

Análise e Projeto de Sistemas

Universidade Federal do Ceará – UFC

Campus de Quixadá

Curso de Sistemas de Informação

Prof. Marcos Antonio de Oliveira (deoliveira.ma@gmail.com)

“O engenheiro de software amador está sempre à procura de magia, de algum método sensacional ou ferramenta cuja aplicação promete tornar trivial o desenvolvimento de software. É uma característica do engenheiro de software profissional saber que tal panacéia não existe.” (GRADY BOOCH)

MODELAGEM DE CLASSES DE ANÁLISE

Esses slides são uma adaptação das notas de aula do professor Eduardo Bezerra autor do livro Princípios de Análise e Projeto de Sistemas com UML

Índice

- Introdução
- Estágios do modelo de classes
- Diagrama de classes
- Diagrama de objetos
- Técnicas para identificação de classes
- Construção do modelo de classes
- Modelo de classes no processo I&I
- Estudo de caso

INTRODUÇÃO

Introdução

- O modelo de casos de uso fornece uma perspectiva do sistema a partir de um ponto de vista **externo**
- De posse da visão de casos de uso, os desenvolvedores precisam prosseguir no desenvolvimento do sistema

Introdução

- A funcionalidade externa de um sistema orientado a objetos é fornecida através de colaborações entre objetos

Externamente: os atores visualizam resultados de cálculos, relatórios produzidos, confirmações de requisições realizadas, etc.

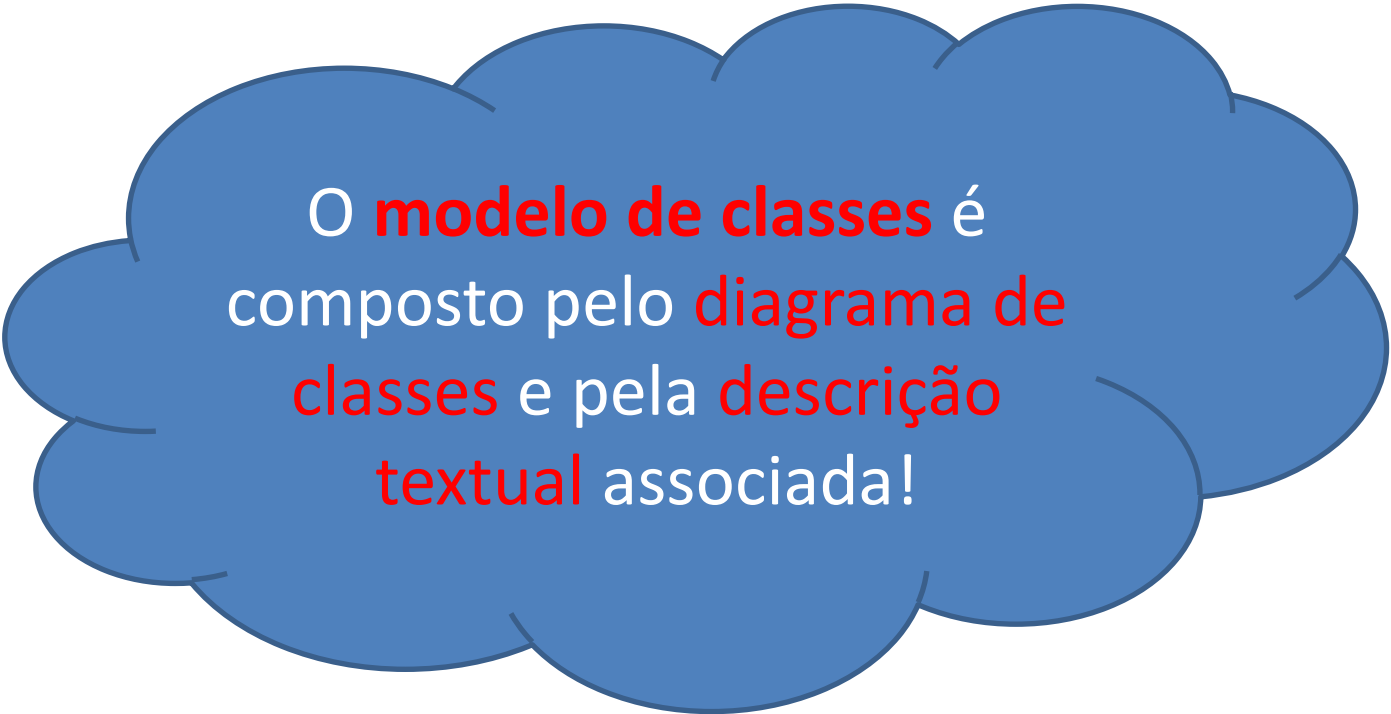
Internamente: os objetos ***colaboram*** uns com os outros para produzir os resultados.

Essa **colaboração** pode ser vista sob o **aspecto dinâmico** e sob o **aspecto estrutural estático**.

ESTÁGIOS DO MODELO DE CLASSES

Modelo de Classes

- O diagrama da UML utilizado para representar o aspecto estático é o **diagrama de classes**



O **modelo de classes** é composto pelo **diagrama de classes** e pela **descrição textual** associada!

Modelo de Classes

- O modelo de classes evolui durante o desenvolvimento do sistema
 - À medida que o sistema é desenvolvido, o modelo de classes é incrementado com novos detalhes



Modelo de Classes

O Modelo de Classes de Análise

Representa as classes no domínio do negócio em questão. Não leva em consideração restrições inerentes à tecnologia a ser utilizada na solução de um problema.

O Modelo de Classes de Especificação

É obtido através da adição de detalhes ao modelo anterior conforme a solução de software escolhida.

O Modelo de Classes de Implementação

Corresponde à implementação das classes em alguma linguagem de programação.

DIAGRAMA DE CLASSES

Introdução

- O diagrama de classes é utilizado na construção do modelo de classes desde o nível de análise até o nível de especificação
- É o digrama mais **rico** da UML em termos de notação

Classes

- Uma classe representa um grupo de objetos semelhantes
- Uma classe descreve esses objetos através de **atributos e operações**
 - Os atributos correspondem às informações que um objeto armazena
 - As operações correspondem às ações que um objeto sabe realizar

Notação para uma Classe

- Representada através de uma “caixa” com no máximo três compartimentos exibidos
- Notação utilizada depende do nível de abstração desejado

Nome da Classe

Nome da Classe
lista de atributos

Nome da Classe
lista de operações

Nome da Classe
lista de atributos
lista de operações

Exemplo (classe ContaBancária)

ContaBancária

ContaBancária
número
saldo
dataAbertura

ContaBancária
número
saldo
dataAbertura
criar()
bloquear()
desbloquear()
creditar()
debitar()

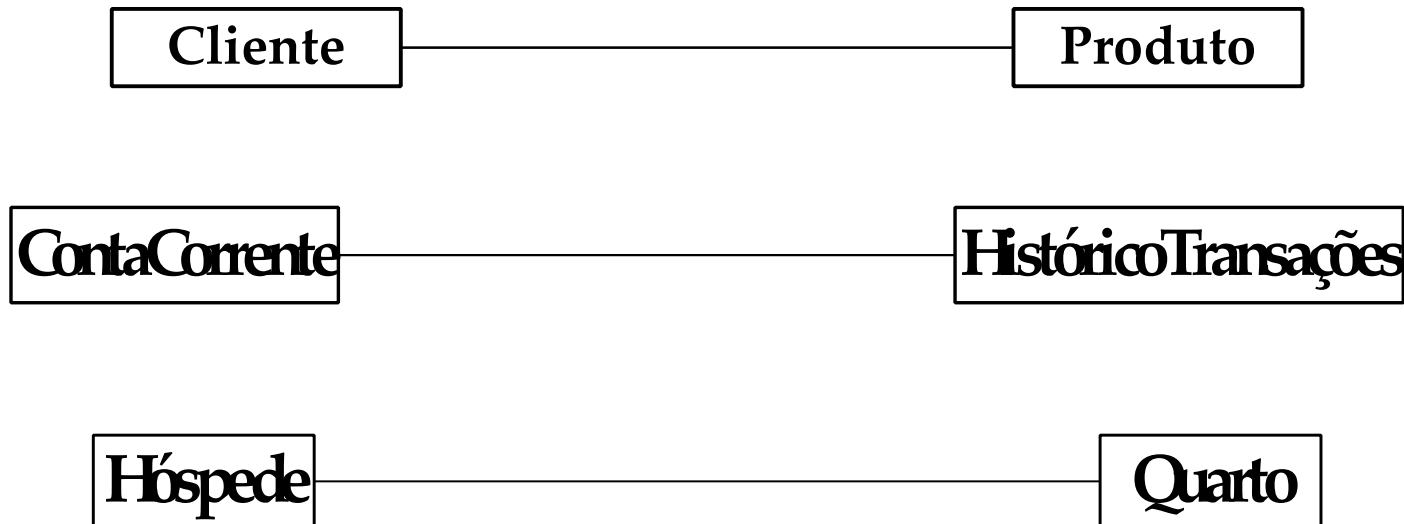
Associações

- Para representar o fato de que objetos podem se relacionar uns com os outros, utiliza-se a ***associação***
- Uma associação representa relacionamentos (ligações) que são formados entre objetos durante a execução do sistema

Embora as associações sejam representadas entre **classes** do diagrama, tais associações representam **ligações possíveis** entre **objetos** das classes envolvidas.

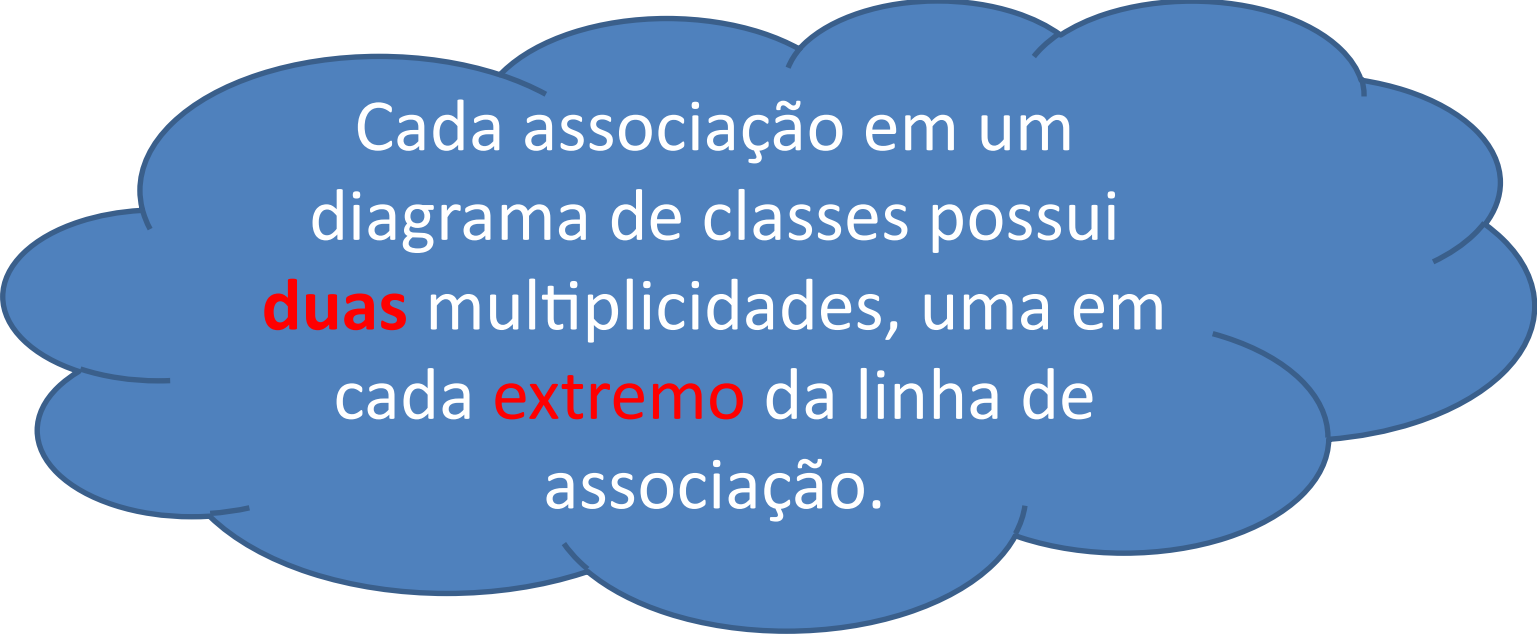
Notação para Associações

- Representada através de um segmento de reta ligando as **classes** cujos **objetos** se relacionam



Multiplicidade

- Representam a informação dos limites **inferior** e **superior** da quantidade de **objetos** aos quais um outro **objeto** pode estar associado



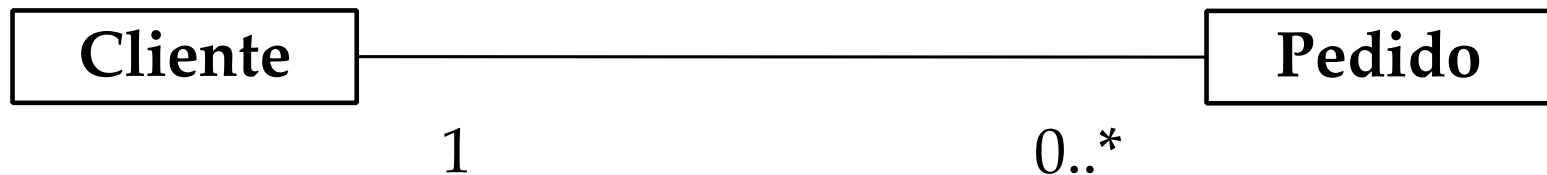
Cada associação em um diagrama de classes possui **duas** multiplicidades, uma em cada **extremo** da linha de associação.

Multiplicidades

Nome	Simbologia
Apenas Um	1..1 (ou 1)
Zero ou Muitos	0..* (ou *)
Um ou Muitos	1..*
Zero ou Um	0..1
Intervalo Específico	$l_i..l_s$

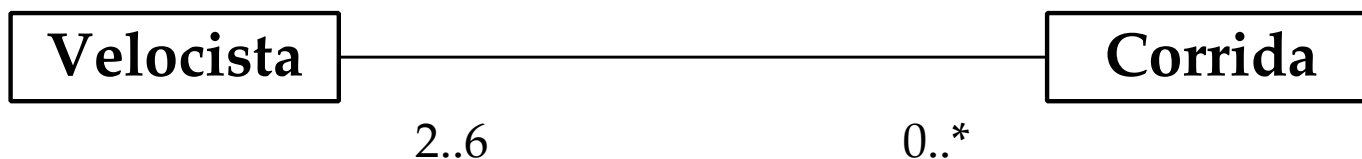
Exemplo (Multiplicidade)

- **Pode** haver um cliente que esteja associado a vários pedidos
- **Pode** haver um cliente que não esteja associado a pedido algum
- Um pedido está associado a um, e somente um, cliente



Exemplo (Multiplicidade)

- Uma corrida está associada a, no mínimo, dois velocistas
- Uma corrida está associada a, no máximo, seis velocistas
- Um velocista ***pode*** estar associado a várias corridas



Conectividade

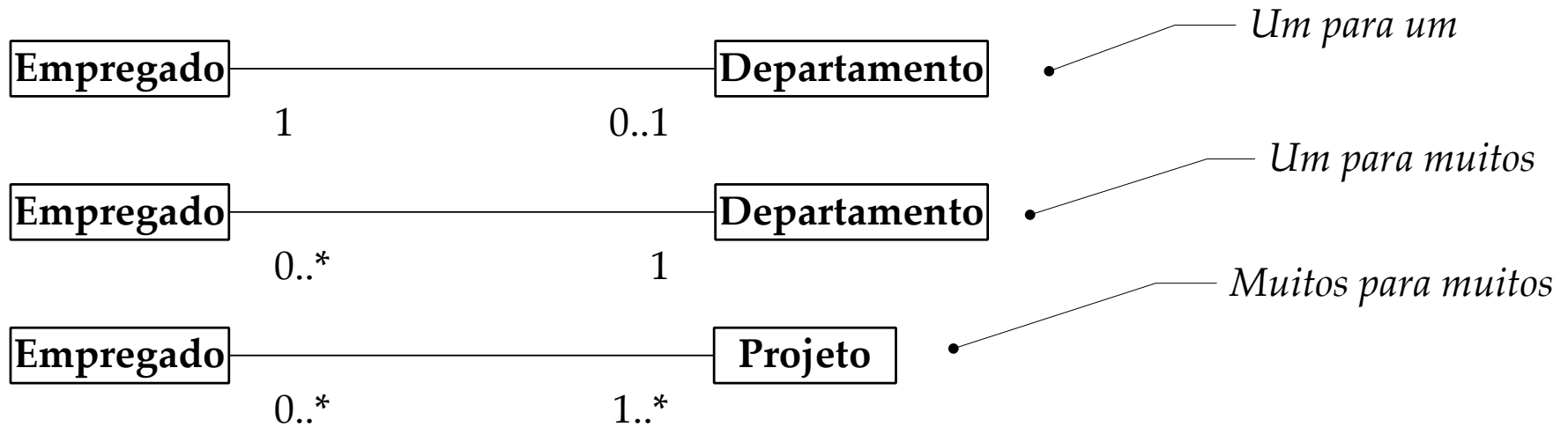
- A **conectividade** corresponde ao tipo de associação entre duas classes
 - “muitos para muitos”
 - “um para muitos”
 - “um para um”

A conectividade da associação entre duas classes depende dos símbolos de multiplicidade que são utilizados na associação!

Conectividade e Multiplicidade

Conectividade	Multiplicidade	
	Em um extremo	Em outro extremo
Um para um	0..1 1	0..1 1
Um para muitos	0..1 1	* 0..* 1..*
Muitos para muitos	* 0..* 1..*	* 0..* 1..*

Exemplo (Conectividade)



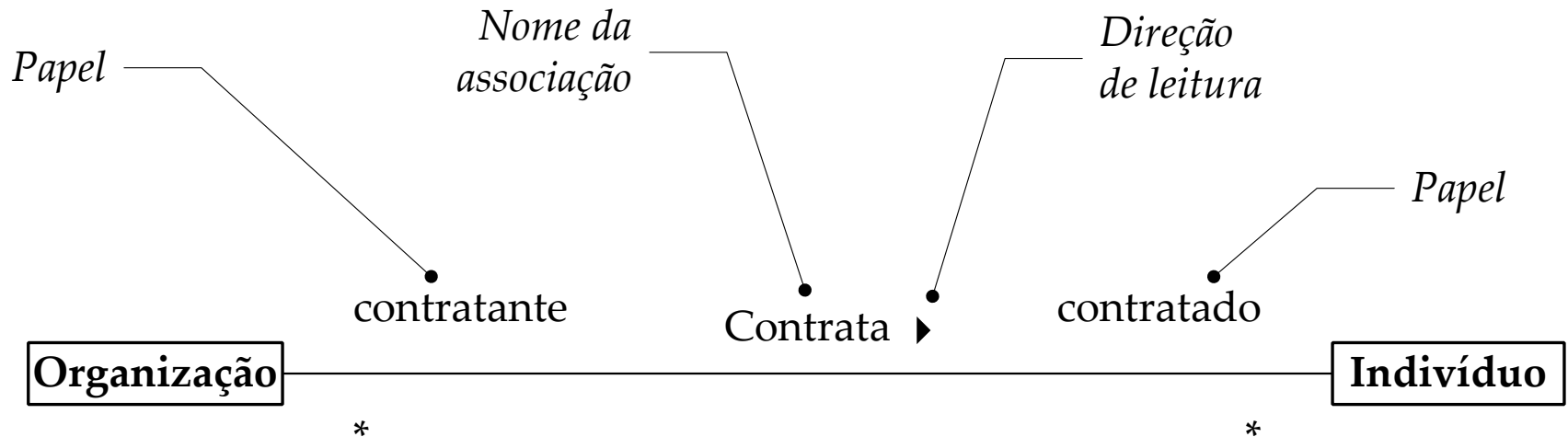
Participação

- Uma característica de uma associação que indica a necessidade (ou não) da existência desta associação entre objetos
- A participação pode ser **obrigatória** ou **opcional**
 - Se o valor mínimo da multiplicidade de uma associação é **igual a 1** (um), significa que a participação é **obrigatória**
 - Caso contrário, a participação é **opcional**

Nome de Associação, Direção de Leitura e Papéis

- Para melhor esclarecer o significado de uma associação no diagrama de classes, a UML define três recursos de notação
 - **Nome da associação:** fornece algum significado semântico a mesma
 - **Direção de leitura:** indica como a associação deve ser lida
 - **Papel:** para representar um papel específico em uma associação

Exemplo (Nome de Associação, Direção de Leitura e Papéis)

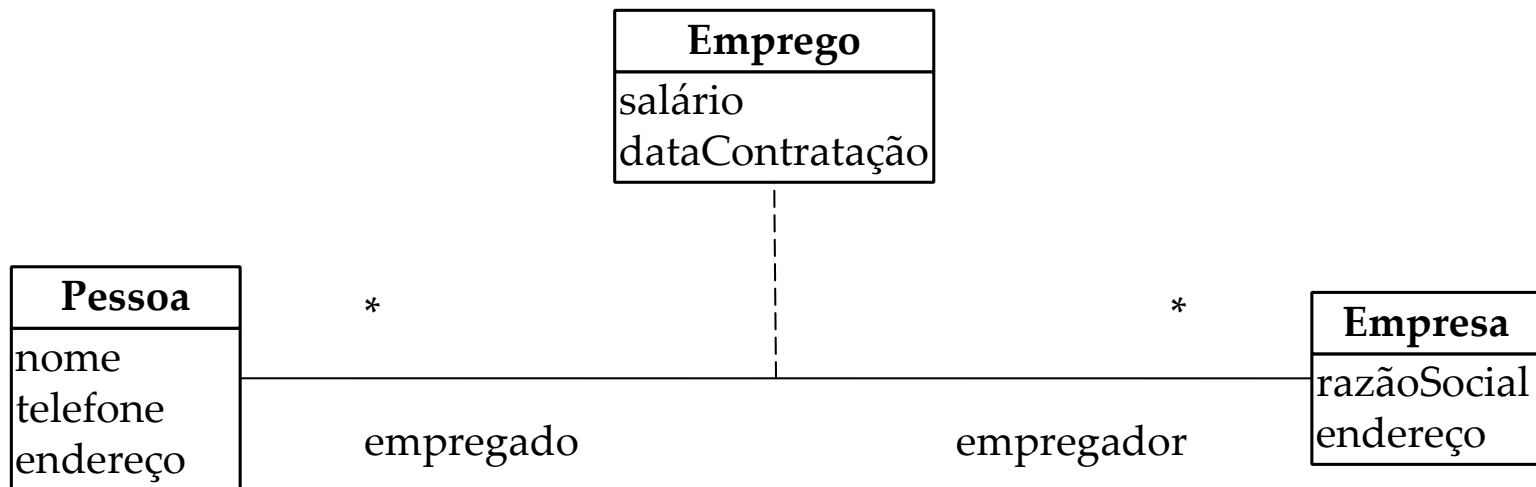


Classe Associativa

- É uma classe que está ligada a uma associação, ao invés de estar ligada a outras classes
- É normalmente necessária quando duas ou mais classes estão associadas, e é necessário manter informações sobre esta associação
- Uma classe associativa pode estar ligada a associações de qualquer tipo de conectividade

Notação para uma Classe Associativa

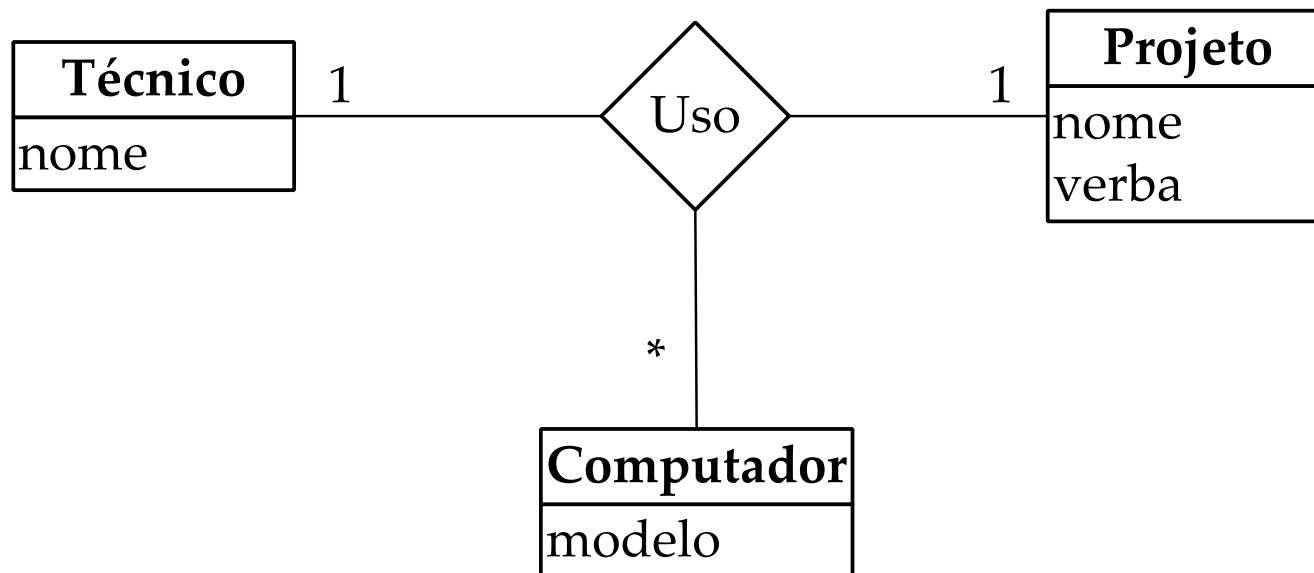
- Representada pela notação utilizada para uma classe. A diferença é que esta classe é ligada a uma associação



Associações N-árias

- São utilizadas para representar a associação existente entre objetos de n classes
- Uma associação ternária é o caso mais comum (menos raro) de associação n -ária ($n = 3$)
- Na notação da UML, as linhas de associação se interceptam em um losango

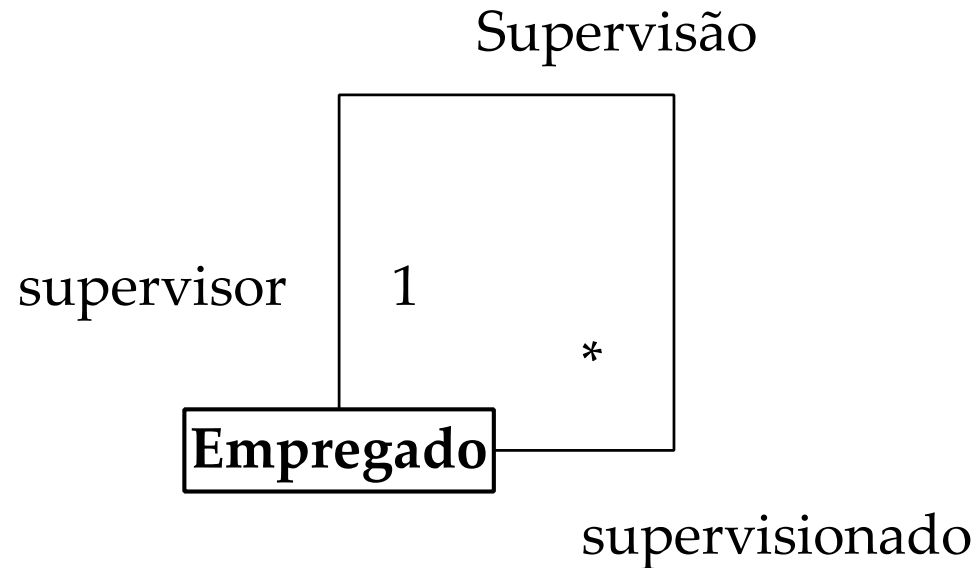
Exemplo (Associação Ternária)



Associações Reflexivas

- Associa objetos da mesma classe
- Cada objeto tem um papel distinto na associação
- A utilização de papéis é bastante importante para evitar ambigüidades na leitura da associação
- Uma associação reflexiva **não** indica que um objeto se associa com ele próprio
 - Ao contrário, indica que objetos de uma mesma classe se associam

Exemplo (Associação Reflexiva)



Agregações e Composições

- É um caso especial da associação...
...conseqüentemente, multiplicidades, participações, papéis, etc. podem ser usados igualmente
- Utilizada para representar conexões que guardam uma relação todo-parte entre si
Onde se puder utilizar uma agregação, uma associação também poderá ser utilizada!

Agregações e Composições

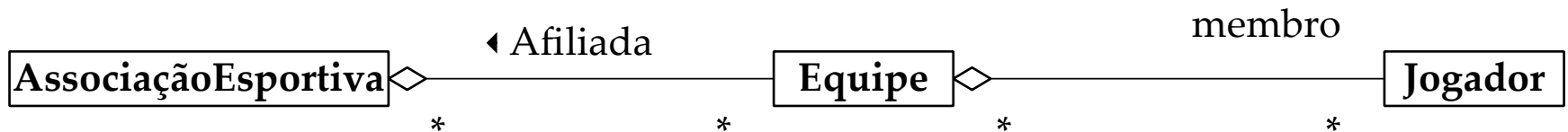
- Características particulares das agregações/composições
 - São assimétricas: se um objeto A é parte de um objeto B, B não pode ser parte de A
 - Propagam comportamento: um comportamento que se aplica a um todo automaticamente se aplica as suas partes
 - As partes são normalmente criadas e destruídas a pelo todo

Agregações e Composições

- Sejam duas classes associadas, X e Y . Se uma das perguntas a seguir for respondida com um sim, provavelmente há uma agregação onde X é todo e Y é parte
 - *X tem um ou mais Y ?*
 - *Y é parte de X ?*

Notação para Agregação

- Uma agregação é representada como uma linha que conecta as classes relacionadas, com um diamante (losango) **branco** perto da classe que representa o todo



Notação para Composição

- Uma composição é representada como uma linha que conecta as classes relacionadas, com um diamante (losango) **negro** perto da classe que representa o todo



Agregações vs Composições

- As diferenças entre agregações e composições não são bem claras, mas essas são as mais visíveis
 - Na agregação, a destruição de um objeto **todo** não implica necessariamente a destruição do objeto **parte**
 - Na composição, os objetos **parte** pertencem a único objeto **todo**, por isso quando um objeto **todo** é destruído os objetos **parte** também são

Generalizações e Especializações

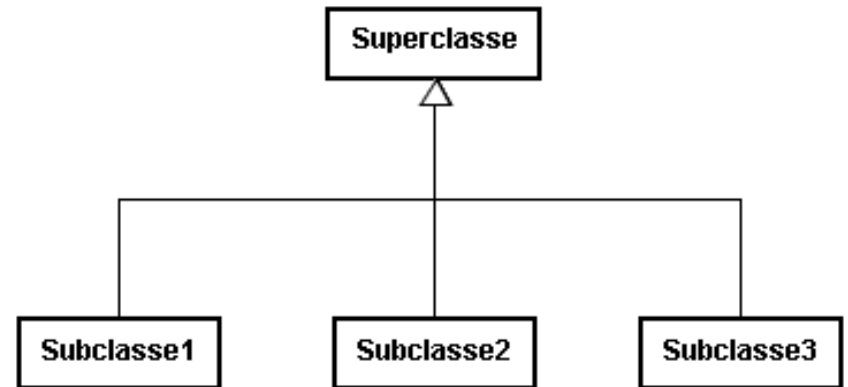
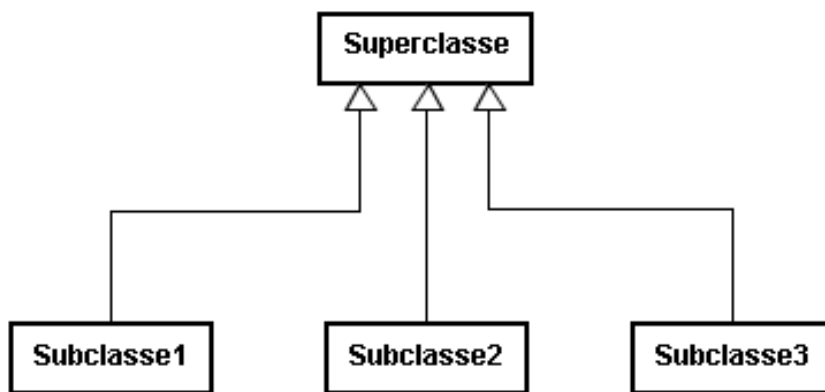
- O modelador também pode representar relacionamentos entre classes
 - Esses denotam relações de **generalidade** ou **especificidade** entre as classes envolvidas
 - Exemplo
 - O conceito *mamífero* é mais genérico que o conceito *ser humano*
 - O conceito *carro* é mais específico que o conceito *veículo*
- Esse é o chamado **relacionamento de herança**
 - Relacionamento de generalização/especialização
 - Relacionamento de gen/espec

Generalizações e Especializações

- Sejam A e B classes que se relacionam através da herança
 - Se A é uma **generalização** de B, então, B é uma **especialização** de A
- Os termos **subclasse** e **superclasse** são utilizados para denotar relações de herança
 - Subclasse é a classe que herda
 - Superclasse é a classe herdada

Generalizações e Especializações

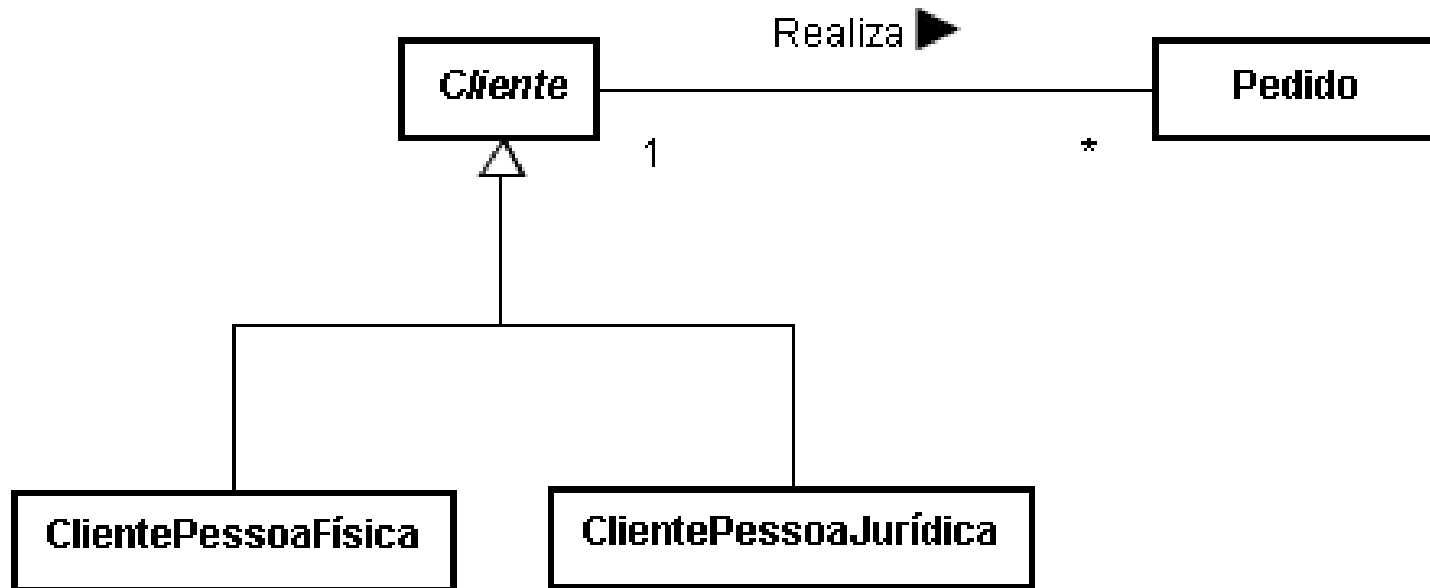
- No diagrama de classes, a herança é representada na UML por uma flecha partindo da **subclasse** em direção a **superclasse**



Herança vs Associação

- A diferença semântica entre a herança e a associação
 - A primeira trata de um relacionamento entre **classes**, enquanto que a segunda representa relacionamentos entre **instâncias** de classes (**objetos**)
 - Na associação, **objetos específicos** de uma classe se associam entre si ou com objetos específicos de outras classes
 - Exemplo
 - Herança: "Gerentes são *tipos especiais* de funcionários"
 - Associação: "Gerentes *chefiam* departamentos"

Exemplo (Herança)

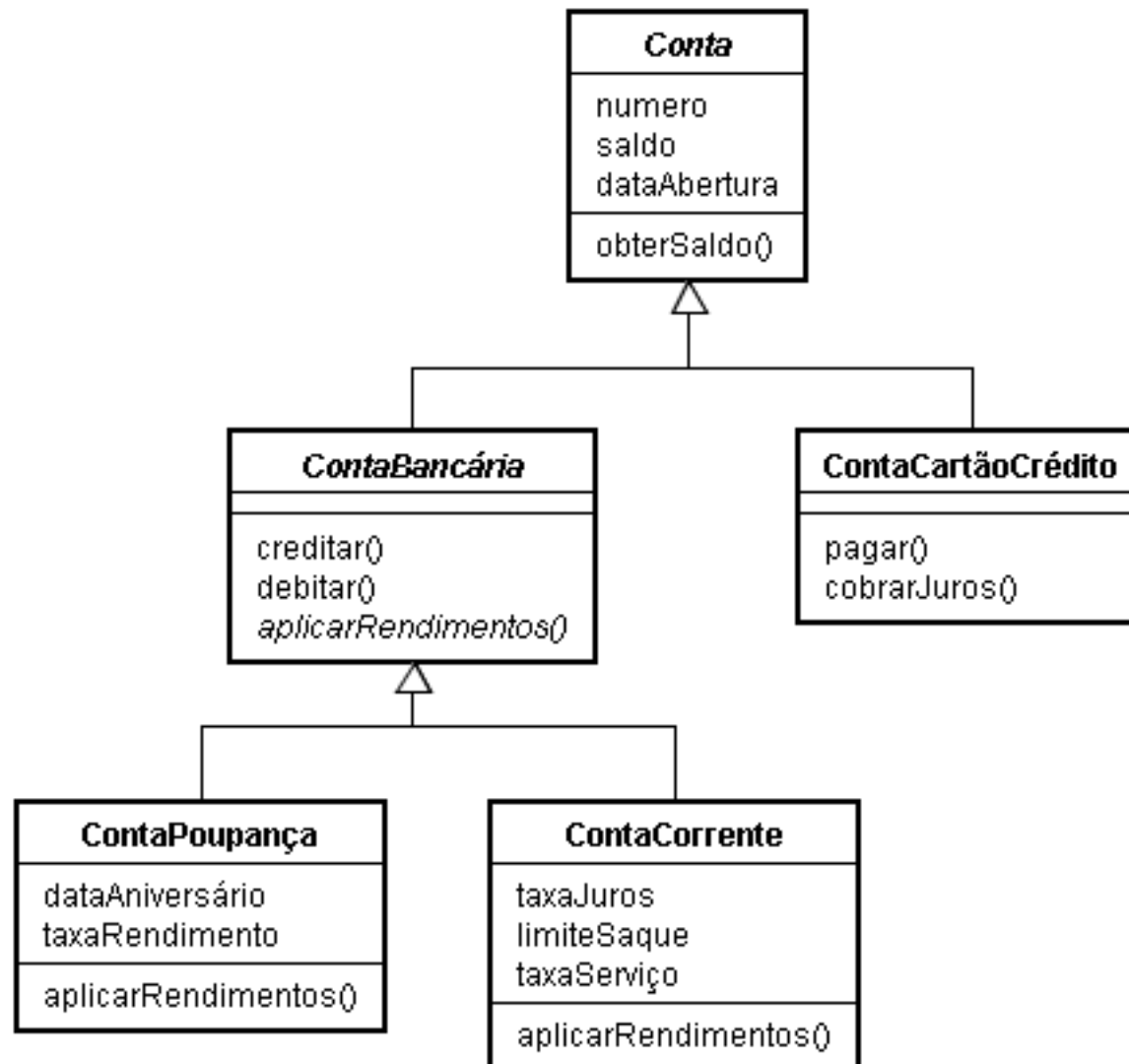


IMPORTANTE: Além das propriedades e operações, as subclasses herdam os relacionamentos!

Herança

- As principais propriedades do relacionamento de herança são
 - *Transitividade*
 - Uma classe herda tanto os **atributos**, **operações** e **relacionamentos** da super classe imediata quanto das superclasses **não imediatas**
 - *Assimetria*
 - Dadas duas classes A e B, se B é subclasse de A, então A não pode ser subclasse de B

Exemplo (Herança)



Classes Abstratas

- Usualmente, a existência de uma classe se justifica pelo fato de haver a possibilidade de gerar instâncias da mesma
 - Essas são as *classes concretas*
- No entanto, podem existir classes que não geram instâncias diretas
 - Essas são as *classes abstratas*

Classes Abstratas

- Classes abstratas são utilizadas para organizar e simplificar uma hierarquia de generalização
 - Propriedades comuns a diversas classes podem ser organizadas e definidas em uma classe abstrata a partir da qual as primeiras herdaram
- Subclasses de uma classe abstrata também podem ser abstratas, mas a hierarquia deve terminar em uma ou mais classes concretas

Notação para Classe Abstrata

- Na UML, uma classe abstrata é representada com o seu nome em **itálico**

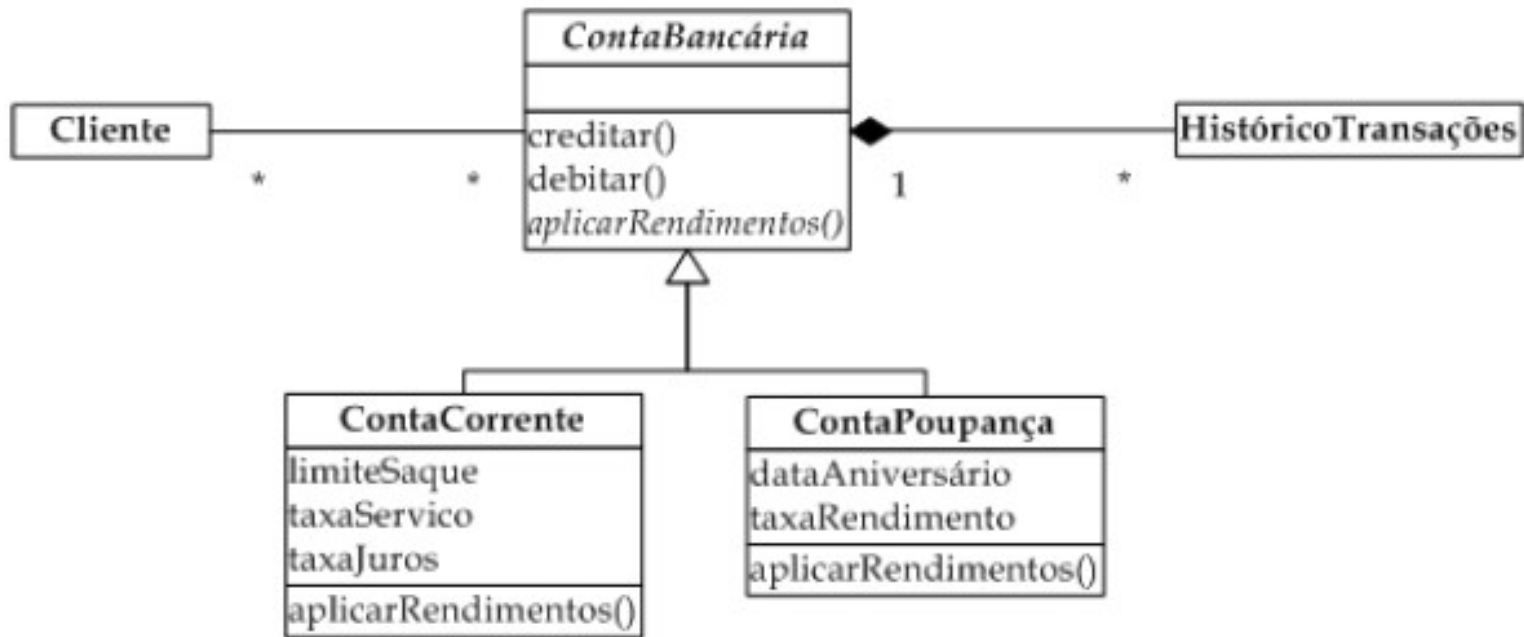


DIAGRAMA DE OBJETOS

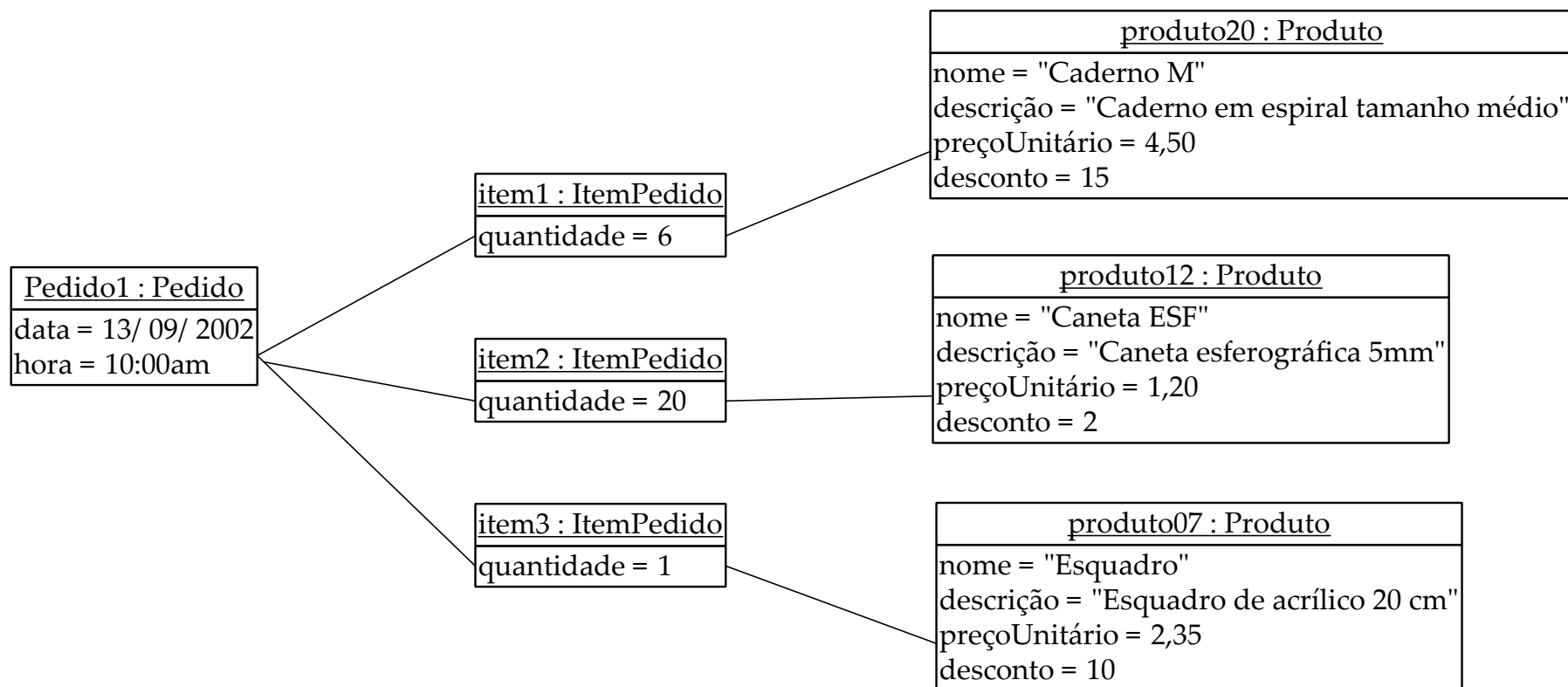
Diagrama de Objetos

- Além do diagrama de classes, a UML define um segundo tipo de diagrama estrutural, o diagrama de objetos
 - Pode ser visto com uma **instância** de diagramas de classes
 - Representa uma "fotografia" do sistema em um certo momento
- Exibe as ligações formadas entre objetos conforme estes interagem e os valores dos seus atributos

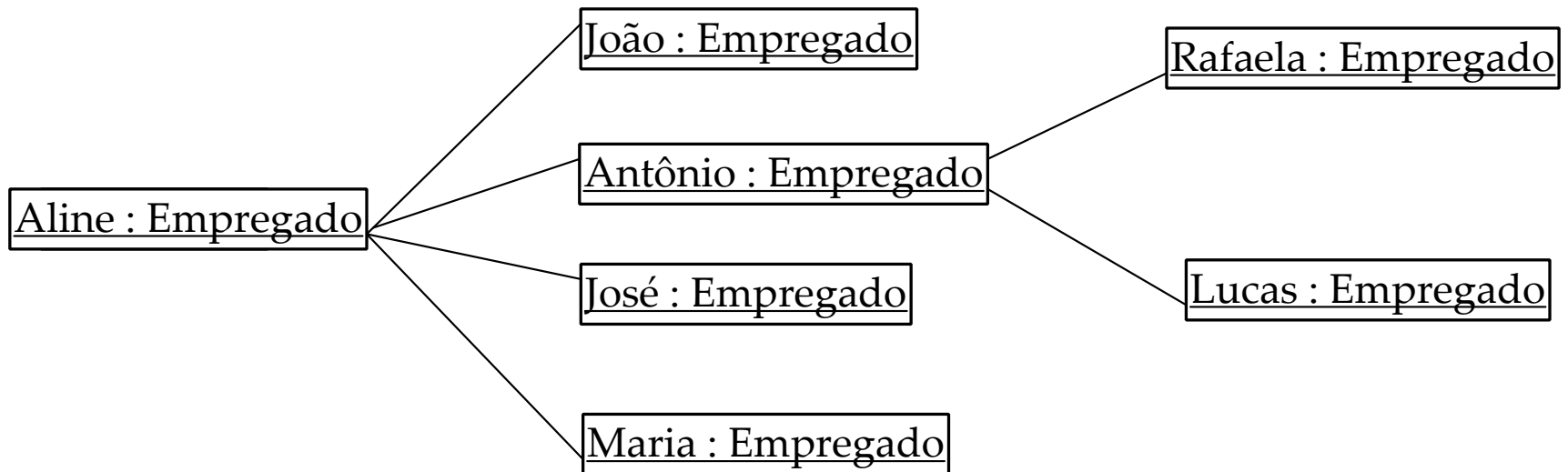
Diagrama de Objetos

Formato	Exemplo
<u>nomeClasse</u>	<u>Pedido</u>
<u>nomeObjeto: NomeClasse</u>	<u>umPedido: Pedido</u>

Exemplo (Diagrama de Objetos)



Exemplo (Diagrama de Objetos)



TÉCNICAS PARA IDENTIFICAÇÃO DE CLASSES

Identificando Classes

Apesar de todas as vantagens que a OO pode trazer ao desenvolvimento de software, um problema fundamental ainda persiste: identificar **corretamente** e **completamente** objetos (classes), atributos e operações.

Identificando Classes

- Um sistema de software orientado a objetos é composto de objetos em colaboração para realizar as tarefas deste sistema
- Por outro lado, todo objeto pertence a uma classe
- Portanto, quando se fala na identificação das classes, o objetivo na verdade é saber quais objetos irão compor o sistema

Identificando Classes

- De uma forma geral, a identificação de classes se divide em duas atividades
 - Primeiramente, classes candidatas são identificadas
 - Depois disso, são aplicados alguns princípios para eliminar classes candidatas desnecessárias

Identificar possíveis classes para um sistema não é complicado. O difícil é eliminar deste conjunto o que não é necessário!

Identificando Classes

- Várias técnicas (de uso não exclusivo) são usadas para identificar classes
 - Categorias de Conceitos
 - Análise Textual de Abbott (*Abbot Textual Analysis*)
 - Análise de Casos de Uso
 - Categorização BCE
 - Padrões de Análise (*Analisis Patterns*)
 - Identificação Dirigida a Responsabilidades

Categorias de Conceitos

- A estratégia é usar uma lista de conceitos comuns

Conceitos	Exemplos
Conceitos concretos	Edifícios, carros e salas de aula
Papéis desempenhados por seres humanos	Professores, alunos, empregados e clientes
Eventos , ou seja, ocorrências em uma data e em uma hora particulares	Reuniões, pedidos, aterrissagens e aulas
Lugares : áreas reservadas para pessoas ou coisas	Escritórios, filiais, locais de pouso e salas de aula
Organizações : coleções de pessoas ou de recursos	Departamentos, projetos, campanhas e turmas
Conceitos abstratos : princípios ou idéias não tangíveis	Reservas, vendas e inscrições

Análise Textual de Abbott

- A estratégia é identificar termos da narrativa de casos de uso e documento de requisitos que podem sugerir classes, atributos e operações
- Neste técnica, são utilizadas diversas fontes de informação sobre o sistema (todos os artefatos)
- Para cada artefato, os nomes (substantivos e adjetivos) que aparecem no mesmo são destacados
- Após isso, os sinônimos são removidos

Análise Textual de Abbott

- Cada termo remanescente se encaixa em uma dessas situações
 - O termo se torna uma classe (ou seja, são classes candidatas)
 - O termo se torna um atributo
 - O termo não tem relevância para o sistema

Análise Textual de Abbott

- Abbott também preconiza o uso de sua técnica na identificação de operações e de associações. Para isso, ele sugere que destaquemos os verbos no texto
 - Verbos de ação (e.g., calcular, confirmar, cancelar, comprar, fechar, estimar, depositar, sacar, etc.) são operações em potencial
 - Verbos com sentido de “ter” são potenciais agregações ou composições
 - Verbos com sentido de “ser” são generalizações em potencial
 - Demais verbos são associações em potencial

Análise Textual de Abbott

Parte do Texto	Componente	Exemplo
Nome próprio	Objeto	Marcos Antonio
Nome simples	Classe	professor
Verbos de ação	Operação	registrar
Verbo ser	Herança	é um
Verbo ter	Todo-parte	tem um

Análise Textual de Abbott

- Uma desvantagem é que seu resultado (as classes candidatas identificadas) depende de os artefatos utilizados como fonte serem completos
 - Dependendo do estilo que foi utilizado para escrever esse documento, essa técnica pode levar à identificação de diversas classes candidatas que não gerarão classes
 - A análise do texto de um documento pode não deixar explícita uma classe importante para o sistema
 - Em linguagem natural, as variações lingüísticas e as formas de expressar uma mesma idéia são bastante numerosas

Análise de Caso de Uso

- Essa técnica é também chamada de **identificação dirigida por casos de uso**, e é um caso particular da técnica de Abbott
- Técnica preconizada pelo Processo Unificado
- Nesta técnica, o MCU é utilizado como ponto de partida

Análise de Caso de Uso

Premissa: um caso de uso corresponde a um comportamento específico do sistema. Esse comportamento somente pode ser produzido por objetos que compõem o sistema (a realização é **responsabilidade** de um conjunto de objetos do sistema).

Ação: com base nisso, o modelador aplica a técnica de análise dos casos de uso para identificar as classes necessárias à produção do comportamento que está documentado na descrição do caso de uso.

Análise de Caso de Uso

Procedimento de Aplicação

1. Suplemente as descrições dos casos de uso. Esse passo envolve complementar a descrição de casos de uso com o objetivo de torná-los completos e facilitar a identificação de todas as classes envolvidas no mesmo.
2. Para cada caso de uso
 - a) Identifique classes a partir do comportamento do caso de uso
 - b) Distribua o comportamento do caso de uso pelas classes identificadas
3. Para cada classe de análise resultante
 - a) Descreva suas responsabilidades
 - b) Descreva seus atributos e associações
4. Unifique as classes de análise identificadas em um ou mais diagramas de classes

Na aplicação deste procedimento, podemos utilizar a categorização **BCE** (*Boundary-Control-Entity, em português Fronteira-Controle-Entidade*).

Categorização BCE

- Na categorização BCE, os objetos de um sistema são agrupados de acordo com o tipo de responsabilidade a eles atribuída

Elemento	Descrição
Objetos de Entidade	Usualmente objetos do domínio do problema
Objetos de Fronteira	Usualmente atores interagem com esses objetos
Objetos de Controle	Servem como intermediários entre objetos de fronteira e de entidade, definindo o comportamento de um caso de uso específico

Categorização BCE

- BCE foi proposta por Ivar Jacobson em 1992
 - Possui correspondência (mas não equivalência!) com o framework *model-view-controller* (MVC)
 - Ligação entre análise (o que; problema) e projeto (como; solução)
- Estereótipos na UML
 - «boundary», «entity» e «control»



Objetos de Entidade



- Repositório para informações e as regras de negócio manipuladas pelo sistema
 - Representam conceitos do domínio do negócio
- Características
 - Normalmente armazenam informações persistentes
 - Várias instâncias da mesma entidade existindo no sistema
 - Participam de vários casos de uso e têm ciclo de vida longo
- Exemplo
 - Um objeto *Pedido* participa dos casos de uso *Realizar Pedido* e *Atualizar Estoque*. Este objeto pode existir por diversos anos ou mesmo tanto quanto o próprio sistema

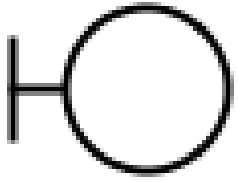


Objetos de Entidade



Atribuições Típicas

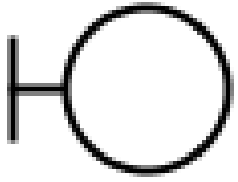
- Informar valores de seus atributos a objetos de controle
- Realizar cálculos simples, normalmente com a colaboração de objetos de entidade associados através de agregações
- Criar e destruir objetos parte (considerando que o objeto de entidade seja um objeto todo de uma agregação)



Objetos de Fronteira



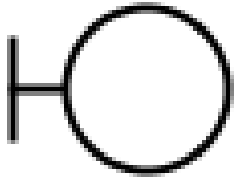
- Realizam a comunicação do sistema com os atores
 - Traduzem os eventos gerados por um ator em eventos relevantes para o sistema (**eventos de sistema**)
 - Também são responsáveis por apresentar os resultados de uma interação dos objetos em algo inteligível pelo ator



Objetos de Fronteira



- Um objeto de fronteira existe para que o sistema se comunique com o mundo exterior. Por consequência, estes objetos são altamente dependentes do ambiente
- Há dois tipos principais de objetos de fronteira
 - Os que se comunicam com o usuário (atores humanos)
 - Exemplo: relatórios, páginas HTML, interfaces gráfica desktop, etc.
 - Os que se comunicam com atores não-humanos (outros sistemas ou dispositivos)
 - Exemplo: protocolos de comunicação

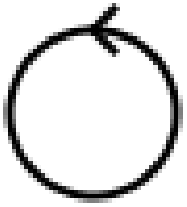


Objetos de Fronteira



Atribuições Típicas

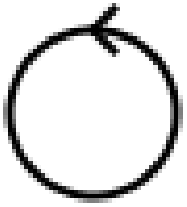
- Notificar aos objetos de controle de eventos gerados externamente ao sistema
- Notificar aos atores do resultado de interações entre os objetos internos



Objetos de Controle



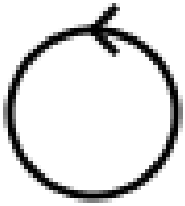
- São a "ponte de comunicação" entre objetos de fronteira e objetos de entidade
- Responsáveis por controlar a lógica de execução correspondente a um caso de uso
- Decidem o que o sistema deve fazer quando um evento de sistema ocorre
 - Eles realizam o controle do processamento
 - Agem como gerentes (controladores) dos outros objetos para a realização de um caso de uso
- Traduzem **eventos de sistema** em operações que devem ser realizadas pelos demais objetos



Objetos de Controle



- São bastante acoplados à lógica da aplicação.
- Ao contrário dos objetos de entidade e de fronteira, objetos de controle são tipicamente ativos
 - Consultam informações e requisitam serviços de outros objetos



Objetos de Controle



Atribuições Típicas

- Realizar monitorações, a fim de responder a eventos externos ao sistema (gerados por objetos de fronteira)
- Coordenar a realização de um caso de uso através do envio de mensagens a objetos de fronteira e objetos de entidade
- Assegurar que as **regras do negócio** estão sendo seguidas corretamente
- Coordenar a criação de associações entre objetos de entidade

Importância da Categorização BCE

- A categorização BCE parte do princípio de que cada objeto em um sistema é especialista em realizar um de três tipos de tarefa, a saber
 - Se comunicar com atores (*fronteira*)
 - Manter as informações (*entidade*)
 - Coordenar a realização de um caso de uso (*controle*)

A categorização BCE é uma "receita de bolo" para identificar objetos participantes da realização de um caso de uso!

Importância da Categorização BCE

- A importância dessa categorização está relacionada à capacidade de **adaptação** a eventuais mudanças
 - Se cada objeto tem atribuições específicas dentro do sistema, mudanças podem ser menos complexas e mais localizadas
 - Uma modificação em uma parte do sistema tem menos possibilidades de resultar em mudanças em outras partes

Importância da Categorização BCE

Tipo de Mudança	Onde Mudar
Mudanças em relação à interface gráfica, ou em relação à comunicação com outros sistemas	Fronteira
Mudanças nas informações manipuladas pelo do sistema	Entidade
Mudanças em funcionalidades complexas (lógica do negócio)	Controle

Importância da Categorização BCE

- **Exemplo:** vantagem de separação de responsabilidades em um sistema para uma loja de aluguel de carros.

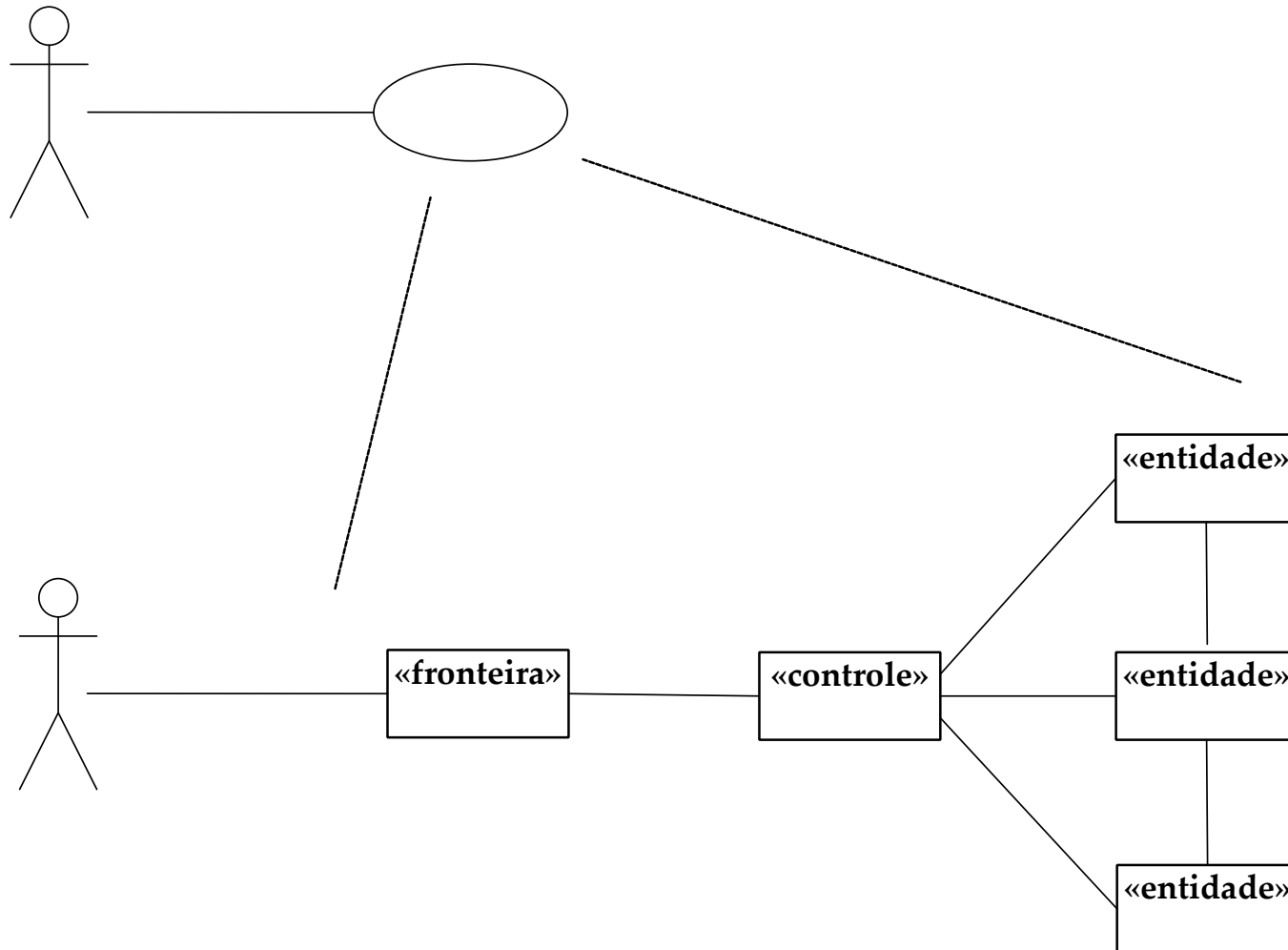
✓ Se o sistema tiver que ser atualizado para que seus usuários possam utilizá-lo pela Internet, a lógica da aplicação não precisaria de modificações.

✓ Considerando a lógica para calcular o valor total das locações feitas por um cliente: se esta lógica estiver encapsulada em uma classe de controle, somente esta classe precisaria de modificação.

Visões de Classes Participantes

- Uma Visão de Classes Participantes (VCP, original em inglês *View Of Participating Classes*) é um diagrama das classes cujos objetos participam da realização de determinado caso de uso
 - É uma recomendação do UP (Unified Process)
 - "definir uma VCP por caso de uso"
- Em uma VCP, são representados objetos de fronteira, de entidade e de controle para um caso de uso particular
- Uma VCP é definida através da utilização da categorização BCE previamente descrita

Estrutura de uma VCP

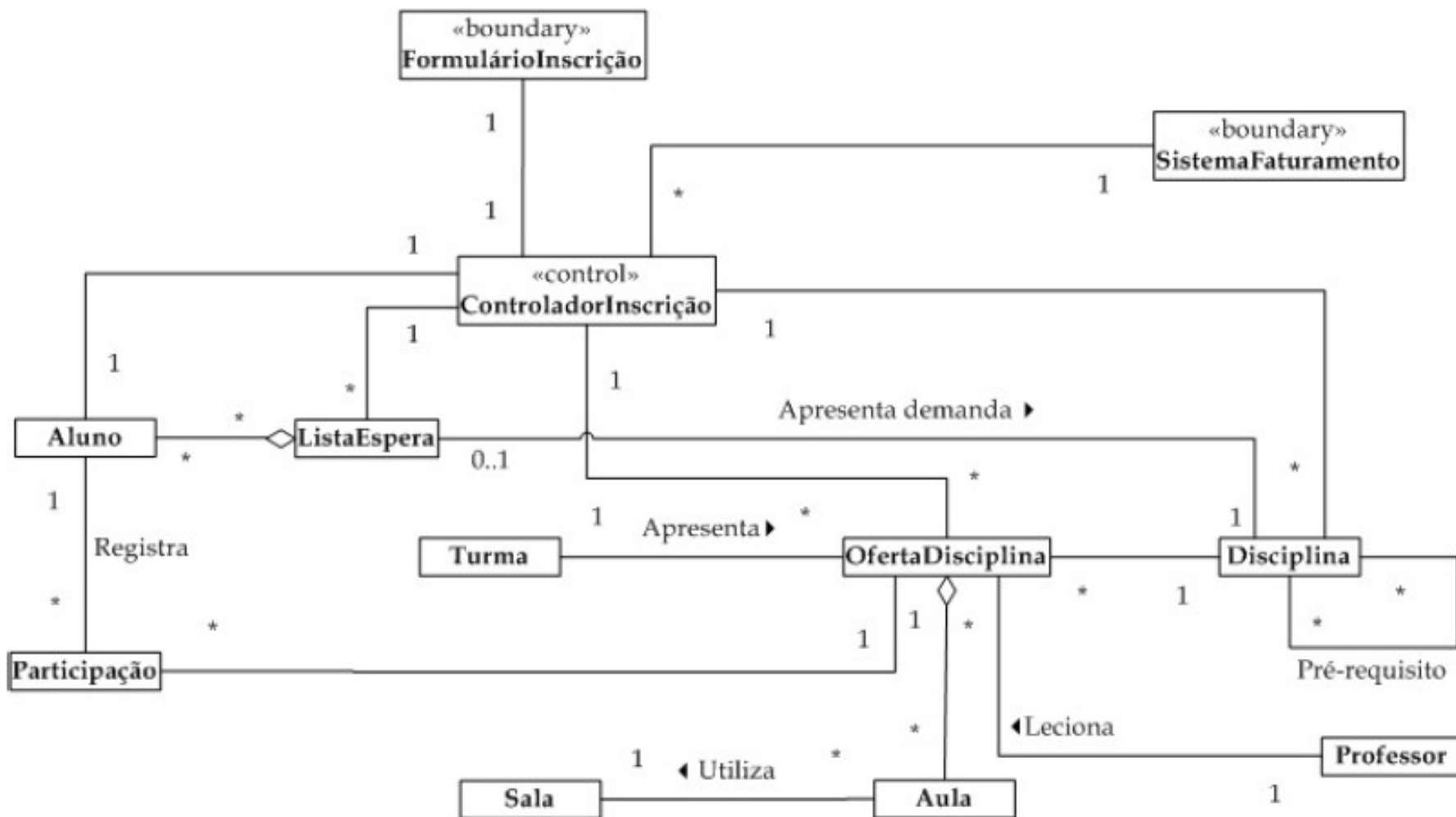


Construção de VCP

Para cada Caso de Uso

1. Adicione uma fronteira para cada elemento de interface gráfica principal, tais com uma tela (formulário) ou relatório
2. Adicione uma fronteira para cada ator não-humano (por exemplo, outro sistema)
3. Adicione um ou mais controladores para gerenciar o processo de realização do caso de uso
4. Adicione uma entidade para cada conceito do negócio (esses objetos são originários do modelo conceitual)

Exemplo (VCP – Realizar Inscrição)



Regras Estruturais de uma VCP

- Durante a atividade de **análise**, use as regras a seguir para definir a VCP para um caso de uso
 - Atores somente podem interagir com objetos de fronteira
 - Objetos de fronteira somente podem interagir com controladores e atores
 - Objetos de entidade somente podem interagir (receber requisições) com controladores
 - Controladores somente podem interagir com objetos de fronteira e objetos de entidade, e com (eventuais) outros controladores

Identificação Dirigida a Responsabilidade

- Nesta técnica, a ênfase está na identificação de classes a partir de seus comportamentos relevantes para o sistema
 - O esforço do modelador recai sobre a identificação das *responsabilidades* que cada classe deve ter dentro do sistema
- Método foi proposto por Rebecca Wirfs-Brock e Brian Wilkerson
 - "*O método dirigido a responsabilidades enfatiza o encapsulamento da estrutura e do comportamento dos objetos.*"

Identificação Dirigida a Responsabilidade

- Essa técnica enfatiza o princípio do encapsulamento
 - A ênfase está na identificação das responsabilidades de uma classe que são úteis externamente à mesma
 - Os detalhes internos à classe (*como* ela faz para cumprir com suas responsabilidades) devem ser abstraídos

“O método dirigido a responsabilidades enfatiza o encapsulamento da estrutura e do comportamento dos objetos.”

Responsabilidades e Colaboradores

- Em sistemas OO, objetos encapsulam tanto dados quanto comportamento
- O comportamento de um objeto é definido de tal forma que ele possa cumprir com suas ***responsabilidades***
- Uma responsabilidade é uma obrigação que um objeto tem para com o sistema no qual ele está inserido
 - Através delas, um objeto colabora (ajuda) com outros para que os objetivos do sistema sejam alcançados

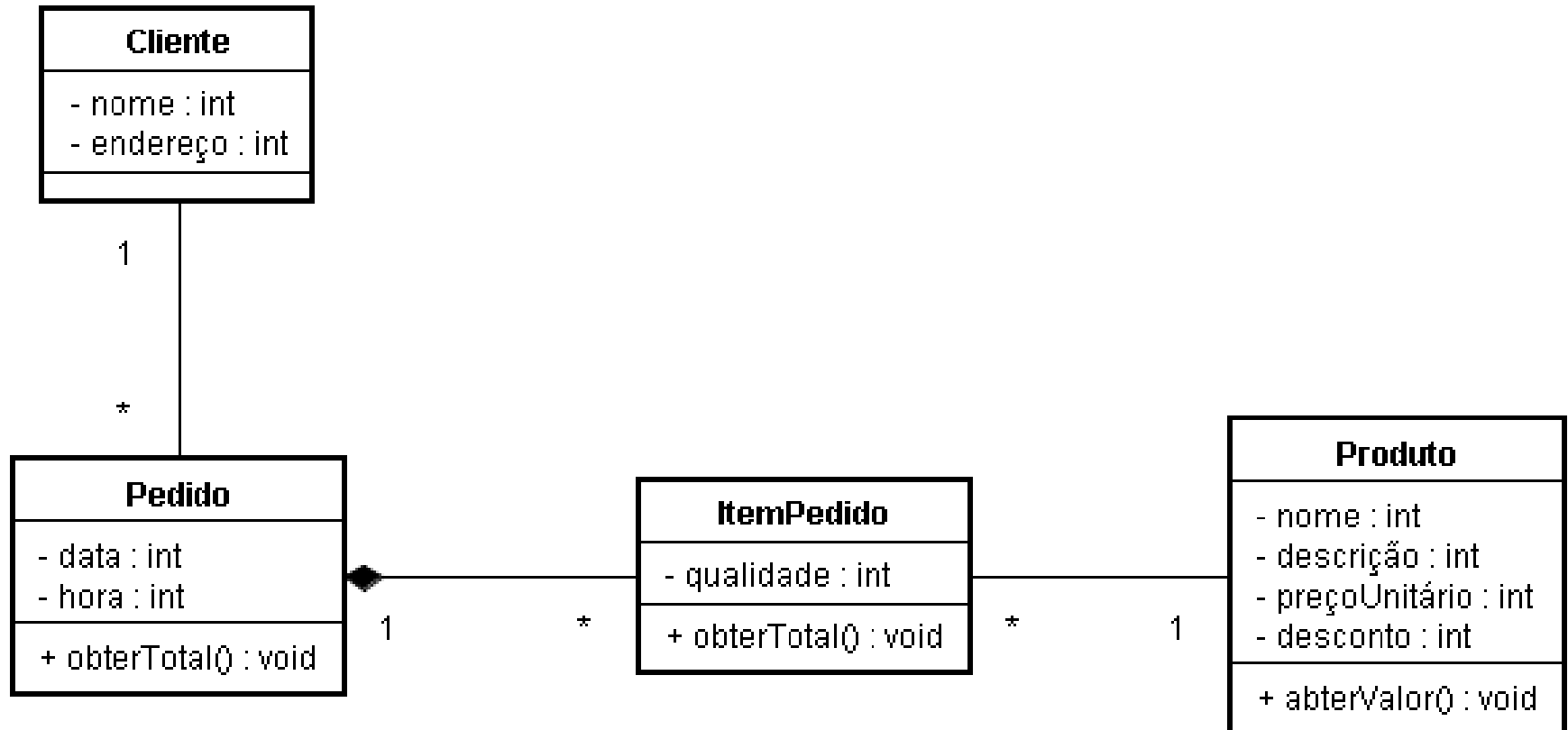
Responsabilidades e Colaboradores

- Na prática, uma responsabilidade é alguma coisa que um objeto *conhece* ou *faz (sozinho ou não)*
 - Um objeto Cliente *conhece* seu nome, seu endereço, seu telefone, etc
 - Um objeto Pedido *conhece* sua data de realização e sabe *fazer* o cálculo do seu total
- Se um objeto tem uma responsabilidade com a qual não pode cumprir sozinho, ele deve requisitar ***colaborações*** de outros objetos

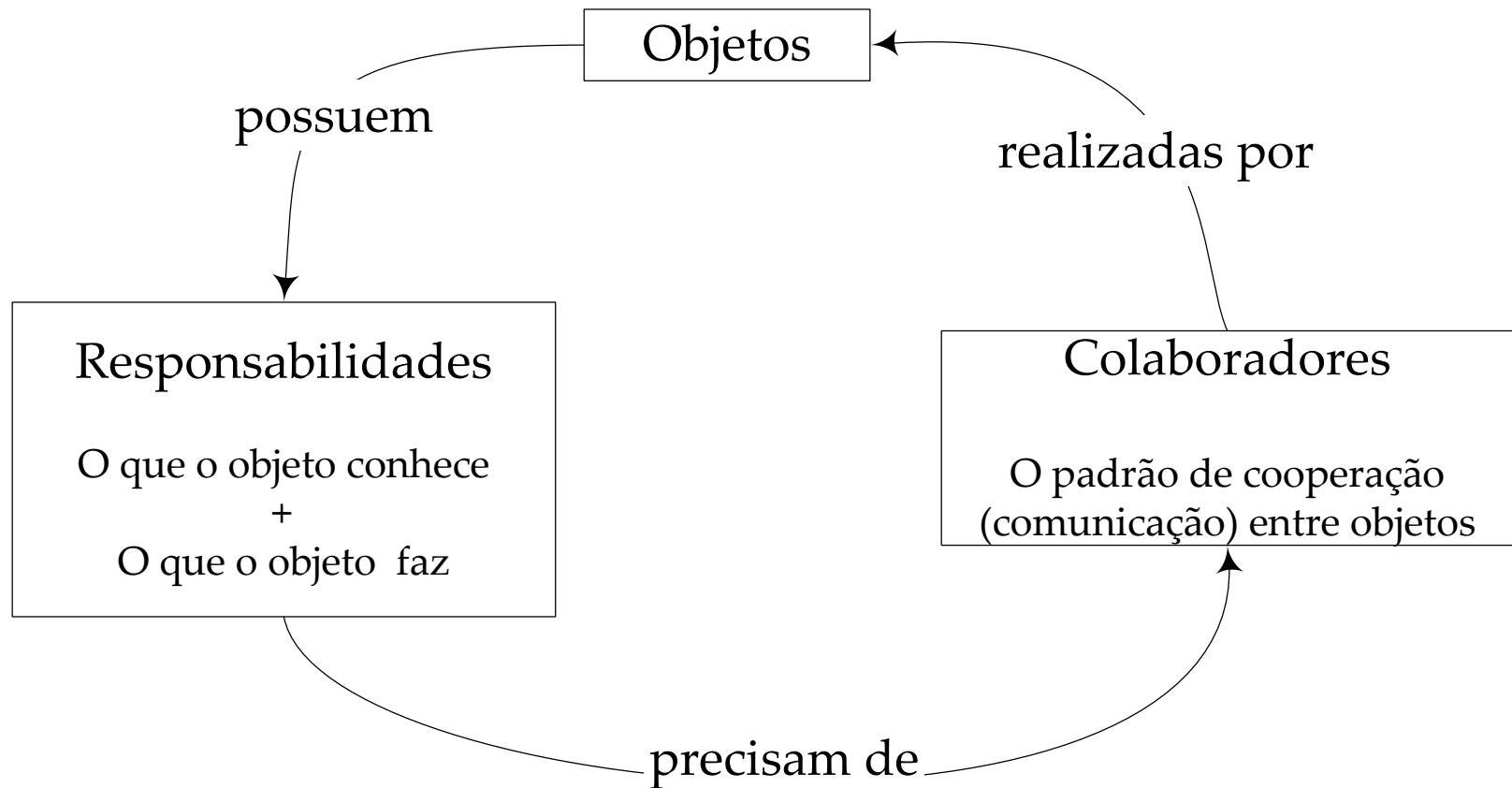
Responsabilidades e Colaboradores

- Um exemplo: quando a impressão de uma fatura é requisitada em um sistema de vendas, vários objetos precisam colaborar
 - Um objeto Pedido pode ter a responsabilidade de fornecer o seu valor total
 - Um objeto Cliente fornece seu nome
 - Cada ItemPedido informa a quantidade correspondente e o valor de seu subtotal
 - Os objetos Produto também colaboraram fornecendo seu nome e preço unitário

Responsabilidades e Colaboradores



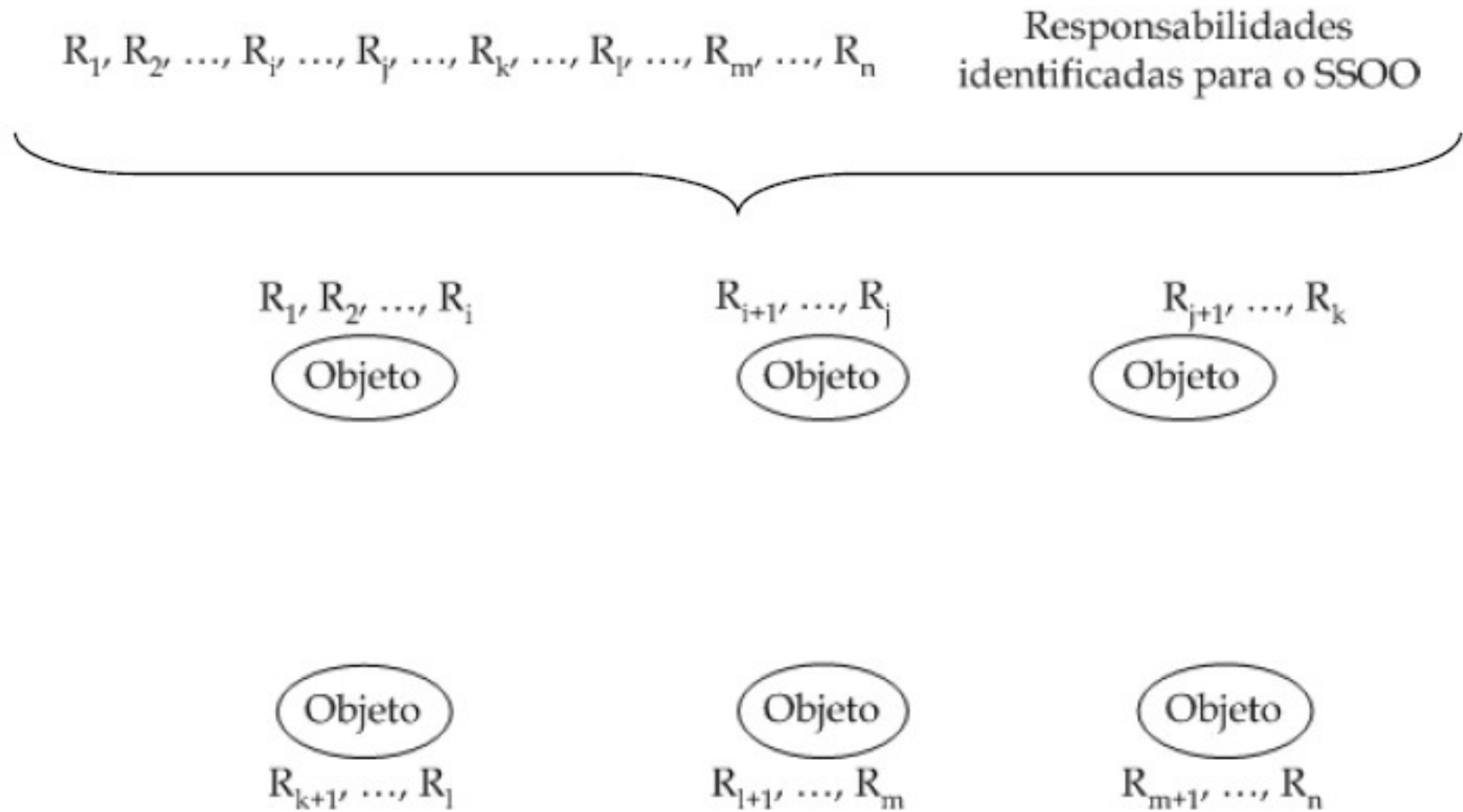
Responsabilidades e Colaboradores



Modelagem CRC

- Essa técnica é denominada *modelagem CRC (Class, Responsibility, Collaboration)*
- A modelagem CRC foi proposta em 1989 por Kent Beck e Ward Cunningham
 - Em princípio, essa técnica de modelagem foi proposta como uma forma de ensinar o paradigma OO a iniciantes
 - Contudo, sua simplicidade de notação a tornou particularmente interessante para ser utilizada na identificação de classes

Modelagem CRC



Sessão CRC

- Para aplicar essa técnica, esses profissionais se reúnem em uma sala, onde tem início uma ***sessão CRC***
- Uma sessão CRC é uma reunião que envolve cerca de meia dúzia de pessoas
- Entre os participantes estão especialistas de domínio, projetistas, analistas e o moderador da sessão

Sessão CRC

- A cada pessoa é entregue um cartão de papel que mede aproximadamente 10cm x 15cm
- Uma vez distribuídos os cartões pelos participantes, um conjunto de cenários de determinado caso de uso é selecionado
- Então, para cada cenário desse conjunto, uma sessão CRC é iniciada
 - Se o caso de uso não for tão complexo, ele pode ser analisado em uma única sessão

Sessão CRC

- Um cartão CRC é dividido em várias partes
 - Na parte superior do cartão, aparece o nome de uma classe
 - A parte inferior do cartão é dividida em duas colunas
 - Na coluna da esquerda, o indivíduo ao qual foi entregue o cartão deve listar as responsabilidades da classe
 - Na coluna direita, o indivíduo deve listar os nomes de outras classes que colaboram com a classe em questão para que ela cumpra com suas responsabilidades

Modelagem CRC

- Normalmente já existem algumas classes candidatas para determinado cenário
 - Identificadas através de outras técnicas
- A sessão CRC começa com algum dos participantes simulando o ator primário que dispara a realização do caso de uso

Modelagem CRC

- Na medida em que esse participante simula a interação do ator com o sistema, os demais participantes encenam a colaboração entre objetos que ocorre internamente ao sistema
- Através dessa encenação dos participantes da sessão CRC, as classes, responsabilidades e colaborações são identificadas

Estrutura do Cartão CRC

<Nome da Classe>	
Responsabilidades	Colaboradores
1ª responsabilidade	1º colaborador
2ª responsabilidade	2º colaborador
...	...
n -ésima responsabilidade	<i>m</i> -ésimo colaborador

Exemplo (Cartão CRC)

ContaBancária	
Responsabilidades	Colaboradores
1ª Conhecer o seu cliente	1º Cliente
2ª Conhecer o seu nome	2º Transação
3ª Conhecer o seu saldo	
4ª Manter um histórico de transações	
5ª Aceitar saques e depósitos	

Dicas para Atribuição de Responsabilidades

- Associar responsabilidades com base na especialidade da classe
- Distribuir a inteligência do sistema
- Agrupar as responsabilidades conceitualmente relacionadas
- Evitar responsabilidades redundantes

Padrões de Análise

- Após produzir diversos modelos para um mesmo domínio, é natural que um modelador comece a identificar **características comuns** entre esses modelos
- Em particular, um mesmo conjunto de classes ou colaborações entre objetos costuma **recorrer** (com algumas pequenas diferenças)
 - Quantos modelos de classes já foram construídos que possuem os conceitos Cliente, Produto, Fornecedor, Departamento, etc?

Padrões de Análise

- A identificação de características comuns acontece em consequência do **ganho de experiência** do modelador em determinado domínio de problema
- Ao reconhecer processos e estruturas comuns em um domínio, o modelador pode descrever (catalogar) a **essência** dos mesmos, dando origem a *padrões de software*

Padrões de Análise

- Quando o problema correspondente ao descrito em um padrão de software acontecer novamente, um modelador pode utilizar a solução descrita no catálogo
- A aplicação desse processo permite que o desenvolvimento de determinado aspecto de um sistema seja feito de forma **mais rápida e menos suscetível a erros**

Padrões de Análise

- Um padrão de software pode então ser definido como uma descrição essencial de um problema recorrente no desenvolvimento de software

"Cada padrão descreve um problema que ocorre freqüentemente no nosso ambiente, e então descreve o núcleo de uma solução para tal problema. Esse núcleo pode ser utilizado um milhão de vezes, sem que haja duas formas de utilização iguais." (Christopher Alexander)



Padrões de Análise

- Padrões de software podem ser utilizados em diversas atividades e existem em diversos níveis de abstração
 - Padrões de Análise (Analysis Patterns)
 - Padrões de Projeto (Design Patterns)
 - Padrões Arquiteturais (Architectural Patterns)
 - Idiomas (Idioms)
 - Anti-padrões (Anti-patterns)
- Um **padrão de análise** normalmente é composto, dentre outras partes, de um fragmento de diagrama de classes que pode ser customizado para uma situação de modelagem em particular

CONSTRUÇÃO DO MODELO DE CLASSES

Propriedades de uma Classe

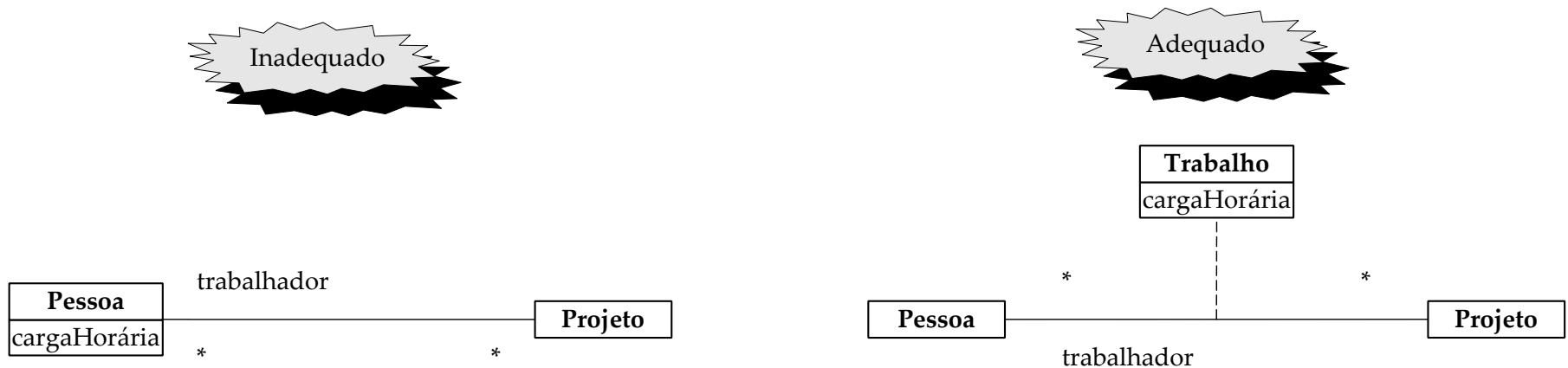
- Uma responsabilidade de conhecer é mapeada para um ou mais atributos
- Operações de uma classe são um modo mais detalhado de explicitar as responsabilidades de fazer
 - Uma operação pode ser vista como uma contribuição da classe para uma tarefa mais complexa representada por um caso de uso
 - Uma definição mais completa das operações de uma classe só pode ser feita após a construção dos **diagramas de interação**

Definição de Associações e Agregações

- O fato de uma classe possuir colaboradores indica que devem existir relacionamentos entre estes últimos e a classe
 - Isto porque um objeto precisa conhecer o outro para poder lhe fazer requisições
 - Portanto, para criar associações, verifique os colaboradores de uma classe
- O raciocínio para definir associações reflexivas, ternárias e agregações é o mesmo

Definição de Classes Associativas

- Surgem a partir de responsabilidades de conhecer que o modelador não conseguiu atribuir a alguma classe
 - Mais raramente, de responsabilidades *de fazer*



Documento do Modelo de Classes

- As responsabilidades e colaboradores mapeados para elementos do modelo de classes devem ser organizados em um diagrama de classes e documentados, resultando no ***modelo de classes de análise***
- Podem ser associados estereótipos predefinidos às classes identificadas
 - <<fronteira>>
 - <<entidade>>
 - <<controle>>



Documento do Modelo de Classes

- A construção de um único diagrama de classes para todo o sistema pode resultar em um diagrama bastante complexo. Um alternativa é criar uma ***visão de classes participantes*** (VCP) para cada caso de uso
- Em uma VCP, são exibidos os objetos que participam de um caso de uso
- As VCPs podem ser reunidas para formar um único diagrama de classes para o sistema como um todo

Documento do Modelo de Classes

- O modelador pode optar por “esconder” as classes de fronteira ou até mesmo as classes de controle
 - Uma ferramenta CASE que dê suporte a essa operação seria de grande ajuda para a equipe de desenvolvimento

Além do diagrama, as classes devem ser documentadas textualmente.

MODELO DE CLASSES NO PROCESSO I&I

Modelo de Classes do Processo I&I

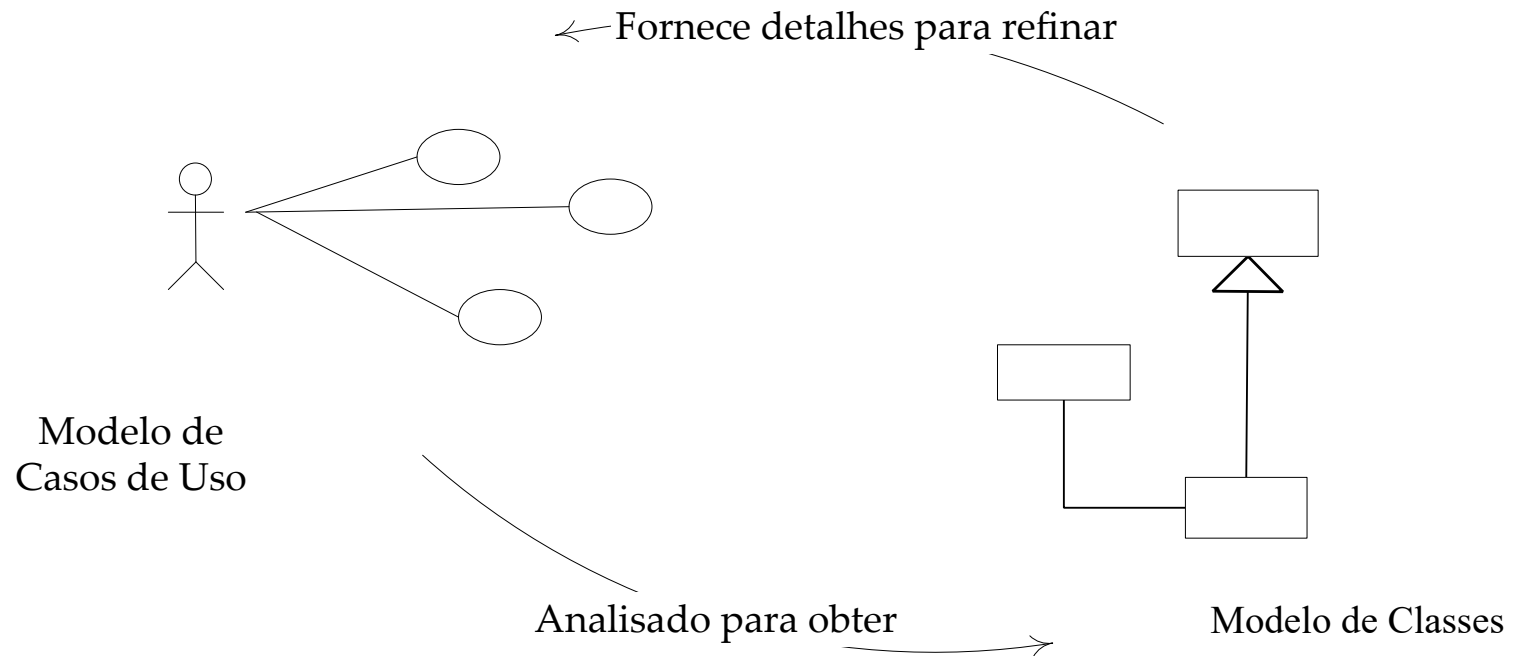
- Em um desenvolvimento dirigido a casos de uso, após a descrição dos casos de uso, é possível iniciar a identificação de classes
- As classes identificadas são refinadas para retirar inconsistências e redundâncias
- As classes são documentadas e o diagrama de classes inicial é construído, resultando no modelo de classes de domínio

Modelo de Classes do Processo I&I

- Inconsistências nos modelos devem ser verificadas e corrigidas
- As construções do modelo de casos de uso e do modelo de classes são retroativas uma sobre a outra
 - Na realização de uma sessão CRC, novos casos de uso podem ser identificados
 - Pode-se identificar a necessidade de modificação de casos de uso preexistentes

Modelo de Classes do Processo I&I

- Detalhes são adicionados aos modelos, à medida que o problema é entendido



Referências

- BEZERRA, E. Princípios de Análise e Projeto de Sistemas com UML. 2ª ed. Rio de Janeiro: Elsevier, 2007.
- FOWLER, M. 3. UML Essencial. 3. ed. Porto Alegre: Bookman, 2007.