Engenharia de Software

Exercícios Avançados

Fernando Pedrosa – fpedrosa@gmail.com

[1] FCC – 2008 – TCE-AL – Programador

A organização de áreas conceituais dentro de uma mesma camada é um princípio de design aplicado na arquitetura da UML denominada

a) modularity.
b) layering
c) partitioning.
d) extensibility.
e) reuse.

Arquitetura UML

- O metamodelo da UML foi projetado considerando os seguintes princípios:

- ***Modularity***
  - Blocos de construção são agrupados em pacotes seguindo princípios de alta coesão e baixo acoplamento
- ***Layering***
  - Níveis de abstração diferentes. Separação de blocos de construção "core" e blocos de construção de alto nível (user)
- ***Partitioning***
  - Organização de áreas conceituais dentro de uma mesma camada

Arquitetura UML

- ***Extensibility***
  - É possível estendera linguagem para plataformas específicas (JEE, .NET, etc...) usando Perfis
  - Ou, também é possível criar uma nova linguagem a partir da UML ampliando metaclasses e metarelacionamentos
- ***Reuse***
  - Uma biblioteca de “metamodelos” é disponibilizada em nível fino de granularidade, permitindo o reúso

[2] FEPESE – 2010 – UDESC – Analista de Sistemas

Considere o diagrama de máquina de estados de UML abaixo.

Analise as afirmativas a seguir, a seu respeito:

1. O elemento “estado1” corresponde a um estado sub-máquina.
2. O diagrama “diagrama estado1”, que refina “estado1”, terá exatamente um estado inicial.

[2] FEPESE – 2010 – UDESC – Analista de Sistemas

3. O diagrama “diagrama estado1”, que refina “estado1”, terá exatamente um estado final.
4. O diagrama “diagrama estado1”, que refina “estado1”, terá exatamente um pseudo-estado ponto de entrada e um pseudo-estado ponto de saída.

Assinale a alternativa que indica todas as afirmativas corretas.

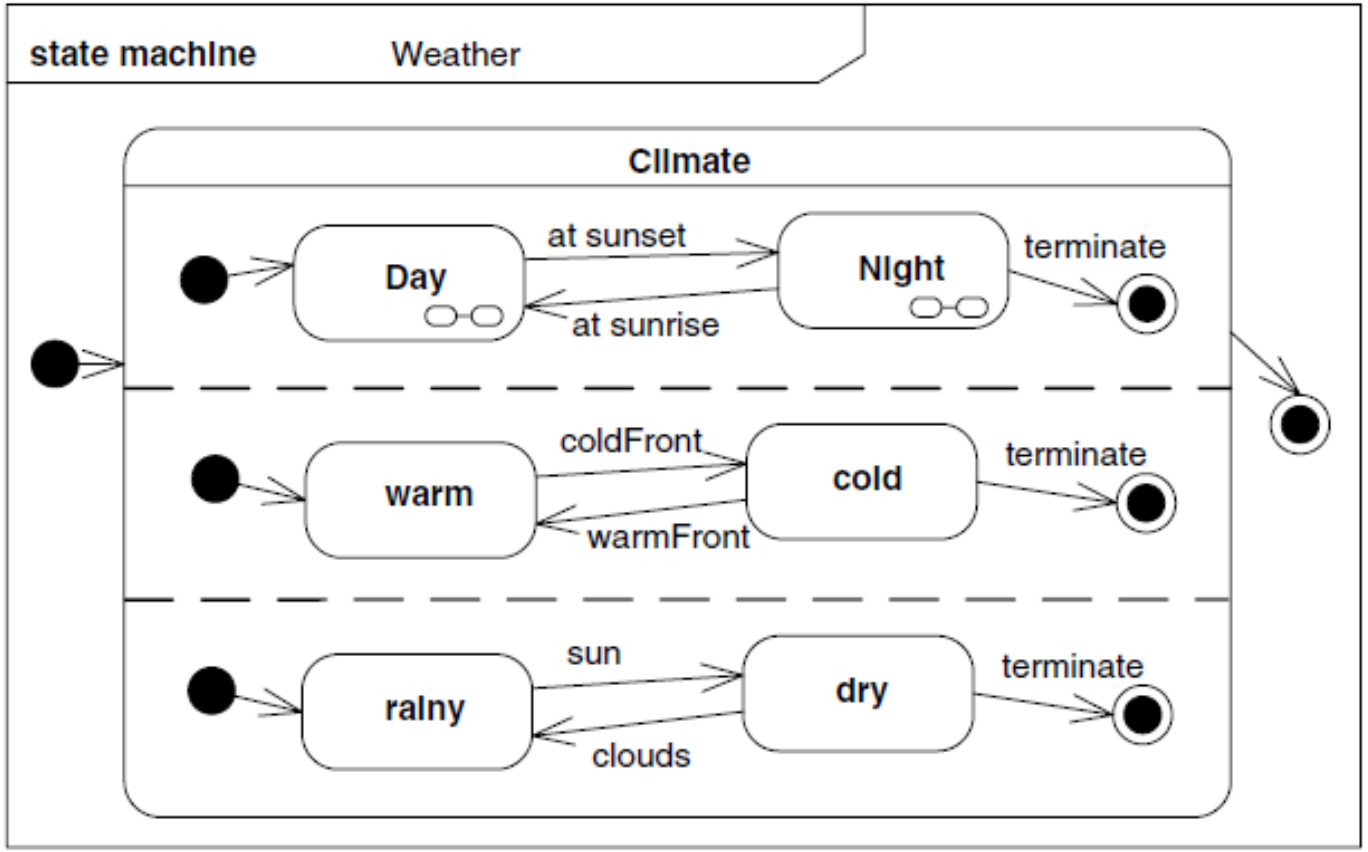
a) São corretas apenas as afirmativas 1 e 2.
b) São corretas apenas as afirmativas 2 e 3.
c) São corretas apenas as afirmativas 3 e 4.
d) São corretas apenas as afirmativas 1, 2 e 4.
e) São corretas apenas as afirmativas 1, 3 e 4.

Statemachine avançado

- 1. É possível ter estados de “sub-máquina” em diagramas de Máquina de Estado
  - *nomeDoEstado:nomeDaSubMáquinaReferenciada*
- 2. Só é possível haver, no máximo, um **pseudoestado inicial** por região em um diagrama de máquina de estados

Statemachine avançado

- Exemplo



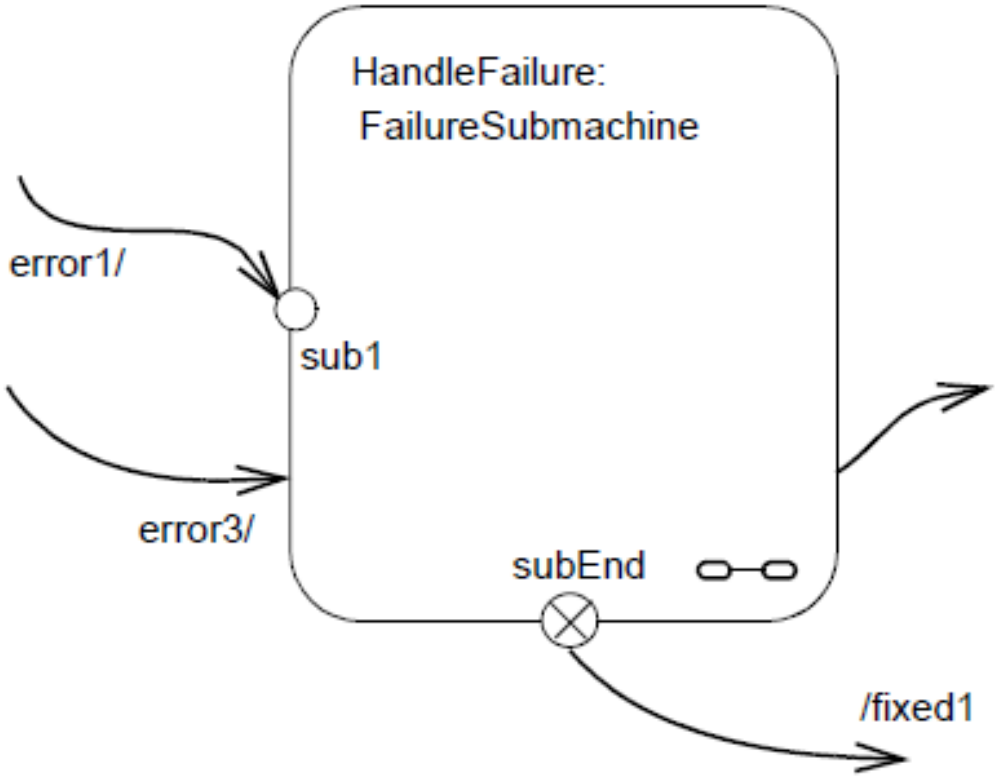
Statemachine avançado

- 3. É possível haver mais de um **estado final** por região, mas no diagrama em questão, mas a questão pressupôs que só havia um

- 4.1 **Pontos de Entrada** – usados para entrar na sub-máquina em um estado diferente do default (inicial)
- 4.2 **Pontos de Saída** – usados para sair da sub-máquina através de pontos diferentes do estado final

Statemachine avançado

- Exemplo



[3] CESPE – 2011 – EBC – Analista – Engenharia de Software

Tendo em vista que, na UML (Unified Modeling Language), a utilização de diversos diagramas possibilita descobrir falhas não detectadas em diagramas gerados anteriormente, diminuindo a possibilidade da ocorrência de erros durante a fase de desenvolvimento do software, julgue os itens a seguir.

O diagrama de comunicação concentra-se em como os objetos estão vinculados e quais mensagens trocam entre si durante o processo. Esse diagrama está associado ao diagrama de classe

Diagrama de Comunicação

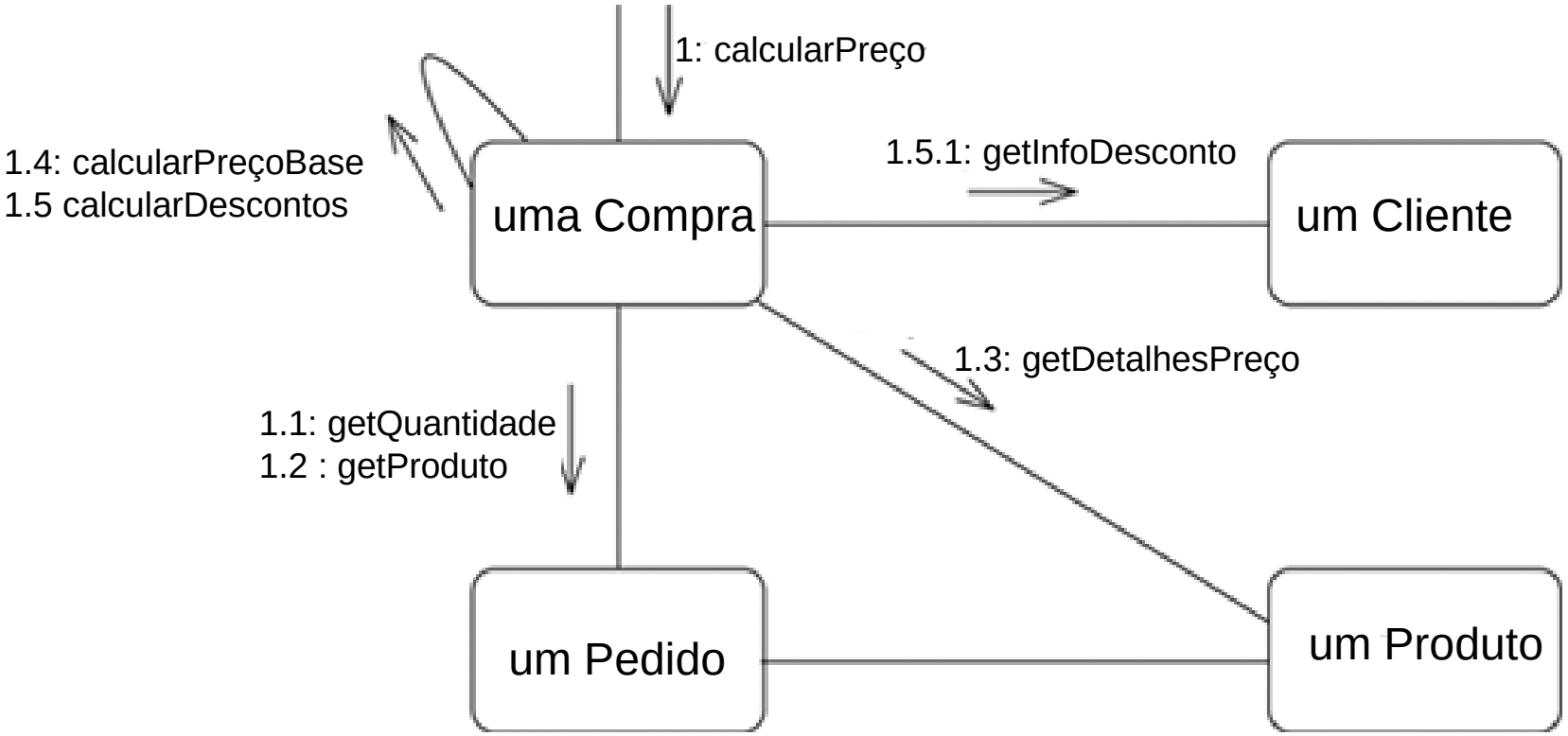


Diagrama de Comunicação

- *Sequence diagrams and communication diagrams are semantically equivalent. As a result, you can take a diagram in one form and convert it to the other without any loss of information.*
  - UML User Guide, 2nd ed, capítulo 19.

[4] . FCC – 2011 – TRE-AP – Analista Jud- Análise de Sistemas

A questão abaixo refere-se à UML 2.0.

Descreve o comportamento de classificadores individuais e de interações de classificadores, concentrando a atenção no momento de ocorrência de eventos que causam mudanças nas condições modeladas das linhas de vida. Trata-se do diagrama de

a) implantação.
b) máquina de estado.
c) evento
d) sequência.
e) tempo.

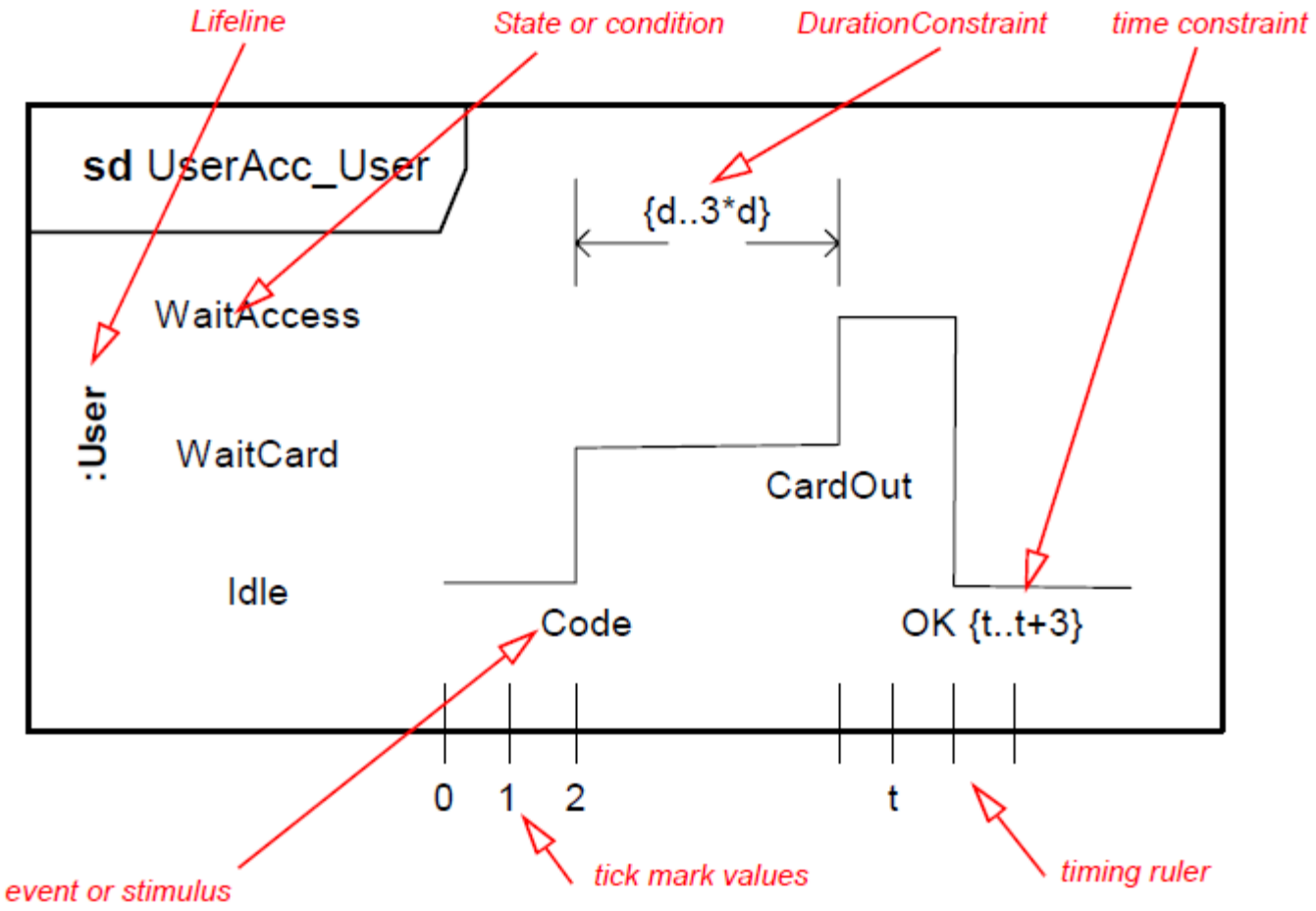
Diagramas de Interação

- Diagrama de Sequência
  - Enfatiza a sequência (ordem temporal) na qual as mensagens são trocadas
- Diagrama de Comunicação
  - Enfatiza os vínculos entre os objetos participantes
- Diagrama de Interação Geral
  - Apresenta a ordem das interações, organizadas em notação de Atividade
- Diagrama de Tempo
  - Enfatiza as mudanças de estados dos objetos participantes em relação ao tempo e as mensagens trocadas

Diagrama de Tempo

- *Timing diagrams describe behavior of both individual classifiers and interactions of classifiers, focusing attention on time of occurrence of events causing changes in the modeled conditions of the Lifelines.*
  - UML Superstructure, 2.4, página 536

Diagrama de Tempo



[5] . VUNESP – 2009 – CETESB – Analista de TI – Banco de Dados

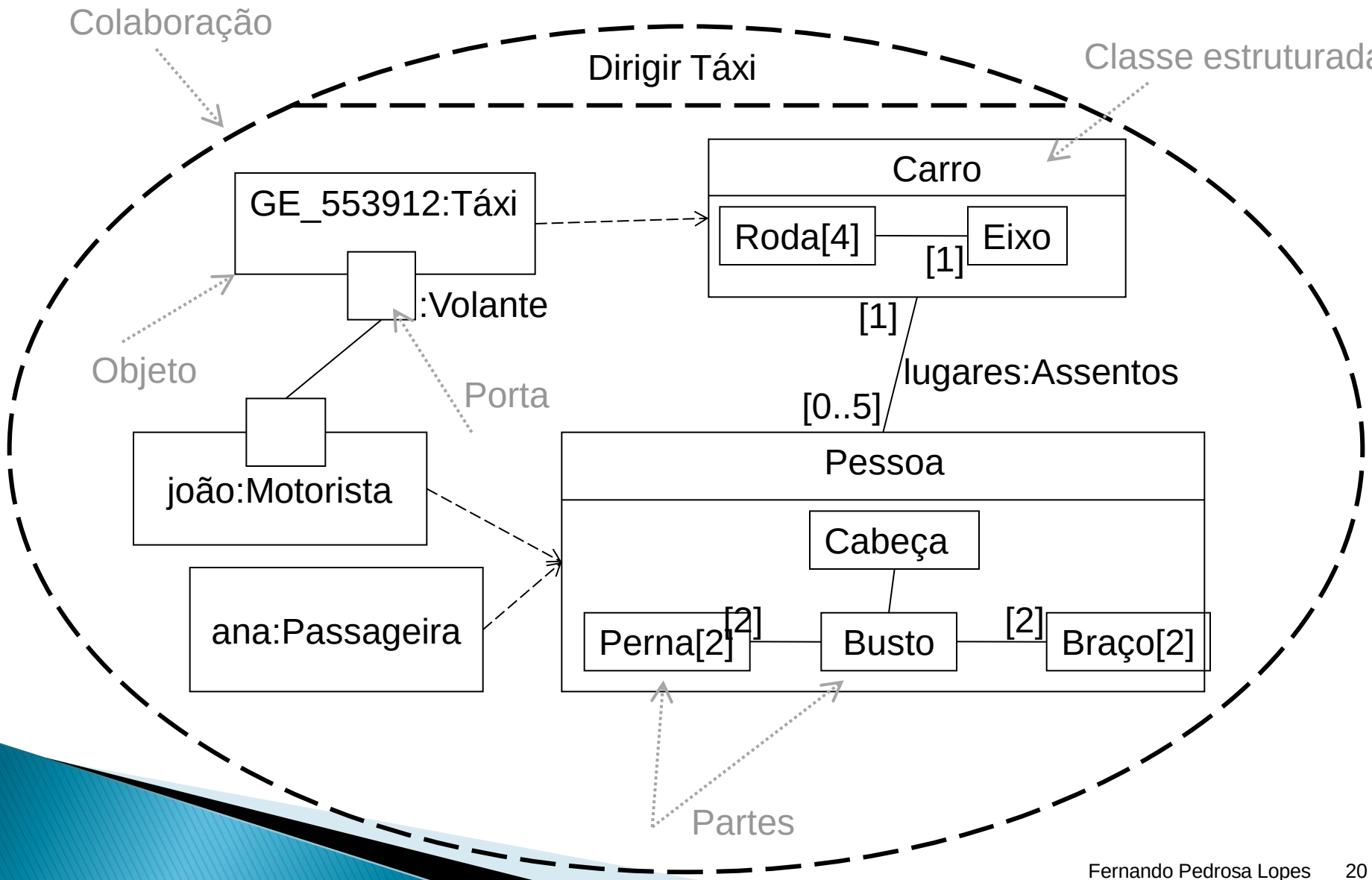
Na UML, quando um processo é descrito por meio do consumo direto de objetos, e estes, por meio da colaboração entre eles, o seu comportamento deve ser descrito utilizando o Diagrama de

a) Caso de Uso.
b) Classe.
c) Colaboração.
d) Estrutura Composta.
e) Objeto.

Diagrama de Estrutura Composta

- Representa a estrutura de composições, ou seja, mostra os relacionamentos das "partes do todo"
  - Além disso mostra como estas partes podem **colaborar** para realizar uma funcionalidade, comportamento ou processo em uma situação específica
- No mesmo diagrama podem ser mostrados Objetos, Classes e Interfaces

Diagrama de Estrutura Composta



[6] . FCC – 2009 – PGE-RJ – Técnico de Análise de Sistemas e Métodos

Na UML, uma porta

a) é o mesmo que interface de pacote.

b) não pode ter múltiplas instâncias em uma única instância de componente.

c) é uma janela explícita em um componente encapsulado.

d) é a coleção de operações utilizadas na especificação de serviços de um componente.

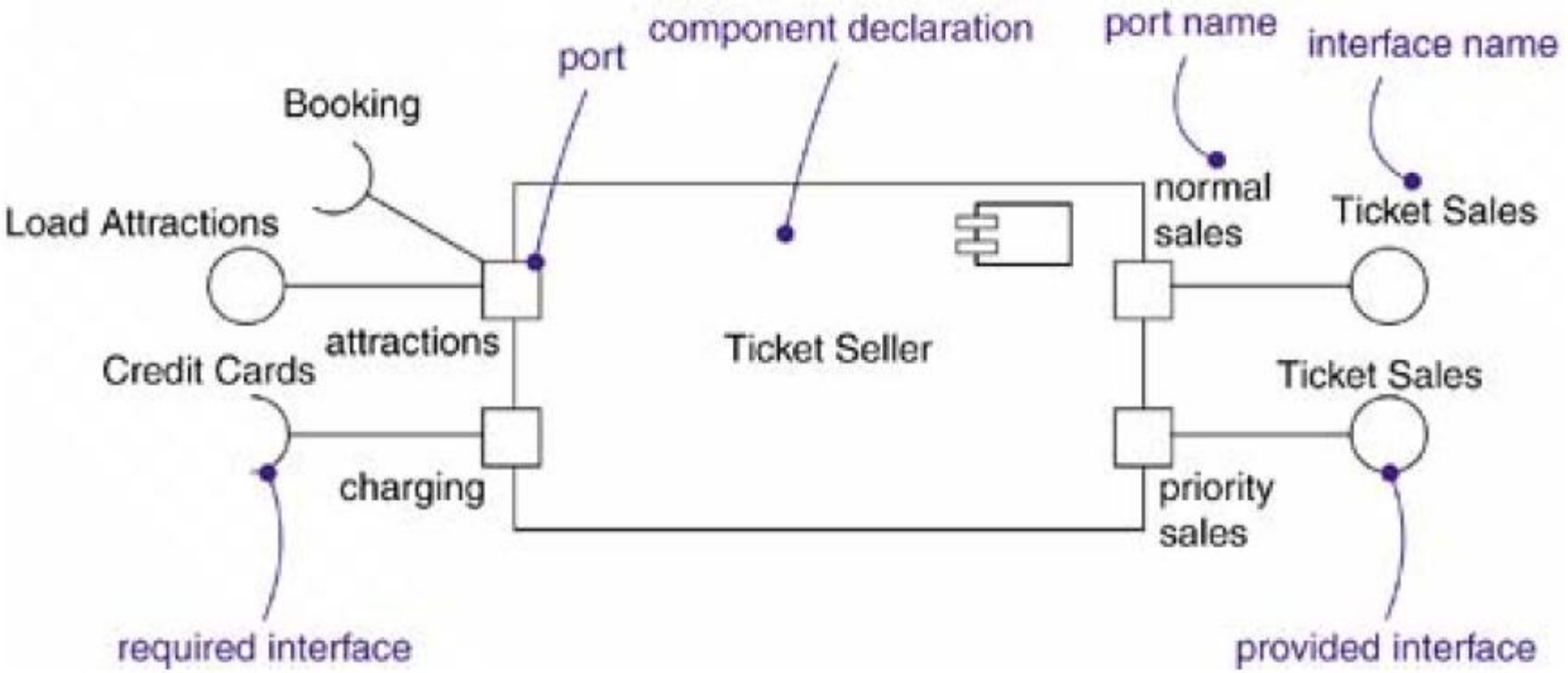
e) é o mesmo que interface de componente.

Porta

- Propriedade especial de um classificador que especifica um ponto único de interação entre o classificador e o ambiente ao seu redor ou entre o classificador e suas partes internas
  - *Referência: UML Superstructure 2.4, pág 185*

- **A port is a specific window into an encapsulated component accepting messages to and from the component conforming to specified interfaces.**
  - *Referência: UML User Guide, cap 15*

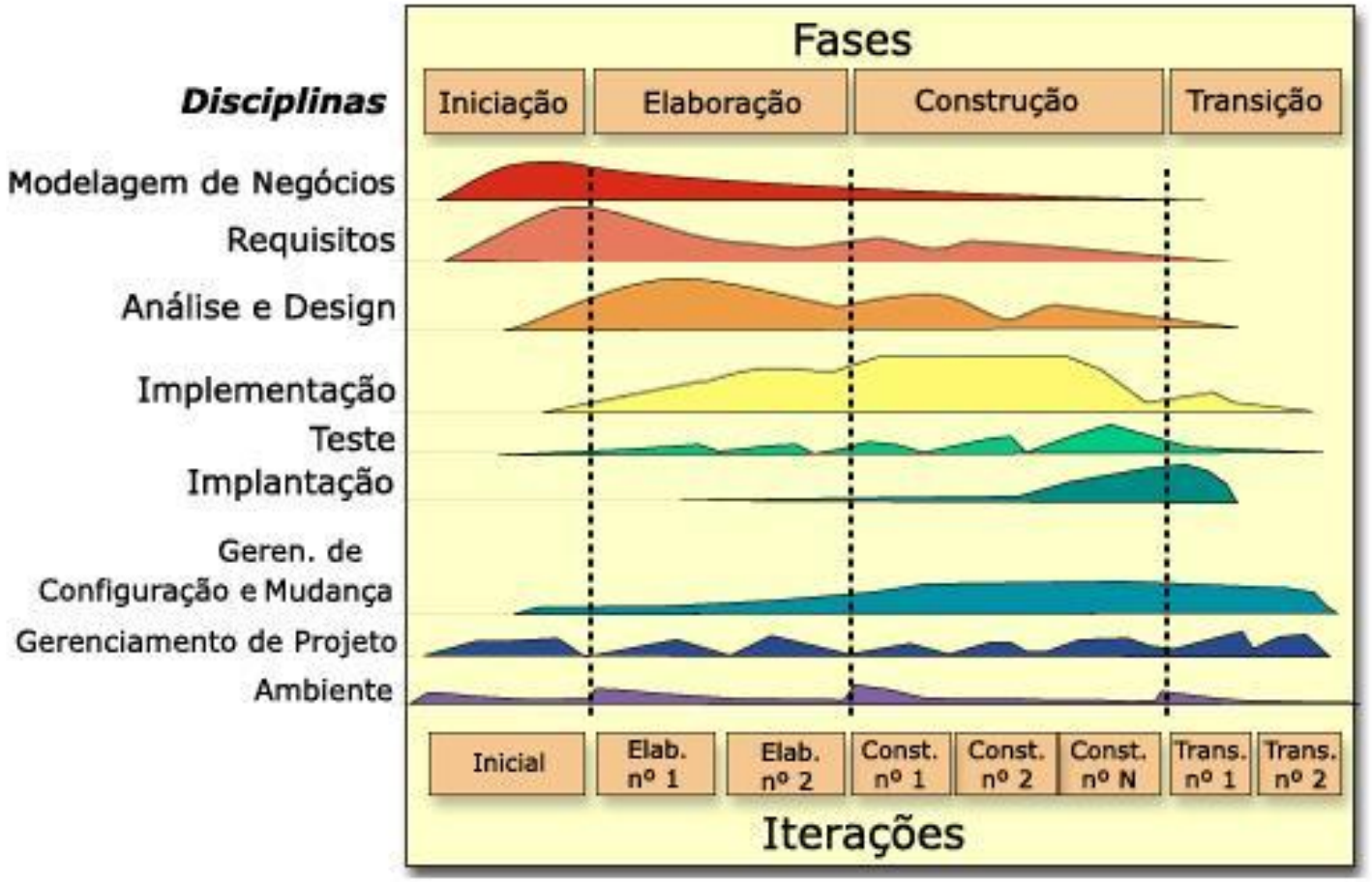
Porta


component declaration
port
port name
interface name
Booking
normal
sales
Ticket Sales
Load Attractions
attractions
Ticket Seller
Credit Cards
Ticket Sales
charging
priority
sales
required interface
provided interface

[7] . CESPE – 2010 – EMBASA – Analista de TI – Desenvolvimento

Programação e testes são atividades que acontecem na fase de concepção do processo unificado, pois a realimentação e os testes precoces servem para evoluir os requisitos.

Gráfico das Baleias (RUP)



[8] . FCC – 2009 – TJ-SE – Análise de Sistemas – Desenvolvimento

A Organization along content do RUP está estruturada em

a) Core Process Workflows e Phases, somente.

b) Phases e Iterations, somente.

c) Core Process Workflows, Core Supporting Workflows, Phases e Iterations.

d) Core Process Workflows, Core Supporting Workflows e Phases, somente.

e) Core Process Workflows e Core Supporting Workflows, somente.

Referência

- Rational Unified Process Best Practices for Software Development Teams
  - A Rational Software Corporation White Paper

[9] . FCC – 2011 – TRT – 23ª REGIÃO – Técnico Judiciário – TI

No processo de desenvolvimento de software, é a atividade que refere-se à garantia de que o sistema de software irá ao encontro de requisitos do produto, como também assegurar que futuros requisitos possam ser atendidos:

a) Especificação.
b) Arquitetura de Software.
c) Análise de Requisitos.
d) Implementação.
e) Suporte e Treinamento.

Fases do ciclo de vida do Software

- **Planejamento**
  - Esboçar escopo e requisitos
  - Fazer estimativas razoáveis sobre recursos, custos e prazos

- **Análise e Especificação de Requisitos**
  - Refinar requisitos e escopo
  - Entender o domínio do problema, com comportamento e funcionalidades esperados

Fases do ciclo de vida do Software

- **Projeto**
  - Incorporar requisitos tecnológicos aos requisitos essenciais do sistema
  - Projetar a arquitetura do sistema

- **Implementação**
  - Traduzir o projeto em uma forma passível de execução pela máquina
  - Codificação

Fases do ciclo de vida do Software

- **Testes**
  - Realizar diversos níveis de teste, de forma a fazer a verificação do software.

- **Implantação, Operação e Manutenção**
  - Colocar o software em produção
  - Treinar pessoas
  - Manter o software
  - Gerenciar os serviços

[10] . CESGRANRIO – 2010 – ELETROBRÁS – Analista de Sistemas

Uma fábrica de software recomenda que a documentação de especificação funcional de um sistema deve ser clara para o cliente e detalhada para o desenvolvedor, estabelecendo um contrato entre eles. Documentos de especificação funcional têm como característica

a) apresentar os diagramas de interação relacionados aos requisitos contidos na especificação.

b) conter os requisitos não funcionais pertinentes ao problema a ser resolvido.

c) conter instruções detalhadas sobre o que o sistema deve fazer e como ele deve ser implementado.

d) ser descrito em uma linguagem matemática formal, facilitando o entendimento do cliente que, geralmente, não é um especialista técnico.

e) definir os recursos responsáveis por implementar as funcionalidades.

[11] . FGV – 2010 – BADESC – Desenvolvimento de Sistemas

Segundo Yourdon, o ciclo de vida de um projeto de sistema é o modo como o projeto é desenvolvido na empresa e uma maneira simples para que qualquer pessoa da área de desenvolvimento de sistemas possa se entrosar com o projeto a ser desenvolvido.

O ciclo de vida de um projeto de sistema é importante pelas razões apresentadas nas alternativas a seguir, à exceção de uma.

Assinale-a.

a) Facilitam o gerenciamento de prazos.
b) Definem as atividades a serem executadas em um projeto.
c) Introduzem pontos de verificação para o controle gerencial de decisões.

[11] . FGV – 2010 – BADESC – Desenvolvimento de Sistemas

d) Mostram a consistência entre muitos projetos de desenvolvimento da organização.

e) Contemplam as ações sequenciais de um projeto de desenvolvimento, do diagnóstico do problema aos testes necessários à sua implementação.

[12] . FCC – 2010 – MPE-RN – Analista de TI – Eng. de Software

Um protótipo de software pode ser usado em um processo de desenvolvimento de software para:

I. Ajudar na descoberta e validação dos requisitos do sistema, durante a engenharia de requisitos.

II. Explorar soluções específicas de software e apoiar o projeto de interface com o usuário, durante o projeto do sistema.

III. Realizar testes completos com o sistema que será entregue para o cliente, durante o processo de teste.

Está correto o que se afirma em

a) I, II e III.

b) II e III, apenas.

c) I e II, apenas. d) II, apenas. e) I, apenas.

Back-to-back testing

- Apresenta o mesmo teste para diferentes versões do sistema e compara os resultados. Resultados diferentes implicam possíveis problemas
- Reduz o custo de examinar resultados de testes (comparação automática)
- Possível quando um protótipo está disponível ou através de testes de regressão com diferentes versões do sistema

[13] . ESAF – 2010 – CVM – Analista de Sistemas – prova 2

Assinale a opção correta.

a) Teste de Valor Referenciado tem por objetivo montar um conjunto de casos de teste que representem tabelas de ações.

b) Blockrwise testing tem por objetivo obter um conjunto de casos de testes significativo dentro de um bloco de citações permutáveis.

c) Parserwork testing tem por objetivo obter um conjunto de casos de parsers significativo dentro de um conjunto imutável de fluxos de citações.

d) Pairwise testing tem por objetivo obter um conjunto de casos de testes significativo dentro de um conjunto imutável de combinações de citações.

e) Teste por Relações tem por objetivo obter um conjunto de casos de testes significativo dentro de um conjunto ponderável de combinações de citações.

Pairwise Testing

- Algumas falhas só acontecem quando há uma combinação de N fatores
- Mas é impossível gerar casos de testes para todas as combinações possíveis
- Pairwise Testing (All-Pairs Testing) concentra os esforços em testar
  - Cada estado de cada variável (single-mode faults)
  - Cada variável, em cada um dos seus estados, pareada com cada outra variável em cada um dos seus estados (double-mode faults)
- Reduz o escopo de casos de teste

[14] . ESAF – 2008 – Pref. de Natal – Auditor do Tesouro Municipal – TI

Com relação aos tipos de testes que podem ser considerados e executados em um projeto de software, é correto afirmar que o objetivo principal do Teste Funcional é assegurar que

a) a interface forneça ao usuário o acesso e a navegação adequados através das funções do software.

b) os objetos contidos na interface funcionam conforme o esperado e estejam em conformidade com padrões estabelecidos.

c) foram exercitados um conjunto de funcionalidades para avaliar a navegação pelo software e a facilidade de uso do mesmo.

d) houve uma correta implementação dos requisitos do sistema, como, por exemplo, regras de negócio, através da interface do usuário.

e) foram executadas transações, variando as cargas de trabalho, para observar e registrar o comportamento do sistema.

Abordagem Caixa Preta

- Baseada em prés e pós condições
- Geralmente utilizada nas etapas posteriores da disciplina de testes
- Busca erros nas seguintes categorias (dentre outras):
  - Funções incorretas ou inexistentes
  - Erros de comportamento ou desempenho
  - Erros de inicialização e término, erros de interface
- Principais técnicas: testes baseados em grafos, matriz ortogonal, análise de valores limítrofes, particionamento de equivalências

[15] . FGV – 2008 – Senado Federal – Analista de Sistemas

Considere as seguintes assertivas sobre o teste de sistema:

I. Teste de mutação é um critério de teste da técnica baseada em defeitos.

II. O tempo médio para falhas (MTTF) pode ser utilizado para medir a confiabilidade do sistema; quanto mais próximo do zero o MTTF, maior a confiabilidade do sistema.

III. No teste funcional não são considerados os aspectos de implementação do software e por isso a técnica é também chamada de caixa-preta.

As assertivas corretas são:

a) somente III. b) somente I e II. c) somente I e III.

d) somente II e III. e) todas.

Mutation Testing

- Avalia a efetividade de um conjunto de testes
- Envolve modificar o código fonte ou *byte code* de um programa sutilmente
  - Essas “mutações” normalmente simulam pequenos erros de programação
- Caso o comportamento do programa continue o mesmo depois da mutação, isso significa que o código não está sendo apropriadamente testado

[16] . FEPESE – 2010 – UDESC – Analista de Sistemas

Considere as seguintes afirmativas relacionadas a métricas de software:

1. A contagem de linhas de código (LOC) constitui um exemplo de métrica direta.
2. A medida de qualidade expressa em erros/KLOC constitui um exemplo de métrica orientada a tamanho (KLOC = 1000.LOC).
3. A medida de qualidade expressa em erros/KLOC constitui um exemplo de métrica indireta (KLOC = 1000.LOC).

Assinale a alternativa que indica todas as afirmativas corretas.

a) É correta apenas a afirmativa 1.

b) É correta apenas a afirmativa 3.

c) São corretas apenas as afirmativas 1 e 2.

d) São corretas apenas as afirmativas 2 e 3.

e) São corretas as afirmativas 1, 2 e 3

Métricas – Categorização

- **Métricas diretas (fundamentais ou básicas)**
  - Medida realizada em termos de atributos objetivamente observáveis
  - Ex.: custo, esforço, número de linhas de código, capacidade de memória, número de páginas (documentação), etc.
- **Métricas indiretas (derivadas)**
  - Medidas obtidas a partir de outras métricas
  - Ex.: complexidade, eficiência, confiabilidade, facilidade de manutenção

Métricas – Categorização

- **Métricas orientadas a tamanho**
  - São medidas derivadas através de atributos associados ao tamanho do software produzido ou processo utilizado
  - Ex.: Erros/KLOC, Dólares/KLOC, Páginas/KLOC, Erros/Pessoas-Mês, KLOC/Pessoas-Mês, Dólares/Páginas de documentação, etc.
- **Métricas orientadas por função**
  - Consiste em um método para medição de software do ponto de vista do usuário, determinando de forma consistente o tamanho e a complexidade de um software.

Métricas – Categorização

- **Métricas de produtividade**
  - Concentram-se na saída do processo de engenharia de software.
  - Ex.: Casos de uso/iteração, LOC/Hora, etc.
- **Métricas de qualidade**
  - Oferecem uma indicação de quanto o software se adequa às exigências implícitas e explícitas do cliente.
  - Ex.: Erros/PF, Erros/Horas de revisão, Erros/Testes, etc.

[17] . UFF – 2009 – UFF – Analista de Tecnologia da Informação

No tocante às métricas de projeto, são objetivos dessas métricas:

a) minimizar o cronograma e avaliar a qualidade do produto;
b) indicar lucro e minimizar perdas;
c) avaliar a qualidade do produto e padronizar o projeto;
d) padronizar o projeto e maximizar o lucro;
e) minimizar intervenções do cliente e apontar padrões utilizados.

[18] . CESGRANRIO – 2010 – Petrobrás – Processos de Negócios

Durante a fase de planejamento de um projeto de software de uma empresa, as seguintes considerações foram feitas:

I – ao estimar o tamanho do projeto, optou-se por usar contagem por linhas de código (LOC), pois é um método mais fácil de se trabalhar na fase inicial do projeto, é de imediata aplicação a linguagens não procedimentais e fácil de ser calculada;

II – caso a opção fosse por estimativa por pontos de função, deveriam ser levados em conta os tipos de transações, tais como arquivos lógicos internos e entradas externas e os tipos de dados, como consultas internas e arquivos de interface, a fim de se fazer o correto cálculo de estimativas;

[18] . CESGRANRIO – 2010 – Petrobrás – Processos de Negócios

III – a identificação de riscos pode ser obtida a partir de reuniões formais com a equipe de desenvolvimento, as quais vão servir para verificar e separar os problemas, e através da participação anônima de pessoas envolvidas no projeto informando problemas futuros e atuais;

IV – foi importante, no planejamento, identificar os itens que servirão para atestar a conformidade do produto gerado de acordo com os padrões estabelecidos, sem a qual não seria possível ter-se a medida de qualidade do mesmo, sendo que esta identificação de qualidade foi planejada para funcionar em paralelo com outros processos do projeto.

Está(ão) correta(s) APENAS a(s) consideração(ões)

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

[19] . FUNCAB – 2010 – PRODAM– Desenvolvimento de Sistemas

Use uma das cinco alternativas a seguir para completar a lacuna do texto abaixo de modo que o texto resultante seja uma assertiva verdadeira.

O uso de _____ era uma prática muito comum durante o design orientado a objetos antes da adoção de linguagens de modelagem como a UML.

a) Diagramas de fluxos de dados.
b) Diagramas de hierarquia modular.
c) Fluxogramas.
d) Cartões CRC.
e) Peudocódigo.

Cartão CRC

- É uma ferramenta de modelagem utilizada para o projeto orientado a objetos
- Tipicamente utilizadas quando se está determinando que classes serão necessárias no sistema e como elas vão interagir
- Os cartões normalmente possuem:
  - O nome da classe
  - Os seus relacionamentos de herança (caso aplicável)
  - As responsabilidades da classe
  - O nome de outras classes com a qual ela colabora

Cartão CRC

TApplication	Flip
Superclasses: TEventHandler	
Subclasses:	
Responsibilities:	Collaborators:
EventLoop	TWindow
DoMenuCommand	TDocument, TWindo
SetupMenus	TWindow
DoMouseCommand	TWindow

TList	Flip
Superclasses: TObject	
Subclasses:	
Responsibilities:	Collaborators:
Initialize	
Insert	
Delete	
EachItemDo	
Free	

TBox	Flip
Superclasses: TShape	
Subclasses:	
Responsibilities:	Collaborators:
Initialize	
Read	
Write	
Draw	

TObject	Flip
Superclasses:	
Subclasses: TEventHandler, TList, TShape	
Responsibilities:	Collaborators:
Free	

[20] . ESAF – 2008 – Prefeitura de Natal Auditor do Tesouro – TI

Analise as seguintes afirmações relacionadas a Orientação a Objetos:

I. Quando os atributos são de instância, cada objeto tem sua própria cópia desses atributos.

II. Quando um código precisa ser executado para a inicialização de um objeto, esse referido código pode ser especificado por meio de um método construtor.

III. Um mesmo nome de método pode ser usado para identificar diferentes métodos em uma mesma classe ou diferentes métodos em classes diferentes. A isto se dá o nome de Herança.

IV. Existem métodos cujo código apenas acessa atributos de classe ou os parâmetros passados. Esses métodos são chamados "Métodos Abstratos".

Indique a opção que contenha todas as afirmações verdadeiras.

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

[21] . FUNCAB – 2010 – PRODAM-AM – Analista de TI

Sejam as seguintes assertivas sobre a arquitetura em camadas:

I. Promove a abstração de design, permitindo que um problema complexo seja decomposto em várias camadas de funções.

II. Facilita a alteração das funções internas – que não são visíveis ao mundo externo –, pois apenas os objetos da camada de interface são visíveis ao mundo externo.

III. Uma camada pode ser reutilizada se a interface for compatível. Uma camada autocontida pode também ser usada como um componente à parte.

IV. Pode degradar o desempenho de um sistema, pois as funções da interface muitas vezes precisam atravessar várias camadas para invocar uma operação em uma camada mais interna.

V. A padronização das camadas de interface pode levar a chamadas de funções confusas e ineficientes.

[21] . FUNCAB – 2010 – PRODAM-AM – Analista de TI

Dentre as assertivas acima, quantas são verdadeiras?

a) 1
b) 2
c) 3
d) 4
e) 5

Arquitetura em Camadas

- Uma forma de organizar a arquitetura é através de camadas de software
  - Cada camada provê um conjunto de funcionalidades em determinado nível de abstração
  - Tipicamente, uma camada de mais alta abstração depende de uma camada de mais baixa abstração, e não o contrário
  - Uma mudança em determinada camada, desde que seja mantida sua interface, não afeta as outras camadas

Arquitetura em Camadas

Vantagens

- Separação de responsabilidades
- Decomposição de complexidade
- Encapsulamento de implementação
- Maior reuso e extensibilidade

Desvantagens

- Podem penalizar a performance do sistema
- Aumento do esforço e complexidade de desenvolvimento

[22] . CESPE – 2011 – EBC – Analista – Administração de Sistemas

O mecanismo de iteração pode ser utilizado para sequenciar comandos, controlando a execução do programa.

Estruturas básicas de programação

- **Sequência**
  - Comandos em sequência
- **Repetição**
  - Loops
  - for, while, do-while...
- **Seleção**
  - Condicionais
  - if, if-else, switch

[23] . FCC – 2007 – MPU – Desenvolvimento de Sistemas

Especificar um valor a ser armazenado na área de memória referenciada por um nome descritivo caracteriza uma

a) variável.
b) instrução de controle.
c) sintaxe de comentário.
d) constante.
e) instrução de atribuição.

[24] . VUNESP – 2009 – CETESB – Analista de TI – Banco de Dados

Na análise essencial, existem dois modelos para a abstração do negócio a ser documentado. Esses modelos são chamados de Modelo

a) Analítico e Modelo Estruturado.
b) Comportamental e Modelo Essencial.
c) Essencial e Modelo de Implementação.
d) de Ambiente e Modelo Comportamental.
e) de Implementação e Modelo Analítico.

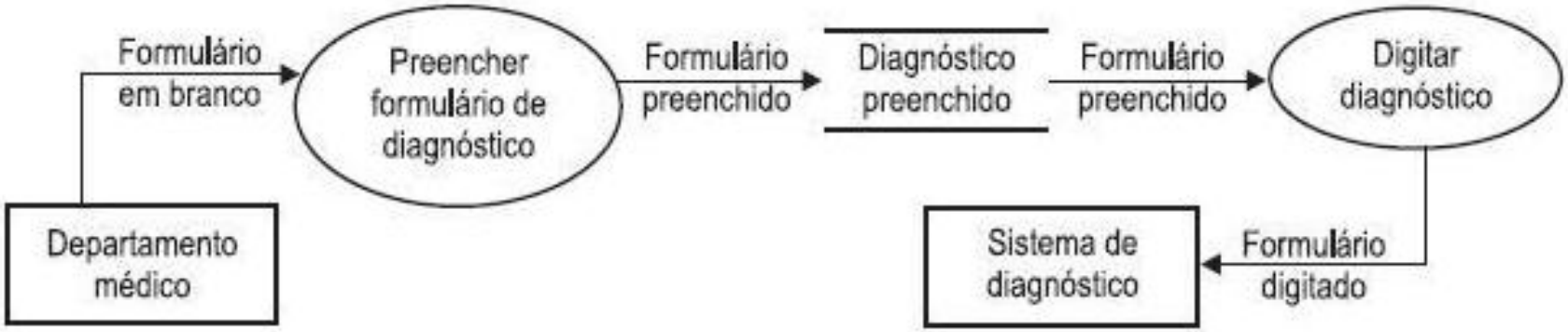
Modelo Essencial

- Modelo do QUE o sistema precisa fazer para satisfazer os seus requisitos
- Idealmente nada deve ser dito sobre como o sistema será implementado
- Sem restrições de tecnologia ou custo
- Composto pelos modelos:
  - Ambiental
  - Comportamental

Modelo de Implementação

- Modelo que reflete a estrutura e comportamento do sistema atual
- Tem como objetivo obter um entendimento e visão **gerais** dos processos existentes
  - Não é necessário documentar cada detalhe
- Utilizado quando existem dúvidas e são necessários esclarecimentos para modelar um novo sistema

[25] . CESGRANRIO - 2010 - ELETROBRÁS - Analista de Sistemas



[25] . CESGRANRIO – 2010 – ELETROBRÁS – Analista de Sistemas

O DFD acima representa um trecho da modelagem de um cadastro de diagnóstico médico. Com base nas informações apresentadas no diagrama, afirma-se que o(a)

a) elemento "Diagnóstico preenchido" realiza validações a respeito das informações fornecidas pelo Departamento Médico.

b) elemento "Diagnóstico preenchido" define um depósito de dados, relacionado a um meio físico, como um fichário ou um arquivo magnético, por exemplo.

c) elemento "Diagnóstico preenchido" representa dados em movimento.

d) função "Digitar diagnóstico" é estimulada por um evento de controle interno ao sistema.

e) função "Digitar diagnóstico" não precisa ser executada imediatamente após a função "Preencher formulário de diagnóstico".

Diagrama de Fluxo de Dados (DFD)

- Ferramenta de modelagem que permite visualizar o sistema como uma rede de processos funcionais
- Composto de:
  - Processos
  - Fluxos
  - Depósito de Dados
  - Entidades Externas (terminators)

Diagrama de Fluxo de Dados (DFD)

- **Processos**
  - São funções que transformam entradas em saídas, ou seja, processam dados
  - Representação gráfica: círculos (bolhas)
- **Fluxos**
  - Representam a movimentação de informações, blocos de dados
  - Representação gráfica: setas que entram ou saem de processos

Diagrama de Fluxo de Dados (DFD)

- **Depósito de Dados**
  - Usado para modelar um conjunto de dados armazenados
  - Normalmente implementados através de arquivos ou bancos de dados, mas podem assumir outras formas de implementação (conceito de alto nível)
  - Representação gráfica: duas linhas paralelas
- **Entidades Externas**
  - Pessoas, organizações, outros sistemas, etc., com os quais o sistema se comunica
  - Representação gráfica: retângulos

[26] . COPEVE-UFAL – 2009 – UNEAL – Analista de Sistemas

É um conjunto de aplicações com uma arquitetura comum específica de aplicação. O núcleo comum da família de aplicações é reusado cada vez que uma nova aplicação é necessária. O novo desenvolvimento pode envolver a configuração de componentes específicos, a implementação de componentes adicionais e a adaptação de alguns componentes.

Qual opção abaixo corresponde à descrição anterior?

a) Modelo de componentes
b) Componentes de software
c) Linha de produto de software
d) Padrão de projeto composite
e) Padrões de projeto

Linha de Produto de Software

- Técnica que desenvolve uma mesma família de produtos com partes e recursos comuns
- Os produtos da linha têm:
  - Funcionalidade comum
  - Missão específica e segmento de mercado
  - Componentes e arquitetura compartilhados

Linha de Produto de Software – Vantagens

- Ganho de produtividade e agilidade na entrega
- Melhor qualidade, consistência e coerência de funcionalidades
- Presença de mercado da organização

Linha de Produto de Software – O que NÃO é

LPS **NÃO** é:

- Desenvolvimento de um único sistema com reuso
  - Desenvolver um novo sistema reutilizando partes de um outro sistema similar anterior não é LPS.
  - LPS requer a produção de vários produtos de uma mesma família de forma deliberada
- Desenvolvimento baseado em componentes
  - LPS requer um desenvolvimento baseado em componentes, mas é necessário que eles estejam de acordo com a arquitetura de linha de produção.

[27] . FCC – 2008 – TCE-AL – Analista de Sistemas

NÃO é uma tarefa pertinente às camadas concêntricas do Software Configuration Management

a) a Auditoria de Modificação.
b) a Auditoria de Configuração.
c) o Controle de Versão.
d) a Preparação de Relatórios de Estado.
e) o Controle de Modificação.

[28] . FCC – 2008 – METRÔ-SP – Ciências da Computação

Enterprise Service Bus, ESB:

a) fortalece o acoplamento entre o serviço chamado e o meio de transporte.

b) implementa arquitetura orientada a serviço (SOA).

c) necessita de Web Services para ser implementado.

d) tem sua base construída a partir da quebra de funções básicas em partes, que são distribuídas onde for preciso.

e) auxilia no aumento de conexões ponto-a-ponto necessárias para permitir a comunicação entre aplicações.

Enterprise Service Bus

- Arquitetura de integração que possibilita a comunicação de diferentes aplicações através de um “barramento”
- Cada aplicação se comunica com o barramento, desacoplando-as umas das outras
  - A ideia é se afastar da comunicação ponto a ponto, que se torna difícil de manter e gerenciar com o passar do tempo

Vantagens

- O barramento desacopla as aplicações
  - Normalmente se utiliza um protocolo de mensagens, como JMS
- Os dados trafegam de maneira padronizada, normalmente como XML
  - Isso significa que cada aplicação entende a linguagem das outras
- Existe um “adaptador” entre as aplicações e o barramento, fazendo conversão de dados

Gabaritos

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