Estratégia Concursos

Aula 00 - Profs. Diego Carvalho e Renato da Costa

DPDF (Analista de Apoio à Assistência Judiciária - Informática - Desenvolvimento de Sistemas)

Engenharia de Software

Autor:
Diego Carvalho, Fernando Pedrosa Lopes

11 de Abril de 2024

Índice

QUALIDADE DE SOFTWARE

Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

Pessoal, por que a qualidade de software se tornou um assunto tão importante a ponto de ser estudada em diversas bibliografias e até cair em concurso público? **Bem, o clamor por maior qualidade de software começou realmente quando o software passou a se tornar cada vez mais integrado em todas as atividades de nossas vidas**. Na década noventa, empresas desperdiçavam bilhões de dólares em software que não apresentava as funcionalidades e requisitos prometidos.

Pior ainda, tanto o governo como as empresas ficavam cada vez mais preocupados com o fato de que **uma falha grave de software poderia inutilizar importantes infraestruturas**, aumentando o custo em dezenas de bilhões. Na virada do século, uma revista de tecnologia deu manchete: "*Chega de desperdiçar US$ 78 bilhões por ano*", lamentando o fato de que "*as empresas americanas gastavam bilhões em softwares que não faziam o que supostamente deveriam fazer*".

Outra revista chamada Information Week escreveu à época:

> *Apesar das boas intenções, código mal feito continua a ser o "fantasma" do mercado de software, sendo responsável por até 45% do tempo de inatividade dos sistemas computacionais e custando às empresas americanas cerca de US$ 100 bilhões no último ano em termos de manutenção e redução da produtividade, afirma o Standish Group, uma empresa de pesquisa de mercado. Isso não inclui o custo da perda de clientes insatisfeitos. Pelo fato de as empresas de TI escreverem aplicações que dependem de pacotes de software de infraestrutura, código de má qualidade pode causar estragos em aplicações personalizadas bem como... Qual o prejuízo causado por software de má qualidade? As definições variam, mas especialistas dizem que bastam três ou quatro defeitos a cada 1.000 linhas de código para fazer com que um programa execute de forma inadequada. Acrescente a isso que a maioria dos programadores insere cerca de um erro a cada 10 linhas de código escrito, multiplicados por milhões de linhas de código em vários produtos comerciais. Assim, deduz-se que o custo dos fornecedores de software será de pelo menos a metade dos seus orçamentos para a realização dos testes e correção dos erros. Percebeu o tamanho do problema?*

Hoje em dia, a qualidade de software continua a ser um problema, mas a quem culpar? Os clientes culpam os desenvolvedores, argumentando que práticas descuidadas levam a um software de baixa qualidade. Os desenvolvedores de software culpam os clientes (e outros interessados), argumentando que datas de entrega absurdas e um fluxo contínuo de mudanças os forçam a entregar software antes de eles estarem completamente validados. **Bem... ambos têm razão!**

Antes de continuar, precisamos definir o que é qualidade! A qualidade é algo pelo qual nos esforçamos para obter nos produtos, processos e serviços. Vejamos o que o dicionário diz:

> *Qualidade: característica inerente ou diferenciada; uma propriedade. b. Um traço pessoal, especialmente um traço de caráter. 2. Caráter essencial; natureza. 3.a. Superioridade de natureza. b. Grau ou classificação de excelência.*

Logo, a qualidade não é um atributo ou uma característica singular – é multidimensional e pode ser possuída por um produto ou por um processo. A qualidade do produto está concentrada na criação do produto certo, enquanto a qualidade do processo está concentrada na criação correta do produto. A definição do dicionário é muito genérica, então vamos ver a definição de qualidade de acordo com o Processo Unificado:

> *Qualidade: característica de ter demonstrado a realização da criação de um produto que atende ou excede os requisitos acordados, conforme avaliado por medidas e critérios acordados, e que é criado em um processo acordado.*

Obter qualidade não é apenas "atender a requisitos" ou produzir um produto que atende às expectativas e necessidades dos usuários. Pelo contrário, também inclui a identificação das medidas e dos critérios para demonstrar a obtenção da qualidade e a implementação de um processo para garantir que o produto por ele criado tenha atingido o grau desejado de qualidade e possa ser repetido e gerenciado. *Bem, vocês sabem o que é a qualidade de projeto?*

Ela se refere às características que os projetistas especificam para um produto. A qualidade dos materiais, as tolerâncias e as especificações de desempenho, todos são fatores que contribuem para a qualidade de um projeto. Quanto mais materiais de alta qualidade forem usados, tolerâncias mais rígidas e níveis de desempenho maiores forem especificados, a qualidade de projeto de um produto aumentará se o produto for fabricado de acordo com essas especificações.

No desenvolvimento de software, a qualidade de um projeto engloba o grau de atendimento às funções e características especificadas no modelo de requisitos. Por fim, antes de falarmos da qualidade de software, nós temos também a qualidade de conformidade. *O que é isso, Diegão?* **Cara, é a qualidade que se foca no grau em que a implementação segue o projeto e no grau em que o sistema resultante atende suas necessidades e as metas de desempenho.**

Professor, a qualidade do projeto e a qualidade de um produto é suficiente para deixar o cliente satisfeito? Não! A qualidade é evidentemente importante, começando pela qualidade do projeto de criação do produto, passando pelo grau de conformidade do produto em relação aos requisitos definidos anteriormente no projeto. **No entanto, há outros fatores: não adianta nada tudo isso sem entregar o projeto dentro do orçamento e prazo previsto, por exemplo!**

(MEC – 2015) A qualidade de software é fundamentada nas necessidades do usuário. A falta de conformidade aos requisitos de software é determinante para a falta de qualidade de software.

_____

Comentários: a qualidade de software é realmente fundamentada nas necessidades do usuário. Ademais, a falta de conformidade de requisitos de software é – sim – determinante para a falta de qualidade de software (Correto).

Já a qualidade de software, no sentido mais geral, pode ser definida como uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam. Não há dúvida nenhuma de que essa definição pode ser modificada ou estendida e debatida interminavelmente. Três autores – **McCall, Richards e Walters** – criaram uma proposta de categorização de fatores de qualidade de software.

Esses fatores se focam em três aspectos importantes de um produto de software: características operacionais; a habilidade de suportar mudanças; e a adaptabilidade a novos ambientes. Galera, entendam que é difícil – e em alguns casos impossível – desenvolver medidas diretas desses fatores de qualidade. **Algumas delas só podem ser medidas indiretamente**, no entanto esses fatores conseguem nos dá uma sólida indicação da qualidade de um software. *Bacana?* Vejamos:

FATORES	DESCRIÇÃO
CORREÇÃO	O quanto um programa satisfaz a sua especificação e atende aos objetivos da missão do cliente.
CONFIABILIDADE	O quanto se pode esperar que um programa realize a função pretendida com a precisão exigida.
EFICIÊNCIA	A quantidade de recursos computacionais e código exigidos por um programa para desempenhar sua função.
INTEGRIDADE	O quanto o acesso ao software ou dados por pessoas não autorizadas pode ser controlado.
USABILIDADE	Esforço necessário para aprender, operar, preparar a entrada de dados e interpretar a saída de um programa.
FACILIDADE DE MANUTENÇÃO	Esforço necessário para localizar e corrigir um erro em um programa.
FLEXIBILIDADE	Esforço necessário para modificar um programa em operação.
TESTABILIDADE	Esforço necessário para testar um programa de modo a garantir que ele desempenhe a função destinada.

PORTABILIDADE	Esforço necessário para transferir o programa de um ambiente de hardware e/ou software para outro.
REUSABILIDADE	O quanto um programa [ou partes de um programa] pode ser reutilizado em outras aplicações.
INTEROPERABILIDADE	Esforço necessário para integrar um sistema a outro.

A Norma ISO 9126 também busca identificar atributos fundamentais da qualidade de software. **O padrão identifica seis atributos fundamentais de qualidade:**

FATORES	DESCRIÇÃO
FUNCIONALIDADE	Trata-se do grau com que o software satisfaz às necessidades declaradas conforme indicado pelos seguintes subatributos: adequabilidade, exatidão, interoperabilidade, conformidade e segurança.
CONFIABILIDADE	Trata-se da quantidade de tempo que o software fica disponível para uso conforme indicado pelos seguintes subatributos: maturidade, tolerância a falhas, facilidade de recuperação.
USABILIDADE	Trata-se do grau de facilidade de utilização do software conforme indicado pelos seguintes subatributos: facilidade de compreensão, facilidade de aprendizagem, operabilidade.
EFICIÊNCIA	Trata-se do grau de otimização do uso, pelo software, dos recursos do sistema conforme indicado pelos seguintes subatributos: comportamento em relação ao tempo, comportamento em relação aos recursos.
MANUTENIBILIDADE	Trata-se da facilidade com a qual uma correção pode ser realizada no software conforme indicado pelos seguintes subatributos: facilidade de análise, facilidade de realização de mudanças, estabilidade, testabilidade.
PORTABILIDADE	Trata-se da facilidade com a qual um software pode ser transposto de um ambiente a outro conforme indicado pelos seguintes subatributos: adaptabilidade, facilidade de instalação, conformidade, facilidade de substituição.

MNEMÔNICO DOS FATORES DE QUALIDADE SEGUNDO A ISO 9126

O gerenciamento da qualidade do software se preocupa em garantir que os sistemas de software desenvolvidos sejam adequados aos seus objetivos, isto é, devem atender às necessidades de seus usuários, deve ter um desempenho eficiente e confiável, e ser entregue dentro do prazo e custo. O uso de técnicas de gerenciamento da qualidade junto com novas tecnologias e métodos de teste levou a melhorias significativas no nível de qualidade de software nos últimos vinte anos.

As técnicas de gerenciamento de qualidade de software têm suas raízes em métodos e técnicas desenvolvidas nas indústrias de manufatura, em que os termos garantia de qualidade e controle de qualidade são amplamente utilizados. **Garantia de qualidade é a definição de processos e normas**

que devem levar a produtos de alta qualidade e à introdução de processos de qualidade no processo de fabricação.

Já o controle de qualidade é a aplicação desses processos para eliminar produtos que não possuem o nível de qualidade exigido. **Tanto a garantia quanto o controle de qualidade fazem parte do gerenciamento da qualidade.** Na indústria de software, algumas empresas veem garantia de qualidade como a definição procedimentos, processos e padrões para garantir que a qualidade do software seja alcançada.

Dito de outra forma, a garantia de qualidade também inclui todo o gerenciamento de configuração, atividades de verificação e validação aplicadas após a entrega de um produto por uma equipe de desenvolvimento. O gerenciamento da qualidade fornece uma verificação independente no processo de desenvolvimento de software, verificando os resultados do projeto para garantir que eles sejam consistentes com os padrões e metas organizacionais.

Em suma, pode-se dizer que o gerenciamento de qualidade estabelece procedimentos e padrões que objetivam o desenvolvimento de software com qualidade. *Vamos resumir o que vimos até agora?* Bem, a preocupação com a qualidade de sistemas de software cresceu à medida que o software passou a se tornar cada vez mais integrado em cada aspecto da vida cotidiano, mas é difícil desenvolver uma descrição completa sobre qualidade de software.

Nós definimos a qualidade como uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam. Foi proposta uma variedade de fatores para qualidade de software ao longo dos anos. **Todos tentam definir um conjunto de características que, se atingidas, levarão teoricamente a um software de alta qualidade.**

Os fatores de qualidade de McCall e da ISO 9126 estabelecem características como confiabilidade, usabilidade, facilidade de manutenção, etc como indicadores de que a qualidade existe. Todas as organizações envolvidas com software deparam com o dilema da qualidade de software. Em essência, todos querem construir sistemas de alta qualidade, **mas o tempo e o esforço necessários para produzir um software "perfeito" simplesmente não existem no mercado atual.**

A pergunta passa a ser: devemos construir software "*bom o suficiente*"? Embora muitas empresas façam isso, há um grande problema que deve ser considerado. **Independentemente da abordagem escolhida, a qualidade tem efetivamente um custo que pode ser discutido em termos de prevenção, avaliação e falha.** (1) Os custos de prevenção incluem todas as ações de engenharia de software que são desenvolvidas para, em primeiro lugar, evitar defeitos.

(2) Os custos de avaliação estão associados àquelas ações que avaliam os artefatos resultantes para determinar sua qualidade. (3) Os custos de falhas englobam o preço de falhas internas e os efeitos externos que a má qualidade gera. *Fechou?* **Não é um assunto complexo, é só um pouquinho decoreba, eu admito! Bem, no próximo tópico nós vamos nos aprofundar em alguns conceitos vistos nessa introdução.**

(TJDFT – 2008) O fator de qualidade flexibilidade de McCall é definido como a capacidade de um software de se adaptar a diferentes sistemas operacionais ou diferentes configurações de hardware.

_____

Comentários: o fator de qualidade **flexibilidade** trata do esforço necessário para modificar um programa em operação – a questão trata, na verdade, do fator de qualidade **portabilidade** (Errado).

(PRODEST/ES – 2014) Segundo a norma ISO 9126, e também pelos estudos de McCall, um dos fatores de qualidade que se aplicam ao software é a confiabilidade, que é definida como:

a) a facilidade de migrar o software de um ambiente computacional para outro.
b) a probabilidade de o software operar sem falhas durante um período de tempo.
c) o esforço dispendido para efetuar correções em um software
d) o nível de aproveitamento dos recursos computacionais pelo software.
e) o nível de facilidade do uso de um software.

_____

Comentários: (a) Errado, a questão trata da portabilidade; (b) Correto, a questão trata da confiabilidade; (c) Errado, a questão trata da facilidade de manutenção; (d) Errado, a questão trata da eficiência; (e) Errado, a questão trata da usabilidade (Letra B).

(MPE/MS – 2013) Segundo Pressman, os fatores categorizados quanto à operação do produto que afetam a qualidade de software são:

a) portabilidade, eficiência, testabilidade, integridade e flexibilidade.
b) interoperabilidade, testabilidade, portabilidade, eficiência e correção.
c) integridade, manutenibilidade, flexibilidade, portabilidade e correção.
d) correção, usabilidade, integridade, eficiência e confiabilidade.
e) flexibilidade, manutenibilidade, interoperabilidade, portabilidade e usabilidade.

_____

Comentários: bastava lembrar do nosso triângulo – os fatores quanto à operação são a correção, usabilidade, integridade, eficiência e confiabilidade (Letra D).

(Prefeitura de Caieiras/SP – 2015) A avaliação de fatores de qualidade revela-se como sendo de grande importância no desenvolvimento de software. Um dos conjuntos de fatores mais utilizados para tal finalidade são os fatores de qualidade de McCall, segundo os quais o fator:

a) confiabilidade enquadra-se no aspecto de transição do produto
b) correção enquadra-se no aspecto de revisão do produto.
c) eficiência enquadra-se no aspecto de operação do produto.
d) flexibilidade enquadra-se no aspecto de transição do produto.

e) portabilidade enquadra-se no aspecto de operação do produto.

_____

Comentários: (a) Errado, enquadra-se no aspecto de operação do produto; (b) Errado, enquadra-se no aspecto de operação do produto; (c) Correto, enquadra-se no aspecto de operação do produto; (d) Errado, enquadra-se no aspecto de revisão do produto; (e) Errado, enquadra-se no aspecto de transição do produto (Letra C).

Garantia e Controle de Qualidade

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Seus lindos, vamos ver um pouquinho agora sobre a diferença entre Garantia de Qualidade e Controle de Qualidade. A garantia de qualidade está focada no processo de desenvolvimento de software e na prevenção de defeitos, já o controle de qualidade está focado no produto entregue ao usuário e a detecção e correção de defeitos. *Professor, você recomenda decorar isso?* Galera, se você entender isso, você já responde 95% das questões de prova sobre esse tema!

A garantia de qualidade é orientada ao processo e busca analisá-lo para descobrir problemas e oportunidades de melhoria com foco no monitoramento do processo – geralmente ocorre no início das fases do ciclo de vida de software. **Já o controle de qualidade é orientado ao produto e busca detectar problemas no produto entregue ao usuário** – geralmente ocorre no final das fases do ciclo de vida.

A garantia de qualidade cuida, por exemplo, da metodologia de desenvolvimento de software utilizada; já o controle de qualidade cuida da qualidade do software em si. **O controle de qualidade engloba um conjunto de ações que ajudam a garantir que cada produto resultante atinja suas metas de qualidade**, permitindo a uma equipe de software ajustar o processo quando qualquer um desses produtos deixe de atender às metas estabelecidas para a qualidade.

Já a garantia de qualidade estabelece a infraestrutura que suporta métodos sólidos de engenharia de software, gerenciamento racional do projeto e ações de controle de qualidade. **Ela consiste em um conjunto de funções de auditoria e de relatórios que possibilita uma avaliação da efetividade e da completude das ações de controle de qualidade**. É importante ressaltar que ela vem antes e após o controle de qualidade.

O controle de qualidade engloba, portanto, tanto a verificação quanto a validação do software. Galera, é importantíssimo que vocês entendam esses conceitos! *Vamos ver um exemplo?* Na imagem a seguir temos uma linha de montagem de um carro. O responsável pelo controle de qualidade buscará encontrar defeitos específicos. Logo, na imagem a seguir, ele vai ligar o carro com a chave e ver se está funcionando.

Agora imaginem que esse não seja um problema pontual, uma vez que tem ocorrido com frequência que os carros montados não estejam ligando. O responsável pela garantia de qualidade buscará identificar no processo da linha de montagem o que está ocorrendo de errado para esses carros não estarem funcionando corretamente. **Ele poderá, por exemplo, identificar um problema no momento de montar a chave e melhorar esse processo para evitar novos erros.**

Conseguiram pegar a diferença? **O controle de qualidade está focado na detecção e correção de defeitos no produto em relação aos requisitos especificados anteriormente**. Já a garantia de qualidade está focada na prevenção de defeitos via processo em relação aos métodos e técnicas utilizados, podendo ocorrer no início ou no fim do ciclo de vida de desenvolvimento do software. Vamos ver uma tabela com as principais diferenças entre garantia e controle de qualidade:

GARANTIA DE QUALIDADE	CONTROLE DE QUALIDADE
Garantia da qualidade garante que o processo é definido e apropriado.	As atividades de controle da qualidade focam na descoberta de defeitos específicos.
Metodologia e padrões de desenvolvimento são exemplos de garantia da qualidade.	Um exemplo de controle da qualidade poderia ser: *"Os requisitos definidos são os requisitos certos?"*
Garantia da qualidade é orientada a processo.	Controle da qualidade é orientado a produto.
Garantia da qualidade é orientada a prevenção.	Controle da qualidade é orientado a detecção.
Foco em monitoração e melhoria de processo.	Inspeções e garantia de que o produto de trabalho atenda aos requisitos especificados.
As atividades são focadas no início das fases no ciclo de vida de desenvolvimento de software.	As atividades são focadas no final das fases no ciclo de vida de desenvolvimento de software.
Garantia da qualidade garante que você está fazendo certo as coisas e da maneira correta.	Controle da qualidade garante que os resultados do seu trabalho são os esperados conforme requisitos.

(UFF – 2009) Segundo Pressman, na qualidade do software, as inspeções, revisões e testes utilizados ao longo do processo de software, para garantir que cada produto de trabalho satisfaça os requisitos estabelecidos, são conhecidas como:

a) garantia de qualidade;
b) custo da qualidade;
c) controle de qualidade;
d) reengenharia de processos;
e) gold plate.

Comentários: garantir que cada produto de trabalho satisfaça os requisitos estabelecidos é o objetivo do controle de qualidade (Letra C).

Verificação x Validação

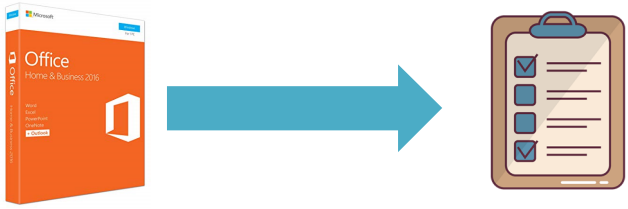
INCIDÊNCIA EM PROVA: ALTÍSSIMA

Pessoal, Roger Pressman diz em seu livro que durante e depois do processo de implementação, o software em desenvolvimento deve ser verificado para certificar-se de que ele atende a sua especificação e de que entrega a funcionalidade esperada pelas pessoas que pagam pelo software – **Verificação e Validação (V&V) é a denominação dada a esses processos e atividades, que ocorrem em cada estágio do processo de software.**

A V&V começa com revisões de requisitos e continua ao longo das revisões de projeto e das inspeções de código até o teste de produto. **No entanto, percebam que Validação e Verificação são coisas diferentes**! *E qual a diferença?* Boehm descreveu, de uma maneira simples e genial, a diferença entre esses dois conceitos por meio de duas perguntas: (1) Verificação: *estamos construindo o produto corretamente?* (2) Validação: *estamos construindo o produto correto?*

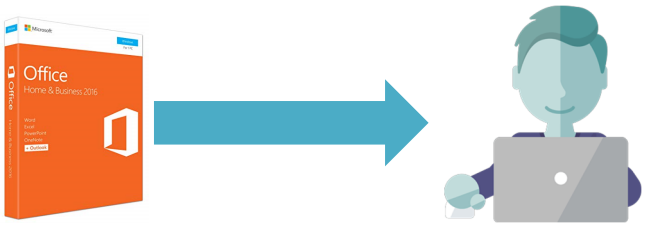
VERIFICAÇÃO (DEPENDE DA ESPECIFICAÇÃO)

ESTAMOS CONSTRUINDO O PRODUTO CORRETAMENTE?
O PRODUTO ESTÁ DE ACORDO COM AS ESPECIFICAÇÕES DO SISTEMA?



VALIDAÇÃO (DEPENDE DO USUÁRIO)

ESTAMOS CONSTRUINDO O PRODUTO CORRETO?
O PRODUTO ESTÁ DE ACORDO COM AS NECESSIDADES DO USUÁRIO?



A Verificação ocorre em ambiente de desenvolvimento do software e envolve a certificação de que o **software construído esteja de acordo com as funções e especificações dos usuários**! Já a Validação ocorre no ambiente do usuário e se certifica de que o software construído está de acordo

com as expectativas do cliente. *Ora, o produto está de acordo com as especificações? Ele satisfaz os anseios dos usuários?*

Eu peço a vocês! **Não, na verdade eu imploro que vocês memorizem a diferença entre esses dois conceitos!** É muito muito muito simples, mas eu já me cansei das incontáveis vezes que eu vi questões de prova tentando confundir os candidatos com esses conceitos e obtendo êxito. *Diego, como você decorou?* Muito simples! Eu sempre me lembrava que a **vERificação** ocorre em relação à **Especificação de Requisitos**! Já a **Validação** ocorre em relação às **expectativas dos usuários**.

Existem dois tipos de verificação: estática e dinâmica! A verificação estática (também chamada Inspeção de Software) trata da análise de documento de requisitos, análise de diagramas de projetos, análise de código-fonte, entre outros. Ela ocorre sem a necessidade de executar o software em si e pode ocorrer de forma automatizada, antes mesmo da implementação do sistema. *Bacana?*

Já a Verificação Dinâmica (também chamada de Teste[1]) envolve executar o software/protótipo, isto é, a partir dos dados de entrada, examina-se o comportamento por meios das saídas esperadas, de modo que se verifique se o desempenho obtido está de acordo com o esperado. **Grosso modo, a verificação estática trata da documentação e a verificação dinâmica trata da execução em si do software.**

Calma, nem tudo são flores! Para fazer uma boa verificação estática, é necessário que as especificações dos artefatos documentais sejam precisas e confiáveis – ademais, não é fácil nem barato! Quanto à verificação dinâmica, nós falaremos mais adiante sobre cada estratégica, técnica

[1] Alguns autores tratam V&V como uma coisa só – integral, inteira.

e tipo de teste que podem ser feitos. **Apenas guardem que a verificação estática e a verificação dinâmica são complementares e, não, opostas!**

Cabe salientar também que a V&V não garante que o software seja completamente livre de defeitos ou que ele se comportará conforme especificado em todas as circunstâncias – é sempre possível que um teste ignorado possa descobrir mais problemas no sistema. Ele tem que ser suficientemente confiável para a utilização pretendida. *Espera aí... e quem diz o que é um software suficientemente confiável?*

Bem, isso depende da criticidade do sistema, das expectativas do utilizador, do ambiente de marketing, entre outros! Imaginemos um sistema de catálogo de livros de uma biblioteca e um sistema de controle de tráfego aéreo: *qual desses necessita de um grau de confiança mais alto?* Evidente que é o segundo! Imaginemos, agora, um sistema de caixa de padaria ou o sistema de estoque de um mercadinho. *Idaí, professor...*

Em geral, o utilizador pode ter baixas expectativas sobre o sistema e, assim, ter um **grau de confiança menor sem prejudicar seu funcionamento**. Nesses casos, é comum aceitar falhas de sistema quando os benefícios do uso ultrapassam as desvantagens. Por fim, algumas vezes um software precisa ser lançado no mercado rapidamente **como resposta à concorrência ou a um ambiente de marketing favorável**.

Por exemplo: quando uma empresa tem poucos concorrentes, ela pode liberar um programa antes que ele tenha sido inteiramente testado e depurado para poder se a primeira do mercado. *Entendido?* Pessoal, algumas pessoas acham que as inspeções de software não têm importância. Ora, têm sim! Elas ocorrem, inclusive, em todos os estágios do processo de desenvolvimento de software – **qualquer representação legível do software pode ser inspecionada**.

É evidente que não é possível usar técnicas estáticas para verificar requisitos não-funcionais (desempenho, confiabilidade, entre outros). **Outra confusão bastante frequente ocorre entre Teste e Depuração**! No entanto, essa é diferença é bastante simples: testes estabelecem a existência de defeitos e geralmente são feitos por uma equipe de testes; depuração localiza e conserta esses defeitos e geralmente é feita por uma equipe de desenvolvimento. *Fechou?*

(CEMIG – 2010 – Letra A) O teste é uma atividade de verificação e validação do software e consiste na análise dinâmica do mesmo, isto é, na execução do produto de software com o objetivo de verificar a presença de defeitos no produto e aumentar a confiança de que o mesmo está correto.

_____

Comentários: alguns autores tratam V&V como uma coisa só! Nós sabemos que, sendo rigorosos, teste é uma atividade de verificação, mas não está errado afirmar que teste é uma atividade de verificação e validação (Correto).

(ANATEL – 2007) Considere as informações abaixo em relação ao desenvolvimento de sistemas:

I. executar um software com o objetivo de revelar falhas, mas que não prova a exatidão do software.
II. correta construção do produto.
III. Construção do produto certo.

Correspondem corretamente a I, II e III, respectivamente,

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

_____

Comentários: (I) Trata-se do teste, visto que revela falhas e realmente não prova a exatidão do software; (II) Trata-se da Verificação – é semelhante a "Estamos construindo o produto corretamente?"; (III) Trata-se da Validação – é semelhante a "Estamos construindo o produto correto?" (Letra C).

(TJ/ES – 2011) Verificação e validação são atividades da análise de software, necessárias para se identificar o que o software precisa executar, seguida de uma avaliação do usuário quanto às atividades definidas.

_____

Comentários: ambas são realmente atividades de análise de software necessárias para se identificar o que o software precisa executar, seguida de uma avaliação do usuário quanto às atividades definidas. Em outras palavras, a verificação ocorre em relação à especificação de requisitos e a validação ocorre em relação às expectativas dos usuários respectivamente (Correto).

(TRT5 – 2008) A diferença entre verificação e validação reside no fato de que a primeira se refere ao conjunto de atividades que garante que o software realiza corretamente uma função específica, enquanto a segunda refere-se a um conjunto diferente de atividades que garante que o software que foi construído é rastreável às exigências do cliente.

_____

Comentários: a questão está perfeita! A verificação realmente garante que o software realiza corretamente alguma função e a validação realmente garante que o software que foi construído é rastreável às exigências do cliente, isto é, satisfazem as expectativas dos clientes (Correto).

(IPEA – 2008) A verificação assegura que o produto, como fornecido, irá atender o seu uso pretendido, ou seja, que se está construindo o produto certo. E a validação confirma que os produtos de trabalho refletem de forma apropriada os requisitos que foram especificados, ou seja, que se está construindo o produto corretamente.

_____

Comentários: a questão inverteu os conceitos de verificação e validação (Errado).

(AFR/SP – 2009) O processo de confirmação que um software vai ao encontro das especificações de software se trata de um conceito-chave de qualidade denominado:

a) Confiabilidade.
b) Validação.
c) Verificação.
d) Precisão.
e) Acurácia.

Comentários: trata-se da v**ER**ificação – **E**specificação de **R**equisitos (Letra C).

(PETROBRÁS – 2011) A verificação de software é um processo mais abrangente que o processo de validação de software.

PORQUE

O objetivo da validação é assegurar que o sistema atenda às expectativas do cliente, enquanto que a verificação envolve testes de correção do produto.

Analisando-se as afirmações acima, conclui-se que:

a) as duas afirmações são verdadeiras, e a segunda justifica a primeira.
b) as duas afirmações são verdadeiras, e a segunda não justifica a primeira.
c) a primeira afirmação é verdadeira, e a segunda é falsa.
d) a primeira afirmação é falsa, e a segunda é verdadeira.
e) as duas afirmações são falsas.

Comentários: validação é um processo mais abrangente que a Verificação. Por que? Porque um produto pode estar completamente aderente a uma especificação, mas não satisfazer às expectativas do usuário; no entanto, é mais difícil um produto satisfazer completamente às expectativas do usuário e não estar aderente a especificação. Já a segunda afirmação – para mim – está correta. No entanto, a banca a considerou errada. Eu não vejo erros nesse item – caso alguém encontre, favor informar (Letra E).

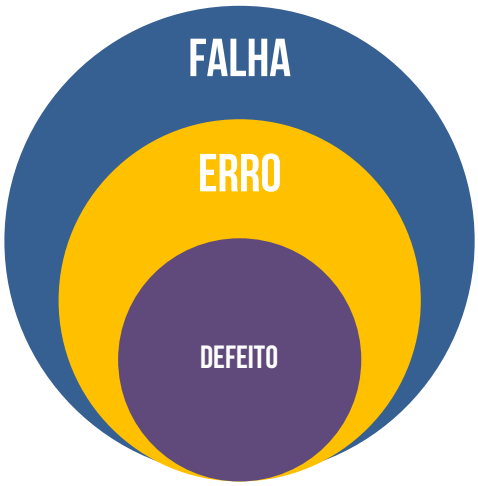
Defeito, Erro e Falha

INCIDÊNCIA EM PROVA: BAIXA

Vamos falar brevemente sobre esses conceitos? Para tal, vamos utilizar as definições de diversos autores e instituições[1]! *Professor, qual vai cair na prova?* **Galera, pode cair alguma delas ou uma totalmente diferente – infelizmente essa é uma aula em que os grandes autores não chegam a nenhum consenso, atrapalhando muito na hora da prova**. Desafortunadamente, não podemos fazer nada a respeito disso. *Certinho?* Vamos lá...

DEFINIÇÃO DE ACORDO COM A IEEE 610	
DEFEITO	Também chamado de Falta, trata-se do ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta. Em outras palavras, trata-se de um passo, processo ou definição de dados incorretos (Ex: instrução ou comando incorreto) - pode ocasionar a manifestação de erros em um produto.
ERRO	Também chamado de Engano, trata-se da ação humana que produz um resultado incorreto (Ex: lógica incorreta escrita pelo programador). Em outras palavras, é a manifestação concreta de um defeito em um artefato de software. É a diferença entre o valor obtido e o valor esperado, isto é, qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa.
FALHA	Trata-se do comportamento operacional do software diferente do esperado pelo usuário. Em outras palavras, trata-se da produção de uma saída incorreta em relação à especificação. Em geral, defeitos e erros são causas e falhas são consequências. Elas afetam diretamente o usuário final da aplicação e pode inviabilizar a utilização de um software.

Defeitos fazem parte do universo físico, isto é, da aplicação propriamente dita. Além disso, eles são causados por pessoas. Defeitos podem ocasionar a manifestação de erros, ou seja, a construção de um software diferente do especificado – que fazem parte do universo de informação. Por fim, erros podem gerar falhas como comportamentos inesperados de um software – que fazem parte do universo do usuário.



[1] Eventualmente, pode cair nomes em inglês: Erro/Error, Falha/Failure e Defeito/Defect.

Vamos ver um exemplo? Imaginem que um cabo de rede de uma impressora se desconectou (**aqui está o defeito!**), provocando um problema de comunicação entre estações de trabalho e servidor de rede (**aqui está o Erro!)** e causando, por fim, a não impressão de arquivos desejados pelo usuário (**aqui está a Falha!**). Percebam que defeitos são observados sob perspectiva interna, isto é, código incorreto, lógica inconsistente, funções ausentes, problemas de hardware, etc.

Em contrapartida, falhas são observadas sob uma perspectiva externa, isto é, sob o ponto de vista da percepção do usuário, como travamento do sistema, terminação anormal, tela azul, etc. **No meio, nós temos a perspectiva intermediária**. Aí, eu pergunto: *defeitos e erros sempre causarão falhas?* Não! Caso o usuário não tente imprimir nada enquanto o cabo estiver desconectado, nenhuma falha se manifestará!

Percebam, então, que defeitos causam erros que podem causar falhas – como mostra a imagem a seguir. Dessa forma, quando há uma diferença entre o resultado observado e o resultado esperado, temos um erro; quando há uma diferença entre o comportamento observado e o comportamento esperado, temos uma falha. **Galera, esse conteúdo infelizmente é repleto de contradições entre autores diferentes e desafortunadamente vocês terão que conviver com essas contradições :(**

Em uma outra abordagem, considera-se que um ser humano está sujeito a cometer um **erro (engano)**, que produz um **defeito**, no código, em um software ou sistema ou em um documento. Se um defeito no código for executado, o sistema **falhará** ao tentar fazer o que deveria (ou, em algumas vezes, o que não deveria), causando uma **falha**. Defeitos no software, sistemas ou documentos resultam em falhas, mas nem todos os defeitos causam falhas.

Galera, essa é a classificação mais comum em prova. No entanto, eventualmente também cai a classificação de acordo com Roger Pressman! Em seu livro, ele afirma que o objetivo do controle da qualidade de software e da gestão da qualidade em geral é, em sentido mais amplo, eliminar problemas de qualidade no software. **Tais problemas são conhecidos por diversos nomes — bugs, falhas, erros ou defeitos, apenas para citar alguns**.

Em seu livro, ele considera uma distinção clara entre erro e defeito – **o erro seria um problema de qualidade encontrado antes de o software ser liberado aos usuários finais; e o defeito seria um problema de qualidade encontrado apenas depois de o software ter sido liberado aos usuários finais**. Essa distinção é feita porque os erros e os defeitos podem acarretar impactos econômicos, comerciais, psicológicos e humanos muito diferentes.

De acordo com o autor, os engenheiros de software têm a missão de encontrar e corrigir o maior número possível de erros antes dos clientes e/ou usuários finais. Devem-se evitar defeitos — pois (de modo justificável) criam uma imagem negativa do pessoal de software. É importante notar, entretanto, que a distinção temporal entre erros e defeitos não é um pensamento dominante. Dito isso, vamos ver a tabela de definições de acordo com nosso querido autor:

DEFINIÇÃO DE ACORDO COM PRESSMAN	
DEFEITO	Problema de qualidade descoberto após o software ser lançado aos usuários finais ou após outra atividade de um processo de software.
ERRO	Problema de qualidade descoberto antes de o software ser lançado aos usuários finais ou após outra atividade de um processo de software.

(INMETRO – 2010 – Item D) Na terminologia de testes, uma falta ou defeito é a causa de um mau funcionamento de um software; uma falha é o resultado incorreto de uma falta ou defeito; um erro é a diferença entre um resultado computado e um resultado esperado. As falhas são descobertas por meio de testes, mas é a correção da falta ou do defeito que eliminará a falha.

_____

Comentários: a questão está perfeita e em conformidade com a IEEE 610. Além disso, é interessante observar que realmente as falhas são descobertas por meio de testes, mas que a correção do defeito que elimina a falha (Correto).

(TRE/BA – 2010) Segundo o IEEE, defeito é um ato inconsistente cometido por um indivíduo ao tentar entender determinada informação, resolver um problema ou utilizar um método ou uma ferramenta; erro é o comportamento operacional do software diferente do esperado pelo usuário, e que pode ter sido causado por diversas falhas; e falha é uma manifestação concreta de um defeito em um artefato de software, ou seja, é qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa.

_____

Comentários: a questão inverteu os conceitos de Erro e Falha. Erro é uma manifestação concreta de um defeito e falha é comportamento operacional do software diferente do esperado pelo usuário (Errado).

(TCE/RO – 2013) No teste de software, defeitos em um produto podem provocar falhas, gerando erros, que são comportamentos inesperados em um software.

_____

Comentários: bastava lembrar do mnemônico **DEF** – **Defeitos** provocam **Erros**, que causam **Falhas** (Errado).

(EBSERH – 2013) Segundo Pressman (2011), a definição de defeito de software é um problema de qualidade encontrado,

a) Somente após a liberação de uso do software para os usuários finais.
b) Antes de o software ser liberado aos usuários finais.
c) Na fase de revisão.
d) Na fase de levantamento de requisitos.
e) Na fase de prototipação.

_____

Comentários: de acordo com Roger Pressman, defeito é um problema de qualidade encontrado após o software ser lançado aos usuários finais (Letra A).

(TRE/BA – 2017) Considerando os conceitos da engenharia de software no escopo de teste de software, julgue os itens a seguir.

I. Denomina-se defeito a produção de uma saída incorreta com relação à especificação.

II. A ação humana que produz um resultado incorreto — como a ação incorreta de um programador — configura engano.

III. Define-se erro a diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do programa.

IV. Falha é uma instrução ou um comando incorreto.

Estão certos apenas os itens:

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

Comentários: (I) Errado, a questão trata de Erro; (II) Correto, configura Erro ou Engano; (III) Correto, Erro é – de fato - a diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do programa; (IV) Errado, a questão trata de Defeito (Letra C).

Métricas de Qualidade de Software

INCIDÊNCIA EM PROVA: BAIXA

A medição de software preocupa-se com a derivação de um valor numérico ou o perfil para um atributo de um componente de software, sistema ou processo. Comparando esses valores entre si e com os padrões que se aplicam a toda a organização, é possível tirar conclusões sobre a qualidade do software ou avaliar a eficácia dos métodos, ferramentas e processos de software. Ex: digamos que uma organização tem a intenção de introduzir uma nova ferramenta de teste de software.

Antes de introduzir a ferramenta, em um determinado momento você registra o número de defeitos de software descobertos. Essa é uma baseline de avaliação da eficácia da ferramenta. Depois de algum tempo usando a ferramenta, esse processo é repetido. Se mais defeitos forem encontrados durante o mesmo período, depois que a ferramenta foi introduzida, você pode decidir se ela fornece suporte útil para o processo de validação de software.

O objetivo a longo prazo de medição de software é usá-la no lugar de revisões para fazer julgamentos sobre a qualidade de software. Usando a medição de software, um sistema poderia, idealmente, ser avaliado usando uma variedade de métricas e, a partir dessa medição, deduzir um valor para a qualidade do sistema. Se o software atingir o limiar de qualidade requerido, então ele poderia ser aprovado sem revisão.

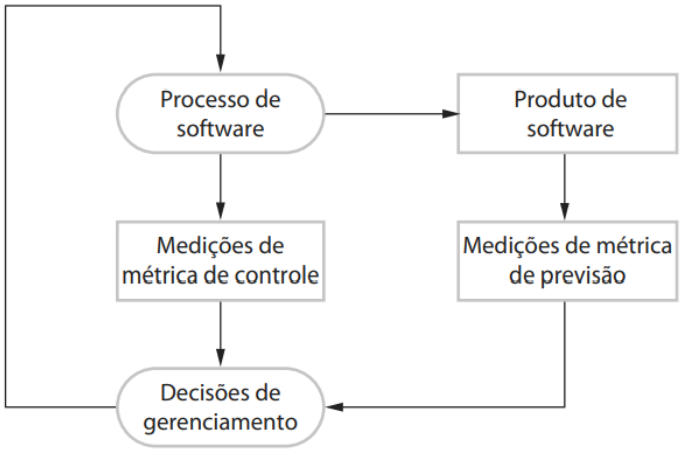
Quando apropriado, as ferramentas de medição também podem realçar áreas do software que poderiam ser melhoradas. No entanto, estamos ainda longe dessa situação ideal e não há sinal de que as avaliações automatizadas de qualidade venham a se tornar realidade em um futuro próximo. Uma métrica de software é uma característica de um sistema de software, documentação de sistema ou processo de desenvolvimento que pode ser objetivamente medido.

Exemplos de métricas incluem: o tamanho de um produto em linhas de código; o Índice Fog, que é uma medida da legibilidade de uma passagem de texto escrito; o número de defeitos relatados em um produto de software entregue, e o número de pessoas/dia requerido para desenvolver um componente de sistema. As métricas de software podem ser métricas de controle ou métricas de previsão.

Como os nomes sugerem, as primeiras suportam os processos de gerenciamento e as outras o ajudam a prever as características do software. As métricas de controle são geralmente associadas com os processos de software. Exemplos de métricas de controle ou de processos são o esforço médio e o tempo necessário para reparar os defeitos relatados. Métricas de previsão são associadas com o software em si e, por vezes, são conhecidas como Métricas de Produto.

São exemplos de métricas de previsão: a complexidade ciclomática de um módulo, o comprimento médio dos identificadores em um programa e o número de atributos e operações associadas com as classes de objeto em um projeto. As métricas de controle e de previsão podem influenciar a tomada de decisão de gerenciamento. Os gerentes usam métricas de processo para decidir se

devem ser feitas alterações no processo; as métricas de previsão são usadas para ajudar a estimar o esforço necessário para fazer as alterações no software.



Existem duas maneiras para o uso das medições de um software de sistema:

1. Para atribuir um valor aos atributos de qualidade de sistema: ao medir as características dos componentes de sistema, bem como sua complexidade ciclomática e, em seguida, agregar essas medições, você pode avaliar os atributos de qualidade do sistema, como a manutenibilidade.

2. Para identificar os componentes de sistema cuja qualidade não atingiu o padrão: As medições podem identificar componentes individuais com características que se desviem da norma. Ex: medir componentes para descobrir aqueles com a mais alta complexidade. Esses são mais passíveis de conter bugs porque a complexidade os torna mais difíceis de entender.

Infelizmente, é difícil fazer medições diretas de muitos dos atributos de qualidade de software. Os atributos de qualidade como manutenibilidade, compreensibilidade e usabilidade são atributos externos relacionados com os desenvolvedores e usuários que experimentam o software. Eles são afetados por fatores subjetivos, como a experiência e a educação do usuário e, portanto, não podem ser medidos objetivamente.

Para fazer um julgamento sobre esses atributos, você deve medir alguns atributos internos do software (como tamanho, complexidade, etc) e assumir que estão relacionados com as características de qualidade com as quais você se preocupa. Os atributos internos de software, como a complexidade ciclomática de um componente, são medidos com o uso de ferramentas de software que analisam o código-fonte do software.

As ferramentas *open source* disponíveis podem ser usadas para fazer essas medições. Embora a intuição sugira que poderia haver um relacionamento entre a complexidade de um componente de software e o número de falhas observadas em uso, é difícil demonstrar objetivamente que é este o caso. Para testar essa hipótese, você precisa de dados de falha de um grande número de componentes e de acesso ao código-fonte do componente para análise.

Poucas empresas fizeram um compromisso a longo prazo para a coleta de dados sobre seu software, portanto, dados de falha raramente são disponibilizados para análise.

As métricas e medições de software são as bases da engenharia de software empírica. Essa é uma área de pesquisa em que as experiências em sistemas de software e a coleta de dados sobre projetos reais foram usadas para formar e validar hipóteses sobre métodos e técnicas. Pesquisadores que trabalham nessa área argumentam que podemos confiar neles apenas se pudermos fornecer evidências concretas de que eles realmente oferecem os benefícios que seus inventores sugerem.

Infelizmente, mesmo quando é possível fazer medições objetivas e tirar conclusões a partir delas, isso pode não convencer os tomadores de decisões necessariamente. Em vez disso, as tomadas de decisões muitas vezes são influenciadas por fatores subjetivos, como a novidade ou a extensão em que as técnicas são de interesse para os profissionais. As métricas de qualidade de software podem ser divididas em métricas de qualidade de produto, processo e projeto.

As métricas de qualidade de processo podem ser utilizadas para melhorar o desenvolvimento e manutenção do software. Estas métricas são importantes em um processo de desenvolvimento, pois medem vários parâmetros nas várias fases do processo. Um exemplo clássico de métrica de qualidade de processo é a efetividade na remoção de defeitos ocorrida durante o desenvolvimento do sistema. Eis a fórmula:

$$\text{DRE} = \frac{\text{Defects removed during a development phase}}{\text{Defects latent in the product}} \times 100\%$$

O denominador é usualmente estimado como: defeitos removidos durante a fase de desenvolvimento + defeitos encontrados após o desenvolvimento. Um fato importante é que a utilização desta métrica em todas as fases do processo de desenvolvimento faz com que seja possível medir a efetividade de remoção de defeitos e a injeção de erros, e com isso poder rastrear as fases que mais injetam erros na aplicação.

Outras métricas de qualidade de processo são: índice de manutenção de erros, cobertura de testes, cobertura de testes modulares ou funcionais, entre outros. Já as métricas de qualidade de projeto descrevem as características do projeto em desenvolvimento como o número de desenvolvedores, utilização de padrões, custo, produtividade, entre outros. Elas são mais voltadas às características do projeto e sua execução.

As métricas de qualidade de projeto são utilizadas a fim de monitorar e avaliar os seguintes aspectos do projeto: progresso em termos de tamanho e complexidade; estabilidade em termos de taxa de mudança na implementação, tamanho ou complexidade; modularidade em termos da extensão da mudança; qualidade em termos do número e tipo de erros; maturidade em termos da frequência de erros; recursos em termos de recursos despendidos contra os planejados.

Por fim, as métricas de qualidade de produto são métricas de previsão usadas para medir atributos internos de um sistema de software (Ex: tamanho de sistema, medido em linhas de código; número de métodos associados a cada classe de objeto), mas as características de software que podem ser facilmente medidas, como tamanho e complexidade ciclomática, não têm uma relação clara e consistente com atributos de qualidade tais como capacidade de compreensão e manutenibilidade.

Os relacionamentos variam de acordo com os processos de desenvolvimento e tecnologia usada, bem como o tipo de sistema que está sendo desenvolvido. Essas métricas se dividem em:

1. **Métricas dinâmicas**, as quais são coletadas por meio de medições efetuadas de um programa em execução. Essas métricas podem ser coletadas durante o teste de sistema ou após o sistema estar em uso. Um exemplo pode ser o número de relatórios de bugs ou o tempo necessário para concluir uma computação.

2. **Métricas estáticas,** que são coletadas por meio de medições feitas de representações do sistema, como o projeto, o programa ou a documentação. Exemplos de métricas estáticas são o tamanho de código e o comprimento médio de identificadores utilizados no código-fonte.

Esses tipos de métrica estão relacionados com diferentes atributos de qualidade. Métricas dinâmicas ajudam a avaliar a eficiência e a confiabilidade de um programa. Métricas estáticas ajudam a avaliar a complexidade, a compreensibilidade e a manutenibilidade de um sistema ou componentes de um sistema de software. Geralmente, existe um relacionamento claro entre as métricas dinâmicas e as características de qualidade de software.

É bastante fácil medir o tempo de execução necessário para funções particulares e avaliar o tempo necessário para iniciar um sistema. Eles se relacionam diretamente com a eficiência do sistema. Da mesma forma, o número de falhas de sistema e o tipo de falha podem ser registrados e relacionados diretamente com a confiabilidade do software. Métricas estáticas têm um relacionamento indireto com atributos de qualidade.

Um grande número de métricas diferentes foi proposto e muitos experimentos tentaram derivar e validar os relacionamentos entre essas métricas e atributos, como a complexidade de sistema e manutenibilidade. Nenhum desses experimentos foi conclusivo, mas o tamanho de programa e a complexidade de controle parecem ser os mais confiáveis mecanismos de previsão de compreensibilidade, complexidade e manutenibilidade de sistema. Vejamos algumas métricas:

MÉTRICAS DE SOFTWARE	DESCRIÇÃO
FAN-IN/FAN-OUT	Fan-in é a medida do número de funções ou métodos que chamam outra função ou método (digamos X). Fan-out é o número de funções que são chamadas pela função de X. Um valor alto para fan-in significa que X está fortemente acoplado ao resto do projeto e alterações em X terão repercussões extensas. Um valor alto para fan-out sugere que a complexidade geral

	do X pode ser alta por causa da complexidade da lógica de controle necessário para coordenar os componentes chamados.
COMPRIMENTO DE CÓDIGO	Essa é uma medida do tamanho de um programa. Geralmente, quanto maior o tamanho do código de um componente, mais complexo e sujeito a erros o componente é. O comprimento de código tem mostrado ser uma das métricas mais confiáveis para prever a propensão a erros em componentes.
COMPLEXIDADE CICLOMÁTICA	Essa é uma medida da complexidade de controle de um programa. Essa complexidade de controle pode estar relacionada à compreensibilidade de programa.
COMPRIMENTO DE IDENTIFICADORES	Essa é uma medida do comprimento médio dos identificadores (nomes de variáveis, classes, métodos etc.) em um programa. Quanto mais longos os identificadores, mais provável que sejam significativos e, portanto, mais compreensível o programa.
PROFUNDIDADE DE ANINHAMENTO CONDICIONAL	Essa é uma medida da profundidade de aninhamento de declarações IF em um programa. Declarações IF profundamente aninhadas são difíceis de entender e potencialmente sujeitas a erros.
ÍNDICE FOG	Essa é uma medida do comprimento médio de palavras e sentenças em documentos. Quanto maior o valor de um índice Fog de um documento, mais difícil a sua compreensão.

Outros autores afirmar que a qualidade de software consiste em dois níveis: qualidade intrínseca do produto e satisfação do cliente. Cobrindo ambos os níveis, temos: Tempo Médio entre Falhas e Densidade do Defeito. A qualidade intrínseca do produto é medida pelo número de bugs ou tempo que o software pode rodar sem falhar. O Mean Time To Failure (MTTF) é a métrica de robustez mais utilizada em sistemas críticos, como sistemas de controle de tráfego aéreo.

Querem um exemplo? O Governo dos EUA exige que seu sistema de tráfego aéreo não fique indisponível por mais de três segundos por ano! *Sinistro, não é?!* Já a Densidade de Defeito, em contraste, é utilizada em muitos sistemas comerciais. Essas métricas são parecidas, mas a primeira mede o tempo entre falhas e a segunda mede os defeitos relativamente ao tamanho do software (linhas de código, pontos de função, etc) e o tempo.

Existe diferença entre falha e defeito? Sim, mas nesse contexto são iguais. A Densidade de Defeitos mede a quantidade de defeitos durante um período de tempo pelo tamanho do software.

$$\text{Defect Density} = \frac{\text{Number of Defects}}{\text{Size}}$$

O número de defeitos pode ser relativo a um determinado período (mês, trimestre, ano, etc), ou a uma determinada fase do ciclo de vida do software, ou ao ciclo de vida de software como um todo. Sua utilização permite a identificação de componentes de alto risco e permite a comparação de

produtos pela medição da qualidade de cada um. Existe também uma métrica similar ao MTTF! *Qual é, professor?* É a Mean Time Between Failure (MTBF).

Trata-se do tempo médio entre falhas, isto é, tempo entre duas falhas sucessivas no sistema, expressa na maioria das vezes em horas – trata-se de uma métrica chave para sistemas que podem ser reparados ou restaurados. Assim, é possível prever quando será a próxima falha do sistema – é mais utilizado com hardware, mas pode ser também utilizado com software. Bem, pessoal... é isso que tínhamos para falar sobre métricas de qualidade.

Métricas de Qualidade de Código

INCIDÊNCIA EM PROVA: BAIXA

A crescente adoção de programas de código aberto e de métodos ágeis pela indústria de software promove o código-fonte a um dos artefatos mais importante para se medir a qualidade de software. **Assim, as métricas de qualidade de código-fonte são mecanismos fundamentais para avaliação desses sistemas**. Sua utilização como critério para a avaliação da qualidade é motivada por estudos que indicam ser viável analisar algumas características para a aceitação de um software.

Que características, professor? São critérios como: flexibilidade, manutenibilidade e complexidade a partir do código-fonte. Você pode se perguntar: *o que tem um código-fonte de qualidade?* Ora, um código-fonte será muito mais lido do que escrito no decorrer de seu ciclo de vida (manutenção, reúso, etc). **Dessa forma, nós podemos afirmar que um código-fonte de qualidade é legibilidade, testabilidade, flexibilidade, compatibilidade e economicidade**.

CRITÉRIOS DE QUALIDADE DE CÓDIGO	DESCRIÇÃO
LEGALIDADE	O código (comentário, não) deve claramente declarar sua intenção. Se o leitor não consegue ver sentido no código, todos os outros esforços para melhorar a qualidade do software estão fadados ao fracasso.
TESTABILIDADE	O código deve ser organizado de uma forma que facilite o teste de unidade. Isso apoia todos os esforços subsequentes (refatoração, correção de defeitos, revisão devido a alteração de especificações, etc).
FLEXIBILIDADE	Dependências em relação a outros códigos devem ser minimizadas. Construir implementações hard-coded (isto é, fixas) sobre tamanhos e estruturas de dados, classes concretas, etc tornam o código difícil de reusar e adaptar.
COMPATIBILIDADE	O código deve cumprir com seus requisitos, funcionais ou não. Notem que uma discussão sobre se os requisitos implementados são os requisitos corretos não cabe aqui.
ECONOMICIDADE	O código deve fazer uso razoável dos recursos do sistema: memória, processamento, entre outros. Devemos pensar sobre o retorno sobre investimento e requer uma reflexão sobre todos os recursos investidos.

NBR ISO/IEC 9126

Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

Esta Norma versa sobre as **características que definem um produto de software de qualidade**. Após diversas revisões, Norma foi dividida em quatro partes:

1. ISO/IEC 9126-1: Modelo de Qualidade;
2. ISO/IEC 9126-2: Métricas Externas;
3. ISO/IEC 9126-3: Métricas Internas;
4. ISO/IEC 9126-4: Métricas de Qualidade em Uso.

Ela permite que a qualidade do produto de software seja especificada e avaliada em diferentes perspectivas pelos envolvidos com aquisição, requisitos, desenvolvimento, uso, avaliação, apoio, manutenção, garantia de qualidade e auditoria de software. **Ademais, descreve um modelo de qualidade do produto de software, composto de duas partes: qualidade interna e externa e qualidade em uso**. *E qual a diferença, professor?*

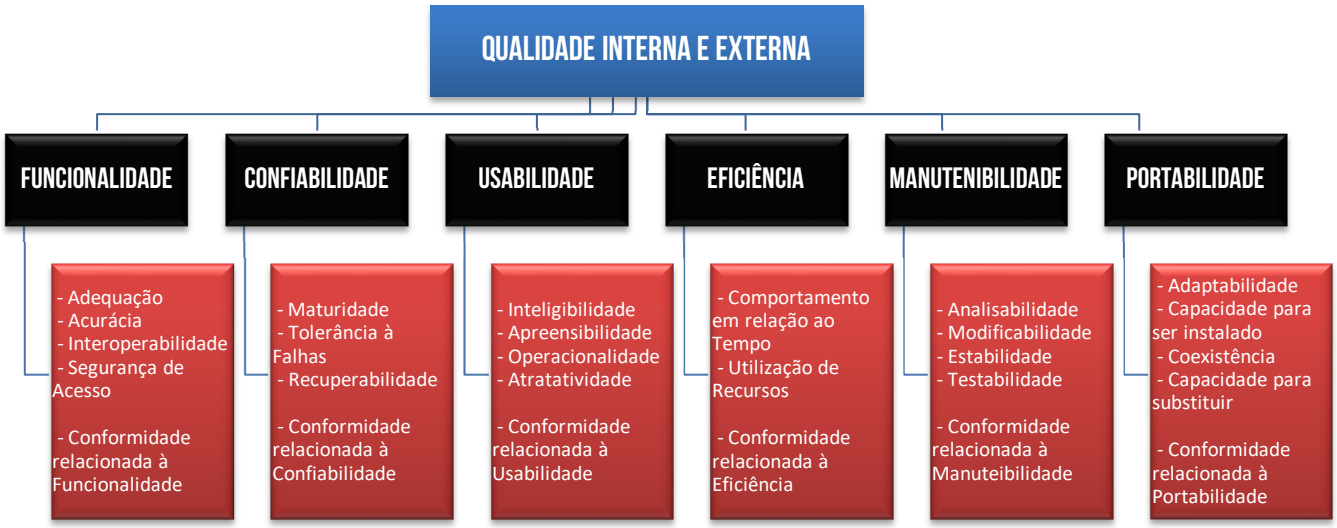
A Qualidade Interna é a totalidade das características do produto de software do ponto de vista interno. **A qualidade interna é medida e avaliada com relação aos requisitos de qualidade interna.** Detalhes da qualidade do produto de software podem ser melhorados durante a implementação do código, revisão e teste, mas a natureza fundamental da qualidade do produto representada pela qualidade interna mantém-se inalterada, a menos que seja reprojetada.

E a Qualidade Externa, professor? **Bem, a Qualidade Externa é a totalidade das características do produto de software do ponto de vista externo**. *Como assim? Não entendi!* É a qualidade quando o software é executado, o qual é tipicamente medido e avaliado enquanto está sendo testado num ambiente simulado, com dados simulados e usando métricas externas. Durante os testes, convém que a maioria dos defeitos seja descoberta e eliminada.

Por outro lado, alguns defeitos podem permanecer após o teste. **Como é difícil corrigir a arquitetura do software ou outro aspecto básico do projeto do software, a base do projeto usualmente permanece inalterada ao longo do teste.** Por fim, a Qualidade em Uso é a visão da qualidade do produto de software do ponto de vista do usuário, quando este produto é usado em um ambiente e um contexto de uso especificados.

Ela mede o quanto usuários podem atingir seus objetivos num determinado ambiente e não as propriedades do software em si. *Entendido, pessoal?* **Então, nós vimos a qualidade interna, a qualidade externa e a qualidade em uso.** Existe um modelo para a qualidade interna e externa e outro para a qualidade em uso, que é apresentado na imagem abaixo composta de quatro características.

Vamos falar sobre a Qualidade Interna e Externa – esse é de longe o modelo que mais cai em prova! *Professor, precisa decorar?* Difícil dizer isso, se você tiver tempo sobrando, eu recomendo. **Bem, ela categoriza os atributos de qualidade de software em seis características** que são, por sua vez, subdivididas em subcaracterísticas (que podem ser medidas por meio de métricas externas e internas)[1].



Define-se, a seguir, cada característica e subcaracterística do software que influencia a característica de qualidade. **A capacidade do software é determinada por um conjunto de atributos internos que podem ser medidos para cada característica e subcaracterística**. As características e subcaracterísticas podem ser medidas externamente pelo grau da capacidade do sistema contendo o software.

FUNCIONALIDADE	Capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas.

[1] MNEMÔNICO SUGERIDO: **EFI**GÊNIO **FUMA POUCO**! EM ORDEM: **EFICIÊNCIA**, **FUNCIONALIDADE**, **MANUTENIBILIDADE**, **PORTABILIDADE**, **USABILIDADE E CONFIABILIDADE**.

SUBCARACTERÍSTICA	DESCRIÇÃO
ADEQUAÇÃO	Capacidade do produto de software de prover um conjunto apropriado de funções para tarefas e objetivos do usuário especificados.
ACURÁCIA	Capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados.
INTEROPERABILIDADE	Capacidade do produto de software de interagir com um ou mais sistemas especificados.
SEGURANÇA DE ACESSO	Capacidade do produto de software de proteger informações e dados, tal que pessoas ou sistemas não autorizados não possam lê-los nem os modificar e que não seja negado o acesso às pessoas ou sistemas autorizados.
CONFORMIDADE RELACIONADA À FUNCIONALIDADE	Capacidade do produto de software de estar de acordo com normas, convenções ou regulamentações previstas em leis e prescrições similares relacionadas à funcionalidade.
CONFIABILIDADE	Capacidade do produto de software de manter um nível de desempenho especificado, quando usado em condições especificadas.

SUBCARACTERÍSTICA	DESCRIÇÃO
MATURIDADE	Capacidade do produto de software de evitar falhas decorrentes de defeitos no software.
TOLERÂNCIA A FALHAS	Capacidade do produto de software de manter um nível de desempenho especificado em casos de defeitos no software ou de violação de sua interface especificada.
RECUPERABILIDADE	Capacidade do produto de software de restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha.
CONFORMIDADE RELACIONADA À CONFIABILIDADE	Capacidade do produto de software de estar de acordo com normas, convenções ou regulamentações relacionadas à confiabilidade.
USABILIDADE	Capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas.
USABILIDADE	Capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas.

SUBCARACTERÍSTICA	DESCRIÇÃO

INTELIGIBILIDADE	Capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas.
APREENSIBILIDADE	Capacidade do produto de software de possibilitar ao usuário aprender sua aplicação.
OPERACIONALIDADE	Capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo.
ATRATIVIDADE	Capacidade do produto de software de ser atraente ao usuário.
CONFORMIDADE RELACIONADA À USABILIDADE	Capacidade do produto de software de estar de acordo com normas, convenções, guias de estilo ou regulamentações relacionadas à usabilidade.

EFICIÊNCIA	Capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas.

SUBCARACTERÍSTICA	DESCRIÇÃO
COMPORTAMENTO EM RELAÇÃO AO TEMPO	Capacidade do produto de software de fornecer tempos de resposta e de processamento, além de taxas de transferência, apropriados, quando o software executa suas funções, sob condições estabelecidas.
UTILIZAÇÃO DE RECURSOS	Capacidade do produto de software de usar tipos e quantidades apropriados de recursos, quando o software executa suas funções sob condições estabelecidas.
CONFORMIDADE RELACIONADA À EFICIÊNCIA	Capacidade do produto de software de estar de acordo com normas e convenções relacionadas à eficiência.

MANUTENIBILIDADE	Capacidade do produto de software de ser modificado. As modificações podem incluir correções, melhorias ou adaptações devido a mudanças no ambiente e nos seus requisitos ou especificações funcionais.

SUBCARACTERÍSTICA	DESCRIÇÃO
ANALISABILIDADE	Capacidade do produto de software de permitir o diagnóstico de deficiências ou causas de falhas no software, ou a identificação de partes a serem modificadas.
MODIFICABILIDADE	Capacidade do produto de software de permitir que uma modificação especificada seja implementada.
ESTABILIDADE	Capacidade do produto de software de evitar efeitos inesperados decorrentes de modificações no software.
TESTABILIDADE	Capacidade do produto de software de permitir que o software, quando modificado, seja validado.

CONFORMIDADE RELACIONADA À MANUTENIBILIDADE	Capacidade do produto de software de estar de acordo com normas ou convenções relacionadas à manutenibilidade.

PORTABILIDADE	Capacidade do produto de software de ser transferido de um ambiente para outro.

SUBCARACTERÍSTICA	DESCRIÇÃO
ADAPTABILIDADE	Capacidade do produto de software de ser adaptado para diferentes ambientes especificados, sem necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software.
CAPACIDADE PARA SER INSTALADO	Capacidade do produto de software para ser instalado em um ambiente especificado.
COEXISTÊNCIA	Capacidade do produto de software de coexistir com outros produtos de software independentes, em um ambiente comum, compartilhando recursos comuns.
CAPACIDADE PARA SUBSTITUIR	Capacidade do produto de software de ser usado em substituição a outro produto de software especificado, com o mesmo propósito e no mesmo ambiente.
CONFORMIDADE RELACIONADA À PORTABILIDADE	Capacidade do produto de software de estar de acordo com normas ou convenções relacionadas à portabilidade.

Vamos detalhar agora a Qualidade em Uso e suas características:

CARACTERÍSTICA	DESCRIÇÃO
EFICÁCIA	Capacidade do produto de software de permitir que usuários atinjam metas especificadas com acurácia e completude, em um contexto de uso especificado.
PRODUTIVIDADE	Capacidade do produto de software de permitir que seus usuários empreguem quantidade apropriada de recursos em relação à eficácia obtida, em um contexto de uso especificado.
SEGURANÇA	Capacidade do produto de software de apresentar níveis aceitáveis de riscos de danos a pessoas, negócios, software, propriedades ou ao ambiente, em um contexto de uso especificado.
SATISFAÇÃO	Capacidade do produto de software de satisfazer usuários, em um contexto de uso especificado.

Falando brevemente agora sobre a NBR ISO/IEC 9126-2: Métricas Externas. Ela se apoia na definição de atributos externos de qualidade correlacionados com uma determinada característica e define indicadores e métricas externas para avaliar um produto de software. As Métricas Externas

referem-se a medições indiretas de um produto de software a partir do comportamento do sistema computacional ou do seu efeito no ambiente, quando da execução de seus programas.

Elas devem ser usadas para avaliar o comportamento do software em situações específicas para predizer a qualidade real durante o uso. Também será usada para avaliar e indicar se o produto satisfaz as verdadeiras necessidades durante a operação real pelo usuário. Se escolhermos uma característica (Ex: funcionalidade) e uma subcaracterística (Ex: adequação). **Uma métrica externa poderia ser a quantidade de funções atendidas desejáveis e obrigatórias.**

Já a NBR ISO/IEC 9126-3 (Parte 3): Métricas Internas define indicadores e métricas internas para avaliar um produto de software. Elas referem-se a medições de um produto de software a partir de suas características internas, sem a necessidade de execução nos programas (Ex: linhas de código, número de erros, etc). As métricas internas fornecem aos usuários a possibilidade de medir a qualidade dos artefatos intermediários e prever a qualidade do produto final.

Isto permite que o usuário identifique problemas de qualidade e inicie a ação corretiva assim que possível no ciclo de vida do desenvolvimento. **Por fim, a NBR ISO/IEC 9126-4 (Parte 4) valida a qualidade do produto em cenários e tarefas comuns ao usuário**. Os atributos da qualidade em uso são categorizados pelas características: eficácia, produtividade, segurança e satisfação. Usuários podem desenvolver e aplicar métricas para seus domínios particulares de aplicação.

Devemos ter em mente que a qualidade interna e externa são aplicáveis a um produto de software; **já a qualidade em uso é aplicável ao efeito do produto de software em um cenário específico**. Ademais, as métricas internas podem ser aplicadas a um produto de software não executável. As métricas externas podem ser usadas para medir a qualidade do produto de software através da medição de seu comportamento em um sistema do qual ele faça parte.

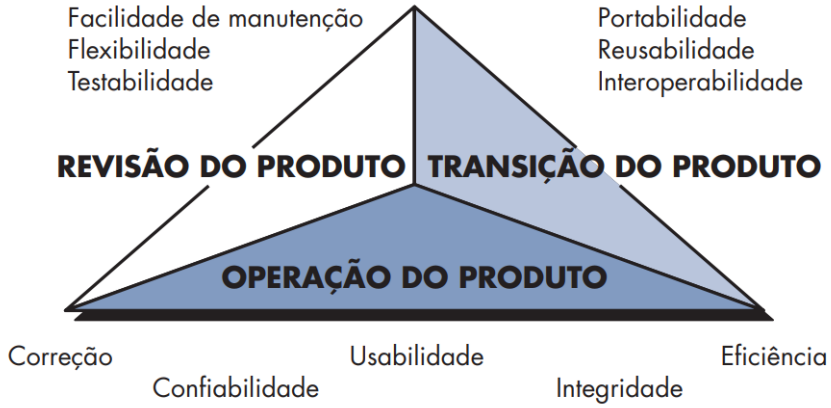
E as Métricas de Qualidade em Uso medem o quanto o produto agrega às necessidades de usuários específicos. Temos, então, três categorias de qualidade e cada uma tem sua especificidade. odemos Galera, essa norma trata de **um conjunto de atributos que têm impacto na capacidade do software de manter o seu nível de desempenho dentro de condições estabelecidas por um dado período de tempo** – ela é responsável pela confiabilidade do software!

RESUMO

QUALIDADE DE SOFTWARE

Característica de ter demonstrado a realização da criação de um produto que atende ou excede os requisitos acordados, conforme avaliado por medidas e critérios acordados, e que é criado em um processo acordado.

CATEGORIAS DE FATORES DE QUALIDADE DE SOFTWARE (POR MCCALL, RICHARDS E WALTERS)



FATORES	DESCRIÇÃO
CORREÇÃO	O quanto um programa satisfaz a sua especificação e atende aos objetivos da missão do cliente.
CONFIABILIDADE	O quanto se pode esperar que um programa realize a função pretendida com a precisão exigida.
EFICIÊNCIA	A quantidade de recursos computacionais e código exigidos por um programa para desempenhar sua função.
INTEGRIDADE	O quanto o acesso ao software ou dados por pessoas não autorizadas pode ser controlado.
USABILIDADE	Esforço necessário para aprender, operar, preparar a entrada de dados e interpretar a saída de um programa.
FACILIDADE DE MANUTENÇÃO	Esforço necessário para localizar e corrigir um erro em um programa.
FLEXIBILIDADE	Esforço necessário para modificar um programa em operação.
TESTABILIDADE	Esforço necessário para testar um programa de modo a garantir que ele desempenhe a função destinada.
PORTABILIDADE	Esforço necessário para transferir o programa de um ambiente de hardware e/ou software para outro.
REUSABILIDADE	O quanto um programa [ou partes de um programa] pode ser reutilizado em outras aplicações.
INTEROPERABILIDADE	Esforço necessário para integrar um sistema a outro.

FATORES (ISO 9126)	DESCRIÇÃO
FUNCIONALIDADE	Trata-se do grau com que o software satisfaz às necessidades declaradas conforme indicado pelos seguintes subatributos: adequabilidade, exatidão, interoperabilidade, conformidade e segurança.
CONFIABILIDADE	Trata-se da quantidade de tempo que o software fica disponível para uso conforme indicado pelos seguintes subatributos: maturidade, tolerância a falhas, facilidade de recuperação.
USABILIDADE	Trata-se do grau de facilidade de utilização do software conforme indicado pelos seguintes subatributos: facilidade de compreensão, facilidade de aprendizagem, operabilidade.
EFICIÊNCIA	Trata-se do grau de otimização do uso, pelo software, dos recursos do sistema conforme indicado pelos seguintes subatributos: comportamento em relação ao tempo, comportamento em relação aos recursos.
MANUTENIBILIDADE	Trata-se da facilidade com a qual uma correção pode ser realizada no software conforme indicado pelos seguintes subatributos: facilidade de análise, facilidade de realização de mudanças, estabilidade, testabilidade.
PORTABILIDADE	Trata-se da facilidade com a qual um software pode ser transposto de um ambiente a outro conforme indicado pelos seguintes subatributos: adaptabilidade, facilidade de instalação, conformidade, facilidade de substituição.

MNEMÔNICO DOS FATORES DE QUALIDADE SEGUNDO A ISO 9126

CAIXA

C	E	F	M	P	U
CONFIABILIDADE	EFICIÊNCIA	FUNCIONALIDADE	MANUTENIBILIDADE	PORTABILIDADE	USABILIDADE

GARANTIA DE QUALIDADE	CONTROLE DE QUALIDADE
Garantia da qualidade garante que o processo é definido e apropriado.	As atividades de controle da qualidade focam na descoberta de defeitos específicos.
Metodologia e padrões de desenvolvimento são exemplos de garantia da qualidade.	Um exemplo de controle da qualidade poderia ser: *"Os requisitos definidos são os requisitos certos?"*
Garantia da qualidade é orientada a processo.	Controle da qualidade é orientado a produto.
Garantia da qualidade é orientada a prevenção.	Controle da qualidade é orientado a detecção.
Foco em monitoração e melhoria de processo.	Inspeções e garantia de que o produto de trabalho atenda aos requisitos especificados.
As atividades são focadas no início das fases no ciclo de vida de desenvolvimento de software.	As atividades são focadas no final das fases no ciclo de vida de desenvolvimento de software.
Garantia da qualidade garante que você está fazendo certo as coisas e da maneira correta.	Controle da qualidade garante que os resultados do seu trabalho são os esperados conforme requisitos.

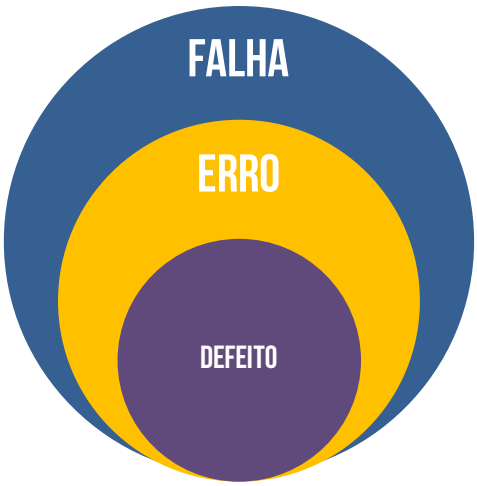
VERIFICAÇÃO (DEPENDE DA ESPECIFICAÇÃO)

ESTAMOS CONSTRUINDO O PRODUTO CORRETAMENTE?
O PRODUTO ESTÁ DE ACORDO COM AS ESPECIFICAÇÕES DO SISTEMA?

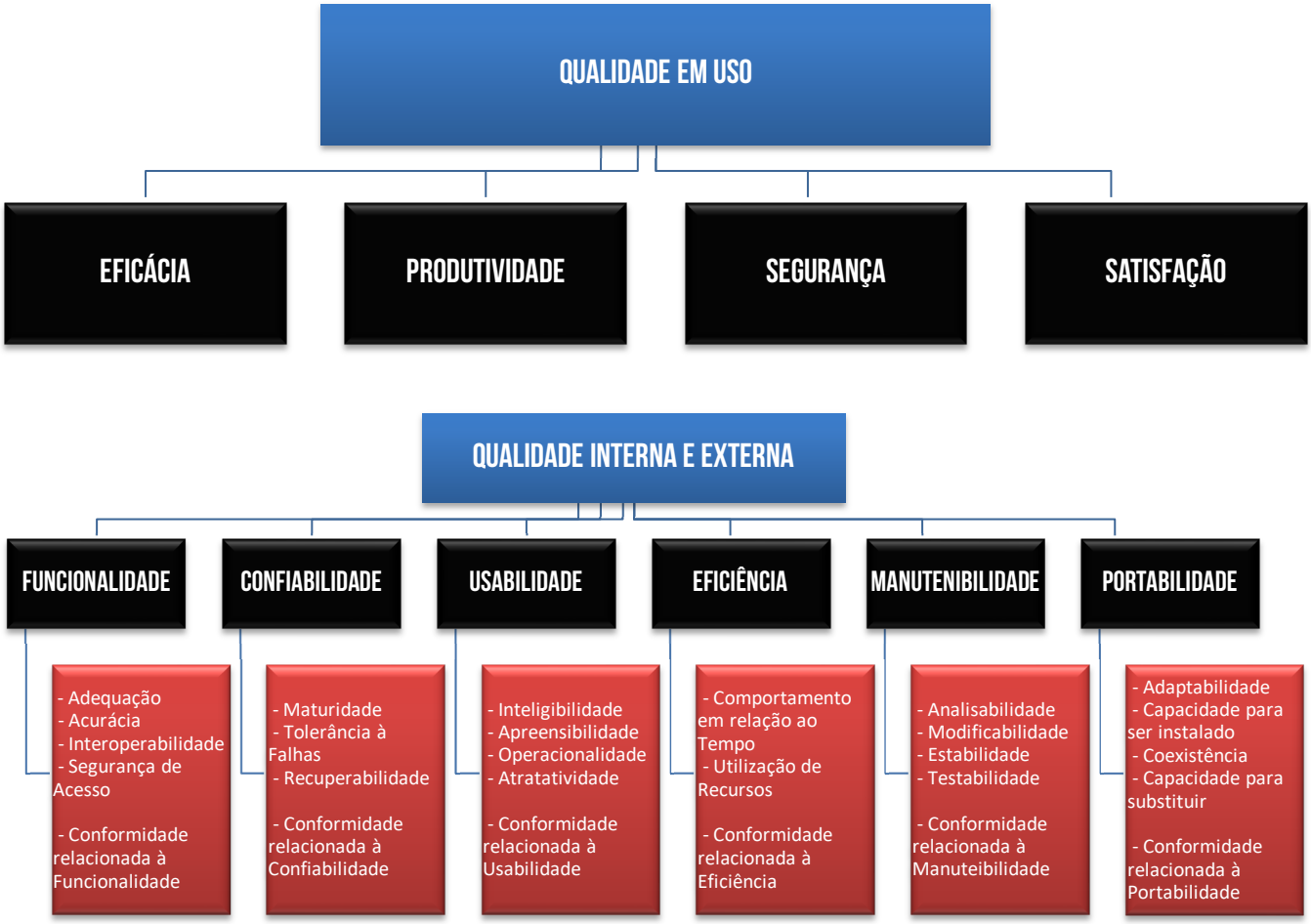
VALIDAÇÃO (DEPENDE DO USUÁRIO)

ESTAMOS CONSTRUINDO O PRODUTO CORRETO?
O PRODUTO ESTÁ DE ACORDO COM AS NECESSIDADES DO USUÁRIO?

DEFINIÇÃO DE ACORDO COM A IEEE 610	
DEFEITO	Também chamado de Falta, trata-se do ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta. Em outras palavras, trata-se de um passo, processo ou definição de dados incorretos (Ex: instrução ou comando incorreto) - pode ocasionar a manifestação de erros em um produto.
ERRO	Também chamado de Engano, trata-se da ação humana que produz um resultado incorreto (Ex: lógica incorreta escrita pelo programador). Em outras palavras, é a manifestação concreta de um defeito em um artefato de software. É a diferença entre o valor obtido e o valor esperado, isto é, qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa.
FALHA	Trata-se do comportamento operacional do software diferente do esperado pelo usuário. Em outras palavras, trata-se da produção de uma saída incorreta em relação à especificação. Em geral, defeitos e erros são causas e falhas são consequências. Elas afetam diretamente o usuário final da aplicação e pode inviabilizar a utilização de um software.



DEFINIÇÃO DE ACORDO COM PRESSMAN	
DEFEITO	Problema de qualidade descoberto após o software ser lançado aos usuários finais ou após outra atividade de um processo de software.
ERRO	Problema de qualidade descoberto antes de o software ser lançado aos usuários finais ou após outra atividade de um processo de software.



FUNCIONALIDADE	Capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas.

SUBCARACTERÍSTICA	DESCRIÇÃO
ADEQUAÇÃO	Capacidade do produto de software de prover um conjunto apropriado de funções para tarefas e objetivos do usuário especificados.
ACURÁCIA	Capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados.
INTEROPERABILIDADE	Capacidade do produto de software de interagir com um ou mais sistemas especificados.

SEGURANÇA DE ACESSO	Capacidade do produto de software de proteger informações e dados, tal que pessoas ou sistemas não autorizados não possam lê-los nem os modificar e que não seja negado o acesso às pessoas ou sistemas autorizados.
CONFORMIDADE RELACIONADA À FUNCIONALIDADE	Capacidade do produto de software de estar de acordo com normas, convenções ou regulamentações previstas em leis e prescrições similares relacionadas à funcionalidade.

CONFIABILIDADE	Capacidade do produto de software de manter um nível de desempenho especificado, quando usado em condições especificadas.

SUBCARACTERÍSTICA	DESCRIÇÃO
MATURIDADE	Capacidade do produto de software de evitar falhas decorrentes de defeitos no software.
TOLERÂNCIA A FALHAS	Capacidade do produto de software de manter um nível de desempenho especificado em casos de defeitos no software ou de violação de sua interface especificada.
RECUPERABILIDADE	Capacidade do produto de software de restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha.
CONFORMIDADE RELACIONADA À CONFIABILIDADE	Capacidade do produto de software de estar de acordo com normas, convenções ou regulamentações relacionadas à confiabilidade.
USABILIDADE	Capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas.

USABILIDADE	Capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas.

SUBCARACTERÍSTICA	DESCRIÇÃO
INTELIGIBILIDADE	Capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas.
APREENSIBILIDADE	Capacidade do produto de software de possibilitar ao usuário aprender sua aplicação.
OPERACIONALIDADE	Capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo.
ATRATIVIDADE	Capacidade do produto de software de ser atraente ao usuário.

CONFORMIDADE RELACIONADA À USABILIDADE	Capacidade do produto de software de estar de acordo com normas, convenções, guias de estilo ou regulamentações relacionadas à usabilidade.

EFICIÊNCIA	Capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas.

SUBCARACTERÍSTICA	DESCRIÇÃO
COMPORTAMENTO EM RELAÇÃO AO TEMPO	Capacidade do produto de software de fornecer tempos de resposta e de processamento, além de taxas de transferência, apropriados, quando o software executa suas funções, sob condições estabelecidas.
UTILIZAÇÃO DE RECURSOS	Capacidade do produto de software de usar tipos e quantidades apropriados de recursos, quando o software executa suas funções sob condições estabelecidas.
CONFORMIDADE RELACIONADA À EFICIÊNCIA	Capacidade do produto de software de estar de acordo com normas e convenções relacionadas à eficiência.

MANUTENIBILIDADE	Capacidade do produto de software de ser modificado. As modificações podem incluir correções, melhorias ou adaptações devido a mudanças no ambiente e nos seus requisitos ou especificações funcionais.

SUBCARACTERÍSTICA	DESCRIÇÃO
ANALISABILIDADE	Capacidade do produto de software de permitir o diagnóstico de deficiências ou causas de falhas no software, ou a identificação de partes a serem modificadas.
MODIFICABILIDADE	Capacidade do produto de software de permitir que uma modificação especificada seja implementada.
ESTABILIDADE	Capacidade do produto de software de evitar efeitos inesperados decorrentes de modificações no software.
TESTABILIDADE	Capacidade do produto de software de permitir que o software, quando modificado, seja validado.
CONFORMIDADE RELACIONADA À MANUTENIBILIDADE	Capacidade do produto de software de estar de acordo com normas ou convenções relacionadas à manutenibilidade.

PORTABILIDADE	Capacidade do produto de software de ser transferido de um ambiente para outro.

SUBCARACTERÍSTICA	DESCRIÇÃO

ADAPTABILIDADE	Capacidade do produto de software de ser adaptado para diferentes ambientes especificados, sem necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software.
CAPACIDADE PARA SER INSTALADO	Capacidade do produto de software para ser instalado em um ambiente especificado.
COEXISTÊNCIA	Capacidade do produto de software de coexistir com outros produtos de software independentes, em um ambiente comum, compartilhando recursos comuns.
CAPACIDADE PARA SUBSTITUIR	Capacidade do produto de software de ser usado em substituição a outro produto de software especificado, com o mesmo propósito e no mesmo ambiente.
CONFORMIDADE RELACIONADA À PORTABILIDADE	Capacidade do produto de software de estar de acordo com normas ou convenções relacionadas à portabilidade.

QUALIDADE EM USO	DESCRIÇÃO
EFICÁCIA	Capacidade do produto de software de permitir que usuários atinjam metas especificadas com acurácia e completude, em um contexto de uso especificado.
PRODUTIVIDADE	Capacidade do produto de software de permitir que seus usuários empreguem quantidade apropriada de recursos em relação à eficácia obtida, em um contexto de uso especificado.
SEGURANÇA	Capacidade do produto de software de apresentar níveis aceitáveis de riscos de danos a pessoas, negócios, software, propriedades ou ao ambiente, em um contexto de uso especificado.
SATISFAÇÃO	Capacidade do produto de software de satisfazer usuários, em um contexto de uso especificado.

QUESTÕES COMENTADAS – CESPE

1. **(CESPE / BANRISUL – 2022)** Os principais recursos de um sistema de controle de versão incluem um repositório de dados que armazena todos os objetos de configuração relevantes e um recurso de gestão de versão que armazena todas as versões de um objeto de configuração.

Comentários:

Perfeito! O repositório de dados mantém todos os objetos de configuração e as suas respectivas versões, que podem ser facilmente acessadas a qualquer momento. O recurso de gestão de versão armazena todas as versões de um objeto de configuração, o que permite que os usuários acessem versões anteriores ou façam comparações entre versões. O controle de versão também permite que os usuários possam reverter para versões anteriores, se necessário.

Gabarito: Correto

---

2. **(CESPE / BANRISUL – 2022)** De acordo com a SQA (Software Quality Assurance), correção, completude e consistência do modelo de requisitos são características da qualidade do código que influenciam a qualidade de todos os produtos.

Comentários:

Correção, completude e consistência não são características da qualidade do código e, sim, da qualidade dos requisitos. De acordo com Pressman, temos que:

Qualidade dos Requisitos: a correção, a completude e a consistência do modelo de requisitos terão forte influência sobre a qualidade de todos os produtos seguintes.

Gabarito: Errado

---

3. **(CESPE / BANRISUL – 2022)** Define-se confiabilidade de software como a probabilidade de operação, sem falhas, de um programa de computador em dado ambiente por determinado tempo.

Comentários:

Perfeito! A confiabilidade de software é definida como a probabilidade do sistema funcionar sem ocorrência de falhas num período e ambiente especificados.

Gabarito: Correto

---

4. **(CESPE / BANRISUL – 2022)** A revisão por pares é uma forma de análise da causa-raiz, na qual a equipe define uma meta ou efeito arquitetural e, então, enuncia as ações relacionadas para o alcance da meta.

Comentários:

A Revisão Por Pares é uma avaliação detalhada, realizada por um grupo de pessoas com diferentes habilidades e conhecimentos, de um produto, processo ou serviço antes de ser liberado para uso. O objetivo é identificar erros, ambiguidades, inconsistências e oportunidades para melhorias e, não, definir uma meta ou efeito arquitetural.

Gabarito: Errado

---

5. **(CESPE / BANRISUL – 2022)** O cálculo do custo da qualidade engloba os custos necessários para a execução de atividades relacionadas à qualidade, mas não os custos gerados pela falta de qualidade.

Comentários:

De acordo com Pressman, custos de qualidade incluem custos necessários para busca de qualidade e para execução de atividades relacionadas à qualidade ou pela falta de qualidade. Ao reunirmos métricas, promovemos a base de custo corrente da qualidade e identificamos oportunidades de redução.

Gabarito: Errado

---

6. **(CESPE / BANRISUL – 2022)** Entre as atividades que ajudam uma equipe a atingir o alto padrão de qualidade de software, a garantia da qualidade é aquela que engloba um conjunto de ações de engenharia de software que contribui para que cada produto resultante atinja suas metas de qualidade.

Comentários:

A garantia da qualidade trata do processo, enquanto o controle de qualidade trata do produto. Logo, o **controle de qualidade** é aquele que engloba um conjunto de ações de engenharia de software que contribui para que cada produto resultante atinja suas metas de qualidade.

Gabarito: Errado

---

7. **(CESPE / BANRISUL – 2022)** Funcionalidade, atributo fundamental de qualidade para software, é aquele que avalia o grau com que o software satisfaz às necessidades declaradas por seus subatributos, tais quais adequabilidade, exatidão, interoperabilidade, conformidade e segurança.

Comentários:

O atributo de funcionalidade trata da capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas. Dentre seus subatributos, temos: adequação, acurácia, interoperabilidade, segurança de acesso e conformidade. A questão mudou um pouquinho os nomes, mas está perfeita!

Gabarito: Correto

---

8. **(CESPE / BANRISUL – 2022)** O DFR (Design For Reuse) deve ser considerado quando se inicia a criação de um novo componente.

Comentários:

O DFR (Design For Reuse) é uma abordagem para maximizar a reutilização e minimizar a redundância ao projetar componentes de software. Esta abordagem ajuda a criar componentes que podem ser facilmente reaproveitados em outros projetos. O DFR também ajuda a reduzir o tempo de desenvolvimento e os custos do projeto. Logo, ele deve ser considerado quando se inicia a criação de um novo componente.

Gabarito: Correto

---

9. **(CESPE / BANRISUL – 2022)** O padrão ISO 9126, desenvolvido como tentativa de identificar os atributos fundamentais de qualidade de software para computador, identifica estes seis atributos fundamentais de qualidade: a funcionalidade, a confiabilidade, a usabilidade, a eficiência, a facilidade de manutenção e a portabilidade.

Comentários:

O padrão ISO 9126 foi desenvolvido como uma tentativa de identificar os atri-butos fundamentais de qualidade para software de computador. O padrão identifica seis atributos fundamentais de qualidade: funcionalidade, confiabilidade, usabilidade, eficiência, facilidade de manutenção e portabilidade.

Gabarito: Correto

---

10. **(CESPE / BANRISUL – 2022)** A portabilidade, atributo fundamental de qualidade do padrão ISO 9126, refere-se ao grau de otimização do uso, pelo software, dos recursos do sistema.

Comentários:

A portabilidade refere-se à facilidade com a qual um software pode ser transposto de um ambiente para outro, conforme indicado pelos seguintes subatri-butos: adaptabilidade, facilidade de

instalação, conformidade, facilidade de substituição. A questão trata - na verdade - do atributo de eficiência.

Gabarito: Errado

---

11. (CESPE / BANRISUL – 2022) As características operacionais, a capacidade de suportar mudanças e a adaptabilidade a novos ambientes são os aspectos de um produto de software em que se concentra a categorização dos fatores que afetam a qualidade de software.

Comentários:

Os fatores de qualidade de software de McCall se concentram em três aspectos dos produtos de software: características operacionais, capacidade de serem alterados e adaptabilidade a novos ambientes. Nesse contexto, temos: correção, confiabilidade, eficiência, integridade, usabilidade, facilidade de manutenção, flexibilidade, testabilidade, portabilidade, reusabilidade e interoperabilidade.

Gabarito: Correto

---

12. (CESPE / BANRISUL – 2022) A usabilidade é um atributo de qualidade de um projeto que avalia se ele fornece os recursos que os usuários precisam.

Comentários:

O atributo de qualidade de um projeto que avalia se ele fornece os recursos que os usuários precisam é a funcionalidade. Ela mede se o produto tem as características e recursos desejados pelos usuários.

Gabarito: Errado

---

13. (CESPE / SERPRO – 2010) A garantia de qualidade tem como objetivo testar os produtos de software de modo a identificar, relatar e remover os defeitos encontrados, enquanto o controle da qualidade provê a gerência sênior da organização com a visibilidade apropriada sobre o processo de desenvolvimento.

Comentários:

A questão apenas inverteu os conceitos de garantia e controle de qualidade. O controle de qualidade tem como objetivo testar os produtos de software de modo a identificar, relatar e remover os defeitos encontrados, enquanto a garantia da qualidade provê a gerência sênior da organização com a visibilidade apropriada sobre o processo de desenvolvimento.

Gabarito: Errado

---

14. (CESPE / SERPRO – 2010) Um processo de gerenciamento da qualidade do projeto tipicamente visa garantir e controlar a qualidade. No controle da qualidade, são executadas atividades planejadas e sistemáticas visando garantir que o projeto empregará os processos necessários para atender aos requisitos. Por sua vez, a garantia da qualidade, diferentemente do controle de qualidade, monitora resultados do projeto a fim de determinar se eles estão de acordo com os padrões relevantes de qualidade e procura identificar meios para eliminar as causas de resultados que sejam insatisfatórios.

Comentários:

Mais uma questão que apenas inverte os conceitos de garantia e controle de qualidade – esse tipo de questão é muito comum, portanto os conceitos devem estar bem consolidados na memória.

Gabarito: Errado

---

15. (CESPE / MEC – 2015) A qualidade deve ser inserida em etapas específicas do ciclo de vida do produto de software.

Comentários:

Opa... a qualidade deve estar inserida em todas as etapas do ciclo de vida do produto de software e, não, em uma específica.

Gabarito: Errado

---

16. (CESPE / TCE-RO – 2013) Controle, planejamento e garantia de qualidade são atividades do gerenciamento de qualidade; o controle de qualidade estabelece procedimentos e padrões que objetivam o desenvolvimento de software com qualidade.

Comentários:

Mais uma inversão de conceitos: a garantia de qualidade estabelece procedimentos organizacionais e padrões que conduzem a um software de alta qualidade.

Gabarito: Errado

---

17. (CESPE / BASA – 2010) Para garantir o desenvolvimento de qualidade, é suficiente que a equipe tenha as ferramentas mais atuais de engenharia de software e os melhores computadores.

Comentários:

Vocês já imaginaram as melhores ferramentas nas mãos de uma equipe ruim? Ou utilizando um processo ruim? Pois é, não! Pessoas, processos e ferramentas devem andar juntas.

Gabarito: Letra E

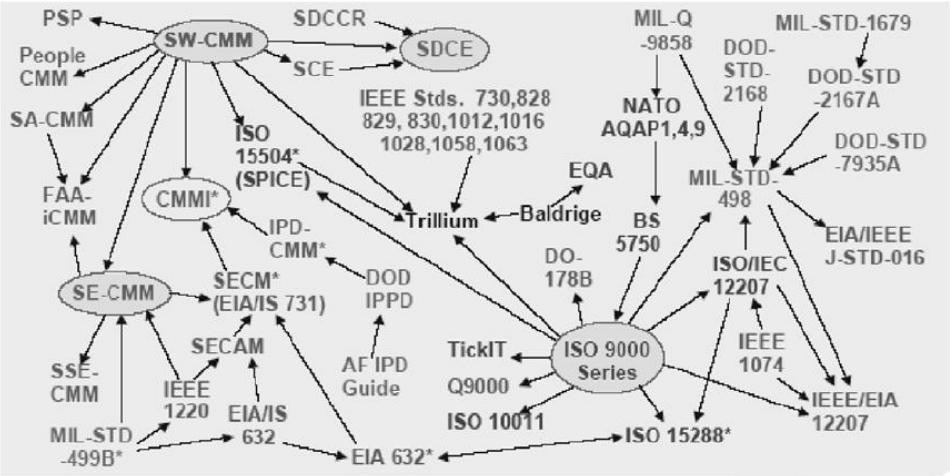
---

18. (CESPE / INMETRO – 2009) Um modelo para a avaliação contínua de capacidade de processos é descrito na norma NBR ISO/IEC 9126, que deriva do ciclo da melhoria contínua presente na norma ISO 9001.

Comentários:

A NBR ISO/IEC 15504 que é responsável pela avaliação contínua de capacidade de processos e, não, NBR ISO/IEC 9126.

Gabarito: Errado



Internet: <www.software.org> (com adaptações).

A figura acima, elaborada no ano de 1998, apresenta uma proposta de relacionamento de precedência de criação e de incorporação de conceitos entre diversos modelos de qualidade de processo de engenharia de software e de sistemas. Não estão representados no modelo as normas e os modelos NBR ISO/IEC 27001, MPS.BR e NBR ISO/IEC 9126.

19. (CESPE / INMETRO – 2009) A norma NBR ISO/IEC 9126-1, que define atributos de qualidade externa e interna, como funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade, não figura no mapa porque, principalmente, não estava disponível à época.

Comentários:

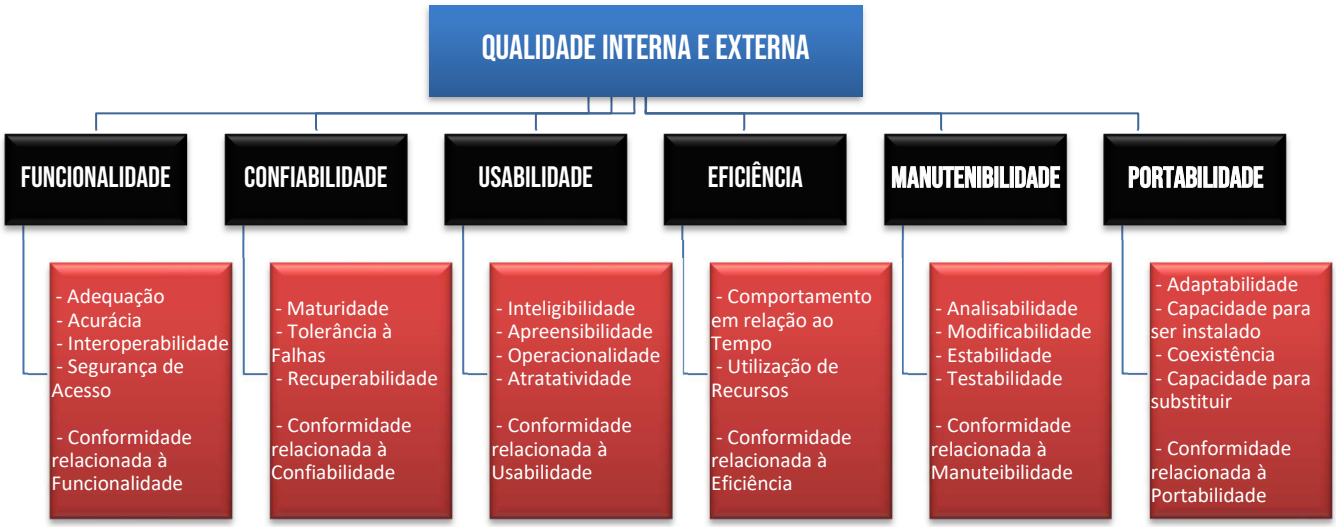
Na verdade, estava disponível - ela é de 1991! É ridículo cobrar datas em prova :(

Gabarito: Errado

---

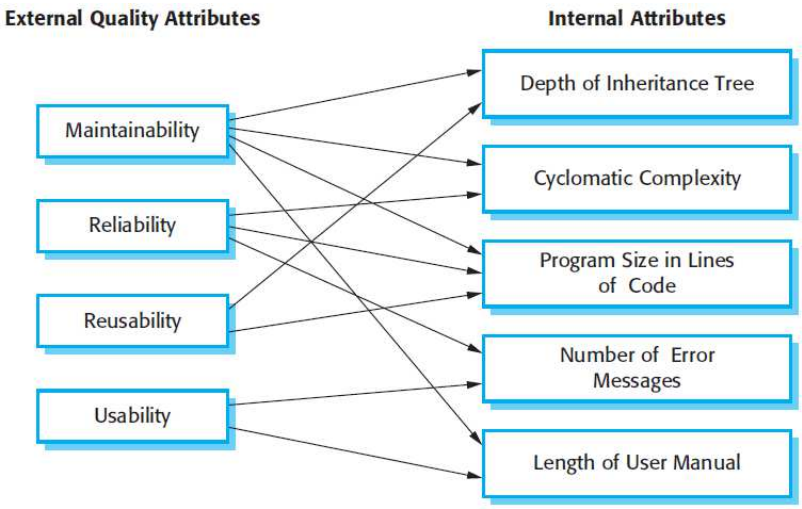
20. (CESPE / STJ – 2015) A manutenibilidade é atributo de qualidade externa que pode ser medida por atributos internos, como a profundidade da árvore de herança e a complexidade ciclomática.

Comentários:



A manutenibilidade é um atributo de qualidade externa (e interna) e pode ser medida por atributos internos (ou subcaracterísticas). No entanto, não encontrei nada na ISO 9126 que fale sobre profundidade da árvore de herança e complexidade ciclomática. Inclusive, essas duas são métricas de complexidade e, não, de manutenibilidade.

Qual é o lance dessa questão? Ela foi retirada do Sommerville, que afirma que o Atributo de Qualidade Externa chamado Manutenibilidade pode ser medido por meio de atributos internos, tais como: Complexidade Ciclomática, Profundidade da Árvore de Herança; Quantidade de Linhas de Código; e Tamanho do Manuel de Usuário. *Entendido?*



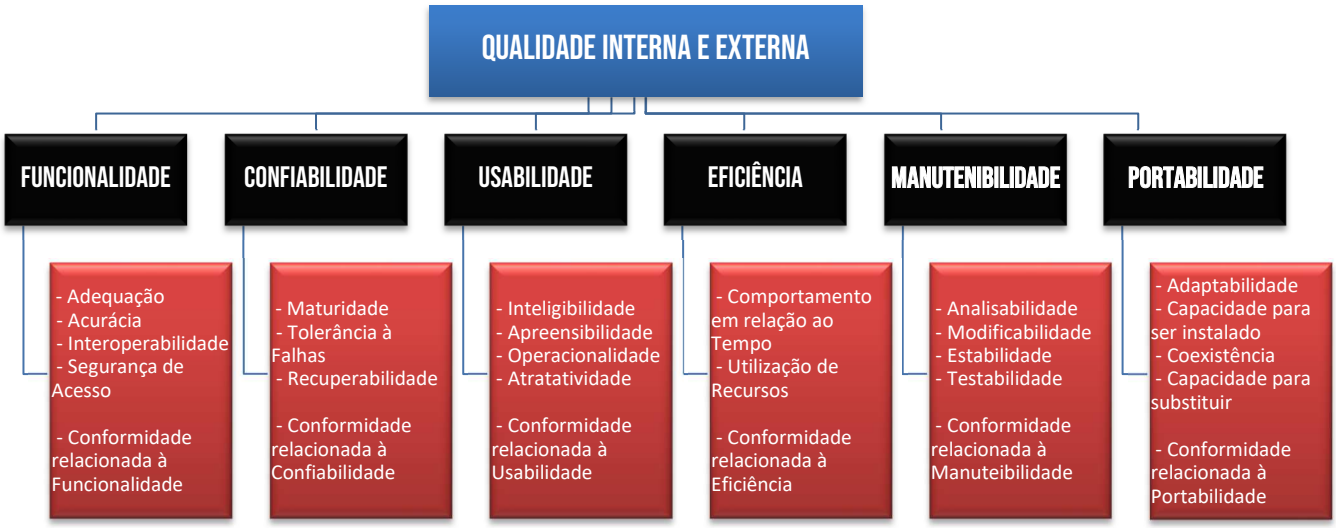
No entanto, o enunciado dessa questão fala efetivamente sobre ISO/IEC 9126! Logo, a questão deveria ter sido anulada!

Gabarito: Correto

---

21. (CESPE / STJ – 2015) A apreensibilidade cuida da capacidade de o usuário compreender se o software é apropriado e como este pode ser usado para a tarefa e as condições específicas.

Comentários:



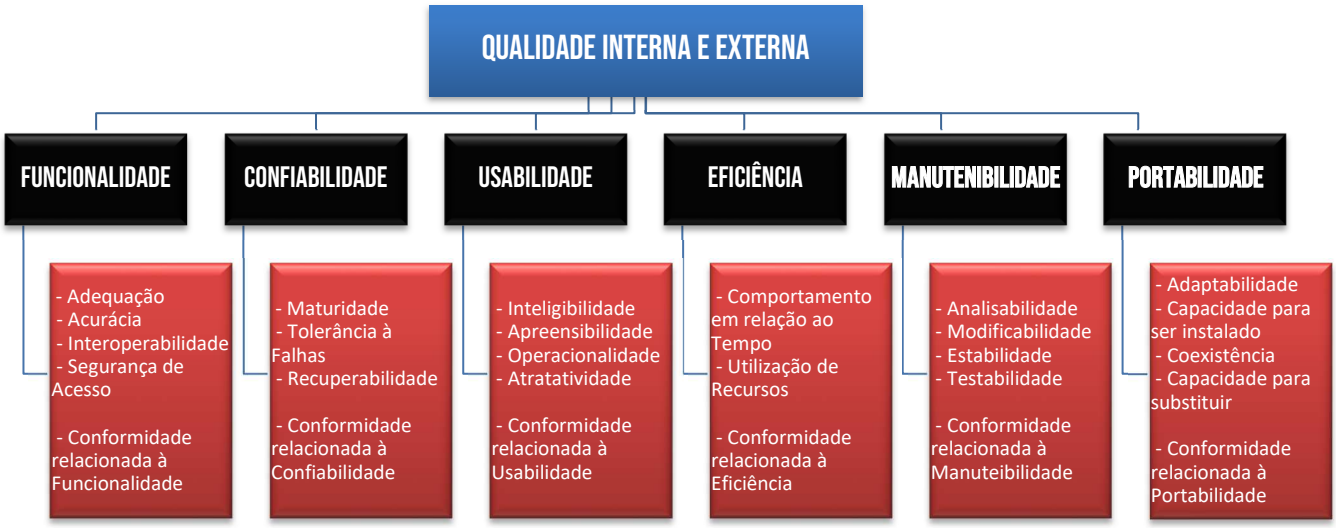
É uma subcaracterística da Usabilidade, mas a questão descreveu a Inteligibilidade, que é outra subcaracterística da Usabilidade. A Apreensibilidade é a capacidade do produto de software de possibilitar ao usuário aprender sua aplicação.

Gabarito: Errado

---

22. (CESPE / STJ – 2015) A funcionalidade e a usabilidade, características dos atributos de qualidade de software, possuem como subcaracterísticas, respectivamente, a operacionalidade e a interoperabilidade.

Comentários:



A questão inverteu as subcaracterísticas. A funcionalidade e a usabilidade, características dos atributos de qualidade de software, possuem como subcaracterísticas, respectivamente, a interoperabilidade e a operacionalidade.

Gabarito: Errado

---

23. (CESPE / STF – 2013) A qualidade de software abrange apenas os aspectos internos e externos decorrentes do uso e, portanto, pode ser medida durante a utilização do software por parte do usuário.

Comentários:

Calma, são coisas diferentes! A qualidade de software abrange a qualidade interna, a qualidade externa e a qualidade em uso. Seria correto dizer que abrange os aspectos internos e externos e os decorrentes de uso. *Ela pode ser medida durante a utilização do software por parte do usuário?* Sim, essa é a qualidade em uso!

Gabarito: Errado

---

24. (CESPE / STF – 2013) Do ponto de vista histórico, o termo usabilidade evoluiu a partir do termo qualidade em uso, que, por sua vez, substituiu o termo interface amigável, principalmente devido à pouca abrangência e subjetividade que estes últimos sugeriam.

Comentários:

Primeiro, o termo qualidade em uso que evoluiu a partir do termo usabilidade. Segundo, esse termo no substituiu esse outro termo. Terceiro, é ridículo cobrar em prova a origem de termos...

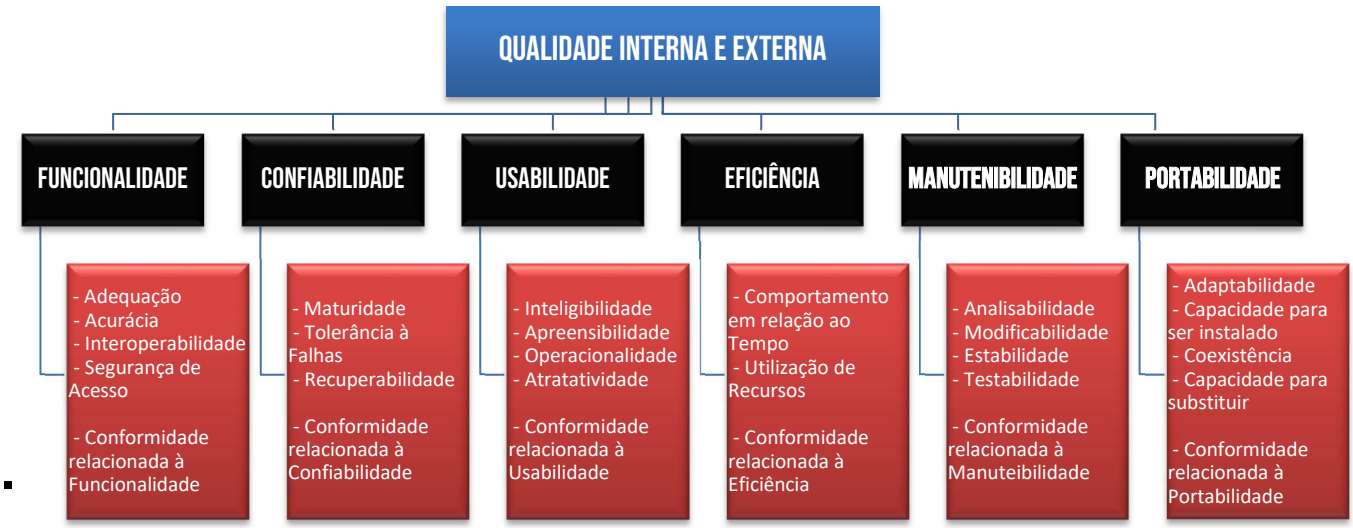
Gabarito: Errado

---

25. (CESPE / TCE-PR – 2016) De acordo com a norma ISO/IEC 9126, os atributos de qualidade de software referentes às características de usabilidade são:

a) inteligibilidade, analisabilidade, conformidade e adaptabilidade.
b) estabilidade, testabilidade, utilização de recursos e acessibilidade.
c) inteligibilidade, comportamento com relação ao tempo, atratividade e operacionalidade.
d) acessibilidade, estética, atratividade, inteligibilidade e apreensibilidade.
e) segurança de acesso, maturidade, atratividade e adaptabilidade.

Comentários:



(a) Errado, visto que Analisabilidade e Adaptabilidade não são atributos de usabilidade; (b) Errado, visto que Estabilidade, Testabilidade, Utilização de Recursos e Acessibilidade não são atributo de usabilidade; (c) Errado, visto que Comportamento com relação ao Tempo não é um atributo de usabilidade; (d) Errado, visto que Acessibilidade e Estética não são atributos de usabilidade; (e) Errado, visto que Segurança de Acesso, Maturidade e Adaptabilidade não são atributos de usabilidade.

Para mim, essa questão deveria ser anulada, visto que não apresenta nenhuma resposta correta.

Gabarito: Letra D

---

26. (CESPE / TCE-PR – 2016) A norma NBR ISO/IEC 9126 define acurácia como a capacidade de um software fornecer resultados com o grau necessário de precisão, sendo, por isso, considerada parte integrante da funcionalidade de um software.

Comentários:

ACURÁCIA	Capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados.

Acurácia trata da capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados, sendo – portanto – parte integrante da funcionalidade de um software.

Gabarito: Correto

---

27. (CESPE / TCE-PR – 2016) Em vez de ser estimada com base na qualidade interna, a qualidade externa do software deve ser avaliada a partir das características do produto pelo ponto de vista externo, segundo a norma NBR ISO/IEC 9126.

Comentários:

A natureza fundamental da qualidade do produto de software representada pela qualidade interna mantém-se inalterada, a menos que seja reprojetada. A qualidade externa é a totalidade das características do produto de software do ponto de vista externo.

Gabarito: Correto

---

28. (CESPE / TRE-PE – 2017) A ISO barra IEC 9126 descreve uma das características do modelo de qualidade de software como capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas. Essa característica corresponde à:

a) confiabilidade.
b) eficiência.
c) manutenibilidade.
d) funcionalidade.
e) usabilidade.

Comentários:

EFICIÊNCIA	Capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas.

A característica que trata da capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas é a Eficiência.

Gabarito: Letra B

---

29. (CESPE / TRE-BA – 2017) As normas da série ISO/IEC 9126 estabelecem como medidas da qualidade de software características como: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Já a série ISO/IEC 14598 estabelece métricas para mensurar o grau de qualidade, bem como requisitos e orientações para a avaliação do produto de software. Com relação às orientações dessas séries, assinale a opção correta.

a) A série 9126 considera as métricas de qualidade internas e externas, mas a série 14598 não considera as métricas de qualidade em uso.

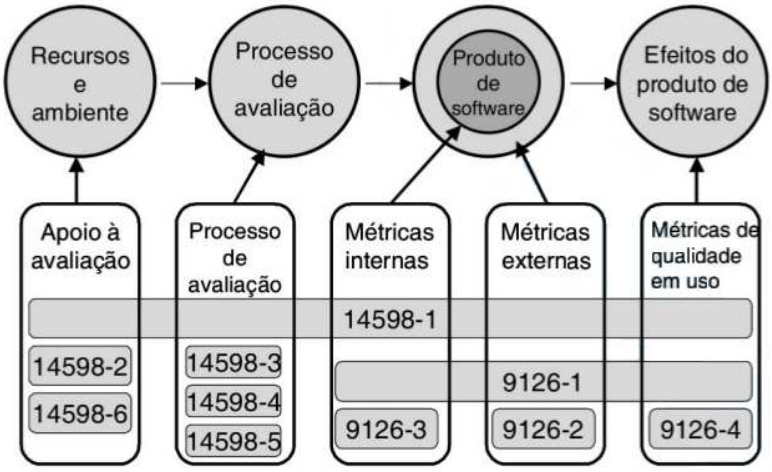
b) A série 14598 considera as métricas de qualidade internas e externas, mas a série 9126 não considera as métricas de qualidade em uso.

c) A série 9126 considera as métricas de qualidade internas, mas a série 14598 não considera as métricas externas nem as de qualidade em uso.

d) A série 14598 considera as métricas de qualidade internas, mas a série 9126 não considera as métricas externas nem as de qualidade em uso.

e) As séries 9126 e 14598 consideram tanto as métricas de qualidade internas e externas quanto as métricas de qualidade em uso.

Comentários:



Relação entre ISO 9126 e ISO 14.598

As séries NBR ISO/ IEC 9126 e NBR ISO/ IEC 14598 consideram tanto as métricas de qualidade internas e externas quanto as métricas de qualidade em uso – a imagem explicita isso!

Gabarito: Letra E

---

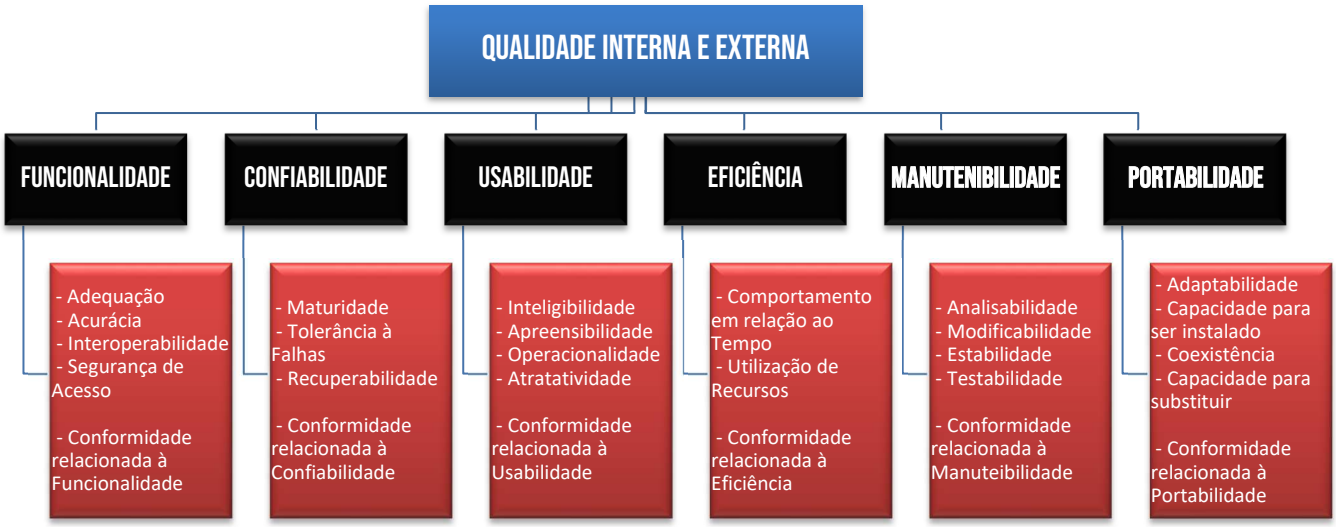
30. (CESPE / TRE-BA – 2017) Um gestor de desenvolvimento de software ficou responsável por avaliar a qualidade de determinado software. Nessa avaliação, ele utilizou atributos

categorizados em características, como, por exemplo, a funcionalidade. Para essa característica — funcionalidade —, o usuário do software pode utilizar como métricas as subcaracterísticas:

a) eficiência e interoperabilidade.
b) manutenibilidade e portabilidade.
c) adequação e acurácia.
d) maturidade e confiabilidade.
e) inteligibilidade e usabilidade.

Comentários:

Questão decoreba! Tinha que lembrar da nossa tabelinha da Norma ISO/IEC 9126: Funcionalidade é a capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas; Adequação é a capacidade do produto de software de prover um conjunto apropriado de funções para tarefas e objetivos do usuário especificados; Acurácia é capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados.



Gabarito: Letra C

---

31. (CESPE / TRE-BA – 2017) Um usuário avaliou um software sob o ponto de vista da qualidade em uso, em complemento à medição de qualidade interna e externa do referido software. O produto, sob a perspectiva do usuário, falhou em lhe permitir o atingimento das metas especificadas com acurácia e completude em um contexto de uso especificado. Nessa situação, o software avaliado falhou no atributo:

a) segurança.
b) eficácia.
c) satisfação.
d) analisabilidade.

e) produtividade.

Comentários:

CARACTERÍSTICA	DESCRIÇÃO
EFICÁCIA	Capacidade do produto de software de permitir que usuários atinjam metas especificadas com acurácia e completude, em um contexto de uso especificado.
PRODUTIVIDADE	Capacidade do produto de software de permitir que seus usuários empreguem quantidade apropriada de recursos em relação à eficácia obtida, em um contexto de uso especificado.
SEGURANÇA	Capacidade do produto de software de apresentar níveis aceitáveis de riscos de danos a pessoas, negócios, software, propriedades ou ao ambiente, em um contexto de uso especificado.
SATISFAÇÃO	Capacidade do produto de software de satisfazer usuários, em um contexto de uso especificado.

A eficácia é a capacidade do produto de software de permitir que usuários atinjam metas especificadas com acurácia e completude, em um contexto de uso especificado.

Gabarito: Letra B

QUESTÕES COMENTADAS – FCC

1. **(FCC / TRT1 – 2014)** A qualidade de software constitui-se em um fator de grande importância no seu desenvolvimento. Dentre as propriedades utilizadas para determinar a qualidade de software,

a) mede-se, exclusivamente, a qualidade da documentação produzida para o software.
b) verifica-se a satisfação de requisitos estabelecidos, incluindo o desempenho.
c) não se abrange questões relativas à interface do software.
d) não há preocupação com a facilidade de manutenção do software.
e) não se inclui a confiabilidade esperada do software.

Comentários:

(a) Errado. *Apenas a documentação?* Não, inclusive a documentação – mas mede-se diversos aspectos do software; (b) Correto. Verifica se os requisitos estabelecidos forem satisfeitos pelo software desenvolvimento – tanto funcionais como não-funcionais (Ex: desempenho); (c) Errado. Abrange, sim! Na verdade, abrange-se tanto requisitos funcionais como não-funcionais (ex: interface); (d) Errado. Há preocupação, sim! Facilidade de manutenção de software é um requisito não-funcional que deve ter preocupação com a qualidade; (e) Errado. Inclui, sim! Esse também é um requisito não-funcional que deve ser incluído na preocupação com a qualidade de um software.

Gabarito: Letra B

---

2. **(FCC / AFR-SP – 2009)** Na prática de garantia de qualidade de software, contrapondo com o controle de qualidade de software, se aplica a atividade:

a) Definir planos de desenvolvimento de teste.
b) Executar teste de software.
c) Desenvolver casos de teste.
d) Definir métricas e medição.
e) Definir estratégias de testes.

Comentários:

Para responder essa pergunta, devemos buscar o item que se foca no processo de desenvolvimento de software e, não, no produto. Observem que todos os itens falam de testes, que se referem em geral ao Controle de Qualidade. Já o quarto item se refere à definição de métricas e medição, que tem função de melhorar o processo de software.

Gabarito: Letra D

---

3. **(FCC / TRE-SE – 2007)** Considere as questões chave apresentadas na seguinte tabela, com o enfoque da ISO 9126 (NBR 13596) ? Qualidade de Software

	Questão chave
I.	Propõe-se a fazer o que é apropriado?
II.	Faz o que foi proposto de forma correta?
III.	Com que freqüência apresenta falhas?
IV.	Há grande risco quando se faz alterações?

As seguintes sub características aplicáveis à avaliação da qualidade do software: Maturidade, Estabilidade, Acurácia e Adequação são aplicáveis, respectivamente, às questões chave:

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

Comentários:

Maturidade é a capacidade do produto de software de evitar falhas decorrentes de defeitos no software; Estabilidade é a capacidade do produto de software de evitar efeitos inesperados decorrentes de modificações no software; Acurácia é a capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados; Adequação é a capacidade do produto de software de prover um conjunto apropriado de funções para tarefas e objetivos do usuário especificados.

Gabarito: Letra C

---

4. **(FCC / TJ-PE – 2012)** No contexto dos atributos de qualidade de software, considere:

I. A resiliência é a capacidade de o sistema voltar ao nível de desempenho anterior a falhas ou comportamento imprevisto de usuários, software ou hardware e recuperar os dados afetados, caso existam.

II. O desempenho e uso de recursos referem-se à capacidade do sistema de alcançar tempos de resposta, latência, tempo de processamento, vazão, etc dentro do período de tempo especificado e ao fato do software exigir mais ou menos recursos de acordo com suas condições de uso.

III. A analisabilidade é o grau de facilidade, com qual seja possível procurar por deficiências no software ou por partes que devem ser modificadas para algum fim.

As subcaracterísticas contidas nos itens I, II e III referem-se, respectivamente, aos atributos de qualidade:

a) funcionabilidade, confiabilidade e usabilidade.
b) eficiência, manutenibilidade e portabilidade.
c) funcionabilidade, usabilidade e manutenibilidade.
d) confiabilidade, eficiência e manutenibilidade
e) confiabilidade, eficiência e portabilidade.

Comentários:

RECUPERABILIDADE	Capacidade do produto de software de restabelecer seu nível de desempenho especificado e recuperar os dados diretamente afetados no caso de uma falha.
COMPORTAMENTO EM RELAÇÃO AO TEMPO	Capacidade do produto de software de fornecer tempos de resposta e de processamento, além de taxas de transferência, apropriados, quando o software executa suas funções, sob condições estabelecidas.
UTILIZAÇÃO DE RECURSOS	Capacidade do produto de software de usar tipos e quantidades apropriados de recursos, quando o software executa suas funções sob condições estabelecidas.
ANALISABILIDADE	Capacidade do produto de software de permitir o diagnóstico de deficiências ou causas de falhas no software, ou a identificação de partes a serem modificadas.

Trata-se da confiabilidade, eficiência e manutenibilidade. Porém, observem que a questão dá alguns nomes diferentes da norma: resiliência e recuperabilidade; e comportamento em relação ao tempo e desempenho. Logo, eu acredito que caberia recurso nessa questão.

Gabarito: Letra D

QUESTÕES COMENTADAS – FGV

1. **(FGV / TJ-RO - 2021)** A equipe de desenvolvimento de software SystemsXYZ vem enfrentando problemas de defeitos associados à qualidade do software. Por isso, a equipe decidiu adotar medições de software baseadas em métricas de produto. Métricas de produto dinâmicas são coletadas por meio de medições efetuadas de um programa em execução, ajudando a avaliar a sua eficiência. Um exemplo associado à métrica dinâmica é o(a):

a) fan-in/fan-out;
b) tamanho de código-fonte;
c) comprimento médio de identificadores;
d) número de relatórios de bugs;
e) complexidade ciclomática.

Comentários:

Todas as alternativas trazem métricas estáticas, exceto a letra (d). As métricas dinâmicas são aquelas que são coletadas por meio de medições efetuadas de um programa em execução. Essas métricas podem ser coletadas durante o teste de sistema ou após o sistema estar em uso. Um exemplo pode ser o número de relatórios de bugs ou o tempo necessário para concluir uma computação.

Gabarito: Letra D

---

2. **(FGV / MEC – 2009)** Analise a citação a seguir.

"*Um conjunto de atributos que têm impacto na capacidade do software de manter o seu nível de desempenho dentro de condições estabelecidas por um dado período de tempo.*"

A Norma que integra os conceitos de ambiente, estratégias e planejamento de testes, é conhecida por:

a) ISO 12207
b) ISO 15504
c) ISO 9126
d) IEEE 829
e) MPS.BR.

Comentários:

A questão fala de atributos que têm impacto na capacidade do software de manter seu nível de desempenho. *O que isso quer dizer, galera?* Qualidade de Software! Um software que mantém seu nível de desempenho é um software de qualidade!

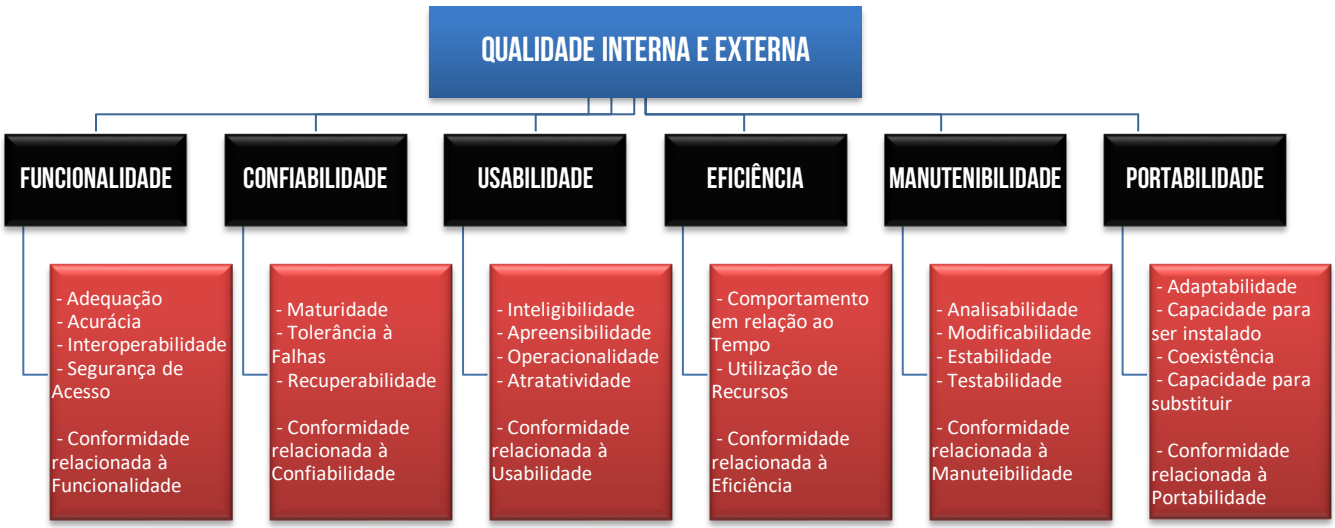
Gabarito: Anulada

---

3. **(FGV / MEC – 2009)** Entre os critérios de qualidade da Norma ISO 9126 não se inclui:

a) a manutenibilidade.
b) a funcionalidade.
c) a confiabilidade.
d) a utilizabilidade.
e) a eficicácia.

Comentários:



Não existe *utilizabilidade* nem *eficicácia* e, por isso, a questão foi anulada.

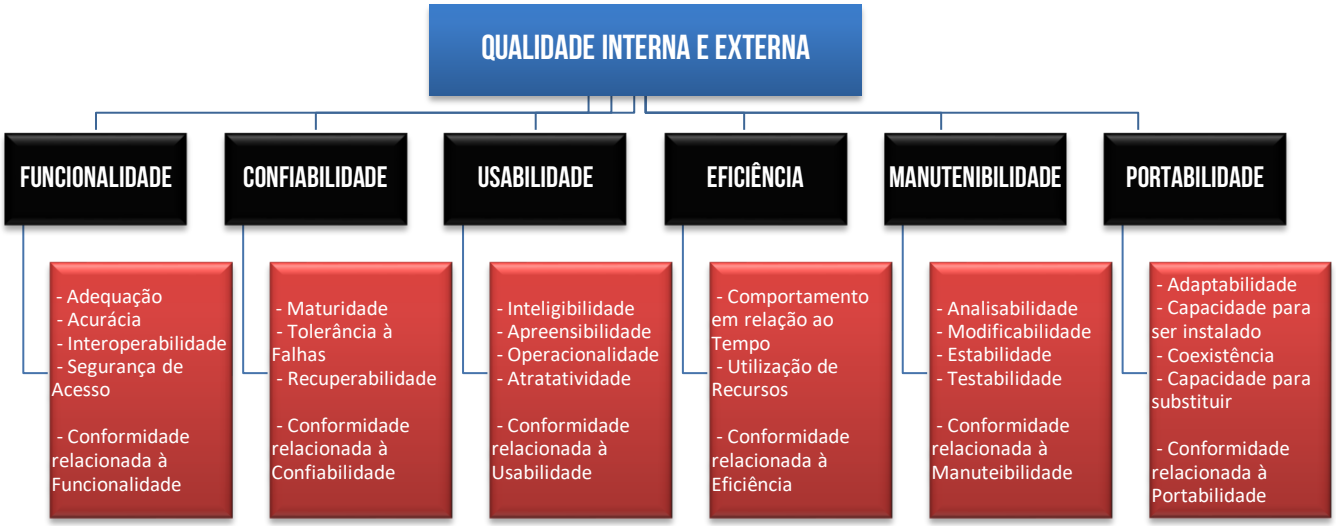
Gabarito: Anulada

---

4. **(FGV / DETRAN-RN – 2010)** Assinale a alternativa que NÃO contém somente atributos para características externas e internas do modelo de qualidade de software, definido na ISO/IEC 9126-1:

a) Funcionalidade, confiabilidade, usabilidade.
b) Funcionalidade, confiabilidade, eficiência.
c) Funcionalidade, confiabilidade, alta gerência.
d) Funcionalidade, usabilidade, portabilidade.
e) Eficiência, manutenibilidade, portabilidade.

Comentários:



Não existe o atributo chamado de Alta Gerência.

Gabarito: Letra C

---

5. **(FGV / ALERJ – 2017)** Um sistema está sendo desenvolvido por uma empresa terceirizada para apoiar as vendas de um mercado varejista da Grande São Paulo denominado "Mendes Sá Colão". Após o desenvolvimento do sistema, a empresa terceirada deverá passar o código fonte para a área de TI da "Mendes Sá Colão", que passará a ser a necessidade de que o sistema, caso ocorra uma falha, se recupere de forma automática e rapidamente. Nesse caso, os atributos de qualidade do sistema com maior peso são:

a) Portabilidade e confiabilidade;
b) Manutenibilidade e confiabilidade;
c) Portabilidade e Eficiência;
d) Confiabilidade e Usabilidade;
e) Manutenibilidade e eficiência.

Comentários:

MANUTENIBILIDADE	Capacidade do produto de software de ser modificado. As modificações podem incluir correções, melhorias ou adaptações devido a mudanças no ambiente e nos seus requisitos ou especificações funcionais.
CONFIABILIDADE	Capacidade do produto de software de manter um nível de desempenho especificado, quando usado em condições especificadas.

Sempre que uma equipe desenvolve um código-fonte e o repassa para que outra equipe passe a mantê-lo, há um risco manutenibilidade. *Por que?* Porque o código pode conter "gambiarras", pode

estar pouco comentado, pode ter baixa qualidade, entre outros. Além disso, a questão destaca que, caso ocorra uma falha, o software deve ser recuperar de forma automática e rápida. Logo, há uma necessidade inerente de que o código seja confiável.

Gabarito: Letra B

QUESTÕES COMENTADAS – DIVERSAS BANCAS

1. **(FADESP / SEFA-PA – 2022)** Sobre as métricas de produtos usadas no gerenciamento de qualidade de projetos de software, analise as afirmativas a seguir, julgando-as verdadeiras (V) ou falsas (F).

   I. As métricas permanentes são coletadas por meio de medições efetuadas em um programa em execução, podendo ser coletadas durante o teste de sistema ou após o sistema estar em uso.

   II. As métricas estáticas são coletadas por meio de medições realizadas nas representações do sistema, como o projeto, o programa ou a documentação.

   III. O índice fog é uma métrica estática de produto de software em que é medido o comprimento médio de palavras e sentenças em documentos. Quanto maior o valor de um índice fog de um documento, mais difícil a sua compreensão.

   A sequência que expressa corretamente o julgamento das afirmativas é:

   a) I - V; II - V; III - F
   b) I - F; II - F; III - V
   c) I - V; II - F; III - F
   d) I - F; II - V; III - F
   e) I - F; II - V; III - V

Comentários:

(I) Errado, essas são as métricas dinâmicas – não existem métricas permanentes; (II) Correto, as métricas estáticas são coletadas por meio de medições feitas de representações do sistema, como o projeto, o programa ou a documentação. Exemplos de métricas estáticas são o tamanho de código e o comprimento médio de identificadores utilizados no código-fonte; (III) Correto, essa métrica é uma medida do comprimento médio de palavras e sentenças em documentos. Quanto maior o valor de um índice Fog de um documento, mais difícil a sua compreensão.

Gabarito: Letra E

---

2. **(IDECAN / BANESTES – 2012)** A garantia da qualidade de software compreende uma variedade de tarefas associadas a atividades como

   I. diagrama de fluxo de dados.
   II. aplicação de métodos técnicos.
   III. aplicação de padrões.
   IV. controle de mudanças.

V. medição.

Estão corretas apenas as alternativas

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

Comentários:

Questão bastante abstrata de uma banca fraca. Enfim, todos os itens estão – de alguma forma – relacionados à qualidade, exceto Diagrama de Fluxo de Dados (DFD) – que não tem absolutamente nenhuma relação com qualidade.

Gabarito: Letra A

---

3. **(FUNIVERSA / CEB – 2010)** Qualidade de software é uma área da engenharia de software que tem como objetivo garantir a qualidade pela definição e normatização dos processos de desenvolvimento de sistemas. O grupo de normas técnicas "ISO 9000/2000" define qualidade como o grau em que um conjunto de características inerentes a um produto, processo ou sistema cumpre os requisitos inicialmente estipulados para esses. Assinale a alternativa que melhor define "qualidade", dentro da área de engenharia de software.

a) Conformidade de um sistema com os requisitos levantados no início do processo de desenvolvimento.

b) Tempo de vida útil de um sistema e sua efetiva utilidade e aplicação.

c) É medida pelo máximo de tempo de uso entre falhas ocorridas (MTBF) no ciclo de vida do software.

d) Desempenho medido pelo tempo de resposta no processamento e apresentação das informações.

e) Equilíbrio entre o prazo de entrega do sistema e o atendimento mínimo dos requisitos levantados.

Comentários:

Os quatro últimos itens apresentam algumas características que permitem definir a qualidade de um software, mas não definem qualidade em si. Já o primeiro o faz corretamente, na medida em

que de fato se trata da conformidade de um sistema com os requisitos levantados no início do processo de desenvolvimento.

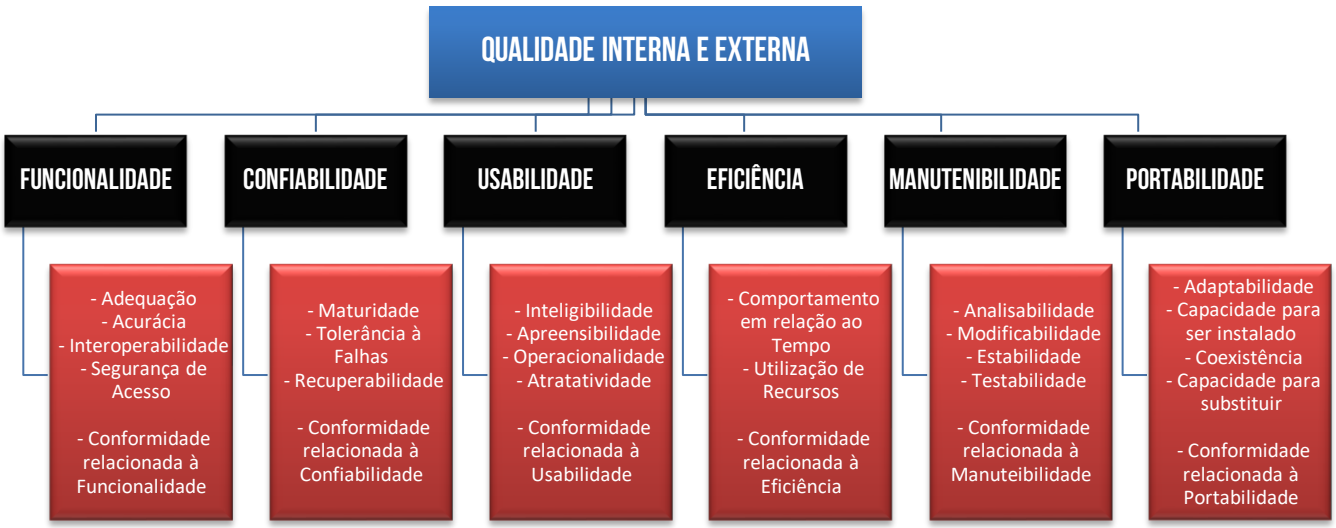
Gabarito: Letra A

---

4. **(CONSULPLAN / Prefeitura de Santa Maria Madalena – 2010)** São categorias de características principais de qualidade de software, segundo a Norma (ISO/IEC 9126: NBR 13596), EXCETO:

a) Eficiência.
b) Segurança.
c) Confiabilidade.
d) Usabilidade.
e) Funcionalidade.

Comentários:



Segurança não faz parte das seis características principais de qualidade de software: Funcionalidade, Confiabilidade, Usabilidade, Eficiência, Manutenibilidade e Portabilidade.

Gabarito: Letra B

---

5. **(COVEST / UFPE – 2013)** A norma ISO/IEC 9126 é um padrão internacional para a qualidade de software, a qual é composta de uma série de características. Sobre essa norma, é correto afirmar que:

a) descreve seis características relacionadas à qualidade de software, e como elas devem ser medidas.

b) compatibilidade é uma das características, que é composta por duas subcaracterísticas: coexistência e interoperabilidade.

c) funcionalidade é a característica que visa verificar se, durante um período de tempo, o sistema funciona de acordo com as condições preestabelecidas.

d) foi atualizada pela norma ISO/IEC 25010, a qual só adicionou a característica de segurança.

e) a característica que determina métricas para avaliar o comportamento do sistema em relação a tempo e a recursos utilizados é a eficiência.

Comentários:

(a) Errado, não descreve como as características devem ser medidas. Na verdade, ela descreve como as métricas devem ser medidas, inclusive apresenta diversas fórmulas. No entanto, percebam que isso é bem sutil; (b) Errado, a compatibilidade não é uma das características; (c) Errado, descreve a capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas; (d) Errado, o item adicionou Segurança, Compatibilidade, Operabilidade e retirou Usabilidade; (e) Correto.

Gabarito: Letra E

---

6. **(IADES / EBSERH – 2013)** De acordo com o padrão de qualidade ISO 9126, são identificados seis atributos fundamentais da qualidade. Sobre o tema, assinale a alternativa correta.

a) A usabilidade diz respeito à quantidade de tempo, que o software fica disponível para uso.

b) A eficiência é o grau com que o software satisfaz às necessidades declaradas.

c) A disponibilidade é o grau de tempo em que o software permanece no ar para utilização

d) A portabilidade é a facilidade com a qual um software pode ser transportado de um ambiente para outro.

e) A confidencialidade é a capacidade de manter partes do software, em sigilo, só sendo permitido o conhecimento, por parte de pessoas autorizadas.

Comentários:

(a) Errado, a questão trata da confiabilidade; (b) Errado, a questão trata da funcionalidade; (c) Errado, a disponibilidade não é um atributo de qualidade da ISO/IEC 9126; (d) Correto; (e) Errado, confidencialidade não é um atributo da ISO/IEC 9126.

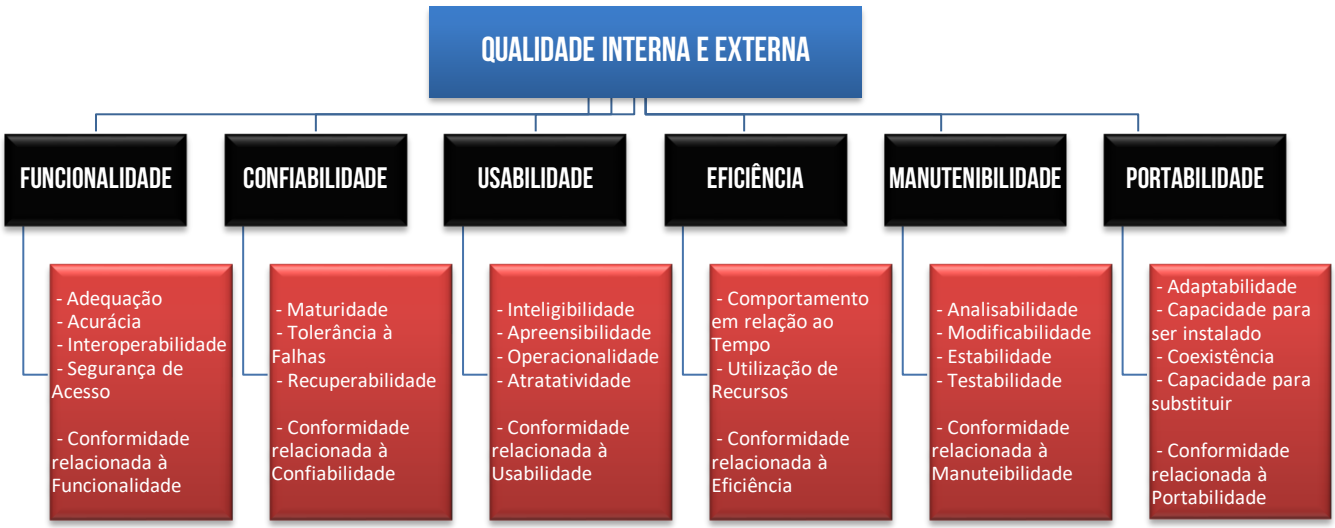
Gabarito: Letra D

---

7. **(UFF / DATAPREV – 2008)** A norma ISO 9.126 foi desenvolvida para identificar atributos de qualidade para software de computador. O período de tempo em que o software está disponível para uso, indicado pelos sub-atributos maturidade, tolerância à falha e recuperabilidade, é caracterizado pelo atributo-chave:

a) portabilidade;
b) confiabilidade;
c) manutenibilidade;
d) eficiência;
e) funcionalidade.

Comentários:



A questão fala em período de tempo disponível para uso, maturidade, recuperabilidade e tolerância a falhas, logo trata-se da confiabilidade!

Gabarito: Letra B

---

8. **(QUADRIX / COBRA – 2015)** De acordo com a norma ISO/IEC 9126, a qualidade do produto software está relacionada às seguintes características: Funcionalidade, Confiabilidade, Usabilidade, Eficiência, Manutenibilidade e Portabilidade. Sobre o tema, assinale a afirmação correta.

a) A Manutenibilidade diz que o produto de software deve ser capaz de manter seu nível de desempenho, ao longo do tempo, nas condições estabelecidas.

b) A Confiabilidade está relacionada ao esforço necessário para a utilização do sistema, baseado em um conjunto de implicações e de condições do usuário.

c) A Usabilidade refere-se à compatibilidade dos recursos e os tempos envolvidos compatíveis com o nível de desempenho requerido pelo software.

d) A Funcionalidade refere-se à existência de funções e propriedades específicas do produto, que satisfazem as necessidades do usuário.

e) A Eficiência diz respeito à facilidade de o software poder ser transferido de um ambiente para outro.

Comentários:

FUNCIONALIDADE	Capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas.

A funcionalidade trata da capacidade do produto de software de prover funções que atendam às necessidades explícitas e implícitas, quando o software estiver sendo utilizado sob condições especificadas.

Gabarito: Letra D

---

9. **(IDECAN / AGU – 2014)** "*Detalhes da qualidade do produto de software podem ser melhorados durante a implementação do código, revisão e teste, mas a natureza fundamental da qualidade do produto de software representada pela qualidade_____ mantém-se inalterada, a menos que seja reprojetada.*"

Assinale a alternativa que completa corretamente a afirmativa anterior.

a) interna
b) em uso
c) externa
d) em uso estimada (ou prevista)
e) externa estimada (ou prevista)

Comentários:

A qualidade interna é a totalidade das características do produto de software do ponto de vista interno. A qualidade interna é medida e avaliada com relação aos requisitos de qualidade interna. Detalhes da qualidade do produto de software podem ser melhorados durante a implementação do código, revisão e teste. A questão trata da qualidade interna...

Gabarito: Letra A

---

10. (VUNESP / DESENVOLVESP – 2014) A norma ISO 9126 (Engenharia de Software – Qualidade do Produto) estabelece um modelo de qualidade com 6 atributos. Dentre eles, está o atributo eficiência, que visa medir:

a) a facilidade de se fazer manutenções corretiva e adaptativa no software.
b) a facilidade de transportar o software de um computador para outro.
c) o número de erros detectados por dia de operação.
d) o nível no qual o software utiliza, de forma otimizada, os recursos do sistema computacional.
e) o tempo máximo decorrido entre duas paradas simultâneas do software.

Comentários:

EFICIÊNCIA	Capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas.

A eficiência trata do nível no qual o software utiliza, de forma otimizada, os recursos do sistema computacional.

Gabarito: Letra D

---

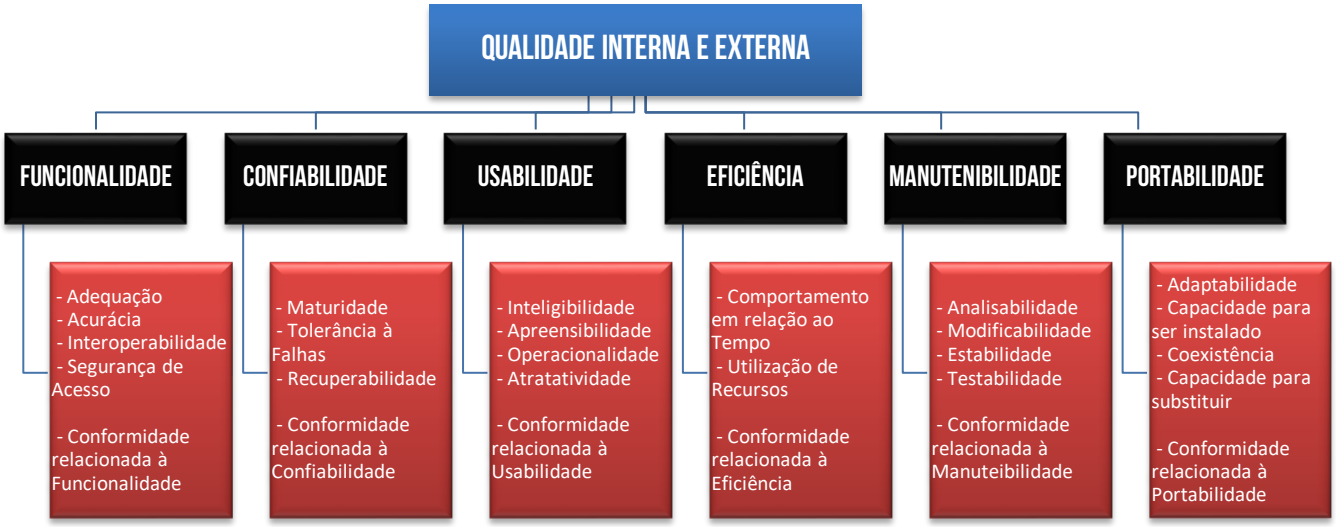
11. (CESGRANRIO / EPE – 2014) Com a evolução das pesquisas na área de qualidade, ficou cada vez mais claro para os pesquisadores que este é um conceito complexo e multifacetado. Muitos autores desenvolveram modelos de qualidade baseados na ideia de descrever qualidade como um conjunto de características ou atributos, organizadas de forma hierárquica. Esse movimento também aconteceu na área de qualidade de software, resultando em múltiplos modelos.

Um marco importante nessa discussão foi o estabelecimento de um modelo padrão de qualidade de software, representado na norma ISO/IEC 9126, que identificou seis características da qualidade de software, cada uma delas com um conjunto de subcaracterísticas.

Com relação a esse padrão, a acurácia, ou seja, a capacidade de o produto de software prover com o grau de precisão necessário resultados ou efeitos corretos ou conforme acordados é uma subcaracterística de:

a) Portabilidade
b) Usabilidade
c) Confiabilidade
d) Eficiência
e) Funcionalidade

Comentários:



A acurácia é uma subcaracterística de funcionalidade.

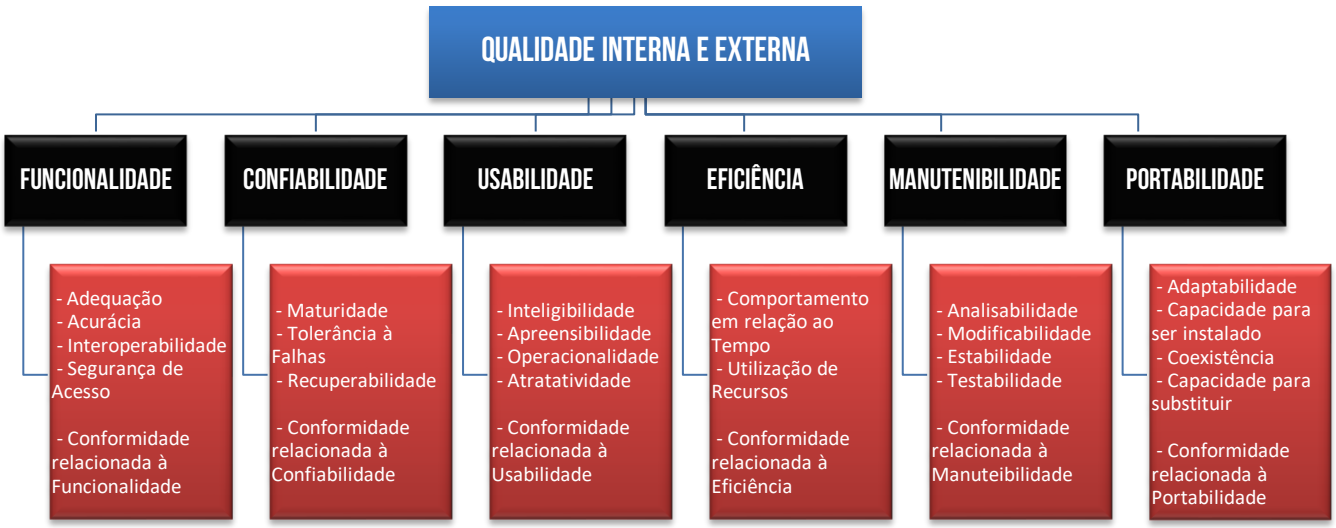
Gabarito: Letra E

---

12. (CESGRANRIO / CHESF – 2012) Dentre os atributos de um software de qualidade, incluem-se:

a) controlabilidade, dependabilidade e eficiência
b) controlabilidade, eficiência e manutenibilidade
c) eficiência, imutabilidade e manutenibilidade
d) eficiência, manutenibilidade e usabilidade
e) imutabilidade, manutenibilidade e usabilidade

Comentários:



A única opção em que todos os itens são atributos de qualidade são: eficiência, manutenibilidade e usabilidade.

Gabarito: Letra D

Lista de Questões – CESPE

1. **(CESPE / BANRISUL – 2022)** Os principais recursos de um sistema de controle de versão incluem um repositório de dados que armazena todos os objetos de configuração relevantes e um recurso de gestão de versão que armazena todas as versões de um objeto de configuração.

2. **(CESPE / BANRISUL – 2022)** De acordo com a SQA (Software Quality Assurance), correção, completude e consistência do modelo de requisitos são características da qualidade do código que influenciam a qualidade de todos os produtos.

3. **(CESPE / BANRISUL – 2022)** Define-se confiabilidade de software como a probabilidade de operação, sem falhas, de um programa de computador em dado ambiente por determinado tempo.

4. **(CESPE / BANRISUL – 2022)** A revisão por pares é uma forma de análise da causa-raiz, na qual a equipe define uma meta ou efeito arquitetural e, então, enuncia as ações relacionadas para o alcance da meta.

5. **(CESPE / BANRISUL – 2022)** O cálculo do custo da qualidade engloba os custos necessários para a execução de atividades relacionadas à qualidade, mas não os custos gerados pela falta de qualidade.

6. **(CESPE / BANRISUL – 2022)** Entre as atividades que ajudam uma equipe a atingir o alto padrão de qualidade de software, a garantia da qualidade é aquela que engloba um conjunto de ações de engenharia de software que contribui para que cada produto resultante atinja suas metas de qualidade.

7. **(CESPE / BANRISUL – 2022)** Funcionalidade, atributo fundamental de qualidade para software, é aquele que avalia o grau com que o software satisfaz às necessidades declaradas por seus subatributos, tais quais adequabilidade, exatidão, interoperabilidade, conformidade e segurança.

8. **(CESPE / BANRISUL – 2022)** O DFR (Design For Reuse) deve ser considerado quando se inicia a criação de um novo componente.

9. **(CESPE / BANRISUL – 2022)** O padrão ISO 9126, desenvolvido como tentativa de identificar os atributos fundamentais de qualidade de software para computador, identifica estes seis atributos fundamentais de qualidade: a funcionalidade, a confiabilidade, a usabilidade, a eficiência, a facilidade de manutenção e a portabilidade.

10. **(CESPE / BANRISUL – 2022)** A portabilidade, atributo fundamental de qualidade do padrão ISO 9126, refere-se ao grau de otimização do uso, pelo software, dos recursos do sistema.

11. **(CESPE / BANRISUL – 2022)** As características operacionais, a capacidade de suportar mudanças e a adaptabilidade a novos ambientes são os aspectos de um produto de software em que se concentra a categorização dos fatores que afetam a qualidade de software.

12. **(CESPE / BANRISUL – 2022)** A usabilidade é um atributo de qualidade de um projeto que avalia se ele fornece os recursos que os usuários precisam.

13. **(CESPE / SERPRO – 2010)** A garantia de qualidade tem como objetivo testar os produtos de software de modo a identificar, relatar e remover os defeitos encontrados, enquanto o controle da qualidade provê a gerência sênior da organização com a visibilidade apropriada sobre o processo de desenvolvimento.

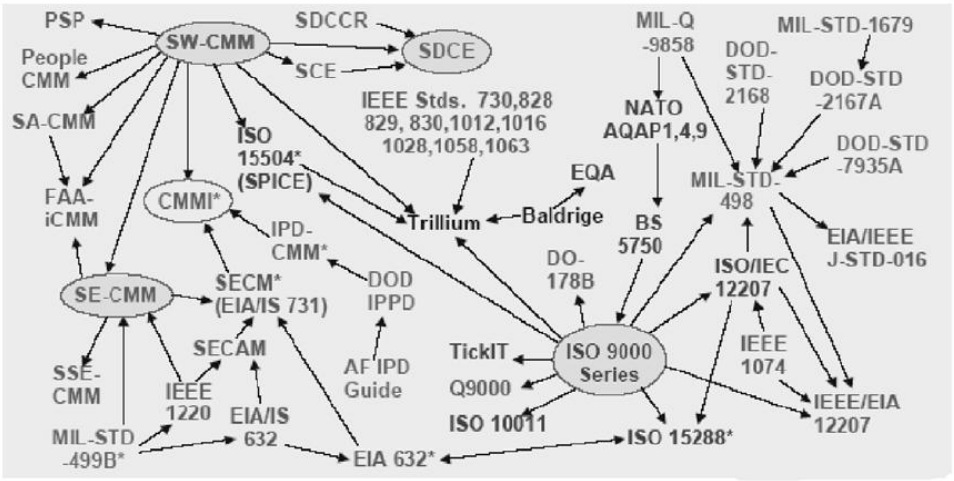
14. **(CESPE / SERPRO – 2010)** Um processo de gerenciamento da qualidade do projeto tipicamente visa garantir e controlar a qualidade. No controle da qualidade, são executadas atividades planejadas e sistemáticas visando garantir que o projeto empregará os processos necessários para atender aos requisitos. Por sua vez, a garantia da qualidade, diferentemente do controle de qualidade, monitora resultados do projeto a fim de determinar se eles estão de acordo com os padrões relevantes de qualidade e procura identificar meios para eliminar as causas de resultados que sejam insatisfatórios.

15. **(CESPE / MEC – 2015)** A qualidade deve ser inserida em etapas específicas do ciclo de vida do produto de software.

16. **(CESPE / TCE-RO – 2013)** Controle, planejamento e garantia de qualidade são atividades do gerenciamento de qualidade; o controle de qualidade estabelece procedimentos e padrões que objetivam o desenvolvimento de software com qualidade.

17. **(CESPE / BASA – 2010)** Para garantir o desenvolvimento de qualidade, é suficiente que a equipe tenha as ferramentas mais atuais de engenharia de software e os melhores computadores.

18. **(CESPE / INMETRO – 2009)** Um modelo para a avaliação contínua de capacidade de processos é descrito na norma NBR ISO/IEC 9126, que deriva do ciclo da melhoria contínua presente na norma ISO 9001.



A figura acima, elaborada no ano de 1998, apresenta uma proposta de relacionamento de precedência de criação e de incorporação de conceitos entre diversos modelos de qualidade de processo de engenharia de software e de sistemas. Não estão representados no modelo as normas e os modelos NBR ISO/IEC 27001, MPS.BR e NBR ISO/IEC 9126.

19. **(CESPE / INMETRO – 2009)** A norma NBR ISO/IEC 9126-1, que define atributos de qualidade externa e interna, como funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade, não figura no mapa porque, principalmente, não estava disponível à época.

20. **(CESPE / STJ – 2015)** A manutenibilidade é atributo de qualidade externa que pode ser medida por atributos internos, como a profundidade da árvore de herança e a complexidade ciclomática.

21. **(CESPE / STJ – 2015)** A apreensibilidade cuida da capacidade de o usuário compreender se o software é apropriado e como este pode ser usado para a tarefa e as condições específicas.

22. **(CESPE / STJ – 2015)** A funcionalidade e a usabilidade, características dos atributos de qualidade de software, possuem como subcaracterísticas, respectivamente, a operacionalidade e a interoperabilidade.

23. **(CESPE / STF – 2013)** A qualidade de software abrange apenas os aspectos internos e externos decorrentes do uso e, portanto, pode ser medida durante a utilização do software por parte do usuário.

24. **(CESPE / STF – 2013)** Do ponto de vista histórico, o termo usabilidade evoluiu a partir do termo qualidade em uso, que, por sua vez, substituiu o termo interface amigável, principalmente devido à pouca abrangência e subjetividade que estes últimos sugeriam.

25. **(CESPE / TCE-PR – 2016)** De acordo com a norma ISO/IEC 9126, os atributos de qualidade de software referentes às características de usabilidade são:

a) inteligibilidade, analisabilidade, conformidade e adaptabilidade.
b) estabilidade, testabilidade, utilização de recursos e acessibilidade.
c) inteligibilidade, comportamento com relação ao tempo, atratividade e operacionalidade.
d) acessibilidade, estética, atratividade, inteligibilidade e apreensibilidade.
e) segurança de acesso, maturidade, atratividade e adaptabilidade.

26. **(CESPE / TCE-PR – 2016)** A norma NBR ISO/IEC 9126 define acurácia como a capacidade de um software fornecer resultados com o grau necessário de precisão, sendo, por isso, considerada parte integrante da funcionalidade de um software.

27. **(CESPE / TCE-PR – 2016)** Em vez de ser estimada com base na qualidade interna, a qualidade externa do software deve ser avaliada a partir das características do produto pelo ponto de vista externo, segundo a norma NBR ISO/IEC 9126.

28. **(CESPE / TRE-PE – 2017)** A ISO barra IEC 9126 descreve uma das características do modelo de qualidade de software como capacidade do produto de software de apresentar desempenho apropriado, relativo à quantidade de recursos usados, sob condições especificadas. Essa característica corresponde à:

a) confiabilidade.
b) eficiência.
c) manutenibilidade.
d) funcionalidade.
e) usabilidade.

29. **(CESPE / TRE-BA – 2017)** As normas da série ISO/IEC 9126 estabelecem como medidas da qualidade de software características como: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Já a série ISO/IEC 14598 estabelece métricas para mensurar o grau de qualidade, bem como requisitos e orientações para a avaliação do produto de software. Com relação às orientações dessas séries, assinale a opção correta.

a) A série 9126 considera as métricas de qualidade internas e externas, mas a série 14598 não considera as métricas de qualidade em uso.

b) A série 14598 considera as métricas de qualidade internas e externas, mas a série 9126 não considera as métricas de qualidade em uso.

c) A série 9126 considera as métricas de qualidade internas, mas a série 14598 não considera as métricas externas nem as de qualidade em uso.

d) A série 14598 considera as métricas de qualidade internas, mas a série 9126 não considera as métricas externas nem as de qualidade em uso.

e) As séries 9126 e 14598 consideram tanto as métricas de qualidade internas e externas quanto as métricas de qualidade em uso.

30. **(CESPE / TRE-BA – 2017)** Um gestor de desenvolvimento de software ficou responsável por avaliar a qualidade de determinado software. Nessa avaliação, ele utilizou atributos categorizados em características, como, por exemplo, a funcionalidade. Para essa característica — funcionalidade —, o usuário do software pode utilizar como métricas as subcaracterísticas:

a) eficiência e interoperabilidade.
b) manutenibilidade e portabilidade.
c) adequação e acurácia.
d) maturidade e confiabilidade.
e) inteligibilidade e usabilidade.

31. **(CESPE / TRE-BA – 2017)** Um usuário avaliou um software sob o ponto de vista da qualidade em uso, em complemento à medição de qualidade interna e externa do referido software. O produto, sob a perspectiva do usuário, falhou em lhe permitir o atingimento das metas especificadas com acurácia e completude em um contexto de uso especificado. Nessa situação, o software avaliado falhou no atributo:

a) segurança.
b) eficácia.
c) satisfação.
d) analisabilidade.
e) produtividade.

GABARITO

1. CORRETO
2. ERRADO
3. CORRETO
4. ERRADO
5. ERRADO
6. ERRADO
7. CORRETO
8. CORRETO
9. CORRETO
10. ERRADO
11. CORRETO
12. ERRADO
13. ERRADO
14. ERRADO
15. ERRADO
16. ERRADO
17. LETRA E
18. ERRADO
19. ERRADO
20. CORRETO
21. ERRADO
22. ERRADO
23. ERRADO
24. ERRADO
25. LETRA D
26. CORRETO
27. CORRETO
28. LETRA B
29. LETRA E
30. LETRA C
31. LETRA B

LISTA DE QUESTÕES – FCC

1. **(FCC / TRT1 – 2014)** A qualidade de software constitui-se em um fator de grande importância no seu desenvolvimento. Dentre as propriedades utilizadas para determinar a qualidade de software,

a) mede-se, exclusivamente, a qualidade da documentação produzida para o software.
b) verifica-se a satisfação de requisitos estabelecidos, incluindo o desempenho.
c) não se abrange questões relativas à interface do software.
d) não há preocupação com a facilidade de manutenção do software.
e) não se inclui a confiabilidade esperada do software.

2. **(FCC / AFR-SP – 2009)** Na prática de garantia de qualidade de software, contrapondo com o controle de qualidade de software, se aplica a atividade:

a) Definir planos de desenvolvimento de teste.
b) Executar teste de software.
c) Desenvolver casos de teste.
d) Definir métricas e medição.
e) Definir estratégias de testes.

3. **(FCC / TRE-SE – 2007)** Considere as questões chave apresentadas na seguinte tabela, com o enfoque da ISO 9126 (NBR 13596) ? Qualidade de Software

Questão chave	
I.	Propõe-se a fazer o que é apropriado?
II.	Faz o que foi proposto de forma correta?
III.	Com que freqüência apresenta falhas?
IV.	Há grande risco quando se faz alterações?

As seguintes sub características aplicáveis à avaliação da qualidade do software: Maturidade, Estabilidade, Acurácia e Adequação são aplicáveis, respectivamente, às questões chave:

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

4. **(FCC / TJ-PE – 2012)** No contexto dos atributos de qualidade de software, considere:

I. A resiliência é a capacidade de o sistema voltar ao nível de desempenho anterior a falhas ou comportamento imprevisto de usuários, software ou hardware e recuperar os dados afetados, caso existam.

II. O desempenho e uso de recursos referem-se à capacidade do sistema de alcançar tempos de resposta, latência, tempo de processamento, vazão, etc dentro do período de tempo especificado e ao fato do software exigir mais ou menos recursos de acordo com suas condições de uso.

III. A analisabilidade é o grau de facilidade, com qual seja possível procurar por deficiências no software ou por partes que devem ser modificadas para algum fim.

As subcaracterísticas contidas nos itens I, II e III referem-se, respectivamente, aos atributos de qualidade:

a) funcionabilidade, confiabilidade e usabilidade.
b) eficiência, manutenibilidade e portabilidade.
c) funcionabilidade, usabilidade e manutenibilidade.
d) confiabilidade, eficiência e manutenibilidade
e) confiabilidade, eficiência e portabilidade.

GABARITO

1. LETRA B
2. LETRA D
3. LETRA C
4. LETRA D

LISTA DE QUESTÕES – FGV

1. **(FGV / TJ-RO - 2021)** A equipe de desenvolvimento de software SystemsXYZ vem enfrentando problemas de defeitos associados à qualidade do software. Por isso, a equipe decidiu adotar medições de software baseadas em métricas de produto. Métricas de produto dinâmicas são coletadas por meio de medições efetuadas de um programa em execução, ajudando a avaliar a sua eficiência. Um exemplo associado à métrica dinâmica é o(a):

a) fan-in/fan-out;
b) tamanho de código-fonte;
c) comprimento médio de identificadores;
d) número de relatórios de bugs;
e) complexidade ciclomática.

Comentários:

Todas as alternativas trazem métricas estáticas, exceto a letra (d). As métricas dinâmicas são aquelas que são coletadas por meio de medições efetuadas de um programa em execução. Essas métricas podem ser coletadas durante o teste de sistema ou após o sistema estar em uso. Um exemplo pode ser o número de relatórios de bugs ou o tempo necessário para concluir uma computação.

Gabarito: Letra D

---

2. **(FGV / MEC – 2009)** Analise a citação a seguir.

"*Um conjunto de atributos que têm impacto na capacidade do software de manter o seu nível de desempenho dentro de condições estabelecidas por um dado período de tempo.*"

A Norma que integra os conceitos de ambiente, estratégias e planejamento de testes, é conhecida por:

a) ISO 12207
b) ISO 15504
c) ISO 9126
d) IEEE 829
e) MPS.BR.

3. **(FGV / MEC – 2009)** Entre os critérios de qualidade da Norma ISO 9126 não se inclui:

a) a manutenibilidade.
b) a funcionalidade.

c) a confiabilidade.
d) a utilizabilidade.
e) a eficicácia.

4. **(FGV / DETRAN-RN – 2010)** Assinale a alternativa que NÃO contém somente atributos para características externas e internas do modelo de qualidade de software, definido na ISO/IEC 9126-1:

a) Funcionalidade, confiabilidade, usabilidade.
b) Funcionalidade, confiabilidade, eficiência.
c) Funcionalidade, confiabilidade, alta gerência.
d) Funcionalidade, usabilidade, portabilidade.
e) Eficiência, manutenibilidade, portabilidade.

5. **(FGV / ALERJ – 2017)** Um sistema está sendo desenvolvido por uma empresa terceirizada para apoiar as vendas de um mercado varejista da Grande São Paulo denominado "Mendes Sá Colão". Após o desenvolvimento do sistema, a empresa terceirada deverá passar o código fonte para a área de TI da "Mendes Sá Colão", que passará a ser a necessidade de que o sistema, caso ocorra uma falha, se recupere de forma automática e rapidamente. Nesse caso, os atributos de qualidade do sistema com maior peso são:

a) Portabilidade e confiabilidade;
b) Manutenibilidade e confiabilidade;
c) Portabilidade e Eficiência;
d) Confiabilidade e Usabilidade;
e) Manutenibilidade e eficiência.

GABARITO

1. LETRA D
2. ANULADA
3. ANULADA
4. LETRA C
5. LETRA B

LISTA DE QUESTÕES – DIVERSAS BANCAS

1. **(FADESP / SEFA-PA – 2022)** Sobre as métricas de produtos usadas no gerenciamento de qualidade de projetos de software, analise as afirmativas a seguir, julgando-as verdadeiras (V) ou falsas (F).

   I. As métricas permanentes são coletadas por meio de medições efetuadas em um programa em execução, podendo ser coletadas durante o teste de sistema ou após o sistema estar em uso.

   II. As métricas estáticas são coletadas por meio de medições realizadas nas representações do sistema, como o projeto, o programa ou a documentação.

   III. O índice fog é uma métrica estática de produto de software em que é medido o comprimento médio de palavras e sentenças em documentos. Quanto maior o valor de um índice fog de um documento, mais difícil a sua compreensão.

   A sequência que expressa corretamente o julgamento das afirmativas é:

   a) I - V; II - V; III - F
   b) I - F; II - F; III - V
   c) I - V; II - F; III - F
   d) I - F; II - V; III - F
   e) I - F; II - V; III - V

2. **(IDECAN / BANESTES – 2012)** A garantia da qualidade de software compreende uma variedade de tarefas associadas a atividades como

   I. diagrama de fluxo de dados.
   II. aplicação de métodos técnicos.
   III. aplicação de padrões.
   IV. controle de mudanças.
   V. medição.

   Estão corretas apenas as alternativas

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

3. **(FUNIVERSA / CEB – 2010)** Qualidade de software é uma área da engenharia de software que tem como objetivo garantir a qualidade pela definição e normatização dos processos de desenvolvimento de sistemas. O grupo de normas técnicas "ISO 9000/2000" define qualidade como o grau em que um conjunto de características inerentes a um produto, processo ou sistema cumpre os requisitos inicialmente estipulados para esses. Assinale a alternativa que melhor define "qualidade", dentro da área de engenharia de software.

   a) Conformidade de um sistema com os requisitos levantados no início do processo de desenvolvimento.

   b) Tempo de vida útil de um sistema e sua efetiva utilidade e aplicação.

   c) É medida pelo máximo de tempo de uso entre falhas ocorridas (MTBF) no ciclo de vida do software.

   d) Desempenho medido pelo tempo de resposta no processamento e apresentação das informações.

   e) Equilíbrio entre o prazo de entrega do sistema e o atendimento mínimo dos requisitos levantados.

4. **(CONSULPLAN / Prefeitura de Santa Maria Madalena – 2010)** São categorias de características principais de qualidade de software, segundo a Norma (ISO/IEC 9126: NBR 13596), EXCETO:

   a) Eficiência.
   b) Segurança.
   c) Confiabilidade.
   d) Usabilidade.
   e) Funcionalidade.

5. **(COVEST / UFPE – 2013)** A norma ISO/IEC 9126 é um padrão internacional para a qualidade de software, a qual é composta de uma série de características. Sobre essa norma, é correto afirmar que:

   a) descreve seis características relacionadas à qualidade de software, e como elas devem ser medidas.

   b) compatibilidade é uma das características, que é composta por duas subcaracterísticas: coexistência e interoperabilidade.

   c) funcionalidade é a característica que visa verificar se, durante um período de tempo, o sistema funciona de acordo com as condições preestabelecidas.

d) foi atualizada pela norma ISO/IEC 25010, a qual só adicionou a característica de segurança.

e) a característica que determina métricas para avaliar o comportamento do sistema em relação a tempo e a recursos utilizados é a eficiência.

6. **(IADES / EBSERH – 2013)** De acordo com o padrão de qualidade ISO 9126, são identificados seis atributos fundamentais da qualidade. Sobre o tema, assinale a alternativa correta.

a) A usabilidade diz respeito à quantidade de tempo, que o software fica disponível para uso.

b) A eficiência é o grau com que o software satisfaz às necessidades declaradas.

c) A disponibilidade é o grau de tempo em que o software permanece no ar para utilização

d) A portabilidade é a facilidade com a qual um software pode ser transportado de um ambiente para outro.

e) A confidencialidade é a capacidade de manter partes do software, em sigilo, só sendo permitido o conhecimento, por parte de pessoas autorizadas.

7. **(UFF / DATAPREV – 2008)** A norma ISO 9.126 foi desenvolvida para identificar atributos de qualidade para software de computador. O período de tempo em que o software está disponível para uso, indicado pelos sub-atributos maturidade, tolerância à falha e recuperabilidade, é caracterizado pelo atributo-chave:

a) portabilidade;
b) confiabilidade;
c) manutenibilidade;
d) eficiência;
e) funcionalidade.

8. **(QUADRIX / COBRA – 2015)** De acordo com a norma ISO/IEC 9126, a qualidade do produto software está relacionada às seguintes características: Funcionalidade, Confiabilidade, Usabilidade, Eficiência, Manutenibilidade e Portabilidade. Sobre o tema, assinale a afirmação correta.

a) A Manutenibilidade diz que o produto de software deve ser capaz de manter seu nível de desempenho, ao longo do tempo, nas condições estabelecidas.

b) A Confiabilidade está relacionada ao esforço necessário para a utilização do sistema, baseado em um conjunto de implicações e de condições do usuário.

c) A Usabilidade refere-se à compatibilidade dos recursos e os tempos envolvidos compatíveis com o nível de desempenho requerido pelo software.

d) A Funcionalidade refere-se à existência de funções e propriedades específicas do produto, que satisfazem as necessidades do usuário.

e) A Eficiência diz respeito à facilidade de o software poder ser transferido de um ambiente para outro.

9. **(IDECAN / AGU – 2014)** "*Detalhes da qualidade do produto de software podem ser melhorados durante a implementação do código, revisão e teste, mas a natureza fundamental da qualidade do produto de software representada pela qualidade_____ mantém-se inalterada, a menos que seja reprojetada.*"

Assinale a alternativa que completa corretamente a afirmativa anterior.

a) interna
b) em uso
c) externa
d) em uso estimada (ou prevista)
e) externa estimada (ou prevista)

10. **(VUNESP / DESENVOLVESP – 2014)** A norma ISO 9126 (Engenharia de Software – Qualidade do Produto) estabelece um modelo de qualidade com 6 atributos. Dentre eles, está o atributo eficiência, que visa medir:

a) a facilidade de se fazer manutenções corretiva e adaptativa no software.
b) a facilidade de transportar o software de um computador para outro.
c) o número de erros detectados por dia de operação.
d) o nível no qual o software utiliza, de forma otimizada, os recursos do sistema computacional.
e) o tempo máximo decorrido entre duas paradas simultâneas do software.

11. **(CESGRANRIO / EPE – 2014)** Com a evolução das pesquisas na área de qualidade, ficou cada vez mais claro para os pesquisadores que este é um conceito complexo e multifacetado. Muitos autores desenvolveram modelos de qualidade baseados na ideia de descrever qualidade como um conjunto de características ou atributos, organizadas de forma hierárquica. Esse movimento também aconteceu na área de qualidade de software, resultando em múltiplos modelos.

Um marco importante nessa discussão foi o estabelecimento de um modelo padrão de qualidade de software, representado na norma ISO/IEC 9126, que identificou seis características da qualidade de software, cada uma delas com um conjunto de subcaracterísticas.

Com relação a esse padrão, a acurácia, ou seja, a capacidade de o produto de software prover com o grau de precisão necessário resultados ou efeitos corretos ou conforme acordados é uma subcaracterística de:

a) Portabilidade
b) Usabilidade
c) Confiabilidade
d) Eficiência
e) Funcionalidade

12. **(CESGRANRIO / CHESF – 2012)** Dentre os atributos de um software de qualidade, incluem-se:

a) controlabilidade, dependabilidade e eficiência
b) controlabilidade, eficiência e manutenibilidade
c) eficiência, imutabilidade e manutenibilidade
d) eficiência, manutenibilidade e usabilidade
e) imutabilidade, manutenibilidade e usabilidade

GABARITO

1. LETRA E
2. LETRA A
3. LETRA A
4. LETRA B
5. LETRA E
6. LETRA D
7. LETRA B
8. LETRA D
9. LETRA A
10. LETRA D
11. LETRA E
12. 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.