Estratégia
CONCURSOS

Aul

Engenharia de Software p/ SLU-DF (Analista - Informática) Com Videoaulas - Pós Edital

Professor: Diego Carvalho, Equipe Informática e TI

SUMÁRIO	PÁGINA
Apresentação	01
- Conceitos Básicos de Engenharia de Software	08
- Processos de Desenvolvimento	26
- Modelo em Cascata	48
- Modelos Iterativos e Incrementais	72
Lista de Exercícios Comentados	77
Gabarito	92

O PROFESSOR...

PROF. DIEGO CARVALHO

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

ESTRATÉGIA CONCURSOS

Já ministrei mais de 120 cursos de Tecnologia da Informação no Estratégia Concursos, incluindo concursos importantes como: TCU; PF; Senado e Câmara Federal; etc. Nossa equipe já possui vasta experiência em concursos, de forma que você não precise procurar mais nenhum outro material de estudos além dos nossos para fazer sua prova.

ENTRE EM CONTATO:

PROFESSORDIEGOCARVALHO@GMAIL.COM

FACEBOOK.COM/PROFESSORDIEGOCARVALHO

ATENÇÃO

ESSA AULA É APENAS UMA **DEMONSTRAÇÃO** DO MÉTODO, ESTILO E ESCRITA DO PROFESSOR. CASO ADQUIRA O CURSO, O RESTANTE DAS AULAS E EXERCÍCIOS SERÃO DISPONIBILIZADOS.

A AVALIAÇÃO DOS PROFESSORES...



Comentário sobre o curso
O curso está excelente para a preparação para o concurso do TCE-SP e outros que tem abordado os mesmos temas. Bem focado nos temas e apresenta os aspectos gerais de cada um deles. O professor comentou muitos exemplos e exercícios. Nem senti falta de vídeos.
Olá professor! de forma geral gostei do curso. Eis alguns pontos para melhorar: - o fato de limitar o tamanho dos parágrafos em poucas linhas facilita, mas acho que não deveriam ser quebrados no meio do mesmo assunto...algumas vezes me perdi... - a observação a respeito do modelo espiral, no qual alguns autores consideram incremental e outros não, poderia ser reforçado no tópico do modelo espiral, mais para o fim da aula (já não lembrava mais desta informação quando cheguei neste modelo...) Desejo muito sucesso!
Achei o material muito bom, objetivo e com muitos exercícios. São resumões sobre os asssuntos, então só vou saber se a profundida é adequada para concursos depois que fizer a prova, mas no geral me pareceu que cobriu bem os temas. Att, Nádia.
Muito bom o curso.
okjok
DADA A QUANTIDADE DE TEMAS ABORDADOS E EXPLICADOS, ACHEI UM CURSO BEM COMPACTO E PRÁTICO.
Apresentou um número de questões comentadas muito bom e com bons esclarecimentos.
Muito bom mesmo. Muitas questões do CESPE mesmo que o concurso fosse da Vunesp. Será que a FCC, FGV e Cesgranrio não tem questões desses assuntos? As questões CESPE são \"diferentes\"... Não tive problemas com isso porque foco o TCU mas fica a dica. Discursivas sobre BI e big data. Sugestões, por favor.
O curso foi excelente e atendeu perfeitamente o meu plano de estudo. Pois precisava de mobilidade e consegui estudar tanto no smartfone , tablet e notebook principalmente no status Off Line porque nem sempre tinha acesso a internet disponível. A didática do professor foi um ponto forte neste curso : A explicação da teoria com uma linguagem clara, cheias de questões comentadas e bom humor como se eu estivesse em uma aula presencial !! Os assuntos-chaves destacados em vermelho otimizou muito as minhas revisões!
Curso com um conteudo excelente, bem produtivo e rico nos detalhes passados que me ajudaram muito.
Gostei muito da linguagem com certeza compraria outros cursos
Não tenho referencia de outros cursos porque este é o primeiro que estou estudando pela internet mas pra mim foi muito bom. A linguagem não é cansativa e acredito que o que se estuda é somente o necessario. Uma coisa que não entendi é que as vezes havia algumas questões que havia a pergunta e a letra da resposta mas não havia as respostas para escolher. Acho que poderia melhorar isso ou então não entendi o conceito.
O professor tinha uma abordagem informal do assunto que facilitava a memorização.
Melhor professor de TI que já vi na área de concursos!

Comentário sobre o curso
Excelente curso! Excelente material, excelente professor, excelente equipe. Recomendo a todos!
"Excelente" é pouco para esse curso. Eu adquiri outras aulas para me preparar para esse concurso, mas nenhuma outra agregou tanto. Imagino o trabalho que deve ter dado para "compilar" todos esses assuntos. Além disso, colocá-los de forma tão didática em aulas relativamente pequenas. Isso é altíssima qualidade! O conteúdo foi muito bem elaborado! E eu reparei isso porque depois que estudei por ele consegui resolver muitas questões que procurei sobre esses assuntos. Questões que antes pareciam ser de "outro mundo" pra mim. A organização da aula também merece destaque... As figuras, os esquemas, as questões, as notas de rodapé e a evolução do conteúdo estudado de acordo com os itens no edital... Sem falar na didática! Quanto ao professor, sem palavras. Respondeu todas minhas dúvidas e sempre se mostrou bastante receptivo. Bom, não sei como irei me sair na prova domingo, mas devo MUITO da minha preparação ao professor Diego Carvalho. Ele nem imagina o quanto me ajudou. Com certeza recomendo esse curso!
Eu me considero uma pessoa leiga em TI, pois sou formado em Engenharia Mecânica e pos graduado em economia, além de nunca ter trabalhado em departamentos de TI nas empresas onde trabalhei. Por isso, apesar de ter tido dificuldades, achei muito adequada a linguagem adotada pelo professor. Os assuntos são muito complexos para serem ensinados a pessoas com pouca base de conhecimento na área, e ainda assim eu entendo que o curso tenha atingido seu objetivo de permitir a transmissão do conhecimento proposto aos alunos. Como pontos de melhoria, sugeriria: - utilização de videoaulas, pois diversos assuntos poderiam ter seu entendimento facilitado com a utilização de imagens - redistribuição na sequência de assuntos. Ainda que eu entenda que vários assuntos sejam interligados, o que dificulta a escolha sobre "o que ensinar primeiro", acho que determinados assuntos poderiam ter sua ordem invertida, ainda que isto não tenha impedido o entendimento da matéria. Muito obrigado. André.
NULL
Excelente curso, muito didático, material de alta qualidade.
Ok

Comentário sobre o curso
òtima didática do Professor Diego!!
"Sempre podemos melhorar" Está muito bom o curso, o pessoal da Estrategia está de parabéns. abs. Airton.
O curso ajuda demais a estruturar e ordenar o conteudo dos cursos, coisa quase impossível de fazer sozinho. Será o primeiro concurso que irei fazer para avaliar realmente a eficácia do curso. Sempre lemabrando que é muito importante e a exempliação da teoria com exemplos práticas reais e ou fictícios que ajudam a fixar o conteúdo, principalmente para aqueles cursos que tem provas discursivas ou questões abertas. Marco Costa
Aulas muito bem explicadas. Respostas muito rápidas. Conteúdo abrangente, focado e muito bem baseado bibliograficamente. Parabéns. Excelente curso.
Esta foi a matéria que mais gostei dentre as abordadas para o concurso do TRT. Gostei muito da linguagem que o professor utilizou e os exercícios no meio do conteúdo deixam a aula mais dinâmica!
Excelente material. Focado para o concurso e muita questão para fixação.
Melhor didática dos pacotes de TI pro TRT.
É uma pena que a última aula chegou tão perto da prova, mas eu entendo a complicação e é fato que o professor fez um trabalho muito bom. Vou revizar o último PDF amanhã, mas até o momento, de longe, foi o melhor material que eu encontrei sobre os outros assuntos. Uma sugestão de melhoria é o próprio site. O sistema de perguntas e respostas é horrível, feio e de baixíssima usabilidade.
Excelente didática! As apostilas dão vontade de ler até quando o assunto não é dos mais legais. Parabéns pelo curso!
O curso é excelente. Sugiro ao Professor lançar um curso regular de Java tratando todos assuntos (programação, JEE, etc.).
Muito bom o material, apostilas com muito exercícios.

O CURSO...

alinhando expectativas...

O curso que eu proponho abrangerá todo o conteúdo do meu cronograma, entretanto é impossível e inviável esgotar cada ponto do edital em uma aula escrita. Como se ministra Java em uma aula? Teríamos uma aula de 800 páginas e não chegaríamos nem perto de matar todo conteúdo! Imaginem agora cada ponto do Edital.



01

Portanto, vou direcioná-los pelo conteúdo da melhor maneira possível. O nosso foco é ter uma visão geral, mas objetiva do que de fato cai em prova e, não, elucubrações sobre cada tema. Meu foco aqui é te fazer passar! Eu sei como é complicado ler muita coisa (ainda mais de TI) e vocês têm outras disciplinas para estudar. Logo, vou ser simples e objetivo! Tranquilo? ;)



02

Além disso, o cronograma será seguido com a maior fidelidade possível, mas ele não é estático e poderá haver alterações no decorrer do curso. Eventualmente, posso tirar o conteúdo de uma aula e colocar em outra de forma que o estudo de vocês fique mais lógico, coeso e fácil de acompanhar; posso também inverter a ordem das aulas (adiantar uma aula e atrasar outra) – sem prejudicá-los.



03

Ademais, vamos usar questões de diversas bancas. Enfim, confiem em mim: o curso vai ajudar bastante! Qualquer dúvida, é só me chamar! Caso haja alguma reclamação, problema, sugestão, comentários, erros de digitação, etc, podem enviar para o nosso fórum que eu tento responder da maneira mais tempestiva possível. Ainda duvidam que PDF não dá certo com Concursos de TI?



04



CONHEÇA O 6º LUGAR – ISS/SALVADOR

www.youtube.com/watch?v=b1w4H3I6mC4#t=1678



CONHEÇA O 1º LUGAR – TRT/RJ

www.facebook.com/video.php?v=790616534367672



CONHEÇA O 3º LUGAR – ALERJ

www.youtube.com/watch?v=SZSvcy4QM8E



CONHEÇA O 1º LUGAR – DATAPREV

www.estrategiaconcursos.com.br/blog/entrevista-andre-furtado-aprovado-em-1o-lugar-no-concurso-dataprev-para-o-cargo-de-analistaarea-de-tecnologia-da-informacao



CONHEÇA O 2º LUGAR – ISS/SALVADOR

www.youtube.com/watch?v=vmUInIJ-aqQ



CONHEÇA NOSSO GRUPO NO FACEBOOK

www.facebook.com/groups/EstrategiaConcursosdeTI



AS AULAS E AS DICAS...

Parágrafos pequenos Observem que os parágrafos têm, no máximo, cinco linhas. Isso serve para que a leitura não fique cansativa e para que vocês não desanimem no meio do material! Para tal, eu tento dividir as disciplinas de maneira que as aulas fiquem objetivas e pequenas (em termos de teoria), mas extensa (em termos de exercícios).	**Visão Geral** Não se atenham a detalhes antes de entender o básico. *Por que?* Ora, não há nada mais irritante do que ir para uma prova que vai cair, por exemplo, RUP, saber vários detalhes, mas não saber as fases e disciplinas. Portanto, caso estejam iniciando os estudos sobre uma matéria, foquem em saber o básico para depois se especializarem.
Destaques em vermelho Quase todos os parágrafos possuem alguma palavra ou frase destacada em negrito e em vermelho. Isso ocorre por suas razões: primeiro, para enfatizar alguma informação importante; segundo, para facilitar a leitura vertical, i.e., após uma primeira leitura, a segunda pode ser passando apenas pelos pontos em destaque.	**Façam muitos exercícios** Ler várias bibliografias é muito trabalhoso e, geralmente, não vale o custo-benefício. Acredito que o que funciona mesmo é entender o básico, depois fazer muitos exercícios e, eventualmente, caso encontrarem algo que não souberem, pesquisem-no separadamente. Além disso, você vai pegando as "manhas" da banca.
Linguagem natural Essa é uma aula para ser lida, o que por si só já pode ser cansativo. Tentarei colocar a linguagem mais coloquial possível, simulando uma conversa. Caso virem frases ou palavras em itálico, ou é uma palavra estrangeira ou é a simulação de uma conversa com vocês. *Tranquilo?*	**Façam resumos** Essa dica somente serve caso vocês tenham disponibilidade. Caso haja pouco tempo para estudar ou pouco tempo até a prova, não compensa! Se não, façam resumos organizados, pois eles economizarão um bom tempo de estudo em suas próximas provas e sempre que descobrirem novas informações, insiram-nas no resumo.
Diversas figuras Essas aulas estarão em constante evolução, sempre à procura de explicar as matérias de maneira mais compreensível e com novas informações/questões. Para tal, na minha opinião, é fundamental a utilização de figuras, gráficos, painéis, etc. Em minha experiência, é bem mais fácil memorizar a partir de imagens.	**Revisem antes da prova** Não adianta querer estudar coisas novas até o último minuto antes da prova e não revisar o que estudou há um mês. Vocês irão esquecer e irão se irritar na hora da prova por não lembrarem de conceitos simples. Tirem uma semana para revisar seus resumos, decorarem algumas coisas e, certamente, irão mais confiantes para a prova.
Fazer Exercícios Muitos exercícios é o meio pelo qual vocês se situarão. *Como assim, professor?* É na hora de fazer os exercícios que vocês descobrirão se estão bem ou mal e avaliarão se precisam estudar mais ou menos. Para tal, há um quadrinho ao final de cada bloco de exercícios para vocês anotarem a quantidade de questões respondidas corretamente ou incorretamente.	**Simulado Final** Ora, fazer um bloco de questões depois de estudar a teoria é tranquilo. No entanto, lembrem-se que a memória de vocês não é infinita e vocês têm um milhão de outras coisas para estudar e decorar. Portanto, se possível, ao fim do curso faremos um simulado com questões escolhidas que foram comentadas dentro das aulas.



ENGENHARIA DE SOFTWARE

Vamos lá, galera! Apesar de hoje em dia haver milhões de profissionais que mexem com software no mundo inteiro, faz pouco tempo que a Engenharia de Software alcançou o status de profissão reconhecida e de disciplina legítima de engenharia. Pois é, ela ganhou tanta importância que é cobrada até em concursos públicos! *Ok, professor... mas o que é a Engenharia de Software?*

A IEEE define engenharia de software como a aplicação de uma abordagem sistemática, disciplinada e quantificável de desenvolvimento, operação e manutenção de software. Já Friedrich Bauer conceitua como a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe em máquinas reais.

Em suma, é uma disciplina de engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, após sua entrada em produção A meta principal da Engenharia de Software é desenvolver sistemas de software com boa relação custo-benefício. *Bacana?*

Aliás, vamos voltar um pouquinho: *o que seria um software?* Bem, em uma visão restritiva, muitas pessoas costumam associar o termo software aos programas de computador. Software não é apenas o programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente. Vamos prosseguir...

De acordo com Pressman: "*A Engenharia de Software ocorre como consequência de um processo chamado Engenharia de Sistemas. Em vez de se concentrar somente no software, a engenharia de sistemas focaliza diversos elementos, analisando, projetando, e os organizando em um sistema que pode ser um produto, um serviço ou uma tecnologia para transformação da informação ou controle*".

Ademais, nosso renomadíssimo autor afirma que a engenharia de sistemas está preocupada com todos os aspectos do desenvolvimento de sistemas computacionais, incluindo engenharia de hardware, engenharia de software e engenharia de processos. Percebam, então, que a Engenharia de Sistemas está em um contexto com várias outras engenharias. *Entenderam direitinho?*

A Engenharia de Software tem por objetivos a aplicação de teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para a desenvolvimento sistemático de software. Associado ao desenvolvimento, é preciso também aplicar processos, métodos e ferramentas sendo que a pedra fundamental que sustenta a engenharia de software é o foco na qualidade.



Isto envolve planejamento de custos e prazos, montagem da equipe e garantia de qualidade do produto e do processo. Finalmente, a engenharia de software visa a produção da documentação formal do produto, do processo, dos critérios de qualidade e dos manuais de usuários finais. Todos esses aspectos devem ser levados em consideração.

Aliás, nosso outro renomadíssimo autor (Sommerville) afirma que: *"A engenharia de software não está relacionada apenas com os processos técnicos de desenvolvimento de software, mas também com atividades como o gerenciamento de projeto de software e o desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de software"*.

A Engenharia de Software surgiu em meados da década de sessenta como uma tentativa de contornar a crise do software e dar um tratamento de engenharia ao desenvolvimento de software completo. Naquela época, o processo de desenvolvimento era completamente fora de controle e tinha grandes dificuldades em entregar o que era requisitado pelo cliente.

Já na década de oitenta, surgiu a Análise Estruturada e algumas Ferramentas CASE que permitiam automatizar algumas tarefas. Na década de noventa, surgiu a orientação a objetos, linguagens visuais, processo unificado, entre outros. E na última década, surgiram as metodologias ágeis e diversos paradigmas de desenvolvimento.

A Engenharia de Software possui alguns princípios, tais como: Formalidade, em que o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva; Abstração, preocupa-se com a identificação de um determinado fenômeno da realidade, sem se preocupar com detalhes, considerando apenas os aspectos mais relevantes.

Há a Decomposição, em que se divide o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica; Generalização, maneira usada para resolver um problema, de forma genérica, com o intuito de reaproveitar essa solução em outras situações; Flexibilização é o processo que permite que o software possa ser alterado, sem causar problemas para sua execução.

Certa vez, um aluno me perguntou: *professor, como a formalidade pode reduzir inconsistências?* Cenário 1: Estamos na fase de testes de software. O testador afirma que fez todos os testes, você - líder de projeto - acredita, e passa o software ao cliente. Esse tenta utilizar o software e verifica que ele não está funcionando corretamente. *Bacana?*

Cenário 2: Estamos na fase de testes de software. O testador afirma que fez todos os testes e entrega um documento de testes com tudo que foi verificado formalmente. Você - líder de projeto - lê o documento de testes e verifica que não foram feitos testes de carga e testes de segurança. Retorna para o testador e pede para ele refazer os testes.

Feito isso, ele passa o software ao cliente, que fica feliz e satisfeito com tudo funcionando corretamente. *Vocês percebem que essas formalidades evitam aquele "telefone-sem-fio"?* Quanto mais eu seguir o processo, o passo-a-passo, o que foi definido por várias pessoas a partir de suas experiências com vários projetos, mais eu tenho chance de obter êxito na construção do meu software. *Bacana?*

ESQUEMA


Disciplina que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, após sua entrada em produção – passando por aspectos humanos, hardware, etc.
Princípios: Decomposição; Generalização; Flexibilização; Formalidade; Abstração
Eng. de Software
Ferramentas
Métodos
Processo
Foco na qualidade

CAIU na prova!
DIVERSAS BANCAS

1. (CESPE - 2013 - TRT - 10ª REGIÃO (DF e TO) - Analista Judiciário - Tecnologia da Informação) A engenharia de software engloba processos, métodos e ferramentas. Um de seus focos é a produção de software de alta qualidade a custos adequados.

Comentários:

A Engenharia de Software tem por objetivos a aplicação de teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para a desenvolvimento sistemático de software. Associado ao desenvolvimento, é preciso também aplicar processos, métodos e ferramentas sendo que a pedra fundamental que sustenta a engenharia de software é a qualidade.



Basta visualizar a imagem para responder à questão! Ela não menciona foco na qualidade, mas não restringe também somente a processos, métodos e ferramentas.

Gabarito: C

---

2. (FCC - 2012 - TST - Analista Judiciário - Análise de Sistemas) A Engenharia de Software:

a) é uma área da computação que visa abordar de modo sistemático as questões técnicas e não técnicas no projeto, implantação, operação e manutenção no desenvolvimento de um software.

b) consiste em uma disciplina da computação que aborda assuntos relacionados a técnicas para a otimização de algoritmos e elaboração de ambientes de desenvolvimento.

c) trata-se de um ramo da TI que discute os aspectos técnicos e empíricos nos processos de desenvolvimento de sistemas, tal como a definição de artefatos para a modelagem ágil.

d) envolve um conjunto de itens que abordam os aspectos de análise de mercado, concepção e projeto de software, sendo independente da engenharia de um sistema.

e) agrupa as melhores práticas para o concepção, projeto, operação e manutenção de artefatos que suportam a execução de programas de computador, tais como as técnicas de armazenamento e as estruturas em memória principal.

Comentários:

> *De acordo com Pressman: "A Engenharia de Software ocorre como consequência de um processo chamado Engenharia de Sistemas. Em vez de se concentrar somente no software, a engenharia de sistemas focaliza diversos elementos, analisando, projetando, e os organizando em um sistema que pode ser um produto, um serviço ou uma tecnologia para transformação da informação ou controle".*

(a) Perfeito, observem as palavras-chave: modo sistemático; questões técnicas e não técnicas; projeto, implantação, operação e manutenção de desenvolvimento de software. (b) *Técnicas para otimização de algoritmos e elaboração de ambientes de desenvolvimento?* Não, isso não é Engenharia de Software. (c) Pessoal, discordo do gabarito! Certa vez, um aluno me disse que talvez fosse porque aspectos empíricos são mais voltados para metodologias ágeis. Sim, é verdade! No entanto, a engenharia de software trata também de metodologias ágeis. Se alguém encontrar o erro, avise :-] (d) A Análise de Mercado serve mais como uma técnica para Análise de Interfaces, mas pode ser vista como um dos aspectos que envolvem a Engenharia de Software. Pressman afirma que: *"A Análise de Mercado pode ser inestimável na definição de segmentos de mercado e no entendimento de como cada segmento poderia usar o software de modos sutilmente diferentes"*. De todo modo, a questão está errada porque a Engenharia de Software depende da Engenharia de Sistema

(como é mostrado acima). (e) *Suportam a execução?* Não, suportam o desenvolvimento de programas de computador.

Gabarito: A

---

3. (FCC - 2012 - TRT - 6ª Região (PE) - Técnico Judiciário - Tecnologia da Informação) Considere: *é uma disciplina que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação. Seu principal objetivo é fornecer uma estrutura metodológica para a construção de software com alta qualidade.* A definição refere-se:

a) ao ciclo de vida do software.
b) à programação orientada a objetos.
c) à análise de sistemas.
d) à engenharia de requisitos.
e) à engenharia de software.

Comentários:

Em suma, é uma disciplina de engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, após sua entrada em produção. A meta principal da Engenharia de Software é desenvolver sistemas de software com boa relação custo-benefício. Bacana?

Conforme vimos em aula, essa é a pura definição de Engenharia de Software!

Gabarito: E

---

4. (CESPE - 2011 - MEC - Gerente de Projetos) A engenharia de software, disciplina relacionada aos aspectos da produção de software, abrange somente os processos técnicos do desenvolvimento de software.

Comentários:

Aliás, nosso outro renomadíssimo autor (Sommerville) afirma que: "A engenharia de software não está relacionada apenas com os processos técnicos de desenvolvimento de software, mas também com atividades como o gerenciamento de projeto de

software e o desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de software".

Conforme vimos em aula, não são apenas processos técnicos!

Gabarito: E

---

5. **(FGV - 2010 - BADESC - Analista de Sistemas - Desenvolvimento de Sistemas)** De acordo com Pressman, a engenharia de software é baseada em camadas, com foco na qualidade.

   Essas camadas são:

   a) métodos, processo e teste.
   b) ferramentas, métodos e processo.
   c) métodos, construção, teste e implantação.
   d) planejamento, modelagem, construção, validação e implantação.
   e) comunicação, planejamento, modelagem, construção e implantação.

Comentários:



Conforme vimos em aula, bastava visualizar a imagem para responder à questão!

Gabarito: B

---

6. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação)** Os princípios de engenharia de software definem a necessidade de formalidades para reduzir inconsistências e a decomposição para lidar com a complexidade.

Comentários:

A Engenharia de Software possui alguns princípios, tais como: Formalidade, em que o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva; Há a Decomposição, em que se divide o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica;

Conforme vimos em aula, são princípios: Formalidade e Decomposição.

Gabarito: C

---

7. (FCC - 2010 - TRE-AM - Analista Judiciário - Tecnologia da Informação) A Engenharia de Software:

a) não tem como método a abordagem estruturada para o desenvolvimento de software, pois baseia-se exclusivamente nos modelos de software, notações, regras e técnicas de desenvolvimento.

b) se confunde com a Ciência da Computação quando ambas tratam do desenvolvimento de teorias, fundamentações e práticas de desenvolvimento de software.

c) tendo como foco apenas o tratamento dos aspectos de construção de software, subsidia a Engenharia de Sistemas no tratamento dos sistemas baseados em computadores, incluindo hardware e software.

d) tem como foco principal estabelecer uma abordagem sistemática de desenvolvimento, através de ferramentas e técnicas apropriadas, dependendo do problema a ser abordado, considerando restrições e recursos disponíveis.

e) segue princípios, tais como, o da Abstração, que identifica os aspectos importantes sem ignorar os detalhes e o da Composição, que agrupa as atividades em um único processo para distribuição aos especialistas.

Comentários:

A IEEE define engenharia de software como a aplicação de uma abordagem sistemática, disciplinada e quantificável de desenvolvimento, operação e manutenção de software. Já Friedrich Bauer conceitua como a criação e a utilização de sólidos

princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe em máquinas reais.

(a) Pelo contrário, ela se baseia em uma abordagem estruturada e sistemática!

A Engenharia de Software tem por objetivos a aplicação de teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para a desenvolvimento sistemático de software. Associado ao desenvolvimento, é preciso também aplicar processos, métodos e ferramentas sendo que a pedra fundamental que sustenta a engenharia de software é a qualidade.

(b) Na verdade, Engenharia de Software é uma disciplina da Ciência da Computação.

(c) *Apenas o tratamento dos aspectos de construção de software? Só construção?* Não!

(d) Perfeito, é exatamente isso!

Há a Decomposição, em que se divide o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica; Generalização, maneira usada para resolver um problema, de forma genérica, com o intuito de reaproveitar essa solução em outras situações; Flexibilização é o processo que permite que o software possa ser alterado, sem causar problemas para sua execução.

A Engenharia de Software possui alguns princípios, tais como: Formalidade, em que o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva; Abstração, preocupa-se com a identificação de um determinado fenômeno da realidade, sem se preocupar com detalhes, considerando apenas os aspectos mais relevantes.

(e) *Composição?* Não, Decomposição! Divide-se o problema em partes para que cada uma possa ser resolvida de uma forma mais específica. Além disso, a abstração ignora detalhes!

Gabarito: D

---

8. **(FCC - 2011 - INFRAERO - Analista de Sistemas - Gestão de TI)** Em relação à Engenharia de Software, é INCORRETO afirmar:

a) O design de software, ao descrever os diversos aspectos que estarão presentes no sistema quando construído, permite que se faça a avaliação prévia para garantir que ele alcance os objetivos propostos pelos interessados.

b) A representação de um design de software mais simples para representar apenas as suas características essenciais busca atender ao princípio da abstração.

c) Iniciar a entrevista para obtenção dos requisitos de software com perguntas mais genéricas e finalizar com perguntas mais específicas sobre o sistema é o que caracteriza a técnica de entrevista estruturada em funil.

d) No contexto de levantamento de requisitos, funcionalidade é um dos aspectos que deve ser levado em conta na abordagem dos requisitos funcionais.

e) A representação é a linguagem do design, cujo único propósito é descrever um sistema de software que seja possível construir.

Comentários:

Galera, descrever um sistema de software que seja possível construir não é o único, mas um dos objetivos da representação. Ela auxilia a comunicação entre as partes interessadas e serve também como documentação.

Gabarito: E

---

9. **(FCC – 2009 – AFR/SP - Analista de Sistemas)** A engenharia de software está inserida no contexto:

a) das engenharias de sistemas, de processo e de produto.
b) da engenharia de sistemas, apenas.
c) das engenharias de processo e de produto, apenas.
d) das engenharias de sistemas e de processo, apenas.
e) das engenharias de sistemas e de produto, apenas.

Comentários:

> *Ademais, nosso renomadíssimo autor afirma que a engenharia de sistemas está preocupada com todos os aspectos do desenvolvimento de sistemas computacionais, incluindo engenharia de hardware, engenharia de software e engenharia de*

processos. Percebam, então, que a Engenharia de Sistemas está em um contexto com várias outras engenharias. Entenderam direitinho?

Observem que a Engenharia de Software está em um contexto em que há também Engenharia de Sistemas, de Processo e de Produto.

Gabarito: A

---

10. (CESPE – 2015 – STJ – Analista de Sistemas) Embora os engenheiros de software geralmente utilizem uma abordagem sistemática, a abordagem criativa e menos formal pode ser eficiente em algumas circunstâncias, como, por exemplo, para o desenvolvimento de sistemas web, que requerem uma mistura de habilidades de software e de projeto.

Comentários:

Galera, basta pensar nas metodologias ágeis! Elas são menos formais e são mais adequadas em determinadas circunstâncias. Notem que isso não vai de encontro ao princípio da formalidade, na medida em que a questão deixa claro que se trata de uma abordagem "*menos formal*" e, não, "*informal*". Além disso, isso consta também do livro do Sommerville:

"No entanto, engenharia tem tudo a ver com selecionar o método mais adequado para um conjunto de circunstâncias, então uma abordagem mais criativa e menos formal pode ser eficiente em algumas circunstâncias. Desenvolvimento menos formal particularmente adequado para o desenvolvimento de sistemas Web, que requerem uma mistura de habilidades de software e de projeto".

Gabarito: C

---

11. (CESPE – 2015 – STJ – Analista de Sistemas) O foco da engenharia de software inclui especificação do sistema, desenvolvimento de hardware, elaboração do projeto de componentes de hardware e software, definição dos processos e implantação do sistema.

Comentários:

Conforme vimos em aula, pode até tratar eventualmente de alguns aspectos de hardware; mas desenvolvimento de hardware, não.

Gabarito: E

12. (FUNIVERSA – 2009 – IPHAN – Analista de Sistemas) A Engenharia de Software resume-se em um conjunto de técnicas utilizadas para o desenvolvimento e manutenção de sistemas computadorizados, visando produzir e manter softwares de forma padronizada e com qualidade. Ela obedece a alguns princípios como (1) Formalidade, (2) Abstração, (3) Decomposição, (4) Generalização e (5) Flexibilização. Assinale a alternativa que apresenta conceito correto sobre os princípios da Engenharia de Software.

a) A flexibilização é o processo que permite que o software possa ser alterado, sem causar problemas para sua execução.

b) A formalidade é a maneira usada para resolver um problema, de forma genérica, com o intuito de poder reaproveitar essa solução em outras situações semelhantes.

c) A generalização preocupa-se com a identificação de um determinado fenômeno da realidade, sem se preocupar com detalhes, considerando apenas os aspectos mais relevantes.

d) Pelo princípio da decomposição, o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva.

e) A abstração é a técnica de se dividir o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica.

Comentários:

A Engenharia de Software possui alguns princípios, tais como: Formalidade, em que o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva; Abstração, preocupa-se com a identificação de um determinado fenômeno da realidade, sem se preocupar com detalhes, considerando apenas os aspectos mais relevantes.

Há a Decomposição, em que se divide o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica; Generalização, maneira usada para resolver um problema, de forma genérica, com o intuito de reaproveitar essa solução em outras situações; Flexibilização é o processo que permite que o software possa ser alterado, sem causar problemas para sua execução.

Conforme vimos em aula, a flexibilização é o processo que permite que o software possa ser alterado, sem causar problemas para sua execução.

Gabarito: A

---

13. (CESPE – 2013 – TCE/RO – Analista de Sistemas) Engenharia de software não está relacionada somente aos processos técnicos de desenvolvimento de softwares, mas também a atividades como gerenciamento de projeto e desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de softwares.

Comentários:

> *Aliás, nosso outro renomadíssimo autor (Sommerville) afirma que: "A engenharia de software não está relacionada apenas com os processos técnicos de desenvolvimento de software, mas também com atividades como o gerenciamento de projeto de software e o desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de software".*

Conforme vimos em aula, a questão está perfeita!

Gabarito: C

---

14. (CESPE – 2010 – DETRAN/ES – Analista de Sistemas) Segundo princípio da engenharia de software, os vários artefatos produzidos ao longo do seu ciclo de vida apresentam, de forma geral, nível de abstração cada vez menor.

Comentários:

Galera, vamos pensar um pouco! A abstração é a subtração de detalhes – focar-se no que é mais relevante e ignorar detalhes menos relevantes. No início do ciclo de vida, os artefatos são bastante abstratos. Temos, por exemplo, uma modelagem do negócio, um documento de requisitos funcionais, a abstração vai diminuindo e temos a modelagem visual de análise, depois a modelagem do projeto, até chegar ao código-fonte e ao executável do software. Nesse ponto, temos o menor nível de abstração (com muitos detalhes!). *Compreenderam?*

Gabarito: C

---

15. (CESPE – 2010 – TRE/BA – Analista de Sistemas) Entre os desafios enfrentados pela engenharia de software estão lidar com sistemas legados, atender à crescente diversidade e atender às exigências quanto a prazos de entrega reduzidos.

Comentários:

Bem, essa questão é bastante intuitiva. De fato, a engenharia de software tem que lidar com sistemas legados, atender à crescente diversidade e atender às exigências quanto aos (curtos) prazos de entrega.

Gabarito: C

---

16. (FCC – 2010 – DPE/SP – Analista de Sistemas) A Engenharia de Software

I. não visa o desenvolvimento de teorias e fundamentações, preocupando-se unicamente com as práticas de desenvolvimento de software.

II. tem como foco o tratamento dos aspectos de desenvolvimento de software, abstraindo-se dos sistemas baseados em computadores, incluindo hardware e software.

III. tem como métodos as abordagens estruturadas para o desenvolvimento de software que incluem os modelos de software, notações, regras e maneiras de desenvolvimento.

IV. segue princípios, tais como, o da Abstração, que identifica os aspectos importantes sem ignorar os detalhes e o da Composição, que agrupa as atividades em um único processo para distribuição aos especialistas.

É correto o que se afirma em:

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

Comentários:

(I) Errado, Sommerville diz: "*Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software*". No entanto, ele não diz que a engenharia de software se preocupa unicamente com as práticas de desenvolvimento de software.

(II) Errado, Pressman diz: "*System engineering is concerned with all aspects of computer-based systems development including hardware, software, and process engineering*" – a questão trata da Engenharia de Sistemas.

(III) Correto, de fato ela tem como métodos as abordagens estruturadas para o desenvolvimento de software que incluem os modelos de software, notações, regras e maneiras de desenvolvimento.

(IV) Errado, o princípio da abstração ignora os detalhes; e o princípio da composição não existe – o que existe é o princípio da decomposição. E ele divide o problema em partes menores.

Em suma, nenhuma das opções nos atende! *Vocês sabem qual opção a banca marcou como correta?* A Letra D!!! *E ela voltou atrás com os recursos?* Não!!! Pois é, galera! Acostumem-se com isso :(

Gabarito: D

---

17. (CESPE – 2013 – TCE/RO – Analista de Sistemas) Assim como a Engenharia de Software, existe também na área de informática a chamada Ciência da Computação. Assinale a alternativa que melhor apresenta a diferença entre Engenharia de Software e Ciência da Computação.

a) A Ciência da Computação tem como objetivo o desenvolvimento de teorias e fundamentações. Já a Engenharia de Software se preocupa com as práticas de desenvolvimento de software.

b) A Engenharia de Software trata da criação dos sistemas de computação (softwares) enquanto a Ciência da Computação está ligada ao desenvolvimento e criação de componentes de hardware.

c) A Engenharia de Software trata dos sistemas com base em computadores, que inclui hardware e software, e a Ciência da Computação trata apenas dos aspectos de desenvolvimento de sistemas.

d) A Ciência da Computação trata dos sistemas com base em computadores, que inclui hardware e software, e a Engenharia de Software trata apenas dos aspectos de desenvolvimento de sistemas.

e) A Ciência da Computação destina-se ao estudo e solução para problemas genéricos das áreas de rede e banco de dados e a Engenharia de Software restringe- se ao desenvolvimento de sistemas.

Comentários:

A Engenharia de Software tem por objetivos a aplicação de teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para a desenvolvimento sistemático de software. Associado ao desenvolvimento, é preciso também aplicar processos, métodos e ferramentas sendo que a pedra fundamental que sustenta a engenharia de software é a qualidade.

Essa questão parece contradizer o que diz Sommerville, mas observem que não! A Engenharia de Software coloca em prática a teoria e fundamentação trazida pela Ciência da Computação. A Engenharia de Software aplica a teoria, mas - grosso modo - ela não tem a finalidade principal de elaborá-la; a finalidade principal é de colocá-la em prática. Já a Ciência da Computação é exatamente o contrário. Enfim, a primeira opção foi retirada literalmente do Sommerville (Pág. 5, 8ª Ed.).

Gabarito: A

---

18. (CESPE – 2009 – ANAC – Analista de Sistemas) O termo engenharia pretende indicar que o desenvolvimento de software submete-se a leis similares às que governam a manufatura de produtos industriais em engenharias tradicionais, pois ambos são metodológicos.

Comentários:

Perfeito! A Engenharia busca os princípios mais consolidados de outras engenharias mais tradicionais – engenharia mecânica, civil, elétrica, etc.

Gabarito: C

---

19. (CESPE - 2016 – TCE/PR – Analista de Sistemas – B) A engenharia de software refere-se ao estudo das teorias e fundamentos da computação, ficando o desenvolvimento de software a cargo da ciência da computação.

Comentários:

A Engenharia de Software tem por objetivos a aplicação de teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para a desenvolvimento sistemático de software. Associado ao desenvolvimento, é preciso também aplicar processos, métodos e ferramentas sendo que a pedra fundamental que sustenta a engenharia de software é a qualidade.

Conforme vimos em aula, não se trata do estudo – trata-se da aplicação. Além disso, o desenvolvimento também fica a cargo da engenharia de software. Em geral, a ciência da computação trata da teoria e a engenharia de software trata da prática.

Gabarito: E

---

20. (CESPE - 2016 – TCE/PR – Analista de Sistemas – E) O conceito de software se restringe ao desenvolvimento do código em determinada linguagem e seu armazenamento em arquivos.

Comentários:

Aliás, vamos voltar um pouquinho: o que seria um software? Bem, em uma visão restritiva, muitas pessoas costumam associar o termo software aos programas de computador. Software não é apenas o programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente. Vamos prosseguir...

Conforme vimos em aula, o software não é apenas o programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente.

Gabarito: E

ACERTEI	ERREI

PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

Vamos começar falando sobre ciclo de vida. Esse termo surgiu lá no contexto da biologia. Ele trata do ciclo de vida de um ser vivo, ou seja, trata do conjunto de transformações pelas quais passam os indivíduos de uma espécie durante sua vida. *Vocês se lembram lá no ensino fundamental quando a gente aprende que um ser vivo nasce, cresce, reproduz, envelhece e morre?* Pois é!



Vejam a imagem acima! Nós temos um bebê, que depois se torna uma criança, depois um adolescente, depois um jovem, depois um senhor, depois um velho, depois um ancião e depois morre! Logo, o ciclo de vida de um ser vivo trata das fases pelas quais um ser vivo passa desde seu nascimento até a sua morte. E esse conceito pode ser aplicado a diversos outros contextos.

Exemplos: ciclo de vida de uma organização, ciclo de vida de sistemas, ciclo de vida de produtos, ciclo de vida de projetos, entre vários outros. *E como esse conceito se dá em outros contextos?* Exatamente da mesma forma! Logo, o ciclo de vida de um produto trata da história completa de um produto através de suas fases (Ex: Introdução, Crescimento, Maturidade e Declínio).

Existe também o ciclo de vida de um projeto, que trata do conjunto de fases que compõem um projeto (Ex: Iniciação, Planejamento, Execução, Controle e Encerramento). *O que todos esses ciclos de vida têm em comum?* Eles sempre tratam das fases pelas quais algo passa desde o seu início até o seu fim. Então, é isso que vocês têm que decorar para qualquer contexto de ciclo de vida.

Na Engenharia de Software, esse termo é geralmente aplicado a sistemas de software artificiais com o significado de mudanças que acontecem na vida de um produto de software. O ciclo de vida trata das fases identificadas entre o nascimento e a morte de um software. *Vocês viram?* É exatamente a mesma coisa – são as fases ou atividades pelas quais passa um software durante sua vida.

Uma definição interessante de ciclo de vida de software afirma que se trata das fases de um produto de software que vão desde quando ele é concebido até quando ele não está mais disponível para uso. Já Steve McConnell afirma que um modelo de ciclo de vida de software é uma representação que descreve todas as atividades que tratam da criação de um produto de software.

Portanto, nós podemos concluir que um ciclo de vida de software se refere às fases pelas quais um sistema de software atravessa desde sua concepção até sua retirada de produção. Bem, entender esse conceito não é o problema, esse é um conceito muito simples, convenhamos! O problema é que existem dezenas de fases diferentes para os ciclos de vida de acordo com cada autor.

Como assim, professor? Infelizmente, não há um consenso entre os autores. Cada um que lança um livro decide criar o ciclo de vida com as fases que ele acha mais corretas e isso acaba prejudicando nosso estudo. Na UnB, eu tive um professor de Engenharia de Software chamado Fernando Albuquerque que já tinha escrito livros sobre esse assunto e ele tinha o seu próprio ciclo de vida.

Agora imaginem a quantidade de autores de livros de Engenharia de Software lançados nas últimas três ou quatro décadas. Além disso, acontece de as vezes o próprio autor mudar seu entendimento sobre as fases. Se vocês olharem a quarta edição do livro do Pressman, ele tem um conjunto de fases; se vocês olharem da sexta edição para frente, ele define um novo conjunto de fases. Mudou de ideia!

Então, eu já vi mais vários ciclos de vida diferentes em provas! *Qual a solução para esse problema, professor?* Galera, vocês sempre têm que pensar no custo/benefício. *Vale a pena decorar todos?* Na minha opinião, nenhum pouco! Isso não é algo que cai muito. Guardem o espaço que vocês têm sobrando na cabeça para memorizar coisas que realmente caem.

Portanto, percebam que há autores que descrevem as fases do ciclo de vida de software de maneira mais ampla e abstrata (com poucas e grandes fases) e autores que o fazem de maneira mais diminuta e detalhada (com muitas e pequenas fases).

Vamos começar a ver alguns ciclos de vida que existem por aí para que vocês visualizem isso com maior clareza!





Vamos lá! Do lado esquerdo, temos um ciclo de vida de software só com quatro fases bem genéricas: Definição do Software, Desenvolvimento do Software, Operação do Software e Retirada do Software. Eu só vi cair esse ciclo de vida uma vez, mas ele é útil para que vocês visualizem um ciclo de vida com poucas fases. Observem que ele trata desde o início até a aposentadoria do software.

Do lado direito, eu coloquei um ciclo de vida um pouco mais detalhado. Vejam que ele destrincha mais a fase de desenvolvimento, separando em análise, projeto, implementação, testes, etc. *Por que eu coloquei esse ciclo de vida?* Porque alguns autores afirmam que, em tese, todo ciclo de vida deveria contemplar todas essas atividades ou fases. Vamos ver isso com mais detalhes depois...





Em verde, nós temos um ciclo de vida de software de acordo com nosso querido autor: Ian Sommerville. Ele contém quatro fases: Especificação, Desenvolvimento, Validação e Evolução. Sendo que, atenção aqui!, o desenvolvimento pode ser dividido em Projeto e Implementação. *Ok?* Esse é um ciclo de vida de software que já cai com uma maior frequência.

Por fim, em laranja, nós temos um ciclo de vida de software de acordo com nosso outro querido autor: Roger Pressman. Ele contém cinco fases: comunicação, planejamento, modelagem, construção e implantação. Esse não cai tanto, mas é importante saber também. *Tudo bem até aqui?* Então vejam que não é tão complicado! Esses são os quatro ciclos de vida de software mais comuns em prova...

Recaptulando: primeiro, nós vimos o que é um Ciclo de Vida! Depois nós vimos o que é um Ciclo de Vida de Software. E, agora, nós vamos ver um terceiro conceito, que é o conceito de Modelo de Ciclo de Vida de Software. *Por que, professor?* Porque Ciclo de Vida de Software é diferente de Modelo de Ciclo de Vida de Software. *E o que é um modelo de ciclo de vida de software?*

Bem, é um modelo que apresenta não só as fases do ciclo de vida de software, mas também a forma como essas fases se relacionam. *Vocês entenderam isso bem?* O modelo contém as fases que nós acabamos de ver, mas ele também nos diz como essas fases se relacionam! *Vocês querem um exemplo?* Existe um modelo de ciclo de vida chamado Modelo em Cascata.

Esse modelo foi pioneiro, foi o primeiro a aparecer na Engenharia de Software e nós vamos vê-lo em mais detalhes em outro momento. O importante sobre o Modelo em Cascata é a forma como as fases se relacionam. Nesse modelo, uma fase somente se inicia após o término completo da fase anterior (exceto a primeira, evidentemente). Bem simples, não é?

Logo, cada Modelo de Ciclo de Vida de Software contém suas respectivas fases, mas também contém como essas fases se relacionam. Vejam os conceitos abaixo:

Ciclo de Vida

O Ciclo de Vida trata das fases pelas quais alguma coisa passa desde o seu início até o seu fim.

Ciclo de Vida de Software

O Ciclo de Vida de Software trata das fases pelas quais um software passa desde o seu início até o seu fim.

Modelo de Ciclo de Vida de Software

O Modelo de Ciclo de Vida de Software trata das fases pelas quais um software passa desde o seu início até o seu fim e como essas fases se relacionam (processo).

E o que seria um processo de software? Então, um processo de software pode ser visto como o conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software. Como o Modelo de Ciclo de Vida nos diz como as fases do ciclo de vida se relacionam, nós podemos – em termos de prova – considerar Modelo de Ciclo de Vida como sinônimo de Modelo de Processo!

Processo de Software

Trata-se de um conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software.

Modelo de Processo de Software

Trata-se exatamente do mesmo conceito de Modelo de Ciclo de Vida. Ele é uma representação abstrata de um processo de software.

Um processo de software não pode ser definido de forma universal. Para ser eficaz e conduzir à construção de produtos de boa qualidade, um processo deve ser adequado às especificidades do projeto em questão. Deste modo, processos devem

ser definidos caso a caso, considerando-se diversos aspectos específicos do projeto em questão, tais como:

- Características da aplicação (domínio do problema, tamanho do software, tipo e complexidade, etc);
- Tecnologia a ser adotada na sua construção (paradigma de desenvolvimento, linguagem de programação, mecanismo de persistência, etc);
- Organização onde o produto será desenvolvido e a equipe de desenvolvimento alocada (recursos humanos).

Há vários aspectos a serem considerados na definição de um processo de software. No centro da arquitetura de um processo de desenvolvimento estão as atividades-chave desse processo: análise e especificação de requisitos, projeto, implementação e testes, que são a base sobre a qual o processo de desenvolvimento deve ser construído. *Entendido?*

No entanto, a definição de um processo envolve a escolha de um modelo de ciclo de vida (ou modelo de processo), o detalhamento (decomposição) de suas macro-atividades, a escolha de métodos, técnicas e roteiros (procedimentos) para a sua realização e a definição de recursos e artefatos necessários e produzidos – é bastante coisa!

Podemos dizer que se trata da representação abstrata de um esqueleto de processo, incluindo tipicamente algumas atividades principais, a ordem de precedência entre elas e, opcionalmente, artefatos requeridos e produzidos. Em geral, um modelo de processo descreve uma filosofia de organização de atividades, estruturando-as em fases e definindo como essas fases estão relacionadas.

A escolha de um modelo de ciclo de vida (para concursos, sinônimo de modelo de processo) é o ponto de partida para a definição de um processo de desenvolvimento de software. Um modelo de ciclo de vida, geralmente, organiza as macro-atividades básicas do processo, estabelecendo precedência e dependência entre as mesmas. *Tudo certo?*

Conforme eu disse anteriormente, alguns autores afirmam que os modelos de ciclo de vida básicos, de maneira geral, contemplam pelo menos as fases de: Planejamento; Análise e Especificação de Requisitos; Projeto; Implementação;

Testes; Entrega e Implantação; Operação; e Manutenção. Abaixo eu trago uma descrição genérica sobre cada uma dessas fases.

FASES	DESCRIÇÃO
PLANEJAMENTO	O objetivo do planejamento de projeto é fornecer uma estrutura que possibilite ao gerente fazer estimativas razoáveis de recursos, custos e prazos. Uma vez estabelecido o escopo de software, com os requisitos esboçados, uma proposta de desenvolvimento deve ser elaborada, isto é, um plano de projeto deve ser elaborado configurando o processo a ser utilizado no desenvolvimento de software. À medida que o projeto progride, o planejamento deve ser detalhado e atualizado regularmente. Pelo menos ao final de cada uma das fases do desenvolvimento (análise e especificação de requisitos, projeto, implementação e testes), o planejamento como um todo deve ser revisto e o planejamento da etapa seguinte deve ser detalhado. O planejamento e o acompanhamento do progresso fazem parte do processo de gerência de projeto.
ANÁLISE E ESPECIFICAÇÃO DE REQUISITOS	Nesta fase, o processo de levantamento de requisitos é intensificado. O escopo deve ser refinado e os requisitos mais bem definidos. Para entender a natureza do software a ser construído, o engenheiro de software tem de compreender o domínio do problema, bem como a funcionalidade e o comportamento esperados. Uma vez capturados os requisitos do sistema a ser desenvolvido, estes devem ser modelados, avaliados e documentados. Uma parte vital desta fase é a construção de um modelo descrevendo o que o software tem de fazer (e não como fazê-lo).
PROJETO	Esta fase é responsável por incorporar requisitos tecnológicos aos requisitos essenciais do sistema, modelados na fase anterior e, portanto, requer que a plataforma de implementação seja conhecida. Basicamente, envolve duas grandes etapas: projeto da arquitetura do sistema e projeto detalhado. O objetivo da primeira etapa é definir a arquitetura geral do software, tendo por base o modelo construído na fase de análise de requisitos. Essa arquitetura deve descrever a estrutura de nível mais alto da aplicação e identificar seus principais componentes. O propósito do projeto detalhado é detalhar o projeto do software para cada componente identificado na etapa anterior. Os componentes de software devem ser sucessivamente refinados em níveis maiores de detalhamento, até que possam ser codificados e testados.
IMPLEMENTAÇÃO	O projeto deve ser traduzido para uma forma passível de execução pela máquina. A fase de implementação realiza esta tarefa, isto é, cada unidade de software do projeto detalhado é implementada.

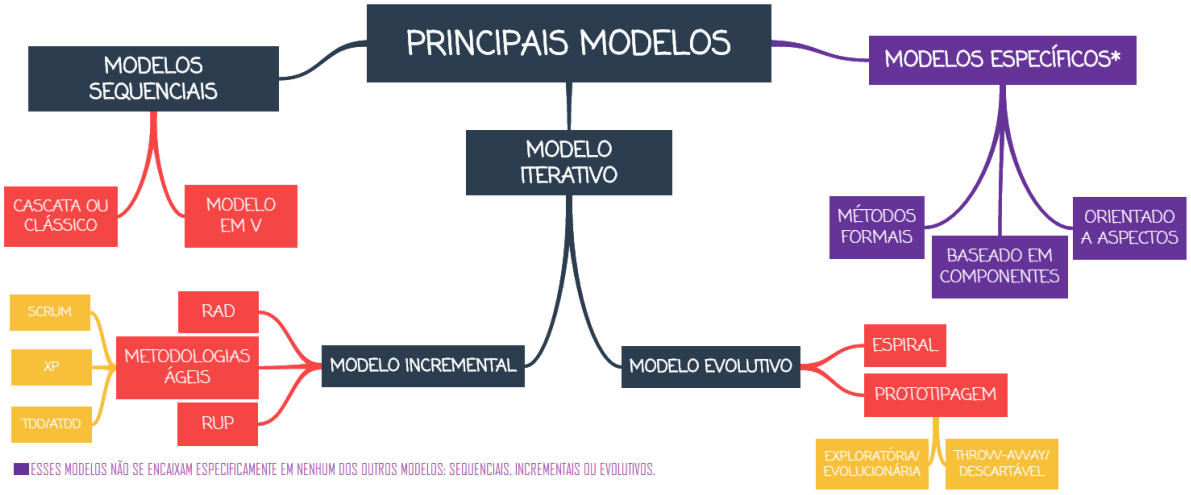
TESTES	Inclui diversos níveis de testes, a saber, teste de unidade, teste de integração e teste de sistema. Inicialmente, cada unidade de software implementada deve ser testada e os resultados documentados. A seguir, os diversos componentes devem ser integrados sucessivamente até se obter o sistema. Finalmente, o sistema como um todo deve ser testado.
ENTREGA E IMPLANTAÇÃO	Uma vez testado, o software deve ser colocado em produção. Para tal, contudo, é necessário treinar os usuários, configurar o ambiente de produção e, muitas vezes, converter bases de dados. O propósito desta fase é estabelecer que o software satisfaz os requisitos dos usuários. Isto é feito instalando o software e conduzindo testes de aceitação. Quando o software tiver demonstrado prover as capacidades requeridas, ele pode ser aceito e a operação iniciada.
OPERAÇÃO	Nesta fase, o software é utilizado pelos usuários no ambiente de produção.
MANUTENÇÃO	Indubitavelmente, o software sofrerá mudanças após ter sido entregue para o usuário. Alterações ocorrerão porque erros foram encontrados, porque o software precisa ser adaptado para acomodar mudanças em seu ambiente externo, ou porque o cliente necessita de funcionalidade adicional ou aumento de desempenho. Muitas vezes, dependendo do tipo e porte da manutenção necessária, essa fase pode requerer a definição de um novo processo, onde cada uma das fases precedentes é reaplicada no contexto de um software existente ao invés de um novo.

Existem outras fases em outros modelos, tais como: análise, responsável por modelar o problema (diferente do projeto, responsável por modelar a solução do problema); homologação, responsável pela aceitação pela parte interessada do produto; gerência de configuração, responsável pela estruturação sistemática dos produtos, artefatos, documentos, modelos, etc.

Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software.

Também de acordo nosso autor, um modelo de processo de software é uma descrição simplificada desse processo de software que apresenta uma visão dele. Os modelos de processo incluem as atividades, que fazem parte do processo de software, e eventualmente os produtos de software e os papéis das pessoas envolvidas na engenharia de software. E não para por aí...

Ele ainda afirma que a maioria dos processos de software é baseada em três modelos gerais: modelo em cascata; desenvolvimento iterativo e engenharia de software baseada em componentes. Isso entra em contradição com o que dizem outros autores, i.e., os principais modelos podem ser agrupados em três categorias: modelos sequenciais, modelos incrementais e modelos evolutivo.



Existem mais um conceito importante nessa aula! É o conceito de Metodologia de Desenvolvimento de Software (também chamada de Processo de Desenvolvimento de Software). *O que é isso, professor?* É uma caracterização prescritiva ou descritiva de como um produto de software deve ser desenvolvido, i.e., define o quê, como e quando fazer algo – um exemplo clássico é o RUP!

Bem, como o nome indica, os modelos sequenciais organizam o processo em uma sequência linear de fases. O principal modelo desta categoria é o Modelo em Cascata (criado por Winston Royce em 1970), a partir do qual diversos outros modelos foram propostos, inclusive a maioria dos modelos incrementais e evolutivos. Outro representante é o Modelo em V (V-Model).

Há muitas situações em que os requisitos são razoavelmente bem definidos, mas o tamanho do sistema impossibilita a adoção de um modelo sequencial, sobretudo pela necessidade de disponibilizar rapidamente uma versão para o usuário. Nesses

casos, um modelo incremental é indicado. No desenvolvimento incremental, o sistema é dividido em subsistemas ou módulos, tomando por base a funcionalidade.

Os incrementos (ou versões) são definidos, começando com um pequeno subsistema funcional que, a cada ciclo, é acrescido de novas funcionalidades. Além de acrescentar novas funcionalidades, nos novos ciclos, as funcionalidades providas anteriormente podem ser modificadas para melhor satisfazer às necessidades dos clientes/usuários.

Vale destacar que a definição das versões (e a correspondente segmentação e atribuição dos requisitos a essas versões) é realizada antes do desenvolvimento da primeira versão. Os principais representantes são RUP (Rational Unified Process), junto do Modelo RAD (Rapid Application Development) e, inclusive, as famosas Metodologias Ágeis.

Sistemas de software, como quaisquer sistemas complexos, evoluem ao longo do tempo. Seus requisitos, muitas vezes, são difíceis de serem estabelecidos ou mudam com frequência ao longo do desenvolvimento. Assim, é importante ter como opção modelos de ciclo de vida que lidem com incertezas e acomodem melhor as contínuas mudanças.

Alguns modelos incrementais, dado que preconizam um desenvolvimento iterativo, podem ser aplicados a esses casos, mas a grande maioria toma por pressuposto que os requisitos são bem definidos. Modelos evolucionários buscam preencher essa lacuna. Enquanto modelos incrementais têm por base a entrega de versões operacionais desde o primeiro ciclo, modelos evolutivos não têm essa preocupação.

Muito pelo contrário: o que ocorre na maioria das vezes é que os primeiros ciclos produzem protótipos ou até mesmo apenas modelos. À medida que o desenvolvimento avança e os requisitos vão ficando mais claros e estáveis, protótipos vão dando lugar a versões operacionais, até que o sistema completo seja construído. *Bacana?*

Assim, quando o problema não é bem definido e ele não pode ser totalmente especificado no início do desenvolvimento, deve-se optar por um modelo evolutivo. A avaliação ou o uso do protótipo/sistema pode aumentar o conhecimento sobre o produto e melhorar o entendimento que se tem acerca dos requisitos. Entretanto, é necessária uma forte gerência do projeto e de configuração.



1. (CESPE – 2009 – TCE/TO – Analista de Sistemas - A) Quanto maior e mais complexo o projeto de software, mais simples deve ser o modelo de processo a ser adotado.

Comentários:

Galera, não existe essa relação! Em geral, quanto mais complexo o projeto mais complexo o modelo. No entanto, isso também não é uma regra. Hoje em dia, existem projetos megacomplexos feitos utilizando metodologias ágeis, por exemplo.

Gabarito: E

---

2. (CESPE – 2009 – TCE/TO – Analista de Sistemas - B) O modelo de ciclo de vida do software serve para delimitar o alvo do software. Nessa visão, não são consideradas as atividades necessárias e o relacionamento entre elas.

Comentários:

> *A escolha de um modelo de ciclo de vida (para concursos, sinônimo de modelo de processo) é o ponto de partida para a definição de um processo de desenvolvimento de software. Um modelo de ciclo de vida, geralmente, organiza as macro-atividades básicas do processo, estabelecendo precedência e dependência entre as mesmas. Tudo certo?*

Pelo contrário, o alvo do software serve para delimitar o modelo de ciclo de vida a ser escolhido. Primeiro, define-se o alvo do software para, só então, escolher o modelo de ciclo de vida mais adequado. Ademais, são consideradas as atividades necessárias e o relacionamento entre elas.

Gabarito: E

---

3. (CESPE – 2009 – TCE/TO – Analista de Sistemas - C) A escolha do modelo do ciclo de vida não depende de características específicas do projeto, pois o melhor modelo é sempre o mais usado pela equipe do projeto.

Comentários:

Um processo de software não pode ser definido de forma universal. Para ser eficaz e conduzir à construção de produtos de boa qualidade, um processo deve ser adequado às especificidades do projeto em questão. Deste modo, processos devem ser definidos caso a caso, considerando-se diversos aspectos específicos do projeto em questão, tais como:

Conforme vimos em aula, não faz o menor sentido! A escolha depende das características do projeto; além disso, não existe um "*melhor*" modelo.

Gabarito: E

---

4. (ESAF - 2012 – CGU – Analista de Sistemas) A escolha de um modelo é fortemente dependente das características do projeto. Os principais modelos de ciclo de vida podem ser agrupados em três categorias principais:

a) sequenciais, cascata e evolutivos.
b) sequenciais, incrementais e ágeis.
c) sequenciais, incrementais e evolutivos.
d) sequenciais, ágeis e cascata.
e) cascata, ágeis e evolutivos.

Comentários:

Ele ainda afirma que a maioria dos processos de software é baseada em três modelos gerais: modelo em cascata; desenvolvimento iterativo e engenharia de software baseada em componentes. Isso entra em contradição com o que dizem outros autores, i.e., os principais modelos podem ser agrupados em três categorias: modelos sequenciais, modelos incrementais e modelos evolutivo.

Conforme vimos em aula, bastava lembrar da imagem: Sequencial, Incremental e Evolutivo. Galera, ágil é iterativo e incremental e, não, um modelo de ciclo de vida.

Gabarito: C

---

5. (CESPE – 2011 – MEC – Analista de Sistemas) O processo de desenvolvimento de software é uma caracterização descritiva ou prescritiva de como um produto de software deve ser desenvolvido.

Comentários:

Existem mais um conceito importante nessa aula! É o conceito de Metodologia de Desenvolvimento de Software (também chamada de Processo de Desenvolvimento de Software). O que é isso, professor? É uma caracterização prescritiva ou descritiva de como um produto de software deve ser desenvolvido, i.e., define o quê, como e quando fazer algo – um exemplo clássico é o RUP!

Perfeito! Segundo Pressman, um modelo prescritivo consiste em um conjunto específico de atividades de arcabouço, como – por exemplo – a NBR ISSO/IEC 12207 que estabelece uma estrutura comum para os processos de ciclo de vida de software. Já um modelo descritivo apresenta o processo em detalhes, é como se fosse um guia para o desenvolvimento de software. Ambas as formas podem ser utilizadas.

Gabarito: C

---

6. (CESPE – 2013 – TRT/10ª – Analista de Sistemas) As atividades fundamentais relacionadas ao processo de construção de um software incluem a especificação, o desenvolvimento, a validação e a evolução do software.

Comentários:

Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software.

Conforme vimos em aula, a questão está corretíssima!

Gabarito: C

---

7. (CESPE – 2010 – TRE/BA – Analista de Sistemas) Um modelo de processo de software consiste em uma representação complexa de um processo de software, apresentada a partir de uma perspectiva genérica.

Comentários:

Podemos dizer que se trata da representação abstrata de um esqueleto de processo, incluindo tipicamente algumas atividades principais, a ordem de precedência entre elas e, opcionalmente, artefatos requeridos e produzidos. Em geral, um modelo de processo descreve uma filosofia de organização de atividades, estruturando-as em fases e definindo como essas fases estão relacionadas.

Um modelo de processo de software ou modelo de ciclo de vida trata da representação abstrata/simplificada de um processo de software, apresentada a partir de uma perspectiva genérica.

Gabarito: E

---

8. **(CESPE – 2011 – MEC – Analista de Sistemas)** Atividades comuns a todos os processos de software incluem a especificação, o projeto, a implementação e a validação.

Comentários:

Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software.

Vejam que ele dividiu a atividade de Desenvolvimento em Projeto e Implementação, mas não invalida a questão. *Professor, ele não falou sobre a evolução!* Sim, mas a questão apenas afirma que essas são atividades comuns e, não, que são as únicas.

Gabarito: C

---

9. **(CESPE – 2015 – STJ – Analista de Sistemas)** As principais atividades de engenharia de software são especificação, desenvolvimento, validação e evolução.

Comentários:

Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software.

Conforme vimos em aula, a questão peca um pouco ao dizer que são atividades da engenharia de software. O ideal seria dizer que são as atividades principais do processo de software. Não sei se entraram com recurso e a banca recusou e se não entraram com recurso. Percebam também que é a terceira vez esse tipo de questão cai em prova.

Gabarito: C

---

10. **(FCC – 2012 – MPE/AP – Analista de Sistemas)** Um processo de software é um conjunto de atividades relacionadas que levam à produção de um produto de software. Existem muitos processos de software diferentes, mas todos devem incluir quatro atividades fundamentais: especificação, projeto e implementação, validação e:

a) teste
b) evolução.
c) prototipação.
d) entrega.
e) modelagem.

Comentários:

Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software.

Conforme vimos em aula, trata-se da evolução!

Gabarito: B

---

11. **(CESPE – 2010 – EMBASA – Analista de Sistemas)** Ciclo de vida de um software resume-se em eventos utilizados para definir o status de um projeto.

Comentários:

O ciclo de vida de software é um conjunto de estágios (e, não, eventos) utilizados para definir o status de um software (e, não, projeto). Sommerville afirma, por exemplo, que um ciclo de vida é composto por estágios que demonstram atividades fundamentais de desenvolvimento. Questão errada!

Gabarito: E

---

12. (CESPE – 2016 – TCE/PR – Analista de Sistemas) As fases do ciclo de vida de um software são:

a) concepção, desenvolvimento, entrega e encerramento.
b) iniciação, elaboração, construção e manutenção.
c) escopo, estimativas, projeto e processo e gerência de riscos.
d) análise, desenvolvimento, teste, empacotamento e entrega.
e) planejamento, análise e especificação de requisitos, projeto, implementação, testes, entrega e implantação, operação e manutenção.

Comentários:

> *Conforme eu disse anteriormente, alguns autores afirmam que os modelos de ciclo de vida básicos, de maneira geral, contemplam pelo menos as fases de: Planejamento; Análise e Especificação de Requisitos; Projeto; Implementação; Testes; Entrega e Implantação; Operação; e Manutenção. Abaixo eu trago uma descrição genérica sobre cada uma dessas fases.*

Conforme vimos em aula, a questão trata da última opção.

Gabarito: E

---

13. (MPE/RS – 2012 – MPE/RS – Analista de Sistemas) O ciclo de vida básico de um software compreende:

a) a implementação, a implantação e o teste.
b) a análise, a segurança e o controle de usuários.
c) a implementação, a análise e o teste.
d) a implementação, a validação e as vendas.
e) a análise, o projeto, a implementação e o teste.

Comentários:

Conforme eu disse anteriormente, alguns autores afirmam que os modelos de ciclo de vida básicos, de maneira geral, contemplam pelo menos as fases de: Planejamento; Análise e Especificação de Requisitos; Projeto; Implementação; Testes; Entrega e Implantação; Operação; e Manutenção. Abaixo eu trago uma descrição genérica sobre cada uma dessas fases.

Aqui temos que escolher a mais correta, na medida em que a primeira, terceira e última opções estão corretas. Nesse sentido, a última opção é a mais correta!

Gabarito: E

---

14. (CESGRANRIO – 2011 – PETROBRÁS – Analista de Sistemas) A especificação de uma Metodologia de Desenvolvimento de Sistemas tem como pré-requisito indispensável, em relação ao que será adotado no processo de desenvolvimento, a definição do:

a) Engenheiro Responsável pelo Projeto
b) Documento de Controle de Sistemas
c) Software para Desenvolvimento
d) Ciclo de Vida do Software
e) Bloco de Atividades

Comentários:

A escolha de um modelo de ciclo de vida (para concursos, sinônimo de modelo de processo) é o ponto de partida para a definição de um processo de desenvolvimento de software. Um modelo de ciclo de vida, geralmente, organiza as macro-atividades básicas do processo, estabelecendo precedência e dependência entre as mesmas. Tudo certo?

Essa questão foi bem escrita (finalmente)! Percebam que ele diferencia metodologia de desenvolvimento de software do ciclo de vida de software. Por exemplo: primeiro, eu defino que eu vou utilizar um ciclo de vida evolutivo e depois eu decido que eu vou utilizar a metodologia de desenvolvimento em espiral.

Gabarito: D

---

15. (CESPE – 2008 – INPE – Analista de Sistemas) O ciclo de vida do software tem início na fase de projeto.

Comentários:

Conforme eu disse anteriormente, alguns autores afirmam que os modelos de ciclo de vida básicos, de maneira geral, contemplam pelo menos as fases de: Planejamento; Análise e Especificação de Requisitos; Projeto; Implementação; Testes; Entrega e Implantação; Operação; e Manutenção. Abaixo eu trago uma descrição genérica sobre cada uma dessas fases.

Conforme vimos em aula, ele não começa com projeto! *Como você começa com projeto sem sequer levantar os requisitos?* Era possível responder usando só lógica!

Gabarito: E

---

16. (CESPE – 2013 – TRT/10 – Analista de Sistemas) O ciclo de vida de um software, entre outras características, está relacionado aos estágios de concepção, projeto, criação e implementação.

Comentários:

Essa questão é péssima! Ela foi retirada do livro *Sistemas de Informação: Uma Abordagem Gerencial* (Steven R. Gordon, Judith R. Gordon). Nesse livro, os autores de fato tratam o ciclo de vida de software como concepção, projeto, criação e implementação. Essa é uma daquelas quase impossível de acertar! Esse não é um autor consagrado de Engenharia de Software.

Gabarito: C

---

17. (CESPE – 2011 – TJ/ES – Analista de Sistemas) Entre as etapas do ciclo de vida de software, as menos importantes incluem a garantia da qualidade, o projeto e o estudo de viabilidade. As demais atividades do ciclo, como a implementação e os testes, requerem maior dedicação da equipe e são essenciais.

Comentários:

Para a definição completa do processo, cada atividade descrita no modelo de ciclo de vida deve ser decomposta e para suas subatividades, devem ser associados métodos, técnicas, ferramentas e critérios de qualidade, entre outros, formando uma

base sólida para o desenvolvimento. Adicionalmente, outras atividades, tipicamente de cunho gerencial, devem ser definidas, como controle e garantia da qualidade. Não existe uma hierarquia de etapas mais importantes e menos importantes!

Gabarito: E

---

18. (CESPE - 2016 – TCE/PR – Analista de Sistemas – A) A engenharia de software está relacionada aos diversos aspectos de produção de software e inclui as atividades de especificação, desenvolvimento, validação e evolução de software.

Comentários:

> *Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software.*

Conforme vimos em aula, a questão está correta.

Gabarito: C

---

19. (CESPE - 2016 – TCE/PR – Analista de Sistemas – D) Um processo de software é composto por quatro atividades fundamentais: iniciação, desenvolvimento, entrega e encerramento.

Comentários:

> *Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software.*

Conforme vimos em aula, as atividades estão incorretas.

Gabarito: E

---

20. (CESPE - 2016 – TCE/PR – Analista de Sistemas – B) A metodologia de processos genérica para a engenharia de software é composta de seis etapas: comunicação, planejamento, modelagem, construção, emprego e aceitação.

Comentários:





Conforme vimos em aula, essas etapas parecem as fases do ciclo de vida de acordo com o Pressman, que são: Comunicação, Planejamento, Modelagem, Construção e Implantação. Percebam que ele troca Implantação por Emprego e Aceitação.

Gabarito: E

---

21. (INSTITUTO CIDADE – 2012 – TCM/GO – Analista de Sistemas) De acordo com a engenharia de software, como todo produto industrial, o software possui um ciclo de vida. Cada fase do ciclo de vida possui divisões e subdivisões. Em qual fase avaliamos a necessidade de evolução dos softwares em funcionamento para novas plataformas operacionais ou para a incorporação de novos requisitos?

a) Fase de operação;
b) Fase de retirada;
c) Fase de definição;
d) Fase de design.
e) Fase de desenvolvimento;

Comentários:





A fase retirada é um grande desafio para os tempos atuais. Diversos softwares que estão em funcionamento em empresas possuem excelente níveis de confiabilidade e de correção. No entanto, eles precisam evoluir para novas plataformas operacionais ou para a incorporação de novos requisitos.

A retirada desses softwares legados em uma empresa é sempre uma decisão difícil: *como abrir mão daquilo que é confiável e ao qual os funcionários estão acostumados, após anos de treinamento e utilização?* Portanto, processos de reengenharia podem ser aplicados para viabilizar a transição ou migração de um software legado para um novo software de forma a proporcionar uma retirada mais suave.

A avaliação da necessidade de evolução do software em funcionamento para novas plataformas operacionais ou para a incorporação de novos requisitos realmente ocorrem na fase de retirada.

Gabarito: B

---

22. **(CESPE - 2010 – DETRAN/ES – Analista de Sistemas)** Quando um aplicativo de software desenvolvido em uma organização atinge, no fim do seu ciclo de vida, a fase denominada aposentadoria, descontinuação ou fim de vida, todos os dados por ele manipulados podem ser descartados.

Comentários:



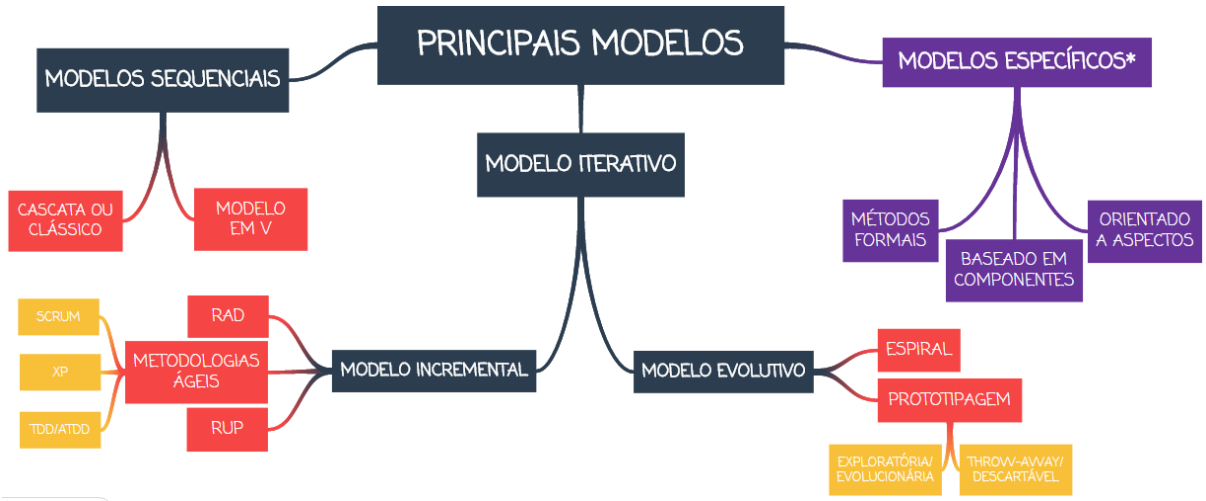


Essa questão é um absurdo! Observem que ela afirma "*podem ser descartados*". Ora, é evidente que podem ser descartados. No entanto, o gabarito oficial é errado!

Gabarito: E

ACERTEI	ERREI

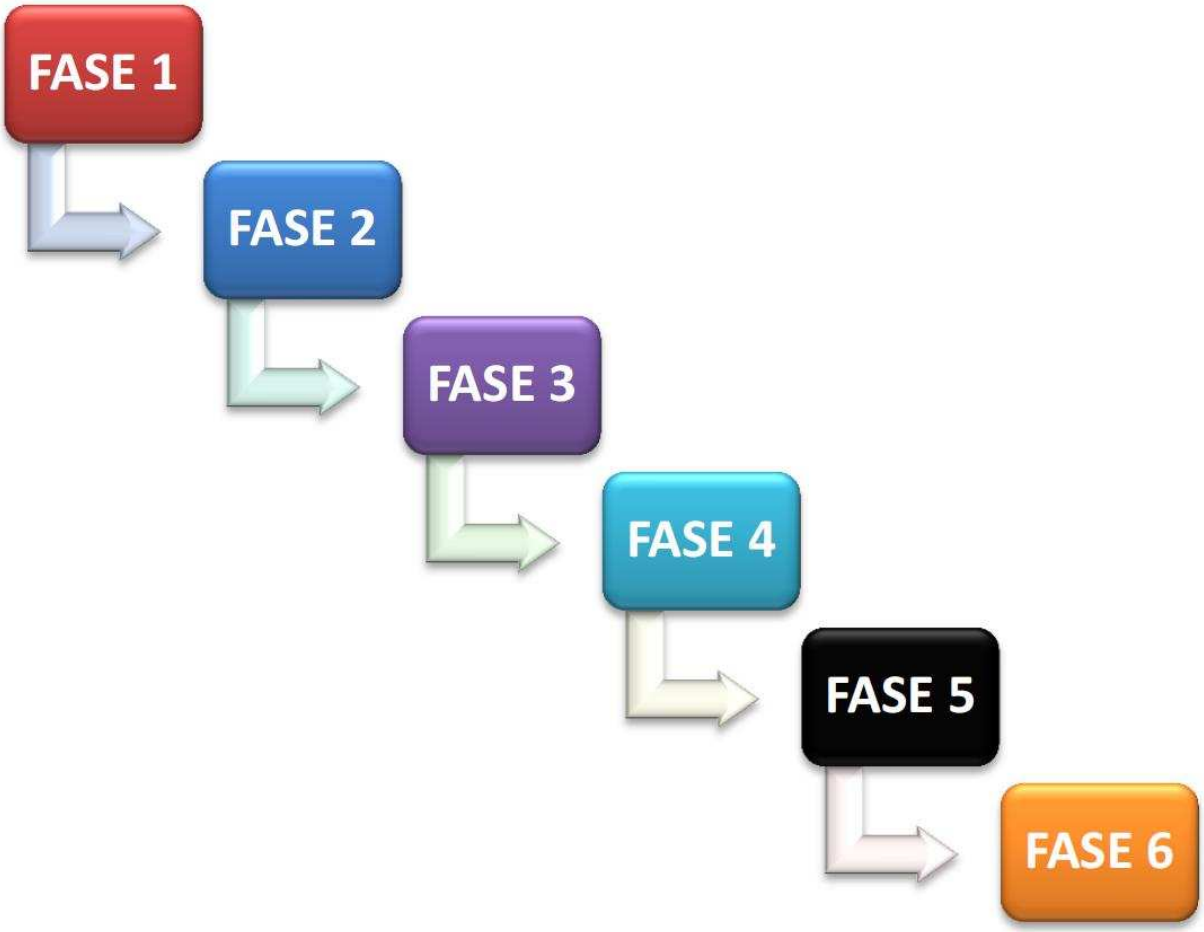
MODELO EM CASCATA



Citado inicialmente em 1970 por W. Royce, também designado Cascata ou Clássico ou Sequencial ou Linear ou Tradicional ou Waterfall ou Rígido ou Monolítico (todos esses nomes já caíram em prova!). Esse nome é devido ao encadeamento simples de uma fase com a outra. Os estágios do modelo demonstram as principais atividades de desenvolvimento. Observem a imagem mais abaixo!

No Modelo em Cascata, uma fase só se inicia após o término e aprovação da fase anterior, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada após o término e aprovação da Fase 4. *Mas que fases são essas?* Bem, agora que complica, porque cada autor resolve criar suas fases!

De acordo com Vasconcelos (2006), no Modelo em Cascata, o projeto segue uma série passos ordenados, ao final de cada fase, a equipe de projeto finaliza uma revisão. Além disso, o desenvolvimento não continua até que o cliente esteja satisfeito com os resultados alcançados. *Vocês conseguem perceber como essas restrições engessam o desenvolvimento?*



Por Sommerville	Por Royce	Por Pressman (4ª Ed.)	Por Pressman (6ª Ed.)
Análise e Definição de Requisitos	Requisitos de Sistema	Modelagem e Engenharia do Sistema/Informação	Comunicação
Projeto de Sistema e Software	Requisitos de Software	Análise de Requisitos de Software	Planejamento
Implementação e Teste de Unidade	Análise	Projeto	Modelagem
Integração e Teste de Sistema	Projeto	Geração de Código	Construção
Operação e Manutenção	Codificação	Teste e Manutenção	Implantação
	Teste		
	Operação		

Percebam que há diferenças gritantes entre os autores! Inclusive, há divergências até entre autor e ele mesmo, dependendo da versão do livro (Exemplo: Pressman mudou as fases na última edição de seu livro). *Professor, você já viu isso cair em prova?* Sim, já vi! *E o que aconteceu?* Bem, polêmica, recursos, etc! Não há o que fazer... minha classificação preferida é a do Royce.

Na prática, esses estágios não são completamente sequenciais, i.e., eles se sobrepõem e trocam informações entre si. Na teoria, são fases sequenciais com o resultado de cada fase consistindo em um ou mais documentos aprovados ou não, dependendo dos problemas. Por exemplo: durante o projeto, são identificados problemas com requisitos.

De modo geral, grande parte dos modelos possuem as seguintes fases:

a) Planejamento: faz-se o esboço do escopo e dos requisitos, além de estimativas razoáveis sobre recursos, custos e prazos.

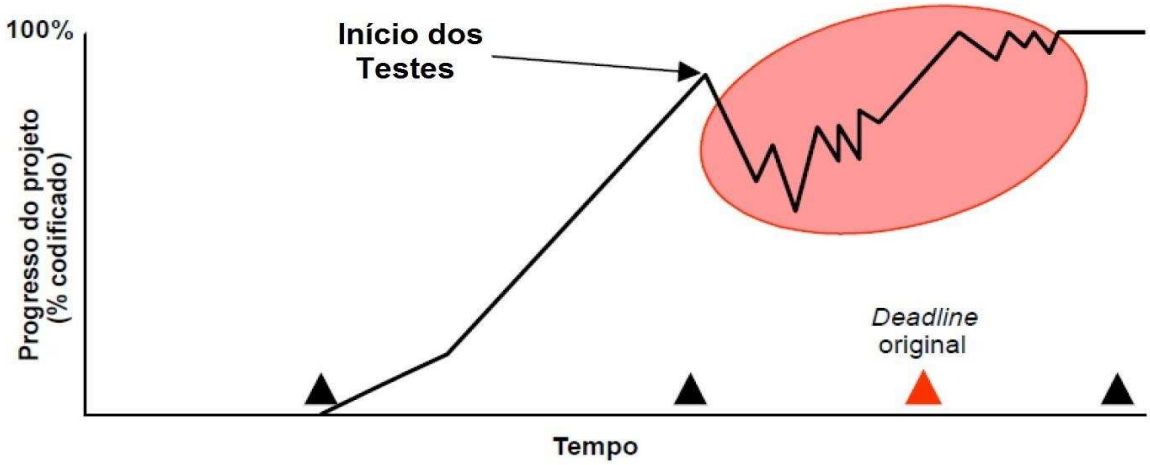
b) Análise e Especificação de Requisitos: durante essa fase, refina-se os requisitos e o escopo e desenha-se o problema em questão.

c) Projeto: durante essa fase, incorpora-se requisitos tecnológicos aos requisitos essenciais do sistema e projeta-se a arquitetura do sistema.

d) Implementação: durante essa fase, codifica-se o software como um conjunto de programas executáveis pela máquina.

e) Teste: o programa é testado como um sistema completo para garantir que os requisitos de software foram atendidos.

f) Implantação, Operação e Manutenção: o sistema de software é liberado para o cliente, treina-se usuários, gerencia serviços e realiza manutenções.



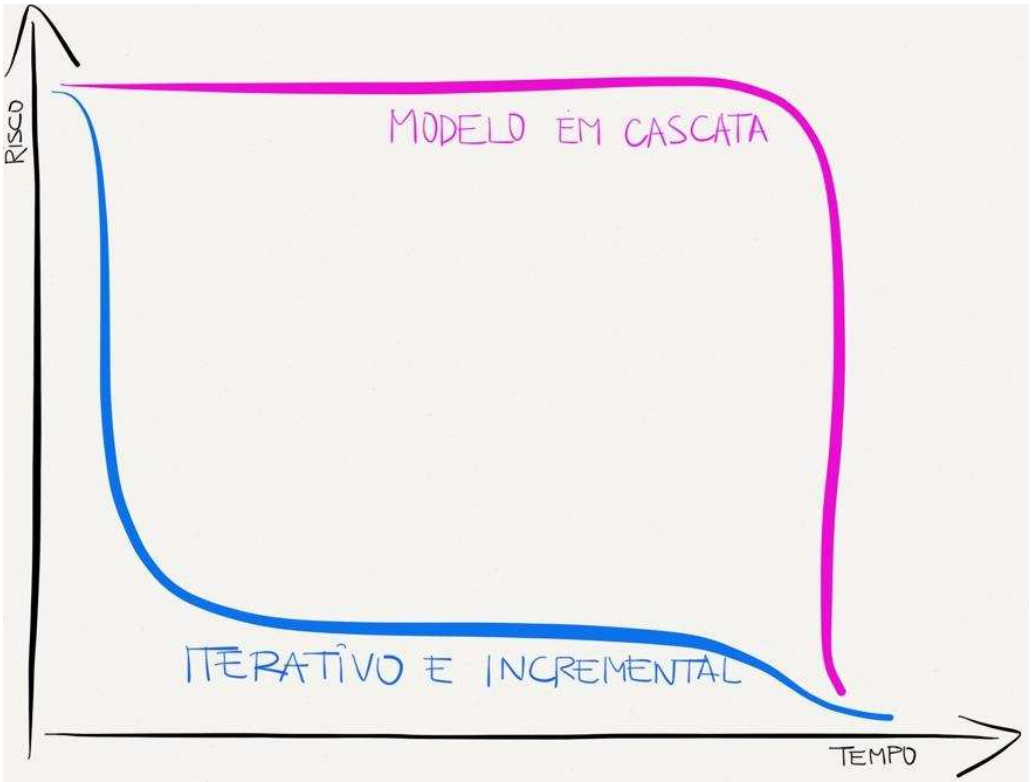
O Modelo em Cascata atrasa a redução de riscos...

Professor, o que você quer dizer com atrasar a redução de riscos? Bem, essa é uma desvantagem recorrente em provas. Como uma fase só se inicia após o término da fase anterior, só é possível em geral verificar se houve erros nas últimas fases – como pode ser visto na imagem abaixo. Em outros modelos, os riscos são reduzidos desde as primeiras fases do processo de desenvolvimento.

Percebam que os riscos deveriam ser descobertos logo no início do processo de desenvolvimento, porém eles são descobertos somente após o início dos testes e integração. Vocês podem notar que, nesse instante (parte vermelha), o progresso do projeto avança e retrai diversas vezes, porque o sistema está sendo corrigido devido a requisitos modificados.

Vejam, também, que o projeto não terminou em seu deadline original. Como a redução dos riscos atrasou, todo andamento do projeto também atrasou. Dessa forma, não se cumpriu nem o prazo do projeto e, provavelmente, nem o orçamento e talvez seu escopo – tendo em vista que, quanto mais ao fim do projeto um erro é identificado, mais caras se tornam as modificações.

Entenderam essa parte direitinho? Um erro na fase de requisitos, por exemplo, que não foi corrigido e foi descoberto no final do processo de desenvolvimento, terá um custo de correção altíssimo, visto que provavelmente terá que se refazer tudo novamente. Ora, se eu peço a construção de um carro e você constrói uma moto, o custo para corrigir esse erro será altíssimo.



Portanto não confundam essas duas coisas: se o erro ocorreu no início e foi identificado no início, terá baixo custo de correção; se o erro ocorreu no início e foi identificado no final, terá alto custo de correção. O custo de correção de um erro está mais focado no momento em que um erro é identificado do que no momento em que ele ocorre. *Bacana?*

Outra maneira de visualizar o atraso é por meio de um gráfico Risco x Tempo, comparando o modelo em cascata com o Modelo Iterativo e Incremental. Observem que o Modelo Iterativo e Incremental já começa a reduzir os riscos desde o início do processo, em contraste com o Modelo em Cascata que acumula os riscos até a fase de testes, integração e implantação do sistema.

Galera, a grande vantagem do Modelo em Cascata é que o desenvolvimento é dividido em fases distintas, padronizadas, etc (antigamente, os softwares eram construídos artesanalmente). É possível colocar pessoas com perfis específicos para cada fase, i.e., quem tem facilidade de se comunicar vai ser analista de requisitos, programadores vão fazer a codificação, etc.

A grande desvantagem é que - em projetos complexos – demora-se muito para chegar até a fase de testes, sendo que o cliente não vê nada rodando até a implantação. Então, pode acontecer de, nas fases finais, os requisitos da organização

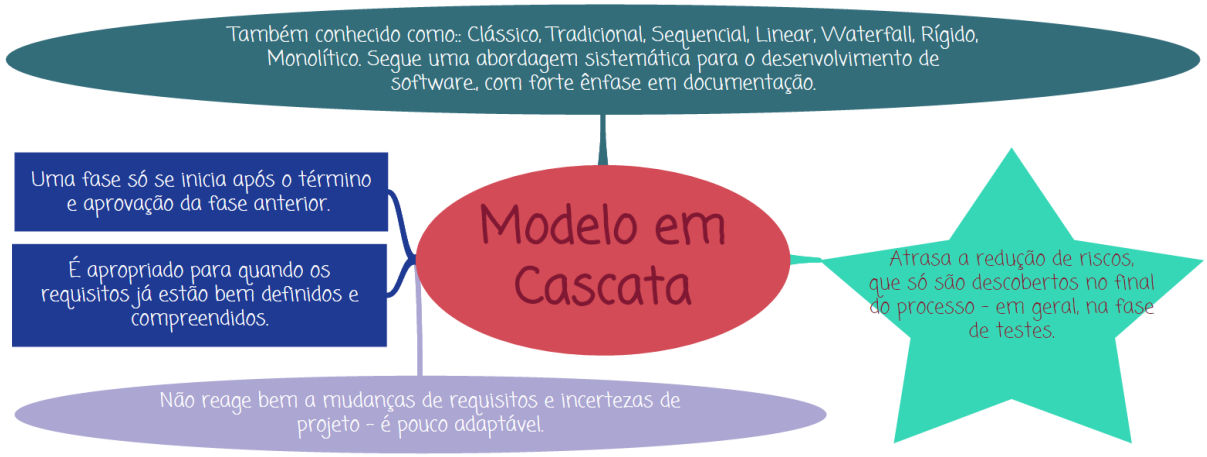
não serem mais os mesmos daqueles do início e o software não ter mais utilidade para organização.

Professor, então o Modelo em Cascata não deve ser usado em nenhuma hipótese? Calma lá, ele pode ser usado! No entanto, sua utilização deve ocorrer preferencialmente quando os requisitos forem bem compreendidos e houver pouca probabilidade de mudanças radicais durante o desenvolvimento do sistema. *Vocês entenderam?*

VANTAGENS	DESVANTAGENS
É simples de entender e fácil de aplicar, facilitando o planejamento.	Divisão inflexível do projeto em estágios distintos.
Fixa pontos específicos para a entrega de artefatos.	Dificuldade em incorporar mudanças de requisitos.
Funciona bem para equipes tecnicamente fracas.	Clientes só visualizam resultados próximos ao final do projeto.
É fácil de gerenciar, devido a sua rigidez.	Atrasa a redução de riscos.
Realiza documentação extensa por cada fase ou estágio.	Apenas a fase final produz um artefato de software entregável.
Possibilita boa aderência a outros modelos de processo.	Cliente deve saber todos os requisitos no início do projeto.
Funciona bem com projetos pequenos e com requisitos bem conhecidos.	Modelo inicial (Royce) não permitia feedback entre as fases do projeto.
	Pressupõe que os requisitos ficarão estáveis ao longo do tempo.
	Não funciona bem com projetos complexos e OO, apesar de compatível.

Podemos afirmar, então, que o Modelo em Cascata é considerado um método tradicional e fortemente prescritivo, i.e., ele busca sempre dizer o que fazer, em geral, por meio de planos definidos no início do projeto. *Que planos, professor?* Escopo, custo, cronograma... tudo isso bem detalhado em uma extensa documentação. *Mudanças são bem-vindas?* Claro que não!

ESQUEMA


Também conhecido como:: Clássico, Tradicional, Sequencial, Linear, Waterfall, Rígido, Monolítico. Segue uma abordagem sistemática para o desenvolvimento de software, com forte ênfase em documentação.
Uma fase só se inicia após o término e aprovação da fase anterior.
É apropriado para quando os requisitos já estão bem definidos e compreendidos.
Modelo em Cascata
Atrasa a redução de riscos, que só são descobertos no final do processo - em geral, na fase de testes.
Não reage bem a mudanças de requisitos e incertezas de projeto - é pouco adaptável.

CAIU na prova!
DIVERSAS BANCAS

1. (FCC - 2012 - TJ-RJ - Analista Judiciário - Análise de Sistemas - E) Dos diferentes modelos para o ciclo de vida de desenvolvimento de um software é correto afirmar que o modelo em cascata é o mais recente e complexo.

Comentários:

Citado inicialmente em 1970 por W. Royce, também designado Cascata ou Clássico ou Sequencial ou Linear ou Tradicional ou Waterfall ou Rígido ou Monolítico (todos esses nomes já caíram em prova!). Esse nome é devido ao encadeamento simples de uma fase com a outra. Os estágios do modelo demonstram as principais atividades de desenvolvimento. Observem a imagem mais abaixo!

Mais recente? Não, muito antigo! *Complexo?* Não, possui um encadeamento simples de fases.

Gabarito: E

---

2. (FCC - 2009 - SEFAZ-SP - Agente Fiscal de Rendas - Tecnologia da Informação - Prova 3 - B) O processo de engenharia de software denominado ciclo de vida clássico refere-se ao modelo incremental.

Comentários:

Citado inicialmente em 1970 por W. Royce, também designado Cascata ou Clássico ou Sequencial ou Linear ou Tradicional ou Waterfall ou Rígido ou Monolítico (todos esses nomes já caíram em prova!). Esse nome é devido ao encadeamento simples de uma fase com a outra. Os estágios do modelo demonstram as principais atividades de desenvolvimento. Observem a imagem mais abaixo!

Não, modelo clássico se refere a modelo em cascata, sequencial, linear, tradicional, waterfall, rígido ou monolítico.

Gabarito: E

---

3. (CESPE – 2009 – INMETRO – Analista de Sistemas) Em uma empresa que tenha adotado um processo de desenvolvimento de software em cascata, falhas no levantamento de requisitos têm maior possibilidade de gerar grandes prejuízos do que naquelas que tenham adotado desenvolvimento evolucionário.

Comentários:

VANTAGENS	DESVANTAGENS
É simples de entender e fácil de aplicar, facilitando o planejamento.	Divisão inflexível do projeto em estágios distintos.
Fixa pontos específicos para a entrega de artefatos.	**Dificuldade em incorporar mudanças de requisitos.**
Funciona bem para equipes tecnicamente fracas.	Clientes só visualizam resultados próximos ao final do projeto.
É fácil de gerenciar, devido a sua rigidez.	Atrasa a redução de riscos.
Realiza documentação extensa por cada fase ou estágio.	Apenas a fase final produz um artefato de software entregável.
Possibilita boa aderência a outros modelos de processo.	Cliente deve saber todos os requisitos no início do projeto.
Funciona bem com projetos pequenos e com requisitos bem conhecidos.	Não fornece feedback entre as fases.
	Pressupõe que os requisitos ficarão estáveis ao longo do tempo.
	Não funciona bem com projetos complexos e OO, apesar de compatível.

O Modelo em Cascata, de fato, não reage bem a mudanças. Já Modelo evolucionário é mais adaptável a mudanças por se utilizar de iterações.

Gabarito: C

---

4. (CESPE – 2011 – MEC – Analista de Sistemas) O modelo Waterfall tem a vantagem de facilitar a realização de mudanças sem a necessidade de retrabalho em fases já completadas.

Comentários:

VANTAGENS	DESVANTAGENS
É simples de entender e fácil de aplicar, facilitando o planejamento.	Divisão inflexível do projeto em estágios distintos.
Fixa pontos específicos para a entrega de artefatos.	**Dificuldade em incorporar mudanças de requisitos.**
Funciona bem para equipes tecnicamente fracas.	Clientes só visualizam resultados próximos ao final do projeto.
É fácil de gerenciar, devido a sua rigidez.	Atrasa a redução de riscos.
Realiza documentação extensa por cada fase ou estágio.	Apenas a fase final produz um artefato de software entregável.
Possibilita boa aderência a outros modelos de processo.	Cliente deve saber todos os requisitos no início do projeto.
Funciona bem com projetos pequenos e com requisitos bem conhecidos.	Não fornece feedback entre as fases.
	Pressupõe que os requisitos ficarão estáveis ao longo do tempo.
	Não funciona bem com projetos complexos e OO, apesar de compatível.

Pelo contrário, há dificuldade de lidar com requisitos voláteis, tendo em vista que dependendo do erro, é necessário refazê-lo desde seu início.

Gabarito: E

---

5. (CESPE – 2008 – TST – Analista de Sistemas) No modelo de desenvolvimento sequencial linear, a fase de codificação é a que gera erros de maior custo de correção.

Comentários:

Entenderam essa parte direitinho? Um erro na fase de requisitos, por exemplo, que não foi corrigido e foi descoberto no final do processo de desenvolvimento, terá um custo de correção altíssimo, visto que provavelmente terá que se refazer tudo novamente. Ora, se eu peço a construção de um carro e você constrói uma moto, o custo para corrigir esse erro será altíssimo.

Portanto não confundam essas duas coisas! Percebam o que eu disse: quanto mais tarde se descobre um erro, mais caro se torna sua correção. Dizendo isso de outra forma: erros nas fases iniciais possuem custo de correção altíssimo. Uma coisa é o momento em que o erro ocorre (quanto mais cedo, mais caro); outra coisa é o momento em que um erro é identificado (quanto mais tarde, mais caro). Bacana?

Percebam que erros nas fases iniciais possuem custos de correção mais altos. *Logo, o maior custo está na fase de codificação?* Não, está na fase de requisitos que é a fase inicial!

Gabarito: E

---

6. (CESPE – 2009 – INMETRO – Analista de Sistemas) Em um processo de desenvolvimento em cascata, os testes de software são realizados todos em um mesmo estágio, que acontece após a finalização das fases de implementação.

Comentários:

Por Sommerville	Por Royce	Por Pressman (4ª Ed.)	Por Pressman (6ª Ed.)
Análise e Definição de Requisitos	Requisitos de Sistema	Modelagem e Engenharia do Sistema/Informação	Comunicação
Projeto de Sistema e Software	Requisitos de Software	Análise de Requisitos de Software	Planejamento
Implementação e Teste de Unidade	Análise	Projeto	Modelagem
Integração e Teste de Sistema	Projeto	Geração de Código	Construção
Operação e Manutenção	Codificação	Teste e Manutenção	Implantação
	Teste		
	Operação		

Todos em um mesmo estágio, não. A grande maioria dos testes ocorrem, de fato, após a finalização das fases de implementação. No entanto, podem ocorrer testes unitários durante a própria implementação.

Gabarito: E

---

7. (CESPE – 2008 – SERPRO – Analista de Sistemas) O modelo em cascata consiste de fases e atividades que devem ser realizadas em sequência, de forma que uma atividade é requisito da outra.

Comentários:

> *No Modelo em Cascata, uma fase só se inicia após o término e aprovação da fase anterior, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada após o término e aprovação da Fase 4. Mas que fases são essas? Bem, agora que complica, porque cada autor resolve criar suas fases! Vejamos: (...)*

Vimos isso exaustivamente: no modelo em cascata, uma fase só se inicia após o término e aprovação da fase anterior.

Gabarito: C

---

8. (CESPE – 2005 – AL/ES – Analista de Sistemas - B) O modelo de desenvolvimento em cascata descreve ciclos sequenciais, incrementais e iterativos, possuindo, entre outras, as fases de requisitos e implementação.

Comentários:

Não! Ele não descreve ciclos, muito menos ciclos iterativos. Na verdade, essa é a definição de Modelo Iterativo e Incremental.

Gabarito: E

---

9. (CESPE – 2004 – STJ – Analista de Sistemas) O modelo de desenvolvimento seqüencial linear, também chamado modelo clássico ou modelo em cascata, caracteriza-se por não acomodar adequadamente as incertezas que existem no

início de um projeto de software, em especial as geradas pela dificuldade do cliente de explicitar todos os requerimentos que o programa deve contemplar.

Comentários:

Professor, o que você quer dizer com atrasar a redução de riscos? Bem, essa é uma desvantagem recorrente em provas. Como uma fase só se inicia após o término da fase anterior, só é possível em geral verificar se houve erros nas últimas fases – como pode ser visto na imagem abaixo. Em outros modelos, os riscos são reduzidos desde as primeiras fases do processo de desenvolvimento.

Perfeito, lembrem-se que ele acumula riscos e não lida bem com requisitos voláteis.

Gabarito: C

---

10. (CESPE – 2009 – IPEA – Analista de Sistema) No modelo em cascata de processo de desenvolvimento, os clientes devem definir os requisitos apenas durante a fase de projeto; e os projetistas definem as estratégias de projeto apenas durante a fase de implementação. As fases do ciclo de vida envolvem definição de requisitos, projeto, implementação, teste, integração, operação e manutenção. Em cada fase do ciclo de vida, podem ser produzidos diversos artefatos.

Comentários:

Essa questão não faz sentido! Os clientes definem os requisitos durante a fase de Definição de Requisitos. Já os projetistas definem as estratégias de projeto apenas durante a fase Projeto.

Gabarito: E

---

11. (CESPE – 2008 – TCE/TO – Analista de Sistema – D) No ciclo de vida em cascata, é possível realizar alternadamente e simultaneamente as atividades de desenvolvimento de software.

Comentários:

No Modelo em Cascata, uma fase só se inicia após o término e aprovação da fase anterior, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada

após o término e aprovação da Fase 4. Mas que fases são essas? Bem, agora que complica, porque cada autor resolve criar suas fases! Vejamos: (...)

Não, é sequencial e linear. Não pode ser alternado e simultâneo!

Gabarito: E

---

12. (CESPE – 2004 – TJ/PA – Analista de Sistema – D) A abordagem sistemática estritamente linear para o desenvolvimento de software é denominada modelo em cascata ou modelo sequencial linear.

Comentários:

Citado inicialmente em 1970 por W. Royce, também designado Cascata ou Clássico ou Sequencial ou Linear ou Tradicional ou Waterfall ou Rígido ou Monolítico (todos esses nomes já caíram em prova!). Esse nome é devido ao encadeamento simples de uma fase com a outra. Os estágios do modelo demonstram as principais atividades de desenvolvimento. Observem a imagem mais abaixo!

Perfeito! Modelo em Cascata, Linear, Sequencial, Waterfall, etc.

Gabarito: C

---

13. (CESPE – 2006 – TSE – Analista de Sistema – D) O modelo em cascata organiza o desenvolvimento em fases. Esse modelo encoraja a definição dos requisitos antes do restante do desenvolvimento do sistema. Após a especificação e a análise dos requisitos, têm-se o projeto, a implementação e o teste.

Comentários:

Por Sommerville	Por Royce	Por Pressman (4ª Ed.)	Por Pressman (6ª Ed.)
Análise e Definição de Requisitos	Requisitos de Sistema	Modelagem e Engenharia do Sistema/Informação	Comunicação
Projeto de Sistema e Software	Requisitos de Software	Análise de Requisitos de Software	Planejamento
Implementação e Teste de Unidade	Análise	Projeto	Modelagem

Integração e Teste de Sistema	Projeto	Geração de Código	Construção
Operação e Manutenção	Codificação	Teste e Manutenção	Implantação
	Teste		
	Operação		

Perfeito! De fato, segue essa ordem!

Gabarito: C

---

14. **(CESPE – 2009 – INMTRO – Analista de Sistema)** No desenvolvimento de software, o modelo em cascata é estruturado de tal maneira que as fases que compõem o desenvolvimento são interligadas. Nessa situação, o final de uma fase implica o início de outra.

Comentários:

No Modelo em Cascata, uma fase só se inicia após o término e aprovação da fase anterior, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada após o término e aprovação da Fase 4. Mas que fases são essas? Bem, agora que complica, porque cada autor resolve criar suas fases! Vejamos: (...)

Perfeito, conforme a definição.

Gabarito: C

---

15. **(CESPE – 2010 – BASA – Analista de Sistema)** No modelo em cascata, o projeto segue uma série de passos ordenados. Ao final de cada projeto, a equipe de projeto finaliza uma revisão. O desenvolvimento continua e, ao final, o cliente avalia a solução proposta.

Comentários:

De acordo com Vasconcelos (2006), no Modelo em Cascata, o projeto segue uma série passos ordenados, ao final de cada fase, a equipe de projeto finaliza uma

revisão. Além disso, o desenvolvimento não continua até que o cliente esteja satisfeito com os resultados alcançados. Vocês conseguem perceber como essas restrições engessam o desenvolvimento?

Ao final de cada *projeto*? Não! Ao final de cada fase.

Gabarito: E

---

16. (CESPE – 2009 – TRE/MT – Analista de Sistema - A) O modelo em cascata é apropriado para software em que os requisitos ainda não foram bem compreendidos, pois é focado na criação de incrementos.

Comentários:

Professor, então o Modelo em Cascata não deve ser usado em nenhuma hipótese? Calma lá, ele pode ser usado! No entanto, sua utilização deve ocorrer preferencialmente quando os requisitos forem bem compreendidos e houver pouca probabilidade de mudanças radicais durante o desenvolvimento do sistema. Vocês entenderam?

Pelo contrário, totalmente errado!

Gabarito: E

---

17. (CESPE – 2009 – UNIPAMPA – Analista de Sistema – D) O modelo em cascata sugere uma abordagem sistemática e sequencial para o desenvolvimento de software. Sua natureza linear leva a estados de bloqueio nos quais, para que nova etapa seja iniciada, é necessário que a documentação associada à fase anterior tenha sido aprovada.

Comentários:

No Modelo em Cascata, uma fase só se inicia após o término e aprovação da fase anterior, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada após o término e aprovação da Fase 4. Mas que fases são essas? Bem, agora que complica, porque cada autor resolve criar suas fases! Vejamos: (...)

Perfeito! Não basta terminar uma fase, é necessário que a sua documentação tenha sido aprovada.

Gabarito: C

18. (CESPE – 2004 – ABIN – Analista de Sistema) O modelo de desenvolvimento seqüencial linear, também denominado modelo em cascata, é incompatível com o emprego de técnica de análise orientada a objetos no desenvolvimento de um sistema de informação.

Comentários:

VANTAGENS	DESVANTAGENS
É simples de entender e fácil de aplicar, facilitando o planejamento.	Divisão inflexível do projeto em estágios distintos.
Fixa pontos específicos para a entrega de artefatos.	Dificuldade em incorporar mudanças de requisitos.
Funciona bem para equipes tecnicamente fracas.	Clientes só visualizam resultados próximos ao final do projeto.
É fácil de gerenciar, devido a sua rigidez.	Atrasa a redução de riscos.
Realiza documentação extensa por cada fase ou estágio.	Apenas a fase final produz um artefato de software entregável.
Possibilita boa aderência a outros modelos de processo.	Cliente deve saber todos os requisitos no início do projeto.
Funciona bem com projetos pequenos e com requisitos bem conhecidos.	Não fornece feedback entre as fases.
	Pressupõe que os requisitos ficarão estáveis ao longo do tempo.
	Não funciona bem com projetos complexos e OO, apesar de compatível.

Ele é compatível, mas não é recomendado! *Por que, não?* Imagina um projeto super complexo que utiliza uma análise orientada a objetos (que é um modelo mais sofisticado que a análise estruturada). Lembre-se que, no Modelo em Cascata, você não pode errar, porque se você errar, os riscos de o projeto falhar são enormes! Por essa razão, ele não é recomendável, apesar de compatível!

Gabarito: E

19. (CESPE – 2004 – TRE/AL – Analista de Sistema) O modelo cascata ou ciclo de vida clássico necessita de uma abordagem sistemática, que envolve, em primeiro lugar, o projeto e, em seguida, a análise, a codificação, os testes e a manutenção.

Comentários:

Por Sommerville	Por Royce	Por Pressman (4ª Ed.)	Por Pressman (6ª Ed.)
Análise e Definição de Requisitos	Requisitos de Sistema	Modelagem e Engenharia do Sistema/Informação	Comunicação
Projeto de Sistema e Software	Requisitos de Software	Análise de Requisitos de Software	Planejamento
Implementação e Teste de Unidade	Análise	Projeto	Modelagem
Integração e Teste de Sistema	Projeto	Geração de Código	Construção
Operação e Manutenção	Codificação	Teste e Manutenção	Implantação
	Teste		
	Operação		

Primeiro Projeto e depois Análise? Não, Análise vem antes do Projeto!

Gabarito: E

20. (CESPE – 2008 – MPE/AM – Analista de Sistema) O modelo de desenvolvimento seqüencial linear tem como característica principal a produção de uma versão básica, mas funcional, do software desde as primeiras fases.

Comentários:

VANTAGENS	DESVANTAGENS

É simples de entender e fácil de aplicar, facilitando o planejamento.	Divisão inflexível do projeto em estágios distintos.
Fixa pontos específicos para a entrega de artefatos.	Dificuldade em incorporar mudanças de requisitos.
Funciona bem para equipes tecnicamente fracas.	**Clientes só visualizam resultados próximos ao final do projeto.**
É fácil de gerenciar, devido a sua rigidez.	Atrasa a redução de riscos.
Realiza documentação extensa por cada fase ou estágio.	**Apenas a fase final produz um artefato de software entregável.**
Possibilita boa aderência a outros modelos de processo.	Cliente deve saber todos os requisitos no início do projeto.
Funciona bem com projetos pequenos e com requisitos bem conhecidos.	Não fornece feedback entre as fases.
	Pressupõe que os requisitos ficarão estáveis ao longo do tempo.
	Não funciona bem com projetos complexos e OO, apesar de compatível.

Pelo contrário, somente nas fases finais que se tem uma versão! Essa definição está mais com cara de modelo de desenvolvimento em prototipagem.

Gabarito: E

---

21. (VUNESP - 2012 - SPTrans - Analista de Informática) Uma das abordagens do processo de desenvolvimento da engenharia de software prevê a divisão em etapas, em que o fim de uma é a entrada para a próxima. Esse processo é conhecido como modelo:

a) Transformação.
b) Incremental.
c) Evolutivo.
d) Espiral.
e) Cascata.

Comentários:

No Modelo em Cascata, uma fase só se inicia após o término e aprovação da fase anterior, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada após o término e aprovação da Fase 4. Mas que fases são essas? Bem, agora que complica, porque cada autor resolve criar suas fases!

Conforme vimos em aula, trata-se do Modelo em Cascata.

Gabarito: E

---

22. (CESGRANRIO – 2010 – PETROBRÁS – Analista de Sistemas – Processos de Negócio) No Ciclo de Vida Clássico, também chamado de Modelo Sequencial Linear ou Modelo Cascata, é apresentada uma abordagem sistemática composta pelas seguintes atividades:

a) Análise de Requisitos de Software, Projeto, Geração de Código, Teste e Manutenção.

b) Modelagem e Engenharia do Sistema/Informação, Análise de Requisitos de Software, Projeto, Geração de Código, Teste e Manutenção.

c) Modelagem e Engenharia do Sistema/Informação, Projeto, Geração de Código, Teste e Manutenção.

d) Levantamento de Requisitos de Software, Projeto, Geração de Código e Manutenção e Análise de Requisitos de Software.

e) Levantamento de Requisitos de Software, Projeto, Geração de Código, Teste Progressivo e Manutenção.

Comentários:

Por Sommerville	Por Royce	Por Pressman (4ª Ed.)	Por Pressman (6ª Ed.)
Análise e Definição de Requisitos	Requisitos de Sistema	Modelagem e Engenharia do Sistema/Informação	Comunicação
Projeto de Sistema e Software	Requisitos de Software	Análise de Requisitos de Software	Planejamento

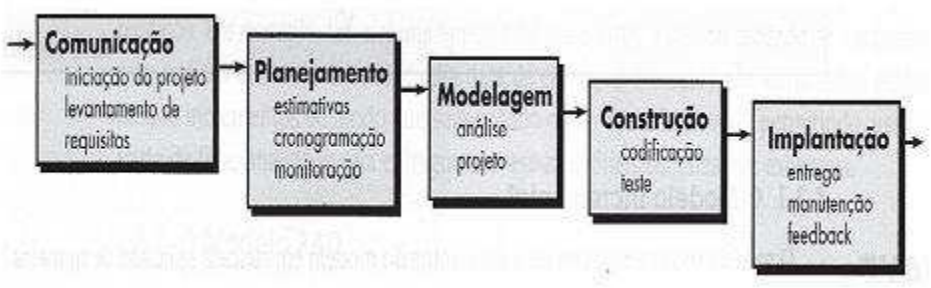
Implementação e Teste de Unidade	Análise	Projeto	Modelagem
Integração e Teste de Sistema	Projeto	Geração de Código	Construção
Operação e Manutenção	Codificação	Teste e Manutenção	Implantação
	Teste		
	Operação		

A Letra B está correta de acordo com o Pressman 4ª Edição, mas está errada de acordo com o Pressman 6ª Edição. Ademais, na questão ele sequer disse que era de acordo com o Pressman. Portanto, percebam que é um assunto polêmico e que as bancas deveriam ignorar, mas eventualmente elas cobram mesmo assim.

Gabarito: B

---

23. (FGV - 2015 – PGE/RO - Análise de Sistemas) A figura abaixo ilustra um modelo de processo, que prescreve um conjunto de elementos de processo como atividades de arcabouço, ações de engenharia de software, tarefas, produtos de trabalho, mecanismos de garantia de qualidade e de controle de modificações para cada projeto.



Esse modelo é conhecido como Modelo:

a) por funções.
b) em cascata.
c) incremental.
d) em pacotes.
e) por módulos.

Comentários:

Por Sommerville	Por Royce	Por Pressman (4ª Ed.)	Por Pressman (6ª Ed.)
Análise e Definição de Requisitos	Requisitos de Sistema	Modelagem e Engenharia do Sistema/Informação	Comunicação
Projeto de Sistema e Software	Requisitos de Software	Análise de Requisitos de Software	Planejamento
Implementação e Teste de Unidade	Análise	Projeto	Modelagem
Integração e Teste de Sistema	Projeto	Geração de Código	Construção
Operação e Manutenção	Codificação	Teste e Manutenção	Implantação
	Teste		
	Operação		

Conforme vimos em aula, trata-se das fases descritas pelo Pressman para o Modelo em Cascata.

Gabarito: B

---

24. (CESPE - 2016 – TCE/PR - Analista de Informática) O modelo de desenvolvimento em cascata é utilizado em caso de divergência nos requisitos de um software, para permitir a evolução gradual do entendimento dos requisitos durante a implementação do software.

Comentários:

Essa questão trata, na verdade, do modelo iterativo e incremental e, não, em cascata.

Gabarito: E

---

25. (CESGRANRIO – 2012 – Transpetro – Analista de Sistemas Júnior – Infra-estrutura) Na engenharia de software, existem diversos modelos de

desenvolvimento de software, e, dentre eles, o modelo em cascata, o qual, no contexto do desenvolvimento de sistemas de software complexos, recomenda:

a) distribuir a elicitação dos requisitos desde o início até o fim do desenvolvimento.

b) dividir o desenvolvimento do produto de software em fases lineares e sequenciais.

c) enfatizar a avaliação e mitigação de riscos durante o desenvolvimento.

d) realizar entregas incrementais do produto de software ao longo do desenvolvimento.

e) usar prototipagem rápida para estimular o envolvimento do usuário no desenvolvimento.

Comentários:

Recomenda dividir o desenvolvimento do produto de software em fases lineares e sequenciais. *Vocês se lembram que o Modelo em Cascata é um modelo sequencial linear?* Pois é!

Gabarito: B

---

26. (CESGRANRIO – 2012 – EPE – Analista de Gestão Corporativa – Tecnologia da Informação) Uma das críticas feitas ao modelo do ciclo de vida do desenvolvimento de software em cascata refere-se a:

a) exigência de conhecimento de avaliação e gerenciamento de risco para evitar grandes surpresas no projeto.

b) comprometimentos na qualidade e nas possibilidades de manutenção a longo prazo, parecendo um protótipo.

c) pouca flexibilidade para mudanças futuras, exigindo compromissos nas fases iniciais do projeto.

d) pouca visibilidade das etapas do processo, tornando cara a documentação de todas as versões dos sistemas.

e) exigências de velocidade as quais levam o engenheiro de software a utilizar linguagens, algoritmos ou ferramentas ineficientes ao longo de todo o projeto.

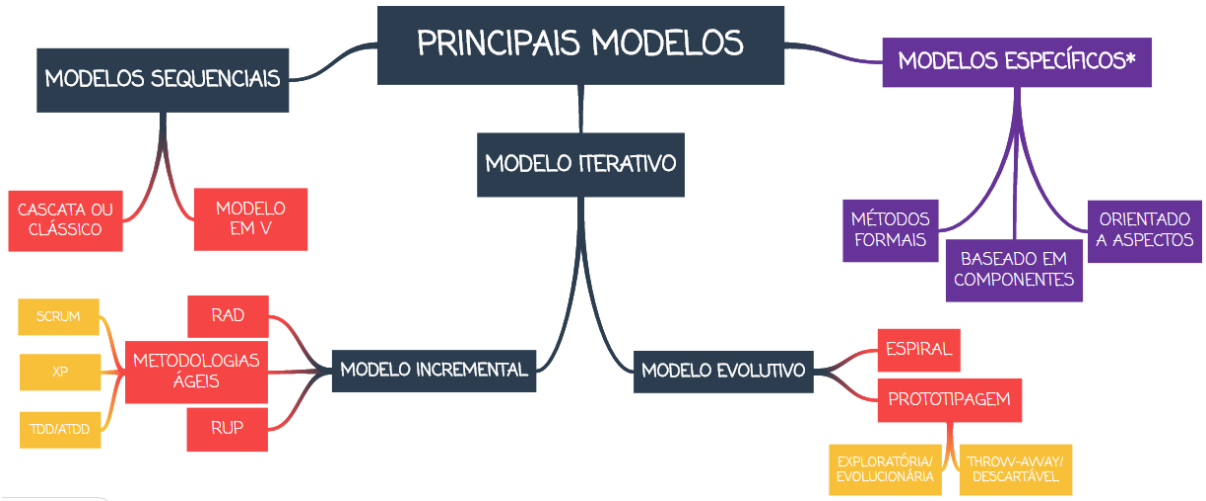
Comentários:

É conhecida a pouca flexibilidade do Modelo em Cascata em se adaptar a mudanças conforme o projeto progride. Mudanças nas fases iniciais do projeto têm menor impacto do que as que são necessárias nas fase finais do projeto. Por isso, o Modelo em Cascata é usado preferencialmente em projetos com requisitos muito bem definidos e com pouca volatilidade.

Gabarito: C

ACERTEI	ERREI

MODELOS ITERATIVOS E INCREMENTAIS



Como foi dito anteriormente, o Modelo em Cascata acumulava riscos e vários projetos começaram a fracassar ao utilizá-lo no mundo inteiro. Então surgiu o Modelo Iterativo como uma tentativa de resolver esse problema de acúmulo de riscos. Vejamos a diferença fundamental entre o Modelo em Cascata e o Modelo Iterativo:

- No Modelo em Cascata, caso haja cem requisitos, analisam-se os cem requisitos, projetam-se os cem requisitos, codificam-se os cem requisitos, testam-se os cem requisitos, e assim, por diante, sequencialmente.

- No Modelo Incremental, caso haja cem requisitos, dividem-se os cem requisitos em vinte miniprojetos de cinco requisitos e utiliza-se o modelo em cascata para cada miniprojeto.

- No Modelo Iterativo, caso haja cem requisitos, analisam-se, projetam-se, codificam-se, testam-se os cem requisitos, porém os requisitos são entregues incompletos e eu repito esse ciclo de refinamento até chegar ao produto final.

Galera, pensem só: é possível combinar a abordagem incremental com uma abordagem iterativa para desenvolver os miniprojetos e entregar partes diferentes do projeto. A imagem abaixo apresenta miniprojetos sendo feitos iterativamente em um modelo iterativo e incremental. Observem que cada iteração (passagem pelas fases de desenvolvimento) refinam a funcionalidade.

PRIMEIRA ITERAÇÃO



Assim, os resultados são mais rápidos, há maior interação com o usuário e há um feedback mais intenso – é possível reagir mais facilmente a mudanças. Essa abordagem permite gerenciamento e mitigação de riscos. *Professor, mas eu fiquei com uma dúvida: qual a diferença entre Modelo Iterativo e Modelo Incremental? Ou eles são exatamente a mesma coisa?*

Bem, galera... eu nunca vi nenhuma prova cobrar essa diferença entre modelo iterativo e modelo incremental! Na verdade, quando se fala em modelo iterativo, presume-se que é incremental (ou evolucionário) e quando se fala em modelo incremental, presume-se que é iterativo. Eles frequentemente andam lado a lado, mas há pequenas diferenças.

IMPORTANTE

Galera, eu já vi essas palavras serem trocadas dezenas de vezes (inclusive em edital). Infelizmente, algumas vezes as bancas não acatam os recursos! De todo modo, saibam que:

- **Iterativo:** reiterado ou repetitivo;
- **Interativo:** participação ou ação mútua.

Professor, mas e se cair em prova? Ora, caso caia em prova, a diferença é que, no modelo incremental, há várias equipes desenvolvendo uma parte do software a serem integradas no fim e, no modelo iterativo, lança-se a versão 1.0, adicionam-se funcionalidades, lança uma versão 2.0, adicionam-se mais funcionalidades e assim por diante.



Modelo Incremental: observem que a imagem mostra um artista com uma ideia completa sobre o quadro, mas ele desenvolve cada parte separadamente até integrar as partes em uma imagem completa. É como se fosse um quebra-cabeças em que cada parte é entregue funcionando e depois integrada. Produz builds, i.e., partes do software.



Modelo Iterativo: observem que a imagem mostra um artista com um esboço do quadro, sendo que ele desenvolve várias versões do quadro até chegar ao resultado final. É como se fosse uma visão abstrata da imagem, que em seguida vai sendo melhorada até chegar a uma visão mais concreta. Produz releases, i.e., versões constantemente melhoradas da imagem.

Uma das vantagens do modelo iterativo e incremental é que o cliente pode receber e avaliar as entregas do produto mais cedo, já no início do desenvolvimento do software. Além disso, há maior tolerância a mudanças com consequência direta de redução do risco de falha do projeto, i.e., ele acomoda melhor mudanças. Ele aumenta o reúso e a qualidade.



1. (CESPE - 2011 – TJ/ES - Analista Judiciário - Análise de Sistemas - Específicos) O modelo de processo incremental de desenvolvimento de software é iterativo, assim como o processo de prototipagem. Contudo, no processo incremental, diferentemente do que ocorre no de prototipagem, o objetivo consiste em apresentar um produto operacional a cada incremento.

Comentários:

De fato, no modelo iterativo e incremental, apresenta-se sempre um produto a cada incremento. Já na prototipação, não. Idealmente, ele serve apenas para identificar requisitos.

Gabarito: C

---

2. (CESPE - 2008 – TJ/DF - Analista Judiciário - Análise de Sistemas) No modelo de desenvolvimento incremental, embora haja defasagem entre os períodos de desenvolvimento de cada incremento, os incrementos são desenvolvidos em paralelo.

Comentários:

Questão perfeita. Os incrementos são codificados não exatamente em paralelo – há uma pequena defasagem.

Gabarito: C

---

3. (CESPE - 2009 – UNIPAMPA - Análise de Sistemas) No modelo de desenvolvimento incremental, a cada iteração são realizadas várias tarefas. Na fase de análise, pode ser feito o refinamento de requisitos e o refinamento do modelo conceitual.

Comentários:

Perfeito, é a fase seguinte à fase de requisitos e busca refiná-los.

Gabarito: C

---

4. (CESPE - 2016 – TCE/PR – Analista de Sistemas – C) No modelo iterativo de desenvolvimento de software, as atividades são dispostas em estágios sequenciais.

Comentários:

A questão trata do modelo em cascata de desenvolvimento de software. No modelo iterativo e incremental, as atividades são dispostas ao longo de diversas iterações

Gabarito: E

ACERTEI	ERREI

LISTA DE EXERCÍCIOS COMENTADOS (DIVERSAS BANCAS)
CONCEITOS BÁSICOS DE ENGENHARIA DE SOFTWARE

1. (CESPE - 2013 - TRT - 10ª REGIÃO (DF e TO) - Analista Judiciário - Tecnologia da Informação) A engenharia de software engloba processos, métodos e ferramentas. Um de seus focos é a produção de software de alta qualidade a custos adequados.

2. (FCC - 2012 - TST - Analista Judiciário - Análise de Sistemas) A Engenharia de Software:

   a) é uma área da computação que visa abordar de modo sistemático as questões técnicas e não técnicas no projeto, implantação, operação e manutenção no desenvolvimento de um software.

   b) consiste em uma disciplina da computação que aborda assuntos relacionados a técnicas para a otimização de algoritmos e elaboração de ambientes de desenvolvimento.

   c) trata-se de um ramo da TI que discute os aspectos técnicos e empíricos nos processos de desenvolvimento de sistemas, tal como a definição de artefatos para a modelagem ágil.

   d) envolve um conjunto de itens que abordam os aspectos de análise de mercado, concepção e projeto de software, sendo independente da engenharia de um sistema.

   e) agrupa as melhores práticas para o concepção, projeto, operação e manutenção de artefatos que suportam a execução de programas de computador, tais como as técnicas de armazenamento e as estruturas em memória principal.

3. (FCC - 2012 - TRT - 6ª Região (PE) - Técnico Judiciário - Tecnologia da Informação) Considere: *é uma disciplina que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação. Seu principal objetivo é fornecer uma estrutura metodológica para a construção de software com alta qualidade.* A definição refere-se:

a) ao ciclo de vida do software.
b) à programação orientada a objetos.
c) à análise de sistemas.
d) à engenharia de requisitos.
e) à engenharia de software.

4. (CESPE - 2011 - MEC - Gerente de Projetos) A engenharia de software, disciplina relacionada aos aspectos da produção de software, abrange somente os processos técnicos do desenvolvimento de software.

5. (FGV - 2010 - BADESC - Analista de Sistemas - Desenvolvimento de Sistemas) De acordo com Pressman, a engenharia de software é baseada em camadas, com foco na qualidade.

   Essas camadas são:

   a) métodos, processo e teste.
   b) ferramentas, métodos e processo.
   c) métodos, construção, teste e implantação.
   d) planejamento, modelagem, construção, validação e implantação.
   e) comunicação, planejamento, modelagem, construção e implantação.

6. (CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação) Os princípios de engenharia de software definem a necessidade de formalidades para reduzir inconsistências e a decomposição para lidar com a complexidade.

7. (FCC - 2010 - TRE-AM - Analista Judiciário - Tecnologia da Informação - A) A Engenharia de Software:

   a) não tem como método a abordagem estruturada para o desenvolvimento de software, pois baseia-se exclusivamente nos modelos de software, notações, regras e técnicas de desenvolvimento.

   b) se confunde com a Ciência da Computação quando ambas tratam do desenvolvimento de teorias, fundamentações e práticas de desenvolvimento de software.

c) tendo como foco apenas o tratamento dos aspectos de construção de software, subsidia a Engenharia de Sistemas no tratamento dos sistemas baseados em computadores, incluindo hardware e software.

d) tem como foco principal estabelecer uma abordagem sistemática de desenvolvimento, através de ferramentas e técnicas apropriadas, dependendo do problema a ser abordado, considerando restrições e recursos disponíveis.

e) segue princípios, tais como, o da Abstração, que identifica os aspectos importantes sem ignorar os detalhes e o da Composição, que agrupa as atividades em um único processo para distribuição aos especialistas.

8. **(FCC - 2011 - INFRAERO - Analista de Sistemas - Gestão de TI)** Em relação à Engenharia de Software, é INCORRETO afirmar:

a) O design de software, ao descrever os diversos aspectos que estarão presentes no sistema quando construído, permite que se faça a avaliação prévia para garantir que ele alcance os objetivos propostos pelos interessados.

b) A representação de um design de software mais simples para representar apenas as suas características essenciais busca atender ao princípio da abstração.

c) Iniciar a entrevista para obtenção dos requisitos de software com perguntas mais genéricas e finalizar com perguntas mais específicas sobre o sistema é o que caracteriza a técnica de entrevista estruturada em funil.

d) No contexto de levantamento de requisitos, funcionalidade é um dos aspectos que deve ser levado em conta na abordagem dos requisitos funcionais.

e) A representação é a linguagem do design, cujo único propósito é descrever um sistema de software que seja possível construir.

9. **(FCC – 2009 – AFR/SP - Analista de Sistemas)** A engenharia de software está inserida no contexto:

a) das engenharias de sistemas, de processo e de produto.
b) da engenharia de sistemas, apenas.
c) das engenharias de processo e de produto, apenas.
d) das engenharias de sistemas e de processo, apenas.
e) das engenharias de sistemas e de produto, apenas.

10. (CESPE – 2015 – STJ – Analista de Sistemas) Embora os engenheiros de software geralmente utilizem uma abordagem sistemática, a abordagem criativa e menos formal pode ser eficiente em algumas circunstâncias, como, por exemplo, para o desenvolvimento de sistemas web, que requerem uma mistura de habilidades de software e de projeto.

11. (CESPE – 2015 – STJ – Analista de Sistemas) O foco da engenharia de software inclui especificação do sistema, desenvolvimento de hardware, elaboração do projeto de componentes de hardware e software, definição dos processos e implantação do sistema.

12. (FUNIVERSA – 2009 – IPHAN – Analista de Sistemas) A Engenharia de Software resume-se em um conjunto de técnicas utilizadas para o desenvolvimento e manutenção de sistemas computadorizados, visando produzir e manter softwares de forma padronizada e com qualidade. Ela obedece a alguns princípios como (1) Formalidade, (2) Abstração, (3) Decomposição, (4) Generalização e (5) Flexibilização. Assinale a alternativa que apresenta conceito correto sobre os princípios da Engenharia de Software.

    a) A flexibilização é o processo que permite que o software possa ser alterado, sem causar problemas para sua execução.

    b) A formalidade é a maneira usada para resolver um problema, de forma genérica, com o intuito de poder reaproveitar essa solução em outras situações semelhantes.

    c) A generalização preocupa-se com a identificação de um determinado fenômeno da realidade, sem se preocupar com detalhes, considerando apenas os aspectos mais relevantes.

    d) Pelo princípio da decomposição, o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva.

    e) A abstração é a técnica de se dividir o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica.

13. (CESPE – 2013 – TCE/RO – Analista de Sistemas) Engenharia de software não está relacionada somente aos processos técnicos de desenvolvimento de softwares,

mas também a atividades como gerenciamento de projeto e desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de softwares.

14. **(CESPE – 2010 – DETRAN/ES – Analista de Sistemas)** Segundo princípio da engenharia de software, os vários artefatos produzidos ao longo do seu ciclo de vida apresentam, de forma geral, nível de abstração cada vez menor.

15. **(CESPE – 2010 – TRE/BA – Analista de Sistemas)** Entre os desafios enfrentados pela engenharia de software estão lidar com sistemas legados, atender à crescente diversidade e atender às exigências quanto a prazos de entrega reduzidos.

16. **(FCC – 2010 – DPE/SP – Analista de Sistemas)** A Engenharia de Software

I. não visa o desenvolvimento de teorias e fundamentações, preocupando-se unicamente com as práticas de desenvolvimento de software.

II. tem como foco o tratamento dos aspectos de desenvolvimento de software, abstraindo-se dos sistemas baseados em computadores, incluindo hardware e software.

III. tem como métodos as abordagens estruturadas para o desenvolvimento de software que incluem os modelos de software, notações, regras e maneiras de desenvolvimento.

IV. segue princípios, tais como, o da Abstração, que identifica os aspectos importantes sem ignorar os detalhes e o da Composição, que agrupa as atividades em um único processo para distribuição aos especialistas.

É correto o que se afirma em:

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

17. **(CESPE – 2013 – TCE/RO – Analista de Sistemas)** Assim como a Engenharia de Software, existe também na área de informática a chamada Ciência da

Computação. Assinale a alternativa que melhor apresenta a diferença entre Engenharia de Software e Ciência da Computação.

a) A Ciência da Computação tem como objetivo o desenvolvimento de teorias e fundamentações. Já a Engenharia de Software se preocupa com as práticas de desenvolvimento de software.

b) A Engenharia de Software trata da criação dos sistemas de computação (softwares) enquanto a Ciência da Computação está ligada ao desenvolvimento e criação de componentes de hardware.

c) A Engenharia de Software trata dos sistemas com base em computadores, que inclui hardware e software, e a Ciência da Computação trata apenas dos aspectos de desenvolvimento de sistemas.

d) A Ciência da Computação trata dos sistemas com base em computadores, que inclui hardware e software, e a Engenharia de Software trata apenas dos aspectos de desenvolvimento de sistemas.

e) A Ciência da Computação destina-se ao estudo e solução para problemas genéricos das áreas de rede e banco de dados e a Engenharia de Software restringe- se ao desenvolvimento de sistemas.

18. **(CESPE – 2009 – ANAC – Analista de Sistemas)** O termo engenharia pretende indicar que o desenvolvimento de software submete-se a leis similares às que governam a manufatura de produtos industriais em engenharias tradicionais, pois ambos são metodológicos.

19. **(CESPE - 2016 – TCE/PR – Analista de Sistemas – B)** A engenharia de software refere-se ao estudo das teorias e fundamentos da computação, ficando o desenvolvimento de software a cargo da ciência da computação.

20. **(CESPE - 2016 – TCE/PR – Analista de Sistemas – E)** O conceito de software se restringe ao desenvolvimento do código em determinada linguagem e seu armazenamento em arquivos.

LISTA DE EXERCÍCIOS COMENTADOS (DIVERSAS BANCAS)
CICLO DE VIDA E PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

1. (CESPE – 2009 – TCE/TO – Analista de Sistemas - A) Quanto maior e mais complexo o projeto de software, mais simples deve ser o modelo de processo a ser adotado.

2. (CESPE – 2009 – TCE/TO – Analista de Sistemas - B) O modelo de ciclo de vida do software serve para delimitar o alvo do software. Nessa visão, não são consideradas as atividades necessárias e o relacionamento entre elas.

3. (CESPE – 2009 – TCE/TO – Analista de Sistemas - C) A escolha do modelo do ciclo de vida não depende de características específicas do projeto, pois o melhor modelo é sempre o mais usado pela equipe do projeto.

4. (ESAF - 2012 – CGU – Analista de Sistemas) A escolha de um modelo é fortemente dependente das características do projeto. Os principais modelos de ciclo de vida podem ser agrupados em três categorias principais:

   a) sequenciais, cascata e evolutivos.
   b) sequenciais, incrementais e ágeis.
   c) sequenciais, incrementais e evolutivos.
   d) sequenciais, ágeis e cascata.
   e) cascata, ágeis e evolutivos.

5. (CESPE – 2011 – MEC – Analista de Sistemas) O processo de desenvolvimento de software é uma caracterização descritiva ou prescritiva de como um produto de software deve ser desenvolvido.

6. (CESPE – 2013 – TRT/10ª – Analista de Sistemas) As atividades fundamentais relacionadas ao processo de construção de um software incluem a especificação, o desenvolvimento, a validação e a evolução do software.

7. (CESPE – 2010 – TRE/BA – Analista de Sistemas) Um modelo de processo de software consiste em uma representação complexa de um processo de software, apresentada a partir de uma perspectiva genérica.

8. (CESPE – 2011 – MEC – Analista de Sistemas) Atividades comuns a todos os processos de software incluem a especificação, o projeto, a implementação e a validação.

9. (CESPE – 2015 – STJ – Analista de Sistemas) As principais atividades de engenharia de software são especificação, desenvolvimento, validação e evolução.

10. (FCC – 2012 – MPE/AP – Analista de Sistemas) Um processo de software é um conjunto de atividades relacionadas que levam à produção de um produto de software. Existem muitos processos de software diferentes, mas todos devem incluir quatro atividades fundamentais: especificação, projeto e implementação, validação e:

    a) teste
    b) evolução.
    c) prototipação.
    d) entrega.
    e) modelagem.

11. (CESPE – 2010 – EMBASA – Analista de Sistemas) Ciclo de vida de um software resume-se em eventos utilizados para definir o status de um projeto.

12. (CESPE – 2016 – TCE/PR – Analista de Sistemas) As fases do ciclo de vida de um software são:

    a) concepção, desenvolvimento, entrega e encerramento.
    b) iniciação, elaboração, construção e manutenção.
    c) escopo, estimativas, projeto e processo e gerência de riscos.
    d) análise, desenvolvimento, teste, empacotamento e entrega.
    e) planejamento, análise e especificação de requisitos, projeto, implementação, testes, entrega e implantação, operação e manutenção.

13. (MPE/RS – 2012 – MPE/RS – Analista de Sistemas) O ciclo de vida básico de um software compreende:

    a) a implementação, a implantação e o teste.
    b) a análise, a segurança e o controle de usuários.
    c) a implementação, a análise e o teste.
    d) a implementação, a validação e as vendas.

e) a análise, o projeto, a implementação e o teste.

14. (CESGRANRIO – 2011 – PETROBRÁS – Analista de Sistemas) A especificação de uma Metodologia de Desenvolvimento de Sistemas tem como pré-requisito indispensável, em relação ao que será adotado no processo de desenvolvimento, a definição do:

    a) Engenheiro Responsável pelo Projeto
    b) Documento de Controle de Sistemas
    c) Software para Desenvolvimento
    d) Ciclo de Vida do Software
    e) Bloco de Atividades

15. (CESPE – 2008 – INPE – Analista de Sistemas) O ciclo de vida do software tem início na fase de projeto.

16. (CESPE – 2013 – TRT/10 – Analista de Sistemas) O ciclo de vida de um software, entre outras características, está relacionado aos estágios de concepção, projeto, criação e implementação.

17. (CESPE – 2011 – TJ/ES – Analista de Sistemas) Entre as etapas do ciclo de vida de software, as menos importantes incluem a garantia da qualidade, o projeto e o estudo de viabilidade. As demais atividades do ciclo, como a implementação e os testes, requerem maior dedicação da equipe e são essenciais.

18. (CESPE - 2016 – TCE/PR – Analista de Sistemas – A) A engenharia de software está relacionada aos diversos aspectos de produção de software e inclui as atividades de especificação, desenvolvimento, validação e evolução de software.

19. (CESPE - 2016 – TCE/PR – Analista de Sistemas – D) Um processo de software é composto por quatro atividades fundamentais: iniciação, desenvolvimento, entrega e encerramento.

20. (CESPE - 2016 – TCE/PR – Analista de Sistemas – B) A metodologia de processos genérica para a engenharia de software é composta de seis etapas: comunicação, planejamento, modelagem, construção, emprego e aceitação.

21. (INSTITUTO CIDADE – 2012 – TCM/GO – Analista de Sistemas) De acordo com a engenharia de software, como todo produto industrial, o software possui um ciclo de vida. Cada fase do ciclo de vida possui divisões e subdivisões. Em qual

fase avaliamos a necessidade de evolução dos softwares em funcionamento para novas plataformas operacionais ou para a incorporação de novos requisitos?

a) Fase de operação;
b) Fase de retirada;
c) Fase de definição;
d) Fase de design.
e) Fase de desenvolvimento;

22. (CESPE - 2010 – DETRAN/ES – Analista de Sistemas) Quando um aplicativo de software desenvolvido em uma organização atinge, no fim do seu ciclo de vida, a fase denominada aposentadoria, descontinuação ou fim de vida, todos os dados por ele manipulados podem ser descartados.

LISTA DE EXERCÍCIOS COMENTADOS (DIVERSAS BANCAS)

MODELO EM CASCATA

1. (FCC - 2012 - TJ-RJ - Analista Judiciário - Análise de Sistemas - E) Dos diferentes modelos para o ciclo de vida de desenvolvimento de um software é correto afirmar que o modelo em cascata é o mais recente e complexo.

2. (FCC - 2009 - SEFAZ-SP - Agente Fiscal de Rendas - Tecnologia da Informação - Prova 3 - B) O processo de engenharia de software denominado ciclo de vida clássico refere-se ao modelo incremental.

3. (CESPE – 2009 – INMETRO – Analista de Sistemas) Em uma empresa que tenha adotado um processo de desenvolvimento de software em cascata, falhas no levantamento de requisitos têm maior possibilidade de gerar grandes prejuízos do que naquelas que tenham adotado desenvolvimento evolucionário.

4. (CESPE – 2011 – MEC – Analista de Sistemas) O modelo Waterfall tem a vantagem de facilitar a realização de mudanças sem a necessidade de retrabalho em fases já completadas.

5. (CESPE – 2008 – TST – Analista de Sistemas) No modelo de desenvolvimento sequencial linear, a fase de codificação é a que gera erros de maior custo de correção.

6. (CESPE – 2009 – INMETRO – Analista de Sistemas) Em um processo de desenvolvimento em cascata, os testes de software são realizados todos em um mesmo estágio, que acontece após a finalização das fases de implementação.

7. (CESPE – 2008 – SERPRO – Analista de Sistemas) O modelo em cascata consiste de fases e atividades que devem ser realizadas em sequência, de forma que uma atividade é requisito da outra.

8. (CESPE – 2005 – AL/ES – Analista de Sistemas - B) O modelo de desenvolvimento em cascata descreve ciclos sequenciais, incrementais e iterativos, possuindo, entre outras, as fases de requisitos e implementação.

9. (CESPE – 2004 – STJ – Analista de Sistemas) O modelo de desenvolvimento seqüencial linear, também chamado modelo clássico ou modelo em cascata, caracteriza-se por não acomodar adequadamente as incertezas que existem no início de um projeto de software, em especial as geradas pela dificuldade do cliente de explicitar todos os requerimentos que o programa deve contemplar.

10. (CESPE – 2009 – IPEA – Analista de Sistema) No modelo em cascata de processo de desenvolvimento, os clientes devem definir os requisitos apenas durante a fase de projeto; e os projetistas definem as estratégias de projeto apenas durante a fase de implementação. As fases do ciclo de vida envolvem definição de requisitos, projeto, implementação, teste, integração, operação e manutenção. Em cada fase do ciclo de vida, podem ser produzidos diversos artefatos.

11. (CESPE – 2008 – TCE/TO – Analista de Sistema – D) No ciclo de vida em cascata, é possível realizar alternadamente e simultaneamente as atividades de desenvolvimento de software.

12. (CESPE – 2004 – TJ/PA – Analista de Sistema – D) A abordagem sistemática estritamente linear para o desenvolvimento de software é denominada modelo em cascata ou modelo sequencial linear.

13. (CESPE – 2006 – TSE – Analista de Sistema – D) O modelo em cascata organiza o desenvolvimento em fases. Esse modelo encoraja a definição dos requisitos antes do restante do desenvolvimento do sistema. Após a especificação e a análise dos requisitos, têm-se o projeto, a implementação e o teste.

14. (CESPE – 2009 – INMTRO – Analista de Sistema) No desenvolvimento de software, o modelo em cascata é estruturado de tal maneira que as fases que

compõem o desenvolvimento são interligadas. Nessa situação, o final de uma fase implica o início de outra.

15. (CESPE – 2010 – BASA – Analista de Sistema) No modelo em cascata, o projeto segue uma série de passos ordenados. Ao final de cada projeto, a equipe de projeto finaliza uma revisão. O desenvolvimento continua e, ao final, o cliente avalia a solução proposta.

16. (CESPE – 2009 – TRE/MT – Analista de Sistema - A) O modelo em cascata é apropriado para software em que os requisitos ainda não foram bem compreendidos, pois é focado na criação de incrementos.

17. (CESPE – 2009 – UNIPAMPA – Analista de Sistema – D) O modelo em cascata sugere uma abordagem sistemática e sequencial para o desenvolvimento de software. Sua natureza linear leva a estados de bloqueio nos quais, para que nova etapa seja iniciada, é necessário que a documentação associada à fase anterior tenha sido aprovada.

18. (CESPE – 2004 – ABIN – Analista de Sistema) O modelo de desenvolvimento seqüencial linear, também denominado modelo em cascata, é incompatível com o emprego de técnica de análise orientada a objetos no desenvolvimento de um sistema de informação.

19. (CESPE – 2004 – TRE/AL – Analista de Sistema) O modelo cascata ou ciclo de vida clássico necessita de uma abordagem sistemática, que envolve, em primeiro lugar, o projeto e, em seguida, a análise, a codificação, os testes e a manutenção.

20. (CESPE – 2008 – MPE/AM – Analista de Sistema) O modelo de desenvolvimento seqüencial linear tem como característica principal a produção de uma versão básica, mas funcional, do software desde as primeiras fases.

21. (VUNESP - 2012 - SPTrans - Analista de Informática) Uma das abordagens do processo de desenvolvimento da engenharia de software prevê a divisão em etapas, em que o fim de uma é a entrada para a próxima. Esse processo é conhecido como modelo:

a) Transformação.
b) Incremental.
c) Evolutivo.
d) Espiral.

e) Cascata.

22. **(CESGRANRIO – 2010 – PETROBRÁS – Analista de Sistemas – Processos de Negócio)** No Ciclo de Vida Clássico, também chamado de Modelo Sequencial Linear ou Modelo Cascata, é apresentada uma abordagem sistemática composta pelas seguintes atividades:

a) Análise de Requisitos de Software, Projeto, Geração de Código, Teste e Manutenção.

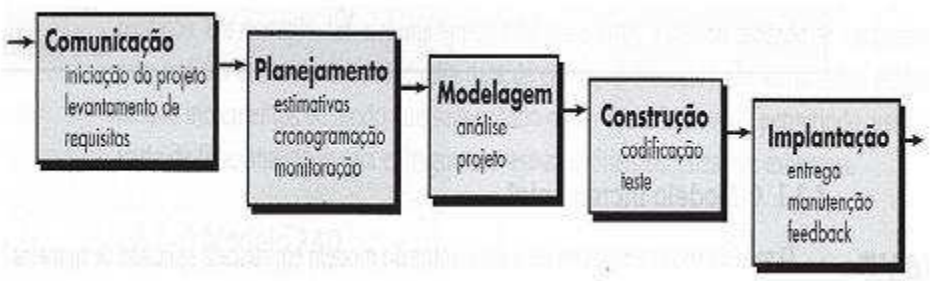
b) Modelagem e Engenharia do Sistema/Informação, Análise de Requisitos de Software, Projeto, Geração de Código, Teste e Manutenção.

c) Modelagem e Engenharia do Sistema/Informação, Projeto, Geração de Código, Teste e Manutenção.

d) Levantamento de Requisitos de Software, Projeto, Geração de Código e Manutenção e Análise de Requisitos de Software.

e) Levantamento de Requisitos de Software, Projeto, Geração de Código, Teste Progressivo e Manutenção.

23. **(FGV - 2015 – PGE/RO - Análise de Sistemas)** A figura abaixo ilustra um modelo de processo, que prescreve um conjunto de elementos de processo como atividades de arcabouço, ações de engenharia de software, tarefas, produtos de trabalho, mecanismos de garantia de qualidade e de controle de modificações para cada projeto.



Esse modelo é conhecido como Modelo:

a) por funções.
b) em cascata.

c) incremental.
d) em pacotes.
e) por módulos.

24. (CESPE - 2016 – TCE/PR - Analista de Informática) O modelo de desenvolvimento em cascata é utilizado em caso de divergência nos requisitos de um software, para permitir a evolução gradual do entendimento dos requisitos durante a implementação do software.

25. (CESGRANRIO – 2012 – Transpetro – Analista de Sistemas Júnior – Infra-estrutura) Na engenharia de software, existem diversos modelos de desenvolvimento de software, e, dentre eles, o modelo em cascata, o qual, no contexto do desenvolvimento de sistemas de software complexos, recomenda:

a) distribuir a elicitação dos requisitos desde o início até o fim do desenvolvimento.

b) dividir o desenvolvimento do produto de software em fases lineares e sequenciais.

c) enfatizar a avaliação e mitigação de riscos durante o desenvolvimento.

d) realizar entregas incrementais do produto de software ao longo do desenvolvimento.

e) usar prototipagem rápida para estimular o envolvimento do usuário no desenvolvimento.

26. (CESGRANRIO – 2012 – EPE – Analista de Gestão Corporativa – Tecnologia da Informação) Uma das críticas feitas ao modelo do ciclo de vida do desenvolvimento de software em cascata refere-se a:

a) exigência de conhecimento de avaliação e gerenciamento de risco para evitar grandes surpresas no projeto.

b) comprometimentos na qualidade e nas possibilidades de manutenção a longo prazo, parecendo um protótipo.

c) pouca flexibilidade para mudanças futuras, exigindo compromissos nas fases iniciais do projeto.

d) pouca visibilidade das etapas do processo, tornando cara a documentação de todas as versões dos sistemas.

e) exigências de velocidade as quais levam o engenheiro de software a utilizar linguagens, algoritmos ou ferramentas ineficientes ao longo de todo o projeto.

LISTA DE EXERCÍCIOS COMENTADOS (DIVERSAS BANCAS)

MODELOS ITERATIVOS E INCREMENTAIS

1. (CESPE - 2011 – TJ/ES - Analista Judiciário - Análise de Sistemas - Específicos) O modelo de processo incremental de desenvolvimento de software é iterativo, assim como o processo de prototipagem. Contudo, no processo incremental, diferentemente do que ocorre no de prototipagem, o objetivo consiste em apresentar um produto operacional a cada incremento.

2. (CESPE - 2008 – TJ/DF - Analista Judiciário - Análise de Sistemas) No modelo de desenvolvimento incremental, embora haja defasagem entre os períodos de desenvolvimento de cada incremento, os incrementos são desenvolvidos em paralelo.

3. (CESPE - 2009 – UNIPAMPA - Análise de Sistemas) No modelo de desenvolvimento incremental, a cada iteração são realizadas várias tarefas. Na fase de análise, pode ser feito o refinamento de requisitos e o refinamento do modelo conceitual.

4. (CESPE - 2016 – TCE/PR – Analista de Sistemas – C) No modelo iterativo de desenvolvimento de software, as atividades são dispostas em estágios sequenciais.

GABARITO DOS EXERCÍCIOS COMENTADOS (DIVERSAS BANCAS)
CONCEITOS BÁSICOS DE ENGENHARIA DE SOFTWARE

1	2	3	4	5	6	7	8	9	10
C	A	E	E	B	C	D	E	A	C
11	12	13	14	15	16	17	18	19	20
E	A	C	C	C	D	A	C	E	E

GABARITO DOS EXERCÍCIOS COMENTADOS (DIVERSAS BANCAS)
CICLO DE VIDA E PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

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

GABARITO DOS EXERCÍCIOS COMENTADOS (DIVERSAS BANCAS)
MODELO EM CASCATA

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

GABARITO DOS EXERCÍCIOS COMENTADOS (DIVERSAS BANCAS)
MODELOS ITERATIVOS E INCREMENTAIS

1	2	3	4	5	6	7	8	9	10
C	C	C	E						

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.