**Questão**

Na modelagem de um sistema que trabalha com impostos a serem pagos: Um empregado recebe R$4.000 de salário livre de impostos. Os próximos R$1.500 são tributados em 10%. E os próximos R$28.000 são tributados em 22%. Qualquer outro valor é tributado em 40%. Para o mais próximo valor inteiro, qual desses grupos de números cai na mesma classe de equivalência?

**a)** R$4.800; R$14.000; R$28.000  
**b)** R$5.200; R$5.500; R$28.000  
**c)** R$28.001; R$32.000; R$35.000  
**d)** R$5.800; R$28.000; R$32.000

**Análise**

Precisamos saber qual alternativa contém valores que caem na mesma classe de equivalência. Portanto precisamos, antes de mais nada, saber quais classes de equivalências temos.

**Resolução**

De acordo com o enunciado da questão temos 4 classes de equivalência, que são:

1. Valores até 4000 = não tributados
2. Valores entre 4001 e 5500 = tributados em 10%
3. Valores entre 5501 e 33500 = tributados em 22%
4. Valores maiores que 33501 = tributados em 40%

Logo os valores da alternativa **D** R$5.800; R$28.000; R$32.000, pertencem a mesma classe de equivalência que é a 3.

**Resposta**

Alternativa: **d) R$5.800; R$28.000; R$32.000**

**Questão**

Considere o seguinte:

Pegar e ler o jornal  
Olhe o que está passando na televisão  
Se tiver um programa que você estiver interesse em assistir, então, veja a TV e  
assista o programa  
Caso contrário  
Continue lendo o jornal  
Se existe uma palavra cruzada no jornal, então tente completar

**a)** CS = 1 e CD = 1  
**b)** CS = 1 e CD = 2  
**c)** CS = 1 e CD = 3  
**d)** CS = 2 e CD = 2  
**e)** CS = 2 e CD = 3

**Análise do código**

Que código?

Esse daqui:

1. Pegar jornal
2. Ler jornal
3. Olhar televisão
4. IF tiver um programa que você estiver interesse em assistir THEN
5. Veja a TV
6. Assista o programa
7. ELSE
8. Continue lendo o jornal
9. IF existe uma palavra cruzada no jornal THEN
10. Tente completar
11. ENDIF
12. ENDIF

Acredito que uma maneira de resolver essa questão é passando o cenário descrito para um pseudocódigo, embora gaste mais tempo. Mas no momento o importante é compreender a questão, leve o tempo que levar.

Analisando o código temos dois IFs, sendo que o primeiro tem um ELSE, e o segundo está associado ao ELSE**,** ou seja, é dependente do primeiro IF.

**Resolução**

**Cobertura de sentença**

Dois testes são necessários: um para passar pelo primeiro IF e outro para passar pelo segundo IF. E passando pelos dois IFs, iremos executar todas as sentenças.

**Cobertura de desvio**  
Já para a cobertura de desvio é preciso 3 testes:

*Teste 1*

Está passando um bom filme e o cidadão vai assistir. (primeiro IF – verdadeiro)

*Teste 2*  
Está passando o Faustão e o cidadão vai continuar a leitura do jornal de domingo. (primeiro IF – falso)  
O cidadão acaba de encontrar uma palavra cruzada e logo tenta completar. (segundo IF – verdadeiro)

*Teste 3*  
Está passando de Volta a Lagoa Azul e o cidadão vai continuar a leitura do jornal. (primeiro IF – falso)  
Não há uma palavra cruzada no jornal. (segundo IF – falso)

**Resposta**

Alternativa: **e) CS = 2 e CD = 3**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Essa é uma questão casca grossa. E a principal razão para eu dizer isso, é que ela quebra uma regra que eu tinha: o número de testes para garantir a cobertura de desvio será sempre maior que o número de testes para garantir a cobertura de sentença.

**Questão**

Dado o seguinte código, qual a alternativa verdadeira:

1. IF A > B THEN
2. C = A – B
3. ELSE
4. C = A + B
5. ENDIF
6. Read D
7. IF C = D Then
8. Print “Error”
9. ENDIF

**a)** 1 teste de cobertura de sentença (comando), 3 para a cobertura de desvio  
**b)** 2 testes de cobertura de sentença (comando), 2 para a cobertura de desvio  
**c)** 2 testes de cobertura de sentença (comando), 3 para a cobertura de desvio  
**d)** 3 testes de cobertura de sentença (comando), 3 para a cobertura de desvio  
**e)** 3 testes de cobertura de sentença (comando), 2 para a cobertura de desvio

**Análise do código**

Temos dois IFs independentes, sendo que o primeiro IF tem um ELSE.

**Resolução**

**Cobertura de sentença**

Com um único teste poderíamos garantir quase toda a cobertura de sentença, por exemplo:

*Teste 1*

A = 20  
B = 10  
C = será 10  
D = 10

Com o teste 1 iremos passar pelo primeiro e segundo IF, porém, não iremos passar pelo ELSE. Portanto precisamos de mais um teste:

*Teste 2*

A = 0  
B = 10  
C = será 10 (o valor de C nem interessa nesse teste)  
D = 9 (o valor de D nem interessa nesse teste)

Com o teste 2 passamos pelo ELSE, porque A é igual a B e nos levará a linha 4, a única pela qual não tínhamos passado.

**Cobertura de desvio**

Podemos usar os mesmos testes feitos na cobertura de sentença. Mas com uma diferença: no teste 2, os valores de C e D nos interessam, pois irão cobrir o resultado falso (a não passagem) do segundo IF.

Portanto, com apenas dois testes também alcançamos a cobertura de desvio.

**Resposta**

Alternativa: **b) 2 testes de cobertura de sentença (comando), 2 para a cobertura de desvio**

**Dica**

Adaptando a minha regra inicial: o número de testes para garantir a cobertura de desvio, **na maioria das vezes**, será maior que o número de testes para garantir a cobertura de sentença.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Retomando o simulado 2. Vamos resolver a questão 13, que é bem simples, comparada as demais apresentadas anteriormente.

**Questão**

Dado o seguinte código:

1. Switch PC on
2. Start “outlook”
3. IF outlook appears THEN
4. Send an email
5. Close Outlook
6. ENDIF

**a)** 1 teste de cobertura de sentença (comando), 1 para a cobertura de desvio  
**b)** 1 teste de cobertura de sentença (comando), 2 para a cobertura de desvio  
**c)** 1 teste de cobertura de sentença (comando), 3 para a cobertura de desvio  
**d)** 2 testes de cobertura de sentença (comando), 2 para a cobertura de desvio  
**e)** 2 testes de cobertura de sentença (comando), 3 para a cobertura de desvio

**Análise do código**

Temos apenas um IF. Ou seja, é mamão com açúcar.

**Resolução**

Para alcançar a **cobertura de sentença** precisamos de apenas um teste, no qual iremos passar pelo IF. Ou seja, o outlook irá aparecer e executaremos as linhas 4 e 5, juntamente com as demais.

Já para alcançar a **cobertura de desvio** precisamos de dois testes, um que passe pelo IF e outro que não passe. Ou seja, num teste o outlook irá aparecer e no outro não.

Não falei que seria fácil.

**Resposta**

Alternativa: **b) 1 teste de cobertura de sentença (comando), 2 para a cobertura de desvio**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Voltando ao simulado 1. Vamos ver a questão de número 14.

**Questão**

Quantos casos de testes são necessários para cobrir todas as possibilidades de  
declarações (caminhos) para o seguinte fragmento de código? Supondo que as duas  
condições são independentes entre elas.  
…………

1. if (Condition 1)
2. then statement 1
3. else statement 2
4. fi
5. if (Condition 2)
6. then statement 3
7. fi

…………  
**a)** 2  
**b)** 3  
**c)** 4  
**d)** Não há como estimar

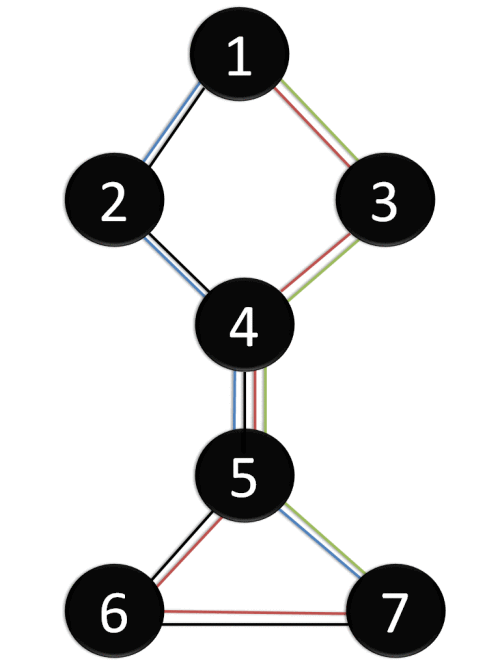
**Análise**

A questão pede o total de testes para cobrir todas as possibilidades de declarações (caminhos), ou seja, pede-se a cobertura de cobertura de caminho.

Quanto ao código, podemos perceber que há dois IFs independentes, como o próprio enunciado já fala.

**Resolução**

A melhor maneira de resolver essa questão é fazendo o fluxo do código para pode visualizar os caminhos existentes:



Olhando a figura acima, podemos visualizar que há 4 caminhos possíveis:

**Azul:** Condition 1 é verdadeira > executa o statement 1 > Condition 2 é falsa > finaliza

**Preto:** Condition 1 é verdadeira > executa o statement 1 > Condition 2 é verdadeira>executa o statement 3 > finaliza

**Vermelho:** Condition 1 é falsa> executa o statement 2 > Condition 2 é verdadeira> executa o statement 3 > finaliza

**Verde:** Condition 1 é falsa > executa o statement 2 > Condition 2 é falsa > finaliza

**Resposta**

Alternativa: **c) 4**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Dois dos assuntos que mais apresentam dificuldade são os de cobertura de sentença (comando) e desvio. E acredito que o motivo é que nós (da área de Teste e Qualidade de Software), na maioria das vezes não somos os responsáveis por usar tais coberturas. E sim os desenvolvedores, já que elas são técnicas baseadas em estrutura.

Irei apresentar a resolução da questão 12 do simulado 2, e em outros posts também abordarei as questões 13, 14 e 15 deste mesmo simulado, que também falam sobre cobertura de sentença (comando) e desvio.

**Questão**

Dado o seguinte código, o que é verdadeiro sobre o número mínimo de casos de  
teste necessários para uma total cobertura de sentença (comando) e desvio:

1. Read P
2. Read Q
3. IF P+Q > 100 THEN
4. Print “Large”
5. ENDIF
6. If P > 50 THEN
7. Print “P Large”
8. ENDIF

**a)** 1 teste de cobertura de sentença (comando), 3 para a cobertura de desvio  
**b)** 1 teste de cobertura de sentença (comando), 2 para a cobertura de desvio  
**c)** 1 teste cobertura de sentença (comando), 1 para a cobertura de desvio  
**d)** 2 testes de cobertura de sentença (comando), 3 para a cobertura de desvio  
**e)** 2 testes de cobertura de sentença (comando), 2 para a cobertura de desvio

**Análise do código**

Temos dois IFs, e um detalhe importante: são dois IFs independentes, ou seja, o resultado do primeiro IF não impacta no segundo IF.

**Resolução**

Antes de ir para resolução propriamente dita, é bom lembrar dos conceitos de cobertura de sentença (comando) e da cobertura de desvio:

**Cobertura de sentença (comando**) = está associada a quantidade de linhas do código que está sendo testada  
**Cobertura de desvio** = está associada a quantidade de desvios que são testados, o que inclui fazer o teste da saída verdadeira e falsa de um desvio (desvio = IF, CASE, SWITCH, WHILE, etc)

Agora vamos para a resolução, usando as duas técnicas:

**Sentença (comando)**

Com um único teste podemos alcançar a cobertura total de sentença, por exemplo:

P = 100  
Q = 1

Iremos passar pelos dois IFs. Logo cobrimos todas as sentenças: 1,2,3,4,5,6,7,8.

**Desvio**

Já para alcançar a cobertura total de desvio precisamos de dois testes: um que passe pelos dois IFs e outro que não passe por eles, por exemplo:

*Teste 1*

P = 100  
Q = 1

Com estes valores de entrada, iremos passar pelo primeiro IF e também pelo segundo.

*Teste 2*

P = 50  
Q = 1

Com estes valores de entrada, não iremos passar pelo primeiro IF e nem pelo segundo.

**Detalhe da questão:** No enunciado da questão pede-se “o número **mínimo** de casos de teste”.

**Resposta**

Alternativa: **b) 1 teste de cobertura de sentença (comando), 2 para a cobertura de desvio**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Nossa! Como está questão meu deu dor de cabeça. Primeiro por culpa minha mesmo, de ter errado na tradução: me esqueci da palavra “válidas” no enunciado, o que acabou tornando a questão bem confusa.

E depois, quando parecia tudo está resolvido, o leitor Antonio Moraes fez uma excelente pergunta no post sobre a [Q1S1](http://qualidadebr.wordpress.com/2009/03/25/resolucao-de-questoes-ctfl-q1s1/). Segue ela abaixo:

[...] de acordo com a correta tradução da questão 5 do simulado 2, a alternativa correta não seria a D. Não entendi o porque do valor 50.000 precisar ser testado.  
Você tem idéia?

Daí resolvi me interessar (como já diria um amigo meu) e entender melhor a questão. E encontrei uma grave inconsistência na questão:

A tradução foi feita com base em um simulado com 5 alternativas (2.PracticeExam1 (English) Q40), que dava como correta a alternativa C, porém a C está assim nesse simulado: **c) 10000, 50000, 9999**

O que está incorreto pois 9999 é um valor inválido e desta maneira a alternativa estaria incorreta. Então pesquisei pela questão original na internet e encontrei a mesma questão ([fonte](http://istqb-exam.blogspot.com/)), com a alternativa C com os valores: 10.000, 50.000, 99.999

A primeira coisa que fiz foi arrumar esse erro na tradução. Porém, novamente cometi mais um erro, não prestei atenção que o simulado que eu me baseei tinha 5 alternativas e uma delas com uma alternativa mais correta do que a alternativa C. A alternativa D.

E após de até ter mudado o simulado. Em uma discusão no grupo [DFTestes](http://br.groups.yahoo.com/group/DFTestes/), onde o pessoal disse que a alternativa correta é a C e não a D. Parei novamente, li e reli a questão original e a traduzida, e percebi que realmente a **C** é a alternativa correta. E um dos motivos para a minha confusão foi que a tradução, ainda não estava tão parecida quanto a questão original, então dei uma melhora na tradução para deixar mais parecida com a original e também entendível em português.

Bem, segue abaixo a questão e a resolução desta questão que gerou tanta discussão e confusão.

**Questão**

O número em um sistema de controle de estoque pode variar entre 10.000 e 99.999 inclusive. Quais das seguintes entradas poderiam ser o resultado da modelagem de teste usando apenas classes de equivalências válidas e limites válidos?

**a)** 1.000, 5.000, 99.999

**b)** 9.999, 50.000, 100.000

**c)** 10.000, 50.000, 99.999

**d)** 10.000, 99.999

**e)** 9.999, 10.000, 50.000, 99.999, 100.000

**Analisando o enunciado da questão**

Pede-se apenas os valores válidos e usando as técnicas de classe de equivalência e de valores limites.

**Resolução**

Seguindo as técnicas propostas:

**Técnica de classe de equivalência** – temos 3 partições:

**Inválida mínima** = valores menores que 10.000;  
**Válida** = valores entre 10.000 e 99.999;  
**Inválida máxima** = valores maiores que 99.999.

**Técnica de valores limites** – temos 4 limites a serem verificados:

**Inválido mínimo** = 9.999;  
**Válido mínimo** = 10.000;  
**Válido máximo** = 99.999;  
**Inválido máximo** = 100.000.

E como a questão pede somente os valores **válidos**, chegamos aos seguintes valores:

**10.000, 50.000 e 99.999**

Daí você pode me perguntar: “O 50.000 não é necessário, pois o 10.000 e o 99.999 já fazem parte da classe de equivalência válida.”

Bem esse foi o pensamento que tive, quando acreditei que a alternativa D estava correta.

Mas, prestando mais atenção percebi que a questão pede para usar a técnica de classe de equivalência e de valores limites, e apenas valores que caracterizem o uso de cada uma delas, lembrando que esses valores precisam ser de classes e limites válidos.

Portanto a alternativa **C** é a correta. Pois:

10.000 = limite mínimo válido  
50.000 = valor da classe de equivalência válida  
99.999 = limite máximo válido

Desta maneira estamos **claramente** usando as duas técnicas.

**Resposta**

Alternativa: **c) 10.000, 50.000, 99.999**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Acredito que a grande dificuldade desta questão é o desconhecimento da complexidade ciclomática de McCabe, eu mesmo não fazia a menor idéia do que era, antes de ver essa questão. E o pior de tudo é que o [Syllabus](http://www.bstqb.org.br/uploads/docs/syllabus_2007br.pdf) não fala dela, e o máximo que podemos encontrar no material sedido pela BSTQB é a sua definição no [ISTQB – Glossário de Termos de Teste](http://www.bstqb.org.br/uploads/docs/glossario_pb_1.4.pdf).

Mas o intuito do post não é explicar a complexidade ciclomática de McCabe, e sim apresentar a resolução da questão 13 do simulado 1. Quem quiser entender melhor a complexidade ciclomática, recomendo os seguintes links abaixo:

A complexidade ciclomática é uma métrica importante que estima a complexidade lógica de um trecho do programa, como um método de uma classe Java.

Dado o grafo de fluxo de controle de um método, o número ou valor da complexidade lógica desse método, também chamado de métrica de McCabe, é obtido pela fórmula:

v(G) = e – n + p

onde…

* **e** representa o número de arestas ou arcos do gráfico,
* **n** o número de nodos ou nós do gráfico e
* **p** o número de componentes ligados

Embora tenha nome pomposo, o número da complexidade ciclomática (CCN) mede algo simples: a quantidade de caminhos lógicos de um código. Por isso, ele é influenciado pela presença de comandos que provocam desvios, como, por exemplo, if, while, case etc, no caso de aplicativos Java.

Podemos inferir do CCN a dificuldade para entender um código e seu grau de manutenibilidade e testabilidade, ou ainda…

Agora vamos para a questão…

Dado o seguinte programa:

**1** IF X < Y

**2** THEN Statement 1;

**3** ELSE IF Y >= Z

**4** THEN Statement 2;

**5** END

A complexidade ciclomática de McCabe é :

**a)** 2

**b)** 3

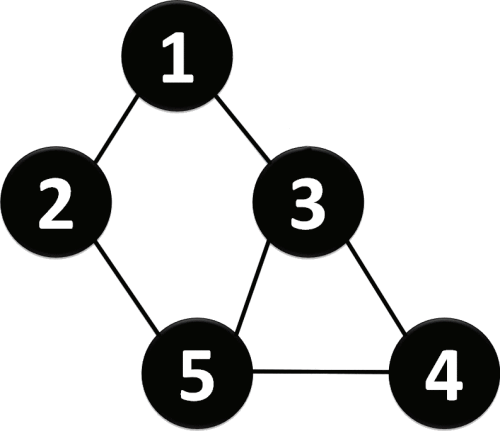
**c)** 4

**d)** 5

**Análise do código**

Temos dois IFs, sendo que o segundo é dependente do primeiro, ou seja, a sua execução depende do resultado do primeiro IF, pois o segundo IF está associado ao ELSE do primeiro IF.

Acredito que essa história de primeiro e segundo IF, pode ter te deixado confuso querido(a) leitor(a). Então, vamos facilitar as coisas, abaixo segue o fluxo do código da questão:



**Resolução**

Temos duas maneiras de resolver essa questão: fazendo o fluxo com os caminhos lógicos possíveis (parecido com a figura acima), ou usando a fórmula da complexidade ciclomática.

Vamos primeiro para a maneira mais difícil, usando a fórmula:

M = E − N + 2P  
M= complexidade ciclomática  
E= número de arestas (linhas/caminhos)  
N= número de nodos  
P= número de componentes conectados

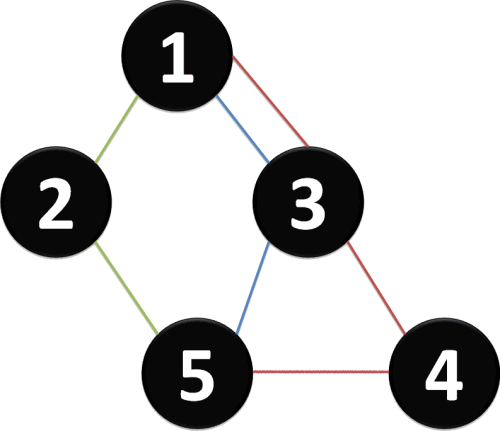
M= 6 – 5 + 2X1

M= 1 + 2

M =3

Portanto já sabemos que a complexidade ciclomática é igual a 3.

Agora vamos resolver essa mesma questão, de uma maneira mais simples e fácil:



Sabendo que a complexidade ciclomática medi o número de caminhos de uma determinada função. Fica claro na figura acima, que o código apresentado tem 3 caminhos lógicos possíveis:

**Verde:** X é menor que Y > executar o Statement 1 > END

**Azul:** X é maior que Y > Y é menos que Z > END

**Vermelho:** X é maior que Y > Y é maior igual que Z > THEN Statement 2 > END

E o motivo por eu achar a segunda maneira mais fácil é simples: eu dificilmente foi lembrar da fórmula da complexidade ciclomática na hora da prova. E como tenho certeza que a pessoa que irá aplicar o exame não irá colocar a fórmula na lousa (como nos tempos de escola), prefiro fazer o fluxo do código, afinal não vou esquecer de como faz círculos e retas, durante a prova.

Logicamente essa é minha opinião, se você acha melhor resolver usando a fórmula, tudo bem.

**Resposta**

Alternativa: **b) 3**

Por hoje é só pessoal. No próximo post irei resolver a questão 5 do simulado 2, a que o Clauriston comentou no post anterior. Até lá!

**Atualização 13 de Novembro de 2009**

Estava hoje dando uma olhada no [artigo](http://classes.cecs.ucf.edu/eel6883/berrios/notes/Paper%204%20(Complexity%20Measure).pdf) publicado pelo próprio McCabe, e vi que ele mesmo diz que é possível calcular a complexidade ciclomática de uma forma bem mais simples (e a Cristina até citou nos comentários).

**Fórmula mágica:**

Quantidade de decisões + 1 = complexidade ciclomática

OBS.: Considerando a análise de um programa que tenha apenas um ponto de entrada e um de saída.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Um campo de entrada (input field) referente ao ano de aniversário aceita valores de 1900 até 2004. Utilizando a análise do valor limite o teste usaria quais valores?**

**a)** 0,1900,2004,2005

**b)** 1900, 2004

**c)** 1899,1900,2004,2005

**d)** 1899, 1900, 1901,2003,2004,2005

**Analisando o enunciado da questão**

O campo aceita valores entre 1900 e 2004. E temos que usar a análise do valor limite para determinar os valores que usaremos no teste.

**Resolução**

A técnica de análise de valor limite faz uso dos valores: mínimo inválido, mínimo válido, máximo válido e máximo inválido. Portanto, teremos que ter quatro valores para o teste. Então já eliminamos duas alternativas: **b** e **d**

Agora que surge a dúvida…

Alternativa **a**

**0 = valor mínimo inválido**

**1900 = valor mínimo válido**

**2004 = valor máximo válido**

**2005 = valor máximo inválido**

Alternativa **c**

**1899 = valor mínimo inválido**

**1900 = valor mínimo válido**

**2004 = valor máximo válido**

**2005 = valor máximo inválido**

Agora você pode está pensando: “Então temos duas alternativas corretas?”

Não, pois a análise do valor limite, como o próprio nome já sugere, está interessada nos valores limites.

“Hmmm, já sei! O **0** não é um valor limite.”

Perfeito! É isso mesmo! O **0** é um valor mínimo inválido, porém está bem abaixo do limite mínimo.

**Resposta**

Alternativa: **c)** 1899,1900,2004,2005

Se você achou alguma questão difícil ou não clara, e gostaria que eu colocasse aqui, só fazer um comentário. Que iremos discutir sobre ela.

Ahh…se a explicação estiver confusa ou não for suficiente para o entendimento, sinta-se à vontade em comentar.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

100% de cobertura de comando cobre 100% de cobertura de desvio? Ou é o contrário?

100% de cobertura de caminho garante 100% de cobertura de LCSAJ?

Se algumas dessas perguntas já passaram pela sua cabeça, garanto que não foi somente pela sua. Pela minha e de várias outras pessoas também.

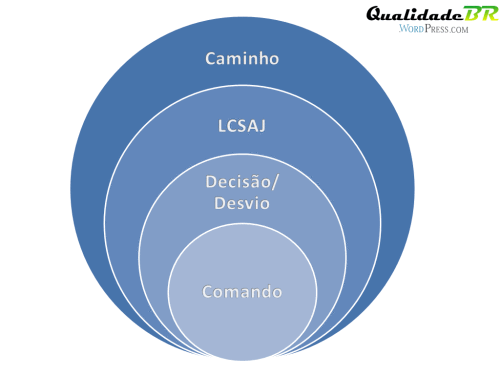
Então o que você acha de solucionar de uma vez todas as dúvidas quanto as coberturas?

*“Se tá brincando, Fabrício*?”

*“Sei, sei e eu também vou te mostrar um esquema para acertar todos os números da loteria”*

Que isso pessoal, é verdade. Eu agarantiu!!!

Eu criei uma representação gráfica de qual cobertura cobre qual, baseada em uma outra que a leitora Renata Eliza (muito obrigado!) me enviou:

[](http://qualidadebr.files.wordpress.com/2009/03/cobertura.gif)

Ficou mais fácil não ficou?

Mas se mesmo assim restaram dúvidas, segue abaixo a listagem de quais coberturas cobrem quais, baseado no material da certificação CTFL, cedido pela [BSTQB](http://www.bstqb.org.br/):

1. 100% da cobertura de decisão ou desvio garante 100% da cobertura de comando, mas não vice-versa;
2. 100% de cobertura de decisão implica em 100% de cobertura de desvio, e vice-versa;
3. 100% de cobertura LCSAJ implica em 100% de cobertura de decisão;
4. 100% de cobertura de caminho implica em 100% de cobertura LCSAJ.

E isso aí pessoal. Espero que eu tenha ajudado.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**O que é o IEEE 1028?**

Como todo padrão elaborado pelo IEEE (lê-se “I três E”), o 1028 é fruto do trabalho voluntário de alguns membros do IEEE. E sendo um padrão, eles nos traz algumas importantes e relevantes informações a respeito de revisão de software. Mas é sempre bom lembrar, que ele deve ser usado com bom senso, pois o contexto sempre prevalece sob o padrão (ou deveria prevalecer).

O IEEE 1028 nos traz cinco tipos de revisão de software, junto com os procedimento necessários para a execuçaõ de cada tipo. Está fora do escopo do padrão questões como: quando uma revisão se faz necessária? como escolher qual tipo de revisão deve ser usado?

Os 5 tipos de revisão abordados são:

* Revisões gerenciais;
* Revisões técnicas;
* Inspeções;
* Walk-throughs;
* Auditórias.

**Os cinco tipos de revisão**

Segue abaixo, a tradução do anexo B do padrão, que contém uma tabela comparativa entre os tipos de revisão (o texto original está numa linguagem meio chata de entender, e não consegui melhorar muito na tradução – se notarem algum erro ou melhoria, por favor me avisem):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Característica** | **Revisão gerencial** | **Revisão técnica** | **Inspeção** | **Walk-through** | **Auditória** |
| Objetivo | Garantir o progresso; recomendar ações corretivas; garantir alocação correta dos recursos | Avaliar a conformidade do estado atual com as especificações e planos; garantir integridade da mudança | Encontrar anomalias; verificar decisões; verificar a qualidade do produto | Encontrar anomalias; examinar alternativas; melhorar o produto; fórum para aprendizado | Avaliação independente de cumprimento com os objetivos de padrões e regulamentos |
| Tomada de decisão | A equipe de gerenciamento traça o curso da ação; decisões são feitas na reunião ou como resultado das recomenda-ções | A equipe de revisão solicita aos gerentes ou a liderança técnica que atuem nas recomendações | A equipe de revisão escolhe as disposições pré-definidas do produto; os defeitos devem ser removidos | A equipe concorda com as mudanças para serem feitas pelo autor | Organização auditada, iniciador, comprador, cliente ou usuário |
| Verificação das mudanças | O líder verifica que itens são fechados; a verificação das mudanças é deixada para outros controles do projeto | O líder verifica que itens são fechados; a verificação das mudanças é deixada para outros controles do projeto | O líder verifica que itens são fechados; a verificação das mudanças é deixada para outros controles do projeto | O líder verifica que itens são fechados; a verificação das mudanças é deixada para outros controles do projeto | Responsabili-dade da organização auditada |
| Tamanho recomendado do grupo | Duas ou mais pessoas | Três ou mais pessoas | Três a seis pessoas | Duas a sete pessoas | Uma a sete pessoas |
| Quem participa | Gerentes, liderença técnica e algumas pessoas de outras áreas | Liderença técnica e algumas pessoas de outras áreas | Pessoas da área com acompanhe-mento documen-tado | Liderença técnica e algumas pessoas de outras áreas | Auditores, organização auditada, pessoal de gerência e técnico |
| Grupo da liderança | Normalmente o gerente responsável | Normalmente o engenheiro líder | Um facilitador treinado | O facilitador ou o autor | O auditor líder |
| Volume de materiais | Moderado para muito, depende dos objetivos da reunião | Moderado para muito, depende dos objetivos da reunião | Relativa-mente baixo | Relativa-mente baixo | Moderado para muito, depende dos objetivos da reunião |