

Grupo Handbook

Handbook de Questões de TI

comentadas para CONCURSOS

Além do gabarito

Volume 5

*Desenvolvimento de Software
Parte 1*

Fundação Cesgranrio

Prefácio

Este é o volume 5 da série *Handbook de Questões de TI Comentadas para Concursos – Além do Gabarito*, que traz para você uma coletânea especial de questões da área de desenvolvimento de software selecionadas de provas organizadas pela Fundação Cesgranrio.

Entre as provas das quais as questões foram selecionadas estão as últimas para Analista de Desenvolvimento do BNDES, para Analista de Desenvolvimento da BR Distribuidora, e para Analista de Processos de Negócio da Petrobras, todas aplicadas no ano de 2008.

Notoriamente, a Fundação Cesgranrio vem sendo a responsável pela organização de concursos de TI de alta concorrência e visibilidade, como os do BNDES, e os da Petrobras e suas subsidiárias. Mais recentemente, a Fundação Cesgranrio também foi a escolhida para realizar o concurso do Banco Central, um dos mais concorridos na área de TI.

Uma interessante característica das provas aplicadas pela Fundação Cesgranrio é o fato das questões possuírem pesos diferentes, o que exige dos candidatos ainda mais conhecimento e sagacidade. Além disso, é comum que sejam cobrados temas recentes de TI, como forma de selecionar profissionais atualizados e dinâmicos.

Por isso, o Grupo Handbook de TI preparou para você uma edição especial com questões da área de desenvolvimento de software, abordando temas clássicos e modernos da computação. Esse volume é especialmente útil para todos que desejam prestar concursos na área de desenvolvimento, em especial os organizados pela Fundação Cesgranrio.

Bons estudos,

Grupo Handbook de TI

Direitos Autorais

Este material é registrado no Escritório de Direitos Autorais (EDA) da Fundação Biblioteca Nacional. Todos os direitos autorais referentes a esta obra são reservados exclusivamente aos seus autores.

Os autores deste material não proíbem seu compartilhamento entre amigos e colegas próximos de estudo. Contudo, a reprodução, parcial ou integral, e a disseminação deste material de forma indiscriminada através de qualquer meio, inclusive na Internet, extrapolam os limites da colaboração. Essa prática desincentiva o lançamento de novos produtos e enfraquece a comunidade concurseira Handbook de TI.

A série *Handbook de Questões de TI Comentadas para Concursos – Além do Gabarito* é uma produção independente e contamos com você para mantê-la sempre viva.

Grupo Handbook de TI

Canais de Comunicação

A equipe Handbook de TI disponibiliza diversos canais de comunicação para seus clientes.

Loja Handbook de TI

<http://www.handbookdeti.com.br>

Serviço de Atendimento

Comunicação direta com a Equipe Handbook de TI pode ser feita em
<http://www.handbookdeti.com.br/contacts>

Twitter do Handbook de TI

Que acompanhar de perto o trabalho do Grupo Handbook de TI. Cadastre-se no twitter e comece a seguir o grupo Handbook de TI em <http://twitter.com/handbookdeti>

1. **Assuntos relacionados:** *Desenvolvimento de Sistemas, Metodologia de Desenvolvimento de Software, Extreme Programming (XP)*,

Banca: Cesgranrio

Instituição: BNDES

Cargo: Analista de Sistemas - Desenvolvimento

Ano: 2008

Questão: 31

Que situação favorece a escolha do uso de XP para um projeto de desenvolvimento de software, em oposição à escolha do RUP ou do modelo Cascata?

- (a). Equipe do projeto localizada em diferentes cidades e com poucos recursos de colaboração.
- (b). Equipe do projeto formada por pessoas com alto grau de competitividade.
- (c). Cliente do projeto trabalhando em parceria com a equipe do projeto e sempre disponível para retirar dúvidas.
- (d). Requisitos do software com pequena probabilidade de mudanças.
- (e). Presença de um processo organizacional que exige a elaboração de vários documentos específicos para cada projeto.

Solução:

O Rational Unified Process (RUP) é um framework muito difundido e utilizado que pode ser adaptado a vários tipos de projetos de software. Podem ser derivados do RUP processos para projetos de vários portes, pois este framework define uma grande lista de papéis, artefatos, atividades e fluxos. No entanto, o RUP é tido como muito complexo, e costuma ser visto como um pesado e burocrático, ao contrário das metodologias ágeis.

Em contrapartida, o EXtreme Programming (XP) aparece como uma alternativa mais leve para times de tamanho pequeno e médio porte, que desenvolvem software em um contexto de requisitos vagos e rapidamente modificados. O XP enfatiza a codificação e os testes de códigos, considerando a presença constante dos clientes no desenvolvimento fundamental. Pela característica de simplicidade que esta técnica apresenta, poucos artefatos, papéis e atividades são definidos. Portanto, a resposta da questão é a alternativa C.

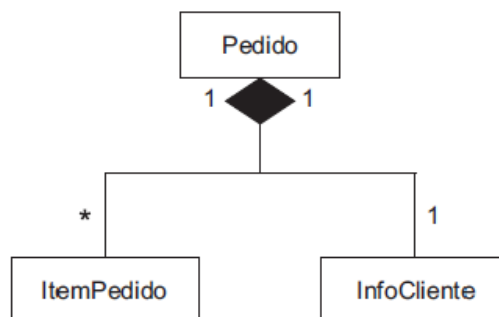
Ainda sobre o XP, são muito úteis ainda as seguintes informações. Os quatro valores fundamentais da metodologia XP são a comunicação, simplicidade, feedback e coragem. Para fazer valer tais valores, a metodologia adota como práticas as seguintes:

- **Jogo de Planejamento:** O desenvolvimento é feito em iterações semanais. No início da semana, desenvolvedores e cliente reúnem-se para priorizar as funcionalidades. O escopo é reavaliado semanalmente, e o projeto é regido por um contrato de escopo negociável;
- **Pequenas Versões:** A liberação de pequenas versões funcionais do projeto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando;
- **Metáforas:** A metodologia prega o uso de metáforas, como forma de facilitar a comunicação com o clientes;
- **Time Coeso:** A equipe de desenvolvimento é formada pelo cliente e pela equipe de desenvolvimento;

- **Ritmo Sustentável:** Trabalhar com qualidade, buscando ter ritmo de trabalho saudável, sem horas extras. Horas extras são permitidas quando trouxerem produtividade para a execução do projeto;
- **Reuniões em pé:** Reuniões em pé para não se perder o foco nos assuntos, produzindo reuniões rápidas, apenas abordando tarefas realizadas e tarefas a realizar pela equipe;
- **Posse Coletiva do Código:** O código fonte não tem dono e ninguém precisa solicitar permissão para poder modificar o mesmo. Tal prática tem como objetivo fazer com que a equipe conheça todas as partes do sistema;
- **Programação em Duplas:** Geralmente a dupla é formada por um iniciante na linguagem e outra pessoa funcionando como um instrutor. Como é apenas um computador, o novato é que fica à frente fazendo a codificação, e o instrutor acompanha ajudando a desenvolver suas habilidades. Desta forma o programa sempre é revisto por duas pessoas, evitando e diminuindo assim a possibilidade de erros;
- **Padrões de Codificação:** A equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir estas regras. Desta forma, parecerá que todo o código foi editado pela mesma pessoa, mesmo quando a equipe possui uma quantidade maior de programadores;
- **Refatoração:** É um processo que permite a melhoria contínua da programação, que tem como alvos a melhoria da legibilidade do código, maiores modularização e reaproveitamento do código.

2. **Assuntos relacionados:** *UML, Diagrama de Classes, Composição Agregada,***Banca:** *Cesgranrio***Instituição:** *BNDES***Cargo:** *Analista de Sistemas - Desenvolvimento***Ano:** *2008***Questão:** *35*

Considere o relacionamento de “todo-parte” ilustrado no diagrama UML abaixo.



É correto afirmar que

- um objeto da classe **InfoCliente** pode participar de mais de um relacionamento de composição desempenhando o papel de “parte”.
- um objeto da classe **ItemPedido** pode participar de mais de um relacionamento de composição desempenhando o papel de “parte”.
- uma instância da classe **InfoCliente** pode existir antes mesmo que a instância da classe **Pedido** com que se relacionará tenha sido criada.
- o relacionamento ilustrado acima é ternário.
- a cardinalidade do pedido no relacionamento com **ItemPedido** igual a **1** não precisaria ser apresentada, uma vez que não poderia assumir outro valor.

Solução:

Lembre-se que diagramas de classe nos permitem identificar tanto o conteúdo de uma classe quanto o relacionamento entre várias classes. Em um diagrama de classe, podemos mostrar as variáveis e métodos membros de uma classe. Podemos também mostrar se uma classe herda de outra, ou se mantém uma referência para outra.

O relacionamento “todo-parte”, também conhecido como composição agregada, ou relacionamento “tem-um” ou “parte-de”, indica que um objeto (o todo) é composto de outros objetos (as partes). Com a composição agregada, o relacionamento entre os objetos é muito mais forte que com a associação, pois nela o todo não pode existir sem suas partes e as partes não podem existir sem o todo. Vários pontos importantes são inerentes a este fato. São estes:

- remoção do todo implica na remoção das partes;
- existe apenas um todo, isto é, as partes não são compartilhadas com outros todos;
- as partes não podem ser acessadas “fora” do todo, ou seja, elas são particulares para o todo;
- uma mensagem destinada a uma parte deve ser enviada para o todo e retransmitida por ele à parte.

Isto significa que a agregação composta deve ser utilizada somente quando um objeto é considerado como uma parte de outro objeto e não apenas uma associação ocasional com existência e visibilidade independentes.

A fim de representar este tipo de relacionamento, utiliza-se uma linha que termina com um símbolo de diamante preenchido, símbolo este colocado contra o todo. Além disso, para evitar qualquer confusão possível, ao todo é atribuída, explicitamente, a multiplicidade de 1 (um), mesmo porque apenas um todo é possível.

A partir do diagrama UML e do que expomos, conseguimos classificar a classe Pedido como sendo o todo e as classes ItemPedido e InfoCliente como as partes do relacionamento. Assim, podemos eliminar as alternativas A e B, pois as partes não podem participar de mais de um relacionamento na composição agregada. Podemos eliminar, também, a alternativa C uma vez que uma parte não pode existir sem o todo. Lembre-se que seria necessário que as 3 (três) entidades estivessem associadas simultaneamente para que tivéssemos um relacionamento ternário, logo, a alternativa D também está errada. Portanto, a alternativa **E** é a correta, pois a multiplicidade 1 (um) atribuída ao todo é utilizada apenas para evitar qualquer confusão possível.

3. **Assuntos relacionados:** *Desenvolvimento de Sistemas, Modelagem Funcional, Diagramas UML,*

Banca: *Cesgranrio*

Instituição: *BNDES*

Cargo: *Analista de Sistemas - Desenvolvimento*

Ano: *2008*

Questão: *36*

O diagrama UML mais indicado para representar o passo a passo do fluxo de eventos principal de um caso de uso de um software orientado a objetos é o diagrama de

- (a). casos de uso.
- (b). atividades.
- (c). eventos e transições.
- (d). classes.
- (e). componentes.

Solução:

A UML (Unified Modeling Language) é uma linguagem que permite modelagem de sistemas por meio de diagramas padronizados. Ela especifica tanto o significado gráfico quanto semântico de cada diagrama. A versão 2.0 da UML contempla os seguintes diagramas:

- Diagramas Estruturais
 - Diagrama de Classes
 - Diagrama de Objetos
 - Diagrama de Componentes
 - Diagrama de Instalação
 - Diagrama de Pacotes
 - Diagrama de Estrutura
- Diagramas Comportamentais
 - Diagrama de Caso de Uso
 - Diagrama de Transição de Estados
 - Diagrama de Atividade
- Diagramas de Interação
 - Diagrama de Sequência
 - Diagrama de Interatividade
 - Diagrama de Colaboração ou Comunicação
 - Diagrama de Tempo

Para determinar a alternativa correta da questão, é importante conhecer cada um dos diagramas apresentados como alternativas.

(A) Diagrama de Caso de Uso

Segundo Ivan Jacobson, podemos dizer que um caso de uso é um documento narrativo que descreve a sequência de eventos de um ator que usa um sistema para completar um processo em um sistema. Em outras palavras, o caso de uso é uma técnica de modelagem

usada para descrever como que o sistema deve se comportar mediante a interação com seus diversos usuários que, nesse contexto, também são conhecidos como atores. Os diagramas de caso de uso, por sua vez, são representações gráficas que ilustram o relacionamento entre os três integrantes básicos de um caso de uso, que são os atores, os casos de uso e o sistema.

(B) Diagrama de Atividades

Os diagramas de atividade podem ser vistos como variações dos diagramas de estado, e têm objetivo capturar ações – **passos** que serão executados – e seus resultados em termos das mudanças de estados dos objetos. Em outras palavras, os diagramas de atividades mostram o **fluxo** sequencial das atividades executadas por uma operação específica do sistema. Os principais elementos de um diagrama de atividades são as atividades em si, os pontos de entrada e saída, as decisões, as subdivisões de atividades paralelas (fork), e as junções de atividades (join).

(C) Diagrama de Eventos e Transições

Os diagramas estados (também conhecidos como diagramas de eventos e transições) mostram os diferentes estados de um objeto durante sua vida, bem como os estímulos que acionam as mudanças de estado. Ou seja, os diagramas de estado vêem o ciclo de vida dos objetos como máquinas de estados (autômatos) finitos. O termo finito significa que existe um número finito de estados que o objeto pode assumir, bem como um é finito o número de estímulos que acionam as mudanças de estado.

(D) Diagrama de Classes

O objetivo dos diagramas de classe é descrever os vários tipos de objetos no sistema, bem como o relacionamento entre eles. Sobre os diagramas de classes, é importante ressaltar que eles podem oferecer três perspectivas distintas que são: (i) Conceitual: destinada aos clientes, representa os conceitos do domínio em estudo; (ii) Especificação: destinada às pessoas que não precisam saber detalhes de desenvolvimento, foca as principais interfaces da arquitetura e métodos, porém não como eles serão implementados; (iii) Implementação: destinada ao time de desenvolvimento, aborda vários detalhes de implementação.

(E) Diagrama de Componentes

O diagrama de componente mostram os componentes do software (por exemplo, componentes CORBA, java beans ou seções do sistema que são claramente distintas) e os artefatos de que eles são feitos, como arquivos de código fonte, bibliotecas de programação ou tabelas de bancos de dados relacionais.

Feitas as devidas apresentações dos diagramas apresentados como alternativas, podemos concluir que a resposta correta é a letra B. As palavras passos e fluxos, em destaque na apresentação do diagrama de atividades, foram a chave para a resolução dessa capciosa questão.

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

Índice Remissivo

Árvores de Decisão, 10

Agregação entre Classes, 91

Alocação de Memória, 30, 66

Análise de Pontos de Função, 55

Análise de Riscos, 101

Aritmética Computacional, 65

Arquitetura de Computadores, 63, 68

Arquitetura Empresarial, 44

Associação entre Classes, 91

Banco de Dados, 14, 18, 21, 24, 26, 100

Caminho Crítico, 49

Casos de Testes, 109

Ciclo de Vida de Projeto, 12

Clean Read, 100

COBIT, 46, 51, 61

Commit Alcançado, 100

Complexidade Ciclomática, 89

Composição Agregada, 6

Consulta SQL, 18

Content Management System (CMS), 95

Controle de Concorrência, 14

Conversão Numérica, 65

Cronograma, 49

Data Warehouse, 97

Deadlock Ausente, 100

DER, 24

Desenvolvimento de Sistemas, 4, 8, 10

Diagrama de Classes, 6

Diagramas UML, 8

Dirty Read, 100

DMA, 70

Domínios de Governança, 51

DTD, 84

Durabilidade, 100

Engenharia de Software, 55, 109, 110

ERP, 93

Escalonamento de Banco de Dados, 100

Escritório de Projetos, 71

Estimativa de Custo, 12

Estrutura Organizacional, 77

Event-Driven Process Chain (EPC), 87

Extreme Programming (XP), 4, 106

Framework de Zachman, 44

Funções Transacionais, 55

Gerência de Memória, 30

Gerência de Projeto, 12, 49, 53, 71, 74

Gerência de Transações, 21

Gerenciamento de Incidentes, 59

Gerenciamento de Memória, 66

Gerenciamento de Portifólios, 101

Gerenciamento de Projetos, 10, 77

Gerenciamento de Riscos, 10

Gerenciamento de Serviços de Terceiros, 61

Gerenciamento Integrado de Projetos, 101

Governança de TI, 44, 46, 51, 58, 59, 61

Grafo de Precedências, 14

HAVING, 26

Herança Múltipla, 91

Herança Simples, 91

HTTP, 38

HTTPS, 41

ICMP, 36

Independência de Fragmentação, 113

Independência de Localização, 113

Independência do Sistema Operacional, 113

ITIL, 58, 59

Java, 81

JOIN, 26

Lógica, 34

Leitura Limpa, 100

Leitura Suja, 100

Metodologia de Desenvolvimento de Software, 4

Missing Deadlock, 100

Modelagem As Is, 115

Modelagem de Negócio, 115

Modelagem de Processos, 87

Modelagem Funcional, 8

Modos de Endereçamento de Memória, 63

On-line Analytical Processing, 97

Operações de Composição de Relações, 18

Operações de Entrada e Saída, 70

Orientação a Objeto, 91

Partições de Equivalência, 109

PERT, 101

Planejamento em Ondas Sucessivas, 53

PMBOK, 49, 53, 71, 74

PMO, 71

Processo Unificado, 106
Projeto de Interface com Usuário, 110
Protocolos de Rede, 38

Qualidade Total, 101

Raciocínio Lógico, 60
Reached Commit, 100
Rede de Petri, 104
Redes de Computadores, 36, 38, 41
Refatoração, 12

Scrum, 106
Segurança da Informação, 41
Serialização, 14
Serializabilidade, 100
Single Point of Contact, 58
Sistemas Distribuídos, 113
Sistemas Operacionais, 30, 66, 68, 70
SOA, 80
Sobrecarga de Método, 81
Sobrescrita de Método, 81
SQL, 21, 26
SSL, 41

Testes de Software, 89
Transações de Banco de Dados, 14

UML, 6

Visões de Banco de Dados, 21

WBS, 53
Websites, 95

XML, 84