ESTUDOS DISCIPLINARES

Formação Específica

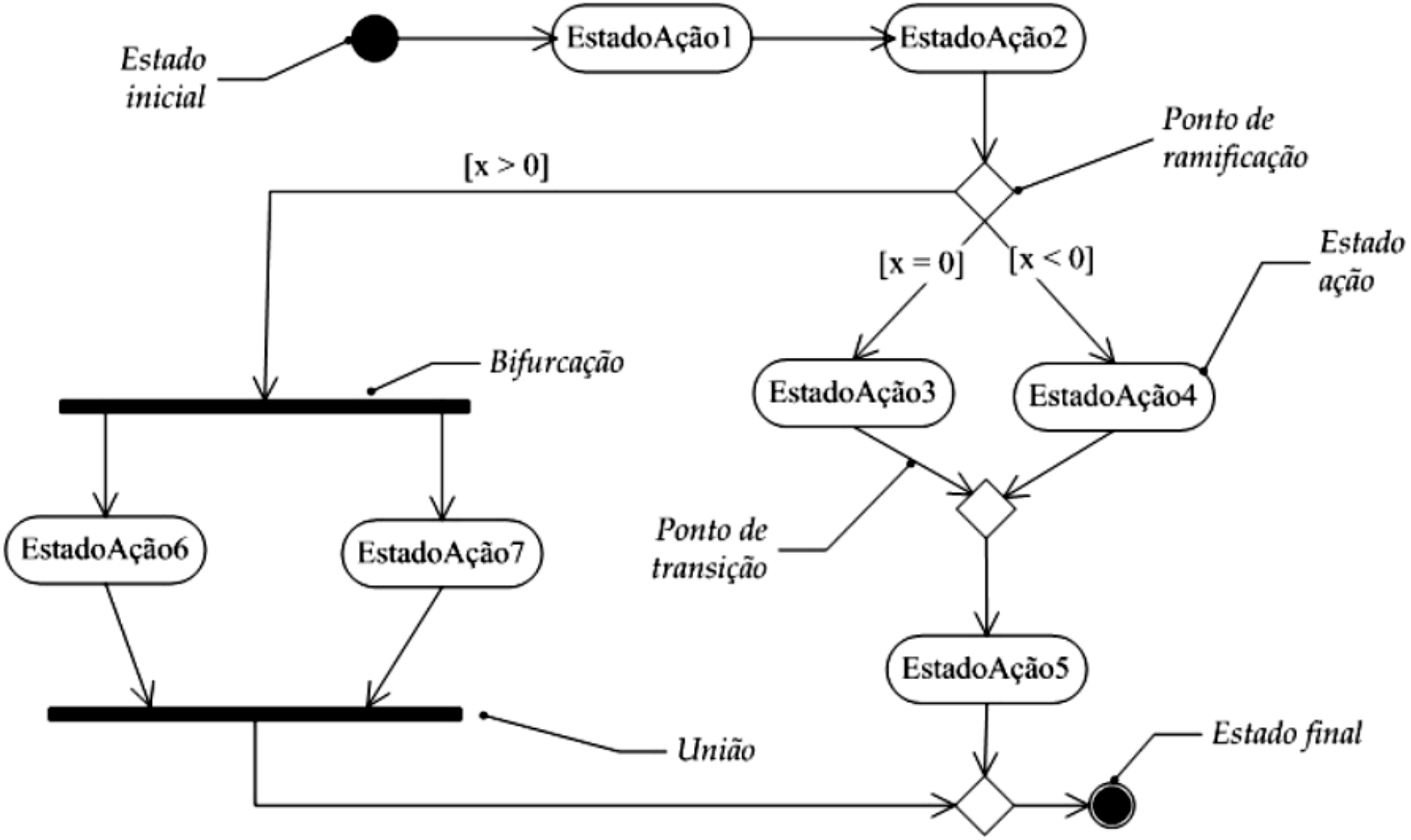
Prof. André Luiz

Diagrama de Atividades

- A partir da versão 2.0 da Unified Modeling Language (UML), o diagrama de atividades passou a ser visto como um diagrama independente. Ele é utilizado para modelar atividades.

- Uma atividade é composta por uma coleção de ações, ou seja, pelos passos necessários para que a atividade seja realizada, conforme mostra a figura a seguir.

Diagrama de Atividades



Fonte: Elementos de um diagrama de atividades. Fonte. BEZERRA; EDUARDO (2007).

Diagrama de Atividades

As coleções de ações podem ser divididas conforme a seguir:

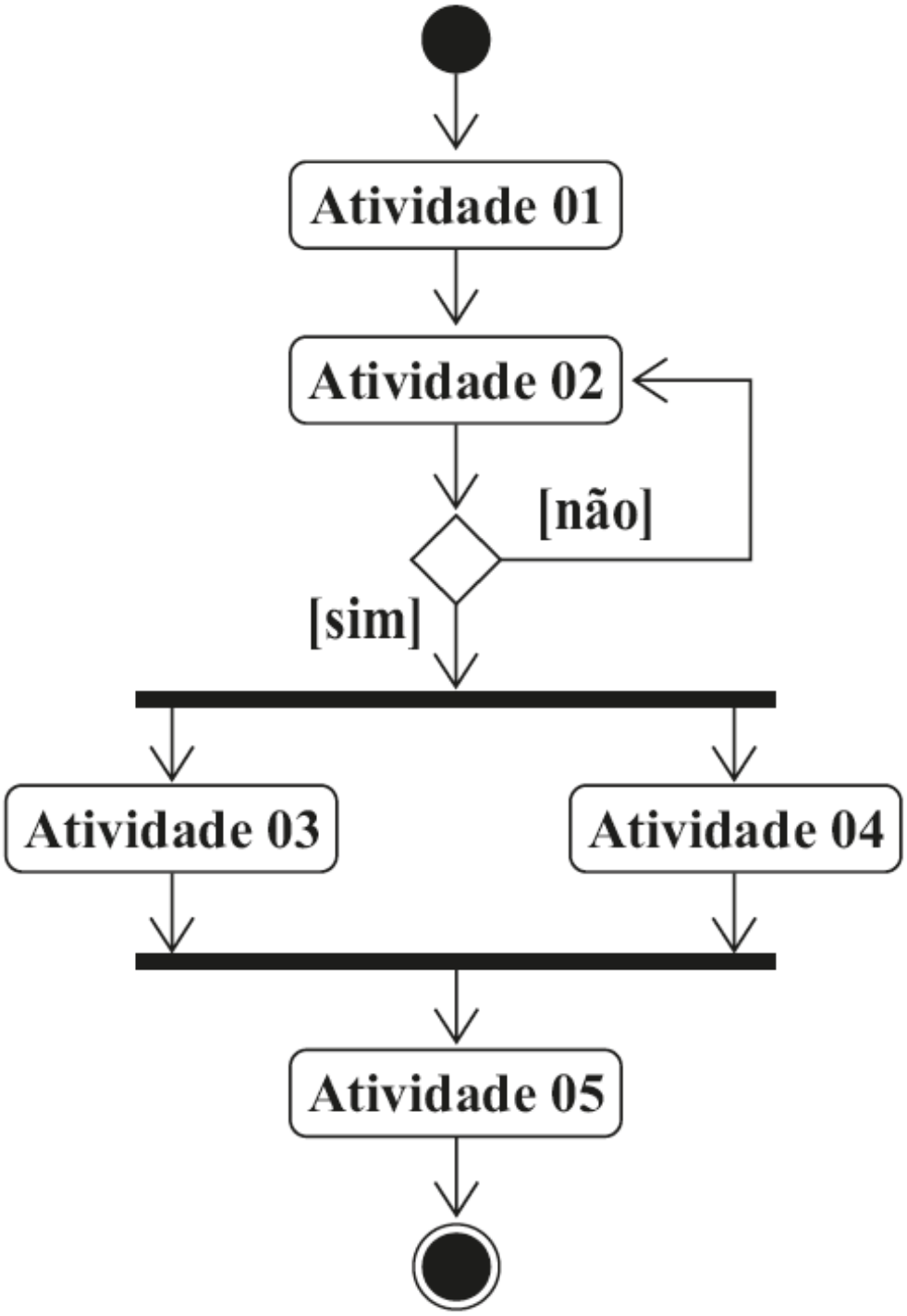
- Estado inicial: é representado por um círculo preenchido. Todo diagrama de atividades deve ter um estado inicial.
- Estado de ação: é realizado instantaneamente e indica que o sistema está realizando algo.
- Estado de atividade: necessita de um tempo para ser executado e indica que o sistema está realizando algo.
- Ponto de ramificação: é representado por um losango e apresenta uma transição de entrada e várias transições de saída.
- Ponto de união: é representado por um losango e reúne várias transições de entrada e uma transição de saída.

Diagrama de Atividades

- **Ponto de transição: ocorre no término de um estado de ação e, consequentemente, no início de outro.**
- **Bifurcação: pode ser representada por uma barra horizontal. Recebe uma transição de entrada e cria dois ou mais fluxos de controle que serão executados independentemente e em paralelo.**
- **União: é representada por uma barra horizontal. Recebe duas ou mais transições de entrada e une os fluxos em um único fluxo de controle.**
- **Estado final: um diagrama de atividade pode não apresentar estado final (quando o processo ou procedimento que está sendo modelado é cíclico) e também pode ter um ou vários estados finais.**

Questão 1 (ENADE, 2011)

- O diagrama de atividades é um dos diagramas disponíveis na UML (Linguagem de Modelagem Unificada) para a modelagem de aspectos dinâmicos de sistemas.



Questão 1 – Resposta

Com relação ao diagrama de atividades apresentado, avalie as afirmações a seguir.

I. A atividade 05 será executada se a atividade 03 ou a atividade 04 for concluída.

II. A ramificação sequencial existente após a atividade 02 significa que, caso o fluxo seja [não], é necessário que sejam executadas novamente as atividades 01 e 02.

III. As atividades 03 e 04 vão ter início ao mesmo tempo, entretanto, não significa que terminarão ao mesmo tempo.

IV. Caso o fluxo da ramificação sequencial existente após a atividade 02 tenha o fluxo [sim], a atividade 02 não será mais executada.

É correto apenas o que afirma em:

A. I e II **B. II e III** **C. III e IV** **D. I, II e IV** **E. I, III e IV**

Resposta

a) I e II

b) II e III

c) III e IV

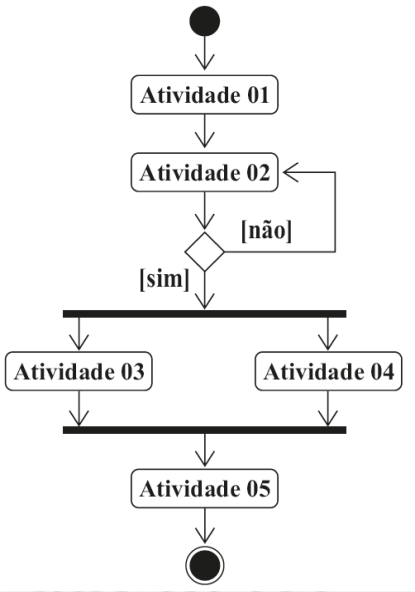
d) I, II e IV

e) I, III e IV

Comentários – Questão 1

I. Afirmativa incorreta

Justificativa. Para a atividade 05 ser executada, tanto a atividade 03 quanto a atividade 04 devem ser concluídas. A barra horizontal que representa a união das duas atividades conecta os fluxos em um único fluxo de controle.



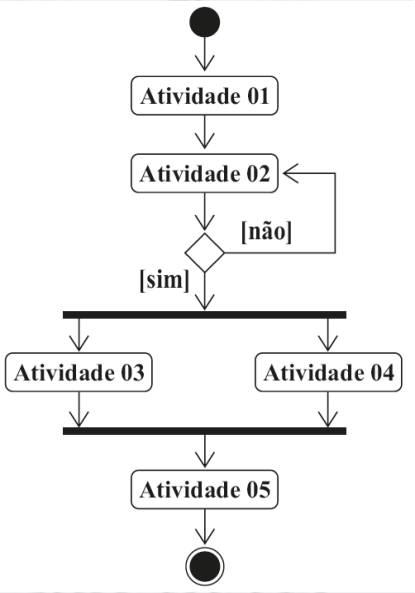
II. Afirmativa incorreta

Justificativa. Caso o fluxo seja [não], a análise do ponto de ramificação existente após a atividade 02 mostra a necessidade de que ela seja executada novamente.

Comentários – Questão 1

III. Afirmativa correta

Justificativa. As atividades 03 e 04 estão entre uma barra de bifurcação e uma barra de união. A barra de bifurcação cria dois fluxos de controle que serão executados independentemente e em paralelo, o que não significa que terminarão ao mesmo tempo. A barra de união recebe as duas entradas e une os fluxos em um único fluxo de controle.



IV. Afirmativa correta

Justificativa. Caso o fluxo da ramificação sequencial após a atividade 02 siga [sim], a atividade 02 não será mais executada, pois o fluxo segue para a barra de bifurcação.

Design Pattern e Abstract Factory

- **Os Design Patterns (Padrões de Projetos) são uma coleção de padrões de projetos de *software* que reúnem soluções para problemas recorrentes e conhecidos, descrevendo uma solução legítima para os problemas comuns.**

- **O padrão de criação Abstract Factory encapsula a criação de objetos. Esse padrão é muito útil para a construção de objetos individuais com propósito específico, sem a necessidade de serem conhecidas as classes específicas instanciadas.**

Design Pattern e Abstract Factory

- Um exemplo de utilização do Abstract Factory é o caso de um sistema que precisa de suporte para múltiplos tipos de interfaces gráficas (Windows, Mac etc.)
- Informa-se à *factory* que o sistema deve se parecer com o Windows.
- Após esse suporte, o sistema retorna à fábrica GUI (Graphical User Interface) dos objetos relativos ao Windows.
- Quando utilizamos um objeto (botão, *check-box* e janelas), a fábrica de GUI retorna às instâncias desses componentes para o Windows.

Questão 2 (ENADE, 2011)

Padrões de criação (creational patterns) abstraem a forma como objetos são criados, tornando o sistema independente de como os objetos são criados, compostos e representados. Um padrão de criação de classe usa a herança para variar a classe que é instanciada, enquanto que um padrão de criação de objeto delegará a instanciação para outro objeto. Há dois temas recorrentes nesses padrões. Primeiro, todos encapsulam conhecimento sobre quais classes concretas são usadas pelo sistema. Segundo, ocultam o modo como essas classes são criadas e montadas. Tudo que o sistema sabe no geral sobre os objetos é que suas classes são definidas por classes abstratas. Os padrões de criação são classificados em Abstract Factory, Builder, Factory Method, Prototype e Singleton.

Fonte: GAMMA, E. *et al*. Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley, 1994 (com adaptações).

Questão 2 (ENADE, 2011)

O padrão Abstract Factory é usado quando

a) o sistema deve ser independente da maneira como seus produtos são criados, relacionados e representados.

b) o algoritmo de criação de um objeto deve ser independente das suas partes e da maneira como ele é montado.

c) houver uma única instância de uma classe e esta for acessada a partir de um ponto de acesso conhecido.

d) classes delegam responsabilidade a alguma das subclasses e deseja-se localizar qual é a subclasse acessada.

e) as classes utilizadas para instanciação são especificadas em tempo de execução ou carregadas dinamicamente.

UNIP

Resposta

O padrão Abstract Factory é usado quando

a) **o sistema deve ser independente da maneira como seus produtos são criados, relacionados e representados.**

b) **o algoritmo de criação de um objeto deve ser independente das suas partes e da maneira como ele é montado.**

c) **houver uma única instância de uma classe e esta for acessada a partir de um ponto de acesso conhecido.**

d) **classes delegam responsabilidade a alguma das subclasses e deseja-se localizar qual é a subclasse acessada.**

e) **as classes utilizadas para instanciação são especificadas em tempo de execução ou carregadas dinamicamente.**

UNIP

Comentários – Questão 2

- **O padrão Abstract Factory é utilizado quando o sistema opera independentemente da maneira como seus produtos são criados, relacionados e representados.**

INTERVALO

Diagrama de Sequência

- O diagrama de Sequência da UML é um diagrama comportamental direcionado à ordem temporal em que as mensagens são trocadas entre os objetos.

- Geralmente é formado com insumos vindos dos Diagramas de Caso de Uso e de Classes.

- O Diagrama de Caso de Uso fornece os atores e o Diagrama de Classes fornece os objetos envolvidos. Desse modo, determina a ordem em que os eventos ocorrem e as mensagens que são enviadas (métodos chamados), estabelecendo a interação entre os objetos.

Diagrama de Sequência

- O Diagrama de Sequência origina-se das funcionalidades identificadas no Diagrama de Caso de Uso. Um Caso de Uso, por sua vez, refere-se a um comportamento do sistema e descreve a funcionalidade executada por um ator.
- Baseia-se, também, no Diagrama de Classes para retirar dele os objetos identificados até o momento. Mas, além disso, o Diagrama de Sequência serve para validar e refinar os dois diagramas, podendo levar a complementações ou modificações.
- A leitura do Diagrama de Sequência sempre é feita de cima para baixo e da esquerda para a direita. Adicionalmente, colocam-se números sequenciais para ordenar as mensagens.

UNIP

Diagrama de Sequência

- Os atores de um Diagrama de Sequência são os mesmos descritos nos Casos de Uso. São desenhados como bonecos magros. Os objetos são representados como quadrados ou estereótipos, mecanismos de extensão da UML, não usados no diagrama em questão.

- Cada ator ou objeto tem uma linha de vida, que é a linha tracejada. Esta se refere geralmente a uma instância e ao tempo de vida em que o objeto ou ator existem no processo.
  A linha de vida é interrompida com um X, que é destruído nesse momento.

Diagrama de Sequência

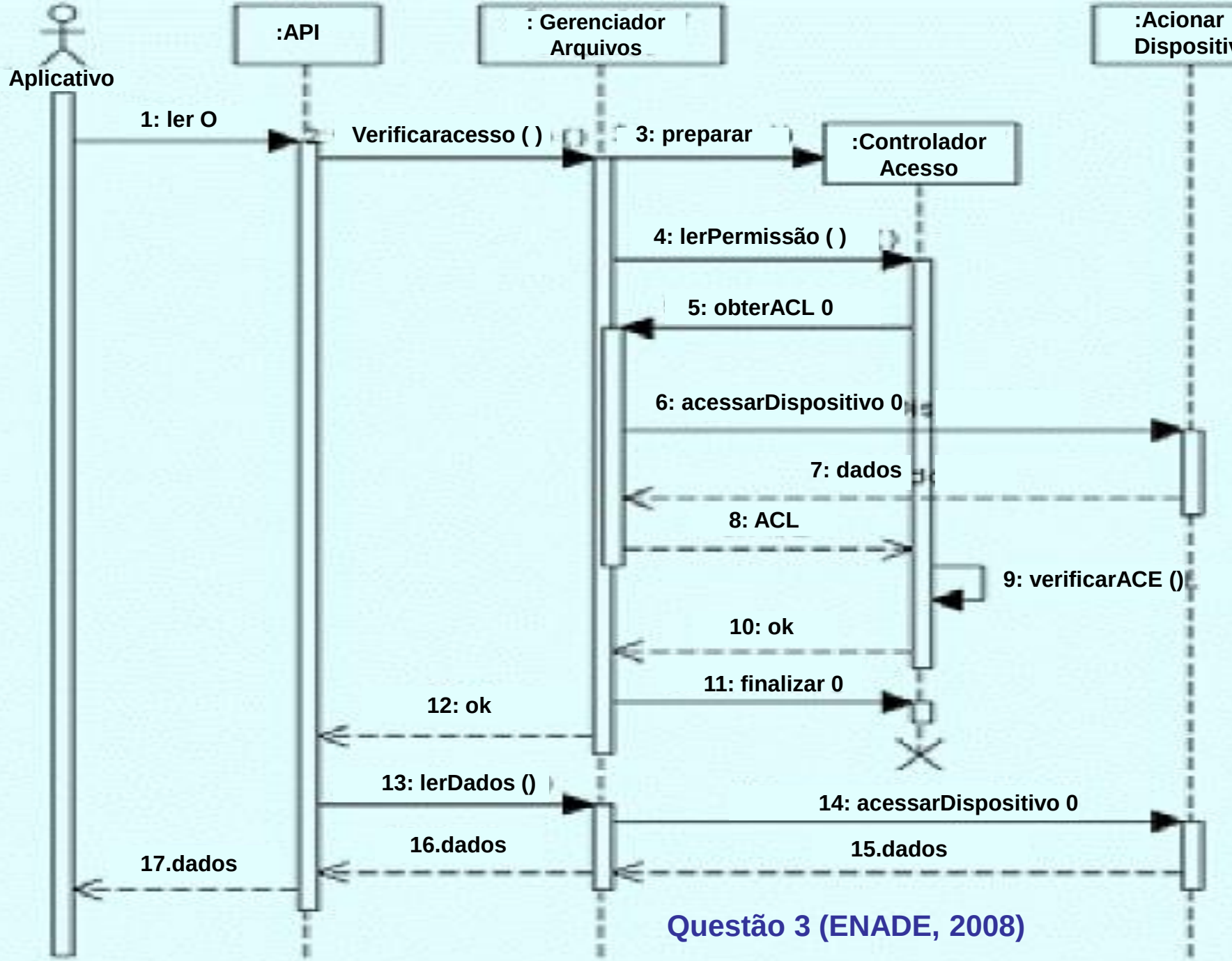
- Quando um objeto está ativamente participando do processo, ele tem o foco de controle. É a linha retangular fina que fica acima da linha tracejada.

- Os objetos que são representados no topo do diagrama existem desde o início do processo.

- Os objetos que estão abaixo no diagrama são criados e, muitas vezes, destruídos durante o processo.

- Um objeto pode enviar mensagem para si mesmo.
  É uma autochamada.

UNIP



Questão 3 (ENADE, 2008)

Questão 3 – Resposta

Com relação ao diagrama, assinale a opção correta.

a) Para economizar tempo e memória, as mensagens de retorno 7: dados e 15: dados poderiam ser mescladas em uma única mensagem.

b) O objeto Controlador Acesso utiliza uma estrutura de repetição para verificar os atributos de acesso a um arquivo.

c) A mensagem 5: obterACL( ) pode levar à repetição da chamada 4: lerPermissao( ).

d) Sempre que um Aplicativo fizer uma leitura, será construído e destruído um objeto ControladorAcesso.

e) A mensagem 3: preparar( ) ocorre simultaneamente (em paralelo) à mensagem 4: lerpermissao( ).

Resposta

Com relação ao diagrama, assinale a opção correta.

a) Para economizar tempo e memória, as mensagens de retorno 7: dados e 15: dados poderiam ser mescladas em uma única mensagem.

b) O objeto Controlador Acesso utiliza uma estrutura de repetição para verificar os atributos de acesso a um arquivo.

c) A mensagem 5: obterACL( ) pode levar à repetição da chamada 4: lerPermissao( ).

d) Sempre que um Aplicativo fizer uma leitura, será construído e destruído um objeto ControladorAcesso.

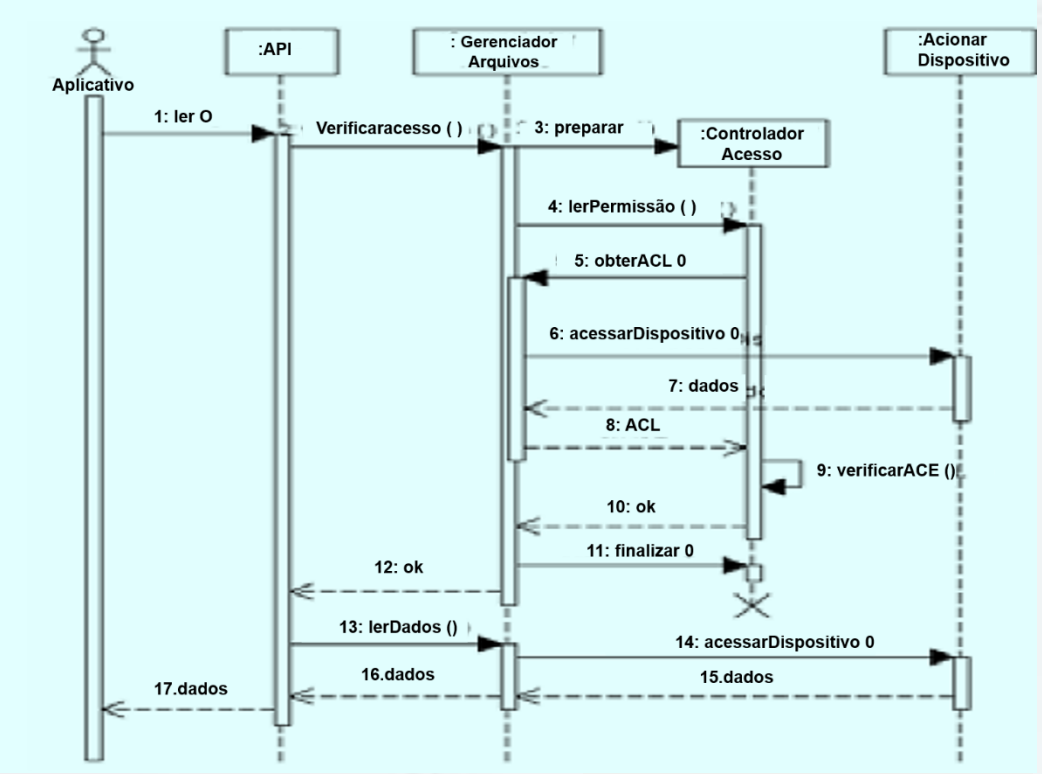
e) A mensagem 3: preparar( ) ocorre simultaneamente (em paralelo) à mensagem 4: lerpermissao( ).

UNIP

Comentários – Questão 3

A. Alternativa incorreta

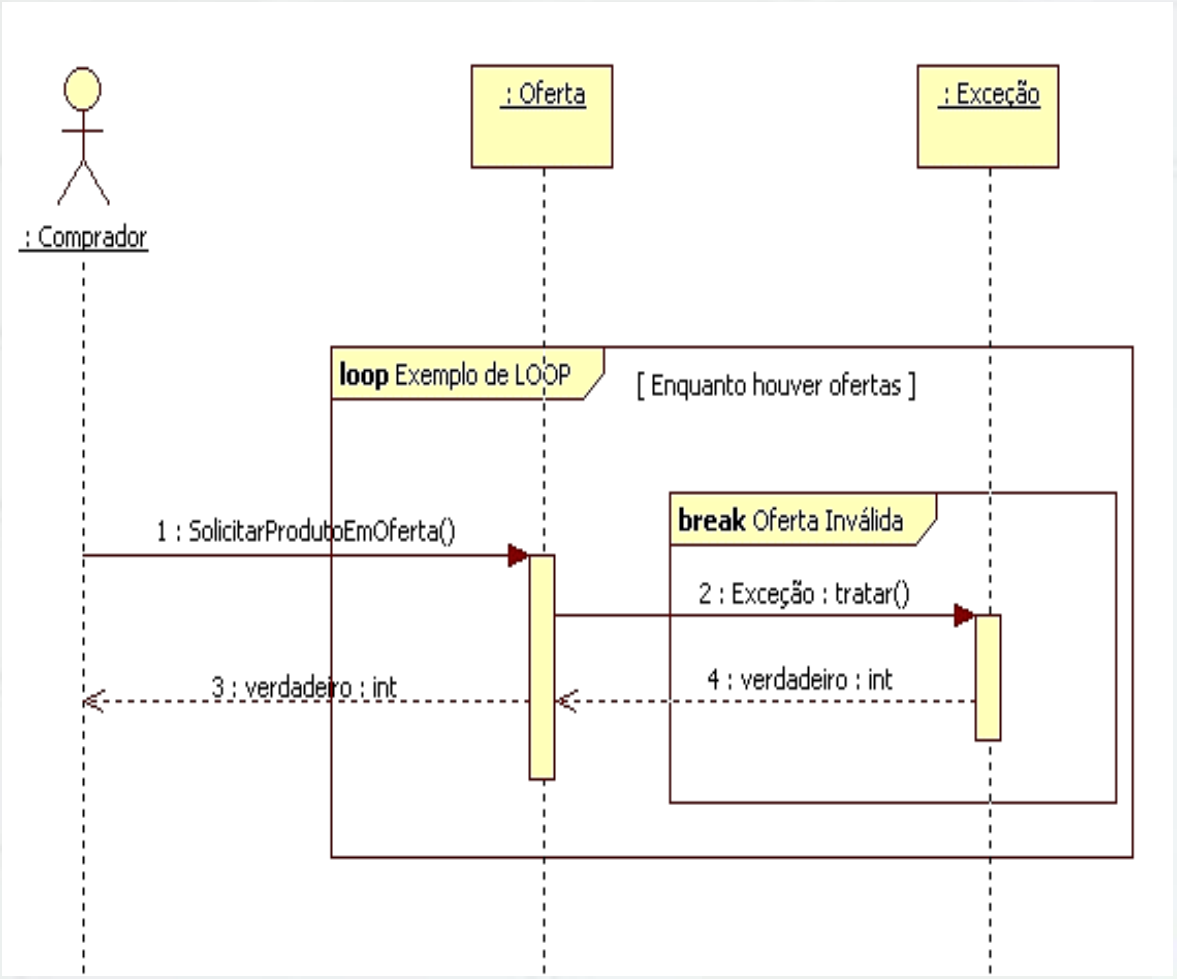
<u>Justificativa</u>. As mensagens 7 e 15 não são diferentes. São as mesmas mensagens disparadas em momentos distintos. É recomendado que mensagens diferentes tenham nomes diferentes para clareza do diagrama. Uma suposta economia de tempo e memória é irrelevante nesse caso.



Comentários – Questão 3

B. Alternativa incorreta

Justificativa. As estruturas de repetição no Diagrama de Sequência são obtidas por meio de fragmentos combinados: *LOOP* (para fazer o laço repetitivo) e *Break* (para interromper a execução). O objeto "Controlador Acesso" não utiliza estrutura de repetição, apenas uma autochamada na mensagem *9. verificarACL()*. Na figura 1, há um exemplo de *LOOP* com *BREAK*.

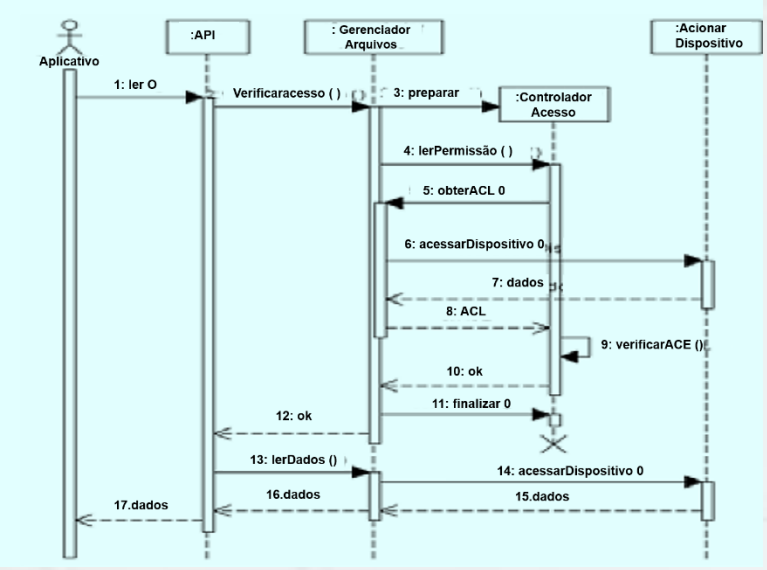


Fonte: Tomo

Comentários – Questão 3

C. Alternativa incorreta

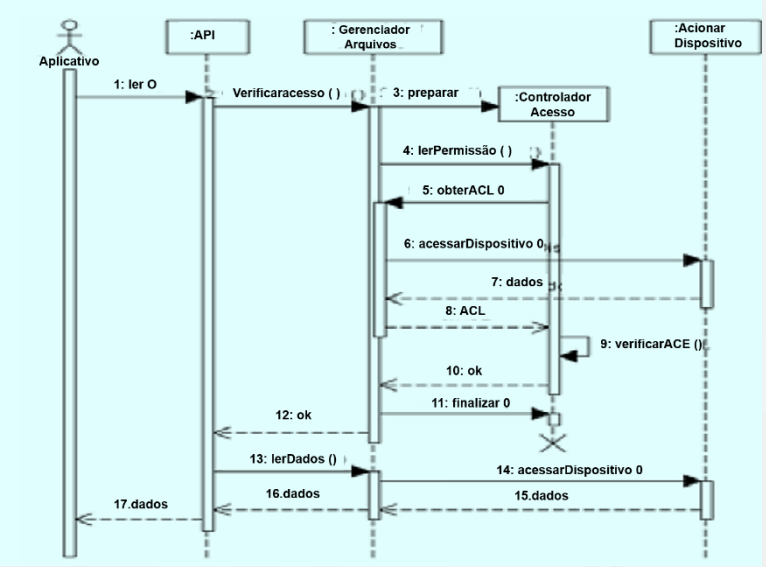
<u>Justificativa</u>. A sequência de leitura do Diagrama de Sequência é de cima para baixo e da esquerda para a direita. Portanto, após a mensagem *5. obterACL()*, o objeto envia a mensagem para *6. acessarDispositivo()*, jamais para *4. lerPermissao()*. Após a mensagem 6, o objeto "Acionador Dispositivo" envia a mensagem *7. dados*, e assim por diante até o final do diagrama.



Comentários – Questão 3

D. Alternativa correta

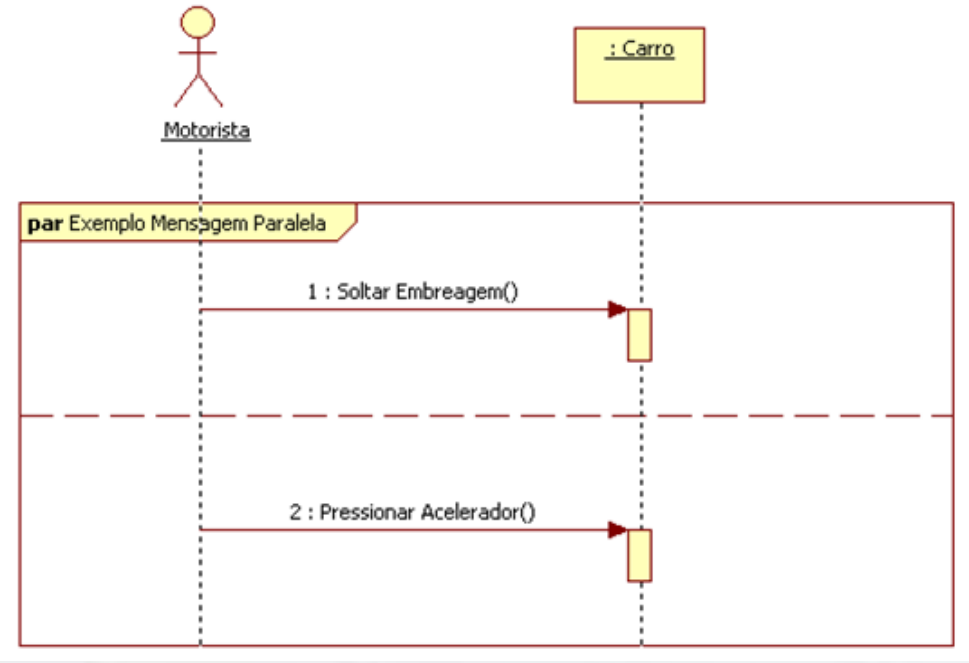
Justificativa. As mensagens *1. Ler()*, *2. verificarAcesso()* e *3.preparar()* são executadas sequencialmente, culminando na criação do objeto "Controlador Acesso". Isso acontece sempre e incondicionalmente até a destruição do objeto após a mensagem *11. Finalizar()*. Não existe nenhuma interrupção entre a mensagem *1. Ler()* e a mensagem *11. Finalizar()* que poderia impedir ou evitar a criação e destruição do objeto.



Comentários – Questão 3

E. Alternativa incorreta

Justificativa. As mensagens *3.preparar()* e *4. lerPermissao()* são sequenciais. Para que uma mensagem seja paralela à outra, no diagrama de sequência é preciso utilizar o fragmento combinado par, conforme ilustrado na figura.



Fonte: Tomo

Diagrama da UML

- Como a UML é uma linguagem desenvolvida para ser extensível e de fácil uso, seus elementos constituintes são partilhados entre diferentes diagramas. Desse modo, a adequada identificação de alguns diagramas só se torna possível pela interpretação da informação contida internamente neles.

- Analisemos, por exemplo, o conteúdo de um diagrama de máquina de estados e de atividades.

- Seus desenhos podem ser perfeitamente idênticos. O que os diferencia são os verbos utilizados para determinar suas atividades ou seus estados.

UNIP

Diagrama da UML

- Verbos no infinitivo (ou em outros tempos do indicativo) são aceitos como padrão para definir atividades.
- Verbos no gerúndio e particípio passado são usados para definir estados.
- Portanto, Registrar Voto designa uma atividade, enquanto que Registrando Voto ou Voto Registrado designam estados.
- Não podemos confundir diagramas com simbologias radicalmente distintas.
- O diagrama de comunicação contém elementos que tornam seu desenho completamente diferente. Nele, falamos da interação entre atores e objetos e não de uma visão de interação entre atividades e estados.

Diagrama da UML

- Embora um tempo verbal mal colocado possa dificultar identificar se um elemento é uma atividade ou estado, a semântica do diagrama de comunicação é completamente diferente. E, visualmente, são diagramas muito diversos, não se admitindo que sejam confundidos.

- As mesmas considerações são feitas em relação ao diagrama de casos de uso, que mostra as interações entre atores e funcionalidades do sistema. Tampouco se admite serem confundidos com os diagramas de atividades ou estados. Muito menos com o de comunicação. Portanto, os quatro diagramas considerados foram os indicados a seguir.

UNIP

Diagrama da UML

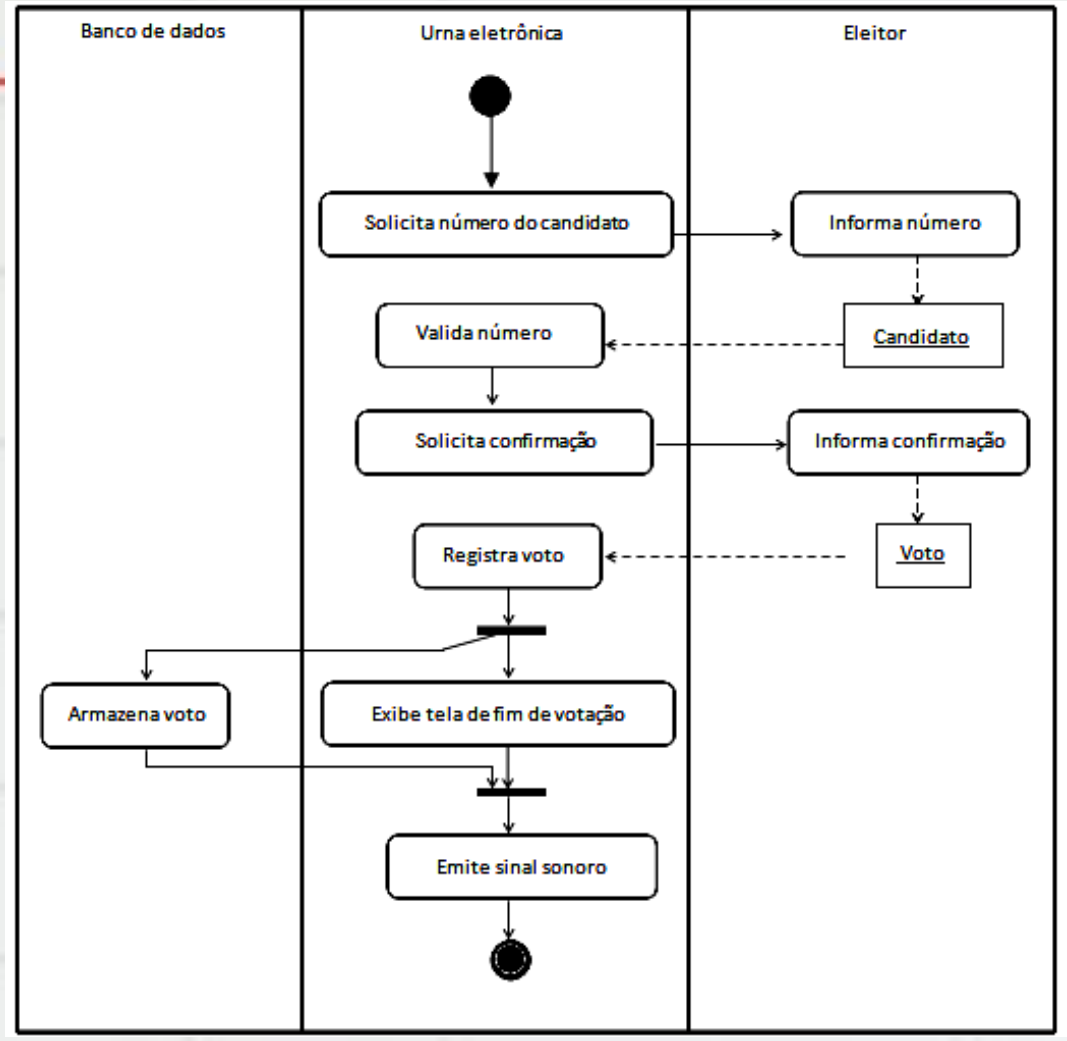
- **Atividades e Estados. Diagramas que podem ser confundidos um com o outro, se não for seguido um padrão rigoroso de nomenclatura dos elementos constituintes.**

- **Comunicação. Mostra a interação entre atores e objetos. Visualmente tem características particulares e inconfundíveis.**

- **Caso de Uso. Mostra a interação entre atores e funcionalidades, igualmente com características visuais inconfundíveis.**

UNIP

Questão 4 (ENADE, 2008)



A figura acima mostra um diagrama de

Questão 4 (ENADE, 2008)

a) comunicação, pois modela o comportamento do sistema e ilustra as interações entre atores e objetos.

b) estados, pois apresenta os possíveis estados do objeto Urna Eletrônica, além dos eventos que dão início à transição de um estado para outro.

c) estados, pois, a partir de um estado inicial, descreve a sequência de possíveis estados que todos os objetos podem assumir.

d) atividades, pois as ações ilustram a forma como o ator Eleitor interage com os objetos em um caso de uso.

e) atividades, pois modela o fluxo de controle de um processo composto por ações sequenciais e paralelas partindo de um estado inicial.

Resposta

a) comunicação, pois modela o comportamento do sistema e ilustra as interações entre atores e objetos.

b) estados, pois apresenta os possíveis estados do objeto Urna Eletrônica, além dos eventos que dão início à transição de um estado para outro.

c) estados, pois, a partir de um estado inicial, descreve a sequência de possíveis estados que todos os objetos podem assumir.

d) atividades, pois as ações ilustram a forma como o ator Eleitor interage com os objetos em um caso de uso.

e) atividades, pois modela o fluxo de controle de um processo composto por ações sequenciais e paralelas partindo de um estado inicial.

UNIP

Comentários – Questão 4

A. Alternativa incorreta

Justificativa. O diagrama não é de comunicação porque está modelando atividades (processos) e não a interação entre atores e objetos. No diagrama, embora sejam apresentados alguns atores, por exemplo, o Eleitor, não apresenta a comunicação entre objetos.

B. Alternativa incorreta

Justificativa. Não é um diagrama de estados porque está mostrando o fluxo do processo executado pelo objeto *Urna Eletrônica*, representado em forma de atividades (e não estados). Os estados para esse objeto seriam, por exemplo, Número do Candidato Informado, Validando Candidato, Solicitando Confirmação e Registrando Voto.

Comentários – Questão 4

C. Alternativa incorreta

Justificativa. É uma variação do erro da resposta anterior ampliada para os demais objetos. O diagrama não mostra os estados da Urna Eletrônica nem de qualquer outro objeto.

D. Alternativa incorreta

Justificativa. É um diagrama de atividades, mas as interações do ator não são com casos de uso. O diagrama de atividades é procedural e o de caso de uso, funcional. São visões distintas, representadas por diagramas diferentes e separados.

Comentários – Questão 4

E. Alternativa correta

Justificativa. O diagrama de atividades modela processos e seu fluxo de controle a partir de um estado inicial. As ações podem ser sequenciais e paralelas. O diagrama de estados modela os estados que também podem ser sequenciais e paralelos. Portanto, é necessário muito cuidado para não confundir estes diagramas.

INTERVALO

PMBOK

- O PMBOK é o Project Management Body of Knowledge, isto é, Corpo de Conhecimento em Gerência de Projetos do PMI (Project Management Institute, Instituto de Gerência de Projetos).

Os processos requeridos para entregar o projeto no prazo acordado estão na Gerência de Tempo. Esses processos são:

- Definição das atividades.
- Sequenciamento das atividades.
- Estimativa de recursos das atividades.
- Estimativa de duração das atividades.
- Desenvolvimento do cronograma.

PMBOK

Os demais processos pertencem a outras áreas, como descrito a seguir.

- **Planejamento, garantia e controle da qualidade pertencem à Gerência de Qualidade.**
- **Planejamento, definição, verificação e controle do escopo pertencem à Gerência de Escopo.**
- **Estimativa de custos, realização do orçamento e controle de custos pertencem à Gerência de Custos.**
- **Planejamento de compra e contratações, seleção de fornecedores e encerramento do contrato pertencem à Gerência de Aquisições (*Procurement*).**

Questão 5 (ENADE, 2008)

- **Após atuar como programador em uma empresa de desenvolvimento de *software* por aproximadamente 10 anos, um funcionário que se destacou por nunca atrasar um cronograma foi nomeado gerente de projetos.**

- **Ao assumir o primeiro projeto, o funcionário foi informado que sua principal responsabilidade era a realização da entrega conforme o cronograma estabelecido no contrato. Para o gerenciamento de tempo, o gerente de projetos irá utilizar o PMBOK.**

Considerando essa situação, é correto afirmar que o gerente de projetos deverá coordenar processos de

UNIP

Questão 5 (ENADE, 2008)

a) planejamento, garantia e controle da qualidade.

b) definição e sequenciamento de atividades, estimativa de recursos e duração da atividade, desenvolvimento e controle do cronograma.

c) planejamento, definição, verificação e controle do escopo.

d) estimativa de custos, realização do orçamento e controle de custos.

e) planejamento de compra e contratações, seleção de fornecedores e encerramento do contrato.

Resposta – Questão 5

a) planejamento, garantia e controle da qualidade.

b) definição e sequenciamento de atividades, estimativa de recursos e duração da atividade, desenvolvimento e controle do cronograma.

c) planejamento, definição, verificação e controle do escopo.

d) estimativa de custos, realização do orçamento e controle de custos.

e) planejamento de compra e contratações, seleção de fornecedores e encerramento do contrato.

Comentários – Questão 5

A. Alternativa incorreta

Justificativa. Planejamento, garantia e controle da qualidade são processos da Gerência de Qualidade.

B. Alternativa correta

Justificativa. Definição e sequenciamento de atividades, estimativa de recursos e duração da atividade, desenvolvimento e controle do cronograma são processos da Gerência de Tempo.

C. Alternativa incorreta

Justificativa. Planejamento, definição, verificação e controle do escopo são processos da Gerência de Escopo.

Comentários – Questão 5

D. Alternativa incorreta

Justificativa. Estimativa de custos, realização do orçamento e controle de custos são processos da Gerência de Custos.

E. Alternativa incorreta

Justificativa. Planejamento de compra e contratações, seleção de fornecedores e encerramento do contrato são processos da Gerência de Aquisições *(Procurement).*

Estrutura Analítica de Projetos

- No início de um projeto de *software* ou de qualquer outra natureza, é importante visualizar o seu escopo, conhecer o que deve ser entregue para o cliente e saber o que deve ser feito para que o plano seja bem-sucedido.

- Uma das ferramentas utilizadas para gerenciar o escopo de um projeto é a Estrutura Analítica do Projeto (EAP), conhecida em inglês como Work Breakdown Structure (WBS).

- Normalmente, a EAP é apresentada como uma árvore hierárquica, sendo que cada nível dessa árvore apresenta os detalhes do projeto.

Estrutura Analítica de Projetos

- A EAP deve ser construída com o objetivo de detalhar as entregas do projeto e de delimitar o escopo dessas tarefas.

- Segundo o Project Management Institute (PMI), no livro Practice Standard for Work Breakdown Structures, é importante que a EAP siga a chamada “regra dos 100%”.

- Essa regra afirma que a EAP deve conter todo o trabalho definido pelo escopo de projeto, nem mais nem menos.

Questão 6 (ENADE, 2014)

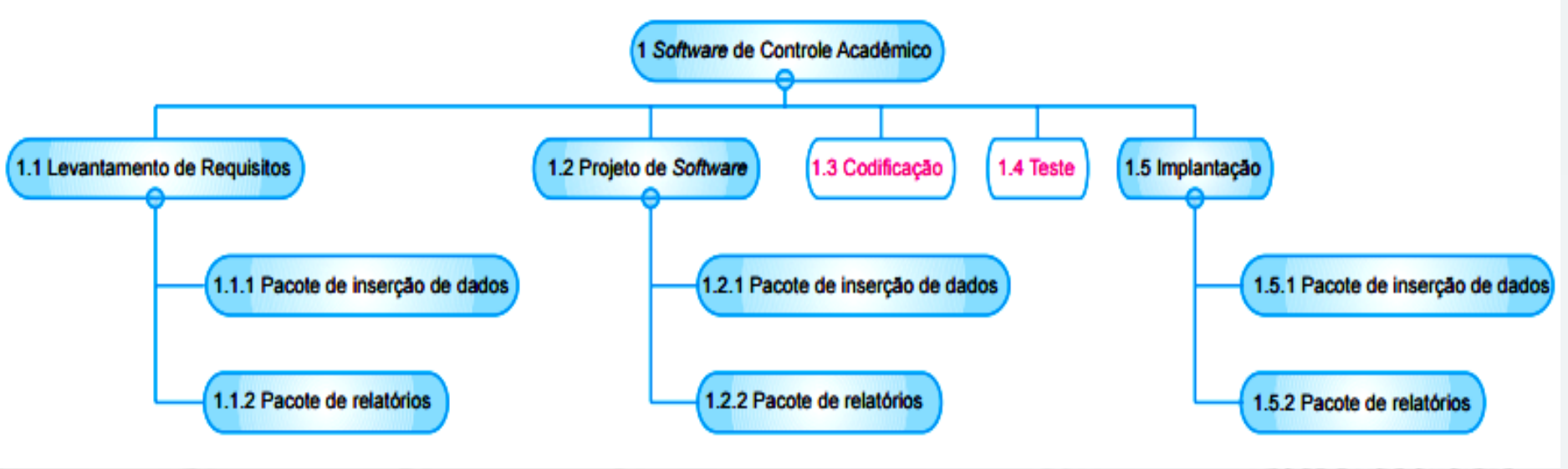
Conforme definido pelo Corpo de Conhecimento para Gerenciamento de Projeto de Software PMBOK, um dos artefatos de maior importância dentro do planejamento de um projeto de software é a Estrutura Analítica do Projeto (EAP).

Fonte: PMI. *Um guia do conhecimento em gerenciamento de projetos.* Guia PMBOK. 4. ed. EUA: Project Management Institute, 2008.

A EAP apresentada na figura a seguir mapeia o processo de produção de um *software* de uma empresa e os pacotes de trabalho que fazem parte desse projeto de *software*. Percebe-se que as atividades de "codificação" e "teste" não possuem pacotes de trabalho. A empresa Alfa optou por terceirizá-las para uma parceria em uma operação de *outsourcing*.

Questão 6 (ENADE, 2014)

Analisando-se o EAP da empresa, conclui-se que as atividades de “codificação” e “teste”



Questão 6 (ENADE, 2014)

a) devem ser inseridas no planejamento do projeto, uma vez que a empresa é a responsável pela implantação do *software* (a terceirização foi uma opção da própria empresa).

b) devem ser inseridas no planejamento do projeto porque é obrigatório que um processo de *software* tenha as atividades de "codificação" e "teste" dentro de seu conjunto de atividades.

c) não devem ser inseridas no planejamento do projeto porque, como elas foram terceirizadas, não é necessário realizar o controle de qualidade dessas atividades.

d) não devem ser inseridas no planejamento do projeto porque, como elas foram terceirizadas, não é necessário realizar o controle de produção – tempo, esforço e custo – dessas atividades.

e) não devem ser inseridas no planejamento do projeto, porque em um processo de *software* não é obrigatório que essas atividades estejam dentro de seu conjunto de atividades.

Resposta – Questão 6

a) devem ser inseridas no planejamento do projeto, uma vez que a empresa é a responsável pela implantação do *software* (a terceirização foi uma opção da própria empresa).

b) devem ser inseridas no planejamento do projeto porque é obrigatório que um processo de *software* tenha as atividades de "codificação" e "teste" dentro de seu conjunto de atividades.

c) não devem ser inseridas no planejamento do projeto porque, como elas foram terceirizadas, não é necessário realizar o controle de qualidade dessas atividades.

d) não devem ser inseridas no planejamento do projeto porque, como elas foram terceirizadas, não é necessário realizar o controle de produção – tempo, esforço e custo – dessas atividades.

e) não devem ser inseridas no planejamento do projeto, porque em um processo de *software* não é obrigatório que essas atividades estejam dentro de seu conjunto de atividades.

Comentários – Questão 6

A. Alternativa correta

Justificativa. A EAP deve definir claramente o escopo do projeto e as atividades de codificação e de teste fazem parte do projeto, independentemente do fato de terem sido terceirizadas. Logo, seguindo a chamada "regra dos 100%", concluímos que a EAP deve conter as atividades de codificação e de teste. Além disso, tais atividades devem ser incluídas na EAP, uma vez que elas devem ser planejadas, executadas, verificadas e entregues.

B. Alternativa incorreta

Justificativa. A inclusão das atividades de codificação e de teste é comum na maioria dos projetos de *software*. Essa situação ocorre em função de tais atividades fazerem parte do escopo e do planejamento do projeto.

Comentários – Questão 6

C e D. Alternativas incorretas

Justificativa. O fato de uma atividade ser terceirizada não significa que ela não deva ser controlada: pelo contrário, uma vez que a sua execução está nas mãos de terceiros, ela deve ser monitorada com mais cuidado.

E. Alternativa incorreta

Justificativa. Uma vez que essas atividades fazem parte do escopo do projeto, elas devem ser inseridas no planejamento.

INTERVALO

Banco de Dados Relacionais

- Frequentemente, problemas na área de computação lidam com grande quantidade de dados, gerados de forma artificial (por um programa), coletados do mundo externo (por exemplo, por meio de cadastros ou de instrumentos conectados ao computador) ou frutos de interação do usuário com o computador. Muitas vezes, queremos que esses dados sejam armazenados de forma permanente ou por um período de tempo potencialmente longo.
- Podemos armazenar dados de várias formas. Por exemplo, um simples arquivo de texto gravado em um pen drive é uma forma válida de armazenamento de dados. Porém, conforme a quantidade de dados armazenados aumenta, pode surgir uma série de problemas: encontrar dados específicos em meio a uma grande quantidade de dados disponíveis, relacionar os dados entre si e atualizar e remover dados.

UNIP

Banco de Dados Relacionais

- Com o aumento da capacidade de armazenamento dos computadores, bem como a sua popularização, várias técnicas de organização e de gerenciamento de dados foram desenvolvidas. Uma das formas que se tornou bastante importante no desenvolvimento de grandes sistemas é a dos bancos de dados relacionais.
- Seguindo a abordagem de Ritchie (2002), podemos dizer, de forma simplificada, que os bancos de dados relacionais são uma coleção de tabelas interligadas.
- As tabelas são formalmente chamadas de relações e cada uma de suas linhas (que são fisicamente registros da tabela) é chamada de tupla. Cada tabela tem várias colunas, chamadas de atributos. Como exemplo, considere a tabela a seguir.

Banco de Dados Relacionais

Compositor	
Nome	**Gênero**
Ludwig van Beethoven	Clássico
Johann Sebastian Bach	Barroco
John Coltrane	Jazz
Miles Davis	Jazz
Antônio Carlos Jobim	MPB

Fonte: Tomo

Tabela 1. Exemplo de uma tabela em um banco de dados.

Na tabela 1, “Compositor”, os atributos (colunas) são “Nome” e “Gênero”.

Uma tupla dessa tabela seria: “Miles Davis; Jazz”.

Banco de Dados Relacionais

- O diagrama de entidades e relacionamento apresenta uma notação especial para o projeto de banco de dados.
- Nesses diagramas existem três elementos básicos: tipos de entidade, relacionamentos e atributos.
- O primeiro elemento, o tipo de entidade, é representado por um quadrado na maioria das notações e é utilizado para representar objetos, coisas, pessoas e também eventos (MANNINO, 2008).
- Em particular, uma entidade é um membro ou instância de um tipo de entidade.
- Os atributos são as propriedades dos tipos de entidade, enquanto os relacionamentos são as associações entre os tipos de entidades (MANNINO, 2008).

Banco de Dados Relacionais

- Outro aspecto importante no modelo relacional é que cada linha de uma tabela deve ter uma combinação única de elementos, não podendo haver linhas repetidas (MANNINO, 2008).
- Contudo, isso não quer dizer que não exista a possibilidade de que alguns valores individuais não se repitam, mas sim de que a linha toda seja única.
- Por exemplo, em uma tabela contendo o nome, o CPF e o salário de cada um dos funcionários de uma empresa, pode haver mais de um funcionário com o mesmo nome e salário (mas nunca com o mesmo CPF, que é único para cada indivíduo). Logo, cada linha dessa tabela é única.

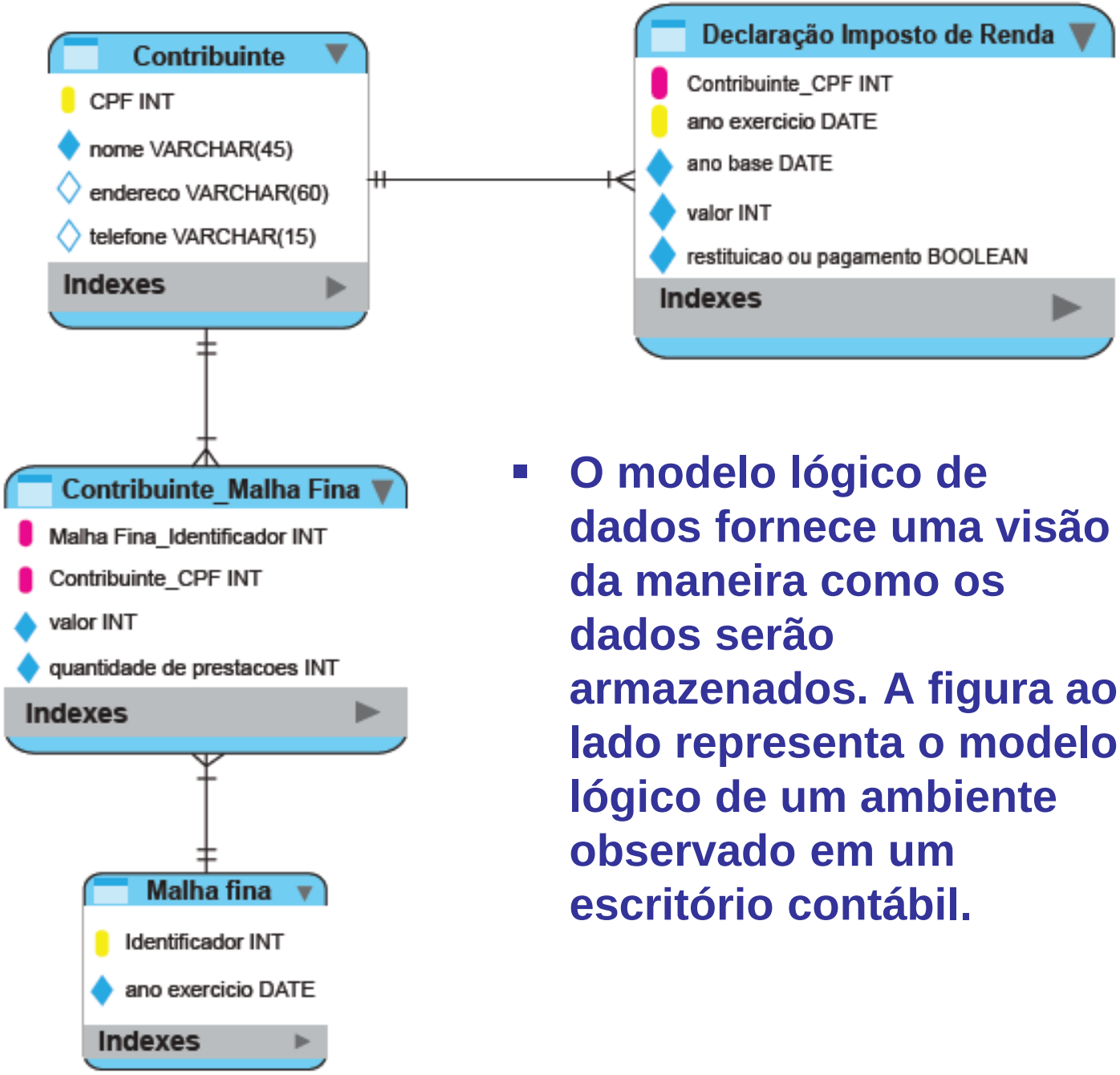
Banco de Dados Relacionais

- Um conjunto de colunas com uma combinação única é chamado de super-chave. Por isso, uma linha completa (com todas as colunas) é sempre uma super-chave.
- Contudo, se pudermos remover elementos da super-chave até o ponto em que temos uma combinação mínima de colunas (mas ainda única), dizemos que essa super-chave mínima é uma chave candidata. Quando uma chave candidata é especialmente designada para uma tabela, dizemos que ela é uma chave primária (MANNINO, 2008).
- Para ser possível referenciar tabelas diferentes, pode-se adicionar a chave primária de uma tabela em outra tabela, o que permite a seleção de um registro específico de uma tabela a partir de outra. A isso chama-se chave estrangeira (CORONEL e MORRIS, 2016; MANNINO, 2008).

UNIP

Banco de Dados Relacionais

- Em um diagrama de entidade-relacionamento, existe a possibilidade de que uma entidade dependa de outra. Essa entidade pode precisar que parte da sua chave primária venha de outros tipos de entidades (ou até mesmo toda a chave). Nesse caso, dizemos que ela é uma entidade fraca (MANNINO, 2008).
- Com o objetivo de garantirmos a integridade das entidades e dos dados armazenados em um banco de dados, introduzimos diversos tipos de restrições, especialmente ligados a cada tipo de integridade.
- Assim, se quisermos garantir a integridade referencial, entende-se que a chave estrangeira de uma tabela deve conter os valores vindos da chave primária da tabela pai (ou da chave candidata) ou o valor nulo (CORONEL e MORRIS, 2016).



Questão 7 (ENADE, 2014)

- O modelo lógico de dados fornece uma visão da maneira como os dados serão armazenados. A figura ao lado representa o modelo lógico de um ambiente observado em um escritório contábil.

Questão 7 (ENADE, 2014)

Em relação ao modelo, avalie as afirmativas a seguir.

I. A entidade Declaração Imposto de Renda é uma entidade fraca.

II. O relacionamento entre Contribuinte e Malha Fina é do tipo N:M (muitos para muitos).

III. O atributo CPF da entidade Contribuinte tem a função de chave estrangeira na entidade Declaração Imposto de Renda e no relacionamento Contribuinte_MalhaFina.

IV. A entidade Malha Fina não possui chave primária, somente chave estrangeira.

V. O relacionamento Contribuinte_MalhaFina é um relacionamento ternário.

Questão 7 (ENADE, 2014)

É correto apenas o que se afirma em:

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

Respostas – Questão 7

É correto apenas o que se afirma em:

a) I, II e III

b) I, II e IV

c) I, IV e V

d) II, III e V

e) III, IV e V

Comentários – Questão 7

I. Afirmativa correta

Justificativa. A entidade "Declaração de Imposto de Renda" depende da existência da entidade "Contribuinte" e tem como discriminador o atributo "ano exercício".

II. Afirmativa correta

Justificativa. A tabela "Contribuinte_Malha Fina" faz o papel de uma tabela de ligação entre as entidades "Contribuinte" e "Malha Fina", justamente porque o relacionamento é do tipo N:M (muitos para muitos).

Comentários – Questão 7

III. Afirmativa correta

Justificativa. Tanto na entidade "Contribuinte_Malha Fina" quanto na entidade "Declaração Imposto de Renda", é necessário o atributo CPF, que aponta para a entidade "Contribuinte" uma chave estrangeira.

IV. Afirmativa incorreta

Justificativa. O atributo "Identificador" é a chave primária da entidade "Malha Fina".

V. Afirmativa incorreta

Justificativa. O relacionamento "Contribuinte_Malha Fina" é um relacionamento entre duas entidades, as entidades "Contribuinte" e "Malha Fina" e, portanto, é um relacionamento binário.

Questão 8 (ENADE, 2014)

O modelo relacional representa o banco de dados como uma coleção de relações (tabelas). Na terminologia formal do modelo relacional, uma linha é chamada de "tupla", o título da coluna é denominado "atributo" e a tabela é chamada de "relação". O tipo de dado que descreve os tipos de valores que podem aparecer em cada coluna é denominado "domínio". Um banco de dados relacional pode impor vários tipos de restrições nos dados armazenados.

Fonte: ELMASRI, R. NAVATHE, S.B. Sistema de Banco de Dados: Fundamentos e Aplicações. Rio de Janeiro: LTC, 2002.

Restrições que permitem controlar situações como, por exemplo, "o salário de um empregado não deve exceder o salário do supervisor do empregado" e utilizam mecanismos chamados *triggers* (gatilhos) na sua implementação, são do tipo

Questão 8 (ENADE, 2014)

a) restrições de domínio.
b) restrições de unicidade.
c) restrições de integridade referencial.
d) restrições de integridade da entidade.
e) restrições de integridade semântica.

Resposta – Questão 8

a) restrições de domínio.

b) restrições de unicidade.

c) restrições de integridade referencial.

d) restrições de integridade da entidade.

e) restrições de integridade semântica.

Comentários – Questão 8

A. Alternativa incorreta

Justificativa. A restrição descrita no enunciado não limita o tipo de dados de um atributo e, portanto, não é uma restrição de domínio.

B. Alternativa incorreta

Justificativa. A restrição descrita no enunciado não garante unicidade.

C. Alternativa incorreta

Justificativa. A restrição descrita no enunciado não menciona tipo de relacionamento entre entidades e, portanto, não pode ser vista como uma restrição de integridade referencial.

Comentários – Questão 8

D. Alternativa incorreta

Justificativa. A restrição descrita no enunciado não garante que mais de um empregado tenha os valores iguais em uma tabela, apenas garante que o empregado ganhe menos do que o seu supervisor. Por exemplo, dois funcionários poderiam ter o mesmo salário, desde que esse salário fosse menor do que o salário do supervisor.

E. Alternativa correta

Justificativa. A restrição descrita no enunciado é uma regra de negócio. Logo, é uma restrição de integridade semântica que significa que cada atributo apresenta um valor consistente com o tipo de dado.

UNIP

ATÉ A PRÓXIMA!

ESTUDOS DISCIPLINARES

Formação Específica

Prof. André Luiz

Missão, visão e valores de uma empresa

- Segundo Silva (2017), o conjunto formado pela missão, pela visão e pelos valores representa a identidade organizacional.

- Todos devem saber claramente a respeito do propósito e da razão da existência da organização, além dos valores em que ela está calcada.

- Esse conjunto é o norteador de todas as ações da empresa e deve garantir a coerência entre o que a organização se propõe a fazer e o que ela realmente faz (MATOS, MATOS e ALMEIDA, 2007).

Plano de negócios

- O plano de negócio é um documento que descreve os objetivos de um negócio e traça os procedimentos que serão adotados para que esses objetivos sejam alcançados, a fim de que os riscos e as incertezas sejam diminuídos.

- Esse plano permite identificar possíveis erros de planejamento antes que eles sejam cometidos no mercado (SEBRAE, 2013).

- Vale enfatizar que o plano de negócios é um documento em que estão descritas a caracterização do negócio, sua operação, suas estratégias e suas projeções de receitas, despesas e resultados financeiros (SALIM, 2005).

Plano de gerenciamento de projetos

- **O plano de gerenciamento de projetos tem como objetivos propor a documentação relativa às premissas e às restrições de determinado projeto e delinear seu escopo, seus prazos, seus riscos, suas definições e suas responsabilidades (POSSI, 2006).**

Plano de gerenciamento de riscos

- **O plano de gerenciamento dos riscos descreve como os processos que envolvem riscos são estruturados e executados.**

- **Esse plano identifica os riscos, realiza análises qualitativas e quantitativas, constrói a proposta de respostas e determina como os riscos são controlados e monitorados (ESCRITÓRIO DE PROJETOS, 2017).**

Plano de contingência

- Para Romero (2016), contingência é uma eventualidade, um acaso, um acontecimento que tem como base a incerteza de ocorrência.
- Assim, qualquer que seja a atividade de uma empresa, é necessário que existam alternativas para mantê-la em perfeito funcionamento. Esse conjunto de alternativas é chamado de plano de contingência.
- Segundo Rezende (2005), para cobrir uma possível falha nos sistemas originais, a área de tecnologia da informação e comunicação (TIC) deve estabelecer um plano de contingência que apresente as alternativas para o processamento de dados e a execução dos sistemas organizacionais.

Questão 9 (ENADE, 2014)

- No cenário de uma economia globalizada, cresce o interesse pelo empreendedorismo e pela busca de meios que levem a uma maior produtividade, competitividade e inovação. Os avanços das tecnologias da informação e comunicação (TIC) auxiliam esse crescimento. Atualmente, uma parcela significativa dos negócios tem uma dependência forte das TIC.
- Desse modo, manter a disponibilidade da informação e a comunicação e manter os negócios operando, sem qualquer paralização, é indispensável. Porém, é preciso analisar o que pode ser afetado, qual o impacto financeiro e quais os impactos na imagem e na reputação da empresa, se cada um dos processos de negócio sofresse uma paralização por conta da TIC.
- A fim de mitigar possíveis riscos, é recomendável documentar um plano para eliminar ou reduzir a possibilidade de ocorrer cenários de indisponibilidade da TIC. Nessa situação, é preciso elaborar um:

Questão 9 (ENADE, 2014)

a) Plano de negócio.

b) Documento de visão.

c) Plano de contingência.

d) Plano de riscos.

e) Plano de gerenciamento de projetos.

Questão 9 – resposta

a) Plano de negócio.

b) Documento de visão.

c) Plano de contingência.

d) Plano de riscos.

e) Plano de gerenciamento de projetos.

Comentários – questão 9

A – Alternativa incorreta.

Justificativa. O plano de negócio estabelece os objetivos de um negócio e delineia os procedimentos adotados para que esses objetivos sejam alcançados, o que diminui os riscos e as incertezas.

B – Alternativa incorreta.

Justificativa. A missão, a visão e os valores de uma organização representam a identidade organizacional. Esse conjunto constitui os princípios sobre os quais a empresa se estabelece.

Comentários – questão 9

C – Alternativa correta.

Justificativa. O plano de contingência procura traçar as alternativas para evitar que, em função de uma falha ou de uma eventualidade, seja interrompida a prestação de determinado serviço ou a produção de um item.

D – Alternativa incorreta.

Justificativa. O plano de gerência de risco identifica os riscos, faz as análises qualitativas e quantitativas desses riscos, constrói o plano de respostas e determina como os riscos são controlados e monitorados.

Comentários – questão 9

E – Alternativa incorreta.

Justificativa. O plano de gerenciamento de projetos documenta as bases e as restrições existentes para o desenvolvimento de determinado projeto. Esse plano estabelece o escopo, os prazos, os riscos, as definições e as responsabilidades.

Questão 10 (FUNIVERSA, 2014)

- No que se refere aos conceitos e à necessidade da gerência de projetos nas empresas, assinale a alternativa correta.

a) Como regra geral, o gerenciamento de projetos não é aplicável em situações nas quais um empreendimento foge ao que é rotineiro dentro de uma empresa.

b) O gerenciamento do projeto só terá sucesso se os gestores o igualar às demais atividades desenvolvidas na empresa.

c) Empresas que necessitam de quantidade de dinheiro superior ao que usualmente é empregado são beneficiadas com o gerenciamento de projetos.

Questão 10 (FUNIVERSA, 2014)

- **No que se refere aos conceitos e à necessidade da gerência de projetos nas empresas, assinale a alternativa correta.**

d) O não cumprimento constante de prazos de contratos, por exemplo, que prejudique a imagem da empresa não caracteriza motivo aceitável para a implantação do gerenciamento de projetos.

e) O gerenciamento de projetos só não é aplicável em empresas que atuam em mercados agitados, uma vez que ele pode prejudicar a criatividade organizacional.

Questão 10 – resposta

a) Como regra geral, o gerenciamento de projetos não é aplicável em situações nas quais um empreendimento foge ao que é rotineiro dentro de uma empresa.

b) O gerenciamento do projeto só terá sucesso se os gestores o igualar às demais atividades desenvolvidas na empresa.

c) Empresas que necessitam de quantidade de dinheiro superior ao que usualmente é empregado são beneficiadas com o gerenciamento de projetos.

Questão 10 (FUNIVERSA, 2014)

d) O não cumprimento constante de prazos de contratos, por exemplo, que prejudique a imagem da empresa não caracteriza motivo aceitável para a implantação do gerenciamento de projetos.

e) O gerenciamento de projetos só não é aplicável em empresas que atuam em mercados agitados, uma vez que ele pode prejudicar a criatividade organizacional.

INTERVALO

PMBOK

- **É uma norma reconhecida para a profissão de gerenciamento de projetos.**
- **Padrão é um documento formal que descreve normas, métodos, processos e práticas estabelecidas.**
- **Evoluiu a partir das boas práticas reconhecidas de profissionais de gerenciamento de projetos que contribuíram para o seu desenvolvimento.**
- **Fornece diretrizes para o gerenciamento de projetos individuais.**

PMBOK – organização

Iniciação

- Obtém o comprometimento para o início do projeto ou de uma fase.

Planejamento

- Detalhamento de um projeto ou fase.

Execução

- Implementação das atividades do projeto ou fase.

Controle

- Monitora os resultados do projeto ou fase.

Encerramento

- Processo de conclusão das atividades de um projeto ou fase.

PMBOK – processos

- Está dividido em 10 áreas de processo com um total de 42 processos.

As áreas de processo são:

- Escopo, prazo, custo, qualidade, riscos, recursos humanos, comunicação, aquisição, integração e envolvidos.

Questão 11 (FGV, 2013)

- **Sobre Gerenciamento de Projetos, de acordo com o PMBOK, analise as afirmativas.**

I. **Gerenciamento de projetos é a aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto a fim de atender a seus requisitos.**

II. **O gerenciamento é realizado pela aplicação e pela integração apropriadas de 42 processos.**

III. **Os processos são agrupados em quatro grupos: iniciação, planejamento, execução e encerramento.**

Assinale:

Questão 11 (FGV, 2013)

a) Se somente a afirmativa I estiver correta.

b) Se somente a afirmativa II estiver correta.

c) Se somente a afirmativa III estiver correta.

d) Se todas as afirmativas estiverem corretas.

e) Se somente as afirmativas I e II estiverem corretas.

Questão 11 (FGV, 2013)

a) Se somente a afirmativa I estiver correta.

b) Se somente a afirmativa II estiver correta.

c) Se somente a afirmativa III estiver correta.

d) Se todas as afirmativas estiverem corretas.

e) Se somente as afirmativas I e II estiverem corretas.

Comentários – questão 11

Afirmativa I – correta.

- **Justificativa. Gerenciamento de projetos é a aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto a fim de atender a seus requisitos, é a definição do próprio PMBOK.**

Afirmativa II – correta.

- **Justificativa. O gerenciamento é realizado pelas 10 áreas de conhecimento, divididas em 42 processos.**

Afirmativa III – incorreta.

- **Justificativa. Os processos são agrupados em CINCO grupos: iniciação, planejamento, execução, controle e encerramento.**

Questão 12

- **Normalmente, empresas que não possuem gerência nos seus processos não apresentam visibilidade do progresso e da qualidade do processo, comprometendo a qualidade do produto final.**

- **Dentre as alternativas a seguir, assinale o item correto em relação às características de organizações sem gerência.**

Questão 12 (FGV, 2013)

a) Os resultados esperados para custo, cronograma, funcionalidade e qualidade do produto usualmente são obtidos.

b) A fidelidade ao processo é objeto de auditoria e de controle.

c) Cronogramas e orçamentos são baseados em desempenho histórico e são realísticos.

d) É difícil obter uma base objetiva para julgar a qualidade do produto ou para resolver problemas de processo ou de produto.

e) Os papéis e as responsabilidades no processo estão claros ao longo de todo o projeto e para toda organização.

Questão 12 – resposta

a) Os resultados esperados para custo, cronograma, funcionalidade e qualidade do produto usualmente são obtidos.

b) A fidelidade ao processo é objeto de auditoria e de controle.

c) Cronogramas e orçamentos são baseados em desempenho histórico e são realísticos.

d) É difícil obter uma base objetiva para julgar a qualidade do produto ou para resolver problemas de processo ou de produto.

e) Os papéis e as responsabilidades no processo estão claros ao longo de todo o projeto e para toda organização.

Comentários – questão 12

A – Alternativa incorreta.

- **Justificativa. Os resultados esperados para custo, cronograma, funcionalidade e qualidade do produto usualmente são obtidos com processos de gerência**

B – Alternativa incorreta.

- **Justificativa. Sem gerência não há auditoria e controle.**

C – Alternativa incorreta.

- **Justificativa. Não há registros de desempenho histórico e são realísticos.**

Comentários – questão 12

D – Alternativa correta.

- **Justificativa. É difícil obter uma base objetiva para julgar a qualidade do produto ou para resolver problemas de processo ou de produto.**

E – Alternativa incorreta.

- **Justificativa. Os papéis e as responsabilidades no processo não são claros para toda organização.**

INTERVALO

Consultas e linguagem SQL

- Durante muito tempo, para obter as informações contidas nas tabelas dos bancos de dados, era preciso saber fisicamente como essas informações estavam armazenadas e organizadas no banco.
- O programador precisava conhecer profundamente a forma como o banco funcionava para poder conseguir os dados necessários. Isso levava a um problema de acoplamento: qualquer mudança no banco afetaria a estrutura do código do programa, mesmo que essa mudança não fosse conceitual e fosse apenas armazenamento dos dados.
- Ao mesmo tempo, mudanças no código também afetavam e costumavam levar a modificações na estrutura do banco de dados.

Consultas e linguagem SQL

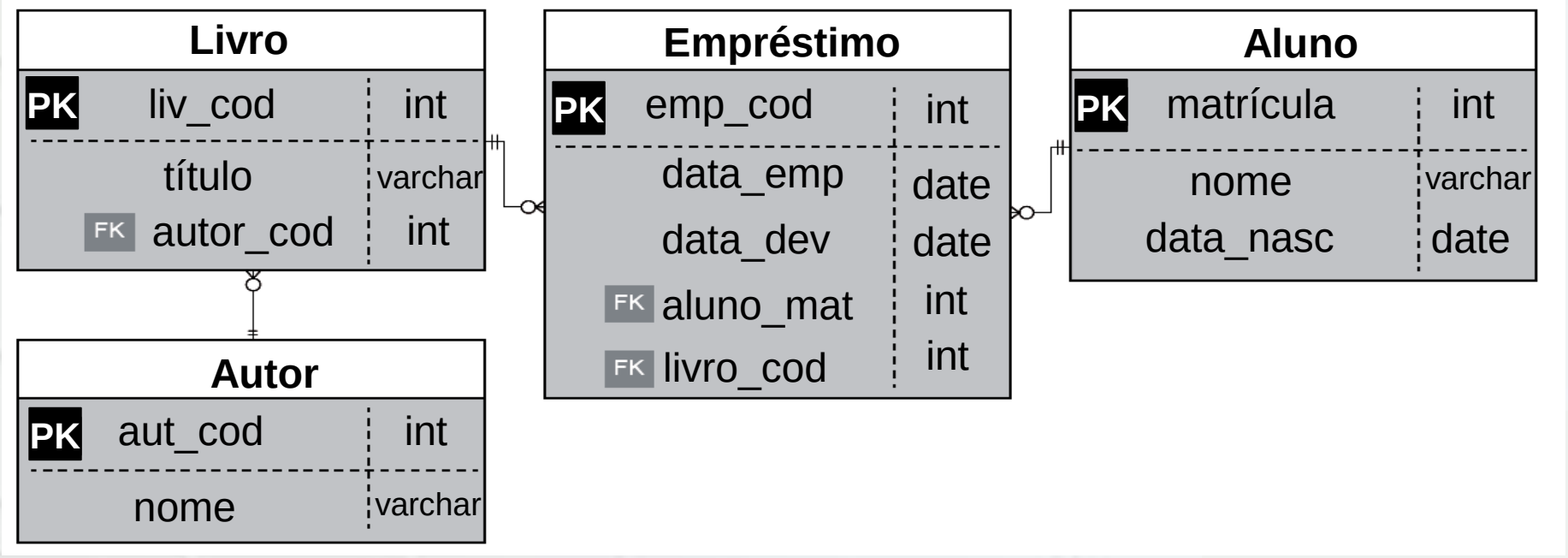
- Para simplificar o desenvolvimento de programas que utilizam bancos de dados, a linguagem SQL foi desenvolvida, partindo-se de um paradigma declarativo, em vez de um paradigma imperativo. Por exemplo, a linguagem C utiliza um paradigma imperativo, enquanto a linguagem Lisp utiliza um paradigma funcional e declarativo.
- Dessa forma, o programador deve concentrar-se no resultado que ele busca (os registros desejados do banco), e não na forma como esses registros são encontrados no banco de dados. Além disso, a linguagem SQL foi desenvolvida especificamente a partir da consulta a banco de dados relacionais.

Consultas e linguagem SQL

- As principais operações SQL são: *insert*, *update*, *delete* e *select*.
- Esses comando utilizam as cláusulas de relacionamento por meio das chaves das tabelas e condições para encontrar as informações.

Questão 13 (ENADE, 2014)

O modelo de entidade relacionamento apresentado representa, de forma sucinta, uma solução para persistência de dados de uma biblioteca. Considerando que um livro está emprestado quando possuir um registro vinculado a ele na tabela "Empréstimo", e essa dupla não possuir valor na coluna "data_dev", o comando SQL que deve ser utilizado para listar os títulos dos livros disponíveis para empréstimo é:



Questão 13 (ENADE, 2014)

A. select titulo from livro
except
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null

B. select titulo from livro
union
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null

Questão 13 (ENADE, 2014)

C. select titulo from livro
except
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is not null

D. select titulo from livro
union select l.titulo from emprestimo e left join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null

E. select titulo from livro
except
select l.titulo from emprestimo e right join livro l
on e.livro_cod = l.liv_cod where e.data_dev is not null

Questão 13 – resposta

A. select titulo from livro
except
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null

B. select titulo from livro
union
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null

Questão 13 – resposta

C. select titulo from livro
except
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is not null

D. select titulo from livro
union select l.titulo from emprestimo e left join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null

E. select titulo from livro
except
select l.titulo from emprestimo e right join livro l
on e.livro_cod = l.liv_cod where e.data_dev is not null

Comentários – questão 13

- O resultado desejado corresponde a “encontrar todos os livros, exceto aqueles que estão emprestados”.

- Um livro está emprestado se está presente na tabela “Empréstimo” e não ter valor na coluna “data_dev” (ou seja, essa coluna deve conter o valor NULL).

- Dessa forma, para selecionarmos todos os livros que estão emprestados, fazemos o que segue.

```
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null
```

Comentários – questão 13

- Lembrando que o comando “inner join” retorna apenas os registros que apresentam o mesmo valor na coluna “livro_cod” da tabela “empréstimo” e da coluna “liv_cod” na tabela “livro”.

- Além disso, observe a condição “e.data_dev is null”, ou seja, apenas os registros que não apresentarem valores em “data_dev”.

Se quiséssemos selecionar todos os livros, independentemente de estarem emprestados ou não, deveríamos fazer:

select titulo from livro

Comentários – questão 13

- Essa consulta vai retornar todos os títulos presentes na tabela livro. Excluiremos os registros que estão emprestados, retornados na consulta anterior. Isso pode ser feito utilizando-se o comando except, que elimina, da primeira consulta, os registros obtidos pela segunda consulta, conforme segue.

```
select titulo from livro
except
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null
```

Arquitetura de *software*

- Uma das maiores dificuldades ao se criar um novo *software* é garantir uma arquitetura que seja suficientemente flexível para poder crescer e incorporar novas funcionalidades, conforme as necessidades do cliente, mas que também seja robusta, para permitir que esse crescimento possa ser feito com o menor esforço possível.
- Essas duas necessidades (flexibilidade e organização), com frequência, caminham em direções aparentemente opostas.
- Na realidade, o problema é que é muito mais fácil fazer um programa crescer de forma desordenada do que crescer de forma ordenada.
- A falta de organização pode ser falsamente percebida como flexibilidade. O desafio do arquiteto de *software* está em encontrar uma arquitetura que englobe tanto flexibilidade quanto organização.

UNIP

Arquitetura de *software*

- Havia pouco conhecimento sobre o desenvolvimento de programas grandes e complexos e não havia muita preocupação com aspectos arquitetônicos de um programa.
- Frequentemente, a postura era a de solucionar o problema da forma mais rápida possível, sem nenhuma preocupação com o futuro do programa. Isso levou à geração de uma grande quantidade de códigos "não gerenciáveis": códigos com má qualidade e com má organização, difíceis de manter e de modificar.
- O aumento da demanda por sistemas computacionais e o crescente uso de computadores levaram ao surgimento da engenharia de *software*. Um de seus objetivos é sistematizar a produção do *software*, incluindo boas práticas no projeto e na codificação de programas.

UNIP

Arquitetura de *software*

- Um dos aspectos fundamentais dessa área é a elaboração de uma arquitetura de *software* que seja suficientemente flexível para permitir o crescimento futuro, mas também organizada e gerenciável, para permitir que o desenvolvimento seja previsível, especialmente do ponto de vista do esforço necessário para a introdução de novas características, com o menor número possível de defeitos.
- Uma das arquiteturas mais difundidas para o desenvolvimento de sistemas computacionais é a chamada arquitetura em camadas.
- Nessa arquitetura, cada camada do sistema tem responsabilidades específicas e uma interface bem definida. Por responsabilidade, queremos dizer que cada camada apresenta um objetivo claro e bem delimitado.

UNIP

Arquitetura de *software*

- **Por interface, queremos dizer que o acesso aos recursos da camada se faz por meio de métodos (funções ou procedimentos, em alguns casos) claramente especificados, padronizados e estáveis. No entanto, os detalhes técnicos de cada camada devem estar suficientemente isolados, de forma que, para a utilização da interface, não seja necessário conhecer os detalhes internos da sua implementação.**
- **Ao se criar uma arquitetura de *software* mais sofisticada, surge a necessidade de mão de obra mais qualificada, bem como de um processo de gerenciamento mais adequado. Costuma-se dizer que o preço que se paga para se ter uma complexidade gerenciável é o aumento da complexidade de cada módulo do sistema.**
- **Esse aumento tem como objetivo elevar a qualidade do *software* desenvolvido, o que torna o desenvolvimento mais previsível e o programa mais flexível.**

Questão 14 (ENADE, 2014)

- Nos anos 1970, os sistemas executavam em *mainframes* com aplicativos escritos em linguagens estruturadas e com todas as funcionalidades em um único módulo, com grande quantidade de linhas de código.
- Acessos a bancos de dados não relacionais, regras de negócios e tratamento de telas para terminais "burros" ficavam no mesmo programa. Posteriormente, uma importante mudança ocorreu: a substituição dos terminais "burros" por microcomputadores, permitindo que todo o tratamento da interface, e de algumas regras de negócios, passassem a ser clientes feitas nas estações.
- Surgiam as aplicações cliente-servidor. A partir dos anos 1990 até os dias atuais, as mudanças foram mais radicais, os bancos de dados passaram a ser relacionais e distribuídos.

Questão 14 (ENADE, 2014)

- **As linguagens passaram a ser orientadas a objetos, cuja modelagem encapsula dados e oferece funcionalidades por meio de métodos. A interface passou a ser *web*. Vive-se a era das aplicações em três camadas.**

Considerando a evolução da arquitetura de *software* de sistemas de informação, conforme citado no texto anterior, avalie as seguintes afirmações:

Questão 14 (ENADE, 2014)

I. A separação dos sistemas em três camadas lógicas torna os sistemas mais complexos, requerendo pessoal mais especializado.

II. A separação dos sistemas em três camadas lógicas torna os sistemas mais flexíveis, permitindo que as partes possam ser alteradas de forma independente.

III. A separação dos sistemas em três camadas lógicas aumentou o acoplamento, dificultando a manutenção.

É correto o que se afirma em:

Questão 14 (ENADE, 2014)

a) II, apenas.

b) III, apenas.

c) I e II, apenas.

d) I e III, apenas.

e) I, II e III.

Questão 14 (ENADE, 2014)

a) II, apenas.

b) III, apenas.

c) I e II, apenas.

d) I e III, apenas.

e) I, II e III.

Comentários – questão 14

I. Afirmativa correta.

- Justificativa. A divisão de um sistema em camadas certamente implica um aumento da complexidade do código, que deve ter uma arquitetura cuidadosamente projetada para separar responsabilidades adequadas para cada uma das camadas. Contudo, quando devidamente gerenciado, esse aumento de complexidade pode ser compensado, em longo prazo, pela maior facilidade no gerenciamento do desenvolvimento do *software* e pela maior facilidade de manutenção.

Comentários – questão 14

II. Afirmativa correta.

Justificativa. Em um sistema dividido em camadas e com interfaces adequadas, a manutenção e a extensão são muito mais fáceis. Com o correto isolamento conceitual entre cada uma das camadas, alterações podem ser feitas de forma transparente em uma das camadas sem a necessidade de alterações em outras camadas.

III. Afirmativa incorreta.

Justificativa. O objetivo da separação em camadas é diminuir o acoplamento, e não aumentá-lo. Cada camada deve ser acessível por uma interface bem definida, com o objetivo de desacoplar as camadas e facilitar a extensão e a manutenção do código.

INTERVALO

Diagramas de estados e atividades

- O objetivo do diagrama de atividades é mostrar o fluxo de atividades em um único processo.
- O diagrama mostra como uma atividade depende uma da outra, como um fluxograma.

Podem especificar:

- Modelo de negócio – sistema todo.
- Detalhar um caso de uso.
- Descrever uma operação de uma classe.

Diagramas de estados e atividades

- Modelam o tempo de vida de um objeto.
- Complementam a descrição de uma classe.
- Construídos somente para classes que possuam estados bem definidos e sejam relevante para o negócio.
- Identificam os eventos que alteram os estados dos objetos.
- Aplicados em comportamentos de classes.

Questão 15

- Os diagramas de estados descrevem os estados possíveis pelos quais os objetos de uma classe podem passar e as modificações dos estados como resultado de estímulos que atingem esses objetos. Nesses diagramas, conseguimos especificar a ordem válida dos estados em que os objetos da classe podem passar ao longo de seu ciclo de vida. Sobre esse tema, analise as afirmativas.

I. Os diagramas de estado são utilizados para modelar um comportamento discreto em sistemas de transição entre estados finitos.

II. Os diagramas de estado são utilizados para modelar um caso de uso discreto em sistemas de transição entre estados dependentes.

III. Um diagrama de estados deve estar relacionado a uma classe modelada no diagrama de classes.

É correto apenas o que se afirma em:

UNIP

Questão 15

a) II.

b) I.

c) I e III.

d) II e III.

e) III.

Questão 15

a) II.

b) I.

c) I e III.

d) II e III.

e) III.

Comentários – questão 15

I. Alternativa correta.

Justificativa. Os diagramas de estado representam o comportamento de um objeto do sistema.

II. Alternativa incorreta.

- Justificativa. Os diagramas de estado não são aplicáveis a casos de uso.

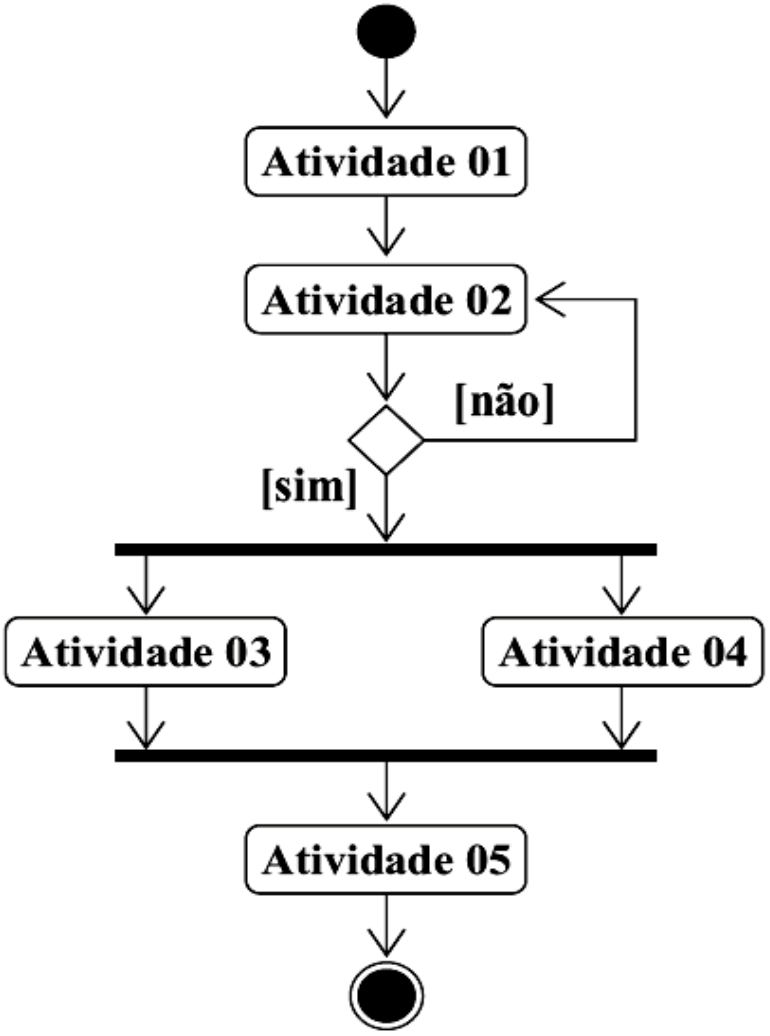
III. Alternativa correta.

- Justificativa. Um diagrama de estado representa as mudanças de estado de um objeto.

Questão 16

- Os diagramas de atividades são elaborados com a intenção de promoverem o entendimento provido da descrição de um caso de uso complexo, normalmente com várias atividades, conforme exemplificado na figura a seguir.

- Com relação ao diagrama de atividades apresentado, ele descreve uma sequência de atividades com suporte para comportamentos.



Questão 16

a) Condicional e paralelo.

b) Relacional e individual.

c) Funcional e não funcional.

d) De fluxo e de estado.

e) Inicial e final.

Questão 16

a) Condicional e paralelo.

b) Relacional e individual.

c) Funcional e não funcional.

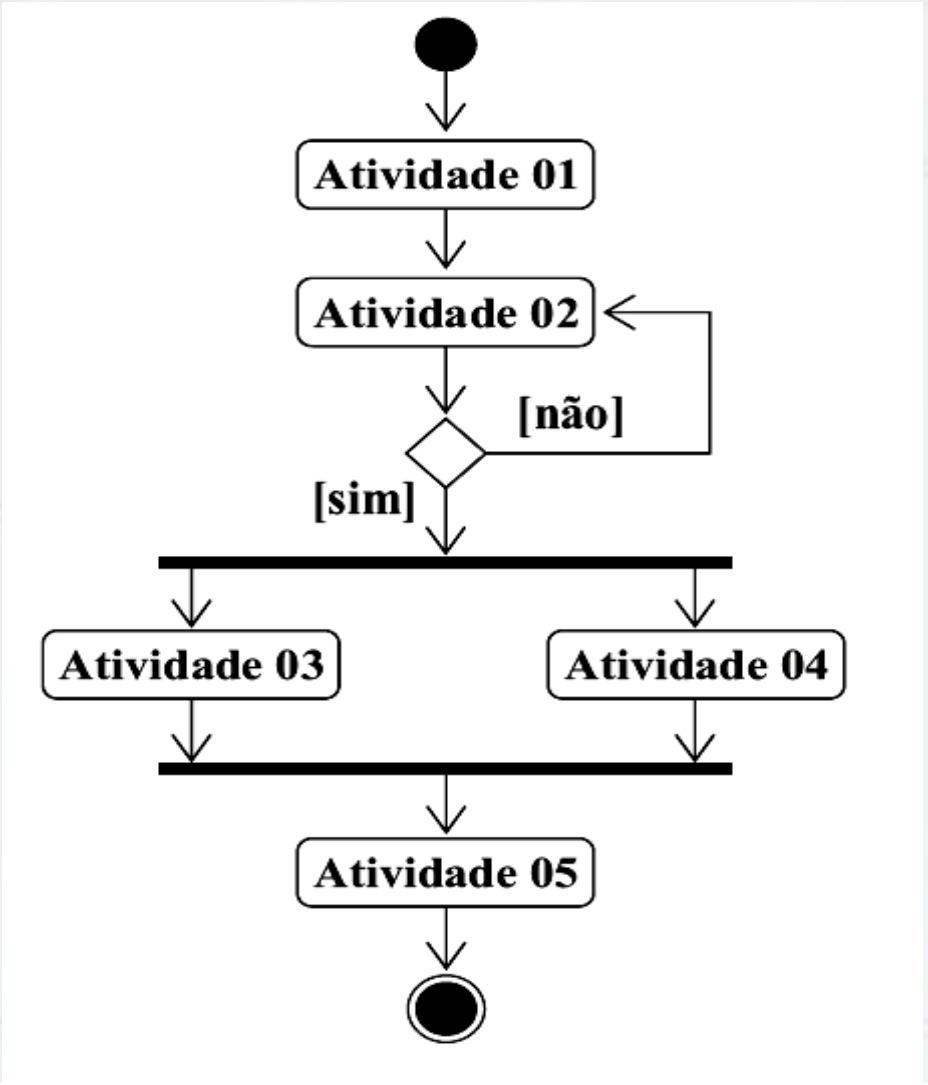
d) De fluxo e de estado.

e) Inicial e final.

Comentários – questão 16

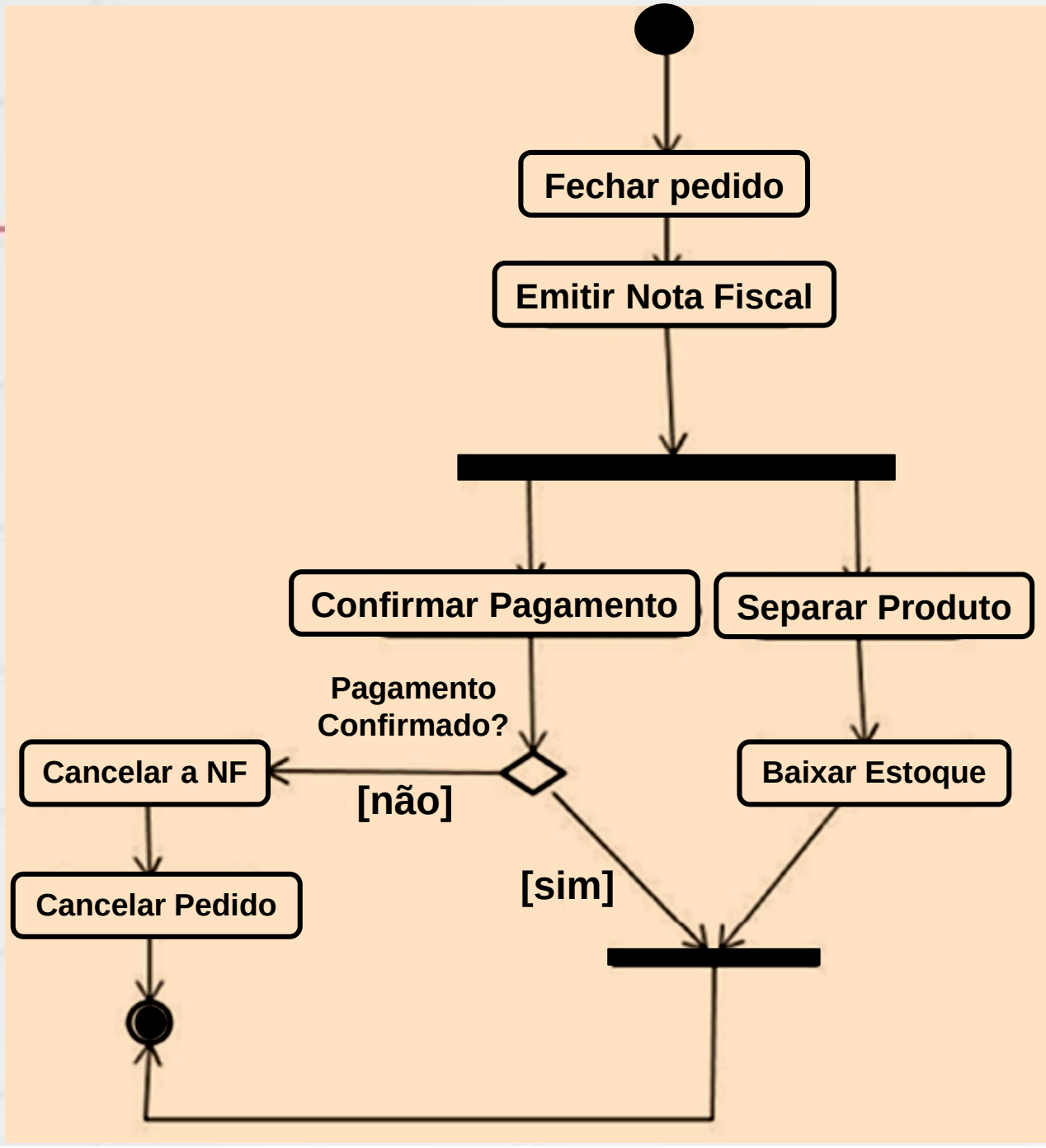
A – Alternativa correta.

Justificativa. O diagrama de atividades representa o fluxo de atividades de um processo, podendo apresentar comportamentos condicionais e paralelos.



Questão 17

- Com relação ao diagrama a seguir, assinale a alternativa correta.



Questão 17

a) As ações confirmar pagamento e separar estoque são ações paralelas.

b) Ao cancelarmos a NF e o pedido, precisamos incluir o produto no estoque, pois ele foi baixado com a ação baixar estoque.

c) Esse diagrama de caso de uso se refere ao processo de faturamento de uma empresa.

d) Ao emitir a nota fiscal, o processo é parado para confirmar pagamento e verificar se possui material em estoque.

e) Esse diagrama de estado apresenta as modificações finais que um pedido pode sofrer.

Questão 17

a) As ações confirmar pagamento e separar estoque são ações paralelas.

b) Ao cancelarmos a NF e o pedido, precisamos incluir o produto no estoque, pois ele foi baixado com a ação baixar estoque.

c) Esse diagrama de caso de uso se refere ao processo de faturamento de uma empresa.

d) Ao emitir a nota fiscal, o processo é parado para confirmar pagamento e verificar se possui material em estoque.

e) Esse diagrama de estado apresenta as modificações finais que um pedido pode sofrer.

Comentários – questão 17

A – Alternativa correta.

- Justificativa. As ações são paralelas representadas pelo símbolo de bifurcação.

B – Alternativa incorreta.

- Justificativa. Não há ação de inclusão de produto representado após o cancelamento da NF/Pedido.

C – Alternativa incorreta.

- Justificativa. Esse não é um diagrama de casos de uso, é um diagrama de atividades.

D – Alternativa incorreta.

- Justificativa. O símbolo após emitir NF é de bifurcação.

E – Alternativa incorreta.

- Justificativa. Esse não é um diagrama de estados, é um diagrama de atividades.

ATÉ A PRÓXIMA!

ESTUDOS DISCIPLINARES

Formação Específica

Prof. André Luiz

Usabilidade

- Segundo Koscianski e Soares (2007), usabilidade corresponde à “facilidade” ou à “dificuldade” na utilização de determinado sistema por um usuário.
- Um sistema empregado por cientistas tem provavelmente um critério de usabilidade diferente de um sistema utilizado por artistas ou profissionais de outras áreas.
- Algumas normas foram desenvolvidas para auxiliar no processo de avaliação da usabilidade de um programa.
- A Norma ISO 9241 provê “requisitos, recomendações e princípios para as atividades de projeto centradas em pessoas ao longo do ciclo de vida de sistemas interativos baseados em computador” (ISO 9241-210,2010). Essa norma refere-se aos requisitos ergonômicos do *software*.

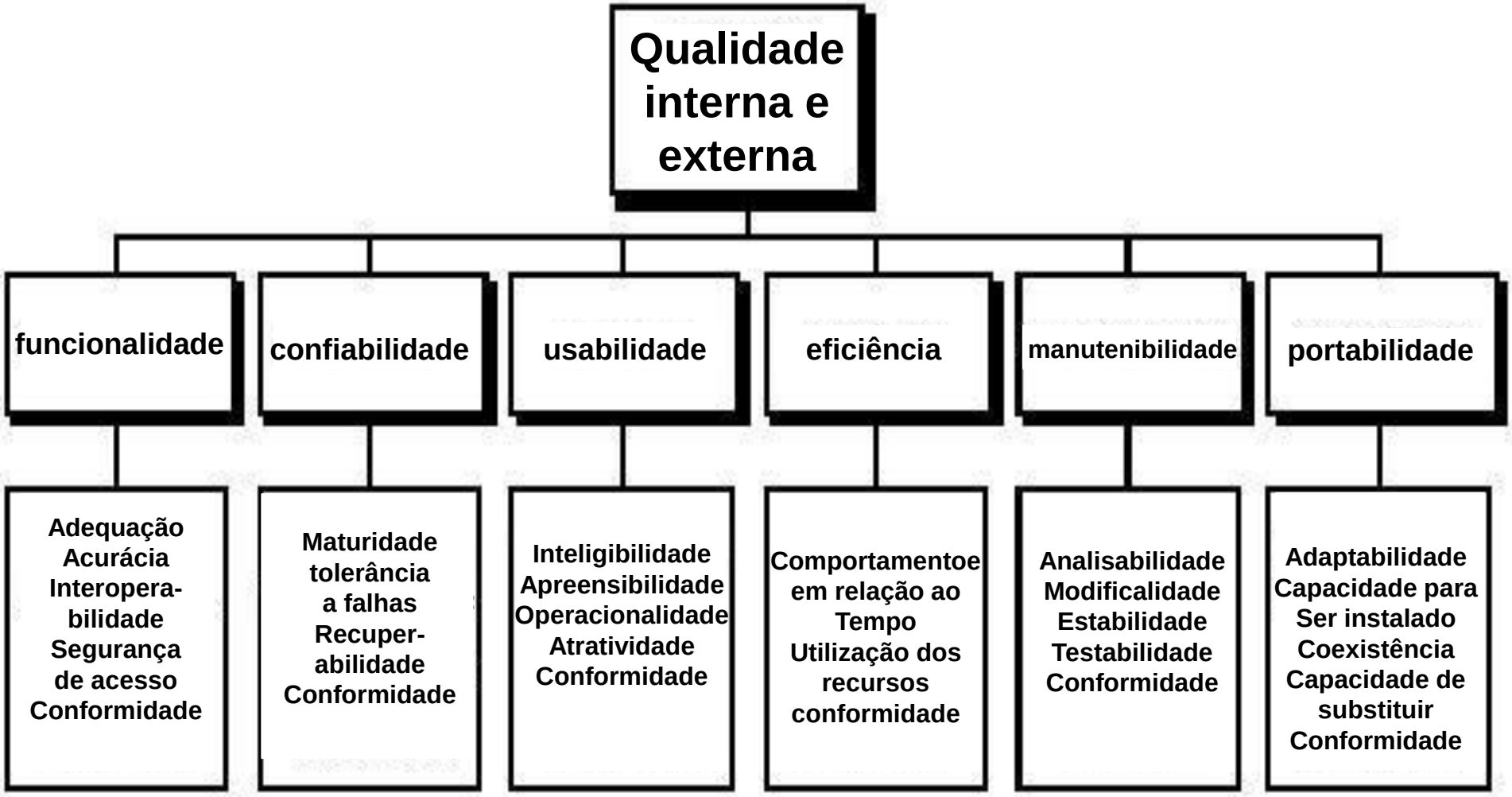
Usabilidade

- **Outra norma importante para a área é a Norma ISO 9126, reorganizada pela Norma ISO/IEC 25000. A Norma ISO 9126 é constituída por várias partes, conforme tabela:**

Norma	Conteúdo
9126-1	Modelo de qualidade de *software*
9126-2	Métricas externas
9126-3	Métricas internas
9126-4	Métricas para qualidade em uso

Norma ISO-9126 e suas partes.
Fonte. KOSCIANSKI e SOARES, 2007 (com adaptações).

Usabilidade – ISO9126



Fonte. O autor.

Usabilidade

- **Adicionalmente, a Norma ISO-14598, que também foi incorporada à norma ISO/IEC 25010:2011, define processos de avaliação e documentação. Ela é conhecida como SQuaRE (*Systems and software Quality Requirements and Evaluation*).**

Essas normas definem uma série de subcaracterísticas da usabilidade, como:

- **operabilidade;**
- **compreensibilidade;**
- **apreensibilidade;**
- **atratividade.**

Usabilidade

- A operabilidade está ligada à forma como o *software* pode ser controlado pelo usuário.
- A compreensibilidade refere-se à capacidade do usuário em compreender se o programa é adequado ou não às suas tarefas.
- A apreensibilidade aplica-se à facilidade do usuário em aprender a utilizar o programa.
- A atratividade está vinculada à capacidade de o usuário manter a atenção na utilização do programa (KOSCIANSKI E SOARES, 2007).

Questão 18 (ENADE, 2014)

- Vivemos em um mundo de produtos de alta tecnologia e praticamente todos requerem interação humana. Para que um produto de *software* seja bem-sucedido, deve apresentar boa usabilidade. Se os mecanismos de interface tiverem sido bem projetados, o usuário flui suavemente através da interação usando um ritmo cadenciado que permite que o trabalho seja realizado sem grandes esforços. Entretanto, se a interface for mal concebida, o usuário se move aos trancos e barrancos, e o resultado será frustração e baixa eficiência no trabalho.

Três regras de ouro são a base para um conjunto de princípios para o projeto de interfaces do usuário:

- 1 - Deixar o usuário no comando;
- 2 - Reduzir a carga de memória do usuário;
- 3 - Tornar a interface consistente.

PRESSMAN, R.S. Engenharia de Software: uma abordagem profissional. 7. ed. McGraw Hill, 2011, p.287-288 (com adaptações).

Questão 18 (ENADE, 2014)

- Com base nessas três regras, avalie as afirmativas a seguir.

I. Um sistema que permita ao usuário desfazer qualquer ação, respeita a regra de ouro 1.

II. Um sistema de pagamento de contas que usa uma imagem de um cartão de crédito para orientar o usuário pelo processo de pagamento de uma conta, respeita a regra de ouro 2.

III. Um conjunto de aplicações ou produtos que implementam as mesmas regras de projeto de modo padronizado, respeita a regra de ouro 3.

- É correto o que se afirma em:

Questão 18 (ENADE, 2014)

a) I, apenas.

b) II, apenas.

c) I e III, apenas.

d) II e III, apenas.

e) I, II e III.

Questão 18 Resposta

a) I, apenas.

b) II, apenas.

c) I e III, apenas.

d) II e III, apenas.

e) I, II e III.

Comentários – Questão 18

I – Afirmativa correta.

- **Justificativa. Sem oportunidade de desfazer ações, um usuário pode sentir apreensão ao utilizar o sistema e ter a sensação de que tem pouco controle sobre o funcionamento do *software*.**

II – Afirmativa correta.

- **Justificativa. A imagem de um cartão remete facilmente à função desejada. Dessa forma, o usuário não precisa memorizar nenhum comando ou nome especial, de acordo com a regra 2.**

Comentários – Questão 18

III – Afirmativa correta.

- **Justificativa. A consistência entre as diversas telas e funcionalidades é fundamental. Durante o uso de um programa, o usuário tende a memorizar a forma como o sistema funciona, de modo que a falta de padrão entre diferentes telas e funcionalidades pode desnortear o usuário.**

INTERVALO

Requisitos não funcionais

- Projetos de *software* envolvem a identificação de diversos tipos de requisitos. Alguns deles estão diretamente atrelados ao comportamento do *software* sob determinada condição (WIEGERS e BEATTY, 2013), enquanto outros referem-se a propriedades do sistema que não estão diretamente ligados à funcionalidade principal e, por isso, costumam ser chamados de requisitos não funcionais (WAGNER, 2013).
- Frequentemente, os requisitos não funcionais estão atrelados a aspectos da qualidade de *software* (WAGNER, 2013).
- Devido aos diversos usos que a palavra “requisito” pode ter na área de *software*, alguns autores, como Wiegers e Beatty (2013), costumam fornecer um vocabulário dos vários possíveis significados associados a esse termo e a outros correlatos.

Requisitos não funcionais

Termo	Definição
Requisito de negócio	Um objetivo de negócio de alto nível de que uma organização dispõe na construção de um produto.
Regra de negócio	Uma prática, uma diretriz, um padrão ou uma regulamentação que define ou restringe algum aspecto do negócio. Está na origem de diversos tipos de requisitos de *software*.
Restrição	A restrição no projeto e a construção de um produto que são impostas nas escolhas disponíveis para o desenvolvedor.
Requisito de interface externa	A descrição de uma conexão entre um sistema de *software* e um usuário, um outro sistema ou um equipamento de que a empresa dispõe.

Tabela. Tipos de requisitos.
Fonte. WIEGERS e BEATTY, 2013 (com adaptações).

Requisitos não funcionais

Termo	Definição
Característica (feature)	Uma ou mais competências do sistema, que são descritas por um conjunto de requisitos funcionais.
Requisitos funcionais	A descrição de um comportamento, que é exibido pelo sistema em condições específicas.
Requisitos não funcionais	Uma descrição ou uma característica de uma propriedade que um sistema exibe ou uma restrição a ser respeitada.

Tabela Tipos de requisitos.
Fonte. WIEGERS e BEATTY, 2013 (com adaptações).

Requisitos não funcionais

Termo	Definição
Atributo de qualidade	Um tipo de requisito não funcional que descreve um serviço ou uma característica da performance de um produto.
Requisito de sistema	Um requisito de alto nível para um produto que contém múltiplos subsistemas e pode referir-se a todo o *software* ou ao *software* e ao *hardware*.
Requisito de usuário	Um objetivo ou uma tarefa específica para uma classe de usuários, a que o sistema precisa ser capaz de atender.

Tabela. Tipos de requisitos.
Fonte. WIEGERS e BEATTY, 2013 (com adaptações).

Requisitos não funcionais

- Ainda em Wiegers e Beatty (2013), encontra-se um capítulo totalmente dedicado às qualidades externas e internas na área de *software*.
- As qualidades externas estão ligadas aos atributos do *software*, perceptíveis quando o *software* é executado, enquanto as qualidades internas contribuem de forma indireta para a satisfação do usuário.
- Entre os aspectos externos de qualidade, podemos destacar os que seguem.

Requisitos não funcionais

- **Usabilidade: facilidade que o usuário tem de aprender, memorizar e utilizar o sistema.**
- **Segurança: sistema que protege o usuário contra danos (isso é especialmente importante em máquinas e dispositivos mecânicos) ou sistema que está protegido em relação ao seu acesso e ao acesso das informações nele contidas.**
- **Desempenho: agilidade com que o sistema responde às entradas dos usuários e a outros eventos.**
- **Confiabilidade: ligada ao tempo que o sistema executa sem experimentar uma falha.**

Questão 19 (ENADE, 2014)

- Requisitos funcionais e não funcionais podem definir características e funcionalidades presentes na interface a ser desenvolvida para um sistema. Em relação aos requisitos não funcionais, também chamados de requisitos de qualidade, avalie as afirmativas a seguir.

I. São levantados e eliciados após os requisitos funcionais, uma vez que os requisitos funcionais determinarão as funcionalidades da interface.

II. Sempre serão definidos de forma mais concreta, através de requisitos funcionais, uma vez que o usuário manipula na interface somente as funcionalidades levantadas.

III. Podem complementar os requisitos funcionais.

- É correto o que se afirma em:

Questão 19 (ENADE, 2014)

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

Questão 19 (ENADE, 2014)

a) II, apenas.

b) III, apenas.

c) I e II, apenas.

d) I e III, apenas.

e) I, II e III.

Comentários – Questão 19

I - Afirmativa incorreta.

- **Justificativa. Os requisitos não funcionais não devem ser levantados apenas depois dos requisitos funcionais. Os requisitos não funcionais não são "menos importantes" do que os requisitos funcionais: eles são apenas diferentes.**

II - Afirmativa incorreta.

- **Justificativa. Os requisitos não funcionais, em geral, envolvem aspectos ligados à qualidade e à usabilidade e, em função disso, podem não estar totalmente ligados aos requisitos funcionais, como a afirmativa sugere.**

Comentários – Questão 19

III - Afirmativa correta.

- **Justificativa. Os requisitos não funcionais complementam os requisitos funcionais, incorporando elementos importantes para o cliente, por exemplo, a facilidade de utilização do sistema.**

Questão 20 (ENADE, 2014)

- **A engenharia de *software* considera diversos aspectos para a garantia da qualidade. Os requisitos funcionais definem como um sistema deverá se comportar em relação as suas funcionalidades básicas, já os requisitos não funcionais avaliam outros aspectos do *software*.**

São exemplos de requisitos não funcionais a serem considerados em um *software*:

Questão 20 (ENADE, 2014)

a) segurança, desempenho, estresse e sistema.
b) usabilidade, segurança, aceitação e confiabilidade.
c) usabilidade, segurança, desempenho e confiabilidade
d) segurança, aceitação, testabilidade e confidencialidade.
e) usabilidade, confidencialidade, aceitação e confiabilidade.

Questão 20 (ENADE, 2014)

a) segurança, desempenho, estresse e sistema.

b) usabilidade, segurança, aceitação e confiabilidade.

c) usabilidade, segurança, desempenho e confiabilidade.

d) segurança, aceitação, testabilidade e confidencialidade.

e) usabilidade, confidencialidade, aceitação e confiabilidade.

Comentários – Questão 20

A - Alternativa incorreta.

- **Justificativa. Estresse e sistema não são exemplos de requisitos não funcionais.**

B, D e E - Alternativas incorretas.

- **Justificativa. A aceitação não é um requisito não funcional.**

C - Alternativa correta.

- **Justificativa. Usabilidade, segurança, desempenho e confiabilidade são bons exemplos de requisitos não funcionais.**

INTERVALO

UML

- De acordo com Booch, Rumbaugh e Jacobson (2006), a UML (*Unified Modeling Language*) é definida como “uma linguagem-padrão para a elaboração da estrutura de projetos de *software*”.
- Os autores destacam os quatro objetivos básicos da UML, quanto aos artefatos de um sistema complexo de *software*: visualizar, especificar, construir e documentar.
- O principal objetivo da UML é estabelecer um modelo ou representar conceitual e fisicamente um sistema (BOOCH, RUMBAUGH e JACOBSON; 2006).

UML para visualização

- Um projeto de *software* envolve mais do que o código-fonte. Além disso, um mesmo problema pode ser resolvido de diferentes formas, com códigos muito distintos. Ainda que programas com códigos diferentes possam atender aos mesmos requisitos, nem todas as soluções são igualmente úteis ou interessantes.
- Se dispusermos apenas do código-fonte de um programa, sem nenhum tipo de documentação, o entendimento da solução fica muito mais difícil. Programas comerciais podem ser muito longos e conter milhões de linhas de código-fonte. Dessa forma, é importante ter uma maneira de visualizar o todo, a estrutura geral de um *software*, sem que, necessariamente, haja a obrigação de se ler o código-fonte de modo completo.
- A UML fornece uma linguagem-padrão que permite que diferentes pessoas e empresas façam e interpretem diagramas e modelos de uma mesma forma.

UNIP

UML para especificação

- Antes de escrevermos o código para um programa, é preciso conhecer quais são as necessidades do cliente. Além disso, grandes projetos envolvem problemas complexos de *software*.
- Muitas empresas contratam profissionais especializados para encontrar a melhor solução, como os arquitetos de *software*.
- Esses profissionais precisam de uma linguagem para especificar, construir e documentar a solução e a UML oferece uma linguagem especificamente desenvolvida para esse propósito.

UML para construção

- Ainda que a UML não seja uma linguagem de programação propriamente dita, é possível estabelecer um mapeamento entre os modelos construídos pela UML e o código-fonte em diversas linguagens de programação, como Java ou C# (BOOCH, RUMBAUGH e JACOBSON, 2006).

- Essa facilidade do mapeamento vem do alto grau de expressividade aliado à baixa ambiguidade, o que não é o caso quando consideramos outras formas de especificação, como um texto puro. Essa precisão faz com que existam ferramentas capazes de converter alguns diagramas UML em código-fonte com pouca ou nenhuma intervenção do usuário.

UML para documentação

- De acordo com Booch, Rumbaugh e Jacobson (2006), um projeto de *software* envolve mais do que apenas o código-fonte. Empresas de *software* costumam produzir diversos outros elementos, chamados de artefatos, como:
- requisitos do *software*;
- documentações da arquitetura;
- projetos e planos de projeto;
- código-fonte;
- testes e planos de teste;
- protótipos.
- A UML é utilizada para documentar diversos desses artefatos, de modo formal e com baixa ambiguidade.

Diagrama de Classes

- Formalmente, define-se uma classe como “uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica” (BOOCH, RUMBAUGH e JACOBSON; 2006).
- É importante diferenciar as classes dos objetos: uma classe define um conjunto de objetos, mas não é um objeto. Por exemplo, podemos definir a classe das “xícaras de café” como sendo objetos cerâmicos, ocos, que servem para armazenar uma quantidade pequena de líquido quente. Essa classe abrange todas as possíveis xícaras, de diferentes formas e tamanhos. Se tomarmos um exemplo específico, como uma xícara de restaurante, dizemos que essa xícara é um objeto, ou uma instância específica de uma classe. Utilizamos uma classe para definir uma categoria de objetos similares.

UNIP

Diagrama de Classes

- A UML apresenta um diagrama específico para a representação das classes, chamado de diagrama de classes. Nesse diagrama, as classes são representadas por retângulos contendo nomes, atributos e operações, como ilustrado na figura abaixo. Os atributos representam propriedades associadas a cada um dos objetos que serão instanciados pela classe. Por exemplo, a classe "Cliente" pode ter um atributo chamado de "nome". Outros atributos que essas classes podem ter são: "endereço", "telefone" e "*e-mail*".

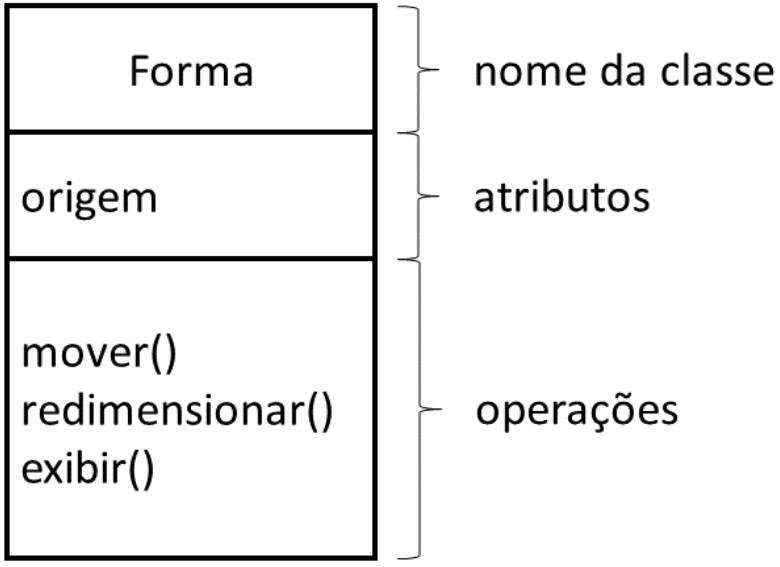


Figura. Representação de uma classe na UML.
Fonte. BOOCH, RUMBAUGH e JACOBSON, 2006 (com adaptações).

Diagrama de Classes

- Além de atributos, uma classe também apresenta comportamentos associados.
- Por exemplo, suponha um editor de imagens que tenha uma funcionalidade de desenho de formas geométricas, como losangos. Essa funcionalidade pode ser implementada por meio de uma classe, chamada de “Losango”, que apresenta uma série de comportamentos associados, como a movimentação do losango e a alteração do seu tamanho,
entre outras. Essas operações são representadas no retângulo da classe, logo abaixo dos atributos.
- As operações têm um nome e terminar com parênteses “()”: elas são similares a “funções” e podem receber parâmetros. Os parâmetros são representados nos parênteses e separados por vírgulas (no caso da existência de mais de um parâmetro).

UNIP

Diagrama de Classes

- As diversas classes que compõem um programa podem estar relacionadas entre si e existem, essencialmente, três tipos de relacionamentos entre classes: Dependência, generalização e associação.
- Utilizamos uma relação de dependência quando queremos dizer que uma classe necessita de outra classe para o seu funcionamento.
- Suponhamos uma classe chamada “Cachorro” e uma classe chamada “Animal”. Podemos dizer que “Animal” representa uma generalização de “Cachorro”, uma vez que um cachorro é um animal, mas nem todo animal é um cachorro.
- Uma associação estabelece o motivo pelo qual duas classes relacionam-se e estabelece as regras que controlam esse relacionamento.

UNIP

Questão 21 (ENADE, 2014)

- **UML é uma linguagem padrão para desenvolver e documentar projetos de software e permite que desenvolvedores visualizem os produtos de seus trabalhos em diagramas padronizados. Ela surgiu como uma proposta de ser uma linguagem para modelagem de dados que usava diversos artefatos para representar o modelo de negócio e um desses artefatos é o diagrama de classes.**
- **PRESSMAN, R.S. *Engenharia de software*. 6. ed. Porto Alegre: Bookman, 2006 (com adaptações).**
- **Se um projeto não tem a documentação apropriada ou se está com a documentação desatualizada, uma opção é a engenharia reversa que possibilita mapear códigos para diagramas UML. A seguir, é apresentado um código na linguagem de programação:**

Questão 21 (ENADE, 2014)

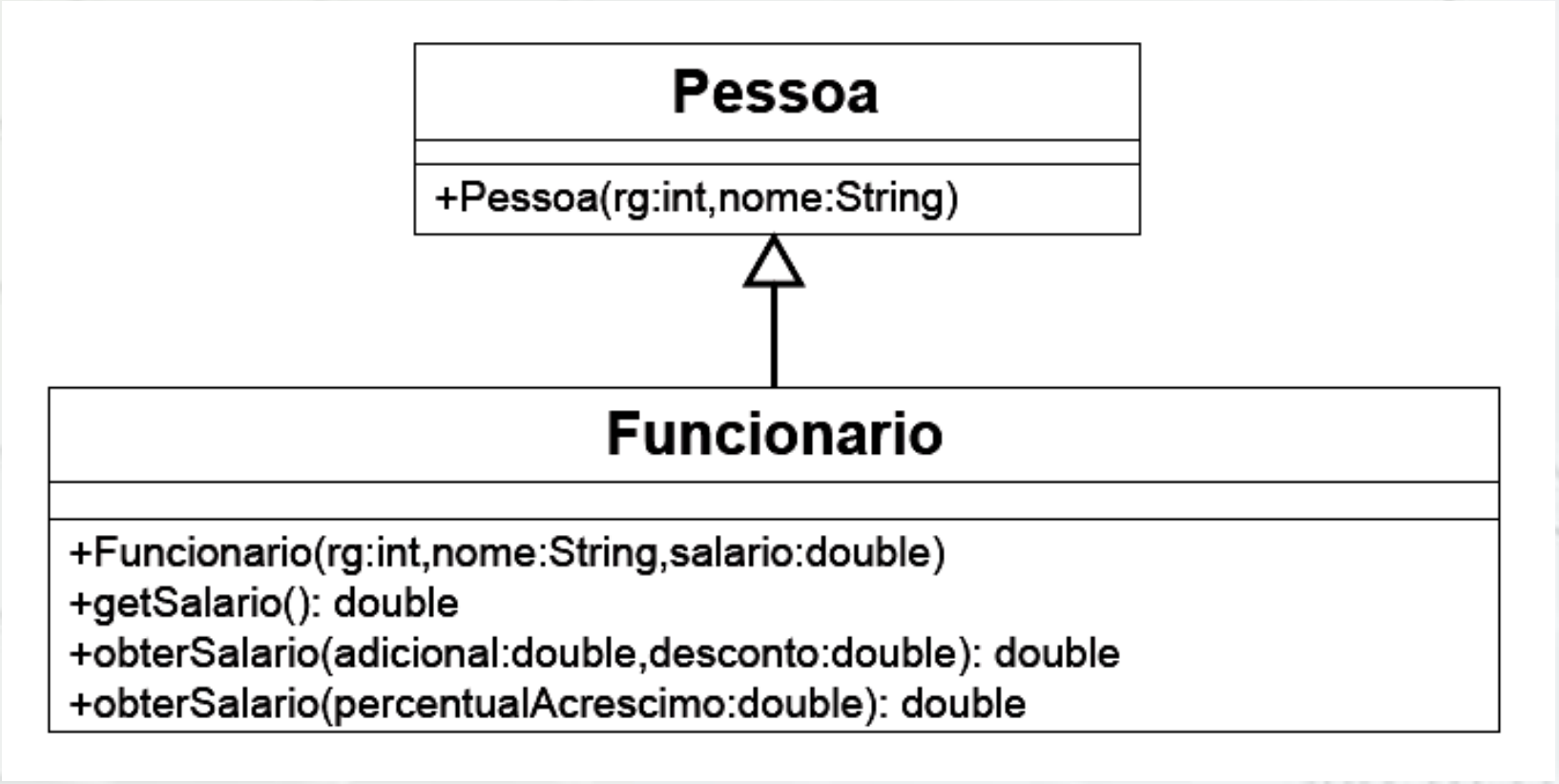
```
1 package default;
2
3 public class Funcionario extends Pessoa {
4    private double salario;
5
6    public Funcionario(int rg, String nome,
7                       double salario) {
8         super(rg, nome);
9         this.salario = salario;
10   }
11   public double getSalario() {
12        return salario;
13
14   }
15   public double obterSalario(double
16        percentualAcrescimo) {
17        double salarioReajustado = salario +
18             salario * percentualAcrescimo / 100;
19        return salarioReajustado;
20
21   }
22   public double obterSalario(double
23        adicional, double desconto){
24        return this.getSalario() + adicional - desconto;
25
26   }
27 }
```

Utilizando a engenharia reversa nesse trecho de código, o diagrama UML de classes correspondente é:

UNIP

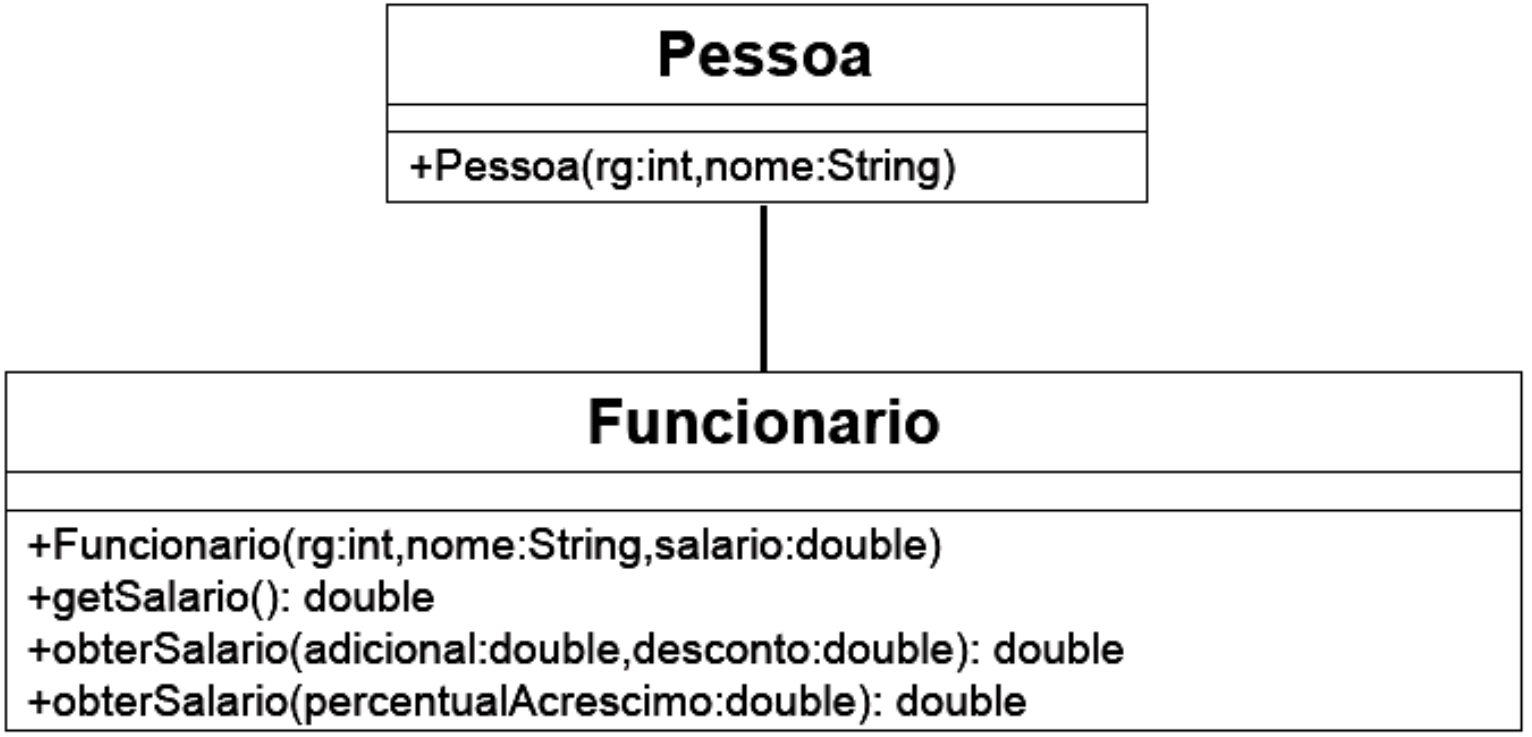
Questão 21 (ENADE, 2014)

a)



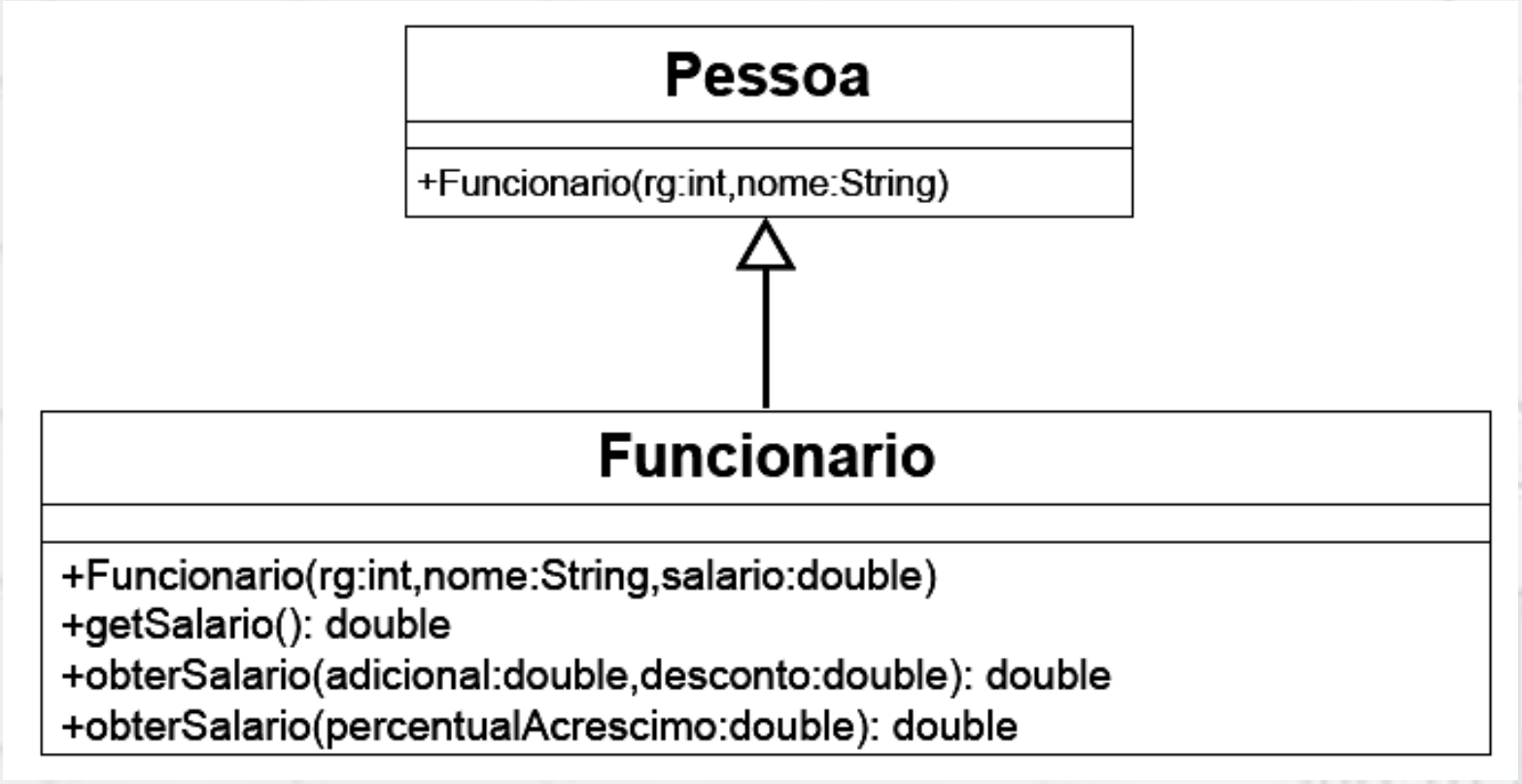
Questão 21 (ENADE, 2014)

b)



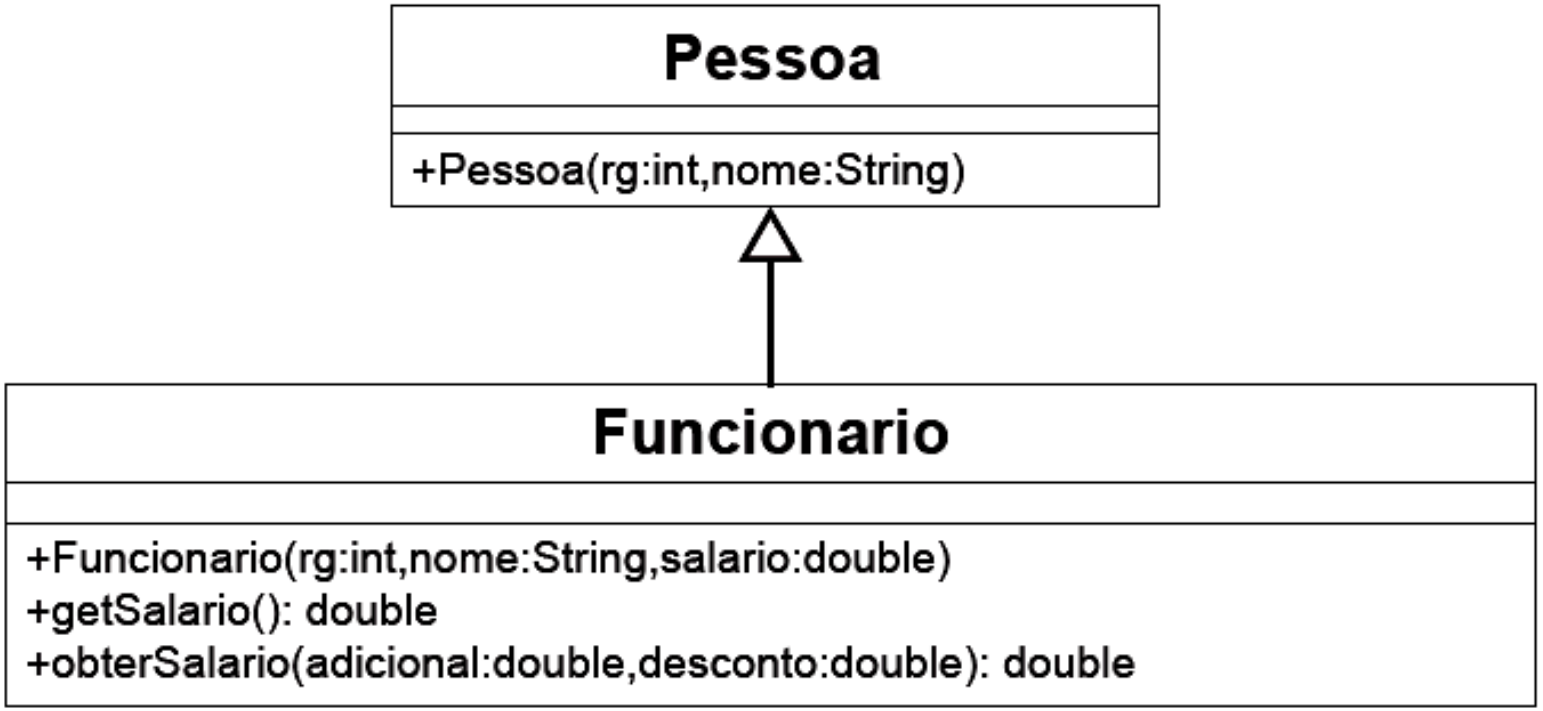
Questão 21 (ENADE, 2014)

c)



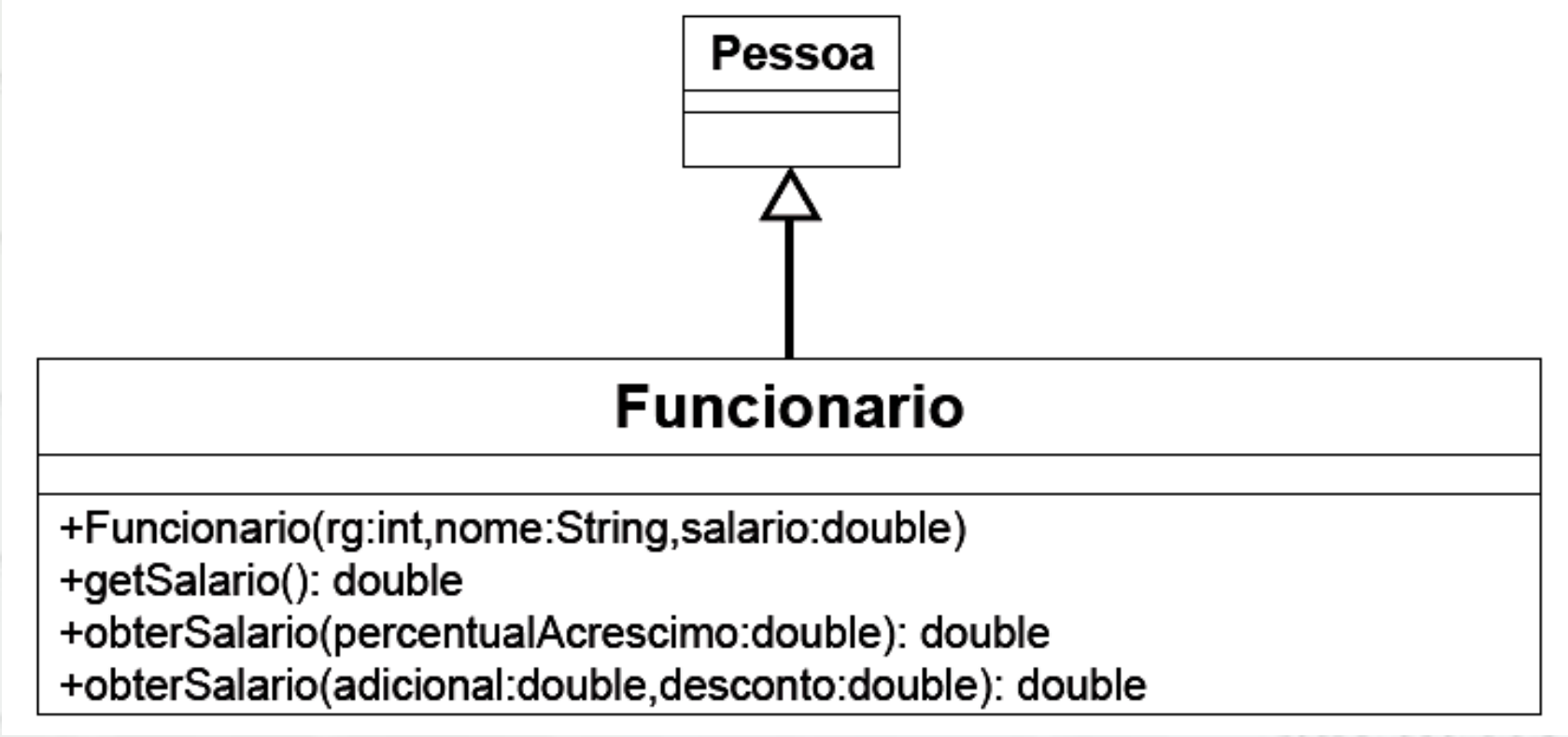
Questão 21 (ENADE, 2014)

d)



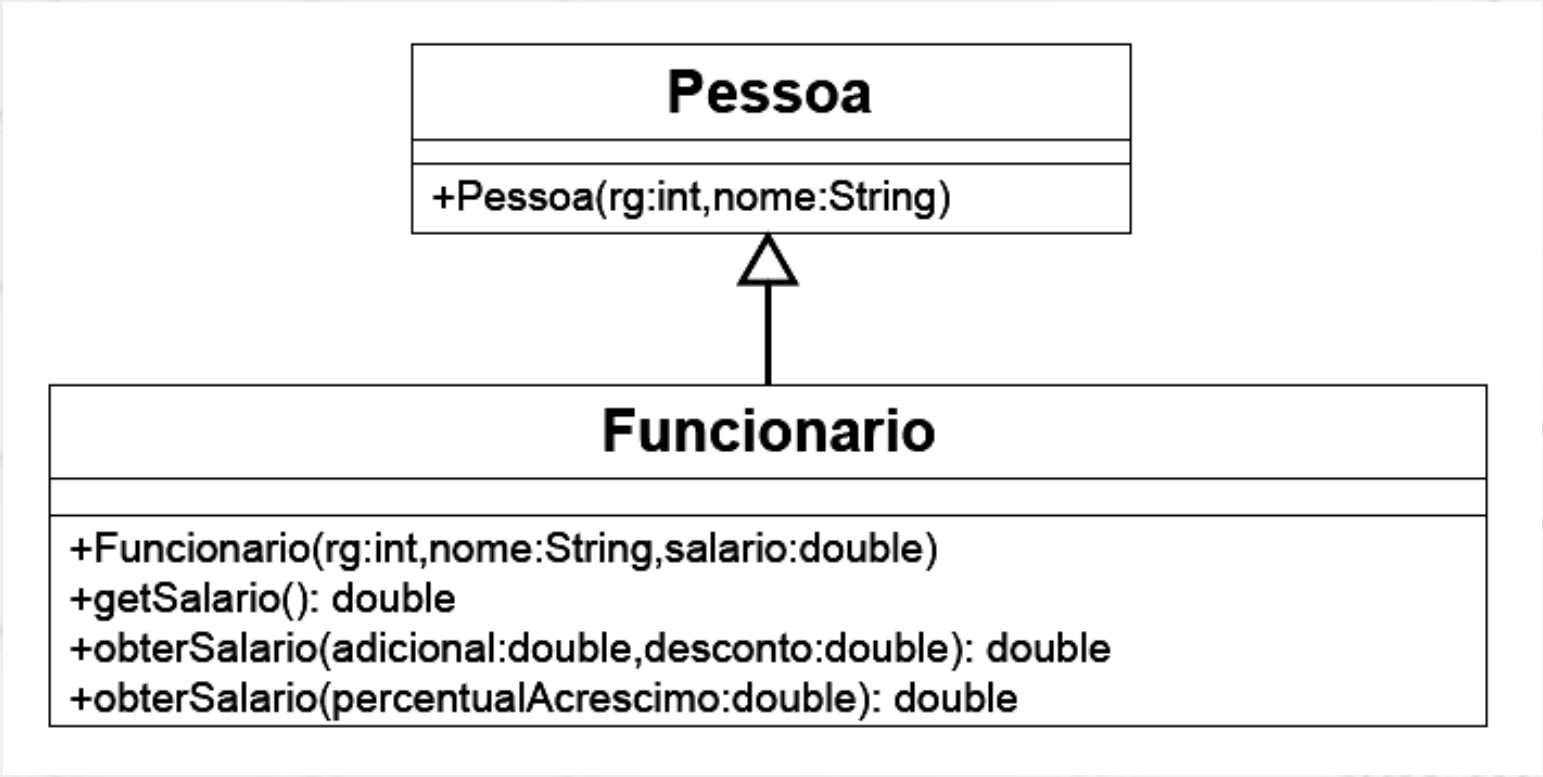
Questão 21 (ENADE, 2014)

e)



Questão 21 (ENADE, 2014)

a)



Comentários – Questão 21

a - Alternativa correta.

- **Justificativa. O trecho de diagrama de classes apresentado está correto. Nele, a classe Pessoa é uma generalização da classe Funcionário.**

b - Alternativa incorreta.

- **Justificativa. O diagrama está incorreto, pois as classes Pessoa e Funcionário estão representadas com relacionamento de associação. Na realidade, essas duas classes deveriam apresentar relacionamento de generalização.**

Comentários – Questão 21

c - Alternativa incorreta.

- **Justificativa. O construtor da classe Pessoa está representado de forma errada. Deveria ser: +Pessoa(rg:int, nome:String), e não +Funcionário(re:int, nome:String).**

d - Alternativa incorreta.

- **Justificativa. Falta um método na classe Funcionário.**

e - Alternativa incorreta.

- **Justificativa. A classe Pessoa do diagrama não apresenta o seu construtor.**

INTERVALO

Casos de Uso

- **Segundo Booch, Rumbaugh e Jacobson (2006),**
- **um caso de uso especifica o comportamento de um sistema ou de parte de um sistema e é uma descrição de um conjunto de sequências de ações, incluindo variantes realizadas pelo sistema para produzir um resultado observável do valor de um ator.**
- **Desenvolvedores de *software* precisam compreender as necessidades dos usuários e comunicá-las, sem precisar entrar em detalhes da implementação. A elaboração de casos de uso facilita esse processo de comunicação entre os diversos envolvidos em um projeto de *software*.**

Casos de Uso

- O modelo de casos de uso é um dos modelos da UML, composto do diagrama de casos de uso que apresenta um conjunto de casos de uso, atores e relacionamentos, e de um detalhamento do diagrama de casos de uso, geralmente uma descrição textual.
- O modelo de casos de uso é construído durante as discussões entre os desenvolvedores do sistema de *software* e os envolvidos (clientes, usuários, áreas da empresa envolvidas diretamente ou indiretamente no sistema) para especificar os requisitos.
- Um modelo de casos de uso apresenta os requisitos funcionais pretendidos do sistema e os elementos externos que interagem com o sistema.

UNIP

Casos de Uso

O objetivo principal, obtêm-se objetivos específicos que são:

- **Descrever os requisitos funcionais do sistema, de acordo, com todos os envolvidos e os desenvolvedores do sistema de *software*;**
- **Fornecer uma clara e consistente descrição do escopo do sistema e o que ele deve fazer;**
- **Fornecer requisitos para elaboração do diagrama de classes;**
- **Fornecer cenários para validação dos testes do sistema;**
- **Definir o perfil dos usuários que irão utilizar o sistema;**
- **Definir as condições de início, término e exceções de cada função do sistema.**

Casos de Uso

Caso de uso: Um caso de uso é um conjunto de ações que o sistema realiza para produzir um resultado observável para um ator.

Caso de Uso

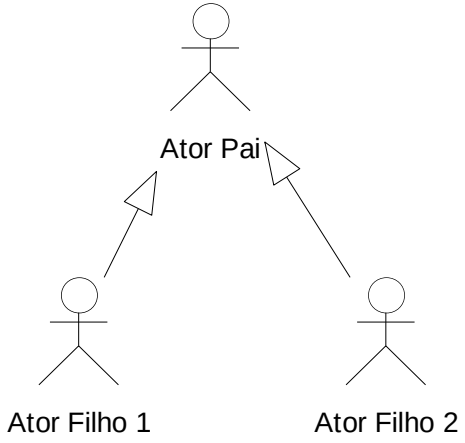
Ator: são elementos externos ao sistema, que interagem com os casos de uso. Consideram-se atores "alguém" ou "alguma coisa" que não faz parte do sistema, mas que interage com ele. Exemplos de atores: usuários diretos do sistema, sistemas externos que irão interagir com o sistema ou *hardware* específico.

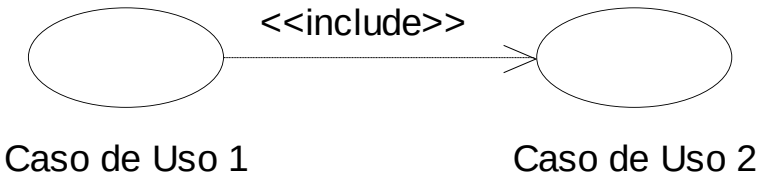
Ator

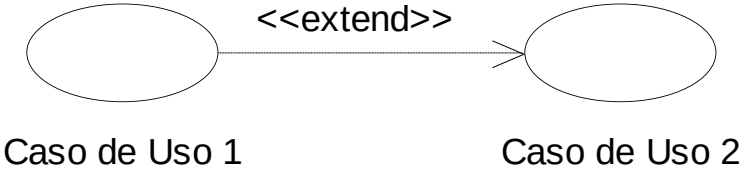
Associação

- **Associação é um relacionamento entre o ator e o caso de uso.**

Casos de Uso – extensões







Questão 22 (ENADE, 2014)

- **Casos de uso podem ser organizados agrupando-os em pacotes do mesmo modo como são organizadas as classes. Também podem ser organizados pela especificação de relacionamentos de generalização, inclusão e extensão, existentes entre eles.**

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. *UML - Guia do Usuário*. Campus, 2006 (com adaptações).

- **Considerando os relacionamentos existentes entre os casos de uso, avalie as afirmativas a seguir:**

Questão 22 (ENADE, 2014)

I. Para casos de uso, a generalização significa que o caso de uso filho herda o comportamento e o significado do caso de uso pai e no caso de uso filho deverá acrescentar ou sobrescrever o comportamento de seu pai.

II. Um relacionamento de inclusão entre casos de uso significa que o caso de uso base incorpora explicitamente o comportamento de outro caso de uso em uma localização especificada. O caso de uso base poderá permanecer isolado, mas, sob certas condições, seu comportamento poderá ser incluído pelo comportamento de outro caso de uso.

Questão 22 (ENADE, 2014)

III. Um relacionamento estendido entre casos de uso significa que o caso de uso base incorpora implicitamente o comportamento de outro caso de uso em um local especificado indiretamente pelo caso de uso estendido. O caso de uso estendido nunca permanece isolado, mas é apenas instanciado como parte de alguma base maior que o estende.

IV. Um relacionamento estendido é utilizado para a modelagem de parte de um caso de uso que o usuário poderá considerar como um comportamento opcional do sistema e para a modelagem de um sub-fluxo separado, que é executado somente sob determinadas condições.

Questão 22 (ENADE, 2014)

É correto apenas o que se afirma em:

a) I e II.

b) I e IV.

c) II e III.

d) I, III e IV.

e) II, III e IV.

Questão 22 – resposta

É correto apenas o que se afirma em:

a) I e II.

b) I e IV.

c) II e III.

d) I, III e IV.

e) II, III e IV.

Comentários – Questão 22

I - Afirmativa correta.

- **Justificativa. A generalização nos casos de uso tem papel similar à generalização nos diagramas de classes (BOOCH, RUMBAUGH e JACOBSON; 2006).**

II - Afirmativa incorreta.

- **Justificativa. Como é o caso de uso-base que incorpora o incluído, o relacionamento incluído não pode ser isolado. Além disso, esse relacionamento tem seu comportamento incluído (por isso o nome) e, também, não é o caso de uso-base.**

Comentários – Questão 22

III - Afirmativa incorreta.

- **Justificativa. A descrição não corresponde a um relacionamento de extensão, pois o caso de uso-base pode permanecer isolado.**

IV - Afirmativa correta.

- **Justificativa. O relacionamento de extensão é utilizado justamente para a modelagem de comportamentos opcionais, que podem ou não ocorrer em determinadas condições.**

ATÉ A PRÓXIMA!