Análise de Pontos de Função

Métricas de software

Fernando Pedrosa – fpedrosa@gmail.com

Bibliografia

- **Vazquez,** Carlos. *Análise de pontos de função: medição, estimativas e gerenciamento de projetos de software*. **Editora:** Érica. **Edição**: 12

- **Manual de Práticas de Contagem.** *Counting Practices Manual* (CPM). **IFPUG, Edição**: 4.3.1

Conceitos básicos

- **Medida**: um padrão usado para dimensionar algo
  - Unidade de tempo, custo, peso, etc.
- **Métrica**: composição de uma ou mais medidas
  - Horas por Ponto de Função, Tempo médio entre falhas, etc.
- **Medição**: ato de capturar informações, coletar resultados sobre as medidas estabelecidas

Por que medir software?

- Métricas ajudam a responder perguntas cruciais em projetos de software e permitem:
- **Estimar o esforço de desenvolvimento**
  - No atendimento a novos requisitos
  - No estudo de viabilidade
- **Acompanhar o progresso do projeto**
  - Controlando e corrigindo problemas
  - Determinando se o projeto está atrasado, se esgotou o orçamento, etc.

Por que medir software?

- **Tomar decisões**
  - Sobre recursos, escopo, produtividade, riscos, cronograma, etc.
- **Realizar análise de *"make or buy"***
  - Através do histórico de desempenho da empresa e comparação à média do mercado
- **Apoiar contratos**
  - Medindo o produto entregue e remunerando o resultado

O que medir?

- Deve ser medido aquilo que é **relevante** para análise

Exemplos:

- Recursos e Custos
  - Pessoas, ferramentas, orçamento...
- Qualidade
  - Tempo entre falhas, incidentes, problemas...
- Cronograma
  - Atrasos, marcos, entregas...

O que medir?

- Progresso
  - Entregáveis, funcionalidades prontas, documentos validados...
- Tamanho
  - Linhas de código, Pontos de casos de uso
  - **Pontos de Função...**

Qual medida melhor representa "tamanho"?

- **À primeira vista, "linhas de código" parece ser uma unidade interessante**
  - Teoricamente, dois programas de mesma linguagem poderiam ser comparados
  - É fácil de medir
  - Parece ser uma métrica objetiva
- Quanto mais linhas de código (*lines of code* – LOC) tivesse uma aplicação, "maior" ela seria

Linhas de Código (LOC)

- Apesar da aparente facilidade, a métrica de linhas de código deve ser evitada
  - A sua contagem envolve uma série de riscos que inviabilizam o seu uso como medida
- Não é tão objetiva quanto parece
  - É necessário esclarecer diversos pontos antes de medir linhas código

Evite linhas de código porque:

- Não há padronização na contagem
- LOC não tem significado “de negócio” (para usuários e clientes)
- Não há como estimar confiavelmente linhas de código nas fases iniciais do ciclo de vida
- Depender de linguagem de programação é inaceitável

Pontos de Função

- **Análise de Pontos de Função** (APF) é o método adotado como padrão mundial para medição funcional de software
  - Pontos de Função são a medida
- APF mede as funcionalidades fornecidas do ponto de vista do usuário
- É independente de tecnologia utilizada
  - APF procura medir o que o software faz, e não COMO ele foi construído.

Pontos de Função

- **Cuidado**: APF não mede diretamente esforço, produtividade ou custo
- Pontos de Função representam exclusivamente o tamanho funcional do software
- Este tamanho funcional, junto com outras variáveis (dados históricos), pode ser usado para derivar outras informações

Exercícios [1]

(FIOCRUZ – FGV 2010)

A métrica “Pontos de Função” (Function Point, FP) é usada efetivamente como meio para medir a funcionalidade entregue por um sistema. Considerando dados históricos, analise as afirmativas associadas ao uso da FP.

I. Estimar o custo ou esforço necessário para projetar, codificar e testar o software.
II. Prever o número de erros que vão ser encontrados durante o teste.
III. Prever o número de componentes e/ou o número de linhas de código projetadas no sistema implementado. Assinale:

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 somente as afirmativas I e II estiverem corretas.
e) se todas as afirmativas estiverem corretas.

Exercícios [2]

(TRE/SP – FCC 2012)

Sobre a análise de pontos por função, considere:

I. É um método de contagem padrão capaz de medir as funcionalidades de um sistema sobre o ponto de vista do desenvolvedor.

II. A contagem sem ajustes (UFPC – unadjusted function point count) reflete as funcionalidades contáveis específicas disponibilizadas pelo sistema ou aplicação para o usuário.

III. É uma ferramenta para ajudar usuários a determinar os benefícios de um pacote de aplicativos para sua empresa por meio de contagem das funcionalidades que especificamente atendem seus requerimentos.

Está correto o que consta em

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

Exercícios [3]

(TRE/ES – CESPE 2011)

Logo após o início das atividades técnicas de um projeto, o gerente e a equipe de desenvolvimento devem estimar o trabalho a ser realizado, os recursos necessários, o tempo de duração e, por fim, o custo do projeto. Para se estimar o tamanho do software, deve-se seguir a métrica de pontos de função (PF), desde que esta seja compatível com a tecnologia empregada na implementação do sistema.

Exercícios [4]

(TER/BA– CESPE 2010)

A precisão de estimativas de tamanho, que depende de informações que nem sempre estão disponíveis no início dos projetos, auxilia a discussão de contratos ou determinação da viabilidade do projeto em termos da análise de custos e benefícios.

(SERPRO – CESPE 2008)

Métricas baseadas em pontos por função e em LOC têm sido consideradas relativamente precisas para prever o esforço e o custo de desenvolvimento de software. No entanto, ao se utilizar pontos por função e LOC para estimativas, não devem ser usadas referências históricas de informação.

Histório da APF/CPM (1970)

- Allan Albrecht (IBM) foi encarregado de medir vários projetos de software
  - Mas os projetos utilizavam diferentes linguagens de programação
  - Era inviável utilizar métricas baseadas em Linhas de Código (LOC)
- Era necessário uma medida que fosse independente de tecnologia
- Nascia a técnica de “Análise de Pontos de Função”

Histório da APF/CPM (1980)

- Com a popularização do método, foi criado o **IFPUG**
  - *International Function Point Users Group*
- O IFPUG tem como objetivo promover o uso da técnica de pontos de função mundialmente
- É quem mantém o Manual de Práticas de Contagem (CPM)
  - ***Counting Practices Manual***

Counting Practices Manual (CPM)

- Detalha a contagem de pontos de função
- Garante que as contagens sejam consistentes de acordos com as práticas do IFPUG
- Fornece um guia de contagem para as metodologias e técnicas mais conhecidas
- Mantém conformidade com a norma ISO/IEC 14143:2007
- Provê um entendimento comum que dê suporte a ferramentas automatizadas

Histórico da APF/CPM (1990)

- **1990 (CPM versão 3.0)**
  - Foi o primeiro grande marco do manual, que passou a ser descrito de maneira coerente, em um único documento
- **1994 (CPM versão 4.0)**
  - Abordou estimativa de contagem nas fases iniciais do ciclo de vida
  - Adicionou práticas para contagem da GUI
  - Exemplos foram incluídos e estudos de casos foram adicionados

Histórico da APF/CPM (2000)

- **2004 (CPM versão 4.2)**
  - Não modificou qualquer regra publicada anteriormente
  - Forneceu esclarecimentos e melhores interpretações das regras existentes
  - Durante muito tempo foi o manual utilizado pelas organizações na contagem de sistemas orientados a objetos

Histórico da APF/CPM (2010)

- **2010 (CPM versão 4.3.1)**
  - Versão mais recente
  - Compatível com a formatação e conceitos da norma ISO 14143
  - Descrito pela norma ISO 20926
  - Diminuiu a ênfase dada às "características gerais do sistema"
- O CPM é um documento "vivo", e o IFPUG tem a intenção de atualizá-lo sempre que necessário

NESMA

- Outro grande grupo de usuários é a NESMA
  - *Netherlands Software Metrics Users Association*
- Suas ações e objetivos são próximos aos do IFPUG, com algumas diretrizes diferentes
- Abrange três tipos de contagem: detalhada, estimativa e indicativa

Pontos por Caso de Uso

- Com a popularização de RUP e UML, foi proposta a técnica de “Pontos por Caso de Uso”
- O método consiste, basicamente, de:
  - Contar atores e casos de uso
  - Calcular os PCUs não-ajustados
  - Ajustar os PCUs de acordo com a “complexidade técnica” e a “complexidade ambiental” da aplicação

Pontos por Caso de Uso

- Porém, é uma técnica que não “pegou”:
  - Só pode ser aplicada em projetos que utilizem Casos de Uso
  - Não pode ser empregada antes de concluída a análise de requisitos
  - É difícil conseguir medidas padronizadas, pois não existe padrão de escrita para CDU
  - A determinação da “complexidade técnica” e “complexidade ambiental” é muito subjetiva
  - Não existe um grupo coeso de usuários

Exercícios [5]

(STM – CESPE 2011)

A NESMA (Netherlands Software Metrics Users Association) tem objetivos e ações bem próximos aos do IFPUG; ambos apresentam abordagens semelhantes para a aplicação da análise de pontos de função em projetos de melhoria de software e na fase inicial do desenvolvimento do produto de software.

Exercícios [6]

(TJ/PE – FCC 2012)
Considere:

I. Contagem de pf detalhada.
II. Contagem de pf estimativa.
III. Contagem de pf indicativa.

Quanto ao tipo de contagem, a Netherlands Software Metrics Association reconhece o que consta em

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

Exercícios [7]

(MEC – FGV 2009)
As métricas de software podem ser utilizadas para estimar o esforço em um projeto de software. Com relação aos pontos de função e pontos de caso de uso, analise as afirmativas a seguir:

I. Na métrica de PCU os atores são classificados e possuem sempre o mesmo nível de complexidade.
II. A métrica de Pontos de Caso de Uso (PCU) pode ser aplicada somente em projetos de software que tenham sido descritos por casos de uso.
III. A análise de pontos de função (APF) é uma técnica para medir o tamanho funcional de um software do ponto de vista do usuário. Assinale:

a) Se somente a afirmativa I estiver correta.
b) Se somente as afirmativas I e II estiverem corretas.
c) Se somente as afirmativas I e III estiverem corretas.
d) Se somente as afirmativas II e III estiverem corretas.
e) Se todas as afirmativas estiverem corretas.

Análise de Pontos de Função

O que é medido?

- APF mede o “tamanho” do software, quantificando tarefas e serviços
- Baseia-se, primariamente, no projeto lógico da aplicação
- Os objetivos são de medir:
  - A funcionalidade implementada, que o usuário solicita e recebe
  - A funcionalidade impactada pelo desenvolvimento, melhoria e manutenção do software

Para quê medir?

- Dar suporte à análise de qualidade e produtividade
- Estimar custo e esforço de desenvolvimento, melhoria e evolução
- Fornecer um fator “normalizado” para comparação de software
- Determinar o tamanho de um pacote de aplicação adquirido de terceiros
- ... e outros propósitos que sejam úteis

Aplicabilidade

- APF pode ser aplicado a todos os domínios funcionais
- O IFPUG identificou diferentes taxas de entrega (Horas/PF)
  - Para diferentes domínios funcionais
  - Calibradas para diversos projetos de tamanhos e complexidades diferentes
- Esta variedade de projetos pode ser comprovada pelo repositório da ISBSG
  - Mais de 6.000 projetos diferentes

Exercícios [8]

(TRT/11 – FCC 2012)
Segundo a IFPUG em relação à métrica do software por análise por pontos de função, considere:

I. Análise por pontos de função executa a medição do software determinando a quantidade de funcionalidades que o software fornece ao usuário baseado principalmente na arquitetura lógica.
II. O objetivo da análise por pontos de função é medir as funcionalidades que o usuário requisita e recebe e, também, medir o desenvolvimento e manutenção do software com dependência na implementação utilizada pela empresa.
III. O processo de contagem dos pontos de função deve ser simples o suficiente para minimizar a sobrecarga do processo de medida e consistente dentre os vários projetos e organizações.

Está correto o que se afirma em:
a) I e II, apenas. b) I e III, apenas. c) II e III, apenas. d) III, apenas. e) I, II e III.

Exercícios [9]

(TRF4 – FCC 2010)

Sobre a métrica análise por pontos de função, é correto afirmar:

a) Não pode ser aplicada para estimar esforço de manutenção em sistemas já em funcionamento.

b) A medida não pode ser aplicada com base na descrição arquitetural do projeto, mas sim no código desenvolvido.

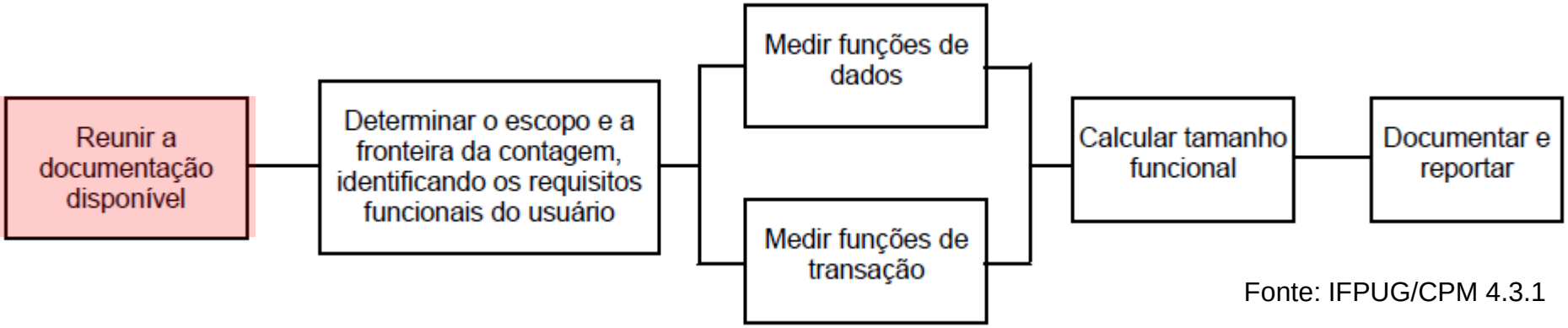
c) É dependente da tecnologia utilizada no desenvolvimento.

d) A contagem de pontos de função pode ser aplicada logo após a definição da arquitetura, permitindo estimar o esforço e o cronograma de implementação de um projeto.

e) Para determinar o número de pontos de função, deve-se desconsiderar a contagem de dados e de transações.

O processo de medição funcional

1. Reunir a documentação disponível



Fonte: IFPUG/CPM 4.3.1

Reunir a Documentação Disponível

- Deve ser obtida documentação suficiente para conduzir a contagem funcional
- Também pode ser requerido o acesso a especialistas capazes de fornecer informações na falta de documentação
- Geralmente, não há um único documento capaz de suprir todas as necessidades de informação

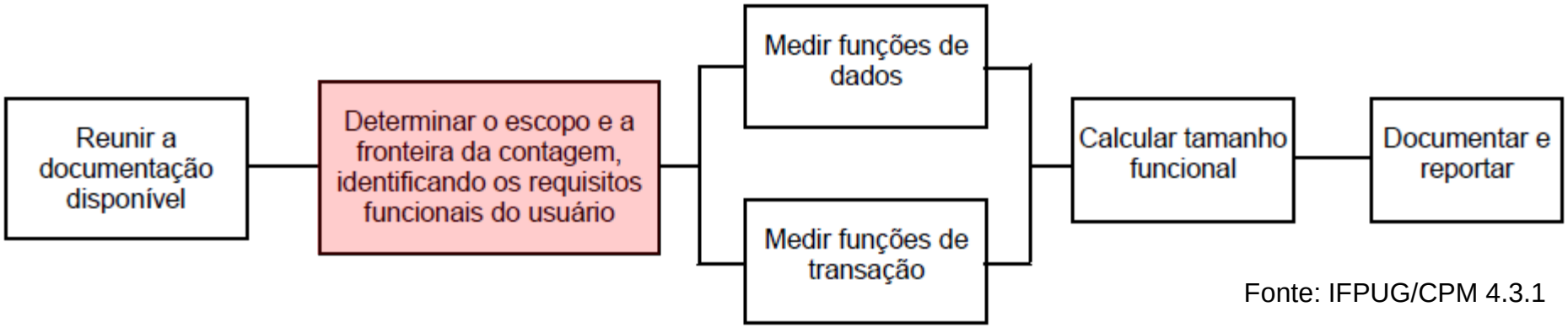
Itens úteis

- Documento de Requisitos
- Diagrama de Entidades
- Modelos de Dados e Objetos
- Exemplos de relatórios, telas e outras interfaces com o usuário
- Guias, Manuais de Uso, Materiais de Treinamento
- Especialistas na Aplicação, Clientes/Usuários da Aplicação

O processo de medição funcional

2. Determinar o escopo e a fronteira da contagem

2.1 – Identificar o propósito da contagem
2.2 – Identificar o tipo da contagem
2.3 – Determinar o escopo da contagem
2.4 – Determinar a fronteira de cada aplicação
2.5 – Identificar os requisitos funcionais



Fonte: IFPUG/CPM 4.3.1

Identificar o propósito da contagem

- Toda contagem de PF deve ser conduzida para responder uma questão de negócio
- O propósito da contagem determina o seu escopo
  - Pode ser para determinar o tamanho de uma release
  - Pode ser para determinar o tamanho de uma aplicação como um todo
- Enfim, o negócio é quem determina

Identificar o tipo da contagem

- Com base no propósito identificado, a contagem pode ser definida como:
- Projeto de desenvolvimento
  - Projeto para desenvolver uma primeira versão de um software
  - Mede-se o que vai ser entregue ao usuário
  - A contagem ocorre várias vezes durante o ciclo de vida, e pode-se dizer que é uma contagem estimada

Identificar o tipo da contagem (cont.)

- Projeto de melhoria
  - Projeto para desenvolver e entregar manutenções no software
  - Medem-se as funcionalidades adicionadas, alteradas ou removidas da aplicação
  - As manutenções podem ser Adaptativas, Corretivas ou Perfectivas

Identificar o tipo da contagem (cont.)

- Contagem de Aplicação
  - É a contagem do "tamanho funcional instalado" (ou *baseline*)
  - Fornece uma medida das funcionalidades atuais que o aplicativo fornece ao usuário
  - O número é inicializado quando o projeto de desenvolvimento é finalizado

Identificar o escopo da contagem

- O escopo define quais funções serão incluídas na contagem
  - Devo incluir toda aplicação? Devo incluir apenas parte dela?
- Pode abranger:
  - Todas as funcionalidades disponíveis
  - Apenas aquelas utilizadas pelo usuário
  - Apenas algumas funcionalidades específicas (relatórios, cadastros, etc.)
- Depende do propósito da contagem!

Identificar a fronteira da aplicação

- A fronteira é uma interface conceitual entre o software e seus usuários
  - É independente de considerações técnicas ou implementação
- A fronteira:
  - Define o que é externo à aplicação
  - Separa o software medido do usuário
  - Atua como uma "membrana" à passagem dos dados (in/out)
  - Depende da visão do usuário

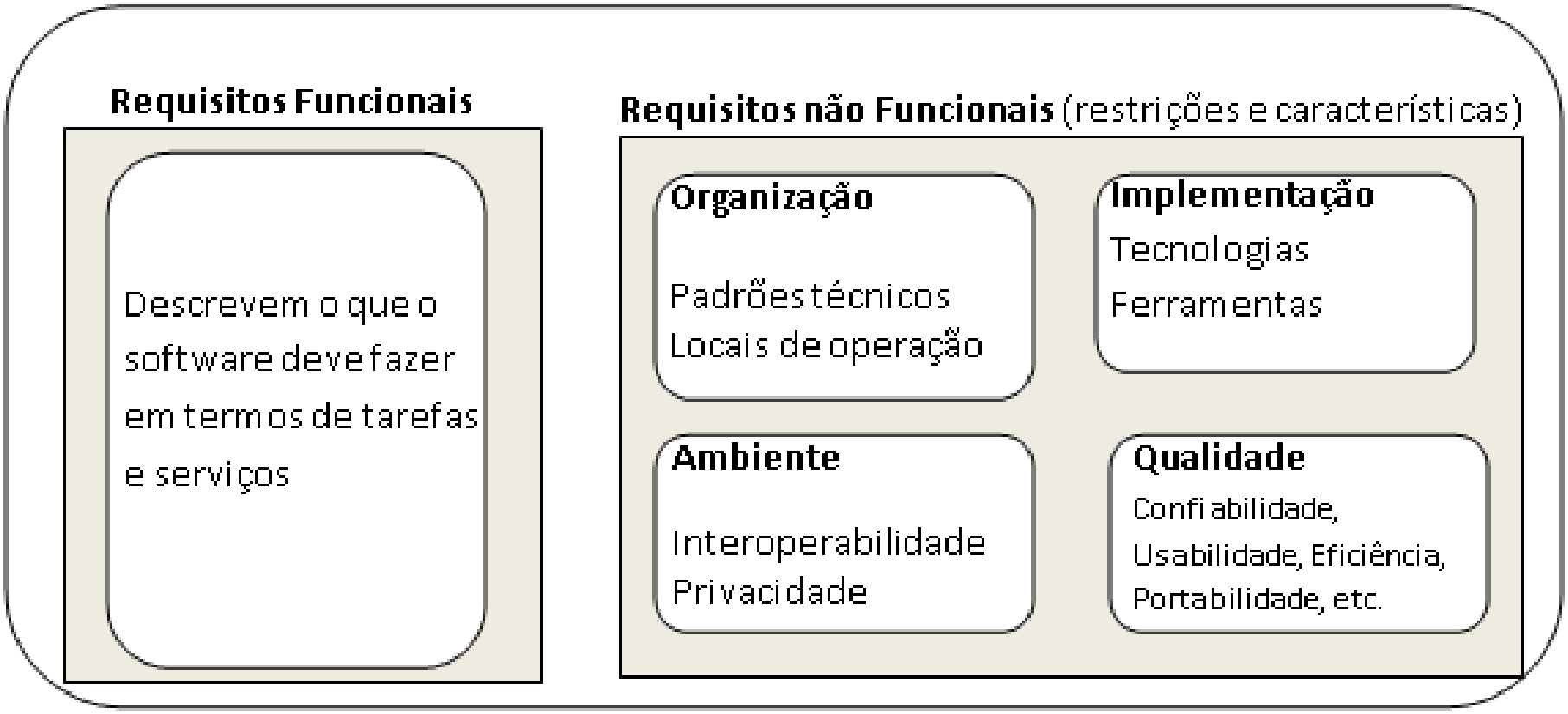
(Sobre o Usuário e sua Visão)

- Um usuário é qualquer entidade que se comunica ou interage com o software
  - Podem ser pessoas, hardware, dispositivos, outros sistemas, etc.
- A visão do usuário representa suas necessidades, na sua linguagem
  - Descreve as funções do negócio
  - Normalmente é declarada verbalmente
  - Pode ser usada para medir funcionalidades
  - É registrada em documentos diversos

Identificar os requisitos funcionais

- Os requisitos do usuário podem misturar requisitos funcionais e não funcionais
- É seu papel identificar quais desses requisitos são funcionais e **descartar** os não funcionais
  - RFs: capturam o que o software deve fazer em termos de funções e serviços
  - RNFs: são restrições ou qualidades específicas do sistema

RF x RNF (ISO 14143)



Exercícios [10]

(TCE/AP – FCC 2012)

Um dos primeiros passos para efetuar a contagem por pontos de função de um sistema, é definir o tipo de contagem que será efetuado. Esses tipos se dividem em

a) entrada, saída e processamento.
b) requisitos, elaboração e testes.
c) desenvolvimento, manutenção e aplicação.
d) controle, mecanismo e processamento
e) lógico, físico e modelagem.

Exercícios [11]

(STM – CESPE 2011)

O conceito de projeto de melhoria do IFPUG envolve as manutenções evolutivas, corretivas e preventivas da aplicação.

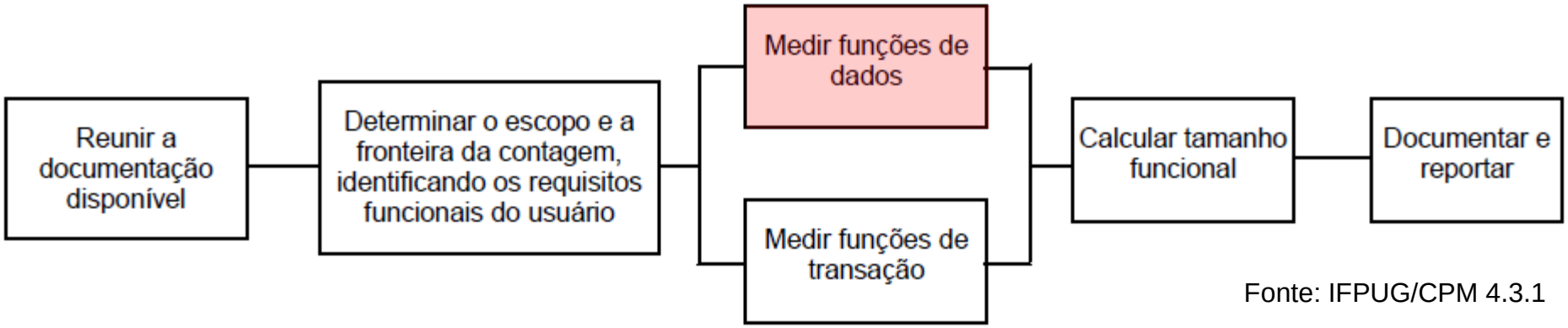
(ANAC – CESPE 2009)

Os tipos de contagem de pontos por função podem ser de projetos de desenvolvimento, projetos de melhorias ou de aplicações, sendo a contagem de pontos por função por estimativa realizada nos estágios iniciais de contagem.

O processo de medição funcional

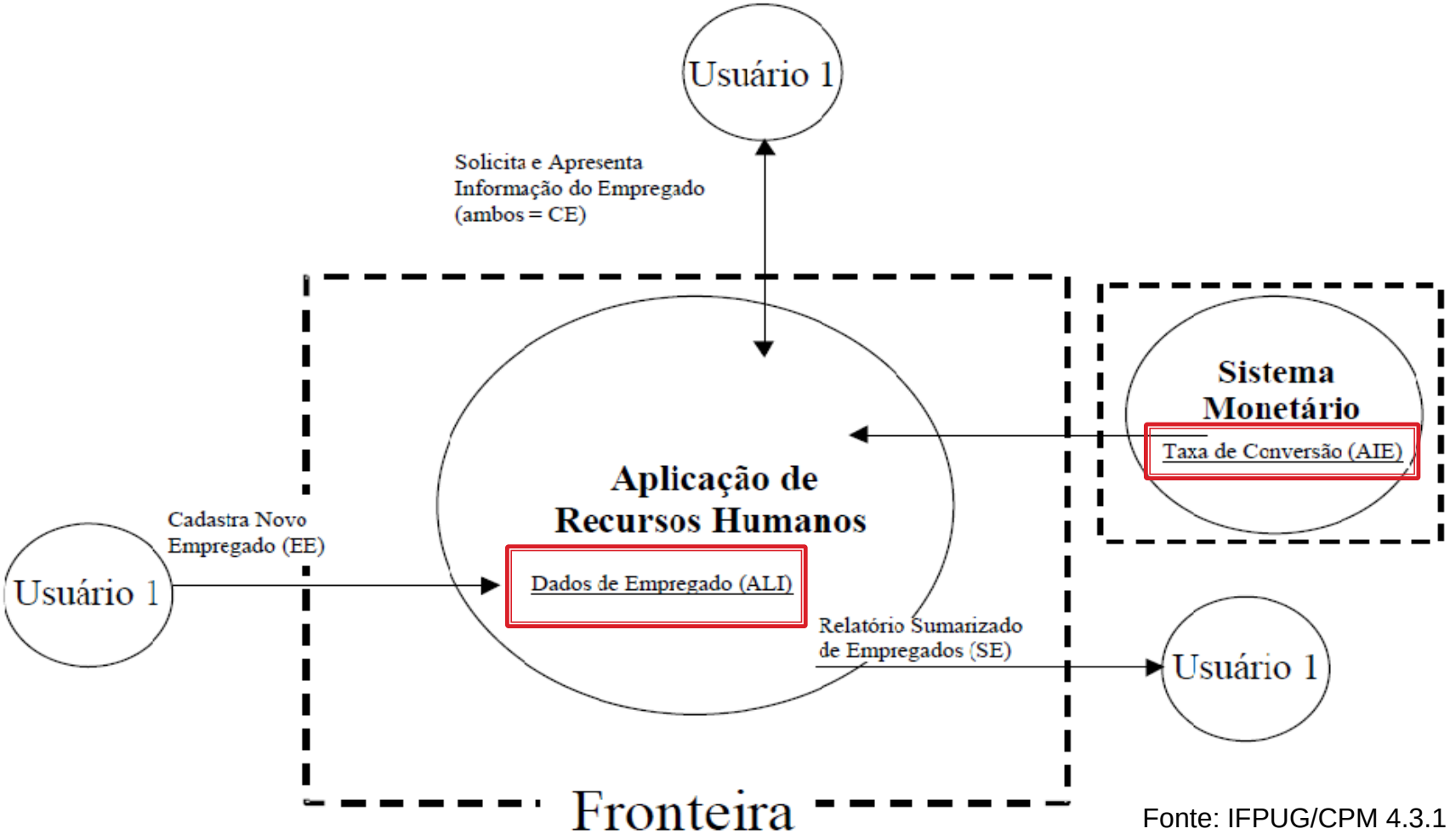
3. Medir funções de dados

3.1 – Identificar funções de dados (ALI e AIE)
3.2 – Contar DERs e RLRs para cada função de dados
3.3 – Determinar a complexidade funcional de cada função de dados
3.4 – Determinar o tamanho funcional de cada função de dados



Fonte: IFPUG/CPM 4.3.1

Aplicação de RH (exemplo)



Fonte: IFPUG/CPM 4.3.1

Identificar funções de dados

- **Arquivo Lógico Interno (ALI)**
  - Grupo de dados ou informações de controle reconhecidos pelo usuário e mantido dentro da fronteira da aplicação
  - Sua principal intenção é a de armazenar dados mantidos pela aplicação sendo medida
- **Arquivo de Interface Externa (AIE)**
  - Grupo de dados ou informações de controle reconhecidos pelo usuário mantido dentro da fronteira de outra aplicação
  - Sua principal intenção é a de armazenar dados referenciados pela aplicação sendo medida
  - Um AIE é, sempre, um ALI em outra aplicação

Identificar DERs e RLRs

- **Dado Elementar Referenciado (DER)**
  - Atributo único, reconhecido pelo usuário, e não repetido
  - É como se fossem campos de uma tabela, ou atributos de um objeto
  - A sua identificação precisa depende da visão e utilização do usuário
- **Registro Lógico Referenciado (RLR)**
  - Subgrupo de dados elementares referenciados, reconhecido pelo usuário dentro de um ALI ou AIE
  - Por *default*, toda função de dados tem um RLR, mas o usuário pode perceber mais subgupos de dados

(Mapeando os conceitos)

Conceito da Modelagem de Dados	Termo da Modelagem de Dados	Termo de Base de Dados Relacional	Termo da APF	Conceito da APF
Menor unidade de dado com nome que tem significado para o mundo real	Item de Dados	Atributo ou Coluna	Tipo de Dado Elementar (DER)	Um tipo de dado elementar (DER) é um campo único, não-repetido, reconhecido pelo usuário
Grupos de itens relacionados os quais são tratados como uma unidade	Registro	Linha ou Tupla	Tipo de Registro Elementar (RLR)	Um tipo de registro elementar (RLR) é um subgrupo de elementos de dados reconhecido pelo usuário e armazenado em um ALI ou AIE
Coleção de registros de um único tipo	Arquivo	Tabela	Arquivo Lógico (Arquivo Lógico Interno - ALI ou Arquivo de Interface Externa - AIE)	Arquivo refere-se a grupos de dados logicamente relacionados e não à implementação física desses grupos de dados

Fonte: IFPUG/CPM 4.3.1

Determinar a complexidade funcional de cada função de dados

A complexidade de cada função de dados deve ser determinada de acordo com a seguinte tabela:

		DERs		
		1 – 19	20 – 50	> 50
RLRs	1	Baixa	Baixa	Média
	2 – 5	Baixa	Média	Alta
	> 5	Média	Alta	Alta

Determinar o tamanho funcional de cada função de dados

O tamanho funcional de cada função de dados deve ser determinado de acordo com a seguinte tabela:

		Tipo	
		ALI	AIE
Complexidade funcional	Baixa	7	5
	Média	10	7
	Alta	15	10

Exercícios [12]

(MPE/PI – CESPE 2012)

Conforme a metodologia definida pelo IFPUG (International Function Point User Group), computam-se como arquivos de interface externa os dados que sejam recebidos de outra aplicação e utilizados para alterar ou remover dados de um arquivo lógico interno

(BRB – CESPE 2011)

Se duas aplicações mantiverem o mesmo arquivo lógico interno, então esse arquivo será contado apenas na aplicação que detém o arquivo físico.

Exercícios [13]

(Correios – CESPE 2011)

Um arquivo de interface externa é obrigatoriamente um ALI de outra aplicação.

(SECONT/ES – CESPE 2009)

A análise de pontos de função mede o software por meio da quantificação da funcionalidade que este provê ao usuário. Nesse método são consideradas as funções de dados e as funções de transação, que contribuem para a contagem de pontos de função não ajustados. Essa contribuição é determinada a partir do tipo e da complexidade das funções. Entre todos os tipos de funções que podem ser identificados em um software, os arquivos lógicos internos de alta complexidade são os que representam a maior contribuição para a contagem de pontos de função não ajustados.

Exercícios [14]

(DPE/SP – FCC 2010)

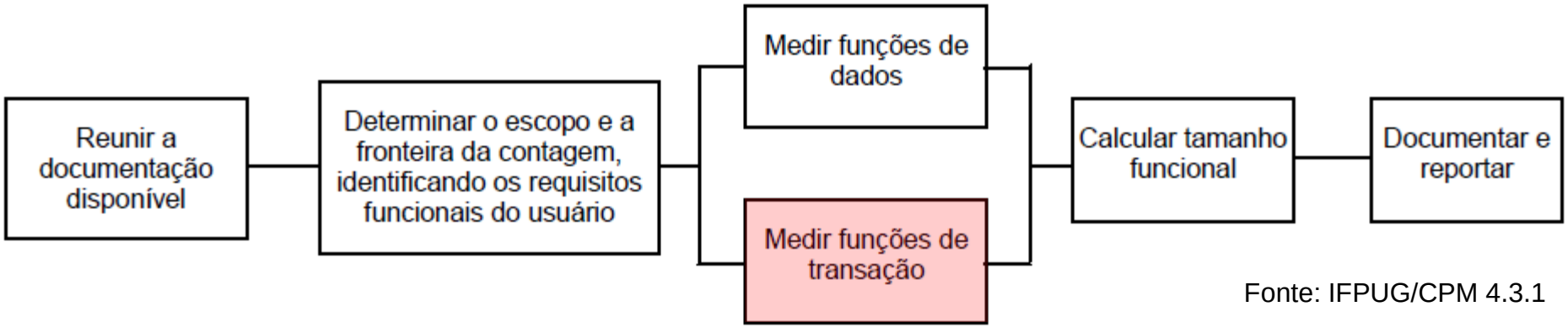
A contagem do tipo de funções de dados, aplicada na Análise de Pontos por Função, cuja entidade lógica e persistente se equivale a um depósito de dados que sofre manutenção fora da aplicação, trata-se do critério

a) EIF ou AIE.
b) ILF ou ALI.
c) EI ou EE.
d) EO ou SE.
e) EQ ou CE

O processo de medição funcional

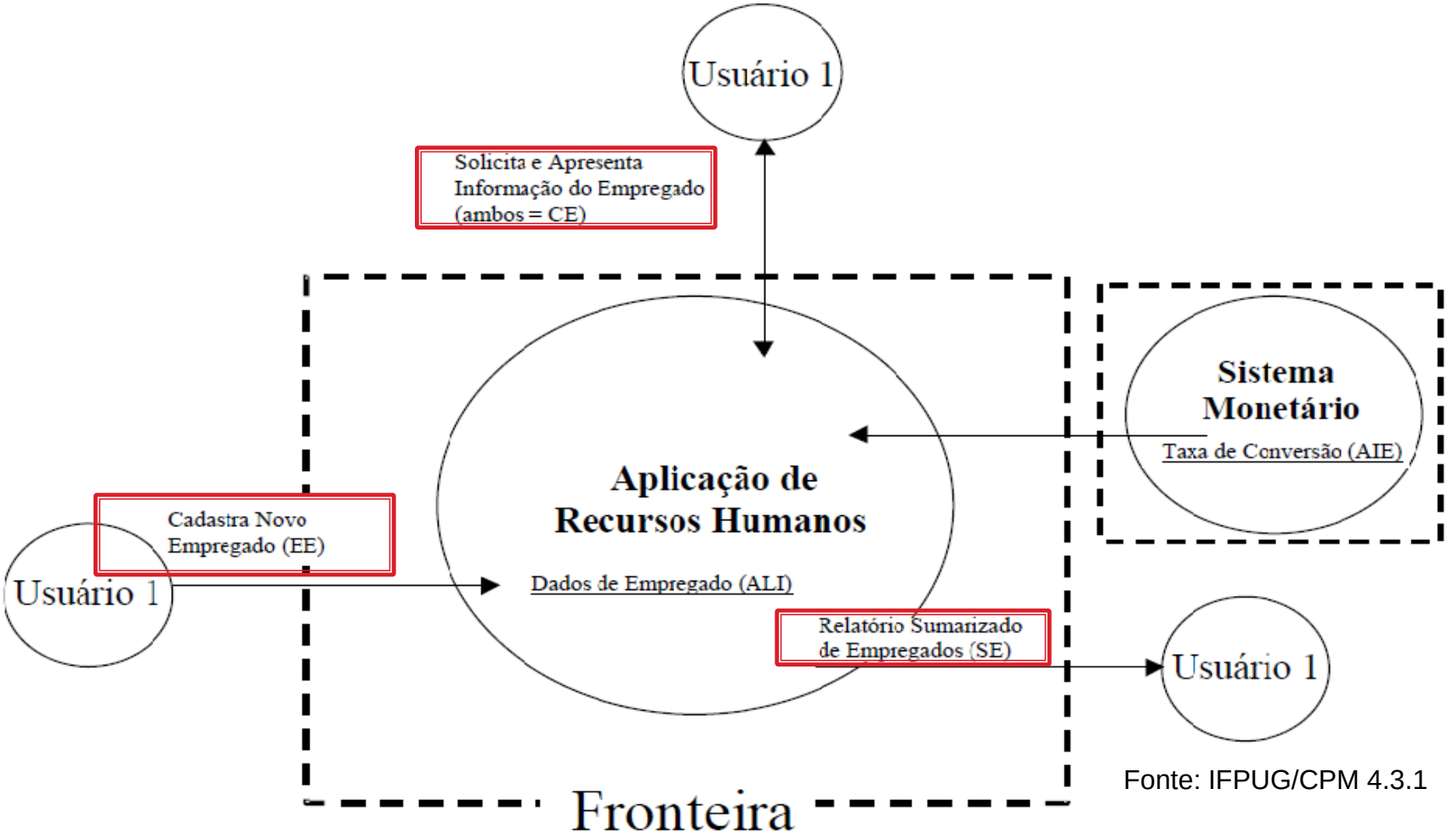
4. Medir funções de transação

4.1 – Identificar cada processo elementar requerido pelo usuário
4.2 – Classificar cada processo elementar como EE, SE ou CE
4.3 – Contar ALRs e DERs
4.4 – Determinar a complexidade funcional de cada função de transação
4.5 – Determinar o tamanho funcional de cada função de transação



Fonte: IFPUG/CPM 4.3.1

Aplicação de RH (exemplo)



Fonte: IFPUG/CPM 4.3.1

Identificar processos elementares

- **Processo Elementar**
  - Menor unidade de atividade significativa para o usuário
  - Constitui uma transação completa e autocontida
  - Ao final de sua execução, deixa o negócio da aplicação em um estado consistente

Para cada processo elementar, identificar EE, SE ou CE

- **Entrada Externa (EE)**
  - Processa dados recebidos de fora da fronteira da aplicação
  - Sua intenção primária é manter ALIs ou alterar o comportamento da aplicação
- **Saída Externa (SE)**
  - Envia dados para fora da fronteira da aplicação, mas inclui processamento adicional
  - Sua intenção primária é de apresentar dados ao usuário através de lógica de processamento que não seja apenas a recuperação das informações

Para cada processo elementar, identificar EE, SE ou CE

- **Consulta Externa (CE)**
  - Processo que envia dados para fora da fronteira da aplicação sem processamento adicional
  - Sua intenção primária é apresentar dados ao usuário através da recuperação destes dados
  - Sua lógica de processamento não contém fórmula matemática, nem cálculo, nem cria dados derivados

(Intenções Primárias: EE, SE, CE)

A Tabela a seguir apresenta um resumo do relacionamento entre a intenção primária e o tipo de função de transação.

Função	Tipo de função de transação		
	EE	SE	CE
Alterar o comportamento da aplicação	IP	F	N/A
Manter um ou mais ALIs	IP	F	N/A
Apresentar informações ao usuário	F	IP	IP

legenda

IP	a intenção primária do tipo de função de transação
F	uma função do tipo de função de transação que às vezes está presente, mas que não é a intenção primária
N/A	o tipo de função de transação não pode executar este tipo de função

Fonte: IFPUG/CPM 4.3.1

Contar ALRs e DERs

- **Arquivo Lógico Referenciado (ALR)**
  - ALI ou AIE que foi acessado (lido e/ou gravado) por uma função de transação
- **Dado Elementar Referenciado (DER)**
  - No contexto de funções de transação, é todo o dado que atravessa a fronteira (entra e/ou sai) durante o processamento da transação

Determinar a complexidade de cada função de transação

A complexidade funcional de cada função de transação será determinada utilizando-se o número de ALRs e DERs, em conformidade com as tabelas a seguir:

Tabela 6 — Complexidade funcional das EE

		DERs		
		1 – 4	5 – 15	> 15
ALRs	0 – 1	Baixa	Baixa	Média
	2	Baixa	Média	Alta
	> 2	Média	Alta	Alta

Tabela 7 — Complexidade funcional das SE e CE

		DERs		
		1 – 5	6 – 19	> 19
ALRs	0 – 1	Baixa	Baixa	Média
	2 – 3	Baixa	Média	Alta
	> 3	Média	Alta	Alta
NOTA	Uma CE tem no mínimo 1 ALR.			

Determinar o tamanho funcional de cada função de transação

O tamanho funcional de cada função de transação será determinado utilizando-se o tipo e a complexidade funcional, de acordo com a tabela a seguir:

		Tipo		
		EE	**SE**	**CE**
Complexidade Funcional	Baixa	3	4	3
	Média	4	5	4
	Alta	6	7	6

Exercícios [15]

(INFRAERO – FCC 2011)

Analise a tabela utilizada no cálculo de Pontos de Função:

Tipo de Função	Complexidade Funcional		
	Simples	Média	Complexa
I	7	10	15
II	5	7	10
III	4	5	7

Preenchem correta e respectivamente os tipos de função I, II e III:

a) ALI, AIE e SE.
b) ALI, CE e AIE.
c) CE, EE e ALI.
d) AIE, AL e EE.
e) EE, CE e SE.

Exercícios [16]

(BACEN - CESGRANRIO 2010)

Uma empresa deseja desenvolver internamente um sistema de controle de visitantes. Foi solicitada uma funcionalidade em que, dado um CPF, sejam retornados, em uma tela, os seguintes dados:

. nome completo; . data de nascimento; . período da última visita;
. quantidade de visitas.

De acordo com a Análise de Pontos de Função, quantas funções transacionais devem ser contabilizadas para essa tela?

a) 0 b) 1 c) 2 d) 3 e) 4

Exercícios [17]

(TRANSPETRO - CESGRANRIO 2011)
Diversas são as métricas utilizadas em engenharia de software. Para a utilização da métrica de pontos de função para medir a funcionalidade entregue por uma aplicação S, a fronteira dessa aplicação deve ser definida. A seguir, diversas contagens devem ser realizadas, como a quantidade de

a) arquivos de interfaces externas (external interfaces files, EIS), agrupamentos físicos de dados mantidos dentro da fronteira de S e usados para que S forneça informações a usuários ou a sistemas externos.

b) arquivos lógicos internos (internal logical files, ILF), agrupamentos físicos de dados armazenados fora da fronteira de S e usados para que S obtenha informações de usuários ou de sistemas externos.

Exercícios [17]

c) consultas externas (external inquiries, EQ), processos elementares que solicitam informações externas à fronteira de S e cujos resultados são armazenados em arquivos lógicos internos.

d) entradas externas (external inputs), processos elementares que processam informações de controle ou de dados provenientes de fora da fronteira de S.

e) milhares de linhas de código da aplicação (KLOC), utilizada para obter a quantidade de arquivos lógicos internos e externos necessários para armazenamento de dados usados por S.

Exercícios [18]

(BRB – CESPE 2011)

Uma consulta que possua contador incrementado é considerada uma saída externa.

(TJ/ES – CESPE 2011)

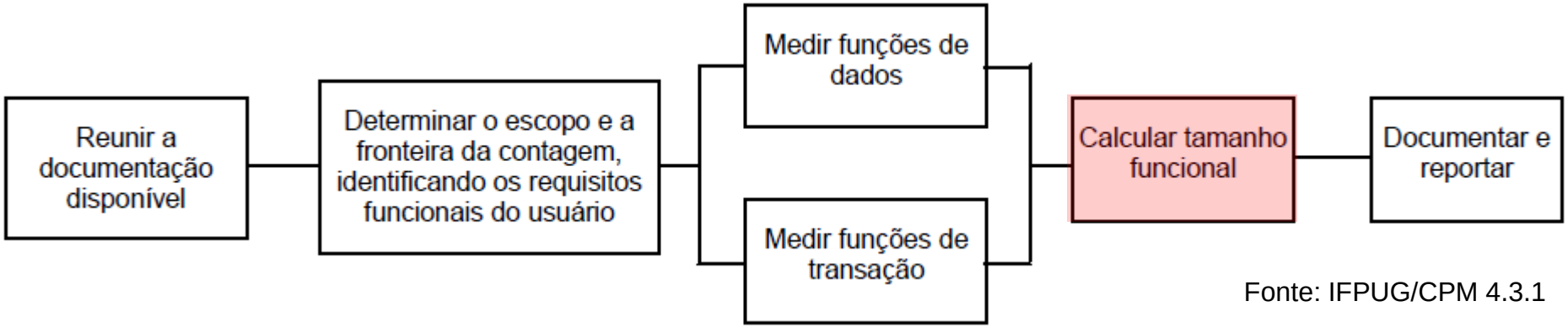
De acordo com o manual de contagem de pontos de função, consulta externa é um processo elementar que envia dados ou informações de controle para fora da fronteira, sendo considerada componente funcional básico.

(Correios – CESPE 2011)

Uma consulta externa disponibiliza informações para o usuário por meio de lógica de processamento, ou seja, não se limita apenas a recuperação de dados. A lógica de processamento deve conter pelo menos uma fórmula matemática ou cálculo, ou criar dados derivados.

O processo de medição funcional

5. Calcular o tamanho funcional



Fonte: IFPUG/CPM 4.3.1

Calculando o tamanho funcional

Projeto de Desenvolvimento

- DFP = ADD + CFP
- Onde “ADD” é o tamanho das funções a serem entregues ao usuário pelo projeto de desenvolvimento
- “CFP” é o tamanho da funcionalidade de conversão
  - São funções construídas e entregues no momento da instalação da aplicação para converter dados

Calculando o tamanho funcional

Contagem de Aplicação

- AFP = ADD
- Onde “ADD” é o tamanho das funções existentes no momento da contagem da aplicação

Calculando o tamanho funcional

Contagem de Melhoria

- EFP = ADD + CHGA + CFP + DEL, onde
- “ADD” é o tamanho das funções incluídas
- “CHGA” é o tamanho das funções alteradas
- “CFP” é o tamanho das funções de conversão
- “DEL” é o tamanho das funções excluídas

Exercícios [19]

(IPHAN – FUNIVERSA 2009)

A tabela abaixo contém o levantamento feito para um sistema que utiliza a Análise por Pontos de Função (APF).

	Baixa	Média	Alta
EE	6	0	4
SE	0	0	3
CE	5	1	1
ALI	3	1	0
AIE	1	0	0

Qual o total de pontos de função (PF) não ajustados que utiliza a contagem detalhada?

a) 116 b) 121 c) 124 d) 126 e) 155

Exercícios [20]

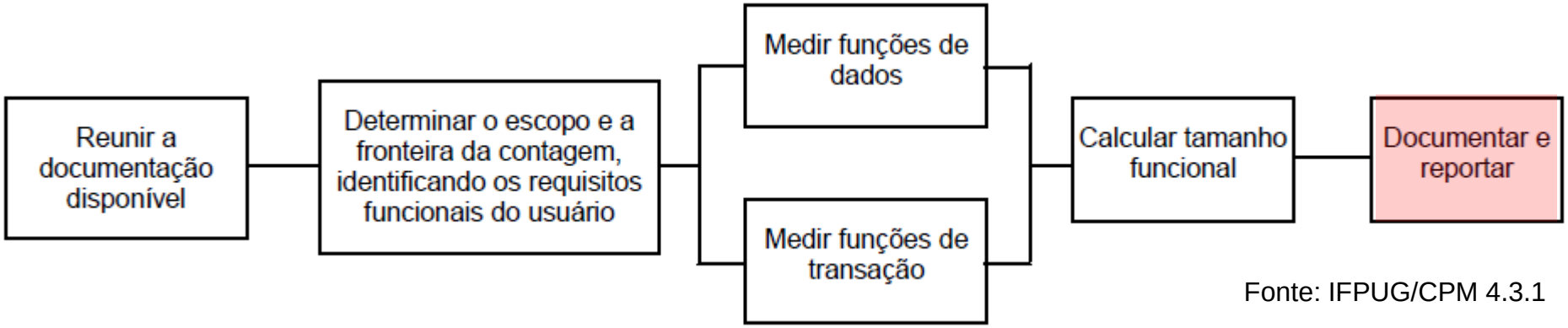
(TRE/CE – FCC 2012)

Considere 3 AIEs simples, 5 EEs médias, 8 CEs complexas, 3 ALIs complexos e 7 SEs médias. O cálculo de PF bruto é

a) 136.
b) 148.
c) 159.
d) 163.
e) 212.

O processo de medição funcional

6. Documentar e reportar



Fonte: IFPUG/CPM 4.3.1

Documentar e Reportar

- Nem sempre o que interessa na contagem é apenas o número final
- É entregue também toda a memória de cálculo (planilha) da contagem, pois isso:
  - Permite-nos conferir se o resultado está certo ou não
  - Agrega valor e confiabilidade à medição
  - Facilita um eventual processo de auditoria;
  - Minimiza os erros do analista responsável

Documentação

- O nível de documentação ser adequado ao propósito da contagem
- O manual propõe que se registrem:
  - O propósito e tipo da contagem
  - O escopo da contagem e a fronteira da aplicação
  - A data da contagem
  - Uma lista de todas as funções contadas, incluindo o respectivo tipo e complexidade
  - O resultado da contagem, ...

Relatório (Reportar)

- É interessante que seja mantido um padrão na prática de reportar os resultados das contagens de PF
  - Isso permite ao leitor identificar que técnica foi utilizada (IFPUG, NESMA, CoCoMo, etc.)
- Padrão IFPUG/CPM:
  - **S FP (IFPUG-IS)**
  - S é o resultado da contagem, FP é a unidade de tamanho e IS significa Padrão Internacional
  - Exemplo: *250 FP (IFPUG-ISSO/IEC 20926:200x)*

Apêndice: Fator de Ajuste

Fator de Ajuste

- O cálculo de pontos de função “brutos” não considera aspectos técnicos ou não funcionais
  - A medição é estritamente funcional
- O Fator de Ajuste é uma tentativa de compensar ou “ajustar” alguns pontos de função com base em características técnicas
  - São as chamadas “Características Gerais do Sistema”

Níveis de Influência

- Com base nos requisitos do usuário, cada CGS deve ter seu nível de influência avaliado numa escala de 0 a 5

Pontue como	Influência no Sistema
0	Não presente ou sem influência
1	Influência Mínima
2	Influência Moderada
3	Influência Média
4	Influência Significativa
5	Forte influência

- Cada CGS contém diretrizes para determinar o seu nível de influência

Exemplo (CGS 06)

- **CGS 06 – Entrada de Dados On-Line**
  - Descreve os níveis segundo os quais os dados são informados ou recuperados através de transações interativas

0.	Todas as transações são processadas de modo batch
1.	1% a 7% das transações são interativas
2.	8% a 15% das transações são interativas
3.	16% a 23% das transações são interativas
4.	24% a 30% das transações são interativas
5.	Mais de 30% das transações são interativas

Características Gerais do Sistema

- **CGS 01 – Comunicação de Dados**
  - Descreve até que ponto a aplicação se comunica diretamente com o processador
- **CGS 02 – Processamento distribuído**
  - Descreve até que ponto a aplicação transfere dados entre seus componentes físicos
- **CGS 03 – Performance**
  - Descreve o grau de influência das características de tempo de resposta e volume de processamento na aplicação

Características Gerais do Sistema

- **CGS 04 – Configuração utilizada**
  - Descreve até que ponto as restrições de recursos computacionais afetam a aplicação
- **CGS 05 – Volume de transações**
  - Descreve o nível segundo o qual a taxa de transações do negócio influencia o desenvolvimento da aplicação
- **CGS 06 – Entrada de Dados On-Line**
  - Descreve os níveis segundo os quais os dados são informados ou recuperados através de transações interativas

Características Gerais do Sistema

- **CGS 07 – Eficiência do usuário final**
  - Descreve a influência de fatores humanos e facilidade de uso da aplicação
- **CGS 08 – Atualização on-line**
  - Descreve os níveis segundo os quais os arquivos lógicos internos são atualizados on-line
- **CGS 09 – Processamento complexo**
  - Descreve os níveis segundo os quais a lógica de processamento afetou o desenvolvimento da aplicação

Características Gerais do Sistema

- **CGS 10 – Reusabilidade**
  - Descreve até que ponto o código da aplicação foi desenvolvido pensando em reuso (em outras aplicações)
- **CGS 11 – Facilidade de instalação**
  - descreve os níveis segundo os quais a conversão de ambientes anteriores influenciou o desenvolvimento da aplicação

Características Gerais do Sistema

- **CGS 12 – Facilidade de operação**
  - Descreve o nível segundo o qual a aplicação atende a aspectos operacionais, tais como inicialização, backup e recuperação
- **CGS 13 – Múltiplos locais**
  - Descreve os níveis segundo os quais a aplicação foi desenvolvida para diferentes ambientes de hardware e software

Características Gerais do Sistema

- **CGS 14 – Facilidade de mudança**
  - Descreve os níveis segundo os quais a aplicação foi desenvolvida para fácil modificação da lógica de processamento
  - ou estrutura de dados

Cálculo do Fator de Ajuste

- Determinados os níveis de influência das 14 características gerais, o fator de ajuste é calculado com a seguinte fórmula:

$$VAF = (TDI \times 0,01) + 0,65$$

- Onde TDI é o somatório dos níveis de influência das CGS
  - Pode chegar a 70 (5x14).

Críticas ao Fator de Ajuste

- As **características gerais do sistema** são muito subjetivas
- A maioria delas está desatualizada
- Apesar de serem subjetivas, podem afetar significativamente a contagem
- Não é apropriado que todas as CGSs possuam o mesmo peso para nível de influência
- Existem requisitos que não são contemplados por nenhuma das 14 CGSs

Exercícios [21]

(SUSEP – ESAF 2010)

São Características Gerais do Sistema (CGS) do fator de ajuste que avaliam a funcionalidade geral da aplicação:

a) Performance, Pontos de transição e Composição on-line.

b) Comunicação de dados, Interface com o usuário e Reusabilidade.

c) Taxa de acertos, Funções de transações e Atualização on-line.

d) Saída de dados on-line, Facilidade de planejamento e Performance.

e) Comunicação de transações, Interação entre programas e Usabilidade

Exercícios [22]

Baseando-se nas Características Gerais do Sistema (CGS), um dos passos para o cálculo do fator de ajuste é:

a) avaliar o impacto de cada uma das 14 CGS no aplicativo que está sendo contado, atribuindo peso de 0 a 5 para cada característica.
b) calcular o nível de influência por meio da multiplicação dos pesos de cada uma das 14 CGS.
c) avaliar as entradas de cada uma das 14 CGS no aplicativo que está sendo contado, atribuindo peso de 0 a 10 para cada característica.
d) avaliar o impacto de cada uma das 14 CGS no aplicativo que está sendo contado, atribuindo peso de 0 a 10 para cada característica.
e) calcular o nível de influência por meio da soma dos pesos da primeira metade das 14 CGS.

Exercícios [23]

(TRT/24 – FCC 2011)
Após a aplicação do fator de ajuste, o total de pontos de função em uma contagem ficou em 110,60. Antes da aplicação do ajuste, os pontos de função brutos estavam em 140,00. Portanto, o somatório dos 14 itens do nível de influência global foi

a) 11.
b) 14.
c) 15.
d) 18.
e) 19.

Exercícios [24]

(TCE/PR – FCC 2011)

O processo de contagem de pontos de função pode ser composto pelos seguintes

passos:

I. Identificação do propósito da contagem para determinar o que se pretende atingir com a contagem que será feita e qual o problema que se pretende resolver com ela.

II. Determinação do tipo de contagem: composta por três tipos de contagem, sendo um deles, o projeto de desenvolvimento, que mede todas as funções que o

projeto entregará e eventuais funções de conversão de dados.

III. Contagem das funções tipo dado, que representam requisitos de armazenamento do usuário, e contagem das funções tipo transação, que representam requisitos de processamento do usuário.

Exercícios [24]

IV. Cálculo do fator de ajuste para representar a influência de requisitos técnicos de qualidade no tamanho do software.
V. Cálculo dos pontos de função ajustados, consistindo basicamente em multiplicar o fator de ajuste pelos pontos de função não ajustados.

Está correto o que se afirma em

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

Exercícios [25]

(TCE/GO – FCC 2009)

Na aplicação da métrica Análise de Pontos por Função, caso haja influência forte em quatro das 14 Características Gerais de Sistema, os pontos ajustados serão

a) 65% dos pontos brutos.
b) 75% dos pontos brutos.
c) 80% dos pontos brutos.
d) 85% dos pontos brutos.
e) 115% dos pontos brutos.

Gabaritos

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

UML

Fernando Pedrosa – fpedrosa@gmail.com

Bibliografia

- **Boch, Jacobson, Rumbaugh;** UML – Guia do Usuário; **Editora:** Elsevier; **Ano**: 2006
- **Martin Fowler;** UML Essencial; **Editora:** Bookman; **Ano**: 2004

Unified Modeling Language

Linguagem de Modelagem Unificada

- Linguagem
  - Usada para expressar e comunicar idéias
  - Não é uma metodologia!
- Modelagem
  - Descrever um sistema em um alto nível de abstração
- Unificada
  - UML se tornou o padrão mundial para modelagem de sistemas – www.omg.org

O que é a UML?

- Linguagem gráfica para especificar, visualizar, construir e documentar os artefatos de software
- Vantagens
  - Usa notação gráfica: mais clara que a linguagem natural (imprecisa) e código (muito detalhado)
  - Ajuda a obter uma visão geral do sistema
  - **Não** é dependente de tecnologia
  - Diminui a fragmentação, aumenta a padronização

Evolução

Industrialização

Padronização

Unificação

Fragmentação

Ano Versão

2011: UML 2.4

2010: UML 2.3

2009: UML 2.2

2003: UML 2.0

2001: UML 1.4

1999: UML 1.3

1997: UML 1.0, 1.1

1996: UML 0.9 & 0.91

1995: Unified Method 0.8

OOSE (Jacobson)

Outros métodos

Booch (OOAD)

OMT (Rumbaugh)

UML 2.2

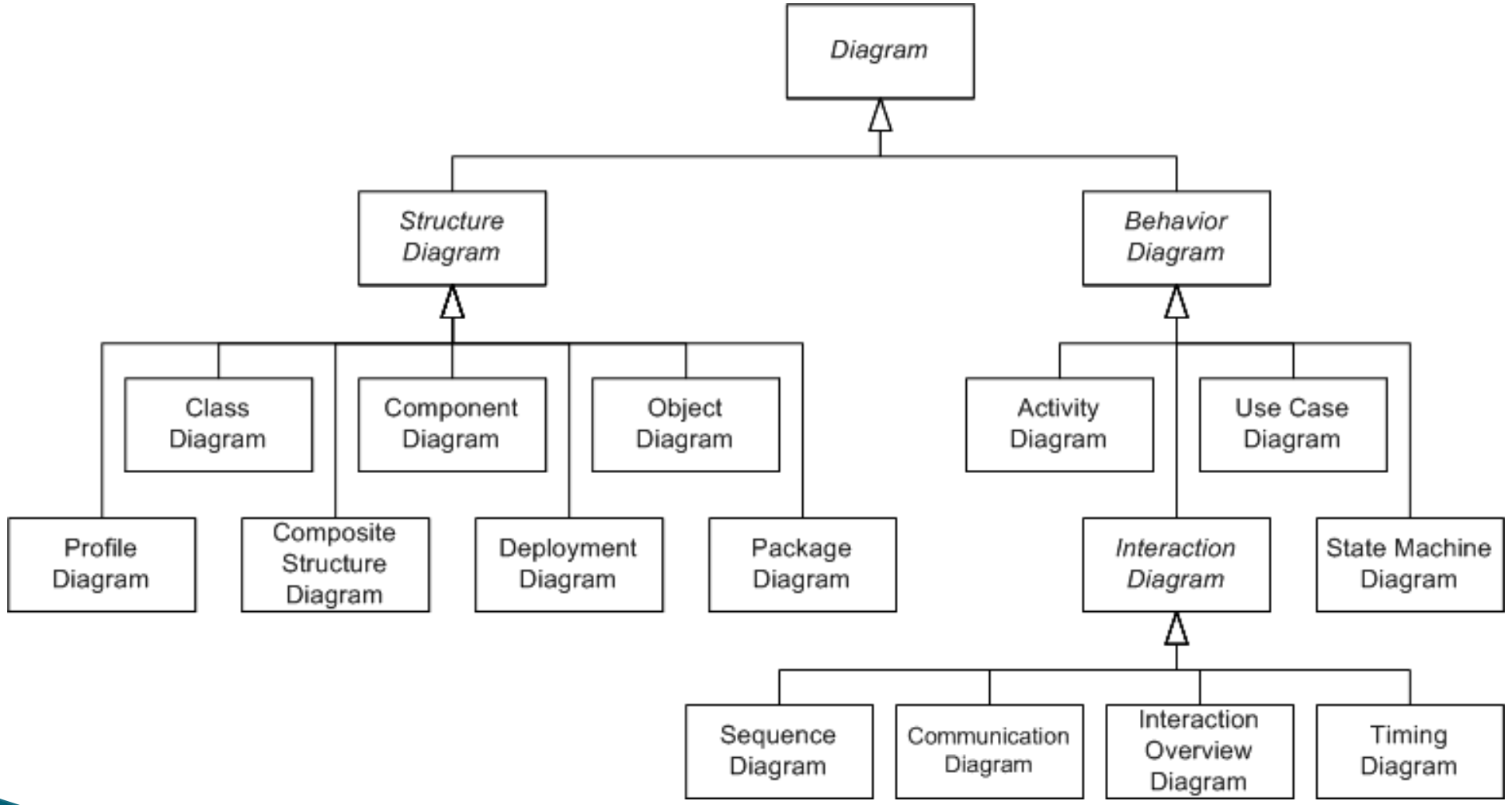


Diagram
Structure Diagram
Behavior Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
Use Case Diagram
Profile Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Interaction Diagram
State Machine Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram
Timing Diagram

Tipos de Diagramas

- Diagramas estruturais
  - Mostram a estrutura estática do sistema e suas partes em diferentes níveis de abstração e como elas se relacionam
  - Não utilizam conceitos relacionados ao tempo
- Diagramas comportamentais
  - Mostram a natureza dinâmica dos objetos do sistema, que pode ser descrita como uma série de mudanças no sistema com o passar do tempo

Exercícios [1]

(Governo do ES – CESPE 2009)

[78] UML é um método para desenvolvimento de software que foi proposto para ser aplicado à análise e projeto de software orientados a objetos.

(EMBASA – CESPE 2009)

[94] Os diagramas em UML podem ser estáticos ou dinâmicos. O diagrama de classes é um exemplo de um diagrama dinâmico.

(SERPRO – CESPE 2008)

[101] UML (universal modelling language) é uma linguagem de modelagem proprietária que pode ser utilizada no desenvolvimento de sistemas de maneira intuitiva para visualização de objetos.

Exercícios [1]

(CGU – ESAF 2008)

[31] – A linguagem de Modelagem Unificada (UML) emergiu como notação de diagramação de padrão, de fato e de direito, para a modelagem orientada a objetos. Desta forma, a sentença que conceitua apropriadamente a UML, segundo o OMG–Object Management Group, é

a) um método para especificar e modelar os artefatos dos sistemas.

b) um processo de especificação e modelagem de sistemas orientados a objeto.

c) uma linguagem para implementar os conceitos da orientação a objetos.

d) uma linguagem visual para especificar, construir e documentar os artefatos dos sistemas.

e) um método comum para a representação da orientação a objetos.

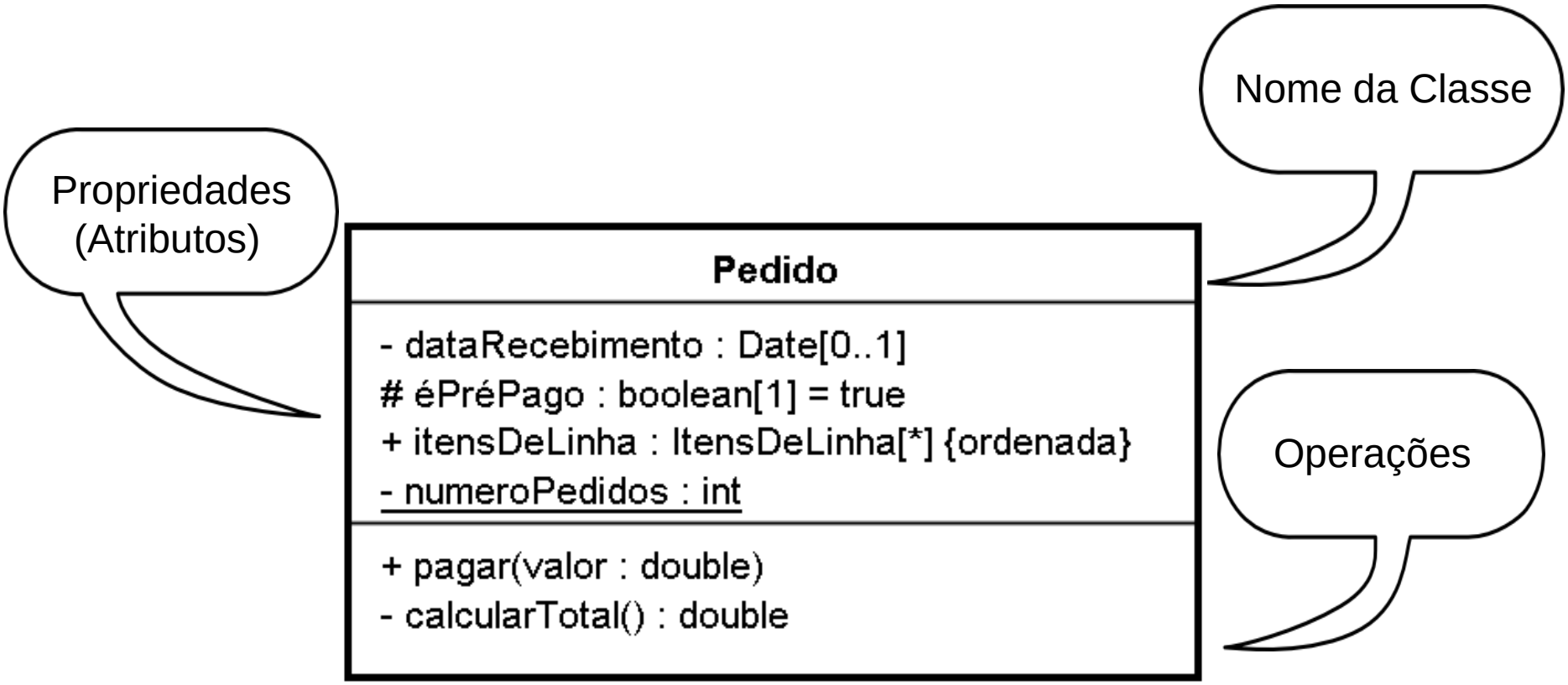
Diagramas Estruturais (estáticos)

- **Diagrama de Classes**
- Diagrama de Objetos
- Diagrama de Componentes
- Diagrama de Pacotes
- Diagrama de Implantação
- Diagrama de Estrutura Composta
- Diagrama de Perfis (UML 2.2)

Diagrama de Classes

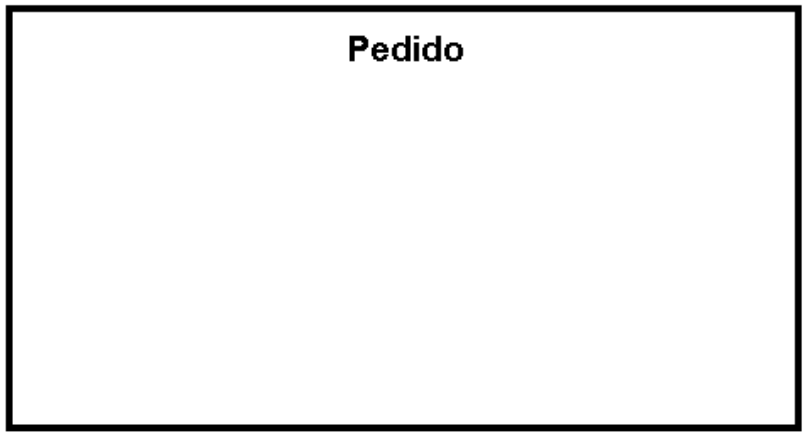
- É um diagrama estático da UML que reúne os elementos mais importantes de um sistema orientado a objetos
- Exibe um conjunto de classes, interfaces e seus relacionamentos
- As classes especificam tanto as propriedades quanto os comportamentos dos objetos

Estrutura da Classe

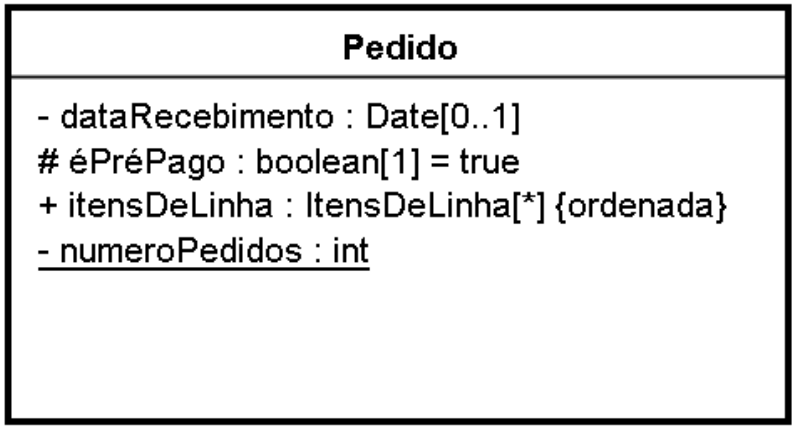


Três formas de representação

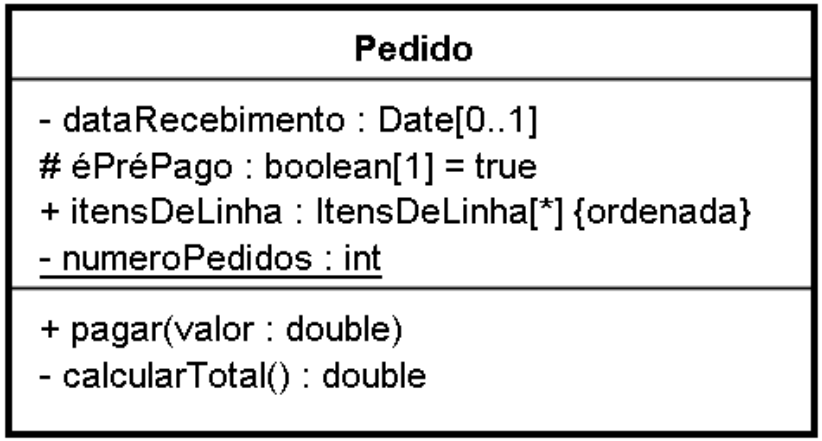
Nome

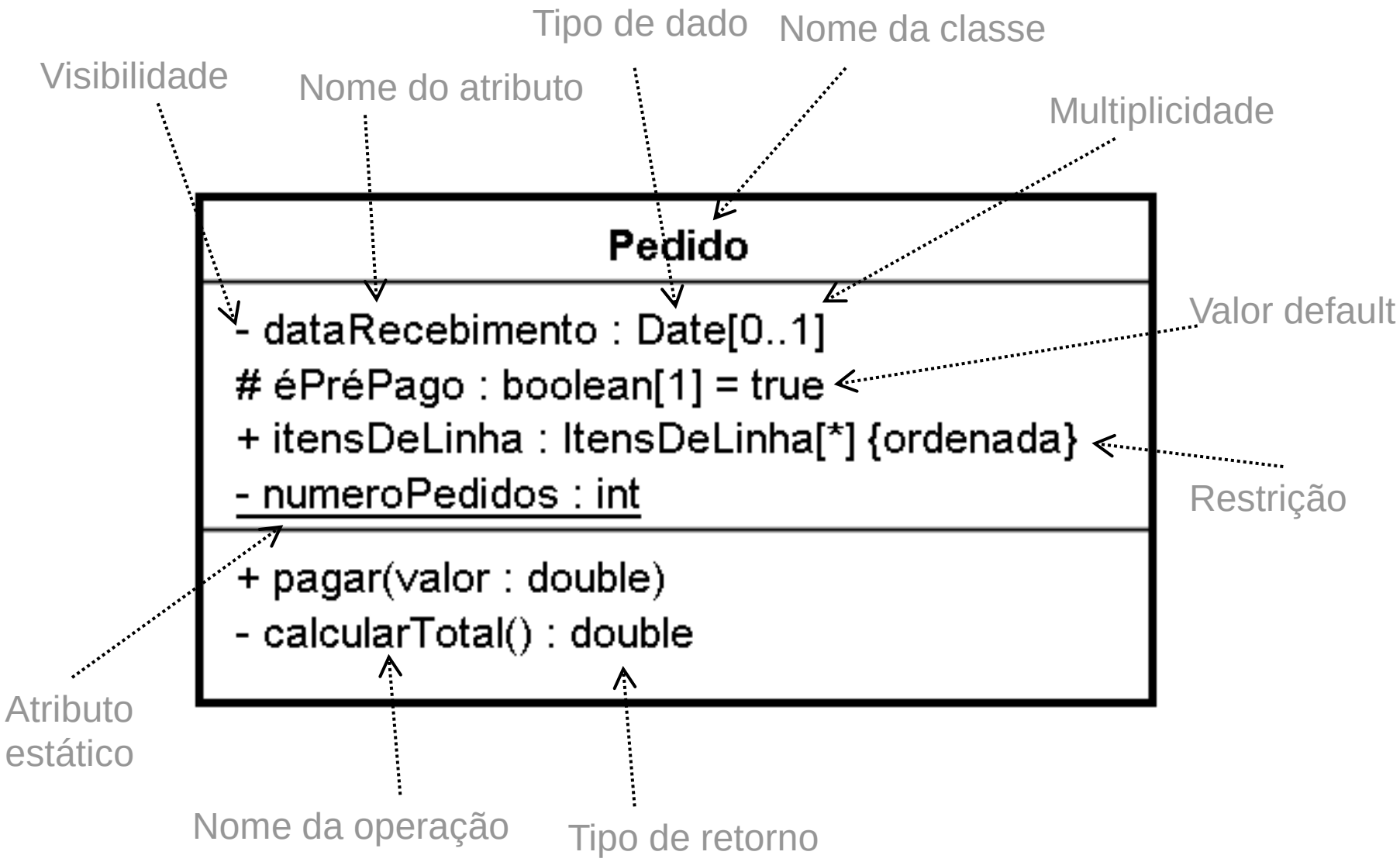


Nome + Atributos



Nome + Atributos + Operações




Tipo de dado
Nome da classe
Visibilidade
Nome do atributo
Multiplicidade
Pedido
- dataRecebimento : Date[0..1]
ésPréPago : boolean[1] = true
+ itensDeLinha : ItensDeLinha[*] {ordenada}
- numeroPedidos : int
+ pagar(valor : double)
- calcularTotal() : double
Valor default
Restrição
Atributo estático
Nome da operação
Tipo de retorno

Atributos

Notação completa:

Visibilidade nome: tipo [multiplicidade] = valor_default {lista de restrições}

Nome: corresponde ao nome do atributo
Tipo: domínio do atributo
Multiplicidade: indicação de quantos objetos podem preencher a propriedade [min..max]
Valor Default: valor do atributo, caso ele seja omitido no momento da criação
Restrição: permite indicar propriedades adicionais. {readOnly}, {ordered}, {unique}, etc.

Atributos – Escopo

- As propriedades (atributos) podem ter dois tipos de escopo
  - **Escopo de instância**: cada objeto tem o seu próprio valor para o atributo. É o escopo *default* da UML.
  - **Escopo de classe (estático)**: o valor do atributo é comum a todos os objetos daquela classe. Para denotar este escopo o atributo deve ser sublinhado.

Operações

Notação completa:

Visibilidade nome (lista de parâmetros): tipo-de-retorno {lista restrições}

Nome: corresponde ao nome da operação
Lista de parâmetros: são os parâmetros da operação.
Tipo de retorno: o tipo de dado retornado pela operação
Restrição: permite indicar propriedades adicionais.
ex: {query}.

Operações abstratas e estáticas

- Operações abstratas, ou seja, que não têm uma implementação específica, devem ser escritas em *itálico*
- Operações estáticas devem ser escritas com fonte sublinhada

Modificadores de Visibilidade

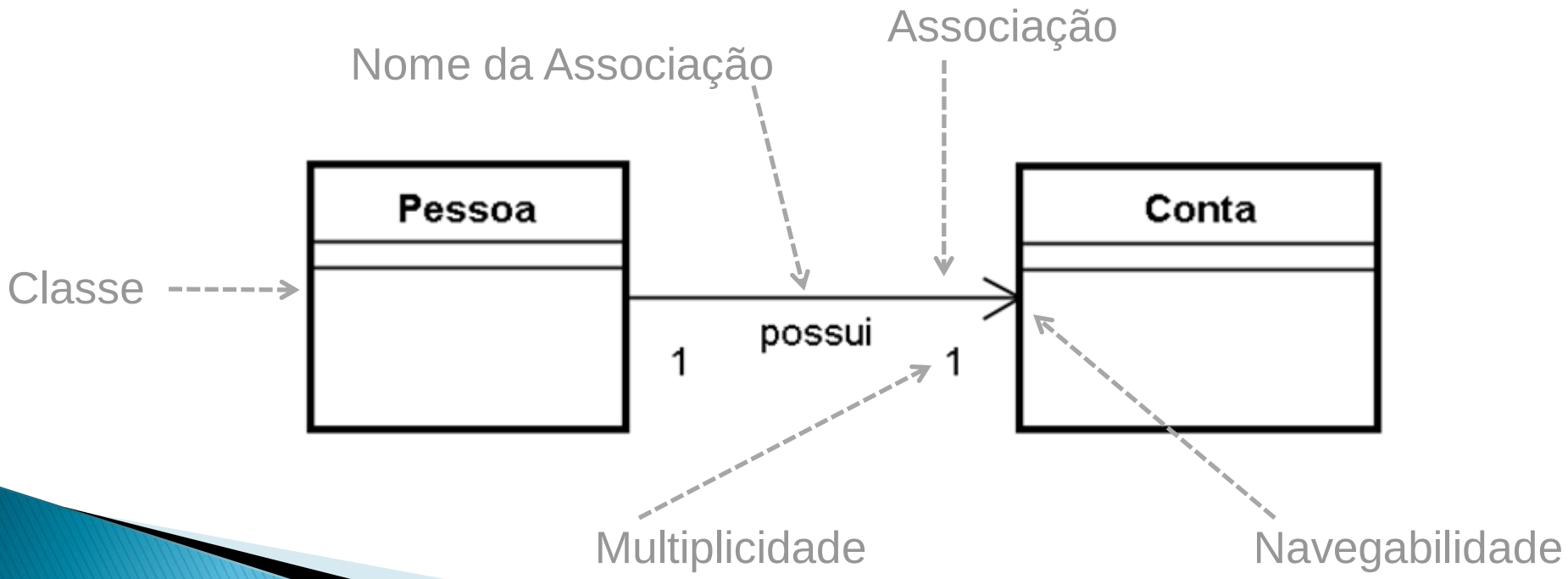
- Público (+)
  - O elemento é visível por qualquer classe
- Protegido (#)
  - O elemento é visível na própria classe e pelas subclasses da classe
- Pacote (~)
  - O elemento é visível apenas pela própria classe ou dentro do pacote onde a classe está localizada
- Privado (-)
  - O elemento é visível apenas pela própria classe

Relacionamentos

- Relacionamentos ligam classes entre si, criando relações lógicas
- Podem ser de:
  - **Associação**
    - Simples
    - Agregação
    - Composição
  - **Generalização**
  - **Dependência**
  - **Realização**

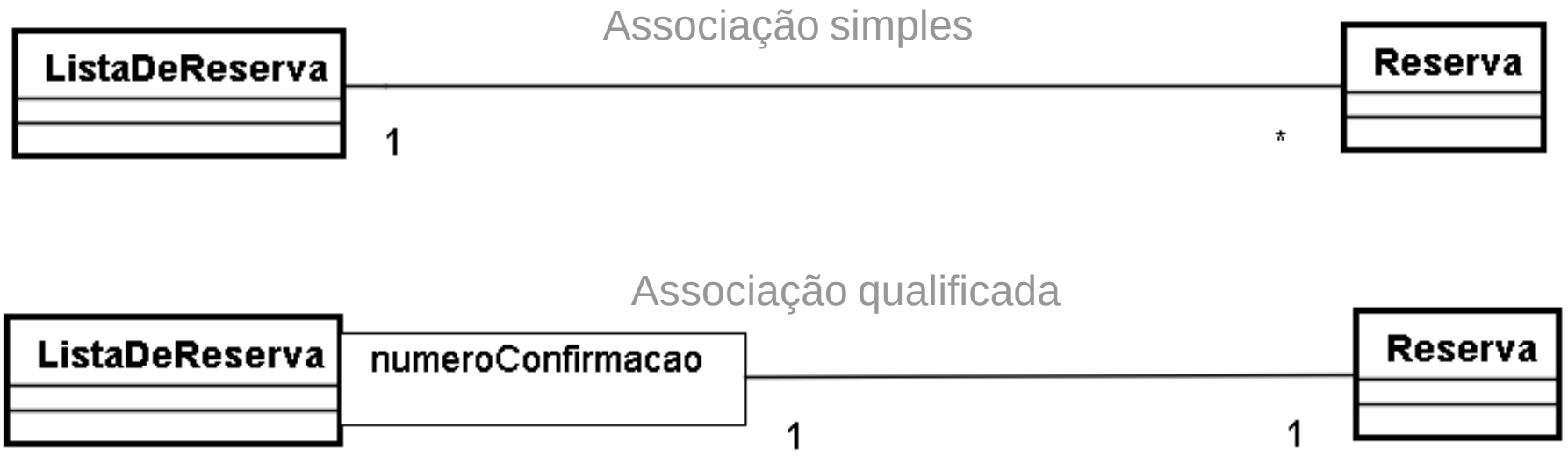
Associação simples

- Indica que objetos de um elemento estão ligados a objetos de outro elemento
- A navegabilidade pode ser unidirecional ou bidirecional



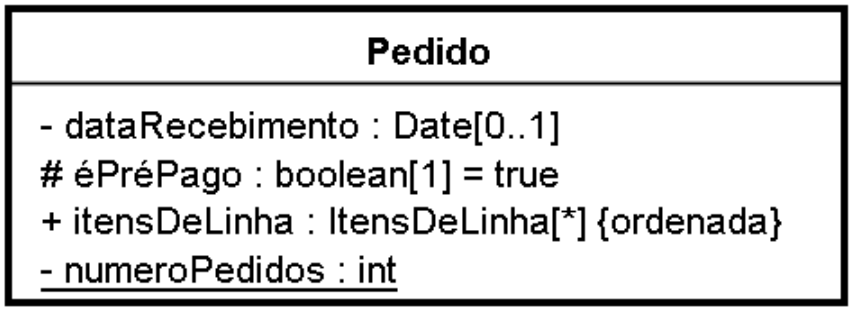
Associação qualificada

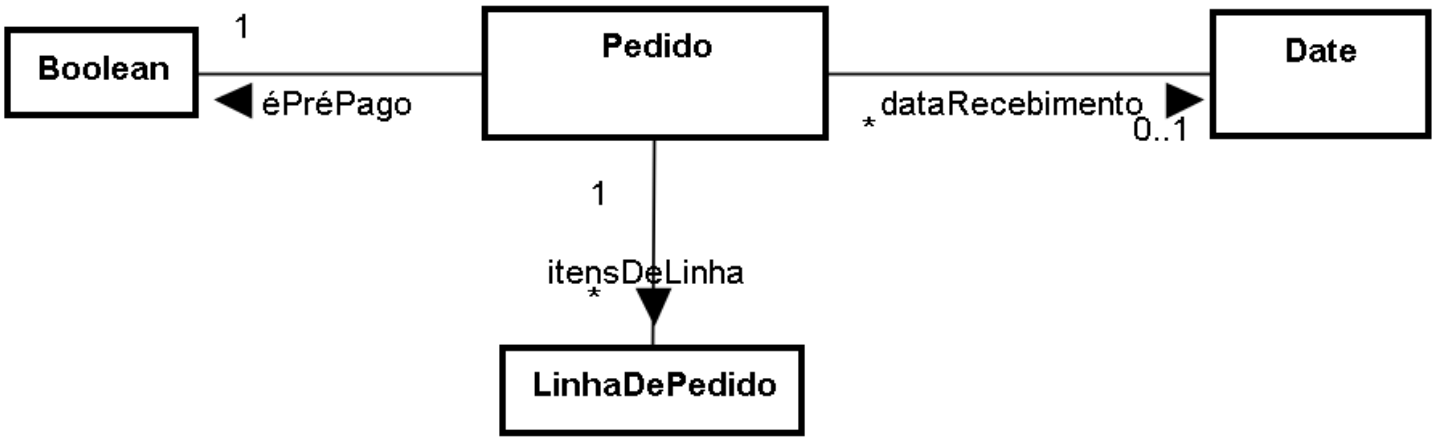
- Um qualificador de associação é um atributo do elemento-alvo capaz de identificar uma instância dentre as demais



Associação x Atributos

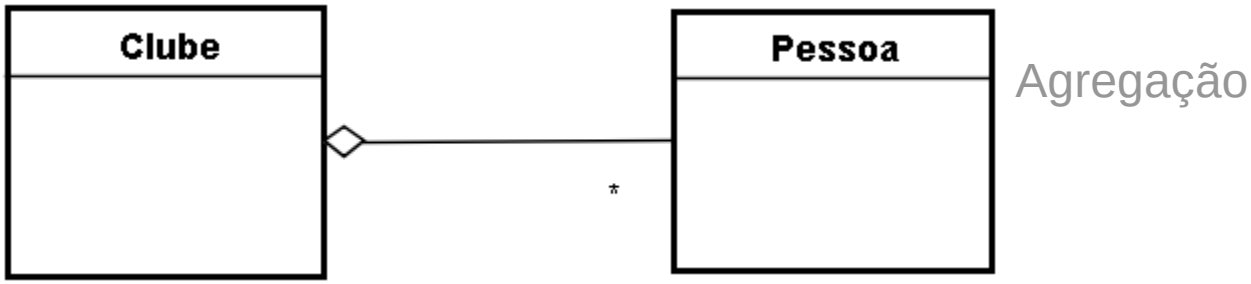
Uma associação pode mostrar as mesmas informações que um atributo





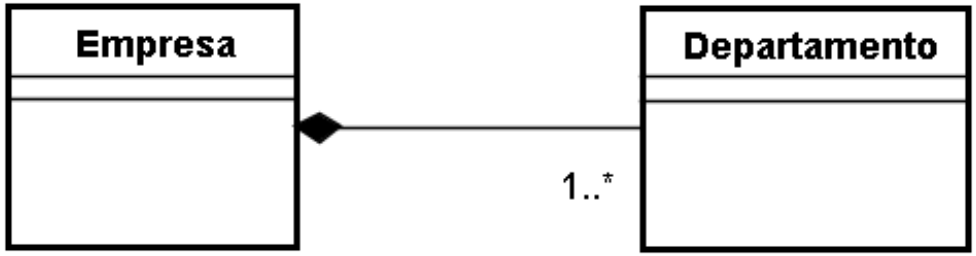
Agregação e Composição

- Relacionamentos “todo-parte”
- **Agregação**: a parte existe sem o todo
- **Composição**: o todo controla o ciclo de vida da parte, e ela não pode ser compartilhada em outros relacionamentos



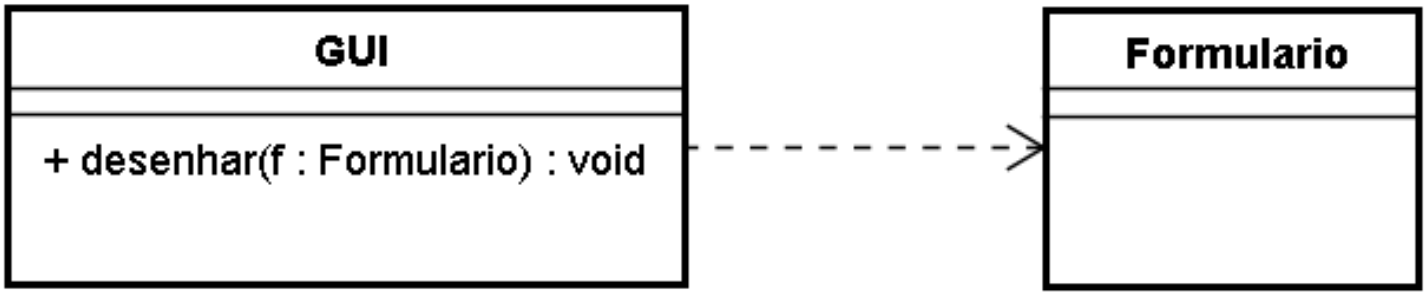
Agregação

Composição



Dependência

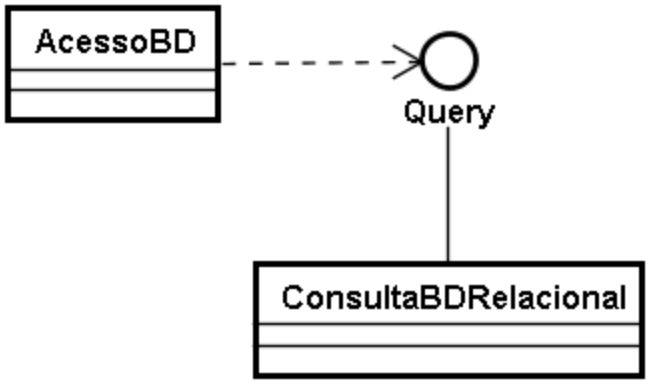
- Indica que mudança em um elemento pode causar mudanças no outro (uso)




```
public class GUI {

        public void desenhar(Formulario f) {
                f.pintarBotao();
                f.pintarMenu();
                f.pintarJanelas();
                ...
        }
}
```

Dependência

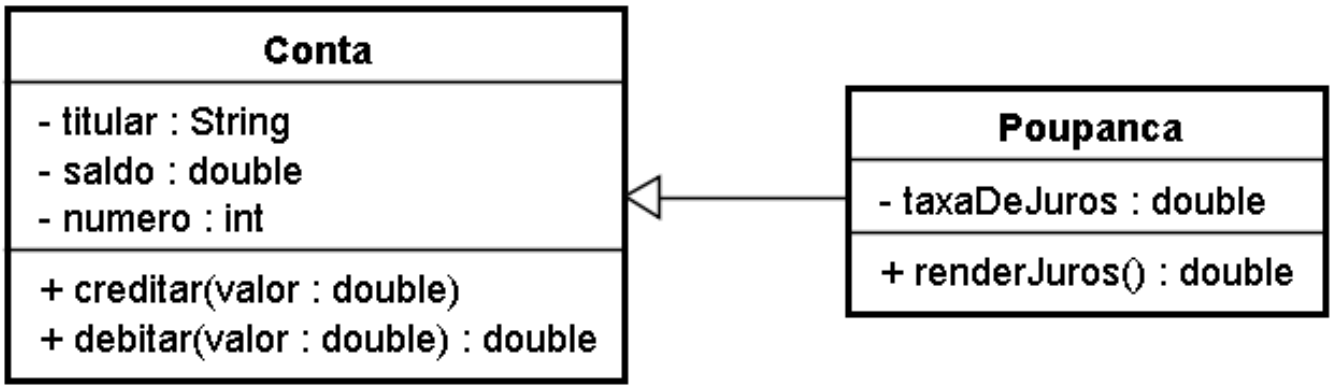
- Pode ocorrer, também, entre uma classe e uma interface




```
public class AcessoBD {
        private Conexao conexao;

        public void consultar(Query query) {
                query.iniciarConsulta();
                ...
        }
}
```

Generalização

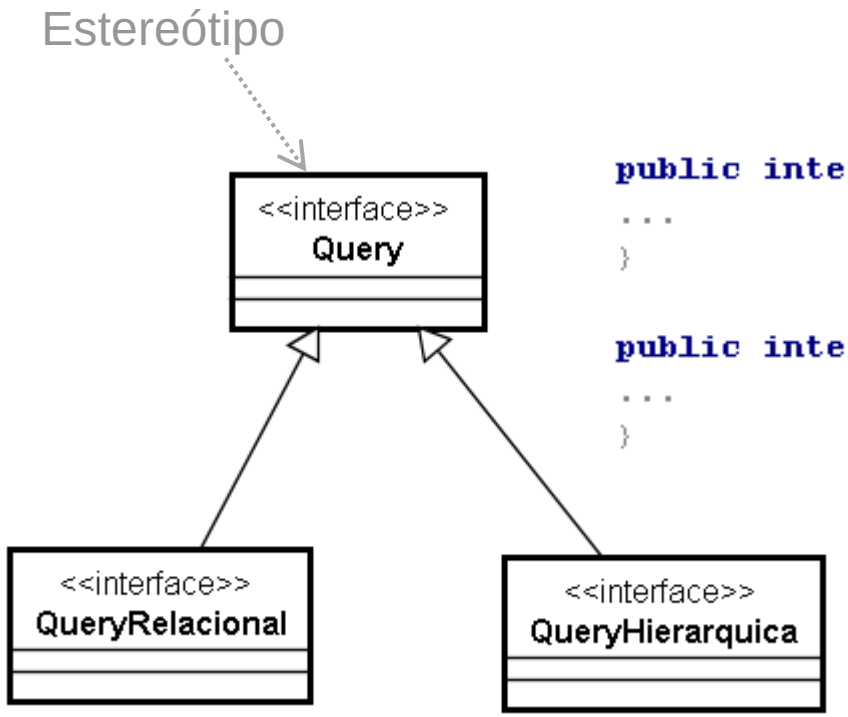
- Relacionamento “é um tipo de”




```
public class Conta {
        private String titular;
        private double saldo;
        private int numero;
        public void creditar(double valor) {...}
        public double debitar (double valor) {...}
}

public class Poupanca extends Conta {
        private double taxaDeJuros;
        public double renderJuros () {...}
}
```

Generalização

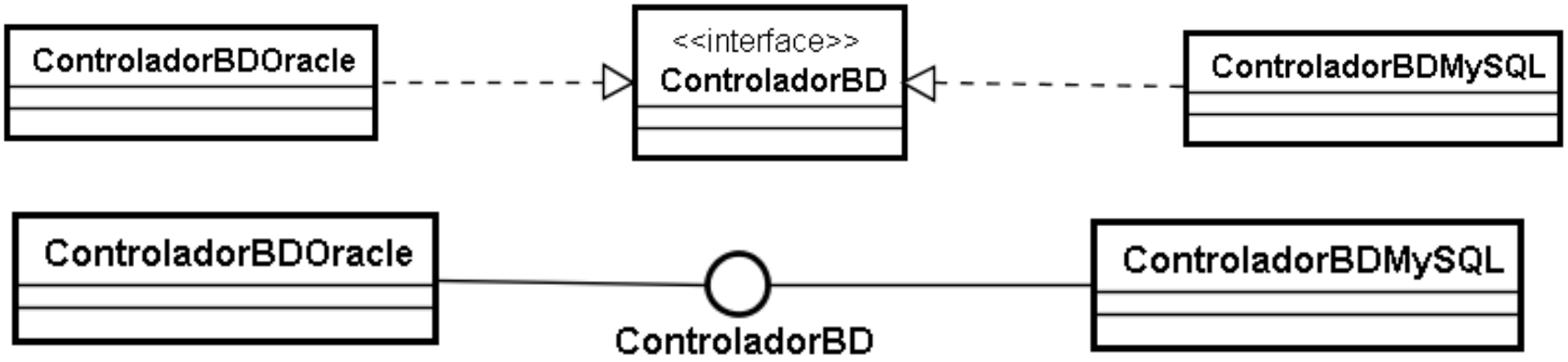
- Pode ocorrer, também, entre interfaces




```
public interface QueryRelacional extends Query {
...
}

public interface QueryHierarquica extends Query {
...
}
```

Realização (Interfaces)

- Há várias notações para realizações

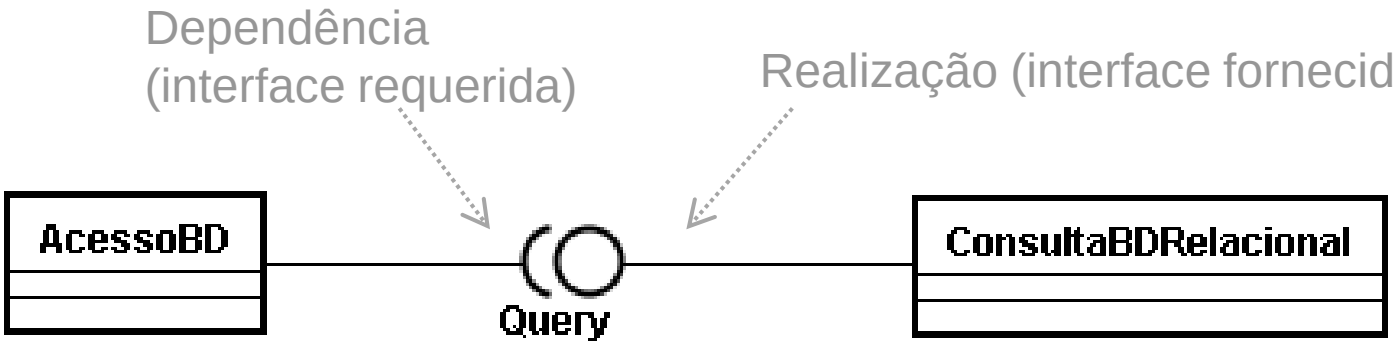



```
public interface ControladorBD {

        public void insert(String SQL);
        public Object select(String SQL);
        ...
}
```

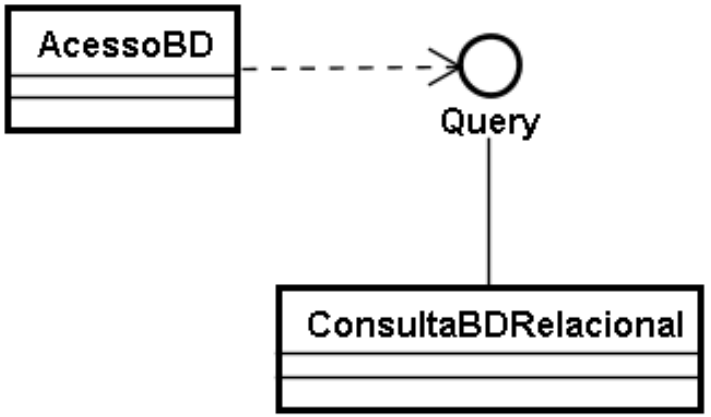
```
public class ControladorOracle
        implements ControladorBD {

        public void insert(String SQL) {
                ...
        }
        public Object select(String SQL) {
                ...
        }
}
```

Realização (Interfaces)

- A notação "bola-soquete" (UML 2.0) é utilizada para modelar uma dependência e uma realização entre duas classes e uma interface



Duas representações equivalentes entre si (semanticamente iguais)



Exercícios [2]

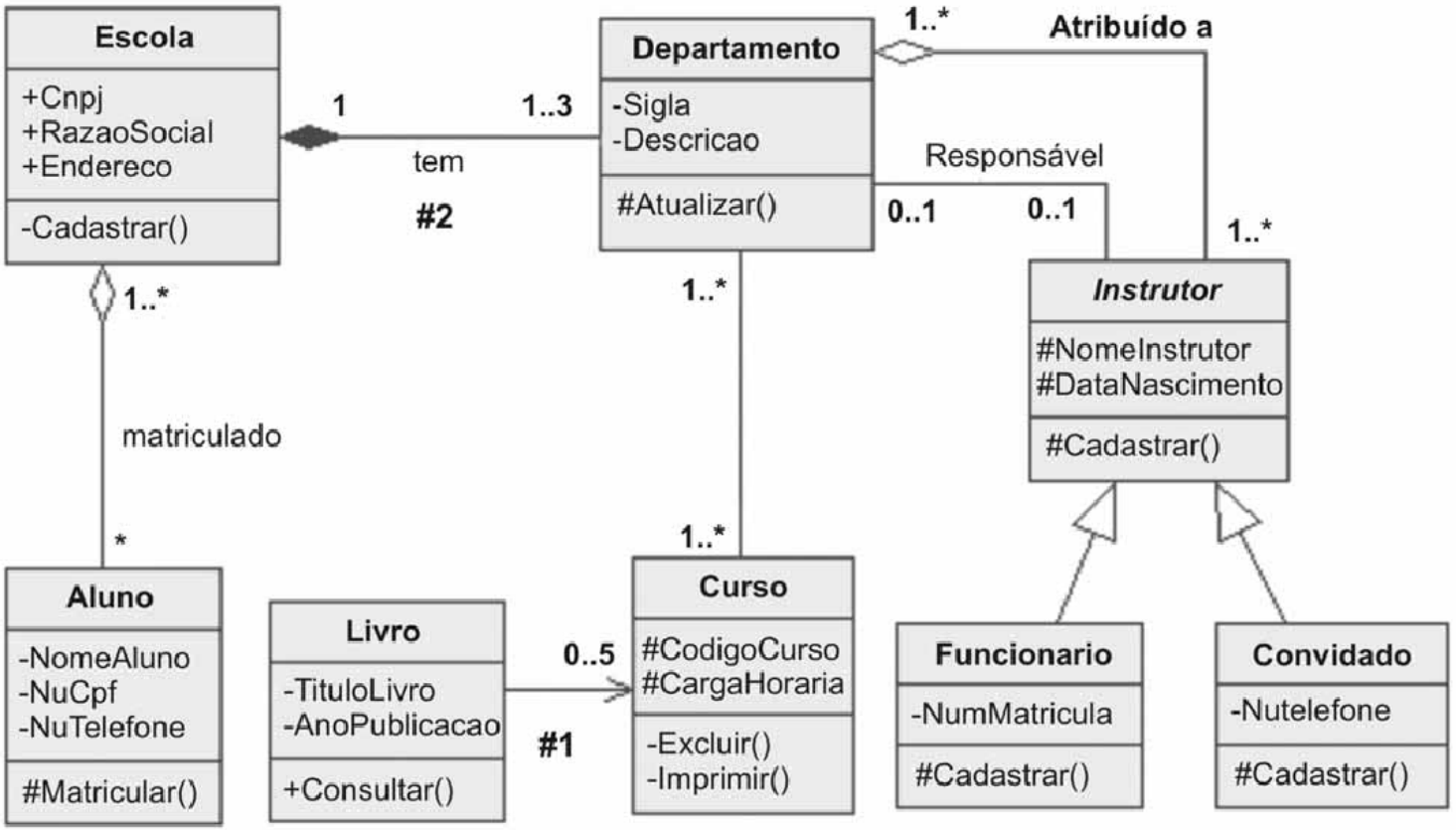
(MPE/AM – CESPE 2008)

[88] Na UML, uma agregação é um relacionamento que estabelece que uma classe define objetos que são parte de um objeto definido por outra classe.

[89] Em um diagrama de classes UML, uma associação entre classes pode ser documentada em termos da multiplicidade da associação.

Exercícios [2]

(TCU – CESPE 2009)



Exercícios [2]

[108] O método #Cadastrar() da classe Instrutor tem visibilidade do modo protegido tal que somente a classe possuidora Instrutor pode utilizá-lo.

[109] Na associação do tipo agregação composta identificado por #2, uma instância da classe Departamento pertence exclusivamente a uma única instância composta em Escola, e um objeto da classe Escola pode relacionar-se com, no máximo, três objetos da classe Departamento.

[110] Instrutor é uma superclasse abstrata; assim, o método #Cadastrar() oferece uma assinatura, que, no entanto, está incompleta, devendo ser implementada pelos métodos de mesmo nome nas suas classes-filhas.

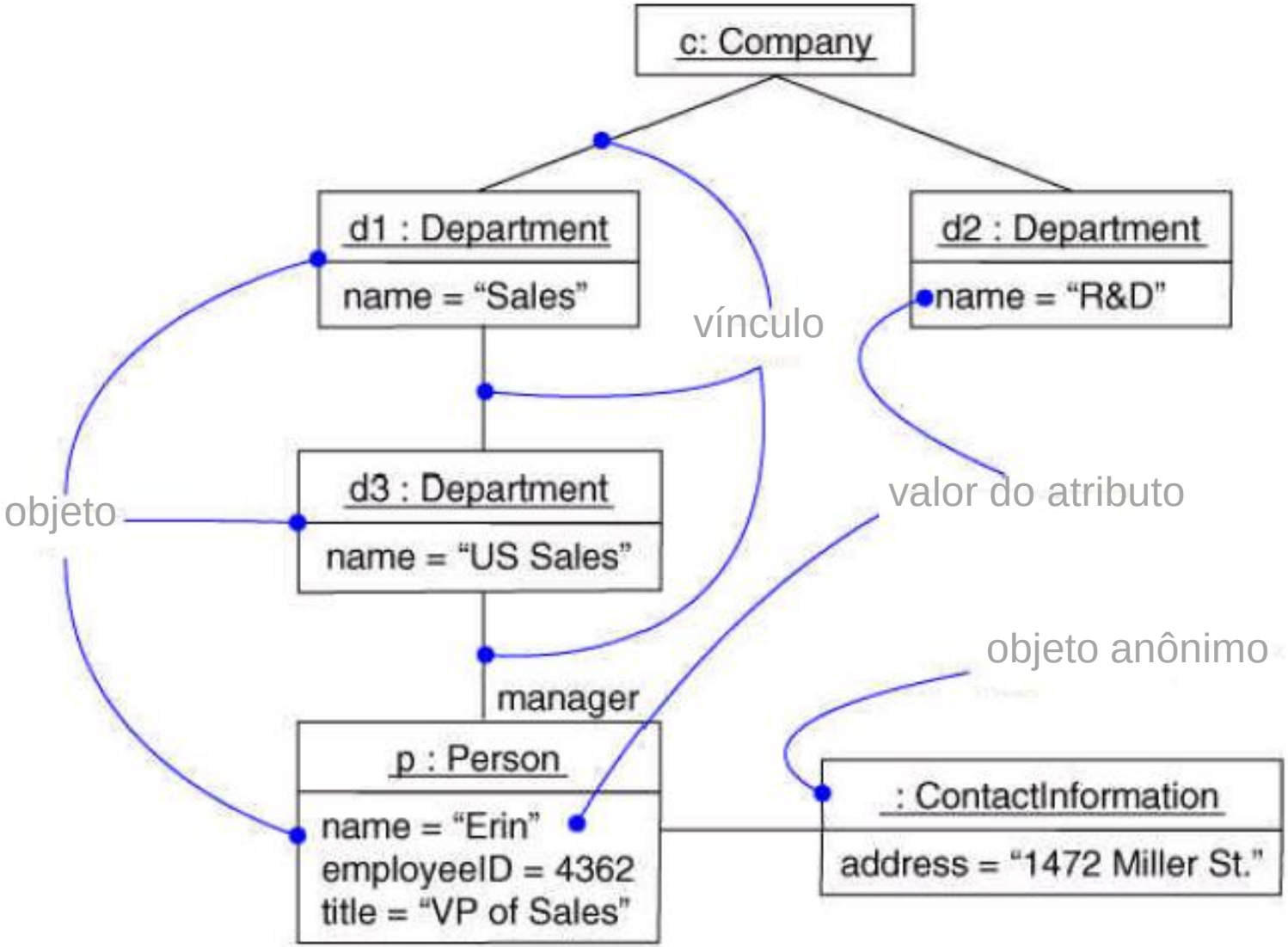
Diagramas Estruturais (estáticos)

- Diagrama de Classes
- **Diagrama de Objetos**
- Diagrama de Componentes
- Diagrama de Pacotes
- Diagrama de Implantação
- Diagrama de Estrutura Composta
- Diagrama de Perfis (UML 2.2)

Diagrama de Objetos

- O diagrama de objetos representa uma fotografia do sistema em um dado momento
- Mostra os vínculos entre os objetos conforme estes interagem e os valores dos seus atributos
- Pode ser visto como uma “instância” do diagrama de classe

Diagrama de Objetos



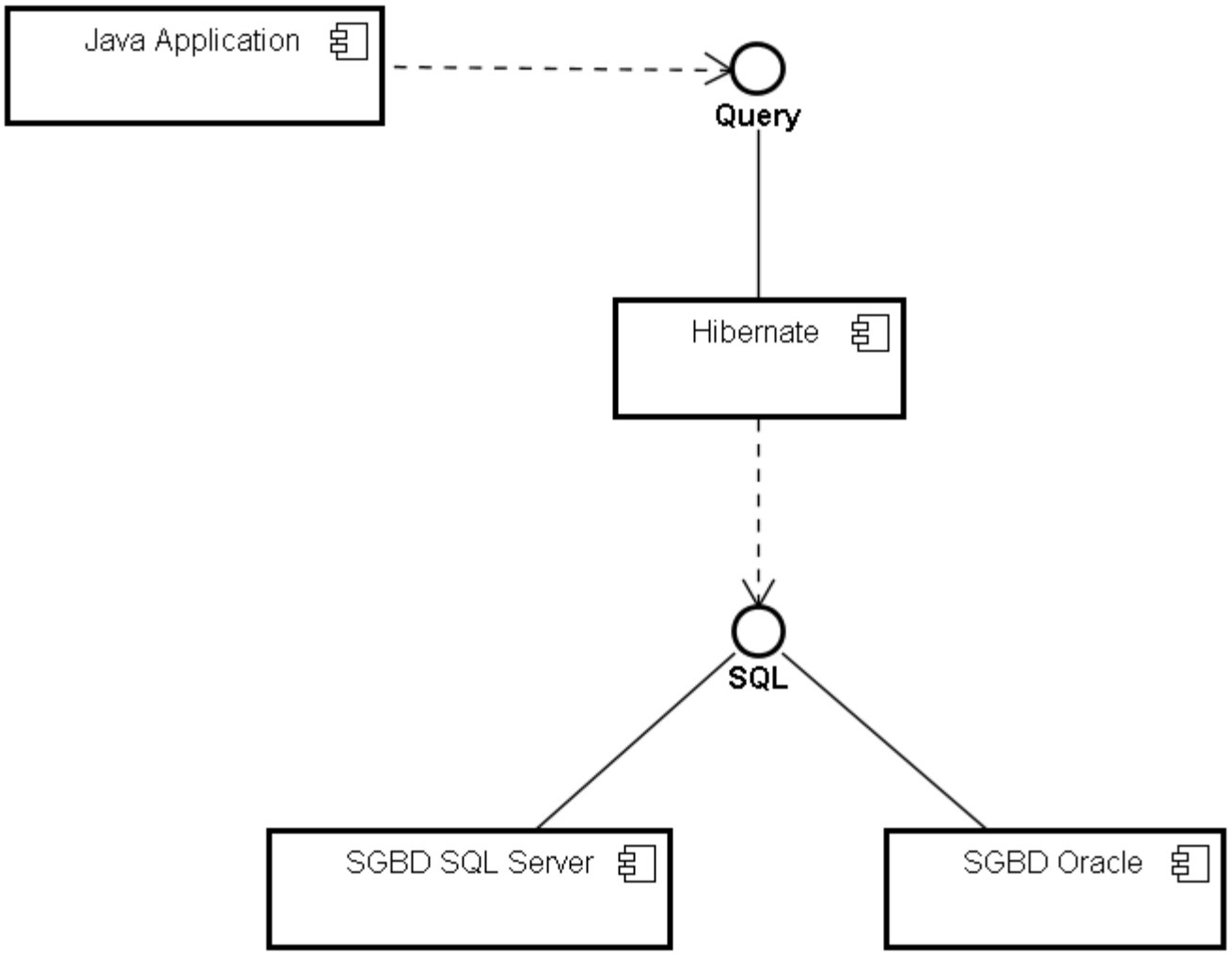
Diagramas Estruturais (estáticos)

- Diagrama de Classes
- Diagrama de Objetos
- **Diagrama de Componentes**
- Diagrama de Pacotes
- Diagrama de Implantação
- Diagrama de Estrutura Composta
- Diagrama de Perfis (UML 2.2)

Diagrama de Componentes

- Modela o sistema em termos de componentes e seus relacionamentos através de interfaces
- Decompõe o sistema em subsistemas que detalham a estrutura interna
- Alguns componentes existem em tempo de ligação, outros em tempo de execução

Diagrama de Componentes



Diagramas Estruturais (estáticos)

- Diagrama de Classes
- Diagrama de Objetos
- Diagrama de Componentes
- **Diagrama de Pacotes**
- Diagrama de Implantação
- Diagrama de Estrutura Composta
- Diagrama de Perfis (UML 2.2)

Diagrama de Pacotes

- Pacotes são estruturas que permitem agrupar qualquer construção da UML em estruturas de alto nível
- Pode mostrar:
  - Pacotes e suas dependências
  - Interfaces entre os pacotes
  - Generalizações entre pacotes

Diagrama de Pacotes

- Duas representações possíveis

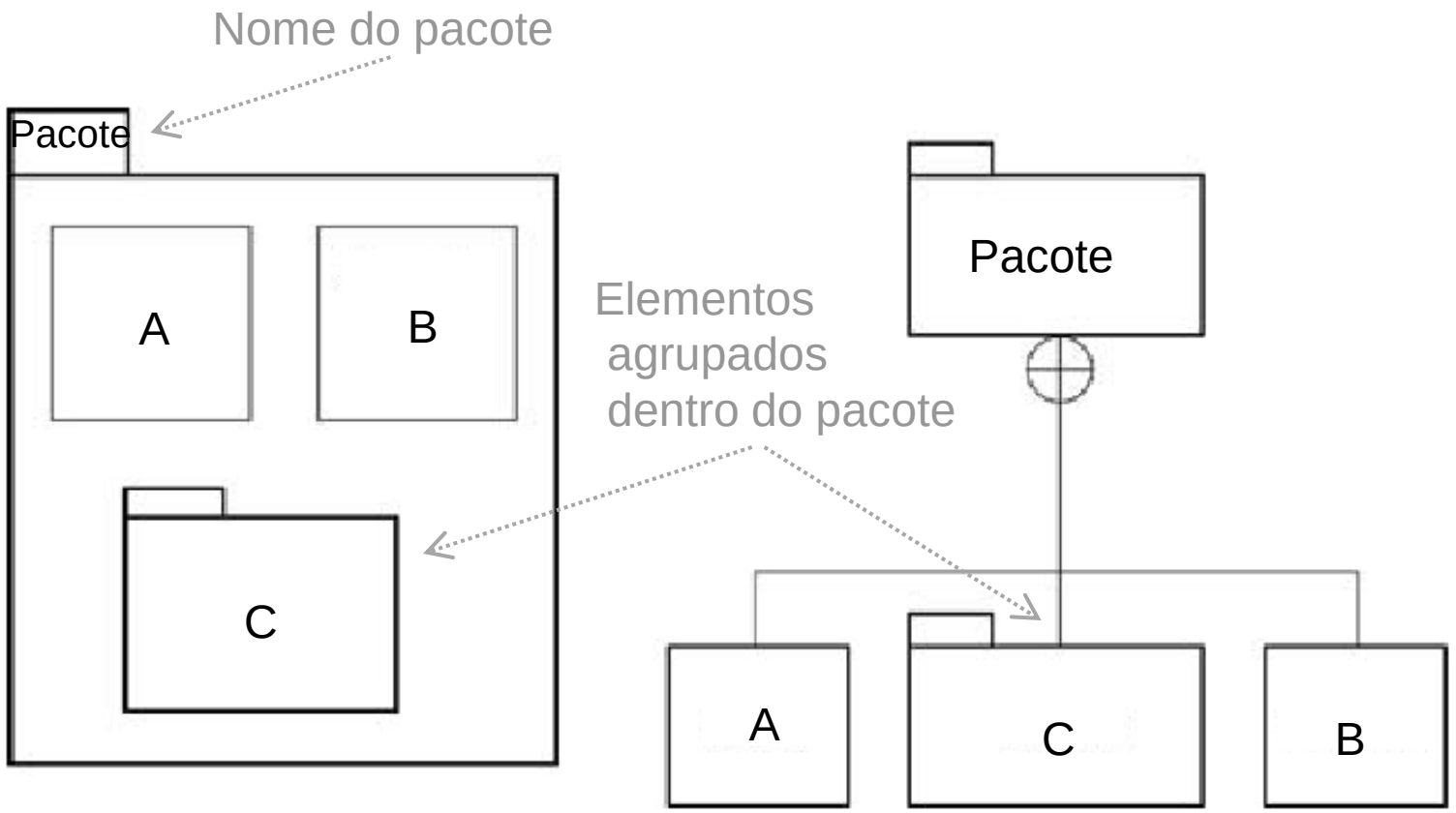
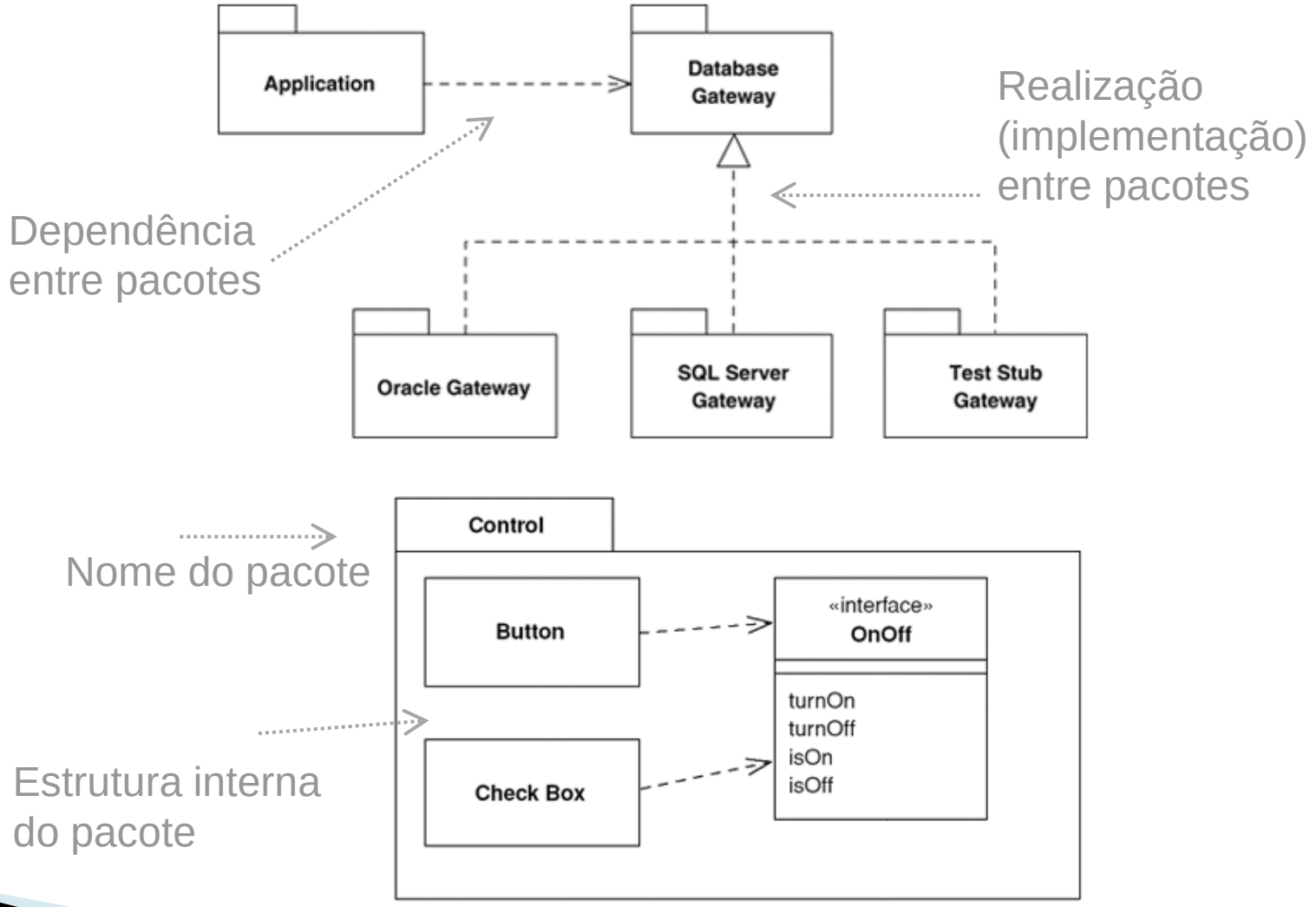


Diagrama de Pacotes



Exercícios [3]

(UNIPAMPA – CESPE 2010)

[106] O diagrama de objetos está amplamente associado ao diagrama de classes, sendo que o primeiro consiste em uma instância do segundo, em determinado momento da execução, ou seja, um diagrama de objetos descreve os objetos, os métodos, os atributos e seus valores, além dos vínculos entre os objetos, sendo ambos diagramas estruturais.

(TRE/TO – CESPE 2007)

[40] Um diagrama de componentes permite mostrar componentes de um sistema e as dependências entre eles. As dependências entre os componentes podem ser, por exemplo, dependências de compilação ou de comunicação.

Exercícios [3]

(EMBASA – CESPE 2009)

[96] O objetivo principal de um diagrama de pacotes é agrupar os pacotes em classes. Esse tipo de diagrama pode usar dependências.

(TJ/PE – FCC 2007)

[40] Diagrama de Pacotes descreve os pacotes ou pedaços do sistema, como o sistema é dividido em agrupamentos lógicos e mostra as dependências entre estes.

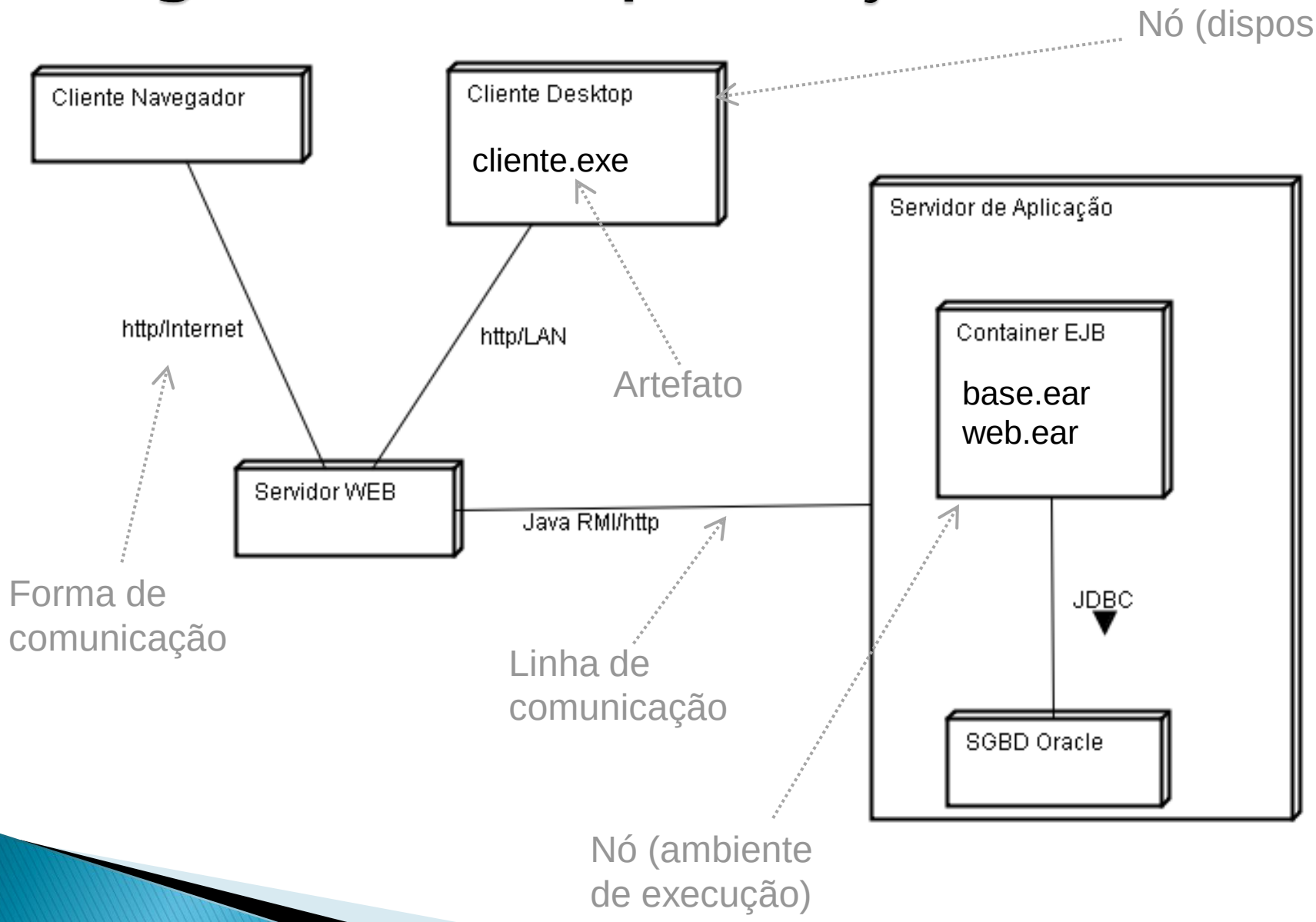
Diagramas Estruturais (estáticos)

- Diagrama de Classes
- Diagrama de Objetos
- Diagrama de Componentes
- Diagrama de Pacotes
- **Diagrama de Implantação**
- Diagrama de Estrutura Composta
- Diagrama de Perfis (UML 2.2)

Diagrama de Implantação

- Modela a configuração física do sistema, revelando que pedaços de software rodam em que equipamentos de hardware
- Inclui
  - Nós
    - Dispositivos (Hardware)
    - Ambientes de Execução
  - Artefatos
    - Código fonte, Código binário
    - Executáveis, etc.

Diagrama de Implantação



Exercícios [4]

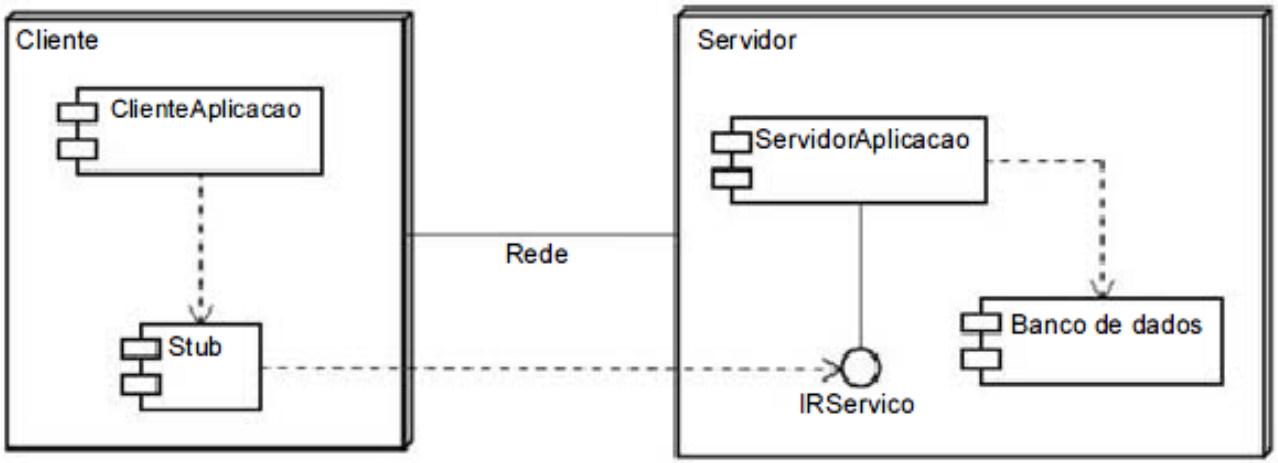


Figura III

(STJ – CESPE 2008)

[54] No diagrama da figura III, há dois nós interligados, que representam duas unidades computacionais; há cinco componentes distribuídos entre os nós; um destes implementa uma interface e um outro depende dessa interface; ClienteAplicacao depende de Stub; ServidorAplicacao depende de Banco de dados.

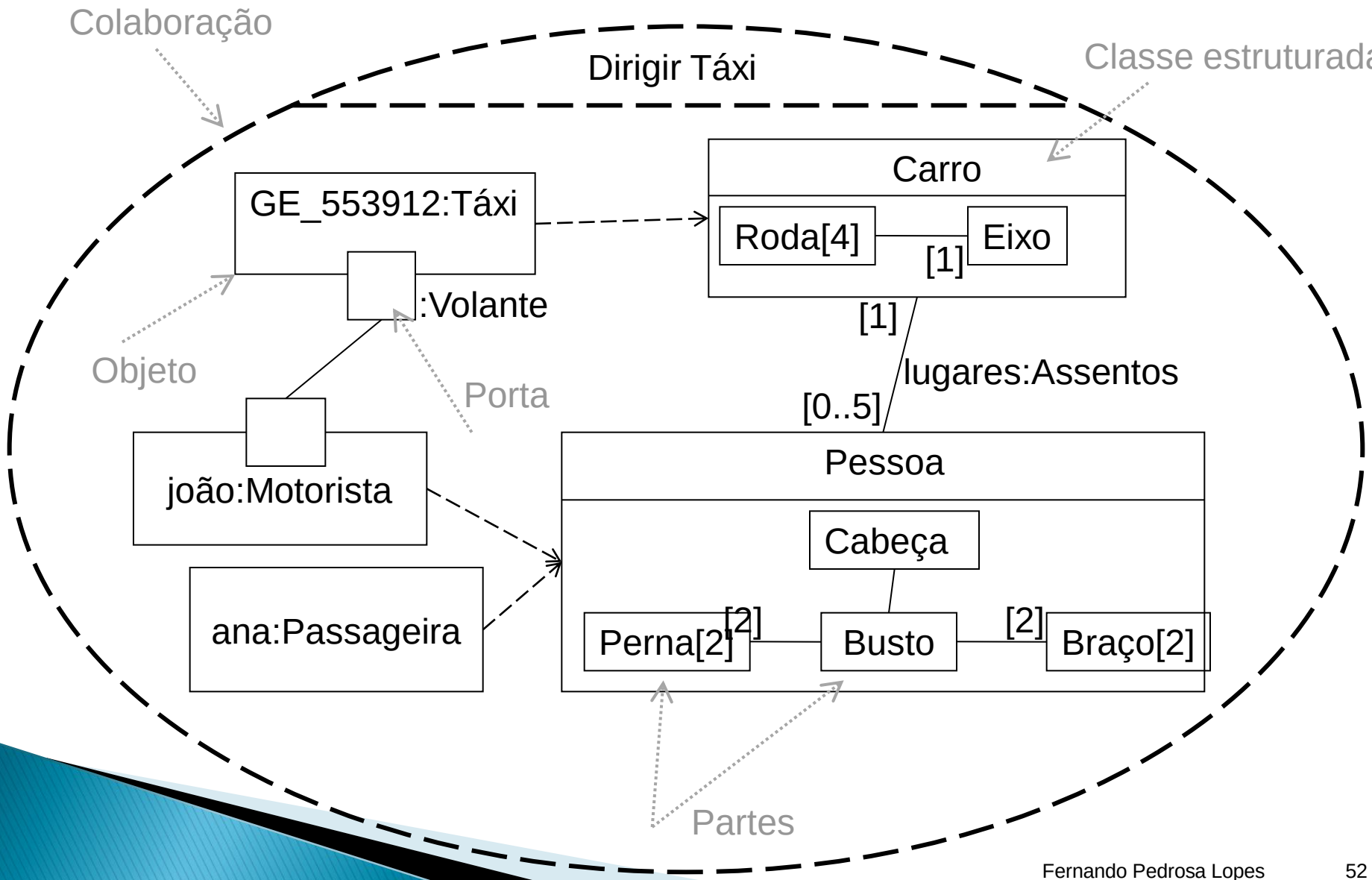
Diagramas Estruturais (estáticos)

- Diagrama de Classes
- Diagrama de Objetos
- Diagrama de Componentes
- Diagrama de Pacotes
- Diagrama de Implantação
- **Diagrama de Estrutura Composta**
- Diagrama de Perfis (UML 2.2)

Diagrama de Estrutura Composta

- É utilizado para modelar colaborações entre interfaces, objetos ou classes
- Pode ser usado para descrever
  - Estruturas de partes interconectadas
  - Estruturas de instâncias interconectadas
- **Parte**: representa o conjunto de uma ou mais instâncias contidas em outro elemento
- **Porta**: ponto de interação entre os elementos

Diagrama de Estrutura Composta



Exercícios [5]

(UNIPAMPA - CESPE 2010)

[105] Na UML 2.0, o diagrama de estrutura composta (composite structure diagram) descreve a estrutura interna de um classificador modelando as colaborações, no qual uma colaboração descreve uma visão de um conjunto de instâncias que cooperam entre si para executar uma função específica entre instâncias de classes, objetos ou interfaces.

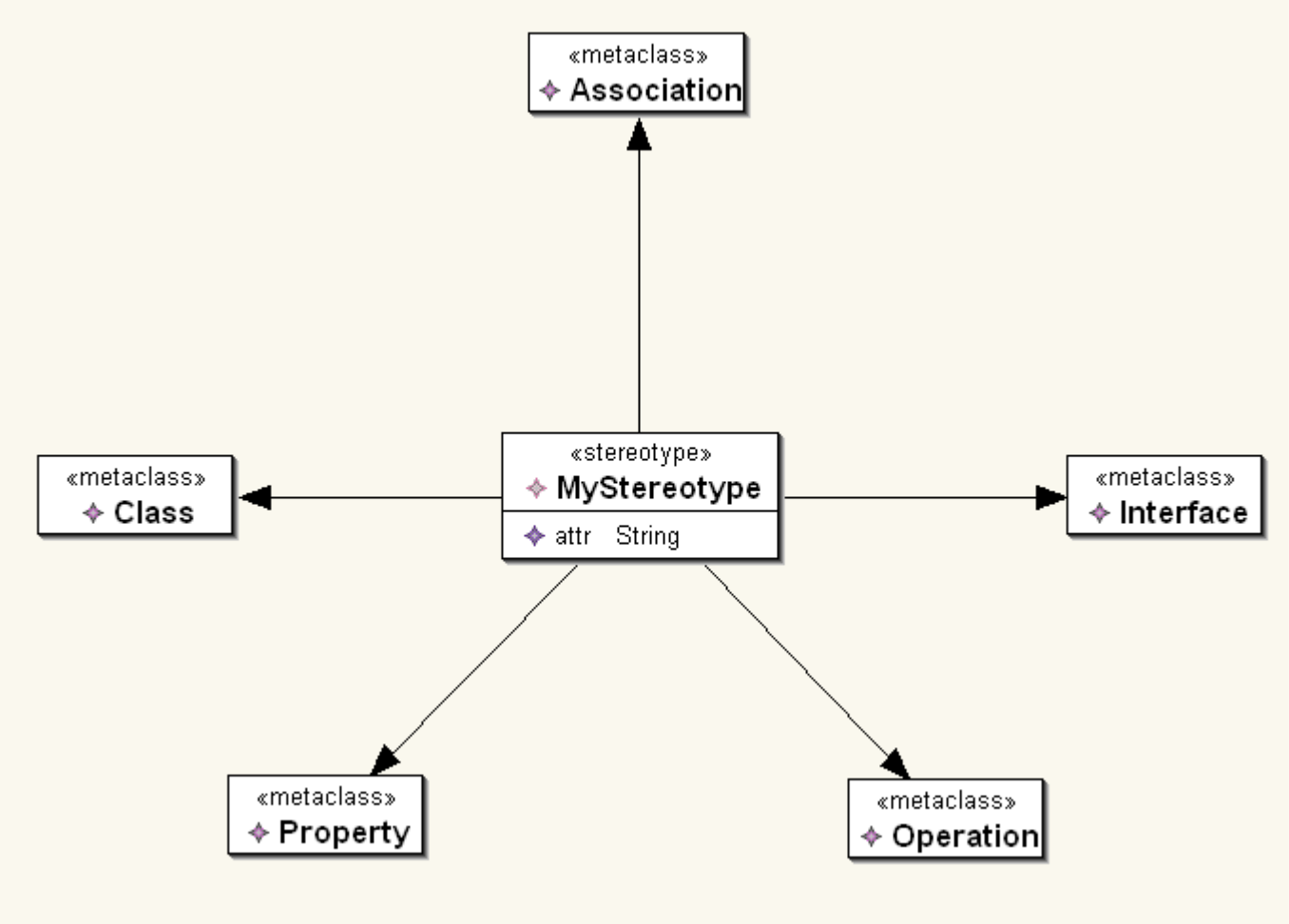
Diagramas Estruturais (estáticos)

- Diagrama de Classes
- Diagrama de Objetos
- Diagrama de Componentes
- Diagrama de Pacotes
- Diagrama de Implantação
- Diagrama de Estrutura Composta
- **Diagrama de Perfis (UML 2.2)**

Diagrama de Perfis (Profile Diagram)

- É um diagrama auxiliar que permite definir tipos padronizados de estereótipos, valores rotulados e restrições
- A UML define o mecanismo de perfis como um “mecanismo leve de extensão” da linguagem
- Permite adaptar os modelos UML para diferentes plataformas e domínios

Diagrama de Perfis (Profile Diagram)



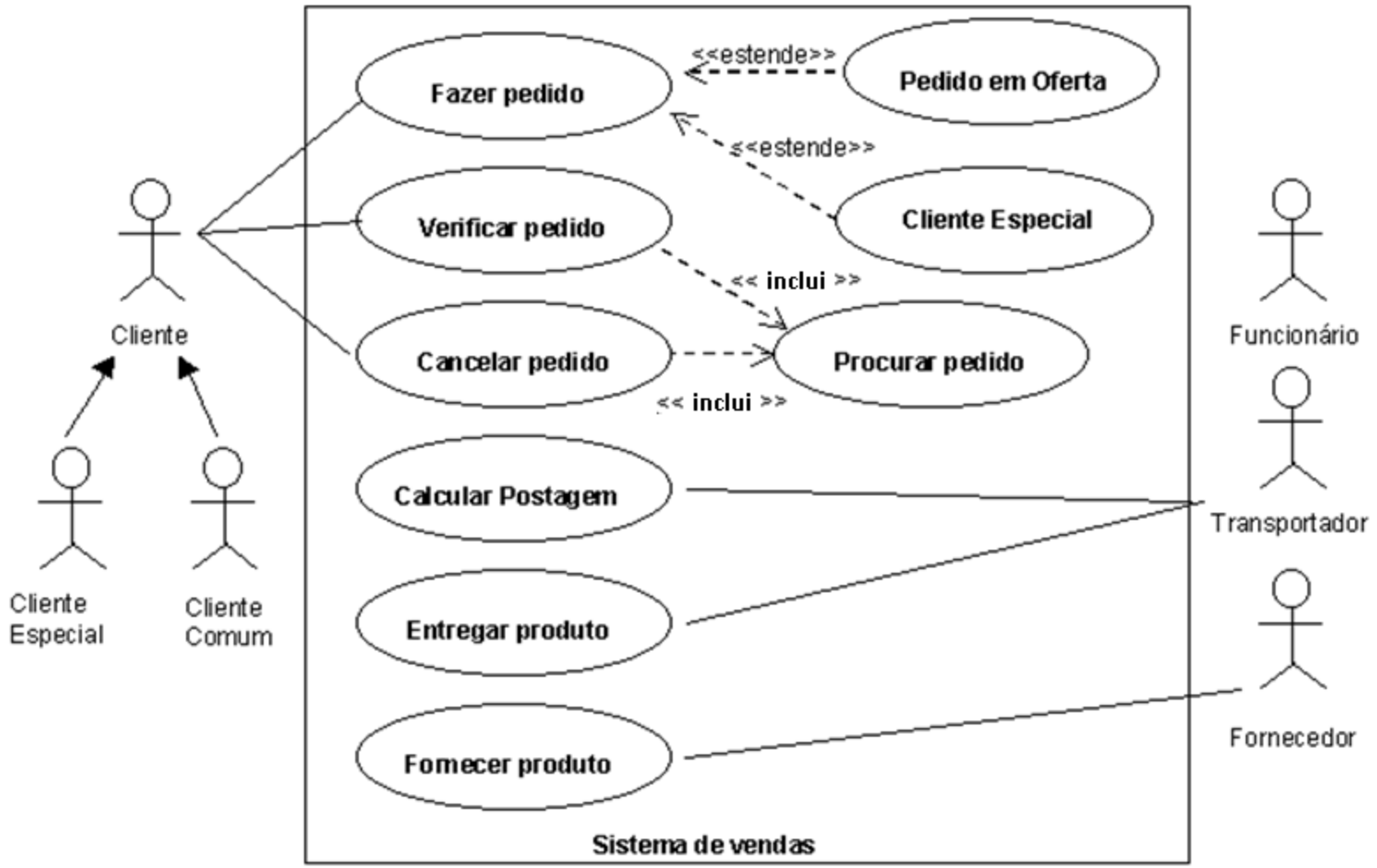
Diagramas Comportamentais (dinâmicos)

- **Diagrama de Casos de Uso**
- Diagrama de Atividade
- Diagrama de Máquina de Estados
- Diagramas de Interação
  - Diagrama de Sequência
  - Diagrama de Comunicação
  - Diagrama de Tempo
  - Diagrama de Interação Geral

Diagrama de Casos de Uso

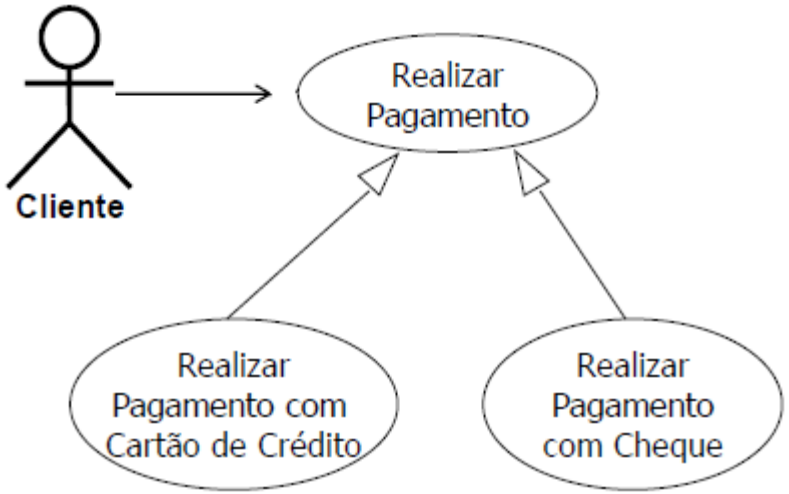
- Contém um conjunto de casos de uso e modela interações entre
  - Atores e o sistema
  - O próprio sistema
- Descreve um conjunto de cenários
- Captura os requisitos do usuário
- Delimita o escopo do sistema

Diagrama de Caso de Uso



Generalizações entre Casos de Uso

- O filho herda o comportamento do pai, podendo adicionar e redefinir passos em pontos arbitrários do comportamento original



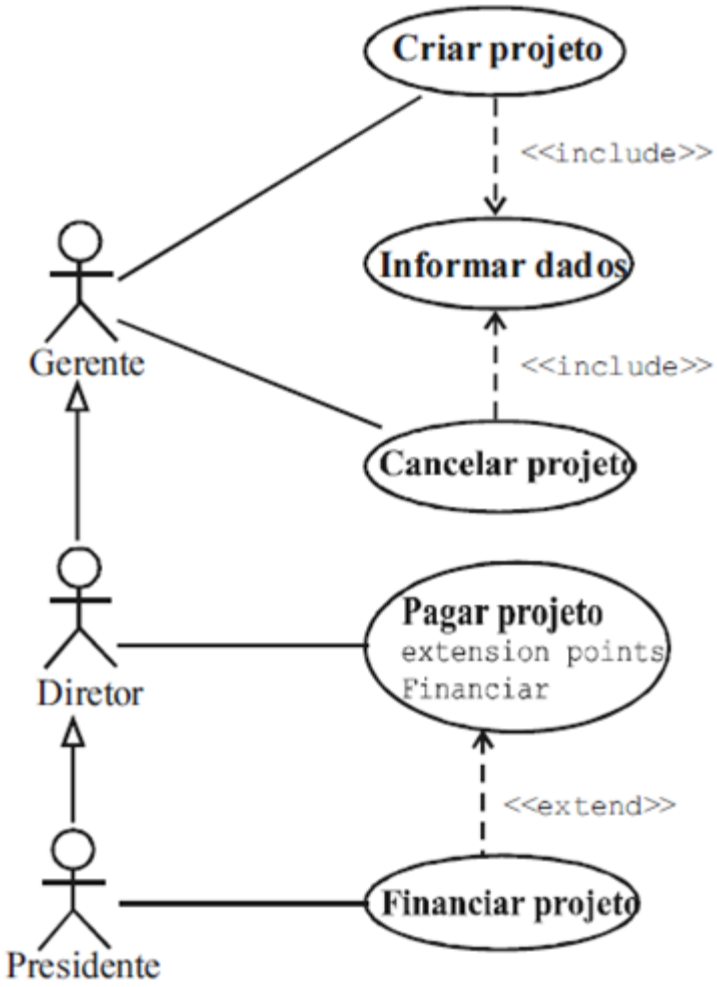
Inclusão e Extensão

- Inclusão
  - Use quando o mesmo comportamento se repete em mais de um Caso de Uso e o processo de realizar X **sempre** envolve realizar Y pelo menos uma vez
- Extensão
  - Use quando você quiser modelar um comportamento **opcional** de um Caso de Uso

Generalização entre Atores

- Use quando um ator (filho) é um **tipo de** outro ator mais genérico (pai)

- Exemplo:



Tipos de Casos de Uso

- Concreto
  - É iniciado por um ator e constitui um fluxo completo de eventos
- Abstrato: nunca é instanciado diretamente
  - Casos de Uso abstratos geralmente são:
    - Incluídos em outros Casos de Uso
    - Estendidos de outros Casos de Uso
    - Generalizações de outros Casos de Uso
- Atores “enxergam” apenas casos de uso concretos

Exercícios [6]

(EMBASA – CESPE 2009)

[95] Um diagrama de casos de uso descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário. É comum o uso de atores nesse diagrama.

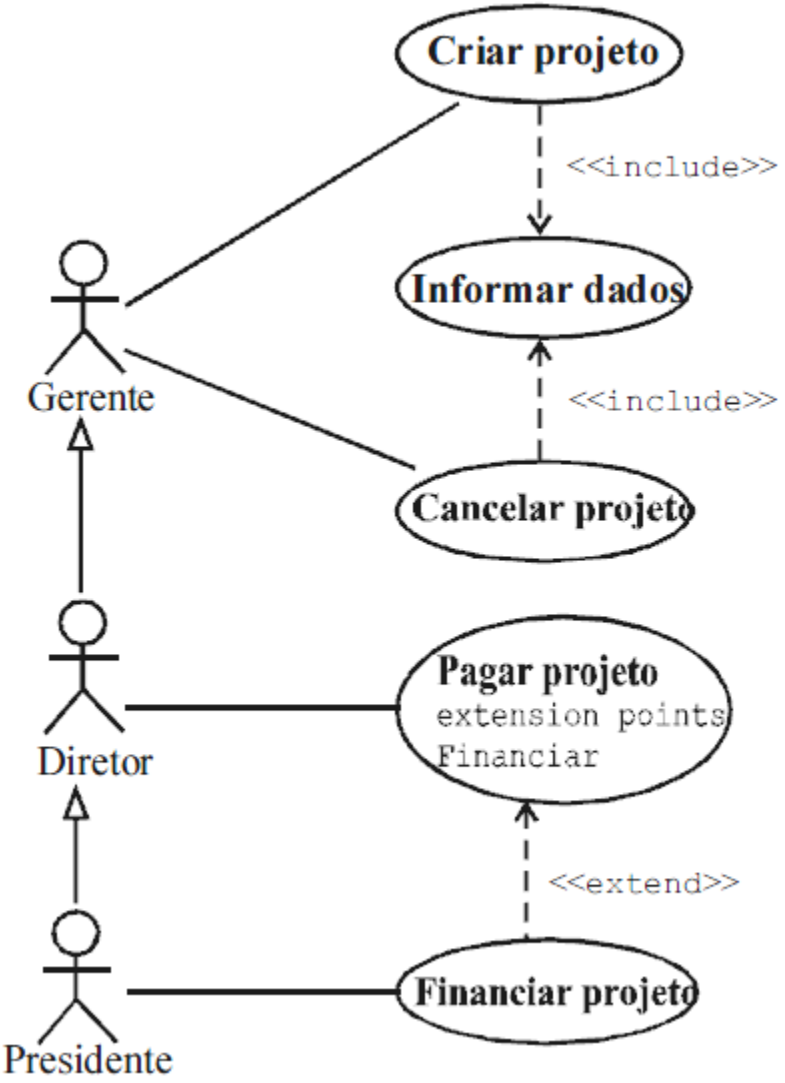
(MPE/AM – CESPE 2008)

[85] Em um diagrama de casos de uso da UML, um ator é definido como um usuário humano do sistema.

Exercícios [6]

(MPE/RR – CESPE 2008)

[87] No diagrama UML ao lado, o ator Presidente está relacionado ao caso de uso Criar projeto; o caso de uso informar dados contém comportamento comum a dois casos de uso; o caso de uso Pagar projeto estende o comportamento Financiar projeto e Cancelar projeto é abstrato.



Diagramas Comportamentais (dinâmicos)

- Diagrama de Casos de Uso
- **Diagrama de Atividade**
- Diagrama de Máquina de Estados
- Diagramas de Interação
  - Diagrama de Sequência
  - Diagrama de Comunicação
  - Diagrama de Tempo
  - Diagrama de Interação Geral

Diagrama de Atividade

- Descreve lógicas de procedimento, processos de negócio e fluxos de trabalho
- Permite que seja mostrado que entidade é responsável por cada ação no diagrama, com uso de raias (*swimlanes*)
  - Quem faz o quê?

Diagrama de Atividade

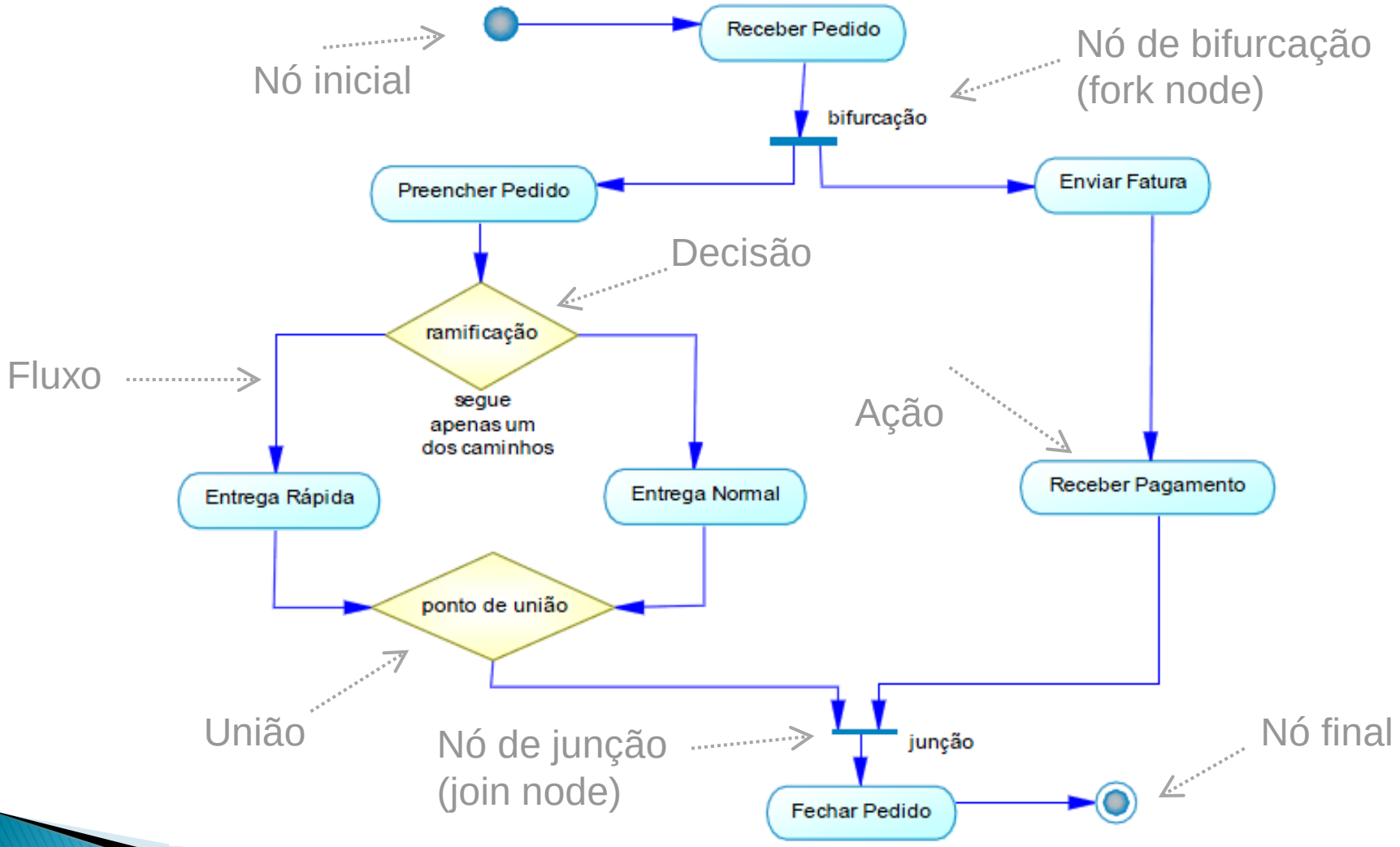
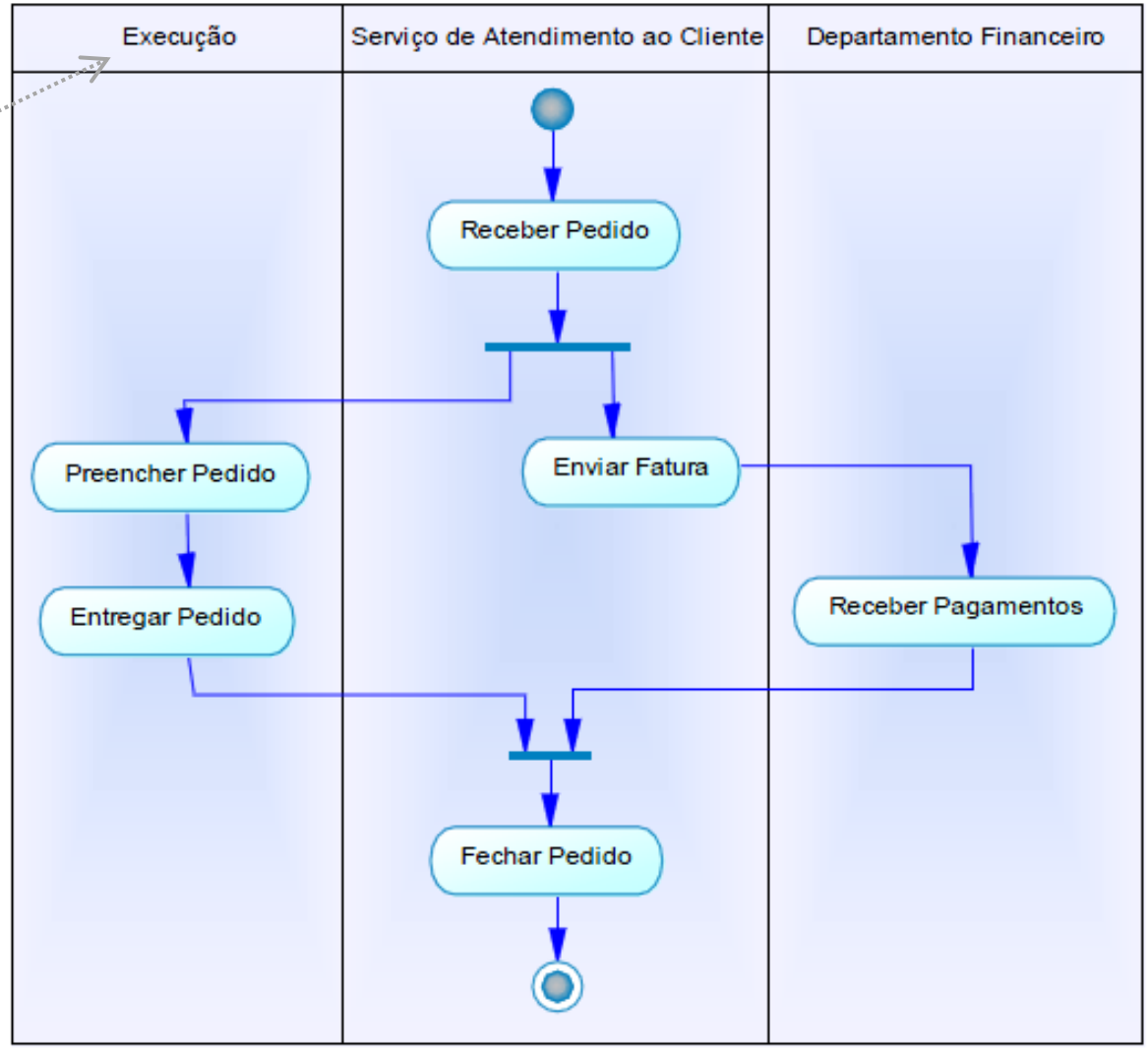


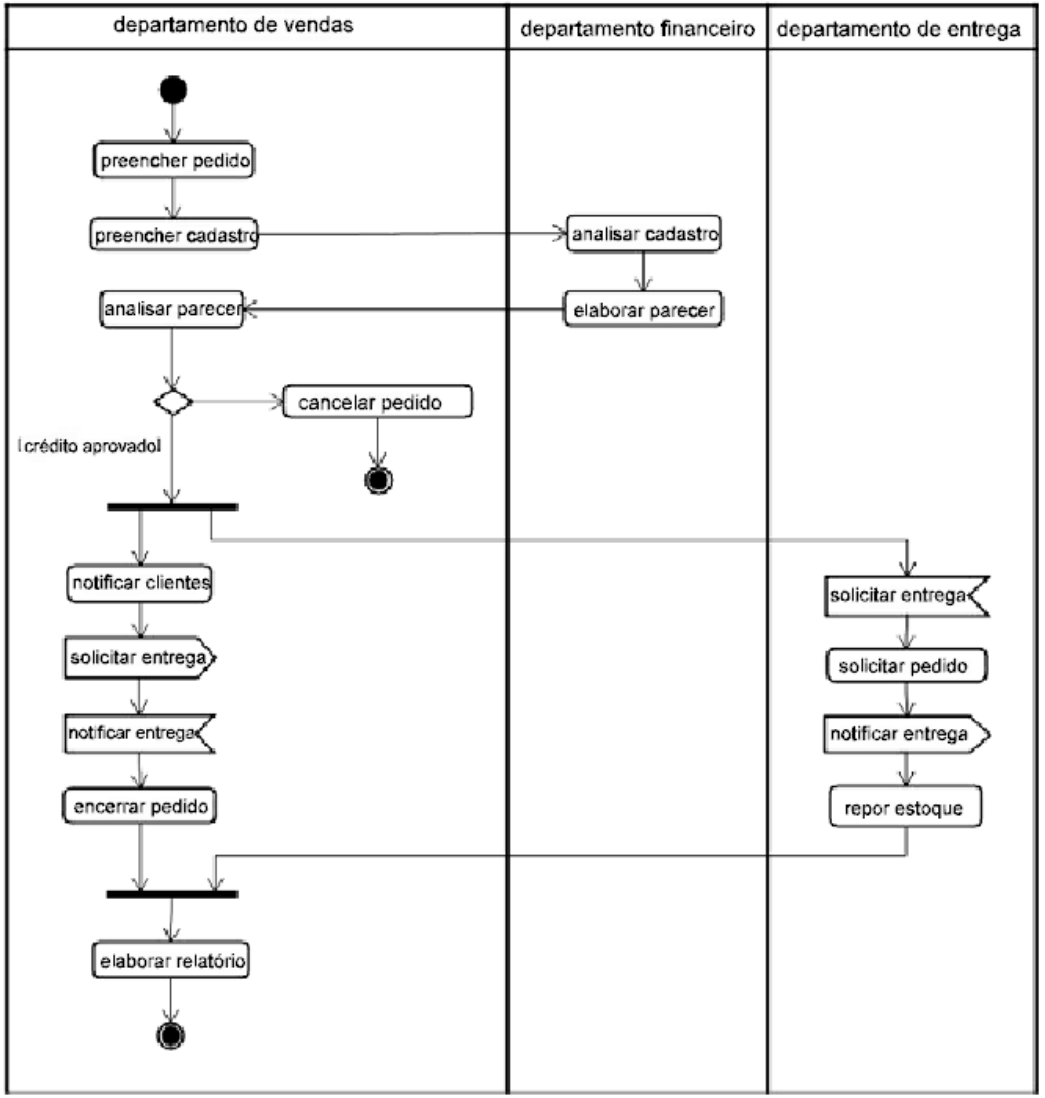
Diagrama de Atividade (swimlanes)

Unidades organizacionais



Exercícios [7]

(PETROBRAS – CESPE 2007)
Com referência ao diagrama de atividades UML ao lado, julgue os itens a seguir.



Exercícios [7]

[88] No diagrama, há duas raias, um estado inicial e dois finais. Por estarem em raias distintas, a atividade Preencher cadastro pode ser realizada em paralelo à atividade Analisar cadastro. Na decisão representada pelo losango, apenas uma condição de guarda é especificada, o que torna o diagrama incorreto.

[89] A atividade Notificar cliente pode ser executada em paralelo à atividade Entregar produto, mas a atividade Encerrar pedido não pode ser executada em paralelo à atividade Repor estoque. A atividade Elaborar relatório será executada após ser concluída a atividade Encerrar pedido ou Repor estoque.

Diagramas Comportamentais (dinâmicos)

- Diagrama de Casos de Uso
- Diagrama de Atividade
- **Diagrama de Máquina de Estados**
- Diagramas de Interação
  - Diagrama de Sequência
  - Diagrama de Comunicação
  - Diagrama de Tempo
  - Diagrama de Interação Geral

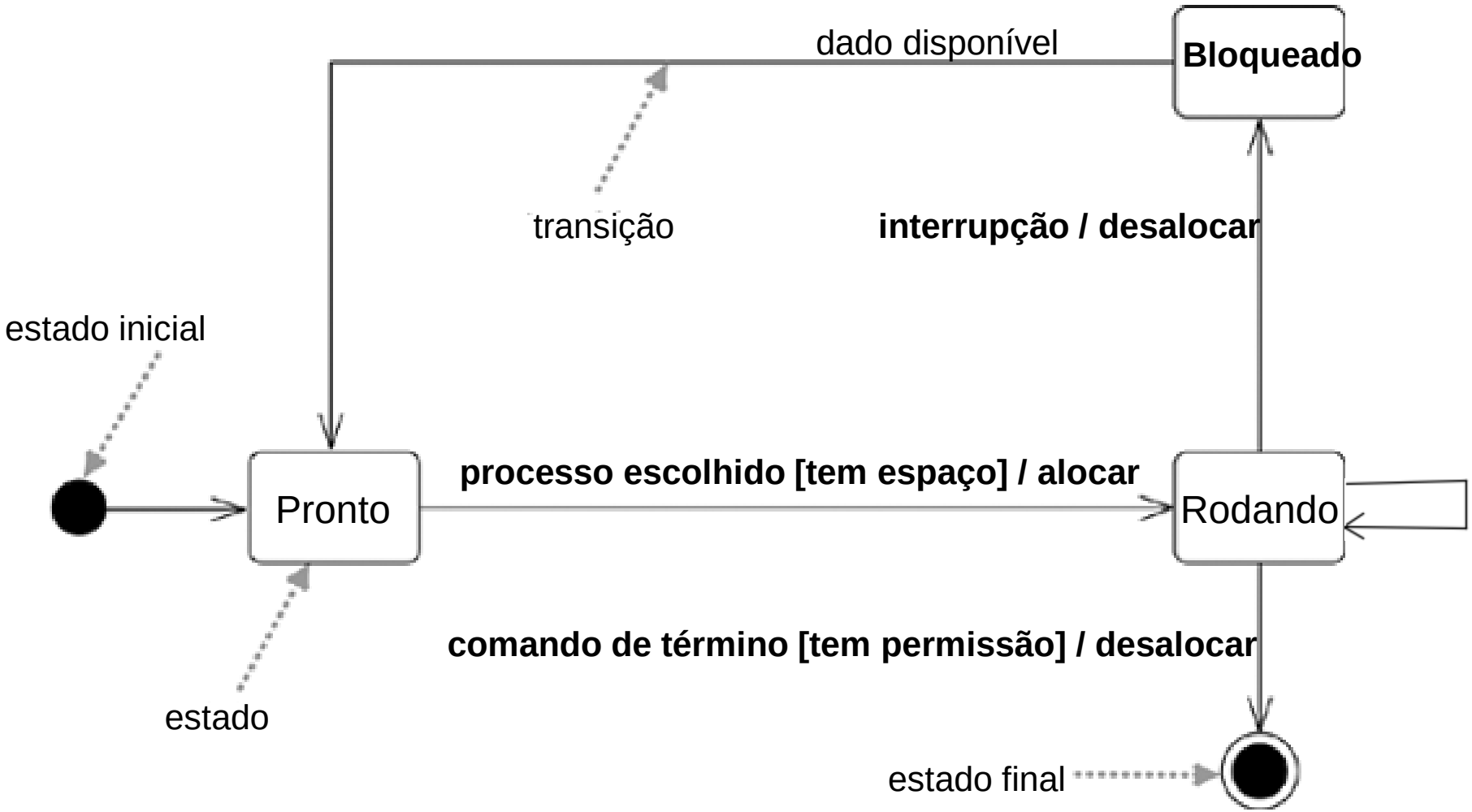
Diagrama de Máquina de Estados

- Mostra os vários estados possíveis por quais um objeto pode passar
- Um objeto muda de estado quando acontece algum evento interno ou externo ao sistema
- Através da análise das transições entre os estados, pode-se prever todas as possíveis operações realizadas, em função de eventos que podem ocorrer

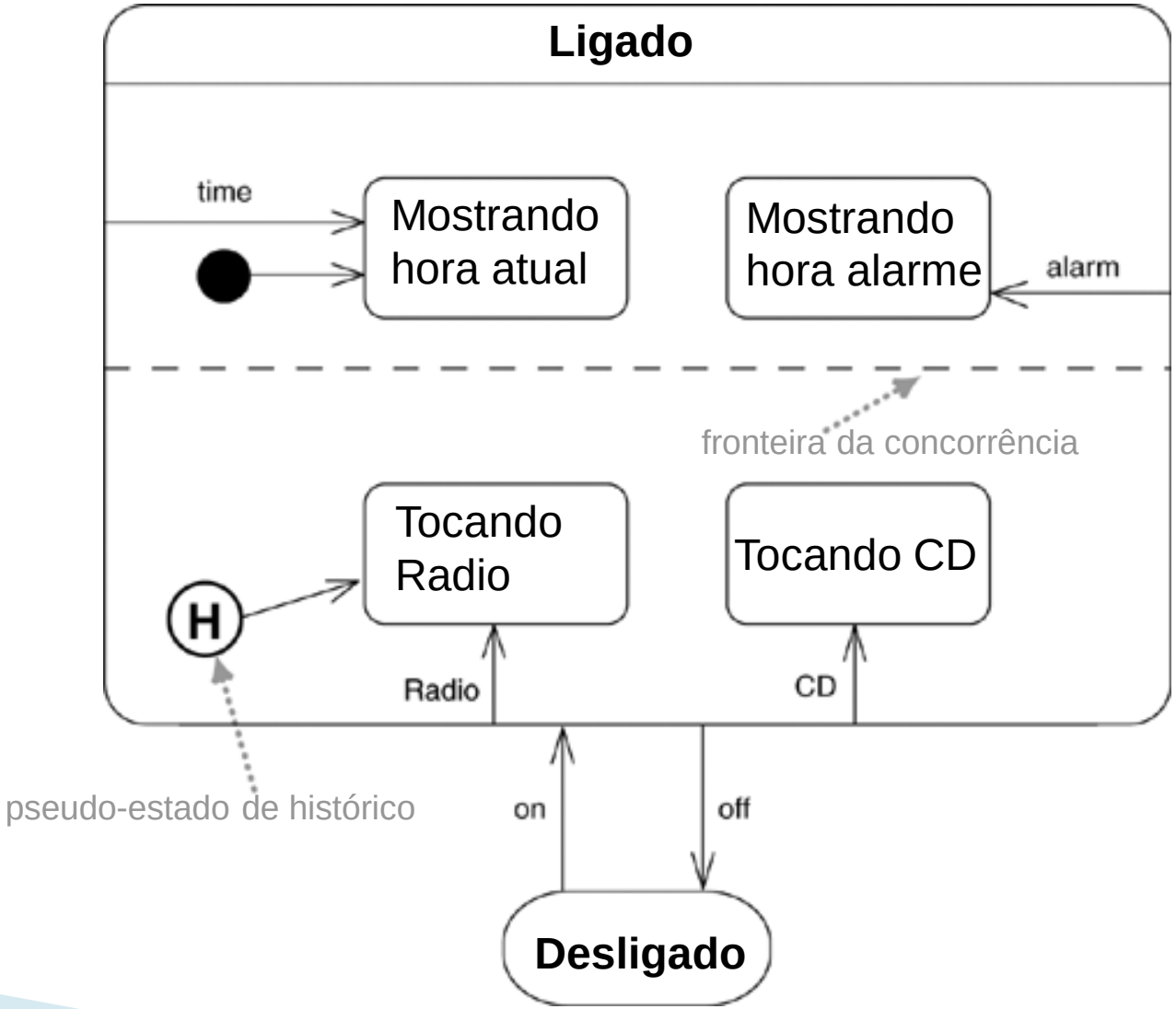
Elementos

- Estados
  - Situações na vida de um objeto na qual ele satisfaz uma condição ou realiza alguma atividade
- Transições
  - Estados são associados através de transições
  - Transições têm eventos associados
    - Sintaxe: **evento [condição]/ação**
- Ações
  - Ao passar de um estado para o outro o objeto pode realizar ações
- Atividades
  - Executadas durante um estado

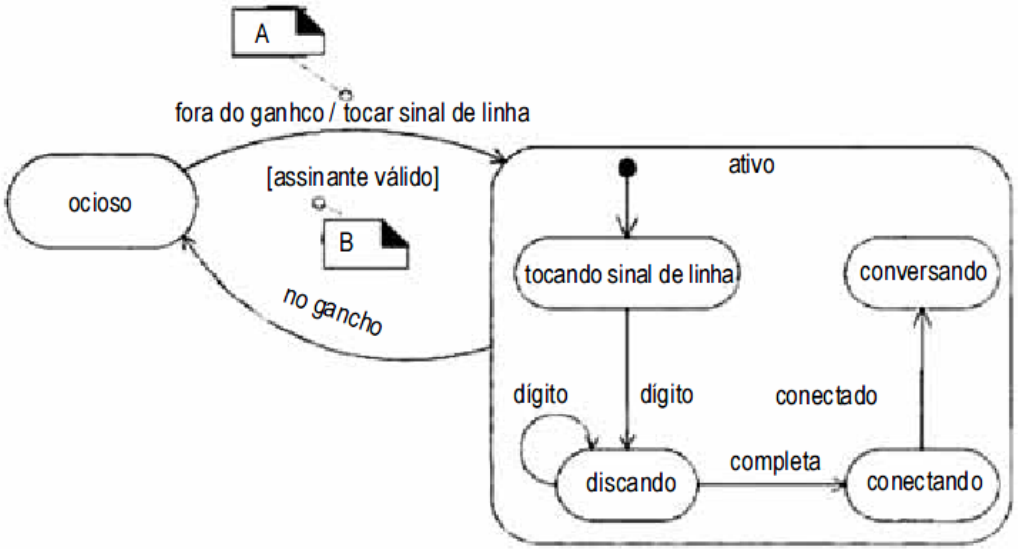
Ex: Escalonamento de Processos



Estados aninhados e concorrentes



Exercícios [8]



(EMBASA – CESPE 2009)

A figura acima é um exemplo de diagrama de transição de estados, que permite modelar como o sistema responde a eventos internos e externos, especificando o que acontece quando o evento ocorre. Ele é útil para modelar o comportamento de sistemas de tempo real, já que tais sistemas lidam com estímulos do ambiente. A respeito desse assunto e da figura acima, julgue os próximos itens.

Exercícios [8]

[73] É possível criar um diagrama de transição de estados que descreva o ciclo de vida de um objeto em níveis de detalhe arbitrariamente simples ou complexos, dependendo das necessidades, pois não há a obrigação de ilustrar todos os eventos possíveis.

[74] Na figura, A associa-se a uma ação de guarda, e B, a uma ação de transição.

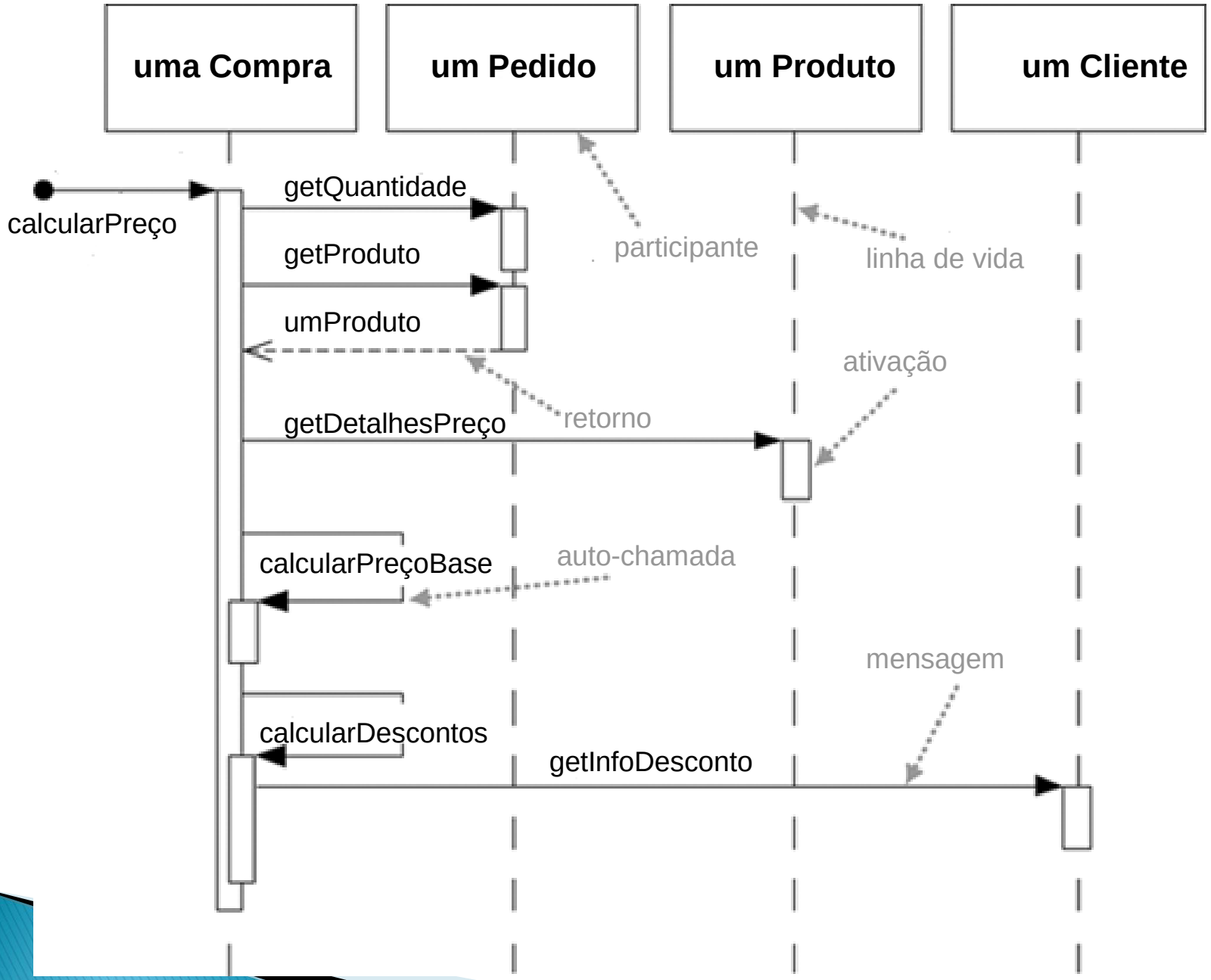
[97] Um diagrama de estado é capaz de mostrar os estados possíveis de um objeto. Além disso, pode mostrar as transações responsáveis pelas suas mudanças de estado.

Diagramas Comportamentais (dinâmicos)

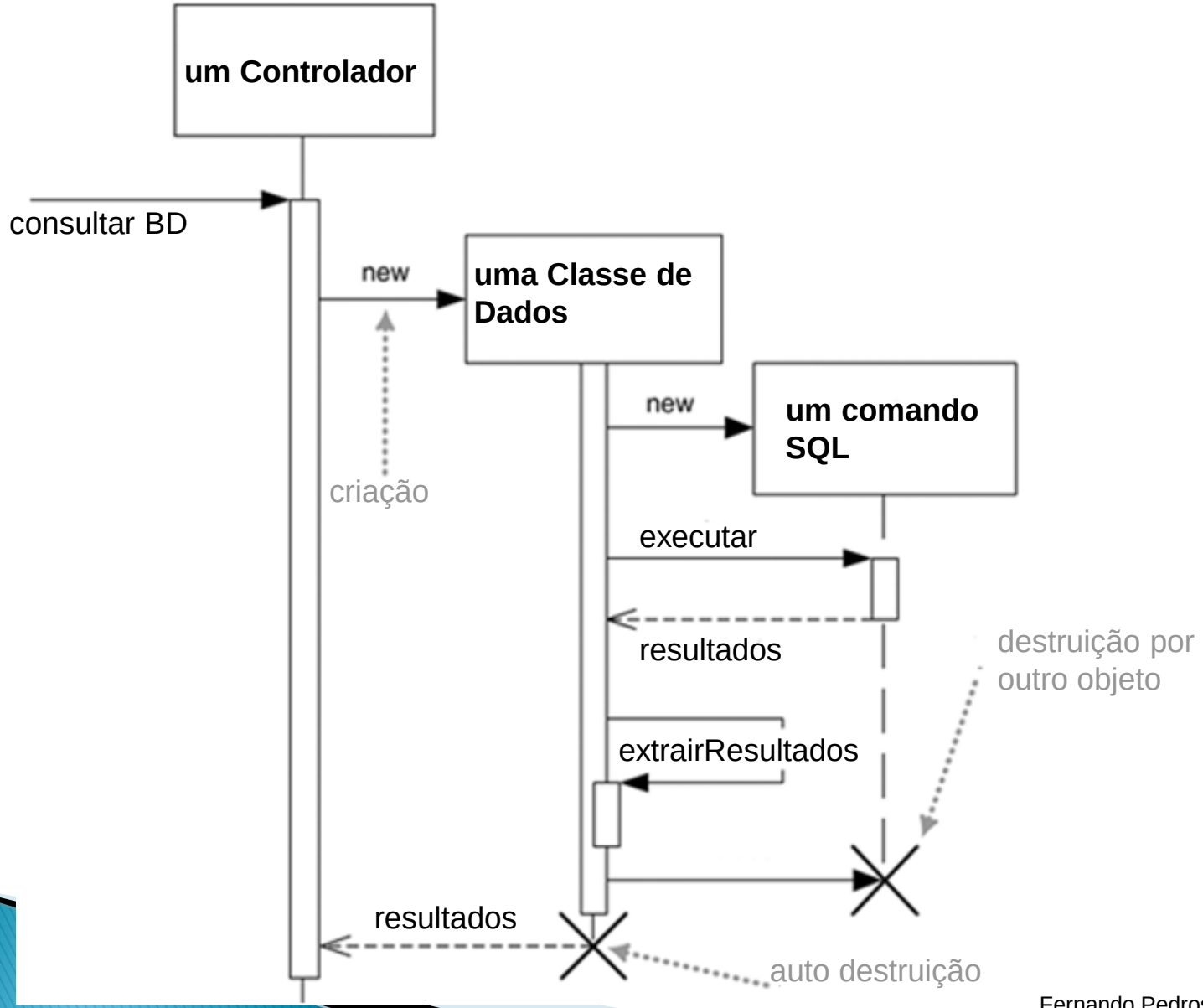
- Diagrama de Casos de Uso
- Diagrama de Atividade
- Diagrama de Máquina de Estados
- **Diagramas de Interação**
  - **Diagrama de Sequência**
  - Diagrama de Comunicação
  - Diagrama de Tempo
  - Diagrama de Interação Geral

Diagrama de Sequência

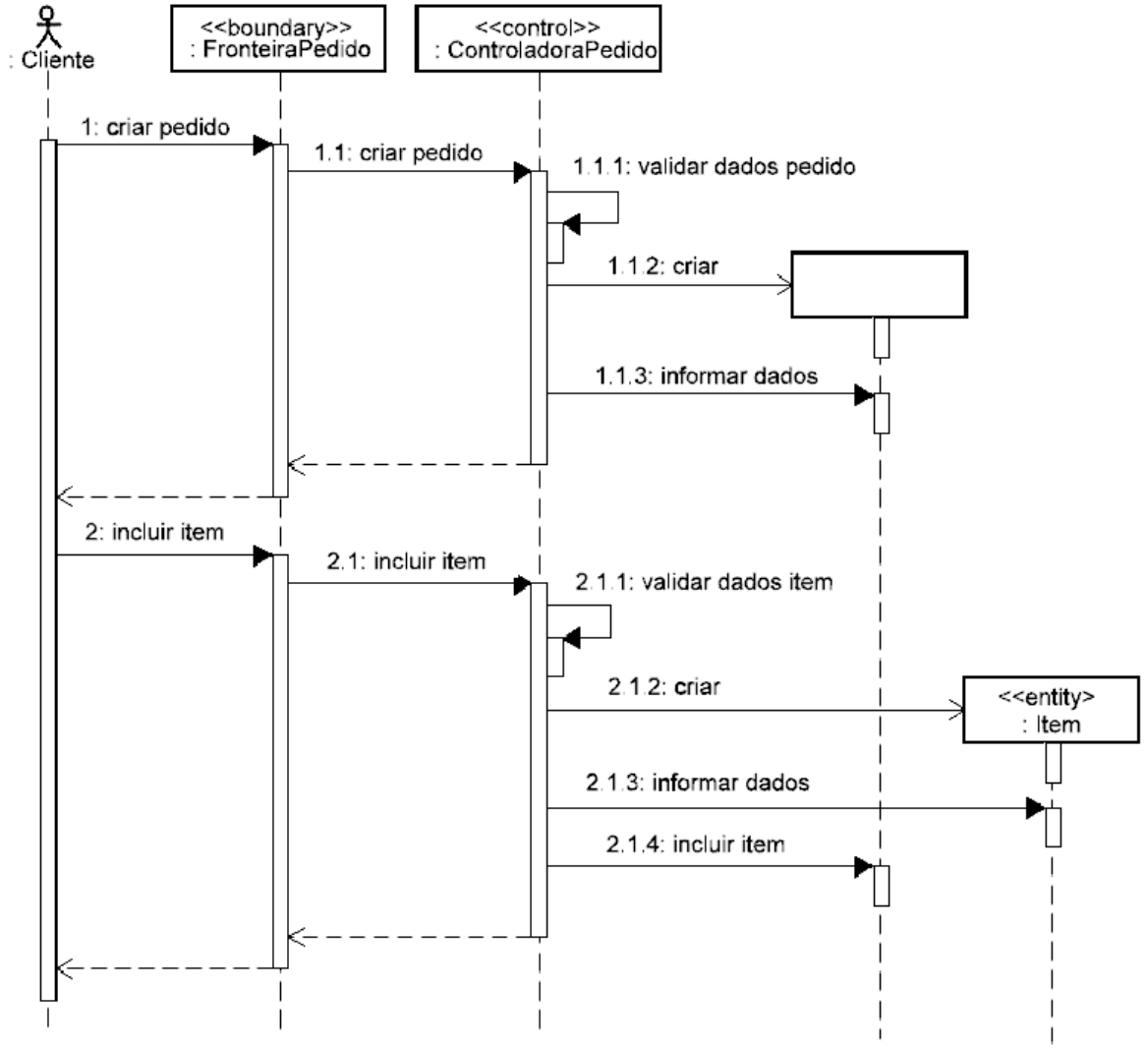
- Captura o comportamento de um determinado cenário
- Mostra os objetos e as mensagens trocadas entre eles
- Enfatiza a **ordem temporal** das mensagens
- É o diagrama mais utilizado na etapa de Projeto OO (solucionar o problema)


uma Compra
um Pedido
um Produto
um Cliente
calcularPreço
getQuantidade
getProduto
umProduto
participante
linha de vida
ativação
retorno
getDetalhesPreço
calcularPreçoBase
auto-chamada
mensagem
calcularDescontos
getInfoDesconto

Criação e destruição de objetos



Exercícios [9]



Exercícios [9]

(PETROBRAS - CESPE 2007)
Julgue os itens a seguir, relativos ao diagrama de seqüência UML apresentado acima.

[91] Dois objetos existiam antes da interação e dois foram criados durante a interação. As setas da instância de ControladoraPedido para a instância de FronteiraPedido são retornos de mensagens. Um dos objetos tem nome Pedido e outro, Item. No diagrama, encontram-se representadas as linhas da vida dos objetos e as áreas de ativação das mensagens.

[92] São assíncronas as mensagens da instância de FronteiraPedido para a de ControladoraPedido. Há um erro no diagrama, pois uma instância de uma classe não pode enviar mensagens para ela mesma.

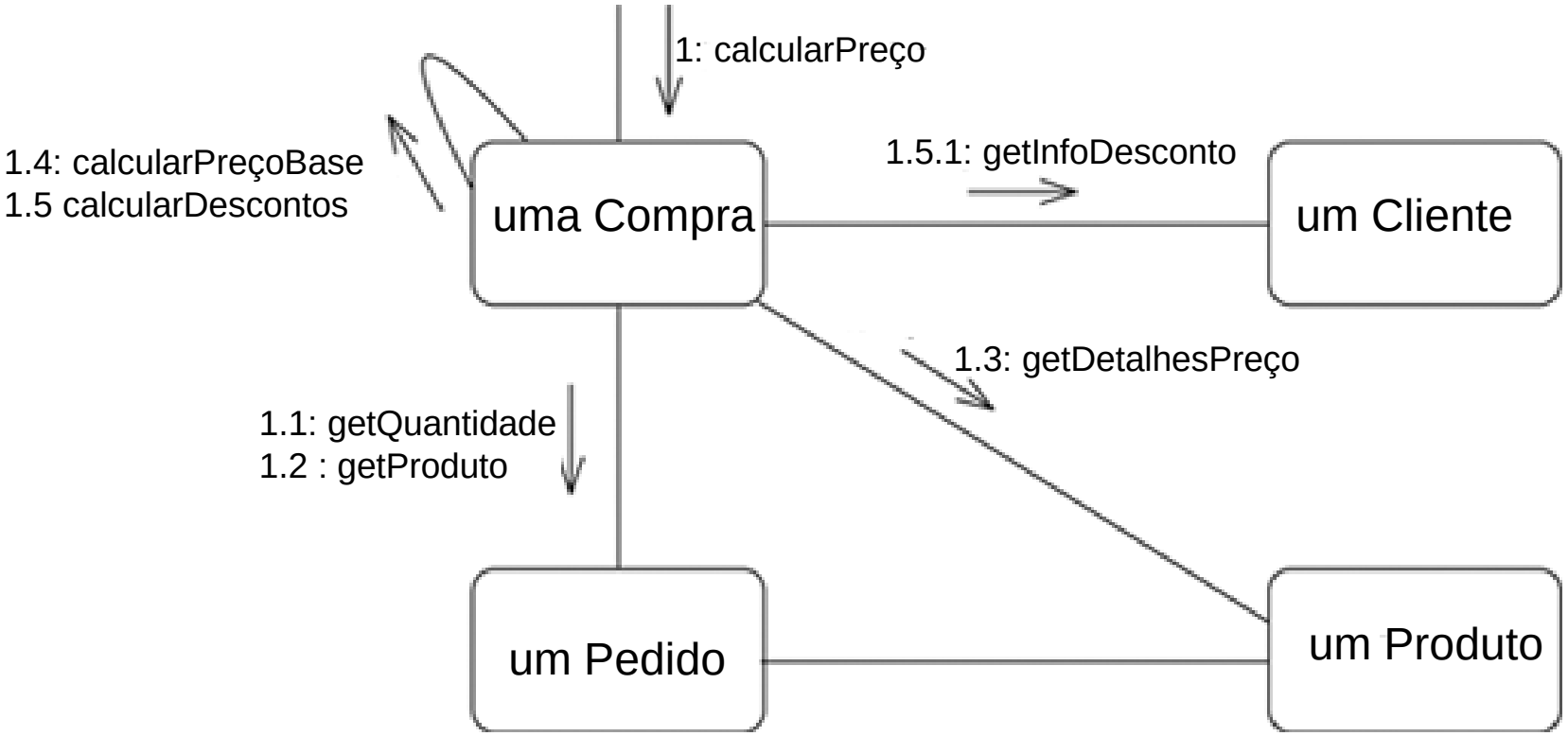
Diagramas Comportamentais (dinâmicos)

- Diagrama de Casos de Uso
- Diagrama de Atividade
- Diagrama de Máquina de Estados
- **Diagramas de Interação**
  - Diagrama de Sequência
  - **Diagrama de Comunicação**
  - Diagrama de Tempo
  - Diagrama de Interação Geral

Diagrama de Comunicação

- Captura o comportamento de um determinado cenário
- Mostra os objetos e as mensagens trocadas entre eles
- Enfatiza a **ordem estrutural** das mensagens (relacionamentos entre objetos)
- É equivalente ao diagrama de sequência

Diagrama de Comunicação



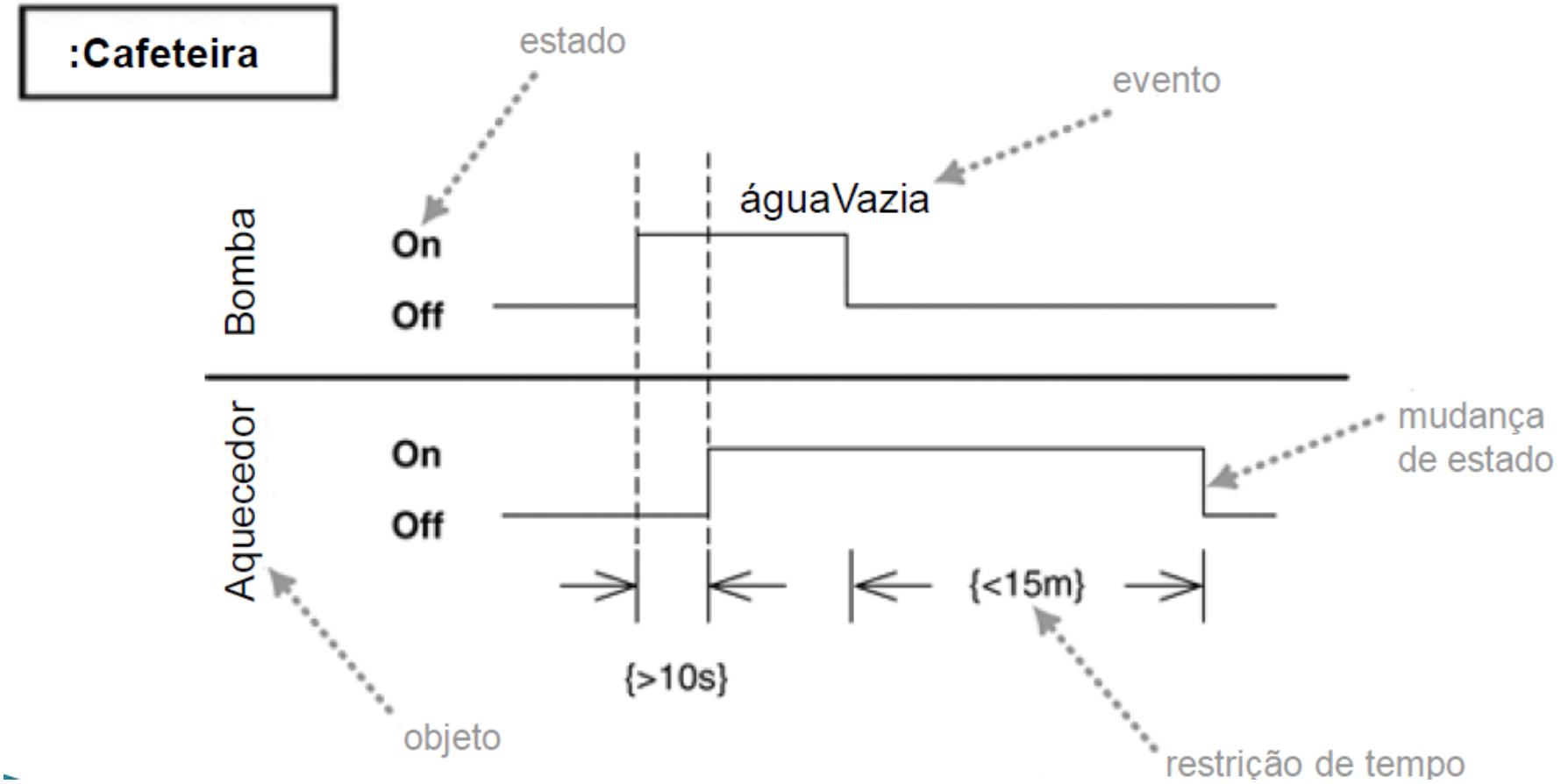
Diagramas Comportamentais (dinâmicos)

- Diagrama de Casos de Uso
- Diagrama de Atividade
- Diagrama de Máquina de Estados
- **Diagramas de Interação**
  - Diagrama de Sequência
  - Diagrama de Comunicação
  - **Diagrama de Tempo**
  - Diagrama de Interação Geral

Diagrama de Tempo

- Captura o comportamento de objetos ao longo do tempo e a duração na qual eles permanecem em determinados estados
- O foco se dá nas **restrições de tempo** das interações
- É uma mistura entre o diagrama de sequência e o diagrama de máquina de estados

Diagrama de Tempo



Diagramas Comportamentais (dinâmicos)

- Diagrama de Casos de Uso
- Diagrama de Atividade
- Diagrama de Máquina de Estados
- **Diagramas de Interação**
  - Diagrama de Sequência
  - Diagrama de Comunicação
  - Diagrama de Tempo
  - **Diagrama de Interação Geral**

Diagrama de Interação Geral

- Fornece uma visão geral do controle de fluxo entre objetos
- É uma mistura entre diagramas de sequência e diagramas de atividade

No exemplo, se o Cliente for externo, os dados são buscados de um XML. Se for interno, os dados são buscados de um banco de dados. A seqüência destes dois fluxos é detalhada. Ao final, é gerado um relatório

Diagrama de Interação Geral

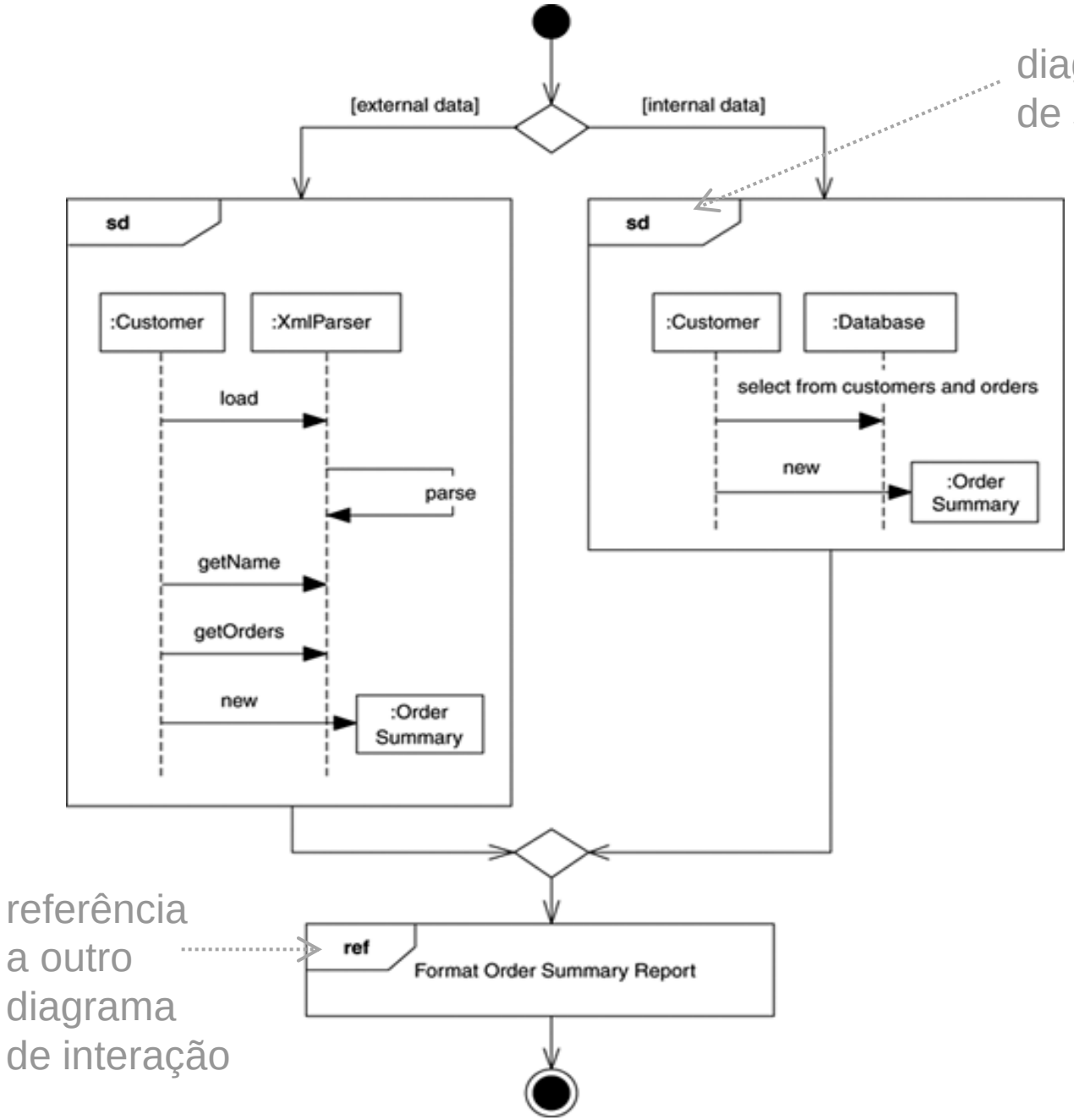


diagrama de sequência

referência a outro diagrama de interação

Object Constraint Language

- Linguagem que faz parte da UML e tem o objetivo de desenvolver modelos mais precisos
- Uma **restrição** (constraint) atua sobre um ou mais valores de um modelo orientado a objetos
- Vantagens
  - Modelos mais completos, consistentes e precisos
  - Comunicação sem ambigüidade
  - Sintaxe e semântica formais

Object Constraint Language

- Exemplos de restrições (regras de um sistema de Universidade)
  - “A avaliação de supervisores acadêmicos deve ser maior que a nota dos seus supervisionados”
  - “A bolsa escolar dos alunos depende da sua avaliação acadêmica”
- Estas regras podem ser escritas em OCL
- E podem ser transformadas em
  - Código
  - *Scripts* de bancos de dados
  - Outros modelos, etc.

Gabaritos dos Exercícios

- [1] 78 E, 94 E, 101 E, 31 D
- [2] 88 C, 89 C, 108 E, 109 C, 110 X (E)
- [3] 106 E, 40 C, 96 E, 40 C
- [4] 54 E
- [5] 105 C
- [6] 95 C, 85 E, 87 E
- [7] 88 E, 89 E
- [8] 73 C, 74 E, 97 C
- [9] 91 E, 92 E

FIM