processos de software
3	Modelos gerais de processo
4	Engenharia de requisitos
5	Validação de software

TABELA B	
	Definição ou caracterização
A	Modelos de processos de software
B	as atividades envolvidas na produção de um sistema de software
C	é o processo de desenvolvimento de uma especificação de software
D	descrevem a organização dos processos de software
E	é o processo de verificação de que o sistema está de acordo com sua especificação e satisfaz às necessidades reais dos usuários do sistema.

a)

TABELA A	TABELA B
1	B
2	A
3	C
4	D
5	E

b)

TABELA A	TABELA B
1	B
2	A
3	D
4	C
5	E

c)

TABELA A	TABELA B
1	C
2	A
3	B
4	E
5	D

d)

TABELA A	TABELA B
1	C
2	B
3	E
4	D
5	A

Comentários:

(1-B) Os processos de software são as atividades envolvidas na produção de um sistema de software; (2-A) Os modelos de processo de software são também chamados de paradigmas de engenharia de software e cada um deles representa uma tentativa de colocar em ordem uma atividade caótica, temos como exemplo: como o cascata, incremental, evolutivo, entre outros; (3-D) Os modelos gerais descrevem a organização dos processos de software; (4-C) A engenharia de requisitos é o processo de desenvolvimento de uma especificação de software, em outras palavras é uma abordagem sistemática para a formulação, análise, documentação e manutenção de requisitos de um sistema; (5-E) Validação de software é o processo de verificação de que o sistema está de acordo com sua especificação e satisfaz às necessidades reais dos usuários do sistema

Gabarito: Letra B

---

18. (IESES / SCGÁS – 2019) Identifique a alternativa que descreve inequivocamente a intenção do teste de verificação de software ou, mais genericamente, verificação e validação (V&V):

a) O teste de verificação de software tem a intenção de mostrar que um software se adequa às suas especificações ao mesmo tempo que satisfaz as especificações do cliente do sistema.

b) O teste de verificação de software tem a intenção de demostrar que um software se adequa às suas especificações ao mesmo tempo que satisfaz as especificações do mercado concorrente.

c) É um processo que verifica a atualização de versão de sistema. Tem relação com softwares básicos.

d) O teste de verificação procura verificar similaridades entre softwares para buscar encontrá-los.

Comentários:

De acordo com Sommerville, verificação e validação (V&V), tem a intenção de mostrar que um software se adequa a suas especificações ao mesmo tempo que satisfaz as especificações do cliente do sistema. Além disso, o autor afirma que o teste de programa é a principal técnica de validação e que a validação também pode envolver processos de verificação, como inspeções e revisões, em cada estágio do processo de software

Gabarito: Letra A

---

19. (IESES / SCGÁS – 2019) Assinale a alternativa correta que apresenta a diferença entre teste de defeito e debugging:

a) Testes de defeito estabelecem a existência de defeitos já o debugging diz respeito à localização e correção desses defeitos.

b) Não há diferença entre os dois. Teste de defeito e debugging são a mesma coisa.
c) O debugging somente pode ser feito depois do teste de defeito.
d) O teste de defeito localiza e correção do defeito enquanto o debugging diz que há erro.

Comentários:

Os testes de defeito expõem defeitos que causam funcionamentos incorretos. O termo bug está relacionado a um erro no software, já o termo debugging trata da localização e correção desses erros.

Gabarito: Letra A

---

20. (IBADE / PREFEITURA DE VILHENA-RO – 2019) Sobre teste de software, podemos diferenciar teste da caixa preta e teste da caixa branca respectivamente da seguinte maneira:

a) enquanto o primeiro ignora o código fonte, o segundo busca garantir que os componentes do software estejam concisos.

b) no primeiro são gerados dados aleatórios para teste, enquanto no segundo os dados são passados manualmente.

c) no primeiro o teste é feito de forma obscura sem saber o que o programa faz, enquanto que no segundo é feito de forma clara, analisando as entradas e saídas esperadas pelo programa.

d) enquanto que no primeiro há a necessidade de conhecer a estrutura do código, no segundo há apenas a necessidade de saber o resultado esperado para cada entrada de dados.

e) enquanto que no primeiro teste não se conhece a entrada de dados, mas se sabe a saída esperada, no segundo não se sabe a saída esperada, mas se conhece as entradas de dados possíveis.

Comentários:

O teste de caixa-preta não tem acesso às estruturas internas (código-fonte), já o teste de caixa-branca consegue acessar os componentes internos, de forma que esses funcionem corretamente.

Gabarito: Letra A

---

21. (INSTITUTO AOCP / UFPB – 2019) Os testes de software são realizados para verificar se um programa realmente faz o que é proposto a fazer e de forma correta, assim é possível descobrir os defeitos do programa antes dele ser utilizado pelo usuário final. Os testes são realizados utilizando dados fictícios em busca de erros e anomalias. Existem diversos tipos de testes.

Assinale a alternativa que apresenta as características dos testes unitários de desenvolvimento de software:

a) Testes em que os potenciais usuários de um sistema testam o sistema em seu próprio ambiente.
b) Testes que são centrados nas interfaces dos componentes.
c) Testes que não têm como objetivo verificar a funcionalidade de objetos e métodos.
d) Testes que são centrados nas interações entre os componentes.
e) Testes em que as unidades individuais de programa ou classes de objetos são testadas individualmente.

Comentários:

(a) Errado, o teste em que os usuários testam o sistema em seu próprio ambiente é o teste de caixa-preta (também chamado de testes de funcionais); (b) Errado, trata-se dos testes de usabilidade; (c) Errado, trata-se dos testes de caixa-preta em que o objetivo é testar se o software funciona ou não, ademais, nos testes de caixa-preta não há acesso às estruturas internas de um software. (d) Errado, trata-se dos de testes de Integração que são caracterizados por testar as interfaces entre os componentes ou interações de diferentes partes de um sistema; (e) Correto, por isso são chamados de testes unitários.

Gabarito: Letra E

---

22. **(INSTITUTO AOCP / IBGE – 2019)** Para chegar a um nível de perfeição de um software, é necessário aplicar muitos testes, sendo que o teste de integração é um dos mais importantes. Considerando o exposto, assinale a alternativa que NÃO apresenta uma característica dos testes de integração de software:

a) Testar as dependências entre os componentes.
b) Testar as interfaces entre as unidades.
c) Simular módulos ainda não implementados que se comunicam ao módulo testado.
d) Testar conformidade com a especificação dos requisitos.
e) Realizar teste estrutural ou caixa-branca.

Comentários:

(a) Errado, o teste de integração testa as interfaces entre os componentes; (b) Errado, é uma característica dos testes de integração; (c) Errado, os testes de integração testam a comunicação entre os módulos; (d) Correto, os testes de integração não têm o objetivo de testar a conformidade com a especificação de requisitos, mas sim a integração entre os módulos; (e) Errado, os testes de integração podem realizar testes de caixa-branca.

Gabarito: Letra D

---

23. (INSTITUTO AOCP / IBGE – 2019) A respeito dos testes de aceitação, analise as assertivas e assinale a alternativa que aponta(s) as correta(s).

I. É um teste que isenta de responsabilidades os usuários finais ou clientes.

II. O propósito do teste não é somente encontrar erros no software mas também erros de instalação do software.

III. É um teste em que o analista deve executar um processo de comparação dos requisitos iniciais do software e das necessidades atuais dos usuários finais.

a) Apenas I.
b) Apenas II.
c) Apenas III.
d) Apenas I e II.
e) Apenas II e III.

Comentários:

(I) Errado, na verdade, em um teste de aceitação é necessária a validação dos clientes; (II) Errado, o propósito de um teste de aceitação é encontrar defeitos e inconsistências com relação a sua especificação de requisitos, ou seja, não há relação com a instalação do software; (III) Correto, é exatamente isso que um teste de aceitação faz.

Gabarito: Letra C

---

24. (INSTITUTO AOCP / IBGE – 2019) Um analista de sistemas do IBGE necessita realizar um teste em um software. Durante o teste, o analista teve como objetivo não se preocupar com o comportamento interno do software e suas estruturas. Ao invés disso, ele se concentrou em encontrar as circunstâncias pelas quais o software não se comportava em conformidade com os seus requisitos. Diante desse cenário, assinale a alternativa que apresenta corretamente o nome do teste realizado pelo analista do IBGE:

a) Teste de caixa preta.
b) Teste de integração.
c) Teste ágil.
d) Teste de aplicação.
e) Teste de funcionalidade.

Comentários:

O teste que não se preocupa com as estruturas internas do software é o teste de caixa-preta.

Gabarito: Letra A

---

25. (QUADRIX / CRO-GO – 2019) Análise de requisitos, implementação e testes são alguns dos processos que fazem parte do desenvolvimento de sistemas orientados a objetos.

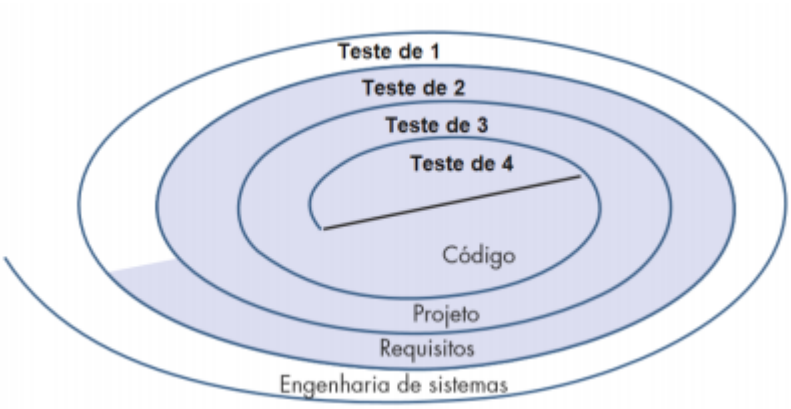
Comentários:

Apesar de existir divergências entre os autores sobre as etapas do desenvolvimento de software, a análise de requisitos a implementação e os testes são etapas fundamentais no processo de desenvolvimento de um software.

Gabarito: Correto

---

26. (COLÉGIO PEDRO II – 2019) Teste é um conjunto de atividades que podem ser planejadas com antecedência e executadas sistematicamente. Deverá ser definido, para o processo de software, um conjunto de etapas nas quais podem-se empregar técnicas específicas de projeto de caso de teste e métodos de teste. O processo de software pode ser visto como a espiral ilustrada na figura a seguir. Inicialmente, a engenharia de sistemas define o papel do software e passa à análise dos requisitos de software, na qual são estabelecidos o domínio da informação, função, comportamento, desempenho, restrições e critérios de validação para o software. Deslocando-se para o interior da espiral, chega-se ao projeto e, finalmente, à codificação.



Uma estratégia para teste de software também pode ser vista no conceito da espiral, como na figura, correlacionando o modelo de teste adotado à fase na qual o software se encontra. A alternativa que corresponde corretamente às respectivas fases de teste numeradas na figura como 1, 2, 3 e 4 é:

a) Recuperação – Esforço – Segurança – Desempenho.
b) Sistema – Verificação – Validação – Depuração.
c) Sistema – Entrevista – Controle – Codificação.
d) Sistema – Validação – Integração – Unidade.

Comentários:

A abordagem em espiral pode ser compreendida da seguinte maneira: no sentido anti-horário temos a visão do processo de software e no sentido horário temos a visão dos testes de software, que é o que a questão pergunta. Desse modo, os testes de software se iniciam pelo teste de unidade, em seguida temos os testes de integração, os testes de validação e os testes de sistema. Observem que a questão pergunta do final do espiral para o começo.

Gabarito: Letra D

---

27. (COVEST-COPSET / UFPE – 2019) A respeito de princípios básicos para elaboração de testes de software, assinale a alternativa correta.

a) A definição da saída ou resultado esperado é uma parte desnecessária em um caso de teste.
b) Testes de software são tarefas repetitivas, pouco criativas ou desafiadoras, pois são fáceis de automatizar quando pensadas no início do projeto.
c) Se muitos erros já foram encontrados em uma seção do programa, a probabilidade de encontrar mais erros ali será baixa.
d) Casos de teste devem ser escritos para condições de entrada inválidas e inesperadas, bem como para aquelas que são válidas e esperadas.
e) Ao se planejar um esforço de testes, faz-se a suposição tácita de que nenhum erro será encontrado.

Comentários:

(a) Errado, as entradas e saídas são importantes nos testes de software; (b) Errado, os testes de software devem ser sempre inovados, não sendo tarefas repetitivas; (c) Errado, se muitos erros foram encontrados há maior chance de encontrar outros erros; (d) Correto, há de se pensar em todas as possibilidades possíveis de casos de teste; (e) Errado, busca-se sempre encontrar erros.

Gabarito: Letra D

---

28. (COVEST-COPSET / UFPE – 2019) No contexto de diferentes técnicas de teste de caixa preta (black-box) e caixa branca (white-box), assinale a alternativa correta:

a) A análise de valor limite é uma técnica caixa branca que foca em testar valores de entrada e saída, acima e abaixo dos limites dos parâmetros de entrada possíveis.

b) A cobertura de decisão é uma técnica caixa branca que foca em escrever casos de teste nos quais se evitam que determinadas ramificações do código sejam executadas, visando reduzir a cobertura para otimizar o tempo de teste.

c) A cobertura de condição é uma técnica caixa preta que foca em verificar se cada condição na decisão é executada pelo menos uma vez.

d) Um teste de partição de equivalência é uma técnica caixa preta em que se consegue escolher conjuntos de entradas que possibilitam reduzir o número de casos de testes que precisam ser desenvolvidos para atingir algum objetivo.

e) A suposição de erros é uma técnica caixa branca que consiste em um processo sistemático para especulação de prováveis tipos de erros, culminando com a escrita de casos de teste para expor estes erros.

Comentários:

(a) Errado, a análise de valor limite é uma técnica do teste de caixa-preta; (b) Errado, a cobertura de decisão é realmente uma técnica caixa-branca, entretanto, ela visa aumentar a cobertura e não diminuir; (c) Errado, o teste de condição é um teste de caixa-branca; (d) Correto, é a definição do teste de partição; (e) Errado, a suposição de erros é um teste de caixa-preta.

Gabarito: Letra D

---

29. (COVEST-COPSET / UFPE – 2019) Quanto aos tipos de testes de software, assinale a alternativa incorreta:

a) Teste unitário ou de unidade é onde unidades de programa individuais ou classes de objeto são testadas. O foco é testar a funcionalidade de objetos ou métodos.

b) Teste de componente é onde várias unidades individuais são integradas para criar componentes compostos. O foco é testar as interfaces de componentes.

c) Teste de sistema é onde alguns ou todos os componentes de um sistema são integrados e o sistema é testado como um todo. O foco é testar as interações dos componentes.

d) Teste de regressão é onde o sistema é restaurado para uma versão anterior. O teste se preocupa em reproduzir comportamento de algum bug reportado através de controle de mudanças, para posterior correção.

e) Testes de performance é onde se testa um requisito não funcional. O teste se preocupa em demonstrar que o sistema atende aos requisitos e em descobrir problemas e defeitos no sistema.

Comentários:

A única alternativa incorreta é a letra (d). O teste de regressão é a reexecução do mesmo subconjunto de testes que já foram executados para assegurar que as alterações não tenham propagado efeitos colaterais indesejados, ou seja, não há restauração do sistema para uma versão anterior.

Gabarito: Letra D

---

30. (COVEST-COPSET / UFPE – 2019) Como parte da prática de Integração Contínua (CI), podemos elaborar testes unitários, testes de integração e testes de aceitação. Ao se executar um teste de aceitação:

a) executa-se o sistema inteiro com um subconjunto básico de funcionalidades para avaliar o funcionamento de tarefas críticas.

b) executa-se a rotina com uma entrada pré-definida e compara-se a saída com um resultado pré-definido.

c) executa-se o sistema inteiro com foco na especificação das funcionalidades para saber se a rotina implementada corresponde ao requisitado.

d) executa-se o sistema inteiro avaliando o fluxo de informação entre as diferentes rotinas do sistema e sua interoperabilidade.

e) executa-se a rotina desejada em conjunto com um subconjunto de rotinas que possuem dependência direta e compara-se a saída com um resultado pré-definido.

Comentários:

O teste de aceitação é também conhecido como teste de validação. De acordo com Pressman, a validação de software é conseguida por meio de uma série de testes que demonstram conformidade com os requisitos. Em suma, o teste de validação tem por finalidade encontrar defeitos e inconsistências com relação a sua especificação de requisitos.

Gabarito: Letra C

---

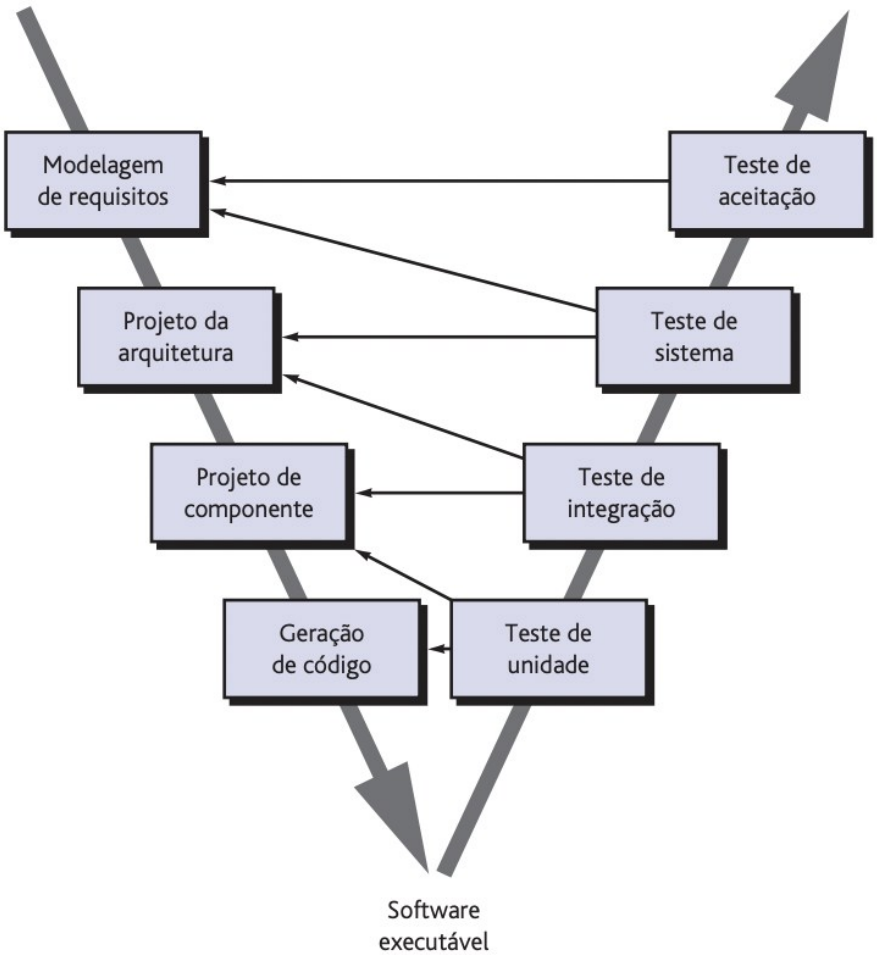
31. (CESGRANRIO / UNIRIO – 2019) José é um desenvolvedor e acabou de fazer uma alteração no código. O gerente de projeto definiu que serão realizados os seguintes testes: unitários/integração; de sistema; de aceitação. A empresa possui uma área de testes independente da equipe de desenvolvimento. O Desenvolvedor, a equipe de teste e o usuário devem executar, respectivamente, os seguintes testes:

a) de sistema; unitário/de integração; de aceitação.
b) de sistema; unitário/de integração; de aceitação.

c) unitário/de integração; de sistema; de aceitação.
d) unitário/de integração; de aceitação; de sistema.
e) de sistema; de aceitação; unitário/de integração.

Comentários:

Os testes começam no nível de componente e progridem em direção à integração do sistema computacional como um todo. Logo, primeiramente devem ser realizados os testes unitários, depois os testes de integração dos componentes, em seguida, os testes de sistema em que o software e outros elementos são testados como um todo. Por fim, temos o teste de aceitação que é executado quando o software está todo montado como um pacote e os erros de interface já foram descobertos e corrigidos. É importante observar que há uma pequena divergência entre os dois principais autores: Pressman e Sommerville. A questão se baseou no entendimento de Pressman. Já o entendimento de Sommerville é que o teste de aceitação (validação) ocorre antes do teste de sistema.



Gabarito: Letra C

---

32. (CESGRANRIO / UNIRIO – 2019) Os testes de integração determinam se as unidades de software desenvolvidas independentemente funcionam corretamente quando estão

conectadas umas às outras. Dentre os tipos de teste que são comumente usados nos testes de integração, estão os testes de:

a) matriz ortogonal e de análise de valor limite.
b) interfaces com o usuário e de cenários de uso.
c) usabilidade e de cenários de uso.
d) desempenho e os beta testes.
e) desempenho e usabilidade.

Comentários:

(a) Errado, matriz ortogonal e análise de valor limite são utilizados em testes de caixa-preta; (b) Correto, pois os testes de Integração são caracterizados por testar as interfaces entre os componentes ou interações de diferentes partes de um sistema; (c) Errado, testes de usabilidade são projetados para determinar o grau com o qual a interface de um software facilita a vida do usuário; (d) Errado, os testes de desempenho são utilizados testar o desempenho em tempo de execução do software dentro do contexto de um sistema integrado; (e) Errado, esses testes não são comuns nos testes de integração.

Gabarito: Letra B

---

33. **(FEPESE / CELESC – 2019)** Assinale a alternativa que apresenta o tipo de teste de software que é baseado nos requisitos funcionais do software. Neste tipo de teste os casos de teste são gerados sem o conhecimento da estrutura interna do software:

a) Caixa Preta.
b) Caixa Branca.
c) Caixa Estrutural.
d) Caixa Funcional.
e) Cobertura Funcional.

Comentários:

O teste em que não há qualquer conhecimento do código ou lógica interna do componente de software é o teste de Caixa Preta. Em suma, ele é baseado nas entradas e saídas especificadas nos requisitos funcionais.

Gabarito: Letra A

---

34. **(IF / MT – 2019)** Analise as sentenças presentes em Pressmann (2006) relacionadas abaixo, acerca de estratégias de teste para software convencional:

I - Teste de_____ é uma técnica sistemática para construir a arquitetura do software enquanto, ao mesmo tempo, conduz testes para descobrir erros associados às interfaces.

II - No teste de _____ são utilizadas as descrições de projeto no nível de componente como guia para testar caminhos de controle importantes e descobrir erros dentro dos limites do módulo.

III - O teste de _____ é uma abordagem de teste de _____ e tem por objetivo exercitar o sistema inteiro, de ponta a ponta, sendo capaz de expor os principais problemas existentes no produto ainda na etapa de construção.

IV - O teste de _____ tem por objetivo verificar se há defeitos de software em modificações recentes que afetam módulos já testados e que antes funcionavam impecavelmente.

As lacunas das sentenças podem ser preenchidas CORRETAMENTE com a opção:

a) Integração, Unidade, Fumaça, Integração, Regressão.
b) Unidade, Integração, Regressão, Usabilidade, Regressão.
c) Interação, Configuração, Regressão, Unidade, Utilidade.
d) Recuperação, Desempenho, Sistema, Desempenho, Unidade.
e) Configuração, Regressão, Integração, Unidade, Fumaça.

Comentários:

(I) O teste que busca verificar se há erros associados às interfaces é o teste de integração; (II) O teste realizado a nível de componente é o teste de unidade; (III) Trata-se do teste de fumaça (smoke testing), que é um tipo de teste de integração. Ele busca verificar se o software roda e provê suas funcionalidades e características básicas; (IV) O teste que busca verificar se há efeitos colaterais é o teste de regressão.

Gabarito: Letra A

---

35. (IF-PE / IF-PE – 2019) Em relação aos testes de software, podemos afirmar que:

I. teste de unidade é a realização de testes sobre unidades do sistema para garantir que a funcionalidade de objetos ou métodos esteja correta.

II. teste de componentes é a realização de testes sobre as interfaces entre os componentes de um software.

III. teste de sistema procura testar a integração de todos os componentes de um sistema.

Está(ão) CORRETA(S), apenas, a(s) proposição(ões):

a) I,II e III
b) Apenas I
c) Apenas I e II
d) Apenas II e III
e) Apenas I e III

Comentários:

(I) Perfeito, o teste de unidade testa cada componente do sistema; (II) Correto, o teste de componente é outro nome para o teste de unidade; (III) Correto, o teste de sistema tem a finalidade de exercitar o sistema como um todo.

Gabarito: Letra A

---

36. **(VUNESP / PREFEITURA DE ITAPEVI-SP – 2019)** Um programador, após desenvolver um programa, iniciou o processo de depuração do código. O teste projetado em função da estrutura interna do software e que visa cobrir a funcionalidade do componente de software é denominado Teste:

a) de Carga.
b) de Estresse.
c) de Segurança.
d) Estrutural (caixa-branca).
e) Funcional (caixa-preta).

Comentários:

O teste que é focado nas estruturas internas do software é o teste de caixa-branca, também chamado de teste estrutural. Ele analisa caminhos lógicos possíveis de serem executados, portanto é necessário ter conhecimento sobre o funcionamento interno dos componentes, ou seja, do código-fonte do software.

Gabarito: Letra D

---

37. **(VUNESP / PREFEITURA DE PIRACICABA-SP – 2019)** Dentre as técnicas de teste de software, pode-se estabelecer uma categorização entre testes de caixa preta e de caixa branca, sendo correto que a técnica de teste denominada:

a) análise de valor limite é um método de teste de caixa branca.
b) matriz ortogonal é um método de teste de caixa branca.
c) teste de condição é um método de teste de caixa preta.

d) fluxo de dados é um método de teste de caixa preta.
e) particionamento de equivalência é um método de teste de caixa preta.

Comentários:

(a) Errado, a análise de valor limite é um teste de caixa-preta; (b) Errado, a matriz ortogonal é um teste de caixa-preta; (c) Errado, o teste de condição é um teste de caixa-branca; (d) Errado, fluxo de dados é um teste de caixa-branca; (e) Correto, particionamento é um teste de caixa-preta.

Gabarito: Letra E

---

38. **(VUNESP / PREFEITURA DE VALINHOS-SP – 2019)** Há dois tipos de testes de validação de software, conhecidos como testes alfa e beta, segundo os quais:

a) o teste alfa é realizado nas instalações do desenvolvedor do software.
b) o teste beta é realizado nas instalações do desenvolvedor do software.
c) os testes alfa e beta são executados em um ambiente tercerizado.
d) no teste alfa, participam apenas os desenvolvedores do software.
e) no teste beta, participam apenas os desenvolvedores do software.

Comentários:

Os testes alfa e beta são realizados apenas usuários finais, e não pelos desenvolvedores. Ademais, os testes alfa são realizados nas instalações do desenvolvedor do software e os testes beta são realizados no ambiente do usuário final.

Gabarito: Letra A

---

39. **(IADES / BRB – 2019)** Há diversos tipos de testes de software e, entre eles, o tipo que consiste no reteste de um sistema ou componente focado em verificar se alguma modificação recente causou efeitos negativos no sistema denomina-se teste de:

a) manutenção.
b) performance.
c) usabilidade.
d) integração.
e) regressão.

Comentários:

O teste que consiste na reexecução de um conjunto de testes, afim de verificar se há efeitos colaterais indesejados é o teste de regressão.

Gabarito: Letra E

---

40. (IDECAN / IF-PB– 2019) O processo de teste tem dois objetivos distintos:

i) Demonstrar ao desenvolvedor e ao cliente que o software atende a seus requisitos e
ii) Descobrir situações em que o software se comporta de maneira incorreta, indesejável ou de forma diferente das especificações.

Sobre testes de software, é incorreto afirmar que:

a) os testes não são capazes de demonstrar que um software é livre de defeitos.
b) o objetivo da verificação é checar se o software atende aos requisitos funcionais e não funcionais.
c) o objetivo da validação é garantir que o software atende às expectativas do cliente.
d) testes de integração focam na descoberta de erros associados com interfaces de módulos.
e) teste fumaça é uma abordagem de testes unitários.

Comentários:

(a) Correto, é impossível testar todas as possibilidades de defeitos de um sistema; (b) Correto, os processos de verificação analisam e verificam os requisitos de um sistema; (c) Correto, a validação tem sucesso quando o software funciona de uma maneira que pode ser razoavelmente esperada pelo cliente.; (d) Correto, o teste de integração visa descobrir erros associados às interfaces; (e) Errado, o teste de fumaça está relacionado ao teste de integração.

Gabarito: Letra E

---

41. (CCV / UFC – 2019) Sobre os tipos de testes de software, marque o item correto.

a) Nos testes caixa-preta, todo o código da aplicação estará disponível para o profissional de teste analisar e especificar quais pontos deverão ser testados.

b) Os testes de fluxo de dados visam analisar os aspectos estruturais da aplicação nos trechos de código onde os dados passam, com foco nas estruturas de controle.

c) Os testes de regressão são realizados pelos clientes com o intuito de checar se todas as funcionalidades e alterações demandadas foram desenvolvidas.

d) Os testes funcionais procuram verificar se o sistema está seguindo a sua especificação (requisitos), sem se preocupar com a estrutura adotada na implementação do sistema.

e) Os testes de unidade são realizados após a conclusão dos diferentes módulos do sistema, onde se busca analisar a corretude do funcionamento da integrado desses módulos.

Comentários:

(a) Errado, nos testes de caixa-preta não há acesso à estrutura interna de um software; (b) Errado, de acordo com Pressman: o teste de fluxo de dados seleciona caminhos de teste de um programa de acordo com a localização de definições e usos de variáveis no programa; (c) Errado, os testes de aceitação é que são realizados pelos clientes; (d) Correto, é como funciona o teste de caixa-preta; (e) Errado, o teste que analisa o funcionamento em conjunto dos módulos de um sistema e o teste de integração.

Gabarito: Letra D

---

42. (CCV / UFC – 2019) Durante o desenvolvimento de um sistema, é necessária a realização de testes, sendo um deles denominado de teste beta. Sobre esse tipo de teste, assinale a alternativa correta:

a) É realizado no ambiente de desenvolvimento, onde somente os desenvolvedores envolvidos na escrita do código realizarão os testes.

b) É considerado como sendo um teste de unidade, onde todos os módulos desenvolvidos separadamente são testados em conjunto.

c) O teste beta é aplicado para os usuários contendo apenas uma versão inicial do sistema, com poucos recursos para a validação de requisitos.

d) Tal teste são realizados com os usuários do sistema analisando o código fonte produzido, propondo correções e melhorias a serem aplicadas.

e) É realizado pelos usuários do sistema, onde os requisitos do sistema são analisados, validados e os erros encontrados reportados para a equipe de desenvolvimento.

Comentários:

Os testes alfa e beta são realizados pelos usuários finais. Ademais, o teste beta é realizado em ambiente não controlado, ou seja, longe dos desenvolvedores. Caso os usuários encontrem erros, estes devem reportar aos desenvolvedores. Ademais, o teste alfa é realizado no ambiente de desenvolvimento.

Gabarito: Letra E

---

43. **(IF / MS – 2019)** Segundo Pressman (2011), considere as seguintes afirmações sobre Engenharia de Software:

I. Erro é um problema de qualidade encontrado após a liberação para o usuário final.

II. O teste de unidade é a reexecução do mesmo subconjunto de testes que já foram executados para assegurar que as alterações não tenham propagado efeitos colaterais indesejados.

III. O modelo espiral é um modelo de processo de software evolucionário que acopla a natureza iterativa da prototipação com os aspectos sistemáticos e controlados do modelo cascata.

Assinale a opção CORRETA:

a) Apenas a afirmação I é verdadeira.
b) Apenas a afirmação II é verdadeira.
c) Apenas a afirmação III é verdadeira.
d) Apenas as afirmações I e III são verdadeiras.
e) Apenas as afirmações II e III são verdadeiras.

Comentários:

(I) Errado, na verdade, são os defeitos que são encontrados após a liberação para o usuário final; (II) Errado, trata-se do teste de regressão; (III) Correto, o modelo em espiral é também conhecido como prototipagem-em-etapas, por combinar, em geral, o modelo em cascata com a prototipação.

Gabarito: Letra C

---

44. **(FUMARC / CÂMARA DE CARMO DO CAJURU-MG – 2018)** Em relação aos tipos de testes de software, julgue os itens a seguir, marcando com (V) a assertiva verdadeira e com (F) a assertiva falsa:

( ) Teste de Regressão significa executar novamente um subconjunto de testes já realizado anteriormente, para garantir que as últimas modificações não propagarão efeitos colaterais indesejáveis no software.

( ) Testes Alfa são realizados no ambiente de produção do usuário final para identificar possíveis problemas nesse ambiente que não foram detectados nas fases anteriores de teste.

( ) O Teste de Estresse executa um sistema de tal forma que ele demande recursos em quantidade, volume ou frequência anormais, com o objetivo de identificar limites de capacidade.

( ) O Teste de Unidade avalia a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente.

A sequência CORRETA, de cima para baixo, é:

a) F, F, V, V.
b) F, V, F, V.
c) V, F, V, V.
d) V, V, F, F.

Comentários:

(V) Correto, perfeita definição de testes de regressão; (F) Errado, trata-se dos testes beta; (V) Correto, é a definição de um teste de estresse, que é um tipo de teste de desempenho; (V) Correto, é a definição de teste de unidade, também chamado teste de componente.

Gabarito: Letra C

---

45. (FAUGRS / UFCSPA-RS – 2018) No teste de _____, os módulos são combinados e testados em grupo. Ele sucede o teste de _____, em que os módulos são testados individualmente, e antecede o teste de _____, em que o sistema completo é testado em um ambiente que simula o ambiente de produção.

Assinale a alternativa que completa, correta e respectivamente, as lacunas do texto acima.
a) integração – unidade – sistema.
b) release – integração – unidade.
c) unidade – caminho – release.
d) requisito – sistema – integração.
e) partições – requisito – caminho.

Comentários:

Testes de Integração visam testar as falhas decorrentes da integração dos módulos do sistema, eles ocorrem após os testes de unidade em que cada componente é testado de forma individual, e antecede o teste de sistema que verifica interfaces entre componentes diferentes de um mesmo sistema.

Gabarito: Letra A

---

46. (FUNDATEC / CIGA-SC – 2018) Uma equipe de teste de software identificou erros em algumas funcionalidades de um aplicativo durante a execução de suas atividades. Os erros foram reportados ao Gerente de Projetos que acionou imediatamente a equipe de desenvolvimento para a realização dos ajustes necessários. Concluídas as correções, a equipe de testes recebeu

novamente as funcionalidades e confirmou que os erros foram devidamente corrigidos. Apesar disso, essa equipe irá realizar mais um ciclo de teste com o objetivo de verificar se a nova versão do software, ajustada pelos desenvolvedores, não introduziu novos defeitos, em outros pontos do aplicativo, em consequência dos ajustes realizados. Nesse caso, esse tipo de teste é chamado de teste:

a) De estresse.
b) Funcional.
c) De regressão.
d) De segurança.
e) De aceitação do produto.

Comentários:

Trata-se reexecutar os testes que já foram realizados com o objetivo de verificar se não foram introduzidos novos erros, trata-se do teste de regressão.

Gabarito: Letra C

---

47. **(FUNDATEC / CIGA-SC – 2018)** A equipe responsável pelo desenvolvimento de um software está agilizando a conclusão de um release para entregá-lo estável ao cliente. No momento, as funcionalidades de tal release estão sendo submetidas a testes intensivos, pela equipe de testes. Essa equipe, antes de iniciar suas atividades, estudou os artefatos da linha base elaborados no projeto, tais como os documentos de viabilidade e visão, diagrama de casos de uso e as suas especificações, casos de teste, regras de negócio, modelo de dados, lista de mensagens e os tipos de dados e valores válidos para os diversos campos do sistema, dentre outros. Esses testadores encontram-se realizando diversas simulações, com a entrada de dados certos e errados, de modo a observar se o sistema se comporta conforme os documentos da linha base, acordados com o cliente. Não faz parte do escopo da equipe de teste, validar a linguagem de programação, a estrutura de dados, os algoritmos ou qualquer outro aspecto da arquitetura e estrutura interna do sistema. Nesse caso, o tipo de teste de software, que se encontra sendo realizado pela equipe de teste, é chamado de teste:

a) Unitário.
b) De caixa preta.
c) De caixa branca.
d) De performance.
e) De desenvolvimento.

Comentários:

A equipe de teste não tem acesso às estruturas internas do software, logo trata-se do teste de caixa-preta.

Gabarito: Letra B

48.(INSTITUO AOCP / PRODEB – 2018) Qual é o tipo de testes em que o testador está preocupado com a funcionalidade e não com a implementação?

a) Teste unitário.
b) Teste de release.
c) Teste disfuncional.
d) Teste de integração.
e) Teste de caixa cinza.

Comentários:

O objetivo principal do processo de teste de release é convencer o fornecedor do sistema de que esse sistema é bom o suficiente para uso. Nesse caso, a funcionalidade é mais importante do que a implementação pois para os clientes – normalmente – não importa como o software foi implementado, o que importa é que o software é funcional.

Gabarito: Letra B

49.(AOCP / UNIR – 2018) O projeto de casos de teste não possui eficácia para a prevenção de defeitos e identificação de erros.

Comentários:

Na verdade, é o contrário os testes são utilizados para identificar erros e prevenir defeitos.

Gabarito: Errado

50.(AOCP / UNIR – 2018) As etapas do teste de software são: implantação, verificação e análise de resultados.

Comentários:

O processo de teste é composto pelas etapas de planejamento, preparação, especificação, execução e entrega.

Gabarito: Errado

51.(AOCP / UNIR – 2018) Os defeitos no processo de desenvolvimento, em sua maior parte, são de origem humana, pois são gerados na comunicação e na transformação de informações, e

continuam presentes nos diversos produtos de software produzidos, liberados e localizados em partes do código raramente executadas.

Comentários:

Exato! Alguns autores consideram que o elemento humano é parte de um sistema, composto ainda por harware, software e tarefas, dessa forma, um erro humano por acarretar defeitos no sistema, ou seja, o não cumprimento de seus requisitos.

Gabarito: Correto

---

52. (AOCP / UNIR – 2018) Para testar a complexidade de um software, pode ser utilizado o teste de integração que avalia a integração do software com vários usuários ao mesmo tempo.

Comentários:

Na verdade, o teste de integração avalia as interfaces. O Teste de Integração é uma técnica sistemática para construir a arquitetura de software ao mesmo tempo que conduz testes para descobrir erros associados com as interfaces.

Gabarito: Errado

---

53. (IBADE / IPM-JP – 2018) No âmbito dos testes de integração, a atividade de reexecução de um mesmo subconjunto dos que já foram executados para assegurar que alterações não tenham propagado efeitos colaterais indesejados é conhecida como teste de:

a) unidade.
b) regressão.
c) validação.
d) verificação.
e) classe.

Comentários:

A reexecução de um mesmo subconjunto de testes para assegurar que não existam efeitos colaterais indesejados trata-se do teste de regressão.

Gabarito: Letra B

---

54. (IBADE / IPM-JP – 2018) Uma estratégia de teste de software pode englobar diferentes tipos de testes para assegurar a qualidade do software. Os que proporcionam a garantia final de que o software satisfaz todos os requisitos informativos, funcionais, comportamentais são conhecidos como testes:

a) de validação.
b) de unidade.
c) de integração.
d) totais.
e) globais.

Comentários:

A validação tem sucesso quando o software funciona de uma maneira que pode ser razoavelmente esperada pelo cliente. Ou seja, o teste de validação tem como principal característica verificar o sistema em relação aos seus requisitos originais e às necessidades atuais do usuário.

Gabarito: Letra A

---

55. (AOCP / SUSIPE-PA – 2018) Sobre os testes de software, assinale a alternativa correta.

a) Um teste de regressão visa refazer os testes feitos anteriormente, visando garantir o funcionamento correto destes.
b) Um teste operacional tem como objetivo testar a aplicação em funcionamento no sistema operacional utilizado, visando encontrar possíveis conflitos de operações.
c) Um teste de configuração visa garantir que as configurações da aplicação não sejam conflitantes com o ambiente utilizado.
d) Um teste de integração visa garantir a interação da aplicação com outras aplicações.
e) Um teste de carga tem como objetivo verificar o comportamento do sistema com uma grande carga de usuários simultâneos.

Comentários:

(a) Errado, o objetivo dos testes de regressão é garantir que não surjam efeitos colaterais indesejados no software; (b) Errado, trata-se do teste de sistema; (c) Errado, trata-se do teste de sistema; (d) Errado, o Teste de Integração é uma técnica sistemática para construir a arquitetura de software ao mesmo tempo que conduz testes para descobrir erros associados com as interfaces; (e) Correto, Testes de Carga são a forma mais simples de testes para compreender o comportamento de um sistema sob uma carga específica.

Gabarito: Letra E

---

56. (INSTITUTO AOCP / PRODEB – 2018) Qual é o objetivo da realização de testes funcionais?

a) Visualizar o sistema como caixa branca.
b) Convencer de que o sistema é bom o suficiente para uso.
c) Justificar as possíveis falhas.

d) Identificar valor em funcionalidades não previstas.
e) Permitir validação feita pelo cliente após a entrega final das funcionalidades.

Comentários:

(a) Errado, nos testes funcionais ignoram-se detalhes de implementação; (b) Correto, nesse tipo de teste focaliza-se os requisitos do sistema, ou seja, se o sistema funciona conforme foi projetado; (c) Errado, o objetivo é analisar como o software deve funcionar; (d) Errado, não há relação com testes funcionais; (e) Errado, trata-se do teste de aceitação.

Gabarito: Letra B

---

57. (PR-4 / UFRJ – 2018) Assinale o teste que focaliza o esforço de verificação da menor unidade de projeto de software:

a) Teste de integração.
b) Teste de regressão.
c) Teste de validação.
d) Teste de unidade.
e) Teste de projeto.

Comentários:

Os Testes de Unidade focalizam cada componente de um software de forma individual, garantindo que o componente funciona adequadamente.

Gabarito: Letra D

---

58. (FAURGS / UFRGS – 2018) Numere a segunda coluna de acordo com a primeira, associando os termos com suas respectivas definições:

(1) Teste de regressão
(2) Teste funcional
(3) Teste caixa-branca
(4) Teste unitário
(5) Teste de estresse
(6) Teste de desempenho
(7) Teste de segurança
(8) Teste de software
(9) Teste de carga

( ) Seu objetivo é verificar o comportamento do software contra a lógica de negócio descrita nos documentos de requisitos e especificação.

( ) Teste que foca na lógica interna de processamento e nas estruturas de dados dentro dos limites de um componente.

( ) Verifica a performance do software durante a execução, principalmente em relação aos critérios ligados a consumo de recursos de processamento, memória e tempo de resposta.

( ) Processo de retestar um software que sofreu modificações.

A sequência numérica correta de preenchimento dos parênteses da segunda coluna, de cima para baixo, é:

a) 5 – 3 – 9 – 6.
b) 2 – 4 – 6 – 1.
c) 4 – 6 – 9 – 1.
d) 2 – 3 – 5 – 9.
e) 5 – 4 – 6 – 8.

Comentários:

(2) O teste que verifica o software da forma como ele deve funcionar, ou seja se ele está de acordo com a especificações de requisitos é o teste funcional; (4) Entre as opções de testes que focam na estrutura interna, temos o teste de caixa-branca e o teste unitário, porém a questão restringe a definição ao limite de um componente, nesse caso, trata-se do teste unitário; (6) O teste de desempenho visa testar o desempenho em tempo de execução do software dentro do contexto de um sistema integrado; (1) O processo de reexecução dos testes está relacionado ao teste de regressão.

Gabarito: Letra B

---

59. **(COPESE / CÂMARA DE PALMAS-TO – 2018)** São consideradas fases da atividade de teste de softwares, EXCETO:

a) Teste de Unidade.
b) Teste de Integração.
c) Teste de Sistemas.
d) Teste de Fault.

Comentários:

Todos são testes de software, exceto a letra (d). Os testes de unidade são aqueles realizados sobre as menores estruturas de código-fonte, os testes de integração visam testar as falhas decorrentes

da integração dos módulos do sistema e os testes de sistema testam se o sistema cumpre seus requisitos funcionais e não funcionais.

Gabarito: Letra D

60.(CESGRANRIO / LIQUIGÁS – 2018) Um grupo de desenvolvedores elaborou vários casos de teste que selecionam caminhos de teste de acordo com as definições e com o uso de variáveis existentes em um programa. Esse tipo de teste caixa-branca é denominado:

a) teste de condição.
b) teste de fluxo de dados.
c) teste de caminho básico.
d) análise de valor-limite.
e) particionamento de equivalência.

Comentários:

A definição trata-se do teste de fluxo de dados. Ele é um é um teste de caixa-branca usado para verificação de softwares que visa exercitar caminhos do programa (executar sequências de linhas de código) com base nas definições e usos de cada variável.

Gabarito: Letra B

61.(UFPR / COREN-PR – 2018) Sobre testes de software, identifique como verdadeiras (V) ou falsas (F) as seguintes afirmativas:

( ) Um teste bem-sucedido identifica defeitos.

( ) Casos de teste são especificações das entradas para o teste e da saída esperada do sistema.

( ) Um teste caixa-preta é um teste estrutural, em que partes específicas de componentes são testadas.

( ) Um teste de aceitação foca em cada unidade do software, ou seja, seu código-fonte.

Assinale a alternativa que apresenta a sequência correta, de cima para baixo.

a) V – V – F – F.
b) F – F – V – V.
c) V – F – F – V.
d) F – V – V – F.
e) V – V – V – V.

Comentários:

(V) Correto, um dos objetivos de um teste é descobrir falhas ou defeitos; (V) Correto, os Casos de Teste são artefatos que contém um conjunto de condições e entradas utilizadas para testar um sistema; (F) Errado, o teste de caixa-preta se baseiam apenas nas interfaces, sem se basear em qualquer conhecimento do código ou lógica interna do componente de software; (F) Errado, o teste de aceitação focaliza apenas em ações visíveis ao usuário.

Gabarito: Letra A

---

62. (FAURGS / BANRISUL – 2018) No desenvolvimento de software, o processo de teste resulta na produção de distintos artefatos. Dentre estes, documentos. Sobre o documento Plano de Teste, assinale a alternativa que apresenta o elemento que NÃO faz parte desse artefato:

a) Escopo dos testes, ou seja, os itens de software que devem ser testados e os que não devem.
b) Estratégia de teste.
c) Necessidades em termos de recursos humanos e treinamentos.
d) Casos de teste.
e) Cronograma.

Comentários:

O Plano de Teste é um artefato que apresenta o planejamento para execução das atividades de testes, apresentando seu escopo, métodos empregados, prazo estimado, nível de qualidade esperado, expectativa de capacidade, recursos que serão utilizados como ferramenta de apoio, e métricas e formas de acompanhamento do processo. Dessa forma, os casos de teste não fazem parte do Plano de Teste, visto que eles contêm um conjunto de condições e entradas utilizadas para testar um software.

Gabarito: Letra D

---

63. (FAURGS / BANRISUL – 2018) _____ é uma técnica utilizada para se projetar casos de teste, na qual o programa ou sistema é considerado como uma caixa-preta. Nesta técnica os detalhes de implementação não são considerados e o software é avaliado segundo o ponto de vista do usuário, procurando explorar determinados tipos de defeitos, estabelecendo requisitos de teste para os quais valores específicos do domínio de entrada do programa devem ser definidos com o intuito de exercitá-los. Utilizando _____, o domínio de entrada é reduzido a um conjunto de estados válidos ou inválidos para as condições de entrada, e com um tamanho passível de ser tratado durante a atividade de teste.

Assinale a alternativa que preenche correta e respectivamente as lacunas do texto acima:

a) Teste funcional – particionamento de equivalência.

b) Teste estrutural – análise de valor limite.
c) Teste funcional – critérios baseados em fluxos de controle.
d) Teste estrutural – critérios baseados em fluxos de controle.
e) Teste estrutural – particionamento de equivalência.

Comentários:

A técnica em que os detalhes de implementação não são conhecidos é o teste de caixa-preta, também conhecido como teste funcional. Ademais, o particionamento de equivalência é um método de teste caixa-preta que divide o domínio de entrada de um programa em classes de dados a partir das quais podem ser criados casos de teste.

Gabarito: Letra A

64.(FAURGS / BANRISUL – 2018) _____ é uma verificação de consistência entre o sistema de software e sua especificação e, portanto, é uma atividade de verificação feita depois que se tem o sistema completo, com todas suas partes integradas para verificar se as funcionalidades especificadas nos documentos de requisitos estão todas corretamente implementadas. Este tipo de teste é focado principalmente na descoberta de falhas e executado pelo grupo de desenvolvimento de testes, tendo também um papel importante para avaliar se o produto pode ser liberado para os consumidores, o que é diferente do seu papel de expor falhas que são removidas para melhorar o produto.

Assinale a alternativa que preenche corretamente a lacuna do texto acima:

a) Teste de sistema
b) Teste de unidade
c) Inspeção
d) Teste de regressão
e) Teste de integração

Comentários:

O teste que é realizado após o teste de integração é o teste de sistema. Além disso, basicamente ele verifica se todos os elementos se combinam corretamente e se a função/desempenho global do sistema é conseguida, analisando se o produto pode ser liberado para os clientes.

Gabarito: Letra A

65.(FAURGS / BANRISUL – 2018) _____ é o teste que tem como foco as menores unidades de um programa, que podem ser funções, procedimentos, métodos ou classes. Neste contexto, espera-se que sejam identificados erros relacionados a algoritmos incorretos ou mal implementados, estruturas de dados incorretas ou simples erros de programação. Como cada

unidade é testada separadamente, este teste pode ser aplicado à medida que ocorre a implementação e pelo próprio desenvolvedor, sem a necessidade de dispor-se do sistema totalmente finalizado.

Assinale a alternativa que preenche corretamente a lacuna do texto acima:

a) Teste de regressão
b) Teste de integração
c) Teste de unidade
d) Teste de sistema
e) Teste de aceitação

Comentários:

O teste que tem como foco as menores unidade de um programa é o teste de unidade.

Gabarito: Letra C

---

66. (FAURGS / BANRISUL – 2018) Considere as seguintes afirmações sobre o Modelo "V" de teste de software.

I - Descreve a relação entre ações de garantia da qualidade e as ações associadas à comunicação, modelagem e atividades iniciais de construção.

II - À medida que a equipe de software desce em direção ao lado esquerdo do "V", os requisitos básicos do problema são refinados em representações, progressivamente, mais detalhadas e técnicas do problema e de sua solução. Ao ser gerado o código, a equipe se desloca para cima, no lado direito do "V", realizando basicamente uma série de testes que validem cada um dos modelos criados, à medida que a equipe se desloca para baixo, no lado esquerdo do "V".

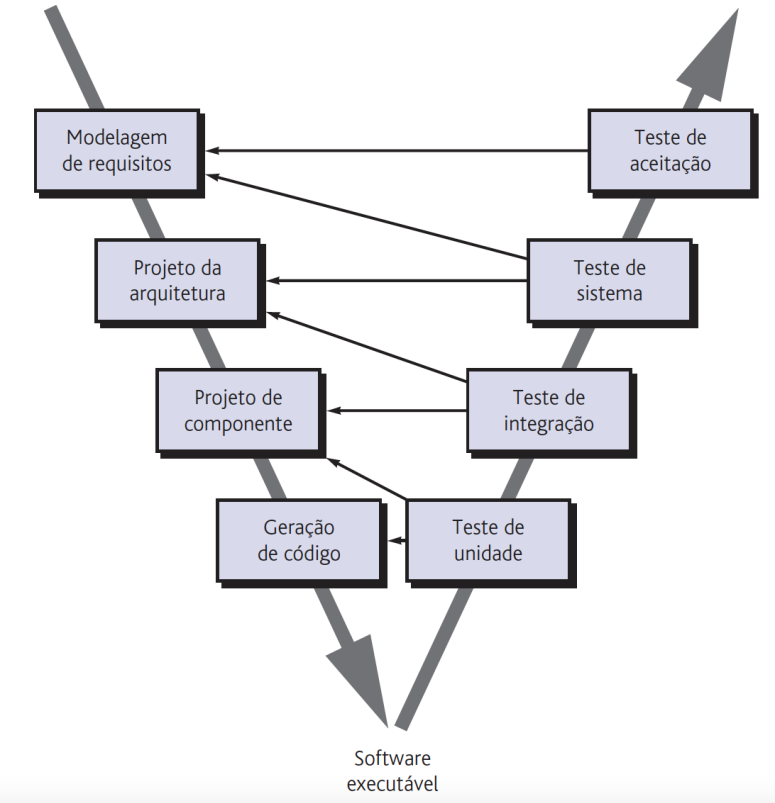
III - Fornece uma forma para visualizar como a verificação e as ações de validação são aplicadas ao trabalho de engenharia anterior.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II, III.

Comentários:

(I) Correto, o modelo V funciona de forma parecida com o modelo em cascata; (II) Correto, conforme a figura abaixo; (III) Correto, verificação e validação é o processo de se demonstrar que um programa atende a sua especificação (verificação) e às necessidades reais de seus stakeholders (validação). O modelo em "V" ilustra os testes responsáveis por isso.



Gabarito: Letra E

---

67. (FAURGS / BANRISUL – 2018) Considere as seguintes afirmações sobre objetivos de teste.

I - A definição dos critérios de aceitação deve idealmente ocorrer depois do contrato do sistema ser assinado, pois os critérios de aceitação não fazem parte do contrato, embora possam ser acordados entre o cliente e o desenvolvedor.

II - O processo de teste deve demonstrar ao desenvolvedor e ao cliente que o software atende aos requisitos propostos.

III - Um dos objetivos do processo de teste é descobrir situações em que o software se comporte de maneira incorreta, indesejável ou de forma diferente das especificações.

Quais estão corretas?

a) Apenas I.

b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II, III.

Comentários:

(I) Errado, os testes de aceitação podem – sim – fazer parte do contrato; (II) Correto, o objetivo de um teste é mostrar que o sistema funciona; (III) Correto, é um segundo objetivo dos testes: encontrar erros e defeitos.

Gabarito: Letra D

---

68. (FAURGS / BANRISUL – 2018) Considere as seguintes afirmações sobre a relação entre requisitos e teste de software.

I - A correção, a completude e a consistência do modelo de requisitos não terão forte influência sobre a qualidade de todos os produtos seguintes do desenvolvimento de software, pois o que importa para o teste é o código fonte.

II - Um dos princípios gerais das boas práticas de engenharia de requisitos é que os requisitos devem ser testáveis, isto é, o requisito deve ser escrito de modo que um teste possa ser projetado para ele. Um testador pode então verificar se o requisito foi satisfeito.

III - Testes baseados em requisitos são uma abordagem sistemática para projeto de casos de teste em que cada requisito é considerado, derivando-se, assim, um conjunto de testes para ele.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II, III.

Comentários:

(I) Errado, na verdade, a qualidade de um software está diretamente ligada a esses fatores; (b) Correto, quanto mais testes houver, a tendência é que haja menos inconsistências no software; (III) Correto, conforme os itens (I) e (II).

Gabarito: Letra D

---

69. **(FAURGS / BANRISUL – 2018)** As alternativas abaixo apresentam características importantes que devem ser consideradas para o teste de aplicativos móveis, EXCETO uma. Assinale-a.

a) Linguagem de desenvolvimento.
b) Usabilidade.
c) Localização geográfica do dispositivo.
d) Volume de dados.
e) Variedade de dispositivos e sistemas operacionais.

Comentários:

Todos os itens são importantes, mas tratando genericamente de teste para aplicativos a linguagem de desenvolvimento não será importante, visto que ela dependerá de uma plataforma específica. Os outros itens não dependem de uma plataforma, são itens gerais que podem ser usados independentemente.

Gabarito: Letra A

---

70. **(FAURGS / BANRISUL – 2018)** Numere a segunda coluna de acordo com a primeira, associando os Níveis de Teste de Software às suas respectivas características.

(1) Teste de Unidade
(2) Teste de Integração
(3) Teste de Sistema
(4) Teste de Aceitação

( ) Avalia o software com respeito ao projeto de seus subsistemas e detecta suposições errôneas sobre pré e pós-condições para execução de um componente, falhas nas interfaces de comunicação dos componentes do software.

( ) Avalia o software com respeito aos seus requisitos e detecta falhas nos requisitos e na interface com o usuário.

( ) Avalia o software com respeito a sua implementação detalhada e detecta falhas de codificação, algoritmos ou estruturas de dados incorretos ou mal implementados.

( ) Avalia o software com respeito ao seu projeto arquitetural e detecta falhas de especificação, desempenho, robustez e segurança.

A sequência correta de preenchimento dos parênteses, de cima para baixo, é:

a) 2 – 4 – 1 – 3.
b) 1 – 4 – 2 – 3.

c) 4 – 2 – 3 – 1.
d) 2 – 1 – 3 – 4.
e) 1 – 3 – 4 – 2.

Comentários:

(2) O Teste que detecta falhas nas interfaces é o teste de integração; (4) O teste que avalia se os requisitos do software estão adequados é o teste de aceitação (validação); (1) O teste que avalia a implementação detalhada, analisando a estrutura interna é o teste de unidade; (3) O teste que avalia o software com a finalidade de exercitar totalmente o sistema é o teste de sistema.

Gabarito: Letra A

---

71. (FUMARC / COPASA – 2018) Teste realizado em ambiente de produção por um grupo de usuários finais para identificar problemas e realizar as devidas correções antes de liberar o software para toda a base de clientes:

a) Teste Alfa.
b) Teste Beta.
c) Teste de Regressão.
d) Teste Fumaça.

Comentários:

Das alternativas apresentadas, apenas as letras (a) e (b) são testes realizados por usuários finais. Ademais, o teste que é realizado no ambiente de produção é o teste beta. E o teste alfa, professor? Ele é realizado no ambiente de desenvolvimento.

Gabarito: Letra B

---

72. (FUMARC / COPASA – 2018) Analise as seguintes afirmativas sobre os tipos de testes:

I. O "Teste de Segurança" verifica se os mecanismos de proteção incorporados a um sistema vão de fato protegê-lo de invasão imprópria.

II. O "Teste de Desempenho" é projetado para submeter o software a situações anormais de funcionamento, demandando recursos excessivos até o limite da capacidade da infraestrutura destinada ao software.

III. O "Teste de Recuperação" força o software a falhar de diversos modos e verifica se a recuperação é adequadamente realizada.

Estão CORRETAS as afirmativas:

a) I, II e III.
b) I e II, apenas.
c) I e III, apenas.
d) II e III, apenas.

Comentários:

(I) Correto, esse é o objetivo de um teste de segurança; (II) Errado, na verdade essa é a definição do teste de estresse, que é um tipo específico de teste de desempenho; (III) Correto, o teste de recuperação é um tipo de teste que força o software a falhar de várias formas e verifica se a recuperação é executada corretamente

Gabarito: Letra C

---

73. **(FUMARC / COPASA – 2018)** Técnica de teste utilizada para descobrir erros associados às interfaces na qual, a partir de componentes testados individualmente, se constrói uma estrutura de programa determinada pelo projeto é:

a) Teste de Desempenho.
b) Teste de Integração.
c) Teste de Segurança.
d) Teste de Unidade.

Comentários:

O teste de unidade testa cada componente separadamente, quando juntamos todos os componentes devemos realizar o teste de integração. Os testes de integração são caracterizados por testar as interfaces entre os componentes ou interações de diferentes partes de um sistema.

Gabarito: Letra B

---

74. **(FUMARC / COPASA – 2018)** Teste realizado na instalação do desenvolvedor com os usuários finais, em um ambiente controlado, para identificar erros e problemas de uso durante a operação do sistema pelos usuários é denominado:

a) Teste Alfa.
b) Teste Beta.
c) Teste de Regressão.
d) Teste Fumaça.

Comentários:

O teste que é realizado por usuários finais no ambiente de desenvolvimento, ou seja, em um ambiente controlado é o teste alfa. *E por que ele é em um ambiente controlado?* Porque o programador fica olhando o usuário utilizar o software e registra os erros que ocorrem.

Gabarito: Letra A

---

75. (IADES / CFM – 2018) A respeito dos processos de verificação, de validação e de teste de software, assinale a opção correta.

a) Verificação, validação e teste são atividades independentes, de maneira que não possuem qualquer vínculo entre si.
b) Validação é uma atividade que permite realizar a verificação e os testes do software.
c) Teste consiste em analisar-se o software construído para confirmar se ele atende às verdadeiras necessidades dos interessados (cliente, usuário etc).
d) Verificação consiste em analisar-se o software para confirmar se ele está sendo construído de acordo com o que foi especificado.
e) Os testes devem ser executados antes das atividades de verificação.

Comentários:

(a) Errado, há um vínculo – sim – entre elas, de modo que ambas abrangem muitas atividades de qualidade de software, justamente por isso elas são chamadas de V&V; (b) Errado, na verdade, a validação refere-se a um conjunto de tarefas que asseguram que o software foi criado e pode ser rastreado segundo os requisitos do cliente; (c) Errado, testes são realizados para verificar se o software funciona corretamente; (d) Correto, o teste de verificação tem por finalidade encontrar defeitos e inconsistências com relação a sua especificação de requisitos; (e) Errado, eles são realizados durante a atividade de verificação.

Gabarito: Letra D

---

76. (CS / SANEAGO-GO – 2018) No âmbito da Engenharia de Software, testes de unidade são aqueles realizados:

a) no sistema como um todo, de maneira que este mostre conformidade em relação à especificação de requisitos.
b) sobre as menores estruturas de código-fonte, como métodos e classes.
c) para verificação de integração entre módulos, de maneira que estes mostrem unidade.
d) em módulos ou unidades do sistema, de maneira que possam validar um componente específico.

Comentários:

Os testes de unidade, também chamado de Teste de Componente/Módulo, focalizam o esforço de verificação nas menores unidades de projeto do software, como métodos e classes.

Gabarito: Letra B

---

77. (COPEREVE / UFSC – 2018) Considere as seguintes afirmativas a respeito de teste de software e assinale a alternativa correta.

I. O teste de unidade concentra o esforço de verificação na menor unidade de design de software.

II. O teste de unidade concentra-se na lógica de processamento interno e nas estruturas de dados dentro dos limites de um componente.

III. O teste de integração se concentra em ações visíveis pelo usuário e na saída reconhecível pelo usuário do sistema.

IV. O teste de integração é uma técnica sistemática para a construção da arquitetura de software, ao mesmo tempo em que realiza testes para descobrir erros associados às interfaces.

a) Somente as afirmativas I, II e IV estão corretas.
b) Todas as afirmativas estão corretas.
c) Somente as afirmativas I e II estão corretas.
d) Somente as afirmativas I e III estão corretas.
e) Somente as afirmativas II, III e IV estão corretas.

Comentários:

(I) Correto, os testes de unidade focalizam o esforço nas menores unidade de projeto do software; (II) Correto, por isso são também chamados de testes de componentes; (III) Errado, esse é o teste de caixa-preta; (IV) Correto, essa é a definição de teste de integração.

Gabarito: Letra A

---

78. (CONSUPLAN / CÂMARA DE BELO HORIZONTE-MG – 2018) Na análise e projeto de sistemas, uma fase considerada muito importante é a de testes. Diversos tipos de testes são executados, desde a fase inicial até a implantação do novo sistema. Os testes têm como objetivo verificar a funcionalidade do sistema se o sistema atende ao que foi projetado. Quatro estágios de testes são conhecidos e cada um tem os seus respectivos tipos de testes. Dois tipos de testes são: testes da caixa preta e testes da caixa branca. Assinale a alternativa que apresenta corretamente qual estágio esses tipos de testes pertencem:

a) Testes de Sistema.

b) Testes de Unidade.
c) Testes de Aceitação.
d) Testes de Integração.

Comentários:

Questão polêmica, a meu ver não há como definir em qual estágio os testes de caixa-branca e caixa-preta pertencem. Por exemplo, não há nada que impeça que um teste de caixa-preta seja aplicado nas fases de teste de integração, sistema ou aceitação.

Gabarito: Letra B

---

79. **(PR4 / UFRJ – 2018)** Com relação a teste de software, quando questionado sobre a construção de um produto corretamente, a referência se dá ao conjunto de atividades que garantem que o software implemente corretamente uma função específica. Este conceito se refere à:

a) validação.
b) homologação.
c) aceitação.
d) recuperação.
e) verificação.

Comentários:

Temos uma função específica, essa função é traduzida em um requisito. O teste que tem a finalidade encontrar defeitos e inconsistências com relação a sua especificação de requisitos é o teste de verificação.

Gabarito: Letra E

---

80. **(PR4 / UFRJ – 2018)** Considere o seguinte texto: Uma técnica sistemática para construir a estrutura do programa enquanto, ao mesmo tempo, conduz testes para descobrir erros associados às interfaces. O objetivo é tomar componentes testados em nível de unidade e construir a estrutura de programa determinada pelo projeto. A afirmação apresentada está se referindo ao teste de:

a) integração.
b) unidade.
c) validação.
d) sistema.
e) depuração.

Comentários:

O Teste de Integração é uma técnica sistemática para construir a arquitetura de software ao mesmo tempo que conduz testes para descobrir erros associados com as interfaces, ou seja, os componentes já foram testados a nível de unidade, agora eles devem ser testados todos juntos.

Gabarito: Letra A

---

81. (CESGRANRIO / TRANSPETRO – 2018) Entre as técnicas de teste de software, aquela que gera versões levemente modificadas de um programa sob teste e exercita tanto o programa original quanto os programas modificados, procurando diferenças entre essas formas, é conhecida como testes:

a) aleatórios
b) exploratórios
c) de mutação
d) de perfil operacional
e) baseados em fluxo de controle

Comentários:

(a) Errado. Teste Aleatório é o chamado Teste Macaco em que as entradas são aleatórias; (b) Errado. Teste Exploratório é um teste em que o analista de teste aprende e testa no mesmo momento, explorando o sistema para aprender durante o teste; (c) Teste de Mutação é tipo de teste onde esses programas levemente modificados servem para identificar testes fracos; (d) Errado. Teste de Perfil Operacional é um teste de qualidade realizado antes da release; (e) Errado. Testes baseados em Fluxo de Controle é um teste caixa-branca que exige que se conheça as estruturas internas para forças todos os estados que geram saídas diferentes.

Gabarito: Letra C

---

82. (FAURGS / BANRISUL – 2018) _____ verifica novamente os casos de teste aprovados em versões prévias do software e assim protege contra alterações indesejadas. Realiza-se durante a manutenção, para mostrar que as modificações efetuadas estão corretas, ou seja, que os novos requisitos implementados funcionam como o esperado e que os requisitos anteriormente testados continuam válidos. Assinale a alternativa que preenche corretamente a lacuna do texto acima.

a) Teste de regressão
b) Teste de sistema
c) Inspeção
d) Refatoração
e) Teste de integração

Comentários:

Quem verifica novamente os casos de teste aprovados em versões prévias do software e assim protege contra alterações indesejadas para mostrar que as modificações efetuadas estão corretas é o teste de regressão.

Gabarito: Letra A

---

83. (UFLA / UFLA – 2018) Sobre teste de regressão de software, são verdadeiras as afirmativas abaixo, EXCETO:

a) Consiste na reexecução de testes previamente realizados no software para garantir que modificações ou novas funcionalidades não introduzam comportamentos indesejáveis ou erros adicionais.
b) É sempre prático e viável executar o teste de regressão após cada modificação do software.
c) Embora normalmente seja conduzido de forma automática, o teste de regressão pode também ser conduzido manualmente.
d) Teste de regressão pode ser configurado para execução em servidores de integração contínua.

Comentários:

Todos os itens estão perfeitos, exceto o segundo – nem sempre é prático executar o teste de regressão a cada modificação. À medida que o teste de integração progride, o número de testes de regressão pode crescer muito. Dessa forma, o conjunto de testes de regressão deve ser projetado de forma a incluir somente aqueles testes que tratam de uma ou mais classes de erros em cada uma das funções principais do programa. É impraticável e ineficiente reexecutar todos os testes para todas as funções do programa quando ocorre uma alteração.

Gabarito: Letra B

---

84. (CESGRANRIO / IBGE – 2016) O sistema que controla as reservas dos clientes de uma rede hoteleira funciona apenas na Web. Entretanto, há uma demanda crescente para que a empresa disponibilize um aplicativo para smartphones. Para oferecer um aplicativo no menor prazo possível, a gerência de TI estabeleceu duas exigências: a primeira é que o novo sistema deve reutilizar ao máximo os módulos atualmente empregados, e a segunda é que a equipe de desenvolvimento deve garantir que as modificações a serem feitas não introduzirão defeitos inexistentes no sistema atual, além de continuar a atender a todos os requisitos anteriormente definidos.

O tipo de teste que deve ser empregado para que a equipe de desenvolvimento atenda à segunda exigência é denominado teste de:

a) estresse
b) volume
c) usabilidade
d) regressão
e) configuração

Comentários:

A finalidade do teste de regressão é justamente aplicar testes em versões mais recentes do software para garantir que não surgiram novos defeitos em componentes já analisados. Ou ainda em situações como a descrita no enunciado em que novos componentes desenvolvidos para a plataforma mobile e/ou alterações em componentes existentes não criem novos defeitos em componentes inalterados e já testados. Caso testes comecem a não passar, então considera-se que o sistema regrediu.

Gabarito: Letra D

---

85. **(IF-PE / IF-PE – 2016)** Em relação aos Testes na Engenharia de Software, qual é o que se refere ao reteste de uma unidade, integração ou sistema, após uma modificação, a fim de verificar se a mudança não introduziu novas falhas?

a) Teste de integração.
b) Teste de regressão.
c) Teste alfa.
d) Teste beta.
e) Teste funcional.

Comentários:

Restes de uma unidade, integração ou sistema, após uma modificação, a fim de verificar se a mudança não introduziu novas falhas é característica do teste de regressão.

Gabarito: Letra B

---

86. **(FCM / IF Farroupilha-RS – 2016)** O teste de software pode ser realizado de diversas formas. Mesmo assim, existem técnicas que podem ser utilizadas para encontrar falhas no software. Analise as afirmativas abaixo:

I - O teste de regressão tem por finalidade repetir o teste em um programa já testado depois de haver uma modificação.

II - O teste de desempenho tem por finalidade elaborar casos de teste que possam subverter as verificações de segurança do programa.

III - O teste de caixa branca trabalha diretamente sobre o código fonte do componente de software.

IV - O teste de caixa preta trabalha diretamente sobre o código fonte do componente de software.

Estão corretas as afirmativas:

a) I e II.
b) I e III.
c) II e IV.
d) II e III.
e) I, II, III e IV.

Comentários:

(I) Correto, esse teste realmente tem por finalidade repetir o teste em um programa já testado depois de haver uma modificação; (II) Errado, esse é o teste de segurança; (III) Correto, ele verifica os componentes internos de um software; (IV) Errado, esse é o teste caixa-branca – testes caixa-preta não possuem acesso ao código-fonte de componentes de software.

Gabarito: Letra B

---

87. (IESES / TRE-MA – 2015) É notório e de comprovado valor que os testes são a melhor maneira de se garantir a qualidade de um software. O teste de regressão é um dos tipos de testes, que tem por objetivo?

a) Certificar que os testes unitários foram realizados antes da própria implementação.
b) Garantir que as novas implementações não tenham quebrado as funcionalidades já existentes.
c) Avaliar se a especificação feita pelo analista de sistemas está correta.
d) Testar o menor nível de detalhe, normalmente este teste é realizado pelo programador.

Comentários:

(a) Errado, esse não é um objetivo do teste de regressão; (b) Correto, esse é o principal objetivo dos testes de regressão; (c) Errado, esse não é um objetivo do teste de regressão; (e) Errado, esse é o objetivo dos testes de unidade.

Gabarito: Letra B

---

88. (CESGRANRIO / Banco da Amazônia – 2014) Um tipo de teste de validação possui as seguintes características:

- Realizado na instalação dos desenvolvedores.
- Conduzido em um ambiente controlado.
- Conta com a participação de usuários e desenvolvedores.

Esse tipo de teste é chamado de:

a) Teste alfa
b) Teste beta
c) Teste de estresse
d) Teste de regressão
e) Teste de desempenho

Comentários:

O teste em que os usuários finais do software o testam sob as observações dos desenvolvedores nas instalações destes – em um ambiente controlado – é o Teste Alfa.

Gabarito: Letra A

---

89. (CESGRANRIO / IBGE – 2014) Antes de lançar seu próximo produto, uma empresa de desenvolvimento de software costuma convidar seus principais clientes para testar uma versão "quase final". Esses clientes são convidados ao local de desenvolvimento e são observados enquanto utilizam o software e registram erros e problemas no uso.

Essa estratégia de teste em um ambiente controlado é conhecida como teste:

a) alfa
b) beta
c) de stress
d) de integração
e) de aceitação do cliente

Comentários:

Local de desenvolvimento? Observados enquanto utilizam o software? Ambiente controlado? Tudo isso nos remete a testes alfa.

Gabarito: Letra A

---

90. (CESGRANRIO / IBGE – 2014) No ciclo de desenvolvimento de sistemas, os testes são de suma importância e podem, dependendo do porte do sistema, ser bastante complexos, exigindo que seu planejamento e realização sejam divididos em fases. Em uma dessas fases, os testes são realizados por um grupo restrito de usuários finais do sistema, que simulam operações de rotina do sistema, de modo a verificar se seu comportamento está de acordo com o solicitado. Essa fase é denominada teste de:

a) integração
b) unidade
c) sistema
d) operação
e) aceitação

Comentários:

O teste que verifica se o sistema cumpre o que foi solicitado é o teste de aceitação ou validação!

Gabarito: Letra E

---

91. (CESGRANRIO / IBGE – 2014) Preocupado com os constantes erros nos sistemas entregues aos usuários, um analista de desenvolvimento resolveu realizar testes conforme o modelo V. A correspondência da validação dos modelos por fases do processo de software, de acordo com esse modelo, está representada em:

a) Teste unitário → Modelagem de requisitos
b) Teste de sistema → Desenvolvimento de componentes
c) Teste de aceitação → Geração de código
d) Teste de integração → Arquitetura do sistema
e) Teste de caixa preta → Desenho das estruturas de dados

Comentários:

O **Teste de Integração** é uma técnica sistemática para construir a **arquitetura de software** ao mesmo tempo que conduz testes para descobrir erros associados com as interfaces. O objetivo é construir uma estrutura de programa determinada pelo projeto a partir de componentes testados em nível de unidade.

Gabarito: Letra D

---

92. (IBFC / TRE-AM – 2014) Assinale o nome do teste de software que consiste tipicamente na aplicação de versões mais recentes do software, para garantir que não surgiram novos defeitos em componentes já analisados:

a) testes unitários.
b) testes de integração.
c) testes de usabilidade.
d) testes de regressão.

Comentários:

Aplicação de versões mais recentes do software para garantir que não surgiram novos defeitos em componentes já analisados é o famoso teste de regressão.

Gabarito: Letra D

---

93. (FUMARC / AL-MG – 2014) Tipo de teste que consiste em aplicar em cada nova versão de um software todos os testes que já foram aplicados nas versões anteriores, possibilitando a identificação dos impactos das implementações da nova versão em funcionalidades que já foram testadas anteriormente e não foram modificadas, é denominado:

a) Teste Caixa-branca.
b) Teste Caixa-preta.
c) Teste de Regressão.
d) Teste Unitário.

Comentários:

Aplicar em cada nova versão de um software todos os testes que já foram aplicados nas versões anteriores, possibilitando a identificação dos impactos das implementações da nova versão em funcionalidades que já foram testadas anteriormente e não foram modificadas, é denominado teste de regressão.

Gabarito: Letra C

---

94. (CCV-UFC / UFC – 2013) O principal objetivo do teste de regressão de software é:

a) Identificar defeitos ou erros no sistema em situação de sobrecarga do sistema (ou parte dele).
b) Verificar a existência de defeitos após alterações em um sistema (ou parte dele) já testado.
c) Identificar defeitos através da inspeção do código-fonte do sistema (ou parte dele).
d) Identificar defeitos através da análise estática do sistema (ou parte dele).
e) Verificar a existência de defeitos no fluxo excepcional.

Comentários:

(a) Errado, isso é teste de estresse; (b) Correto, ele realmente verifica a existência de defeitos após alterações em um sistema já testado; (c) Errado, isso é teste caixa-branca; (d) Errado, isso não é teste de regressão; (e) Errado, isso não é teste de regressão.

Gabarito: Letra B

---

95. (CESGRANRIO / EPE – 2012) Em uma discussão sobre testes, um grupo de programadores emitiu as afirmativas a seguir.

I - Durante um teste, é possível provar apenas a existência de erros, não sua ausência.

II - Durante um teste de validação, são construídos casos de teste com a finalidade de expor defeitos.

III - Na verificação, procura-se saber se o produto está sendo construído de forma correta.

Estão corretas as afirmativas:

a) I, apenas.
b) II, apenas.
c) III, apenas.
d) I e III, apenas.
e) I, II e III.

Comentários:

(I) Correto, testes de maneira alguma provam a ausência de erros, apenas os evidencia; (II) Correto, a validação responde ao seguinte questionamento: *Fizemos o software correto?* Logo, ela tem o objetivo de avaliar se o que foi entregue atende às expectativas do cliente, ou seja, se o sistema é realmente aquilo que o cliente deseja; (III) Correto, a verificação responde ao seguinte questionamento: *Fizemos o software corretamente?* Logo, ela tem o objetivo de avaliar se o que foi planejado realmente foi realizado, ou seja, se os requisitos e funcionalidades documentados foram implementados, além disso a verificação também pode ser realizada para especificação de sistemas, para avaliar se os requisitos estão sendo documentados como deveriam e ainda prever falhas ou inconsistências entre requisitos.

Gabarito: Letra E

---

96. (ESAF / CVM – 2010) Teste de Equivalência de Classe é:

a) É uma técnica que trabalha por dedução física.

b) É uma técnica que tem por objetivo primário reduzir o número de casos de testes.

c) É uma técnica que trabalha por dedução indutiva.

d) É uma técnica que tem por objetivo primário ampliar o número de casos de testes.

e) É uma técnica que tem por objetivo primário montar um conjunto de regras de decisão a partir de uma tabela.

Comentários:

Essa é uma técnica que permite reduzir o número de casos de teste (cobertura) do sistema ao agrupar valores de entrada em categorias de dados.

Gabarito: Letra B

---

97. (FEPESE / SEFAZ-SC – 2010) Analise a definição abaixo.

Teste de software que procura descobrir erros por meio da reaplicação parcial dos testes a um programa modificado.

Assinale a alternativa que cita corretamente o conceito ao qual se refere a definição.

a) Teste de sistema
b) Teste de unidade
c) Teste de regressão
d) Teste de integração
e) Teste de requisitos

Comentários:

Essa é a definição impecável de teste de regressão.

Gabarito: Letra C

---

98. (NCE-UFRJ / UFRJ – 2008) Considere as seguintes afirmativas sobre testes de software:

I - O teste de regressão consiste na re-execução de testes já executados para garantir que modificações introduzidas não geraram efeitos colaterais.
II - O teste fumaça (smoke test) é um tipo de teste de integração que é executado diariamente.
III - O teste de validação focaliza ações e saídas tais como percebidas pelo usuário final.

A(s) afirmativa(s) correta(s) é/são somente:

a) I
b) II
c) III
d) I e II
e) I, II e III

Comentários:

(a) Correto, ele realmente reexecuta os testes já executados anteriormente para garantir que modificações introduzidas não geraram efeitos colaterais; (b) Correto, testes de fumaça são um tipo de teste de integração executado diariamente; (c) Correto, testes de validação realmente focalizam ações e saídas tais como percebidas pelo usuário final.

Gabarito: Letra E

Lista de Questões – CESPE

1. **(CESPE / CNPq – 2024)** Os testes de carga e os testes de esforço são testes de desempenho que exigem instrumentação de hardware e software, uma vez que frequentemente é necessário medir a utilização dos recursos de forma precisa.

2. **(CESPE / CAU-BR – 2024)** Considerada uma técnica sistemática para construir a arquitetura de software concomitantemente à realização de testes para descobrir erros associados às interfaces, o teste de integração realiza testes a partir de componentes testados em unidade.

3. **(CESPE / CAU-BR – 2024)** Considerando que o teste funcional objetiva determinar se um recurso funciona corretamente sem problemas, é possível automatizar esse tipo de teste mesmo que o sistema seja web, em que é possível simular os retornos esperados.

4. **(CESPE / CAU-BR – 2024)** Os testes de regressão são realizados por ocasião da ocorrência de mudanças no software.

5. **(CESPE / TST – 2024)** Assinale a opção em que é apresentada uma técnica de desenvolvimento de software orientada a testes que é voltada para o atendimento dos requisitos do sistema com base no negócio, que utiliza exemplos e dublês de teste e que descreve funcionalidades por meio da sintaxe dado que, quando e então:

   a) teste unitário de software
   b) desenvolvimento orientado por comportamento (BDD)
   c) desenvolvimento guiado por testes (TDD)
   d) desenvolvimento guiado por testes de aceitação (ATDD)
   e) testes de aceitação de usuário (UAT)

6. **(CESPE / TST – 2024)** O projeto de software é a primeira atividade técnica voltada à construção de software e deve ser iniciada logo após:

   a) o teste de software.
   b) o projeto de dados.
   c) o projeto arquitetural.
   d) a análise e a especificação de requisitos.
   e) a codificação.

7. **(CESPE / MPE-GO – 2024)** Caso seja necessário verificar se o software desenvolvido está funcionando conforme o esperado e garantir que suas principais funções não apresentem grandes falhas, na execução rápida de seus principais recursos, indica-se a realização do teste fumaça.

8. **(CESPE / LNA – 2024)** Assinale a opção em que é corretamente apresentado o tipo de teste de software responsável por verificar se diferentes partes do sistema de software foram projetadas

para interagir entre si e se fazem essa interação corretamente, avaliando, inclusive, como os dados são transferidos entre elas.

a) teste de desempenho
b) teste unitário
c) teste funcional
d) teste de integração
e) teste de aceitação

9. **(CESPE / LNA – 2024)** É uma característica-chave de um bom framework de automação de teste:

a) a falta de suporte a linguagens de programação populares.
b) a escalabilidade limitada.
c) a alta complexidade.
d) a baixa manutenibilidade.
e) a flexibilidade mínima.

10. **(CESPE / LNA – 2024)** Os frameworks de teste de software:

a) permitem o aumento da produtividade, apesar de não serem úteis para testar softwares em diferentes plataformas.

b) permitem o aumento da produtividade, ainda que não possam ser integrados a processos de integração contínua.

c) são usados exclusivamente para testes manuais.

d) são ferramentas para criar bugs no software.

e) fornecem estruturas e funcionalidades para automatizar os testes de software.

11. **(CESPE / LNA – 2024)** A abordagem que se concentra principalmente em examinar as estruturas internas ou os funcionamentos de uma aplicação de software é denominada teste de:

a) sistema.
b) caixa preta.
c) caixa branca.
d) aceitação.
e) caixa cinza.

12. **(CESPE / MPO – 2024)** Realiza-se o teste de estresse para confrontar os programas com situações anormais, de forma a exigir recursos em maior quantidade, frequência ou volume.

13. **(CESPE / MPO – 2024)** O teste automatizado pode conter recursos de auditoria eletrônica com avaliadores e geradores automáticos de testes.

14. **(CESPE / INPI – 2024)** Um teste de software de regressão estará corretamente projetado quando se considera, em cada uma das funções principais do software, apenas os testes que tratam de uma ou mais classes de erros.

15. **(CESPE / CAU-BR – 2024)** Os estágios das atividades de teste de software devem ser realizados na seguinte ordem: teste do sistema, teste de integração e teste de unidade.

16. **(CESPE / SEFIN de Fortaleza-CE – 2023)** Acerca dos testes de software e das ferramentas para automatização de testes, bem como do desenvolvimento orientado por comportamento, julgue o item que se segue.

    Na análise do valor limite, casos de teste podem ser derivados dos domínios de entrada e de saída.

17. **(CESPE / DATAPREV – 2023)** Apesar de primar pela agilidade, testes ágeis exigem processos bem definidos, sob pena de perda de qualidade do produto final.

18. **(CESPE / SERPRO – 2023)** Os mocks são métodos utilizados para realizar testes unitários quando é impossível testar o objeto real, seja porque ele não está disponível, seja porque não é possível executá-lo durante o teste.

19. **(CESPE / SEFIN de Fortaleza-CE – 2023)** Em um teste de integração, cada uma das unidades é testada separadamente para se observar se elas funcionam de forma adequada.

20. **(CESPE / SEFIN de Fortaleza-CE – 2023)** Em um teste funcional de software, os elementos de uma classe devem se comportar de maneira equivalente.

21. **(CESPE / SEFIN de Fortaleza-CE – 2023)** No particionamento de equivalências para a criação de casos de teste, devem ser consideradas apenas as partições válidas.

22. **(CESPE / EMPREL – 2023)** A escolha de casos de teste de unidade é fundamental para a redução dos custos dos testes e uma estratégia que pode ser adotada para a seleção de casos é a de teste de partição de equivalência. Nesse contexto, considere-se um programa que precisa aceitar as seguintes opções de tamanho de folha para impressão.

    - 10,5 cm × 14,8 cm
    - 14,8 cm × 21,0 cm
    - 21,0 cm × 29,7 cm
    - 29,7 cm × 42,0 cm

    Nessa hipótese, para a aplicação correta da técnica de equivalência de partição com o número mínimo de casos de teste, é necessário realizar:

a) apenas um caso de teste, para verificar se o programa pode imprimir folhas em todos os tamanhos exigidos.

b) três casos de teste, para verificar se o programa pode imprimir folhas de 15 cm, 22 cm ou 30 cm.

c) quatro casos de teste, para verificar se o programa pode imprimir folhas em todos os tamanhos exigidos.

d) seis casos de teste, para verificar se o programa pode imprimir folhas em todos os tamanhos exigidos e para tamanhos menores que 10,5 cm e maiores de 29,7 cm.

e) cinco casos de teste, para verificar se o programa pode imprimir folhas em todos os tamanhos exigidos e para tamanhos maiores que 29,7 cm.

23. **(CESPE / MPE-RO – 2023)** O teste de integração de software é responsável por:

a) examinar os módulos desenvolvidos sem considerar acoplamento.
b) iniciar a recuperação de um sistema após uma falha.
c) analisar a forma como um sistema se comporta quando seus recursos são muito acessados.
d) validar, de forma conjunta, os módulos construídos individualmente.
e) verificar se os mecanismos de proteção de um software impedem acessos indevidos.

24. **(CESPE / DATAPREV – 2023)** No teste de acompanhamento, um grupo é designado para verificar quaisquer problemas que necessitem ser resolvidos e quaisquer alterações que devam ser feitas no ambiente de recuperação de desastres.

25. **(CESPE / TBG – 2023)** Os testes dos tipos alfa e beta são executados em um ambiente controlado e com a presença de, pelo menos, um desenvolvedor.

26. **(CESPE / TBG – 2023)** O teste de regressão deve ser efetuado para garantir que novos componentes não tenham causado problema nas funções que antes funcionavam corretamente.

27. **(CESPE / TBG – 2023)** O teste de segurança estático (SAST) trabalha diretamente com o código e é empregado de forma complementar ao teste de segurança dinâmico (DAST).

28. **(CESPE / BANRISUL – 2022)** Na gestão de defeitos, o princípio de teste da regra 10 de Myers estabelece que as atividades de teste estático e dinâmico devem ser planejadas muito antes de serem iniciadas.

29. **(CESPE / BANRISUL – 2022)** No teste de fumaça (smoke test), os códigos do software são integrados em componentes bloqueadores de erros com módulos reutilizáveis necessários para implementar as funções do software.

30. **(CESPE / BANRISUL – 2022)** É possível que um defeito que resida em código sem causar uma falha não seja encontrado em um teste dinâmico.

31. **(CESPE / BANRISUL – 2022)** O teste estático é uma técnica de verificação de software que revisa o código do programa para identificar se ele representa as especificações produzidas para o software.

32. **(CESPE / BANRISUL – 2022)** Os testes de estresse devem verificar o uso da memória ao longo do tempo para garantir que não existam perdas acumulativas.

33. **(CESPE / BANRISUL – 2022)** O objetivo do teste de integração é verificar se os requisitos atendem a especificação e se as funcionalidades do sistema foram implementadas corretamente, sendo todo o sistema testado de modo a simular um ambiente de execução real.

34. **(CESPE / BANRISUL – 2022)** Os testes unitários são realizados com o objetivo de isolar cada parte do sistema para garantir que elas estejam funcionando conforme especificado.

35. **(CESPE / BANRISUL – 2022)** O teste unitário é o processo de testar os componentes de programa, como métodos ou classes de objeto.

36. **(CESPE / BANRISUL – 2022)** Ao se testarem as classes do objeto, devem-se testar as amostras de operações a ele associadas, não havendo necessidade de simular todos os eventos que causam mudança de estado.

37. **(CESPE / BANRISUL – 2022)** Devem ser escolhidos casos efetivos de teste unitário, o que significa que os casos de teste devem mostrar que, quando usado como esperado, o componente que se está testando faz o que ele é proposto a fazer e, se houver defeitos nos componentes, estes devem ser revelados por casos de teste.

38. **(CESPE / TRT-AP-PA – 2022)** A respeito de testes automatizados, no contexto de DevOps e DevSecOps, assinale a opção correta.

a) Em um teste unitário, os métodos da classe sendo testada e suas dependências podem ter relação com recursos externos.

b) Os bugs são detectados no final do ciclo de desenvolvimento, o que pode aumentar o tempo na criação de novos produtos.

c) Os testes unitários são testes de caixa preta com cada função que compõe o software.

d) O TDD (Test Driven Development) eleva o nível dos testes unitários e tem como característica criar a classe de testes antes da classe de produção, de forma que os testes guiem o código a ser implementado.

e) Os testes de integração são caracterizados pela verificação de partes internas do sistema, que se inter-relacionam entre si, conforme definido pelos clientes.

39. **(CESPE / BNB – 2022)** Na seleção de casos para os testes de unidade, uma estratégia eficaz é a do teste baseado em diretriz, em que os casos são escolhidos com base nas indicações geradas a partir de erros mais comuns identificados no desenvolvimento dos programas.

40. **(CESPE / BNB – 2022)** O teste com base em casos de uso é um procedimento efetivo para se alcançar o resultado pretendido com um teste de integração do sistema.

41. **(CESPE / BNB – 2022)** O teste automatizado usualmente é mais apropriado que o teste manual quando a interface do usuário do aplicativo muda consideravelmente em prazos curtos e a automação de teste ainda não está disponível.

42. **(CESPE / TJ-AM – 2019)** Erro e defeito são conceitos distintos: erro pode ser o resultado de uma falha; defeito é uma imperfeição ou inconsistência no produto do software ou em seu processo.

43. **(CESPE / TJ-AM – 2019)** Validação refere-se a um conjunto de atividades destinadas a garantir que o sistema esteja de acordo com os requisitos do usuário.

44. **(CESPE / TJ-AM – 2019)** O teste caixa preta trata o software como uma caixa cujo conteúdo é desconhecido e da qual só é possível visualizar os dados de entrada fornecidos e as respostas produzidas como saída.

45. **(CESPE / TJ-AM – 2019)** O teste de esforço é uma continuidade do teste de carga, e ambos são modalidades do teste de desempenho.

46. **(CESPE / TJ-AM – 2019)** O teste de integração descendente da modalidade primeiro em largura (breadth-first) move-se pela hierarquia de controle e integra todos os componentes em um caminho selecionado como principal.

47. **(CESPE / MC-PA – 2019)** Consoante os princípios dos métodos ágeis, na construção de um sistema, os testes de unidade do código criado devem ser sempre executados primeiramente:

a) pela equipe de testes, somente.
b) pela equipe de testes e pelo gerente do projeto.
c) pelo gerente do projeto, somente.
d) pelo programador.
e) pelo programador com o apoio da equipe de testes.

48. **(CESPE / FUB – 2018)** Os testes de caixa-branca buscam verificar o comportamento interno do software, ou seja, os elementos relacionados ao código-fonte desse software.

49. **(CESPE / BNB – 2018)** Determinada equipe de desenvolvimento de softwares desejava realizar testes que avaliassem o comportamento do sistema por meio do estudo das entradas e das saídas relacionadas, sem validação da implementação do software e sem acesso ao seu código-fonte. Para isso, a equipe sugeriu a utilização dos testes de caixa-preta e de caixa-branca. Nessa situação, somente o teste tipo caixa-preta é corretamente aplicável, pois o tipo caixa-branca depende de acesso ao código-fonte do sistema.

50. **(CESPE / BNB – 2018)** Determinada equipe de desenvolvimento de softwares pretendia realizar testes que permitissem avaliar cenários com os quais os usuários reais do sistema pudessem se relacionar. Esses cenários deveriam descrever uma maneira de usar o sistema. Para isso, foram sugeridos os testes de release e de cenário. Nessa situação, será correto aplicar testes de cenários, que são incompatíveis com os de release, devendo a aplicação desses últimos ser descartada.

51. **(CESPE / BNB – 2018)** O planejamento de testes é governado pela necessidade de selecionar alguns poucos casos de teste de um grande conjunto de possíveis casos. O exame que avalia se um grupo de entrada de dados resultou nas saídas pretendidas, levando-se em consideração a especificação do programa, é denominado teste:

a) de stress.
b) da caixa preta.
c) da caixa branca.
d) da caixa cinza.
e) de integração.

52. **(CESPE / STJ – 2018)** Em um serviço de integração contínua, testes de unidade são executados automaticamente com a finalidade de detectar erros funcionais.

53. **(CESPE / STJ – 2018)** Enquanto os testes de unidade propiciam a qualidade externa, os testes de aceitação ajudam o desenvolvedor a avaliar a qualidade interna do código, dando feedback sobre o design dos módulos e permitindo a manutenção com menor custo.

54. **(CESPE / STJ – 2018)** Teste de software pode ser definido como o processo de execução de um programa ou sistema com a intenção de se verificar se o mesmo está de acordo com o planejado nas especificações dos seus requisitos.

55. **(CESPE / ABIN – 2018)** As ferramentas de execução do teste são classificadas como ferramentas de suporte para execução e registro e têm, como vantagem, o fato de não requererem um grande esforço para a obtenção de resultados expressivos.

56. **(CESPE / ABIN – 2018)** No teste funcional, que é uma das fases do processo de validação, não é necessário o conhecimento das estruturas internas do software.

57. **(CESPE / ABIN – 2018)** No teste de integração, o foco é a comunicação entre os módulos do software, não as suas funcionalidades; portanto, nessa fase, testes funcionais não podem ser utilizados.

58. **(CESPE / STM – 2018)** Em um processo de cascata, testes de sistemas testam todo o sistema, enquanto, em processos de desenvolvimento iterativo, será testado apenas um incremento a ser entregue ao cliente.

59. **(CESPE / CGM DE JOÃO PESSOA-PB – 2018)** O particionamento de equivalência é uma técnica de teste caixa-preta caracterizada por dividir o domínio de entrada de um módulo em classes de equivalência, a partir das quais casos de teste são derivados.

60. **(CESPE / STM – 2018)** Testes de regressão servem ao propósito de verificar se o sistema pode operar na carga necessária, fazendo-a regredir constantemente até que o comportamento de falha do sistema seja testado ou que defeitos sejam identificados.

61. **(CESPE / STM – 2018)** Em testes de integração, a estratégia de integração bottom-up integrará componentes de infraestrutura que fornecem serviços comuns, adicionando a eles componentes funcionais; para testar uma nova característica, pode ser necessário integrar componentes diferentes.

62. **(CESPE / BNB – 2018)** 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.

63. **(CESPE / TRE-BA – 2017)** Durante o planejamento de um projeto e a elaboração dos casos de uso, foram incluídos diversos componentes para cálculos de tributos e da quantidade de recursos orçamentários alocados. De acordo com os níveis de testes existentes, o resultado de um dos componentes em questão poderá ser validado por meio do teste:

a) unitário.
b) de sistema.
c) de aceitação.
d) de codificação.
e) de integração.

64. **(CESPE / TRE-BA – 2017)** O gestor de um órgão organizador de concursos públicos pretende oferecer condições para que mais de um milhão de candidatos inscritos em determinado evento possa obter o gabarito das provas a partir do acesso ao seu sistema eletrônico. Nessa situação, para verificar se o sistema eletrônico suportará uma quantidade grande de acessos simultâneos, a equipe de TI do órgão, ao preparar o ambiente de acesso eletrônico, deverá realizar o teste:

a) de estresse.
b) unitário.
c) integrado.
d) de sistema.
e) de regressão.

65. **(CESPE / TRT-CE – 2017)** A respeito de engenharia de software, assinale a opção correta.

a) A finalidade dos testes de segurança é garantir que o sistema se recupere de uma falha e esteja apto a retomar o processamento em um prazo preestabelecido.

b) Efetuar testes de regressão consiste em reexecutar testes já finalizados para garantir que eventuais alterações não tenham impactado funções que antes funcionavam corretamente.

c) Os testes de integração ascendentes são caracterizados pelo fato de a sua realização ocorrer conforme o desenvolvimento dos módulos.

d) Na etapa de desenvolvimento de um software, os testes de validação e de integração são executados simultaneamente, para identificar inconsistências antes da entrega final.

66. **(CESPE / TJDFT – 2015)** As atividades de validação incluem os testes unitários e os de aceitação.

67. **(CESPE / MEC – 2015)** A automação de testes apresenta maior impacto positivo sobre a realização de testes de regressão do que testes de usabilidade.

68. **(CESPE / MPU – 2013)** Para se avaliar a documentação do projeto do software, deve ser utilizado o teste de unidade.

69. **(CESPE / ANTT – 2013)** O teste de aceitação pode utilizar um processo chamado de teste alfa e beta, sendo conduzido por desenvolvedores e podendo contar com a participação do usuário. O teste alfa é realizado em ambiente real e o beta em ambiente controlado.

70. **(CESPE / TCE-RO – 2013)** Os principais níveis de teste de software são os de caixa branca, os de caixa preta, os de sistema e os de aceitação.

71. **(CESPE / MPU – 2013)** Testes funcionais são aplicados para identificar não conformidades entre o programa e seus requisitos.

72. **(CESPE / MPU – 2013)** Um dos critérios do teste de unidade é o particionamento de equivalência, que consiste no particionamento do domínio de entrada do programa de modo que o conjunto de testes resultantes corresponda a uma representação satisfatória de todo o domínio.

73. **(CESPE / MPU – 2013)** Para realizar testes de unidade ou estrutural, pode-se utilizar uma representação conhecida como grafo de fluxo de controle de um programa. A partir do grafo, executam-se todos os caminhos do programa, principalmente na presença de laços.

74. **(CESPE / INPI – 2013)** De modo geral, o teste de release é um processo de teste do tipo caixa-branca em que as funcionalidades são verificadas e validadas mediante a avaliação interna dos módulos.

75. **(CESPE / MPE-PI – 2012)** Os testes de unidade são feitos por equipes especializadas em testes, de forma a se garantir que os módulos que compõem o sistema sob construção estejam funcionando de acordo com as especificações.

76. **(CESPE / MPE-PI – 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.

77. **(CESPE / MEC – 2011)** Ao ser estabelecido, um plano de testes necessita de diversos insumos, sendo um deles a estratégia de testes.

78. **(CESPE / MEC – 2011)** Na definição do documento referente ao plano de testes, devem ser incluídos os tipos e a metodologia dos testes. No entanto, critérios de aceitação e processos associados fogem ao escopo desse documento e devem ser inseridos na análise dos riscos.

79. **(CESPE / TJ-ES – 2011)** No plano de teste, um documento de nível gerencial, definem-se como o teste vai ser realizado, quem vai executar os testes, o prazo estimado e o nível de qualidade esperado.

80. **(CESPE / MEC – 2011)** O teste denominado caixa-preta é utilizado para verificar se os requisitos do software são atendidos, sem verificar o código ou a lógica do componente testado.

81. **(CESPE / MEC – 2011)** O teste caixa-branca ou teste de caixa de vidro é um método de projeto de casos de teste que usa a estrutura de controle do projeto procedimental para derivar casos de teste. Dessa maneira garante-se que todos os caminhos independentes de um módulo tenham sido exercitados pelo menos uma vez, já que erros lógicos e pressupostos incorretos são inversamente proporcionais à probabilidade de que um caminho de programa vai ser executado.

82. **(CESPE / MEC – 2011)** Quando o objetivo é testar uma funcionalidade, assegurando-se que, para todo tipo de entrada, a saída observada corresponda àquela esperada, pode-se alcançar esse objetivo fazendo-se uso de testes do tipo caixa-branca.

83. **(CESPE / BRB – 2011)** O teste de regressão tem o objetivo de localizar defeitos na estrutura interna do produto, exercitando, suficientemente, os possíveis caminhos de execução do sistema.

84. **(CESPE / SAD-PE – 2010)** A respeito do plano de teste, um registro do processo de planejamento de testes de software, assinale a opção correta.

a) O processo de planejamento de testes é usualmente descrito em um plano de testes.

b) Um plano de teste de software é um registro da execução de um caso de teste de software.

c) A automação de um teste de integração é mais facilmente empreendida que a de um teste de módulo.

d) A produção de scripts de teste deve preceder a eventual construção de casos de teste.

e) Ao se inspecionar o conteúdo de um plano de testes, devem-se encontrar, entre outras, as seguintes descrições: escopo de testes, abordagens de teste, recursos para realização dos testes e cronograma das atividades de teste a serem realizadas.

85. **(CESPE / MPE-TO – 2010)** Entre os diversos níveis possíveis de testes de software, há os chamados testes de unidade (Unit Tests), que procuram testar o programa como um todo, dentro de um contexto totalmente integrado, procurando validar todas as suas potencialidades de forma unificada.

86. **(CESPE / TJ-ES – 2010)** No teste de unidade, o software é forçado a falhar de diversos modos a fim de verificar se os requisitos funcionais foram adequadamente implementados. As unidades, sejam funções, procedimentos, métodos ou classes, são testadas duas a duas. Nesse teste, espera-se identificar erros relacionados a algoritmos incorretos ou mal implementados, estruturas de dados incorretas ou simples erros de programação.

87. **(CESPE / ABIN – 2010)** Nos testes de caixa branca, o código-fonte do programa é usado para identificar testes de defeitos potenciais, particularmente no processo de validação, o qual demonstra se um programa atende a sua especificação.

88. **(CESPE / INMETRO – 2010)** Os testes caixa preta (Black Box) avaliam as cláusulas de código, a lógica interna do componente codificado, as configurações e outros elementos técnicos.

89. **(CESPE / INMETRO – 2010)** Testes de caixa preta são usualmente fundamentados na análise do código de um programa. Por outro lado, entre as técnicas de teste não relacionadas a testes de caixa preta, estão aquelas embasadas na intuição do testador, em especificações comportamentais e no uso.

90. **(CESPE / INMETRO – 2010)** Os testes caixa branca (White Box) verificam a funcionalidade e a aderência aos requisitos, em uma ótica externa ou do usuário, sem que se tenha qualquer conhecimento do código e da lógica interna do componente testado.

91. **(CESPE / INMETRO – 2010)** O teste de caminho básico é uma técnica que identifica as rotinas normalmente usadas, deixando de lado as rotinas eventualmente executadas.

92. **(CESPE / TRE-ES – 2010)** O teste de partições caracteriza-se por ser um projeto de caso de teste, em que o conhecimento da estrutura do programa é utilizado para projetar testes que verificam todas as partes desse programa.

93. **(CESPE / TRE-BA – 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.

94. **(CESPE / TJ-ES – 2010)** O teste de integração, a exemplo do teste caixa-branca, focaliza o esforço de validação na menor unidade de projeto do software e, com o uso de técnicas de componentização, caminhos de controle relevantes são testados para descobrir erros dentro dos limites do componente.

95. **(CESPE / MPU – 2010)** O teste de integração geralmente é um processo de teste de caixa-preta no qual os testes são derivados da especificação do sistema, cujo comportamento pode ser determinado por meio do estudo de suas entradas e saídas.

96. **(CESPE / TJ-ES – 2010)** O teste de caixa-preta é utilizado quando uma nova versão do software está sendo lançada ou quando um novo ciclo de testes for necessário em paralelo ao desenvolvimento do mesmo.

97. **(CESPE / INMETRO – 2010)** Um teste de regressão pode ser o primeiro teste a ser realizado no software.

98. **(CESPE / TRE-BA – 2010)** Se um software já testado receber modificações e, após isso, somente essas modificações forem testadas, a aplicação do teste de regressão a esse software testará inclusive as partes que não tenham sido modificadas.

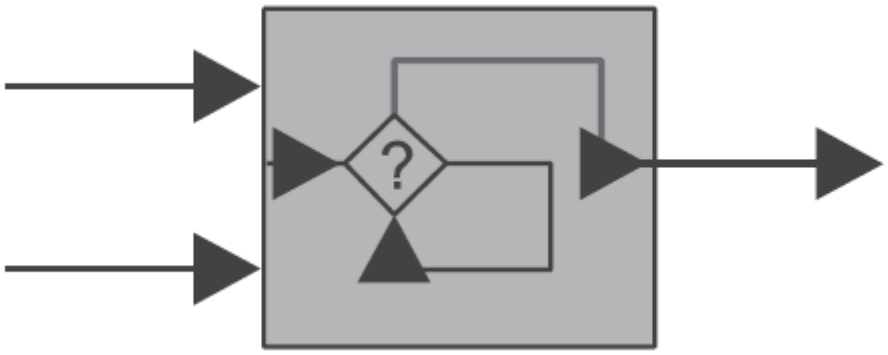
99. **(CESPE / TRE-MT – 2010)** O teste alfa é conduzido pelo cliente em seu ambiente de uso final.

100. **(CESPE / INMETRO – 2010)** Um teste de recuperação deve evitar que o sistema apresente falhas que interrompam o seu funcionamento.

101. **(CESPE / TRE-PR – 2009)** Nos testes de integração, realizados antes dos testes unitários, os componentes são construídos e testados separadamente.

102. **(CESPE / TRE-PR – 2009)** O teste de aceitação envolve a integração de dois ou mais componentes que implementam funções ou características do sistema. Existem duas fases distintas de teste do sistema: testes de integração e teste de caixa de vidro.

103. **(CESPE / TRE-BA – 2009)** A figura a seguir ilustra esquematicamente a técnica estrutural de teste de software (ou teste caixa-branca), que avalia o comportamento interno do componente de software, atuando diretamente sobre o código-fonte do componente para realizar testes de condição, de fluxo de dados, de ciclos e de caminhos lógicos.



104. **(CESPE / TRE-PR – 2009)** Enquanto o teste caixa-preta é estrutural ou orientado à lógica, o teste caixa-branca é funcional, orientado a dado ou orientado a entrada e saída.

105. **(CESPE / TRE-PR – 2009)** Entre os tipos de testes de caixa preta, encontram-se o teste baseado em grafos; o particionamento de equivalência; a análise de valor-limite; e o teste de matriz ortogonal.

106. **(CESPE / CEHAP-PB – 2009)** Aplicado ao final do processo de teste, o teste caixa-preta ou comportamental é baseado nos requisitos funcionais do software.

107. **(CESPE / CEHAP-PB – 2009)** O teste gama envolve a liberação do sistema a uma série de clientes potenciais que concordam em usar esse sistema.

108. **(CESPE / CEHAP-PB – 2009)** O teste alfa, conhecido como teste de aceitação, encerra-se quando cliente e projetista concordam que o sistema é uma implementação aceitável dos requisitos do sistema e não se aplica a sistemas desenvolvidos para um único cliente.

109. **(CESPE / MPE-RR – 2008)** No Processo Unificado, um modelo de teste é tipicamente composto por casos de teste, os quais podem especificar como testar cenários específicos de casos de uso. Os casos de teste tipicamente especificam entradas, resultados esperados e outras condições relevantes para as verificações dos cenários.

110. **(CESPE / IPEA – 2008)** O teste caixa-preta ou comportamental, aplicado no início do processo de teste, é embasado nos requisitos funcionais do software. Identifica, entre outros, erros de iniciação e término, erros de estrutura de dados, erros de interface e funções incorretas ou omitidas.

111. **(CESPE / Hemobrás – 2008)** Teste de usabilidade consiste na análise de um website por um grupo de experts em usabilidade.

112. **(CESPE / Hemobrás – 2008)** As técnicas de avaliação de usabilidade experimentais ou empíricas contam com a participação direta dos usuários e compreendem, basicamente, os testes com usuários por meio do monitoramento de sessões de uso do produto, ou protótipo, em consideração. Em geral, os testes de usabilidade com a participação dos usuários são avaliações confiáveis.

113. **(CESPE / TSE – 2007)** Entre os artefatos produzidos por um processo de teste, têm-se os casos de teste. Um caso de teste é uma situação real de uso, pois não pode ser sintetizado a partir de parâmetros predefinidos.

114. **(CESPE / TSE – 2006)** Os testes são realizados em várias fases de um desenvolvimento. Testes de unidade são de baixo nível, testes de sistema são executados após os de integração, testes beta empregam apenas desenvolvedores.

115. **(CESPE / TSE – 2006)** Um teste de unidade pode ser projetado usando-se uma estratégia caixa branca. Nesse caso, há um foco nos mecanismos internos da unidade sendo testada. A realização de testes caixa branca pode ser apoiada por métricas de cobertura.

116. **(CESPE / PMV – 2005)** O teste de usabilidade em um sítio da Web tem como objetivo identificar problemas de usabilidade e coletar dados relacionados ao desempenho e às preferências dos usuários.

117. **(CESPE / SESPA-PA – 2004)** Para efeito de validação de um software, o beta teste é realizado pelo cliente usuário do software em um ambiente controlado, normalmente nas instalações do desenvolvedor.

118. **(CESPE / STJ – 2004)** Um software-produto, antes de ser lançado no mercado normalmente deve ser testado por usuários reais do sistema. Nessa etapa, configura-se a realização de beta testes.

GABARITO

1. CORRETO
2. CORRETO
3. CORRETO
4. CORRETO
5. LETRA B
6. LETRA D
7. CORRETO
8. LETRA D
9. LETRA D
10. LETRA E
11. LETRA C
12. CORRETO
13. CORRETO
14. CORRETO
15. ERRADO
16. CORRETO
17. ANULADA
18. ANULADA
19. ERRADO
20. CORRETO
21. ERRADO
22. LETRA C
23. LETRA D
24. CORRETO
25. ERRADO
26. CORRETO
27. ERRADO
28. ERRADO
29. ERRADO
30. CORRETO
31. CORRETO
32. ERRADO
33. ERRADO
34. CORRETO
35. CORRETO
36. ERRADO
37. CORRETO
38. LETRA D
39. CORRETO
40. CORRETO
41. ERRADO
42. CORRETO
43. CORRETO
44. CORRETO
45. CORRETO
46. ERRADO
47. LETRA D
48. CORRETO
49. CORRETO
50. ERRADO
51. LETRA B
52. CORRETO
53. ERRADO
54. CORRETO
55. ERRADO
56. CORRETO
57. ERRADO
58. CORRETO
59. CORRETO
60. ERRADO
61. CORRETO
62. ERRADO
63. LETRA A
64. LETRA A
65. LETRA B
66. ANULADO
67. CORRETO
68. ERRADO
69. ERRADO
70. ERRADO
71. CORRETO
72. ERRADO
73. ERRADO
74. ERRADO
75. ERRADO
76. CORRETO
77. CORRETO
78. ERRADO
79. CORRETO
80. CORRETO
81. CORRETO
82. ERRADO
83. ERRADO
84. LETRA E
85. ERRADO
86. ERRADO
87. ERRADO
88. ERRADO
89. ERRADO
90. ERRADO
91. ERRADO
92. ERRADO
93. CORRETO
94. ERRADO
95. ERRADO
96. ERRADO
97. ERRADO
98. CORRETO
99. ERRADO
100. ERRADO
101. ERRADO
102. ERRADO
103. CORRETO
104. ERRADO
105. CORRETO
106. CORRETO
107. ERRADO
108. ERRADO
109. CORRETO
110. ERRADO
111. ERRADO
112. CORRETO
113. ERRADO
114. ERRADO
115. CORRETO
116. CORRETO
117. ERRADO
118. CORRETO

LISTA DE QUESTÕES – FCC

1. **(FCC / SANASA CAMPINAS – 2019)** Considere que está em desenvolvimento um projeto de software na SANASA e os Analistas optaram pela reexecução de alguns subconjuntos de testes que já foram conduzidos para garantir que as modificações não tenham propagado efeitos colaterais no software. Este tipo de teste ajuda a garantir que mudanças não insiram erros e comportamentos indesejados e é denominado:

a) Regressão.
b) Fumaça.
c) Unidade.
d) Alfa.
e) Showstopper.

2. **(FCC / SEMEF MANAUS-AM – 2019)** Considerando a realização de testes de caixa branca e preta de software, a equipe técnica deve considerar que o teste de caixa:

a) preta não visa testar a estrutura lógica interna do módulo de software sob teste.
b) branca deve ser feito somente com o sistema completo, com todos os módulos integrados.
c) preta é um teste que exclui do programa o código testado.
d) preta deve ser executado no modo de segurança do compilador em uso.
e) branca visa testar apenas a interface de cada módulo de software.

3. **(FCC / SEMEF MANAUS-AM – 2019)** Ao realizar testes de unidade de módulos de software, um técnico de TI deve atentar que:

a) um módulo pseudocontrolador é um módulo que contém apenas o número da versão do módulo sob teste.
b) não é necessária a utilização de módulos pseudocontrolados, mas apenas de pseudocontroladores.
c) um módulo pseudocontrolador substitui módulos chamados pelo módulo sob teste.
d) não é necessária a utilização de módulos pseucontroladores, mas apenas de pseudocontrolados.
e) se admite apenas um módulo pseudocontrolado para cada módulo sob teste.

4. **(FCC / SEMEF MANAUS-AM – 2019)** A equipe de teste de software deve ter bem entendido que um dos objetivos principais de um teste de software é:

a) determinar o nível de qualidade do software sob análise.
b) reduzir o tamanho do código fonte do software sob análise.
c) detectar falhas ou defeitos no software, de acordo com o estabelecido em sua especificação.
d) demonstrar que o software sob análise não é cópia de outro software.

e) verificar se o software sob análise não contém dados sigilosos.

5. **(FCC / SEMEF MANAUS-AM – 2019)** Uma equipe de assistentes técnicos está encarregada de realizar os testes do software referente a um projeto. Dessa forma, essa equipe deve considerar que há um tipo de teste de software, no qual são reexecutados conjuntos de testes já realizados, de forma a garantir que a adição de novos módulos de software em um teste de integração não introduza erros até então inexistentes. Tal tipo de teste denomina-se:

a) de regressão.
b) ascendente.
c) descendente.
d) fracionado.
e) integral.

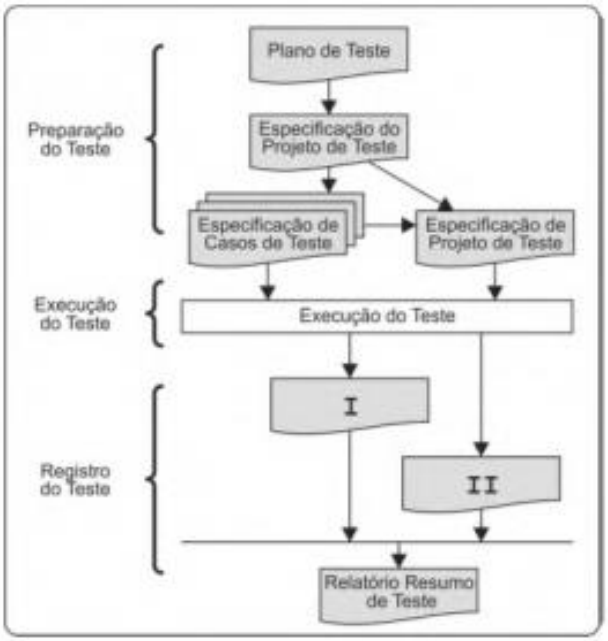
6. **(FCC / SEMEF MANAUS-AM – 2019)** A Fazenda Municipal aplica, em seus projetos de software, as práticas de construção de software, dentre as quais está a codificação, que conta com três princípios fundamentais: de preparação, de codificação propriamente dita e de validação, sendo certo que:

a) entender a arquitetura do software é um dos princípios de preparação.
b) conduzir inspeções de código é um dos princípios de validação.
c) selecionar nomes significativos para as variáveis do software é um dos princípios de preparação.
d) entender os princípios e conceitos básicos do projeto é um dos princípios de preparação.
e) realizar testes unitários e corrigir erros do software é um dos princípios de codificação propriamente dita.

7. **(FCC / SEFAZ-BA – 2019)** Suponha que uma Auditora Fiscal da área de TI atue na etapa de testes e avaliação da qualidade de um software em desenvolvimento. Como o software sofria alterações a cada nova funcionalidade a ele incorporada, a Auditora propôs que a equipe de testes adotasse como padrão um tipo de teste que garantisse que as mudanças recentes no código deixassem o resto do código intacto, visando impedir a introdução de erros. A equipe decidiu realizar um tipo de teste para testar a parte modificada e as áreas adjacentes que podem ter sido afetadas, dentro de uma abordagem baseada em risco. Assim, os testadores destacariam as áreas de aplicação que poderiam ser afetadas pelas recentes alterações de código e selecionariam os casos de testes relevantes para o conjunto de testes. Procedendo desta forma, seriam realizados testes:

a) de Revisão de Funcionalidade.
b) Gama.
c) de Aceite Operacional.
d) de Regressão.
e) de Caixa-preta.

8. **(FCC / SEFAZ-BA – 2019)** Considere o procedimento apresentado na figura a seguir, no qual são utilizados documentos consistentes e adequados capazes de definir, registrar e prover condições de análise dos resultados obtidos ao longo do processo de testes de software.



Na etapa de Registro do Teste, I corresponde ao:

a) Registro de Testes Verde-Vermelho e II corresponde aos Critérios para Homologação de Teste.
b) Log de Teste e II corresponde ao Relatório de Incidentes de Teste.
c) Gap de Teste e II corresponde à Refatoração de Testes.
d) Registro de Testes Funcionais e Não Funcionais e II corresponde ao Relatório de Análise Ciclomática de Teste.
e) Relatório de Testes de Usabilidade e II corresponde ao Relatório de Testes Sincronizados.

9. **(FCC / SEFAZ-SC – 2018)** Os testes unitários são aplicados em subprogramas individuais ou em componentes maiores construídos com unidades altamente coesas e são executados:

I. sempre com acesso ao código que está sendo testado.
II. normalmente com o suporte de ferramentas de depuração.
III. sempre pelos programadores que escreveram o código.
IV. para verificar o funcionamento dos elementos de software separadamente.

Está correto o que consta de:

a) I e IV, apenas.
b) I, II e III, apenas.
c) I, II e IV, apenas.

d) II, III e IV, apenas.
e) I, II, III e IV.

10. **(FCC / DP-AM – 2018)** Considere, por hipótese, que na Defensoria esteja sendo desenvolvido um projeto com prazo crítico, sendo necessário que os desenvolvedores avaliem o software frequentemente. A equipe envolvida decidiu utilizar uma abordagem de teste de integração que trabalha da seguinte maneira:

I. Componentes necessários para implementar funções do software, como arquivos de dados, bibliotecas, módulos reutilizáveis etc são integrados em uma build (construção).

II. Diversos testes são projetados para que erros que possam impedir a build em andamento de desempenhar de forma adequada sua função, com o objetivo de descobrir showstoppers que impliquem em atrasos no cronograma.

III. A build é integrada a outras builds e todo o software passa diariamente por este tipo de teste, podendo usar abordagem ascendente ou descendente de integração.

O teste de integração descrito é denominado teste:

a) de fumaça.
b) de regressão.
c) top-down.
d) breadth-first.
e) de caixa cinza (grey box).

11. **(FCC / TCM-GO – 2015)** Um Auditor de Controle Externo do Tribunal de Contas dos Municípios do Estado de Goiás da área de TI indicou a seguinte estratégia convencional para testes de um sistema que está sendo desenvolvido:

I. Para cada componente ou módulo, testar a interface, a estrutura de dados local, os caminhos independentes ao longo da estrutura de controle e as condições-limite para garantir que a informação flui adequadamente para dentro e para fora do módulo, que todos os comandos tenham sido executados e que todos os caminhos de manipulação de erros sejam testados.

II. Aplicar uma abordagem incremental de testes para a construção da arquitetura do sistema, de forma que os módulos testados sejam integrados a partir do módulo de controle principal e os testes sejam conduzidos à medida que cada componente é inserido.

O Auditor indicou em I e II, respectivamente, os testes de:

a) caixa branca e de caixa preta, que são suficientes para validar todo o sistema.

b) unidade e de integração; na sequência, indicou os testes de validação e de sistema que são adequados para validar todo o sistema.

c) unidade e de interoperabilidade; na sequência, indicou os testes de caixa branca e de caixa preta que são adequados para validar todo o sistema.

d) carga e de desempenho; na sequência, indicou os testes de usabilidade e interoperabilidade que são adequados para validar todo o sistema.

e) caixa preta e de caixa branca, que são suficientes para validar todo o sistema.

12. **(FCC / TCM-GO – 2015)** Um Auditor de Controle Externo do Tribunal de Contas dos Municípios do Estado de Goiás da Área de TI recebeu a tarefa de identificar testes que sejam capazes de verificar:

– a validade funcional do sistema;
– o comportamento e o desempenho do sistema;
– quais classes de entrada vão constituir bons casos de teste;
– se o sistema é sensível a certos valores de entrada;
– quais taxas e volumes de dados o sistema pode tolerar;
– que efeito combinações específicas de dados terão na operação do sistema.

A indicação correta do Auditor é utilizar:

a) testes de caixa branca.
b) mais de um tipo de teste, pois não há um único tipo de teste capaz de avaliar todas estas situações.
c) um tipo diferente de teste para cada uma das situações elencadas.
d) testes de caixa preta.
e) testes de desempenho para os 2 primeiros e de carga para os demais.

13. **(FCC / TRT1 – 2014)** Considerando o teste de software, há o chamado teste de unidade, que consiste em testar:

a) o software completo, incluindo todos os seus componentes ou módulos, no ambiente de testes.
b) o funcionamento dos compiladores que estiverem sendo utilizados no desenvolvimento do software.
c) individualmente, componentes ou módulos de software que, posteriormente devem ser testados de maneira integrada.
d) o software completo em seu ambiente final de operação, já com o hardware base do projeto.
e) apenas componentes ou módulos de software cujo código fonte tenha mais de 100 linhas.

14. (FCC / DPE-SP – 2013) Para aplicações convencionais, o software é testado a partir de duas perspectivas diferentes: a lógica interna do programa é exercitada usando técnicas de projeto de caso de teste ...I... e os requisitos de software são exercitados usando técnicas de projeto de casos de teste ...II... .

O teste ...I... fundamenta-se em um exame rigoroso do detalhe procedimental. Os caminhos lógicos do software e as colaborações entre componentes são testados exercitando conjuntos específicos de condições e/ou ciclos.

O teste ...II... faz referência a testes realizados na interface do software. Esse tipo de teste examina alguns aspectos fundamentais de um sistema, com pouca preocupação em relação à estrutura lógica interna do software.

As lacunas I e II são preenchidas correta e respectivamente, com:

a) de caminho básico - caixa-de-vidro
b) alfa - beta
c) caixa branca - caixa preta
d) de ciclo - de usabilidade
e) unitário - de interface

15. (FCC / TRT-PE – 2012) No que se refere a testes de software, é correto afirmar que:

a) o teste de operação é a fase onde é testada a ergonomia da interface de uso do software.

b) o teste da caixa preta (teste funcional), baseia-se em analisar os arquivos de log do sistema procurando por mensagens de funcionamento inconsistente.

c) um teste bem sucedido é um teste que não encontra nenhum erro no software.

d) o teste da caixa branca (teste estrutural), baseia-se em testar as estruturas do código fonte, como comandos condicionais e de repetição.

e) um caso de teste é uma categoria de possíveis resultados na execução de testes.

16. (FCC / TRE-PE – 2011) Com relação aos testes de software, é correto afirmar:

a) Um princípio muitas vezes adotado ao testar um software é o de Pareto. Ele afirma que existe um forte desequilíbrio entre causas e efeitos, entre esforços e resultados e entre ações e objetivos alcançados.

b) Testes sempre podem mostrar a ausência de erros.

c) Para que o resultado de um teste de software seja confiável, é preciso garantir que os casos de teste utilizados cubram um número reduzido de possibilidades de execução.

d) Um software que produz saídas corretas deve ser aprovado, pois isso demonstra que todos os erros foram corrigidos.

e) Um programador deve testar seu próprio código porque facilmente conseguirá criar um caso de teste que rompe com a lógica de funcionamento do seu código.

17. **(FCC / TRE-PE – 2011)** Com relação aos testes de software, é correto afirmar:

a) Um princípio muitas vezes adotado ao testar um software é o de Pareto. Ele afirma que existe um forte desequilíbrio entre causas e efeitos, entre esforços e resultados e entre ações e objetivos alcançados.

b) Testes sempre podem mostrar a ausência de erros.

c) Para que o resultado de um teste de software seja confiável, é preciso garantir que os casos de teste utilizados cubram um número reduzido de possibilidades de execução.

d) Um software que produz saídas corretas deve ser aprovado, pois isso demonstra que todos os erros foram corrigidos.

e) Um programador deve testar seu próprio código porque facilmente conseguirá criar um caso de teste que rompe com a lógica de funcionamento do seu código.

18. **(FCC / INFRAERO – 2011)** Na direção dos tipos de teste focados pela engenharia de software, os testes de integração cuidam dos tópicos associados com os problemas de verificação:

a) da engenharia de sistemas.
b) do projeto do software.
c) dos códigos do programa.
d) dos requisitos funcionais.
e) dos requisitos não funcionais.

19. **(FCC / TRT14 – 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.

20. **(FCC / TRT9 – 2010)** O teste de sistema que força o software a falhar de diversos modos e verifica o retorno do processamento dentro de um tempo pré-estabelecido é um tipo de teste de:

a) Integração.
b) Estresse.
c) Recuperação.
d) Desempenho.
e) Segurança.

21. **(FCC / SEFAZ-SP – 2009)** Garantir que um ou mais componentes de um sistema combinados funcionam corretamente é o objetivo do tipo de teste:

a) de sistema.
b) de integração.
c) de configuração.
d) operacional.
e) funcional.

22. **(FCC / TRT15 – 2009)** Os testes de integração têm por objetivo verificar se:

a) os módulos testados produzem os mesmos resultados que as unidades testadas individualmente.
b) os módulos testados suportam grandes volumes de dados.
c) as funcionalidades dos módulos testados atendem aos requisitos.
d) os valores limites entre as unidades testadas individualmente são aceitáveis.
e) o tempo de resposta dos módulos testados está adequado.

23. **(FCC / TRT-MG – 2009)** NÃO se trata de uma técnica para testar software o teste de:

a) caixa preta.
b) regressão.
c) desempenho.
d) unidade.
e) carga

24. **(FCC / TRT-MA – 2009)** Há um tipo de teste que vislumbra a "destruição do programa" por meio de sua submissão a quantidades, frequências ou volumes anormais que é o teste:

a) de recuperação.
b) de configuração.
c) beta.
d) de desempenho.
e) de estresse.

25. **(FCC / AFR-SP – 2009)** Garantir que um ou mais componentes de um sistema combinados funcionam corretamente é o objetivo do tipo de teste:

a) funcional.
b) de sistema.
c) de integração.
d) de configuração.
e) operacional.

26. **(FCC / TRT-GO – 2008)** Uma sistemática para construção da arquitetura do software enquanto, ao mesmo tempo, conduz ao descobrimento de erros associados às interfaces é a estratégia de teste de software denominada de:

a) sistema.
b) unidade.
c) validação.
d) arquitetura.
e) integração.

27. **(FCC / METRÔ-SP – 2008)** Um critério de teste de software baseado no fluxo de dados de aplicação pode ser utilizado como uma técnica de teste baseada:

a) na especificação.
b) no código.
c) em falhas.
d) no uso da aplicação.
e) na intuição e experiência do engenheiro.

GABARITO

1.	LETRA A	10.	LETRA A	19.	LETRA A
2.	LETRA A	11.	LETRA B	20.	LETRA C
3.	LETRA C	12.	LETRA D	21.	LETRA B
4.	LETRA C	13.	LETRA C	22.	LETRA C
5.	LETRA A	14.	LETRA C	23.	LETRA D
6.	LETRA D	15.	LETRA D	24.	LETRA E
7.	LETRA D	16.	LETRA A	25.	LETRA C
8.	LETRA B	17.	LETRA A	26.	LETRA E
9.	LETRA C	18.	LETRA B	27.	LETRA B

LISTA DE QUESTÕES – FGV

1. **(FGV / TJ-MS – 2024)** A testabilidade é um fator importante para o desenvolvimento e a implementação de um software. Uma característica de um software testável é que os estados do software devem ser visíveis e podem ser consultados durante a execução. Essa característica é chamada de:

   a) compreensibilidade;
   b) controlabilidade;
   c) estabilidade;
   d) observabilidade;
   e) operabilidade.

2. **(FGV / SES-MT – 2024)** Existem várias técnicas de teste de software, cada uma com seus próprios objetivos e métodos específicos. Relacione os testes de software listados a seguir, às suas respectivas definições.

   1. Teste Funcional
   2. Teste de Conformidade
   3. Teste de Caixa Preta
   4. Teste de Desempenho

   ( ) Avalia como o sistema se comporta em termos de velocidade, escalabilidade e estabilidade sob diferentes condições de carga.
   ( ) Testa o software sem conhecimento interno da lógica ou estrutura do código, focando nos requisitos e funcionalidades visíveis.
   ( ) Verifica se as funções do software estão operando conforme esperado. Isso pode incluir testes de casos de uso, fluxos de trabalho e requisitos funcionais.
   ( ) Verifica se o software atende a padrões, regulamentos e requisitos legais.

   Assinale a opção que indica a relação correta, na ordem apresentada.

   a) 2 – 4 – 3 – 1
   b) 2 – 1 – 3 – 4.
   c) 4 – 3 – 1 – 2.
   d) 4 – 2 – 1 – 3.

3. **(FGV / SES-MT – 2024)** A excelência no desenvolvimento de software começa com testes meticulosos, assim como a precisão é crucial em uma obra de arte. Diversas técnicas de teste de software existem, cada uma com propósito e abordagem específicos. Assinale a opção que indica o tipo de teste que assegura que as funcionalidades previamente implementadas não serão afetadas pelas alterações feitas no código:

a) Teste Unitário.
b) Teste de Integração.
c) Teste de Usabilidade.
d) Teste de Regressão.

4. **(FGV / AL-PR – 2024)** No contexto da Engenharia de Software, os testes de software desempenham um papel consideravelmente importante no âmbito do processo de desenvolvimento. Nesse contexto, os testes de unidade caracterizam-se por:

a) dependerem do sucesso do teste de integração descendente (top-down).
b) representarem validações em componentes que representam programas independentes.
c) serem responsáveis pela verificação na menor unidade do componente ou módulo de software.
d) substituírem paulatinamente a abordagem de teste por fumaça na verificação do software.
e) utilizarem como metodologia de base a abordagem de testes de regressão.

5. **(FGV / CGE-SC – 2023)** O tipo de teste de software que serve para garantir que todas ou algumas partes de um sistema estão dialogando e funcionando corretamente em conjunto é o teste:

a) de regressão.
b) de aceitação.
c) de integração.
d) de validação.
e) unitário.

6. **(FGV / SEFAZ-MT - 2023)** A estratégia de teste software cujo objetivo principal é verificar como um dado software se comporta em um cenário que exige recursos computacionais em quantidades, frequência ou volumes anormais é o teste de:

a) estresse.
b) integração.
c) regressão.
d) unidade.
e) usabilidade.

7. **(FGV / TCE-TO – 2022)** O analista de sistemas Carlos está desenvolvendo o software CharlieApp e implementou o teste C. O teste C consiste apenas em determinar se o método A do código de CharlieApp retorna o resultado esperado C ao chamar o método B que realiza uma consulta ao banco de dados de CharlieApp.

Portanto, o teste C implementado por Carlos é de:

a) unidade;
b) aceitação;

c) ponta a ponta;
d) exploração;
e) integração.

8. **(FGV / IMBEL - 2021)** Com referência às metodologias de teste de software, a técnica que avalia as funcionalidades sem ter contato com o código-fonte, mas apenas com as respostas que o sistema dá a determinadas ações, é conhecida como:

a) Caixa Branca.
b) Caixa Cinza.
c) Caixa Preta.
d) Regressão.
e) Testes não funcionais.

9. **(FGV / FUNSAÚDE-CE - 2021)** No contexto da testagem de software, os testes do tipo Unitário, aplicam-se normalmente:

a) à aderência a padrões.
b) às funções codificadas.
c) às interfaces de entrada de dados.
d) à integração dos componentes.
e) aos limites de carga.

10. **(FGV / TCE-AM - 2021)** A Equipe de Desenvolvimento de Software (EDS) de um tribunal de contas está trabalhando na construção de componentes de um novo sistema de software.

Para verificar o funcionamento do software no nível de componente, a EDS deverá aplicar testes de caixa:

a) branca, para validar parâmetros de entrada;
b) preta, para garantir que caminhos independentes dos componentes tenham sido testados;
c) branca do tipo análise de valor-limite;
d) preta como alternativa a testes de caixa branca;
e) branca, para exercitar decisões lógicas em seus lados verdadeiro e falso.

11. **(FGV / DPE-RJ – 2019)** Uma empresa foi contratada por um órgão governamental para modificar e adaptar um sistema para gerenciamento eletrônico de documentos, com base nas especificações criadas pelo próprio órgão. A contratada entregou ao órgão uma parte do sistema com as alterações solicitadas, e um grupo de usuários finais do sistema está simulando operações de rotina, para atestar se seu comportamento está de acordo com as expectativas da empresa.

Conclui-se que está sendo realizado o teste de:

a) unidade;
b) regressão;
c) integração;
d) aceitação;
e) cobertura.

12. **(FGV / DPE-RJ – 2019)** No processo de validação de software, quando os componentes individuais são avaliados para garantir que eles possam operar corretamente, sendo testados independentemente, isto é, sem a presença de outros componentes do sistema, isto é conhecido como teste de:

a) módulo.
b) aceitação.
c) subsistema.
d) unidade.
e) sistema.

13. **(FGV / AL-RO – 2018)** O teste de software que visa verificar que, por exemplo, a correção de uma falha (ou bug) não introduziu uma nova falha (ou bug), é o teste:

a) revisional.
b) de integração.
c) funcional.
d) de regressão.
e) de recuperação.

14. **(FGV / MPE-AL – 2018)** Eduardo é o líder técnico do Sistema de Vendas de uma rede de farmácias. O sistema deve ser utilizado em mais de 40 unidades espalhadas por vários estados. O sistema entrou em produção e, já na primeira semana de uso, ficou muito lento e diversas vezes indisponível para os operadores das lojas. Diante deste cenário, assinale a opção que indica a técnica de teste que foi negligenciada:

a) de fumaça.
b) funcional de limite.
c) de desempenho.
d) caixa-branca.
e) de análise de valor-limite.

15. **(FGV / BANESTES – 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.

16. **(FGV / BANESTES – 2018)** O termo "Alfa teste" caracteriza testes de software que:

a) empregam primordialmente técnicas conhecidas como "Black box";
b) são equivalentes aos testes conhecidos pelo termo "Beta teste";
c) focam em pontos como performance e confiabilidade;
d) são realizados em ambientes controlados pelo desenvolvedor;
e) são realizados por usuários externos, em condições de uso semelhantes às de produção.

17. **(FGV / COMPESA - 2018)** Com relação à análise estática de código, considere as afirmativas a seguir.

I. É um tipo de teste de software.

II. Visa detectar e corrigir defeitos existentes em programas.

III. É capaz de detectar defeitos do tipo bad smell (termo que significa incorreções técnicas ou anomalias que não evitam o sistema de ser executado, mas causam efeitos inesperados durante a execução).

Está correto o que se afirma em:

a) I, somente.
b) II, somente.
c) III, somente.
d) I e II, somente.
e) I, II e III.

18. **(FGV / IBGE - 2017)** Testes devem ser realizados durante o desenvolvimento de um sistema computacional para garantir a qualidade e detectar falhas antes que ele seja disponibilizado para os seus usuários finais. Analise as afirmativas a seguir sobre tipos de teste de software:

I. Teste de caixa preta é uma técnica de teste onde o código-fonte e a estrutura interna do sistema são considerados para modelar os casos de teste.

II.Teste de regressão tem a função de verificar se alguma modificação recente causou algum efeito indesejado e certificar se o sistema ainda atende aos requisitos.

III.Teste de desempenho foca na experiência do usuário, ergonomia da interface e acesso às funcionalidades.

Está correto o que se afirma em:

a) somente I.
b) somente II.
c) somente III.
d) somente I e III.
e) I, II e III.

19. **(FGV / ALERJ – 2017)** A atividade de teste de software contribui para revelar defeitos latentes nos programas. Em relação às técnicas de testes de software, é correto afirmar que:

a) testes de caixa branca têm por objetivo testar o código-fonte, testar cada linha de código possível, testar os fluxos básicos e os alternativos;

b) testes de regressão têm por objetivo verificar se o sistema se mantém funcionando de maneira satisfatória após longos e intensos períodos de uso;

c) todas as declarações internas do programa devem ser testadas pelo menos uma vez durante os testes funcionais;

d) testes de unidade se preocupam em exercitar o sistema além de sua carga máxima de projeto, até que ele falhe;

e) testes de usabilidade verificam se o software instala como planejado, em diferentes hardwares e sob diferentes condições.

20. **(FGV / IBGE – 2016)** Os testes de aceitação são muitas vezes a última etapa de testes antes de implantar o software em produção. Seu objetivo maior é verificar se o software está apto para utilização por parte dos usuários finais, de acordo com os requisitos de implementação definidos. Há três estratégias de implementação de testes de aceitação: a aceitação formal, a aceitação informal (ou teste alfa) e o teste beta.

Com relação às três estratégias de implementação dos testes de aceitação, é correto afirmar que:

a) o teste de aceitação informal, ou teste alfa, é conduzido nas instalações do usuário final, geralmente sem a presença do desenvolvedor;

b) o teste beta é conduzido na instalação do desenvolvedor por um grupo representativo de usuários finais;

c) o teste de aceitação formal utiliza todo o conjunto de casos de teste aplicados durante o teste do sistema, para procurar novos problemas;

d) o teste beta é focado na busca de defeitos e seu progresso é facilmente medido;

e) o teste de aceitação formal pode ser realizado de forma automatizada.

21. **(FGV / IBGE – 2016)** Trata-se de um teste que desconhece o conteúdo do código fonte. Nesse teste o componente testado é tratado como uma caixa preta: são fornecidos dados de entrada e o resultado comparado com aquele esperado e previamente conhecido. Além disso, esse teste pode ser aplicado em diversas fases de teste. A questão retrata características do teste:

a) funcional;
b) de integração;
c) de desempenho;
d) de carga;
e) unitário.

22. **(FGV / Prefeitura de Paulínia - SP - 2016)** A equipe de desenvolvimento de sistemas da empresa "Sistemas Unidos" está trabalhando em um software com a utilização do processo unificado. Seguindo essa metodologia, as equipes realizam diversas disciplinas ao longo do desenvolvimento, dentre as quais estão os testes. A partir deste momento, a equipe deverá avaliar como os módulos trabalham em conjunto.

A equipe estará realizando os testes do tipo:

a) aceitação.
b) unitário.
c) integração.
d) sistema.
e) regressão.

23. **(FGV / TJ-PI - 2015)** A equipe de desenvolvimento da empresa "Sosistemas" utiliza o modelo V para desenvolver seus sistemas de informação. Seguindo essa metodologia, as equipes realizam diversos tipos de testes ao longo do desenvolvimento. No momento atual, o funcionário José está testando um grupo de classes para avaliar seu funcionamento em conjunto. Para fazer essa avaliação, José está realizando testes do tipo:

a) unitário.
b) de integração.
c) de aceitação.
d) de segurança.
e) de carga.

24. **(FGV / DPE-RJ - 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.

25. **(FGV / PROCEM-PA – 2014)** A verificação dinâmica está baseada nas três dimensões de testes, listadas a seguir: tipos de teste, técnicas de teste e níveis de teste. Assinale a opção que apresenta somente itens da dimensão tipos de teste.

a) Teste de Aceitação – Teste de Regressão – Teste Estrutural
b) Teste de Funcionalidade – Teste de Desempenho – Teste de Unidade
c) Teste de Interface – Teste de Carga – Teste de Segurança
d) Teste Funcional – Teste de Volume – Teste de Sistema
e) Teste de Usabilidade – Teste de Funcionalidade – Teste de Integração

26. **(FGV / FIOCRUZ – 2010)** Um tipo de teste de sistemas de software é também chamado de "teste comportamental" e focaliza os requisitos funcionais do software, permitindo ao engenheiro de software derivar conjuntos de condições de entrada que vão exercitar plenamente todos os requisitos funcionais de um programa. Esse tipo de teste tende a ser aplicado durante os últimos estágios do teste e tenta encontrar erros em funções incorretas ou omitidas, de interfaces, de estrutura de dados ou de acesso à base de dados externa, de comportamento ou desempenho de iniciação e término. Além disso, é um tipo de teste que despreza, de propósito, a estrutura de controle, sendo a atenção focalizada no domínio da informação. Esse tipo é conhecido por teste:

a) caixa-preta.
b) caixa-branca.
c) de fluxo de dados.
d) de caminho básico.
e) de lógica composta.

27. **(FGV / BADESC – 2010)** O teste de software que projeta casos de testes derivados do conhecimento da estrutura e da implementação do software é conhecido por:

a) teste de releases.
b) teste caixa-claro.
c) teste caixa-preta.
d) teste de aceitação.

e) teste de integração.

GABARITO

1. LETRA D
2. LETRA C
3. LETRA D
4. LETRA C
5. LETRA C
6. LETRA A
7. LETRA E
8. LETRA C
9. LETRA B
10. LETRA E
11. LETRA D
12. LETRA D
13. LETRA D
14. LETRA C
15. LETRA E
16. LETRA D
17. LETRA E
18. LETRA B
19. LETRA A
20. LETRA E
21. LETRA A
22. LETRA C
23. LETRA B
24. LETRA B
25. LETRA C
26. LETRA A
27. LETRA B

LISTA DE QUESTÕES – DIVERSAS BANCAS

1. **(CESGRANRIO / IPEA – 2024)** Uma desenvolvedora foi contratada para a equipe de desenvolvimento de uma empresa e teve, como primeira tarefa, estudar sobre stub. Ao pesquisar sobre o assunto, ela encontrou uma boa definição para esse termo, que explicava que stub é(são):

   a) a implementação real, mas não necessariamente igual à implementação que estará no ambiente de produção.

   b) a implementação que permite fornecer respostas prontas, sendo usada nas situações em que se deseja validar apenas o resultado.

   c) a técnica que permite criar métodos com o mesmo nome em uma mesma classe, e o que varia entre os métodos escritos são os tipos de informações que poderão receber em seus parâmetros.

   d) os objetos fornecidos, mas não utilizados, sendo geralmente usados no preenchimento da lista de parâmetros.

   e) os objetos pré-programados que verificam se um ou mais métodos foram ou não chamados, a ordem de chamadas, se foram chamados com os argumentos certos e quantas vezes foram chamados.

2. **(CESGRANRIO / TRANSPETRO – 2023)** Uma equipe de desenvolvimento pretende convidar um conjunto de representantes dos usuários finais de uma solução, que aparentemente está completamente desenvolvida, para testar sua última versão antes de disponibilizá-la no mercado. Caso essa versão seja aprovada, será colocada em produção. A situação descrita faz referência aos testes:

   a) unitários
   b) de aceitação
   c) de desempenho
   d) de integração
   e) de sistema

3. **(CESGRANRIO / TRANSPETRO – 2023)** O principal objetivo dos testes estáticos é o de reduzir os defeitos de um software por meio da redução de defeitos na documentação a partir da qual o software foi desenvolvido. Uma das técnicas mais importantes para a execução de testes estáticos é a de:

   a) regressão
   b) walk-through

c) análise do valor limite
d) análise de casos de uso
e) particionamento de equivalência

4. **(CESGRANRIO / IPEA – 2024)** Uma nova funcionalidade acabou de ser desenvolvida para um software que tem sido usado há anos por uma empresa. A líder da equipe de desenvolvimento informou à equipe a relevância de executar um conjunto de testes a toda nova versão desse software, a fim de garantir que mudanças realizadas nas novas versões não impactem o restante do sistema. Com base no cenário descrito, qual(is) teste(s) valida(m) se o que foi criado em versões anteriores desse mesmo software continua funcionando a partir de mudanças em suas outras funcionalidades?

a) Carga
b) Stress
c) Regressão
d) Portabilidade
e) Não funcionais

5. **(CESGRANRIO / IPEA – 2024)** Um desenvolvedor de sistemas, ao analisar algumas ferramentas de testes, deparou-se com o xUnit, o qual, dentre outras características, destaca-se por:

a) suportar a criação de testes unitários e de testes de interface para as linguagens .NET e Swift.
b) ser código fechado e ser voltado a criar testes unitários para as linguagens orientadas a objetos.
c) ser código aberto e permitir a criação de testes unitários para linguagens .NET.
d) ser uma biblioteca gratuita que permite a criação de testes unitários para aplicações desenvolvidas em Java e Swift.
e) ser uma biblioteca gratuita e de código fechado que permite a criação de testes unitários e funcionais para diversas linguagens, como, por exemplo, C# e F#.

6. **(CESGRANRIO / IPEA – 2024)** Em projetos de desenvolvimento de softwares, é importante que eles sejam testados em diferentes situações comuns de ocorrerem, de modo a contribuir para que eles atinjam a qualidade esperada. Por isso, nesses projetos, quatro níveis de teste são importantes de serem considerados, a fim de ajudar a lidar com tais situações. Esses quatro níveis de teste são os seguintes:

a) acessibilidade, usabilidade, unidade e sistema
b) desempenho, funcionais, componente e aceitação
c) funcionais, não funcionais, componente e desempenho
d) funcionais, não funcionais, usabilidade e acessibilidade
e) unidade, integração, sistema e aceitação

7. **(CESGRANRIO / IPEA – 2024)** Considere que um novo software foi desenvolvido e está prestes a entrar no ambiente de produção de uma empresa, mas, antes disso, serão realizados testes

finais. Para isso, um conjunto de representantes dos usuários finais deve participar desse estágio de testes. Caso se perceba que o software está tendo o comportamento esperado, ele será implantado em produção. Qual estágio de teste está descrito no cenário acima?

a) Aceitação
b) Componente
c) Configuração
d) Desempenho
e) Usabilidade

8. **(CESGRANRIO / AGERIO – 2023)** Ao planejar um projeto de sistema seguindo um ciclo de vida linear, um gerente de projeto resolveu instituir uma estratégia global de teste de software.

Considerando-se uma ordem do mais específico para o mais geral, ou seja, terminando-se com o teste de ordem superior, qual a ordem dos testes a serem realizados?

a) Teste de integração, teste de validação, teste de sistema, teste de unidade
b) Teste de sistema, teste de validação, teste de unidade, teste de integração
c) Teste de validação, teste de integração, teste de unidade, teste de sistema
d) Teste de validação, teste de sistema, teste de unidade, teste de integração
e) Teste de unidade, teste de integração, teste de validação, teste de sistema.

9. **(CESGRANRIO / Caixa – 2024)** Em um contrato para o desenvolvimento de um software de gestão empresarial, uma cláusula específica solicita a execução, pelo fornecedor, de um "teste alfa" antes da entrega do produto ao cliente. Para atender a essa cláusula do contrato, o fornecedor deve:

a) realizar uma série de testes que visa garantir que mudanças recentes no código não afetem as funcionalidades existentes do software, mantendo a integridade do sistema após atualizações ou correções.

b) distribuir o software para um grupo externo de usuários para que estes o utilizem em condições reais e forneçam feedback sobre a experiência antes de distribuir amplamente.

c) verificar a comunicação e o funcionamento adequados entre diferentes módulos ou componentes do software, assegurando que eles trabalhem juntos conforme esperado.

d) testar individualmente os menores pedaços de código do software, como funções ou métodos, para garantir que funcionem corretamente.

e) avaliar o software com uma equipe interna que simula o comportamento do usuário final, buscando identificar falhas antes da liberação para usuários externos.

10. **(VUNESP / TCM-SP – 2023)** Dentre as várias estratégias de teste de software, há uma delas que estabelece que quando da realização de testes de integração de módulos de software, alguns testes são executados novamente de modo a verificar se a adição de novos módulos não tenha provocado erros até então inexistentes. A essa técnica atribui-se a denominação de teste:

a) adaptativo.
b) principal.
c) de regressão.
d) secundário.
e) estratégico.

Figura 12 – Tela de entrada de dados

11. **(FUNDATEC / ISS-Porto Alegre – 2022)** Sabe-se que a equipe responsável pelo desenvolvimento da funcionalidade Agendar Atendimento, do software Sistema de Atendimento Agendado (SAA), realizou testes intensivos, com o objetivo de entregar tal funcionalidade estável e sem erros. A equipe de testes, antes de iniciar suas atividades, estudou os artefatos elaborados no projeto, tais como documento visão, diagramas e especificações de casos de uso, histórias de usuário, casos de teste, regras de negócio, modelo de dados, lista de mensagens, tipos de dados e valores válidos de entrada e saída, dentre outros. Os testadores realizaram muitas simulações, inserindo, na tela de entrada de dados, dados certos e errados, de modo a observar o comportamento do software e as correspondentes saídas de dados. Não cabia a essa equipe realizar testes na arquitetura do software e nem a validação de algoritmos, linguagem de programação ou quaisquer outras estruturas de dados, dessa funcionalidade. Nesse caso, pode-se afirmar que a equipe realizou o seguinte tipo de testes de software:

a) Teste unitário.
b) Teste de stress.
c) Teste de regressão.
d) Teste de caixa preta.
e) Teste de caixa branca.

12. **(IDHTEC / PREFEITURA DE MARAGOGI-AL – 2019)** Engenharia de software é uma abordagem sistemática e disciplinada para o desenvolvimento de software (PRESSMAN, 2006). Considere V para afirmativa verdadeira e F para falsa:

( ) Análise de requisito - Através da análise de requisito é o momento onde efetua a programação do código fonte para desenvolver o software (JALOTE, 2005).

( ) Design do software - Pelo design do software é o momento que o engenheiro de software realiza o planejamento da solução do problema que foi levantado no documento de requisito (JALOTE, 2005).

( ) Codificação - A codificação é o momento que criptografa e transformará em uma linguagem de programação (JALOTE, 2005).

( ) Teste - O teste de software é o processo que tem a intenção de encontrar defeitos nos artefatos de software (MYERS, 2004). O teste é uma maneira de medir o controle da qualidade do software durante o desenvolvimento de software (JALOTE, 2005).

A sequência correta, de cima para baixo, é:

a) F, V, V, V
b) V, F, V, F.
c) F, V, F, F.
d) V, F, F, F.
e) F, V, F, V.

13. **(IDHTEC / PREFEITURA DE MARAGOGI-AL – 2019)** Quanto à Automação de Testes, julgue os itens a seguir:

I. Ferramentas de automação não possuem outros usos, além da medição de performance de aplicações. Elas também não podem ser usadas para preparar um ambiente de teste com um grande volume de dados.

II. No teste de interface gráfica, uma plataforma gera os eventos de entrada na interface de utilizador do sistema e observa as mudanças na saída.

III. No teste baseado em código, a interface pública das classes, módulos ou bibliotecas são testadas com uma variedade de argumentos de entrada, observando-se a saída.

Está (estão) correto(s):

a) Somente I.
b) Somente I e II.

c) Somente II e III.
d) I, II e III.
e) Somente II.

14. **(IDHTEC / PREFEITURA DE MARAGOGI-AL – 2019)** Sobre testes, analise as afirmativas a seguir:

I. Teste de regressão corresponde a um nível de teste, mas não 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.

II. A técnica de teste de Estrutural é recomendada para os níveis de Teste da Unidade e Teste da Integração, cuja responsabilidade principal fica a cargo dos desenvolvedores do software, que são profissionais que conhecem bem o código-fonte desenvolvido e dessa forma conseguem planejar os casos de teste com maior facilidade.

III. Teste Funcional é a Técnica de teste em que o componente de software a ser testado é abordado como se fosse uma caixa-preta, ou seja, não se considera o comportamento interno do mesmo.

Está (estão) correta(s):

a) Somente I.
b) Somente I e II.
c) Somente II e III.
d) I, II e III.
e) Somente II.

15. **(IDHTEC / PREFEITURA DE MARAGOGI-AL – 2019)** Com relação aos testes realizados no processo de desenvolvimento de software, julgue as afirmativas a seguir:

I. Diversas atividades de testes são executadas a fim de se validar o produto de software, testando cada funcionalidade de cada módulo, buscando, levando em consideração a especificação feita na fase de projeto.

II. Na fase de Testes de Integração as unidades do sistema são testados de forma combinada, o objetivo é detectar falhas na interação entre as unidades integradas.

III. Na fase de Testes de Verificação de Unidade serão testados apenas os módulos das funcionalidades requeridas pelo cliente durante o projeto, garantindo o pleno funcionamento. Deve ser feito, preferencialmente, pelo usuário final.

Está (estão) correta(s):

a) Apenas a afirmativa I.
b) Apenas a afirmativa II.
c) Apenas a afirmativa III.
d) Apenas as afirmativas I e II.
e) As afirmativas I, II e III.

16. **(AVANÇA SP / CÂMARA MUNICIPAL DE TABOÃO DA SERRA-SP – 2019)** No que se refere às técnicas de teste de software, há os testes conhecidos como "caixa preta" e "caixa branca". Sobre o tema, analise os itens a seguir e, ao final, assinale a alternativa correta:

I – Testes do tipo "caixa branca" são realizados apenas após o software estar completamente integrado.

II – Testes do tipo "caixa preta" não são aplicáveis a software de pequeno porte.

III – Testes do tipo "caixa preta" tem a finalidade de exercitar as interfaces do software sob teste.

a) Apenas o item I é verdadeiro.
b) Apenas o item II é verdadeiro.
c) Apenas o item III é verdadeiro.
d) Apenas os itens I e II são verdadeiros.
e) Todos os itens são verdadeiros.

17. **(FUNDEP / PREFEITURA DE LAGOA SANTA-MG – 2019)** Assinale a associação correta presente na tabela ASSOCIAÇÕES que define corretamente os elementos a definir da TABELA A com as definições ou caracterizações da TABELA B.

TABELA A	
	A definir
1	Os processos de software são
2	Modelos de processos de software
3	Modelos gerais de processo
4	Engenharia de requisitos
5	Validação de software

TABELA B	
	Definição ou caracterização
A	Modelos de processos de software
B	as atividades envolvidas na produção de um sistema de software
C	é o processo de desenvolvimento de uma especificação de software
D	descrevem a organização dos processos de software
E	é o processo de verificação de que o sistema está de acordo com sua especificação e satisfaz às necessidades reais dos usuários do sistema.

a)

TABELA A	TABELA B
1	B
2	A
3	C
4	D
5	E

b)

TABELA A	TABELA B
1	B
2	A
3	D
4	C
5	E

c)

TABELA A	TABELA B
1	C
2	A
3	B
4	E
5	D

d)

TABELA A	TABELA B
1	C
2	B
3	E
4	D
5	A

18. (IESES / SCGÁS – 2019) Identifique a alternativa que descreve inequivocamente a intenção do teste de verificação de software ou, mais genericamente, verificação e validação (V&V):

a) O teste de verificação de software tem a intenção de mostrar que um software se adequa às suas especificações ao mesmo tempo que satisfaz as especificações do cliente do sistema.

b) O teste de verificação de software tem a intenção de demostrar que um software se adequa às suas especificações ao mesmo tempo que satisfaz as especificações do mercado concorrente.

c) É um processo que verifica a atualização de versão de sistema. Tem relação com softwares básicos.

d) O teste de verificação procura verificar similaridades entre softwares para buscar encontrá-los.

19. (IESES / SCGÁS – 2019) Assinale a alternativa correta que apresenta a diferença entre teste de defeito e debugging:

a) Testes de defeito estabelecem a existência de defeitos já o debugging diz respeito à localização e correção desses defeitos.
b) Não há diferença entre os dois. Teste de defeito e debugging são a mesma coisa.
c) O debugging somente pode ser feito depois do teste de defeito.
d) O teste de defeito localiza e correção do defeito enquanto o debugging diz que há erro.

20. (IBADE / PREFEITURA DE VILHENA-RO – 2019) Sobre teste de software, podemos diferenciar teste da caixa preta e teste da caixa branca respectivamente da seguinte maneira:

a) enquanto o primeiro ignora o código fonte, o segundo busca garantir que os componentes do software estejam concisos.

b) no primeiro são gerados dados aleatórios para teste, enquanto no segundo os dados são passados manualmente.

c) no primeiro o teste é feito de forma obscura sem saber o que o programa faz, enquanto que no segundo é feito de forma clara, analisando as entradas e saídas esperadas pelo programa.

d) enquanto que no primeiro há a necessidade de conhecer a estrutura do código, no segundo há apenas a necessidade de saber o resultado esperado para cada entrada de dados.

e) enquanto que no primeiro teste não se conhece a entrada de dados, mas se sabe a saída esperada, no segundo não se sabe a saída esperada, mas se conhece as entradas de dados possíveis.

21. **(INSTITUTO AOCP / UFPB – 2019)** Os testes de software são realizados para verificar se um programa realmente faz o que é proposto a fazer e de forma correta, assim é possível descobrir os defeitos do programa antes dele ser utilizado pelo usuário final. Os testes são realizados utilizando dados fictícios em busca de erros e anomalias. Existem diversos tipos de testes. Assinale a alternativa que apresenta as características dos testes unitários de desenvolvimento de software:

a) Testes em que os potenciais usuários de um sistema testam o sistema em seu próprio ambiente.
b) Testes que são centrados nas interfaces dos componentes.
c) Testes que não têm como objetivo verificar a funcionalidade de objetos e métodos.
d) Testes que são centrados nas interações entre os componentes.
e) Testes em que as unidades individuais de programa ou classes de objetos são testadas individualmente.

22. **(INSTITUTO AOCP / IBGE – 2019)** Para chegar a um nível de perfeição de um software, é necessário aplicar muitos testes, sendo que o teste de integração é um dos mais importantes. Considerando o exposto, assinale a alternativa que NÃO apresenta uma característica dos testes de integração de software:

a) Testar as dependências entre os componentes.
b) Testar as interfaces entre as unidades.
c) Simular módulos ainda não implementados que se comunicam ao módulo testado.
d) Testar conformidade com a especificação dos requisitos.
e) Realizar teste estrutural ou caixa-branca.

23. **(INSTITUTO AOCP / IBGE – 2019)** A respeito dos testes de aceitação, analise as assertivas e assinale a alternativa que aponta(s) as correta(s).

I. É um teste que isenta de responsabilidades os usuários finais ou clientes.

II. O propósito do teste não é somente encontrar erros no software mas também erros de instalação do software.

III. É um teste em que o analista deve executar um processo de comparação dos requisitos iniciais do software e das necessidades atuais dos usuários finais.

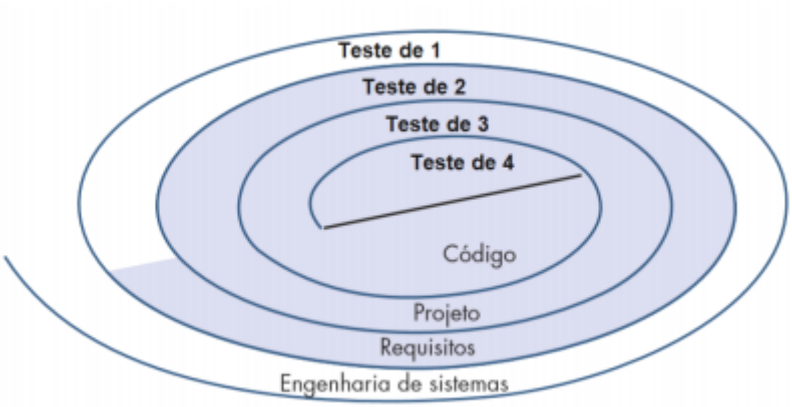
a) Apenas I.
b) Apenas II.
c) Apenas III.
d) Apenas I e II.
e) Apenas II e III.

24. **(INSTITUTO AOCP / IBGE – 2019)** Um analista de sistemas do IBGE necessita realizar um teste em um software. Durante o teste, o analista teve como objetivo não se preocupar com o comportamento interno do software e suas estruturas. Ao invés disso, ele se concentrou em encontrar as circunstâncias pelas quais o software não se comportava em conformidade com os seus requisitos. Diante desse cenário, assinale a alternativa que apresenta corretamente o nome do teste realizado pelo analista do IBGE:

a) Teste de caixa preta.
b) Teste de integração.
c) Teste ágil.
d) Teste de aplicação.
e) Teste de funcionalidade.

25. **(QUADRIX / CRO-GO – 2019)** Análise de requisitos, implementação e testes são alguns dos processos que fazem parte do desenvolvimento de sistemas orientados a objetos.

26. **(COLÉGIO PEDRO II – 2019)** Teste é um conjunto de atividades que podem ser planejadas com antecedência e executadas sistematicamente. Deverá ser definido, para o processo de software, um conjunto de etapas nas quais podem-se empregar técnicas específicas de projeto de caso de teste e métodos de teste. O processo de software pode ser visto como a espiral ilustrada na figura a seguir. Inicialmente, a engenharia de sistemas define o papel do software e passa à análise dos requisitos de software, na qual são estabelecidos o domínio da informação, função, comportamento, desempenho, restrições e critérios de validação para o software. Deslocando-se para o interior da espiral, chega-se ao projeto e, finalmente, à codificação.



Uma estratégia para teste de software também pode ser vista no conceito da espiral, como na figura, correlacionando o modelo de teste adotado à fase na qual o software se encontra. A alternativa que corresponde corretamente às respectivas fases de teste numeradas na figura como 1, 2, 3 e 4 é:

a) Recuperação – Esforço – Segurança – Desempenho.
b) Sistema – Verificação – Validação – Depuração.
c) Sistema – Entrevista – Controle – Codificação.
d) Sistema – Validação – Integração – Unidade.

27. **(COVEST-COPSET / UFPE – 2019)** A respeito de princípios básicos para elaboração de testes de software, assinale a alternativa correta.

a) A definição da saída ou resultado esperado é uma parte desnecessária em um caso de teste.
b) Testes de software são tarefas repetitivas, pouco criativas ou desafiadoras, pois são fáceis de automatizar quando pensadas no início do projeto.
c) Se muitos erros já foram encontrados em uma seção do programa, a probabilidade de encontrar mais erros ali será baixa.
d) Casos de teste devem ser escritos para condições de entrada inválidas e inesperadas, bem como para aquelas que são válidas e esperadas.
e) Ao se planejar um esforço de testes, faz-se a suposição tácita de que nenhum erro será encontrado.

28. **(COVEST-COPSET / UFPE – 2019)** No contexto de diferentes técnicas de teste de caixa preta (black-box) e caixa branca (white-box), assinale a alternativa correta:

a) A análise de valor limite é uma técnica caixa branca que foca em testar valores de entrada e saída, acima e abaixo dos limites dos parâmetros de entrada possíveis.

b) A cobertura de decisão é uma técnica caixa branca que foca em escrever casos de teste nos quais se evitam que determinadas ramificações do código sejam executadas, visando reduzir a cobertura para otimizar o tempo de teste.

c) A cobertura de condição é uma técnica caixa preta que foca em verificar se cada condição na decisão é executada pelo menos uma vez.

d) Um teste de partição de equivalência é uma técnica caixa preta em que se consegue escolher conjuntos de entradas que possibilitam reduzir o número de casos de testes que precisam ser desenvolvidos para atingir algum objetivo.

e) A suposição de erros é uma técnica caixa branca que consiste em um processo sistemático para especulação de prováveis tipos de erros, culminando com a escrita de casos de teste para expor estes erros.

29. **(COVEST-COPSET / UFPE – 2019)** Quanto aos tipos de testes de software, assinale a alternativa incorreta:

a) Teste unitário ou de unidade é onde unidades de programa individuais ou classes de objeto são testadas. O foco é testar a funcionalidade de objetos ou métodos.

b) Teste de componente é onde várias unidades individuais são integradas para criar componentes compostos. O foco é testar as interfaces de componentes.

c) Teste de sistema é onde alguns ou todos os componentes de um sistema são integrados e o sistema é testado como um todo. O foco é testar as interações dos componentes.

d) Teste de regressão é onde o sistema é restaurado para uma versão anterior. O teste se preocupa em reproduzir comportamento de algum bug reportado através de controle de mudanças, para posterior correção.

e) Testes de performance é onde se testa um requisito não funcional. O teste se preocupa em demonstrar que o sistema atende aos requisitos e em descobrir problemas e defeitos no sistema.

30. **(COVEST-COPSET / UFPE – 2019)** Como parte da prática de Integração Contínua (CI), podemos elaborar testes unitários, testes de integração e testes de aceitação. Ao se executar um teste de aceitação:

a) executa-se o sistema inteiro com um subconjunto básico de funcionalidades para avaliar o funcionamento de tarefas críticas.

b) executa-se a rotina com uma entrada pré-definida e compara-se a saída com um resultado pré-definido.

c) executa-se o sistema inteiro com foco na especificação das funcionalidades para saber se a rotina implementada corresponde ao requisitado.

d) executa-se o sistema inteiro avaliando o fluxo de informação entre as diferentes rotinas do sistema e sua interoperabilidade.

e) executa-se a rotina desejada em conjunto com um subconjunto de rotinas que possuem dependência direta e compara-se a saída com um resultado pré-definido.

31. **(CESGRANRIO / UNIRIO – 2019)** José é um desenvolvedor e acabou de fazer uma alteração no código. O gerente de projeto definiu que serão realizados os seguintes testes: unitários/integração; de sistema; de aceitação. A empresa possui uma área de testes independente da equipe de desenvolvimento. O Desenvolvedor, a equipe de teste e o usuário devem executar, respectivamente, os seguintes testes:

a) de sistema; unitário/de integração; de aceitação.
b) de sistema; unitário/de integração; de aceitação.
c) unitário/de integração; de sistema; de aceitação.
d) unitário/de integração; de aceitação; de sistema.
e) de sistema; de aceitação; unitário/de integração.

32. **(CESGRANRIO / UNIRIO – 2019)** Os testes de integração determinam se as unidades de software desenvolvidas independentemente funcionam corretamente quando estão conectadas umas às outras. Dentre os tipos de teste que são comumente usados nos testes de integração, estão os testes de:

a) matriz ortogonal e de análise de valor limite.
b) interfaces com o usuário e de cenários de uso.
c) usabilidade e de cenários de uso.
d) desempenho e os beta testes.
e) desempenho e usabilidade.

33. **(FEPESE / CELESC – 2019)** Assinale a alternativa que apresenta o tipo de teste de software que é baseado nos requisitos funcionais do software. Neste tipo de teste os casos de teste são gerados sem o conhecimento da estrutura interna do software:

a) Caixa Preta.
b) Caixa Branca.
c) Caixa Estrutural.
d) Caixa Funcional.
e) Cobertura Funcional.

34. **(IF / MT – 2019)** Analise as sentenças presentes em Pressmann (2006) relacionadas abaixo, acerca de estratégias de teste para software convencional:

I - Teste de_____ é uma técnica sistemática para construir a arquitetura do software enquanto, ao mesmo tempo, conduz testes para descobrir erros associados às interfaces.

II - No teste de _____ são utilizadas as descrições de projeto no nível de componente como guia para testar caminhos de controle importantes e descobrir erros dentro dos limites do módulo.

III - O teste de _____ é uma abordagem de teste de _____ e tem por objetivo exercitar o sistema inteiro, de ponta a ponta, sendo capaz de expor os principais problemas existentes no produto ainda na etapa de construção.

IV - O teste de _____ tem por objetivo verificar se há defeitos de software em modificações recentes que afetam módulos já testados e que antes funcionavam impecavelmente.

As lacunas das sentenças podem ser preenchidas CORRETAMENTE com a opção:

a) Integração, Unidade, Fumaça, Integração, Regressão.
b) Unidade, Integração, Regressão, Usabilidade, Regressão.
c) Interação, Configuração, Regressão, Unidade, Utilidade.
d) Recuperação, Desempenho, Sistema, Desempenho, Unidade.
e) Configuração, Regressão, Integração, Unidade, Fumaça.

35. **(IF-PE / IF-PE – 2019)** Em relação aos testes de software, podemos afirmar que:

I. teste de unidade é a realização de testes sobre unidades do sistema para garantir que a funcionalidade de objetos ou métodos esteja correta.

II. teste de componentes é a realização de testes sobre as interfaces entre os componentes de um software.

III. teste de sistema procura testar a integração de todos os componentes de um sistema.

Está(ão) CORRETA(S), apenas, a(s) proposição(ões):

a) I,II e III
b) Apenas I
c) Apenas I e II
d) Apenas II e III
e) Apenas I e III

36. **(VUNESP / PREFEITURA DE ITAPEVI-SP – 2019)** Um programador, após desenvolver um programa, iniciou o processo de depuração do código. O teste projetado em função da estrutura interna do software e que visa cobrir a funcionalidade do componente de software é denominado Teste:

a) de Carga.
b) de Estresse.
c) de Segurança.
d) Estrutural (caixa-branca).
e) Funcional (caixa-preta).

37. **(VUNESP / PREFEITURA DE PIRACICABA-SP – 2019)** Dentre as técnicas de teste de software, pode-se estabelecer uma categorização entre testes de caixa preta e de caixa branca, sendo correto que a técnica de teste denominada:

a) análise de valor limite é um método de teste de caixa branca.
b) matriz ortogonal é um método de teste de caixa branca.
c) teste de condição é um método de teste de caixa preta.
d) fluxo de dados é um método de teste de caixa preta.
e) particionamento de equivalência é um método de teste de caixa preta.

38. **(VUNESP / PREFEITURA DE VALINHOS-SP – 2019)** Há dois tipos de testes de validação de software, conhecidos como testes alfa e beta, segundo os quais:

a) o teste alfa é realizado nas instalações do desenvolvedor do software.
b) o teste beta é realizado nas instalações do desenvolvedor do software.
c) os testes alfa e beta são executados em um ambiente tercerizado.
d) no teste alfa, participam apenas os desenvolvedores do software.
e) no teste beta, participam apenas os desenvolvedores do software.

39. **(IADES / BRB – 2019)** Há diversos tipos de testes de software e, entre eles, o tipo que consiste no reteste de um sistema ou componente focado em verificar se alguma modificação recente causou efeitos negativos no sistema denomina-se teste de:

a) manutenção.
b) performance.
c) usabilidade.
d) integração.
e) regressão.

40. **(IDECAN / IF-PB– 2019)** O processo de teste tem dois objetivos distintos:

i) Demonstrar ao desenvolvedor e ao cliente que o software atende a seus requisitos e
ii) Descobrir situações em que o software se comporta de maneira incorreta, indesejável ou de forma diferente das especificações.

Sobre testes de software, é incorreto afirmar que:

a) os testes não são capazes de demonstrar que um software é livre de defeitos.

b) o objetivo da verificação é checar se o software atende aos requisitos funcionais e não funcionais.
c) o objetivo da validação é garantir que o software atende às expectativas do cliente.
d) testes de integração focam na descoberta de erros associados com interfaces de módulos.
e) teste fumaça é uma abordagem de testes unitários.

41. **(CCV / UFC – 2019)** Sobre os tipos de testes de software, marque o item correto.

a) Nos testes caixa-preta, todo o código da aplicação estará disponível para o profissional de teste analisar e especificar quais pontos deverão ser testados.

b) Os testes de fluxo de dados visam analisar os aspectos estruturais da aplicação nos trechos de código onde os dados passam, com foco nas estruturas de controle.

c) Os testes de regressão são realizados pelos clientes com o intuito de checar se todas as funcionalidades e alterações demandadas foram desenvolvidas.

d) Os testes funcionais procuram verificar se o sistema está seguindo a sua especificação (requisitos), sem se preocupar com a estrutura adotada na implementação do sistema.

e) Os testes de unidade são realizados após a conclusão dos diferentes módulos do sistema, onde se busca analisar a corretude do funcionamento da integrado desses módulos.

42. **(CCV / UFC – 2019)** Durante o desenvolvimento de um sistema, é necessária a realização de testes, sendo um deles denominado de teste beta. Sobre esse tipo de teste, assinale a alternativa correta:

a) É realizado no ambiente de desenvolvimento, onde somente os desenvolvedores envolvidos na escrita do código realizarão os testes.

b) É considerado como sendo um teste de unidade, onde todos os módulos desenvolvidos separadamente são testados em conjunto.

c) O teste beta é aplicado para os usuários contendo apenas uma versão inicial do sistema, com poucos recursos para a validação de requisitos.

d) Tal teste são realizados com os usuários do sistema analisando o código fonte produzido, propondo correções e melhorias a serem aplicadas.

e) É realizado pelos usuários do sistema, onde os requisitos do sistema são analisados, validados e os erros encontrados reportados para a equipe de desenvolvimento.

43. **(IF / MS – 2019)** Segundo Pressman (2011), considere as seguintes afirmações sobre Engenharia de Software:

I. Erro é um problema de qualidade encontrado após a liberação para o usuário final.

II. O teste de unidade é a reexecução do mesmo subconjunto de testes que já foram executados para assegurar que as alterações não tenham propagado efeitos colaterais indesejados.

III. O modelo espiral é um modelo de processo de software evolucionário que acopla a natureza iterativa da prototipação com os aspectos sistemáticos e controlados do modelo cascata.

Assinale a opção CORRETA:

a) Apenas a afirmação I é verdadeira.
b) Apenas a afirmação II é verdadeira.
c) Apenas a afirmação III é verdadeira.
d) Apenas as afirmações I e III são verdadeiras.
e) Apenas as afirmações II e III são verdadeiras.

44. **(FUMARC / CÂMARA DE CARMO DO CAJURU-MG – 2018)** Em relação aos tipos de testes de software, julgue os itens a seguir, marcando com (V) a assertiva verdadeira e com (F) a assertiva falsa:

( ) Teste de Regressão significa executar novamente um subconjunto de testes já realizado anteriormente, para garantir que as últimas modificações não propagarão efeitos colaterais indesejáveis no software.

( ) Testes Alfa são realizados no ambiente de produção do usuário final para identificar possíveis problemas nesse ambiente que não foram detectados nas fases anteriores de teste.

( ) O Teste de Estresse executa um sistema de tal forma que ele demande recursos em quantidade, volume ou frequência anormais, com o objetivo de identificar limites de capacidade.

( ) O Teste de Unidade avalia a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente.

A sequência CORRETA, de cima para baixo, é:

a) F, F, V, V.
b) F, V, F, V.
c) V, F, V, V.
d) V, V, F, F.

45. **(FAUGRS / UFCSPA-RS – 2018)** No teste de _____, os módulos são combinados e testados em grupo. Ele sucede o teste de _____, em que os módulos são testados individualmente,

e antecede o teste de _____, em que o sistema completo é testado em um ambiente que simula o ambiente de produção.

Assinale a alternativa que completa, correta e respectivamente, as lacunas do texto acima.
a) integração – unidade – sistema.
b) release – integração – unidade.
c) unidade – caminho – release.
d) requisito – sistema – integração.
e) partições – requisito – caminho.

46. **(FUNDATEC / CIGA-SC – 2018)** Uma equipe de teste de software identificou erros em algumas funcionalidades de um aplicativo durante a execução de suas atividades. Os erros foram reportados ao Gerente de Projetos que acionou imediatamente a equipe de desenvolvimento para a realização dos ajustes necessários. Concluídas as correções, a equipe de testes recebeu novamente as funcionalidades e confirmou que os erros foram devidamente corrigidos. Apesar disso, essa equipe irá realizar mais um ciclo de teste com o objetivo de verificar se a nova versão do software, ajustada pelos desenvolvedores, não introduziu novos defeitos, em outros pontos do aplicativo, em consequência dos ajustes realizados. Nesse caso, esse tipo de teste é chamado de teste:

a) De estresse.
b) Funcional.
c) De regressão.
d) De segurança.
e) De aceitação do produto.

47. **(FUNDATEC / CIGA-SC – 2018)** A equipe responsável pelo desenvolvimento de um software está agilizando a conclusão de um release para entregá-lo estável ao cliente. No momento, as funcionalidades de tal release estão sendo submetidas a testes intensivos, pela equipe de testes. Essa equipe, antes de iniciar suas atividades, estudou os artefatos da linha base elaborados no projeto, tais como os documentos de viabilidade e visão, diagrama de casos de uso e as suas especificações, casos de teste, regras de negócio, modelo de dados, lista de mensagens e os tipos de dados e valores válidos para os diversos campos do sistema, dentre outros. Esses testadores encontram-se realizando diversas simulações, com a entrada de dados certos e errados, de modo a observar se o sistema se comporta conforme os documentos da linha base, acordados com o cliente. Não faz parte do escopo da equipe de teste, validar a linguagem de programação, a estrutura de dados, os algoritmos ou qualquer outro aspecto da arquitetura e estrutura interna do sistema. Nesse caso, o tipo de teste de software, que se encontra sendo realizado pela equipe de teste, é chamado de teste:

a) Unitário.
b) De caixa preta.
c) De caixa branca.
d) De performance.

e) De desenvolvimento.

48. **(INSTITUO AOCP / PRODEB – 2018)** Qual é o tipo de testes em que o testador está preocupado com a funcionalidade e não com a implementação?

a) Teste unitário.
b) Teste de release.
c) Teste disfuncional.
d) Teste de integração.
e) Teste de caixa cinza.

49. **(AOCP / UNIR – 2018)** O projeto de casos de teste não possui eficácia para a prevenção de defeitos e identificação de erros.

50. **(AOCP / UNIR – 2018)** As etapas do teste de software são: implantação, verificação e análise de resultados.

51. **(AOCP / UNIR – 2018)** Os defeitos no processo de desenvolvimento, em sua maior parte, são de origem humana, pois são gerados na comunicação e na transformação de informações, e continuam presentes nos diversos produtos de software produzidos, liberados e localizados em partes do código raramente executadas.

52. **(AOCP / UNIR – 2018)** Para testar a complexidade de um software, pode ser utilizado o teste de integração que avalia a integração do software com vários usuários ao mesmo tempo.

53. **(IBADE / IPM-JP – 2018)** No âmbito dos testes de integração, a atividade de reexecução de um mesmo subconjunto dos que já foram executados para assegurar que alterações não tenham propagado efeitos colaterais indesejados é conhecida como teste de:

a) unidade.
b) regressão.
c) validação.
d) verificação.
e) classe.

54. **(IBADE / IPM-JP – 2018)** Uma estratégia de teste de software pode englobar diferentes tipos de testes para assegurar a qualidade do software. Os que proporcionam a garantia final de que o software satisfaz todos os requisitos informativos, funcionais, comportamentais são conhecidos como testes:

a) de validação.
b) de unidade.
c) de integração.
d) totais.

e) globais.

55. **(AOCP / SUSIPE-PA – 2018)** Sobre os testes de software, assinale a alternativa correta.

a) Um teste de regressão visa refazer os testes feitos anteriormente, visando garantir o funcionamento correto destes.
b) Um teste operacional tem como objetivo testar a aplicação em funcionamento no sistema operacional utilizado, visando encontrar possíveis conflitos de operações.
c) Um teste de configuração visa garantir que as configurações da aplicação não sejam conflitantes com o ambiente utilizado.
d) Um teste de integração visa garantir a interação da aplicação com outras aplicações.
e) Um teste de carga tem como objetivo verificar o comportamento do sistema com uma grande carga de usuários simultâneos.

56. **(INSTITUTO AOCP / PRODEB – 2018)** Qual é o objetivo da realização de testes funcionais?

a) Visualizar o sistema como caixa branca.
b) Convencer de que o sistema é bom o suficiente para uso.
c) Justificar as possíveis falhas.
d) Identificar valor em funcionalidades não previstas.
e) Permitir validação feita pelo cliente após a entrega final das funcionalidades.

57. **(PR-4 / UFRJ – 2018)** Assinale o teste que focaliza o esforço de verificação da menor unidade de projeto de software:

a) Teste de integração.
b) Teste de regressão.
c) Teste de validação.
d) Teste de unidade.
e) Teste de projeto.

58. **(FAURGS / UFRGS – 2018)** Numere a segunda coluna de acordo com a primeira, associando os termos com suas respectivas definições:

(1) Teste de regressão
(2) Teste funcional
(3) Teste caixa-branca
(4) Teste unitário
(5) Teste de estresse
(6) Teste de desempenho
(7) Teste de segurança
(8) Teste de software
(9) Teste de carga

( ) Seu objetivo é verificar o comportamento do software contra a lógica de negócio descrita nos documentos de requisitos e especificação.

( ) Teste que foca na lógica interna de processamento e nas estruturas de dados dentro dos limites de um componente.

( ) Verifica a performance do software durante a execução, principalmente em relação aos critérios ligados a consumo de recursos de processamento, memória e tempo de resposta.

( ) Processo de retestar um software que sofreu modificações.

A sequência numérica correta de preenchimento dos parênteses da segunda coluna, de cima para baixo, é:

a) 5 – 3 – 9 – 6.
b) 2 – 4 – 6 – 1.
c) 4 – 6 – 9 – 1.
d) 2 – 3 – 5 – 9.
e) 5 – 4 – 6 – 8.

59. **(COPESE / CÂMARA DE PALMAS-TO – 2018)** São consideradas fases da atividade de teste de softwares, EXCETO:

a) Teste de Unidade.
b) Teste de Integração.
c) Teste de Sistemas.
d) Teste de Fault.

60. **(CESGRANRIO / LIQUIGÁS – 2018)** Um grupo de desenvolvedores elaborou vários casos de teste que selecionam caminhos de teste de acordo com as definições e com o uso de variáveis existentes em um programa. Esse tipo de teste caixa-branca é denominado:

a) teste de condição.
b) teste de fluxo de dados.
c) teste de caminho básico.
d) análise de valor-limite.
e) particionamento de equivalência.

61. **(UFPR / COREN-PR – 2018)** Sobre testes de software, identifique como verdadeiras (V) ou falsas (F) as seguintes afirmativas:

( ) Um teste bem-sucedido identifica defeitos.

( ) Casos de teste são especificações das entradas para o teste e da saída esperada do sistema.

( ) Um teste caixa-preta é um teste estrutural, em que partes específicas de componentes são testadas.

( ) Um teste de aceitação foca em cada unidade do software, ou seja, seu código-fonte.

Assinale a alternativa que apresenta a sequência correta, de cima para baixo.

a) V – V – F – F.
b) F – F – V – V.
c) V – F – F – V.
d) F – V – V – F.
e) V – V – V – V.

62. (FAURGS / BANRISUL – 2018) No desenvolvimento de software, o processo de teste resulta na produção de distintos artefatos. Dentre estes, documentos. Sobre o documento Plano de Teste, assinale a alternativa que apresenta o elemento que NÃO faz parte desse artefato:

a) Escopo dos testes, ou seja, os itens de software que devem ser testados e os que não devem.
b) Estratégia de teste.
c) Necessidades em termos de recursos humanos e treinamentos.
d) Casos de teste.
e) Cronograma.

63. (FAURGS / BANRISUL – 2018) _____ é uma técnica utilizada para se projetar casos de teste, na qual o programa ou sistema é considerado como uma caixa-preta. Nesta técnica os detalhes de implementação não são considerados e o software é avaliado segundo o ponto de vista do usuário, procurando explorar determinados tipos de defeitos, estabelecendo requisitos de teste para os quais valores específicos do domínio de entrada do programa devem ser definidos com o intuito de exercitá-los. Utilizando _____, o domínio de entrada é reduzido a um conjunto de estados válidos ou inválidos para as condições de entrada, e com um tamanho passível de ser tratado durante a atividade de teste.

Assinale a alternativa que preenche correta e respectivamente as lacunas do texto acima:

a) Teste funcional – particionamento de equivalência.
b) Teste estrutural – análise de valor limite.
c) Teste funcional – critérios baseados em fluxos de controle.
d) Teste estrutural – critérios baseados em fluxos de controle.
e) Teste estrutural – particionamento de equivalência.

64. (FAURGS / BANRISUL – 2018) _____ é uma verificação de consistência entre o sistema de software e sua especificação e, portanto, é uma atividade de verificação feita depois que se tem o sistema completo, com todas suas partes integradas para verificar se as

funcionalidades especificadas nos documentos de requisitos estão todas corretamente implementadas. Este tipo de teste é focado principalmente na descoberta de falhas e executado pelo grupo de desenvolvimento de testes, tendo também um papel importante para avaliar se o produto pode ser liberado para os consumidores, o que é diferente do seu papel de expor falhas que são removidas para melhorar o produto.

Assinale a alternativa que preenche corretamente a lacuna do texto acima:

a) Teste de sistema
b) Teste de unidade
c) Inspeção
d) Teste de regressão
e) Teste de integração

65. (FAURGS / BANRISUL – 2018) _____ é o teste que tem como foco as menores unidades de um programa, que podem ser funções, procedimentos, métodos ou classes. Neste contexto, espera-se que sejam identificados erros relacionados a algoritmos incorretos ou mal implementados, estruturas de dados incorretas ou simples erros de programação. Como cada unidade é testada separadamente, este teste pode ser aplicado à medida que ocorre a implementação e pelo próprio desenvolvedor, sem a necessidade de dispor-se do sistema totalmente finalizado.

Assinale a alternativa que preenche corretamente a lacuna do texto acima:

a) Teste de regressão
b) Teste de integração
c) Teste de unidade
d) Teste de sistema
e) Teste de aceitação

66. (FAURGS / BANRISUL – 2018) Considere as seguintes afirmações sobre o Modelo "V" de teste de software.

I - Descreve a relação entre ações de garantia da qualidade e as ações associadas à comunicação, modelagem e atividades iniciais de construção.

II - À medida que a equipe de software desce em direção ao lado esquerdo do "V", os requisitos básicos do problema são refinados em representações, progressivamente, mais detalhadas e técnicas do problema e de sua solução. Ao ser gerado o código, a equipe se desloca para cima, no lado direito do "V", realizando basicamente uma série de testes que validem cada um dos modelos criados, à medida que a equipe se desloca para baixo, no lado esquerdo do "V".

III - Fornece uma forma para visualizar como a verificação e as ações de validação são aplicadas ao trabalho de engenharia anterior.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II, III.

67. (FAURGS / BANRISUL – 2018) Considere as seguintes afirmações sobre objetivos de teste.

I - A definição dos critérios de aceitação deve idealmente ocorrer depois do contrato do sistema ser assinado, pois os critérios de aceitação não fazem parte do contrato, embora possam ser acordados entre o cliente e o desenvolvedor.

II - O processo de teste deve demonstrar ao desenvolvedor e ao cliente que o software atende aos requisitos propostos.

III - Um dos objetivos do processo de teste é descobrir situações em que o software se comporte de maneira incorreta, indesejável ou de forma diferente das especificações.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II, III.

68. (FAURGS / BANRISUL – 2018) Considere as seguintes afirmações sobre a relação entre requisitos e teste de software.

I - A correção, a completude e a consistência do modelo de requisitos não terão forte influência sobre a qualidade de todos os produtos seguintes do desenvolvimento de software, pois o que importa para o teste é o código fonte.

II - Um dos princípios gerais das boas práticas de engenharia de requisitos é que os requisitos devem ser testáveis, isto é, o requisito deve ser escrito de modo que um teste possa ser projetado para ele. Um testador pode então verificar se o requisito foi satisfeito.

III - Testes baseados em requisitos são uma abordagem sistemática para projeto de casos de teste em que cada requisito é considerado, derivando-se, assim, um conjunto de testes para ele.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II, III.

69. **(FAURGS / BANRISUL – 2018)** As alternativas abaixo apresentam características importantes que devem ser consideradas para o teste de aplicativos móveis, EXCETO uma. Assinale-a.

a) Linguagem de desenvolvimento.
b) Usabilidade.
c) Localização geográfica do dispositivo.
d) Volume de dados.
e) Variedade de dispositivos e sistemas operacionais.

70. **(FAURGS / BANRISUL – 2018)** Numere a segunda coluna de acordo com a primeira, associando os Níveis de Teste de Software às suas respectivas características.

(1) Teste de Unidade
(2) Teste de Integração
(3) Teste de Sistema
(4) Teste de Aceitação

( ) Avalia o software com respeito ao projeto de seus subsistemas e detecta suposições errôneas sobre pré e pós-condições para execução de um componente, falhas nas interfaces de comunicação dos componentes do software.

( ) Avalia o software com respeito aos seus requisitos e detecta falhas nos requisitos e na interface com o usuário.

( ) Avalia o software com respeito a sua implementação detalhada e detecta falhas de codificação, algoritmos ou estruturas de dados incorretos ou mal implementados.

( ) Avalia o software com respeito ao seu projeto arquitetural e detecta falhas de especificação, desempenho, robustez e segurança.

A sequência correta de preenchimento dos parênteses, de cima para baixo, é:

a) 2 – 4 – 1 – 3.
b) 1 – 4 – 2 – 3.
c) 4 – 2 – 3 – 1.
d) 2 – 1 – 3 – 4.
e) 1 – 3 – 4 – 2.

71. **(FUMARC / COPASA – 2018)** Teste realizado em ambiente de produção por um grupo de usuários finais para identificar problemas e realizar as devidas correções antes de liberar o software para toda a base de clientes:

a) Teste Alfa.
b) Teste Beta.
c) Teste de Regressão.
d) Teste Fumaça.

72. **(FUMARC / COPASA – 2018)** Analise as seguintes afirmativas sobre os tipos de testes:

I. O "Teste de Segurança" verifica se os mecanismos de proteção incorporados a um sistema vão de fato protegê-lo de invasão imprópria.

II. O "Teste de Desempenho" é projetado para submeter o software a situações anormais de funcionamento, demandando recursos excessivos até o limite da capacidade da infraestrutura destinada ao software.

III. O "Teste de Recuperação" força o software a falhar de diversos modos e verifica se a recuperação é adequadamente realizada.

Estão CORRETAS as afirmativas:

a) I, II e III.
b) I e II, apenas.
c) I e III, apenas.
d) II e III, apenas.

73. **(FUMARC / COPASA – 2018)** Técnica de teste utilizada para descobrir erros associados às interfaces na qual, a partir de componentes testados individualmente, se constrói uma estrutura de programa determinada pelo projeto é:

a) Teste de Desempenho.
b) Teste de Integração.
c) Teste de Segurança.
d) Teste de Unidade.

74. **(FUMARC / COPASA – 2018)** Teste realizado na instalação do desenvolvedor com os usuários finais, em um ambiente controlado, para identificar erros e problemas de uso durante a operação do sistema pelos usuários é denominado:

a) Teste Alfa.
b) Teste Beta.

c) Teste de Regressão.
d) Teste Fumaça.

75. **(IADES / CFM – 2018)** A respeito dos processos de verificação, de validação e de teste de software, assinale a opção correta.

a) Verificação, validação e teste são atividades independentes, de maneira que não possuem qualquer vínculo entre si.
b) Validação é uma atividade que permite realizar a verificação e os testes do software.
c) Teste consiste em analisar-se o software construído para confirmar se ele atende às verdadeiras necessidades dos interessados (cliente, usuário etc).
d) Verificação consiste em analisar-se o software para confirmar se ele está sendo construído de acordo com o que foi especificado.
e) Os testes devem ser executados antes das atividades de verificação.

76. **(CS / SANEAGO-GO – 2018)** No âmbito da Engenharia de Software, testes de unidade são aqueles realizados:

a) no sistema como um todo, de maneira que este mostre conformidade em relação à especificação de requisitos.
b) sobre as menores estruturas de código-fonte, como métodos e classes.
c) para verificação de integração entre módulos, de maneira que estes mostrem unidade.
d) em módulos ou unidades do sistema, de maneira que possam validar um componente específico.

77. **(COPEREVE / UFSC – 2018)** Considere as seguintes afirmativas a respeito de teste de software e assinale a alternativa correta.

I. O teste de unidade concentra o esforço de verificação na menor unidade de design de software.

II. O teste de unidade concentra-se na lógica de processamento interno e nas estruturas de dados dentro dos limites de um componente.

III. O teste de integração se concentra em ações visíveis pelo usuário e na saída reconhecível pelo usuário do sistema.

IV. O teste de integração é uma técnica sistemática para a construção da arquitetura de software, ao mesmo tempo em que realiza testes para descobrir erros associados às interfaces.

a) Somente as afirmativas I, II e IV estão corretas.
b) Todas as afirmativas estão corretas.
c) Somente as afirmativas I e II estão corretas.
d) Somente as afirmativas I e III estão corretas.

e) Somente as afirmativas II, III e IV estão corretas.

78. **(CONSUPLAN / CÂMARA DE BELO HORIZONTE-MG – 2018)** Na análise e projeto de sistemas, uma fase considerada muito importante é a de testes. Diversos tipos de testes são executados, desde a fase inicial até a implantação do novo sistema. Os testes têm como objetivo verificar a funcionalidade do sistema se o sistema atende ao que foi projetado. Quatro estágios de testes são conhecidos e cada um tem os seus respectivos tipos de testes. Dois tipos de testes são: testes da caixa preta e testes da caixa branca. Assinale a alternativa que apresenta corretamente qual estágio esses tipos de testes pertencem:

a) Testes de Sistema.
b) Testes de Unidade.
c) Testes de Aceitação.
d) Testes de Integração.

79. **(PR4 / UFRJ – 2018)** Com relação a teste de software, quando questionado sobre a construção de um produto corretamente, a referência se dá ao conjunto de atividades que garantem que o software implemente corretamente uma função específica. Este conceito se refere à:

a) validação.
b) homologação.
c) aceitação.
d) recuperação.
e) verificação.

80. **(PR4 / UFRJ – 2018)** Considere o seguinte texto: Uma técnica sistemática para construir a estrutura do programa enquanto, ao mesmo tempo, conduz testes para descobrir erros associados às interfaces. O objetivo é tomar componentes testados em nível de unidade e construir a estrutura de programa determinada pelo projeto. A afirmação apresentada está se referindo ao teste de:

a) integração.
b) unidade.
c) validação.
d) sistema.
e) depuração.

81. **(CESGRANRIO / TRANSPETRO – 2018)** Entre as técnicas de teste de software, aquela que gera versões levemente modificadas de um programa sob teste e exercita tanto o programa original quanto os programas modificados, procurando diferenças entre essas formas, é conhecida como testes:

a) aleatórios
b) exploratórios

c) de mutação
d) de perfil operacional
e) baseados em fluxo de controle

82. (FAURGS / BANRISUL – 2018) _____ verifica novamente os casos de teste aprovados em versões prévias do software e assim protege contra alterações indesejadas. Realiza-se durante a manutenção, para mostrar que as modificações efetuadas estão corretas, ou seja, que os novos requisitos implementados funcionam como o esperado e que os requisitos anteriormente testados continuam válidos. Assinale a alternativa que preenche corretamente a lacuna do texto acima.

a) Teste de regressão
b) Teste de sistema
c) Inspeção
d) Refatoração
e) Teste de integração

83. (UFLA / UFLA – 2018) Sobre teste de regressão de software, são verdadeiras as afirmativas abaixo, EXCETO:

a) Consiste na reexecução de testes previamente realizados no software para garantir que modificações ou novas funcionalidades não introduzam comportamentos indesejáveis ou erros adicionais.
b) É sempre prático e viável executar o teste de regressão após cada modificação do software.
c) Embora normalmente seja conduzido de forma automática, o teste de regressão pode também ser conduzido manualmente.
d) Teste de regressão pode ser configurado para execução em servidores de integração contínua.

84. (CESGRANRIO / IBGE – 2016) O sistema que controla as reservas dos clientes de uma rede hoteleira funciona apenas na Web. Entretanto, há uma demanda crescente para que a empresa disponibilize um aplicativo para smartphones. Para oferecer um aplicativo no menor prazo possível, a gerência de TI estabeleceu duas exigências: a primeira é que o novo sistema deve reutilizar ao máximo os módulos atualmente empregados, e a segunda é que a equipe de desenvolvimento deve garantir que as modificações a serem feitas não introduzirão defeitos inexistentes no sistema atual, além de continuar a atender a todos os requisitos anteriormente definidos.

O tipo de teste que deve ser empregado para que a equipe de desenvolvimento atenda à segunda exigência é denominado teste de:

a) estresse
b) volume

c) usabilidade
d) regressão
e) configuração

85. (IF-PE / IF-PE – 2016) Em relação aos Testes na Engenharia de Software, qual é o que se refere ao reteste de uma unidade, integração ou sistema, após uma modificação, a fim de verificar se a mudança não introduziu novas falhas?

a) Teste de integração.
b) Teste de regressão.
c) Teste alfa.
d) Teste beta.
e) Teste funcional.

86. (FCM / IF Farroupilha-RS – 2016) O teste de software pode ser realizado de diversas formas. Mesmo assim, existem técnicas que podem ser utilizadas para encontrar falhas no software. Analise as afirmativas abaixo:

I - O teste de regressão tem por finalidade repetir o teste em um programa já testado depois de haver uma modificação.

II - O teste de desempenho tem por finalidade elaborar casos de teste que possam subverter as verificações de segurança do programa.

III - O teste de caixa branca trabalha diretamente sobre o código fonte do componente de software.

IV - O teste de caixa preta trabalha diretamente sobre o código fonte do componente de software.

Estão corretas as afirmativas:

a) I e II.
b) I e III.
c) II e IV.
d) II e III.
e) I, II, III e IV.

87. (IESES / TRE-MA – 2015) É notório e de comprovado valor que os testes são a melhor maneira de se garantir a qualidade de um software. O teste de regressão é um dos tipos de testes, que tem por objetivo?

a) Certificar que os testes unitários foram realizados antes da própria implementação.

b) Garantir que as novas implementações não tenham quebrado as funcionalidades já existentes.
c) Avaliar se a especificação feita pelo analista de sistemas está correta.
d) Testar o menor nível de detalhe, normalmente este teste é realizado pelo programador.

88. **(CESGRANRIO / Banco da Amazônia – 2014)** Um tipo de teste de validação possui as seguintes características:

- Realizado na instalação dos desenvolvedores.
- Conduzido em um ambiente controlado.
- Conta com a participação de usuários e desenvolvedores.

Esse tipo de teste é chamado de:

a) Teste alfa
b) Teste beta
c) Teste de estresse
d) Teste de regressão
e) Teste de desempenho

89. **(CESGRANRIO / IBGE – 2014)** Antes de lançar seu próximo produto, uma empresa de desenvolvimento de software costuma convidar seus principais clientes para testar uma versão "quase final". Esses clientes são convidados ao local de desenvolvimento e são observados enquanto utilizam o software e registram erros e problemas no uso.

Essa estratégia de teste em um ambiente controlado é conhecida como teste:

a) alfa
b) beta
c) de stress
d) de integração
e) de aceitação do cliente

90. **(CESGRANRIO / IBGE – 2014)** No ciclo de desenvolvimento de sistemas, os testes são de suma importância e podem, dependendo do porte do sistema, ser bastante complexos, exigindo que seu planejamento e realização sejam divididos em fases. Em uma dessas fases, os testes são realizados por um grupo restrito de usuários finais do sistema, que simulam operações de rotina do sistema, de modo a verificar se seu comportamento está de acordo com o solicitado. Essa fase é denominada teste de:

a) integração
b) unidade
c) sistema
d) operação

e) aceitação

91. (CESGRANRIO / IBGE – 2014) Preocupado com os constantes erros nos sistemas entregues aos usuários, um analista de desenvolvimento resolveu realizar testes conforme o modelo V. A correspondência da validação dos modelos por fases do processo de software, de acordo com esse modelo, está representada em:

a) Teste unitário ⟶ Modelagem de requisitos
b) Teste de sistema ⟶ Desenvolvimento de componentes
c) Teste de aceitação ⟶ Geração de código
d) Teste de integração ⟶ Arquitetura do sistema
e) Teste de caixa preta ⟶ Desenho das estruturas de dados

92. (IBFC / TRE-AM – 2014) Assinale o nome do teste de software que consiste tipicamente na aplicação de versões mais recentes do software, para garantir que não surgiram novos defeitos em componentes já analisados:

a) testes unitários.
b) testes de integração.
c) testes de usabilidade.
d) testes de regressão.

93. (FUMARC / AL-MG – 2014) Tipo de teste que consiste em aplicar em cada nova versão de um software todos os testes que já foram aplicados nas versões anteriores, possibilitando a identificação dos impactos das implementações da nova versão em funcionalidades que já foram testadas anteriormente e não foram modificadas, é denominado:

a) Teste Caixa-branca.
b) Teste Caixa-preta.
c) Teste de Regressão.
d) Teste Unitário.

94. (CCV-UFC / UFC – 2013) O principal objetivo do teste de regressão de software é:

a) Identificar defeitos ou erros no sistema em situação de sobrecarga do sistema (ou parte dele).
b) Verificar a existência de defeitos após alterações em um sistema (ou parte dele) já testado.
c) Identificar defeitos através da inspeção do código-fonte do sistema (ou parte dele).
d) Identificar defeitos através da análise estática do sistema (ou parte dele).
e) Verificar a existência de defeitos no fluxo excepcional.

95. (CESGRANRIO / EPE – 2012) Em uma discussão sobre testes, um grupo de programadores emitiu as afirmativas a seguir.

I - Durante um teste, é possível provar apenas a existência de erros, não sua ausência.

II - Durante um teste de validação, são construídos casos de teste com a finalidade de expor defeitos.

III - Na verificação, procura-se saber se o produto está sendo construído de forma correta.

Estão corretas as afirmativas:

a) I, apenas.
b) II, apenas.
c) III, apenas.
d) I e III, apenas.
e) I, II e III.

96. (ESAF / CVM – 2010) Teste de Equivalência de Classe é:

a) É uma técnica que trabalha por dedução física.

b) É uma técnica que tem por objetivo primário reduzir o número de casos de testes.

c) É uma técnica que trabalha por dedução indutiva.

d) É uma técnica que tem por objetivo primário ampliar o número de casos de testes.

e) É uma técnica que tem por objetivo primário montar um conjunto de regras de decisão a partir de uma tabela.

97. (FEPESE / SEFAZ-SC – 2010) Analise a definição abaixo.

Teste de software que procura descobrir erros por meio da reaplicação parcial dos testes a um programa modificado.

Assinale a alternativa que cita corretamente o conceito ao qual se refere a definição.

a) Teste de sistema
b) Teste de unidade
c) Teste de regressão
d) Teste de integração
e) Teste de requisitos

98. (NCE-UFRJ / UFRJ – 2008) Considere as seguintes afirmativas sobre testes de software:

I - O teste de regressão consiste na re-execução de testes já executados para garantir que modificações introduzidas não geraram efeitos colaterais.

II - O teste fumaça (smoke test) é um tipo de teste de integração que é executado diariamente.
III - O teste de validação focaliza ações e saídas tais como percebidas pelo usuário final.

A(s) afirmativa(s) correta(s) é/são somente:

a) I
b) II
c) III
d) I e II
e) I, II e III

GABARITO

1. LETRA B
2. LETRA B
3. LETRA B
4. LETRA C
5. LETRA C
6. LETRA E
7. LETRA A
8. LETRA E
9. LETRA E
10. LETRA C
11. LETRA D
12. LETRA E
13. LETRA C
14. LETRA C
15. LETRA D
16. LETRA C
17. LETRA B
18. LETRA A
19. LETRA A
20. LETRA A
21. LETRA E
22. LETRA D
23. LETRA C
24. LETRA A
25. CORRETO
26. LETRA D
27. LETRA D
28. LETRA D
29. LETRA D
30. LETRA C
31. LETRA C
32. LETRA B
33. LETRA A
34. LETRA A
35. LETRA A
36. LETRA D
37. LETRA E
38. LETRA A
39. LETRA E
40. LETRA E
41. LETRA D
42. LETRA E
43. LETRA C
44. LETRA C
45. LETRA A
46. LETRA C
47. LETRA B
48. LETRA B
49. ERRADO
50. ERRADO
51. CORRETO
52. ERRADO
53. LETRA B
54. LETRA A
55. LETRA E
56. LETRA B
57. LETRA D
58. LETRA B
59. LETRA D
60. LETRA B
61. LETRA A
62. LETRA D
63. LETRA A
64. LETRA A
65. LETRA C
66. LETRA E
67. LETRA D
68. LETRA D
69. LETRA A
70. LETRA A
71. LETRA B
72. LETRA C
73. LETRA B
74. LETRA A
75. LETRA D
76. LETRA B
77. LETRA A
78. LETRA B
79. LETRA E
80. LETRA A
81. LETRA C
82. LETRA A
83. LETRA B
84. LETRA D
85. LETRA B
86. LETRA B
87. LETRA B
88. LETRA A
89. LETRA A
90. LETRA E
91. LETRA D
92. LETRA D
93. LETRA C
94. LETRA B
95. LETRA E
96. LETRA B
97. LETRA C
98. LETRA E

RPA

Conceitos Básicos

INCIDÊNCIA EM PROVA: BAIXA

O papo agora é RPA (*Robotic Process Automation*)! Imagine que você tem um assistente pessoal altamente eficiente que pode realizar tarefas repetitivas para você, como enviar e-mails, preencher planilhas e baixar arquivos. Esse assistente pode executar essas tarefas de forma rápida e precisa, sem cometer erros ou se cansar. **O RPA é como um assistente pessoal automatizado para realizar tarefas repetitivas e de rotina, com o objetivo de melhorar a produtividade e eficiência.**

Ele usa software especializado para imitar as ações humanas em sistemas de software, executando tarefas repetitivas e rotineiras de maneira mais rápida e precisa do que um ser humano poderia fazer manualmente. Isso libera os funcionários de tarefas tediosas e repetitivas, permitindo que eles se concentrem em atividades mais estratégicas e criativas. Esse tema tem ganhado cada vez mais destaque no mundo corporativo e nas discussões sobre automação de processos. Vejamos...

- **Robotic (Robótico):** refere-se ao uso de software, ou robôs de software, que são programados para realizar tarefas repetitivas e de rotina de forma automatizada. Esses robôs podem realizar diversas tarefas, desde a captura e entrada de dados até a realização de cálculos e tomadas de decisão, tudo sem a intervenção humana.

- **Process (Processo):** refere-se a um conjunto de atividades que são executadas para atingir um objetivo específico. Isso significa que os robôs de software são programados para executar tarefas em uma sequência lógica, seguindo um processo estabelecido, para atingir um objetivo específico.

- **Automation (Automação):** refere-se à capacidade de executar tarefas de forma automatizada, sem a necessidade de intervenção humana. A automação é alcançada por meio da programação de robôs de software que executam tarefas rotineiras e repetitivas de forma consistente e precisa, liberando as pessoas outras atividades que exigem outras habilidades.

Nos anos 2000, foram criados os primeiros softwares de automação de processos. No entanto, a tecnologia só começou a se popularizar na última década, com a evolução da inteligência artificial e da robótica. **Hoje em dia, as tendências do RPA indicam que a tecnologia continuará a evoluir, se tornando cada vez mais sofisticada e capaz de realizar tarefas mais complexas e cognitivas.** Sério, já existem coisas absurdas que podem ser feitas com RPA!

Devido aos benefícios oferecidos pelo RPA – como redução de custos, aumento da produtividade e melhoria da qualidade do trabalho – muitas empresas já estão investindo em soluções de

automação. *Legal, mas como funciona essa tecnologia?* Para implementá-lo, é necessário definir uma arquitetura e selecionar os componentes necessários. **Isso envolve a escolha das ferramentas de automação, dos robôs que executarão as tarefas e da plataforma de gerenciamento do processo de automação.**

O processo de automação consiste em definir os fluxos de trabalho e os processos que serão automatizados. **Esses processos são, então, modelados em um software que irá orientar o robô a realizar as tarefas.** É importante que esse processo seja cuidadosamente planejado e que todas as etapas do processo sejam mapeadas com precisão.

A automação também envolve integrações com outros sistemas e plataformas. Isso permite que o robô acesse e utilize informações de diferentes fontes e realize tarefas em vários sistemas. **Uma vez implementado, o RPA pode ser configurado para rodar em diferentes horários, como em horários de menor demanda, para evitar impacto na performance dos sistemas ou nos usuários finais.**

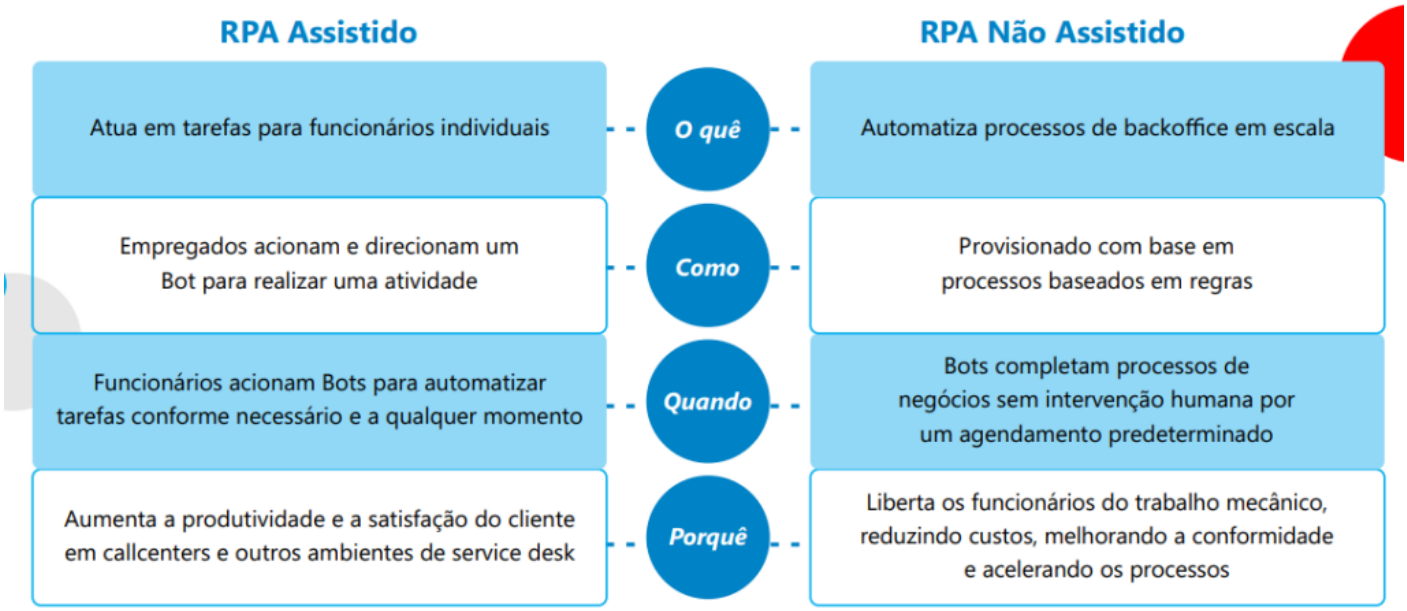
Existem diversas ferramentas que oferecem diferentes soluções para automação de processos em empresas de diversos setores, tais como UiPath, Automation Anywhere, Blue Prism, WorkFusion e Kofax – algumas gratuitas e outras pagas. **Cada ferramenta possui suas próprias vantagens e desvantagens, como a facilidade de uso, a escalabilidade, a segurança e a integração com outros sistemas.**

Por exemplo, a UiPath é conhecida por sua facilidade de uso e integração com outros sistemas, enquanto a Blue Prism é considerada uma ferramenta altamente escalável e segura.

Além disso, existem diversos exemplos de uso do RPA em diferentes setores, como financeiro, de saúde, de logística e varejo. **Algumas empresas têm usado o RPA para automatizar processos de faturamento, atendimento ao cliente, processamento de pedidos, monitoramento de estoque e processamento de sinistros, entre outros.** É importante saber também que existem três tipos de RPA:

TIPO DE RPA	DESCRIÇÃO
ASSISTIDO	Também conhecido como RPA autônomo ou desassistido, esse tipo de RPA executa tarefas de forma totalmente automática, sem intervenção humana. Ele é programado para executar tarefas de rotina em um horário específico ou em resposta a um evento específico.
NÃO ASSISTIDO	Também conhecido como RPA de área de trabalho, esse tipo de RPA trabalha em conjunto com os usuários finais, permitindo que eles controlem o processo e garantam a precisão dos dados. Ele automatiza tarefas em um ambiente de desktop e fornece orientação para o usuário.

HÍBRIDO	Esse tipo de RPA combina a automação assistida e não assistida para permitir a execução de processos completos que incluem tarefas repetitivas e não repetitivas. Ele é adequado para processos mais complexos, que requerem intervenção humana em alguns pontos, mas podem ser automatizados em outros.



Galera, eu já cheguei a utilizar RPA em meu trabalho na Secretaria do Tesouro Nacional (STN) porque é realmente muito útil. **No meu caso, a ideia era automatizar o preenchimento de campos de um sistema para realizar pagamentos.** Isso era feito manualmente todos os dias por servidor e simplesmente foi substituído por robozinho. *Vocês têm ideia de como isso ajuda a melhorar a produtividade em diversos contextos?* Pois é...

Os casos de sucesso incluem a redução de custos, aumento da eficiência e da produtividade, bem como a melhoria da qualidade do trabalho realizado pelos funcionários. Agora um exemplo internacional: a empresa norte-americana de seguros AIG implementou o RPA para automatizar processos manuais de faturamento e conseguiu reduzir o tempo de processamento de sinistros de horas para minutos.

A automação de tarefas repetitivas e rotineiras pode levar a uma redução significativa de custos e aumento da eficiência, mas não é só isso: ela ajuda a melhorar da qualidade do trabalho, a redução de erros e retrabalhos, oferece a possibilidade de trabalhar 24 horas por dia, 7 dias por semana, e a liberação dos funcionários para tarefas mais estratégicas e criativas. **Ao ajudar empresas a lidarem com um grande volume de dados, podemos ter uma tomada de decisão mais rápida e precisa.**

É claro que também há possíveis desvantagens, como a necessidade de um gerenciamento de projetos eficiente para garantir que os processos sejam automatizados de forma eficaz e eficiente. Além disso, a governança é importante para garantir a segurança dos dados e evitar erros e

inconsistências. **E é importante ter um monitoramento constante para garantir a continuidade da operação e o desempenho dos robôs de software – para tal, utiliza-se um Control Room.**

Também chamado de Sala de Controle, trata-se de uma plataforma centralizada que permite a monitoração, gerenciamento e controle das atividades dos robôs de software. **Ele é a interface de gerenciamento que permite que os usuários iniciem, parem e gerenciem os processos de automação, além de monitorar o desempenho dos bots e a saúde do sistema de automação.** O Control Room permite acessar o painel de controle, agendar tarefas e criar fluxos de automação.

Além disso, o Control Room também pode fornecer relatórios de atividades dos bots e insights sobre como otimizar a automação e melhorar a eficiência dos processos. Isso permite que os usuários monitorem e gerenciem a automação em tempo real e tomem decisões informadas com base em dados precisos. *Bacana?* Esse é um tema ainda pouco comum em provas, logo vamos ver as questões que eu encontrei.

(SEFAZ/AM – 2022) A respeito do Robotic Process Automation (RPA), assinale a afirmativa correta.

a) Possui baixa precisão nos resultados.
b) Executa tarefas humanas de rotina.
c) Realiza tarefas desde que envolva cálculos.
d) Necessita de customização para interagir com outros sistemas.
e) Dispensa a utilização de controlroom (sala de controle).

Comentários: (a) Errado, ele tem alta precisão nos resultados, já que é programado para seguir um conjunto de regras e padrões, eliminando erros humanos; (b) Correto, ele realmente executa tarefas humanas repetitivas ou de rotina; (c) Errado, ele pode realizar uma ampla gama de tarefas, não se limitando apenas a cálculos; (d) Errado, embora possa precisar de alguma customização, ele é projetado para interagir facilmente com outros sistemas e aplicativos; (e) Errado, O RPA geralmente requer o uso de um controlroom para monitorar e gerenciar as atividades do robô de software.

(BNB – 2022) RPA consiste num microcontrolador de 8 bits, com componentes complementares, para facilitar a programação e a incorporação em outros circuitos.

Comentários: é o que? RPA é uma tecnologia de software que usa robôs de software (bots) para automatizar tarefas repetitivas e de rotina, liberando os funcionários para se concentrarem em outras atividades mais estratégicas – não há nenhuma relação com microcontrolador de 8 bits.

(Eletrobrás – 2022) Uma empresa pública precisa fazer uso de RPA (Automação de Processos Robotizados) para validar endereços, CNPJs, situação na Receita Federal e consulta ao Serasa relacionados aos fornecedores de produtos, para garantir que as propostas desses fornecedores estejam dentro das bases legais exigidas, em caso de

licitações ou similares. Para essa situação, faz-se necessário que seja utilizado um modelo de RPA não assistido.

A configuração dessa RPA deve ser de tal modo que haja:

a) acionamento das RPAs via ação humana, mas sem controle posterior do usuário que disparou a RPA, com cada obtenção de dados avisando ao usuário disparador sobre o dado recuperado.

b) execução local dessa RPA em uma estação de trabalho específica, com modo de atividades de front-office, sem intervenção humana, mas programada por temporização.

c) execução local assistida por meio remoto por um supervisor humano, que vai acionar a execução através de um evento de uma máquina externa, controlando passo a passo a obtenção dos dados.

d) manipulação dos dados em segundo plano, em servidores de back-end, sem intervenção humana, sendo essas RPAs acionadas por eventos ou através de execução programada.

e) manipulação de dados em servidores front-end, acionando buscas nas bases de dados necessárias, sem intervenção humana, mas disparadas sem evento auxiliar ou execução programada.

Comentários: (a) Errado, isso é típico de um RPA assistido, já que a ação humana é necessária para acionar a RPA; (b) Errado, execução local com modo de atividades de front-office e com uso de um temporizador é típico de RPA assistido; (c) Errado, se é assistida por um supervisor humano, trata-se de RPA assistido; (d) Correto, ele permite a automação sem intervenção humana e a validação de dados em grande quantidade, em servidores de back-end. Além disso, o fato de poder acionar as RPAs por meio de eventos ou execução programada permite uma automação mais flexível e adaptável às necessidades da empresa; (e) Errado, se é front-end e disparada sem execução programada é RPA assistido (Letra D).

(CEBRASPE / AGER-MT – 2023) 1. Para criar, por meio de bots, uma forma de automatizar tarefas repetitivas de software, como entrada de dados em formulários, usando-se tecnologias que imitam tarefas de back-office de trabalhadores humanos, uma solução seria a implementação de:

a) MVP (minimum viable product).
b) tratamento do débito técnico.
c) RPA (robotic process automation).
d) técnicas de refatoração de software.
e) low code.

Comentários: (a) Errado. MVP (Minimum Viable Product) refere-se ao desenvolvimento de um produto com as funcionalidades mínimas necessárias para ser lançado, não à automação de tarefas repetitivas;

(b) Errado. Tratamento do débito técnico envolve resolver problemas no código que foram adiados, mas não se relaciona diretamente com a automação de tarefas;

(c) Correto. RPA é a tecnologia usada para automatizar tarefas repetitivas de software, como a entrada de dados em formulários, imitando ações humanas em processos de back-office;

(d) Errado. Refatoração de software consiste em melhorar a estrutura interna do código sem alterar seu comportamento externo, mas não é uma forma de automatizar tarefas repetitivas;

(e) Errado. Low code é uma abordagem que permite criar aplicações com mínima codificação manual – não é especificamente voltado para a automação de tarefas repetitivas (Letra C).

(FCC / COPERGÁS-PE – 2023) Robot Process Automation – RPA é uma tecnologia de software

a) não invasiva que pode ser implementada rapidamente para acelerar a transformação digital. No entanto, não deve ser utilizada para automatizar workflows que envolvem sistemas legados sem APIs, infraestrutura de desktop virtual (VDI) ou acesso a banco de dados.

b) que pode ser utilizada por funcionários sem formação em TI, pois estes podem desenvolver assistentes robóticos com base no desenvolvimento slow-code/no-code, criando suas próprias automações até para aplicações complexas.

c) cuja implementação inicia-se pela automação de processos grandes e complexos, como sistemas centrais, áreas funcionais e atividades corporativas importantes. É fator crítico de sucesso a união da abordagem zero-top-down com a melhoria contínua dos processos.

d) diversa da Inteligência Artificial (IA), mas recursos avançados de IA, como modelos de Machine Learning (ML), processamento de linguagem natural (NLP) etc, podem ser inseridos em robôs. Os robôs seriam configurados para aplicar modelos de ML para processos e análises automatizados de tomada de decisão, por exemplo.

e) que permite às empresas lidar com automações complexas sem a necessidade de ter uma equipe de TI com desenvolvedores e arquitetos de soluções, uma vez que chatbots automatizados e robôs que compreendam documentos baseados em recursos de machine learning podem ser criados com desenvolvimento no-code.

Comentários: (a) Errado. Embora RPA seja uma tecnologia não invasiva e possa ser implementada rapidamente, ela é frequentemente utilizada justamente para automatizar workflows que envolvem sistemas legados, incluindo aqueles sem APIs e que utilizam infraestrutura de desktop virtual (VDI);

(b) Errado. RPA pode ser usada por funcionários sem formação em TI para desenvolver automações simples, mas aplicações complexas geralmente exigem conhecimento técnico avançado e não podem ser realizadas exclusivamente com desenvolvimento no-code ou slow-code;

(c) Errado. A implementação de RPA geralmente começa com a automação de processos menores e mais repetitivos para gerar resultados rápidos e valor imediato, não por processos grandes e complexos. A abordagem top-down mencionada não é a prática recomendada para RPA, que prefere uma abordagem mais incremental;

(d) Correto. RPA é diferente da Inteligência Artificial (IA), mas pode ser complementada por recursos de IA, como Machine Learning (ML) e Processamento de Linguagem Natural (NLP), para criar robôs mais avançados que automatizam processos e análises complexas;

(e) Errado. Embora RPA permita a automação de processos sem exigir uma equipe de TI especializada para tarefas simples, automações complexas ainda requerem envolvimento de desenvolvedores e arquitetos de soluções, especialmente quando envolve machine learning ou compreensão de documentos (Letra D).

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.