UnB Departamento de Ciência da Computação

Análise OO

O engenheiro de software amador está sempre à procura da mágica, 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

Edison Ishikawa, D. Sc.

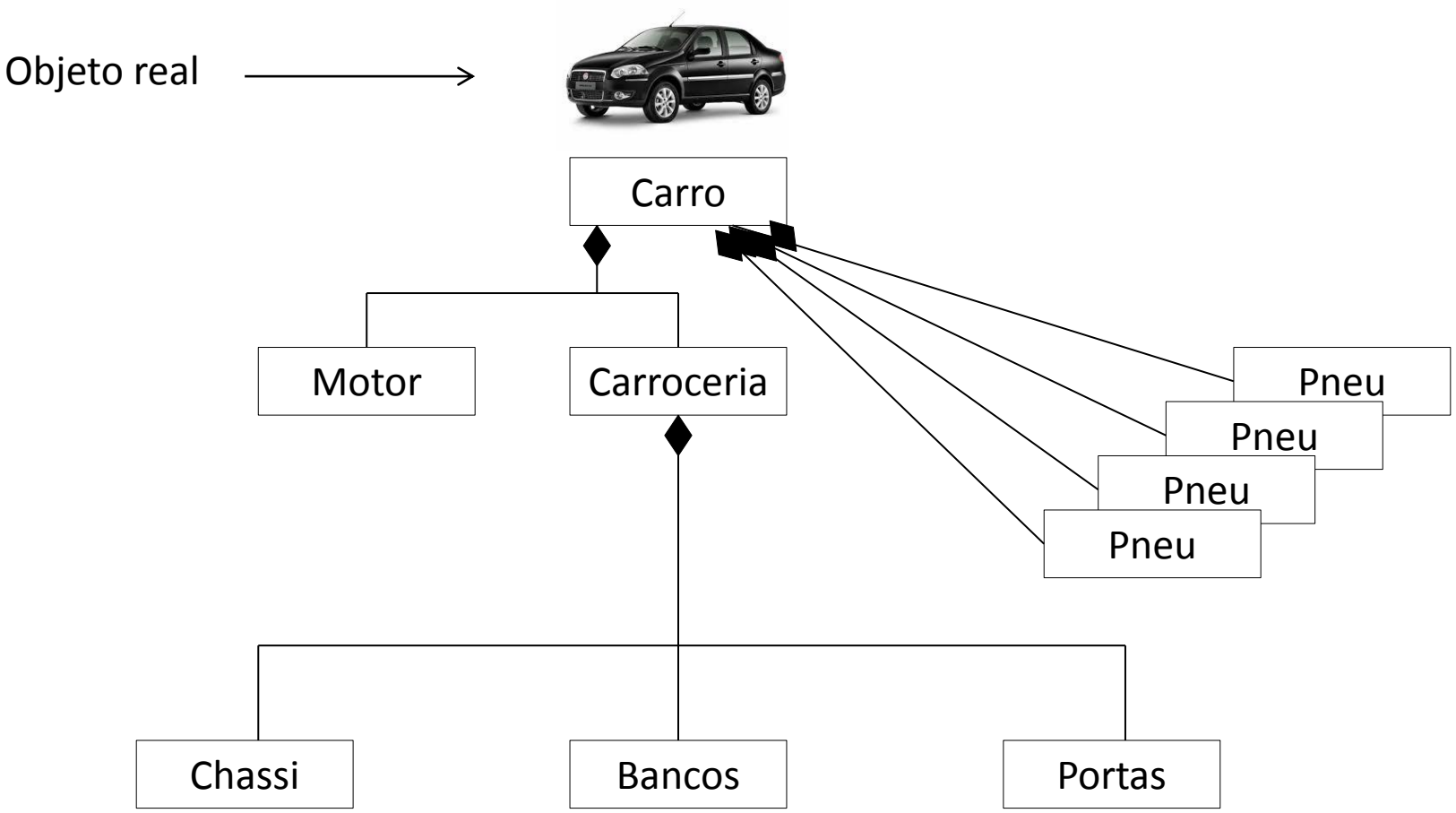
Introdução

- Objetivo
  - Aprender a modelar Classes de Análise

Sumário

- Introdução
- Estágios de Modelos de Classe
- Diagrama de Classes
- Diagrama de Objetos
- Técnicas para identificação de classes
- Padrões de análise
- Construção do modelo de classes
- Modelo de classes no processo de desenvolvimento
- Referências

Orientação a Objeto



Orientação a Objeto

- Abstração
  - pelo princípio da abstração, nós isolamos os objetos que queremos representar do ambiente complexo em que se situam, e nesses objetos representamos somente as características que são relevantes para o problema em questão".
    - Identidade
    - Propriedades
    - Métodos
- Encapsulamento
- Herança
- Polimorfismo

Orientação a Objeto

- Abstração
- Encapsulamento
  - ocultar partes da implementação e desta forma construir softwares que atinjam suas funcionalidades e escondam os detalhes de implementação do mundo exterior
  - funcionam como uma caixa preta, sabe-se da sua interface externa, mas não precisa se preocupar com o que acontece dentro dela
  - “pessoas que usam os objetos não precisam se preocupar em saber como eles são constituídos internamente acelerando o tempo de desenvolvimento”
  - Quando ligamos a televisão não sabemos o que ocorre por trás do botão “Ligar”, podemos dizer que os métodos que ligam efetivamente a televisão estão encapsulados.
  - níveis de acessos.
    - **Público** – todos os objetos tem acesso;
    - **Protegido** – o acesso é apenas para instância, no caso para o objeto e todas as subclasses;
    - **Privado** – o acesso é apenas para o objeto da instância.
- Herança
- Polimorfismo

Orientação a Objeto

- Abstração
- Encapsulamento
- Herança
  - A reutilização de códigos nas linguagens orientadas a objetos é uma característica que otimiza o desenvolvimento de um aplicativo tanto em economia de tempo, quanto em número de linhas de código.
  - Podemos criar uma classe com todas as características dos animais mamíferos. E uma outra classe para felinos que herda as características dos mamíferos, já que todo felino é um mamífero. Se precisarmos criar uma classe para os caninos, esta também herda as características dos mamíferos, já que todo canino é um mamífero. Assim os códigos comuns às classes Caninos e Felinos só seram escritos uma vez, economizando códigos e tempo de desenvolvimento.
- Polimorfismo

Orientação a Objeto

- Abstração
- Encapsulamento
- Herança
- Polimorfismo
  - O polimorfismo está diretamente ligado à hereditariedade das classes, este trabalha com a redeclararão de métodos herdados, ou seja, os métodos têm a mesma assinatura (têm o mesmo nome), mas a forma de implementação utilizada diferem o da superclasse
  - “de sua própria maneira, o polimorfismo é o distúrbio das múltiplas personalidades do mundo do software, pois um único nome pode expressar muitos comportamentos diferentes (Sintes)”.

Introdução

- Visão Externa
  - Visão de casos de uso
    - Perspectiva do sistema a partir de um ponto de vista externo
    - Externamente atores visualizam resultados, confirmam requisições, etc
- Visão Interna

Introdução

- Visão Externa
- Visão Interna
  - Objetos do sistema colaboram uns com os outros para produzir os resultados visíveis de fora
    - Aspecto dinâmico
      - Descreve troca de mensagens e sua reação a eventos
      - Modelo de Interações
    - Aspecto estrutural estático
      - Permite compreender como o sistema está estruturado internamente
        - Estático porque não apresenta informações de interação no decorrer do tempo
        - Estrutural porque a estrutura das classes de objetos componentes do sistema e as relações entre elas são representadas
      - Representado pelo Modelo de Classes
        - Diagrama de Classes e descrição textual associado ao mesmo

Estágios do Modelo de Classes

- Modelos de classe de análise
  - Construído na fase de análise
  - Foca sobre o que o sistema deve fazer
  - Juntamente com os casos de uso são os dois principais modelos criados na fase de análise
  - Se fosse uma casa pensaríamos em salas, quartos, banheiros, portas, janelas, etc
- Modelos de classe de especificação
- Modelos de classe de implementação

Estágios do Modelo de Classes

- Modelos de classe de análise
- Modelos de classe de especificação
  - É um detalhamento do anterior
  - Conhecido como modelo de classe de projeto
  - Neste estágio descobre-se a necessidade de se criar outras classes
    - Por que começamos a focar em “como” o sistema deve funcionar
  - Se fosse uma casa teríamos que pensar em detalhes como encanamento, instalação elétrica, etc...
- Modelos de classe de implementação

Estágios do Modelo de Classes

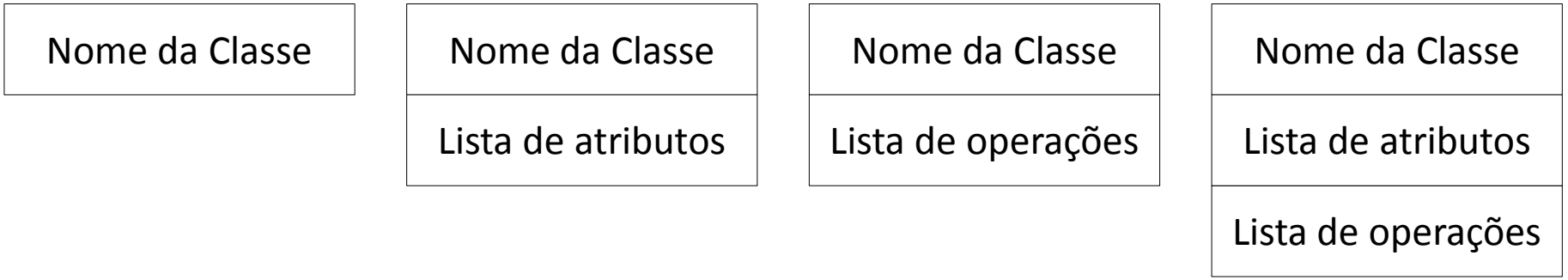
- Modelos de classe de análise
- Modelos de classe de especificação
- Modelos de classe de implementação
  - Detalhamento do anterior
  - Corresponde à implementação das classes em alguma linguagem de programação (C++, Java, etc)

Diagrama de Classes

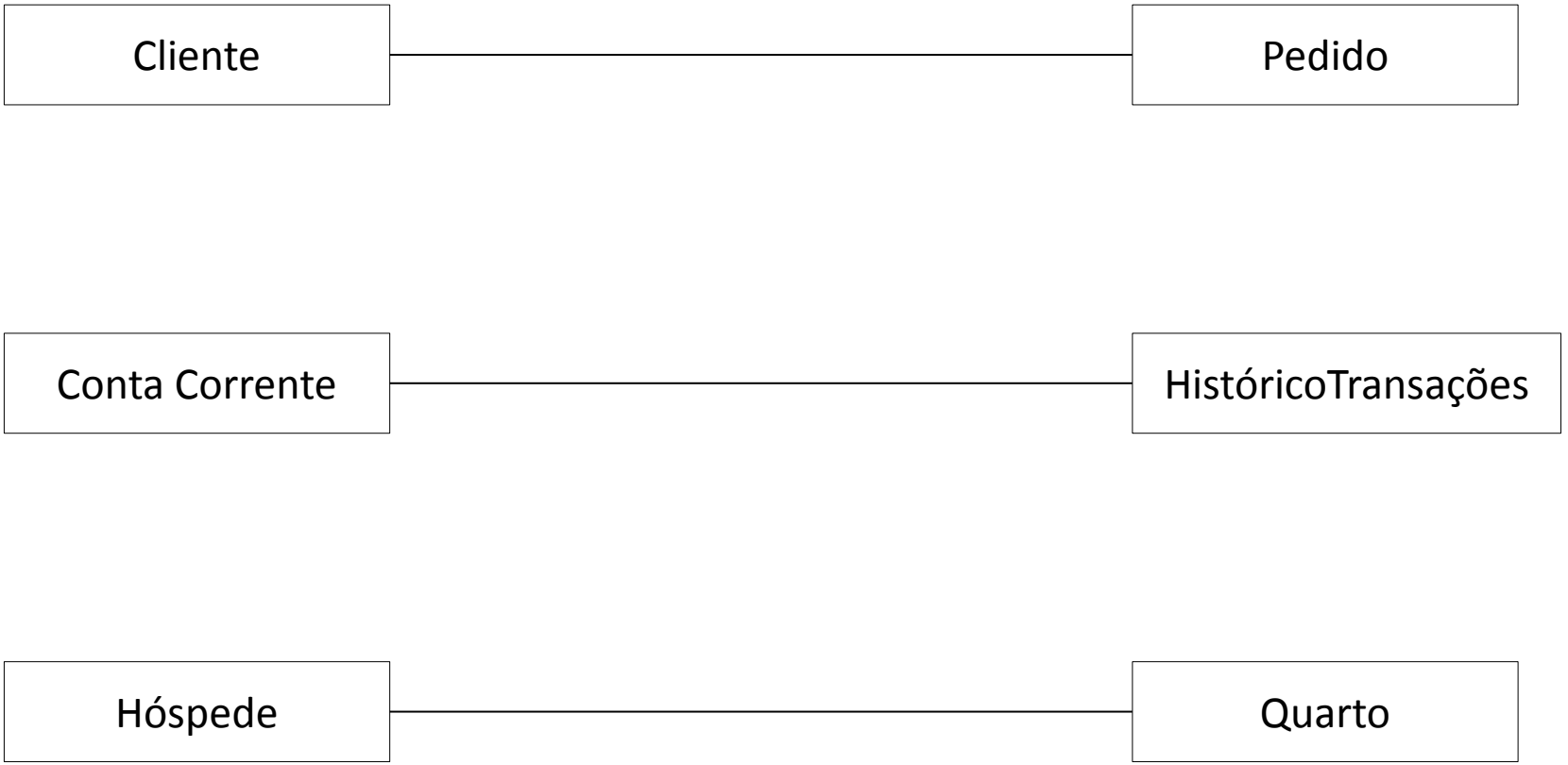
- Classes
- Associações
  - Multiplicidade
  - Participações
  - Nome de associações, direção de leitura e papéis
  - Classes associativas
  - Associações ternárias
  - Associações reflexivas
  - Agregações e composições
- Generalizações e especializações

Classes

- Notações



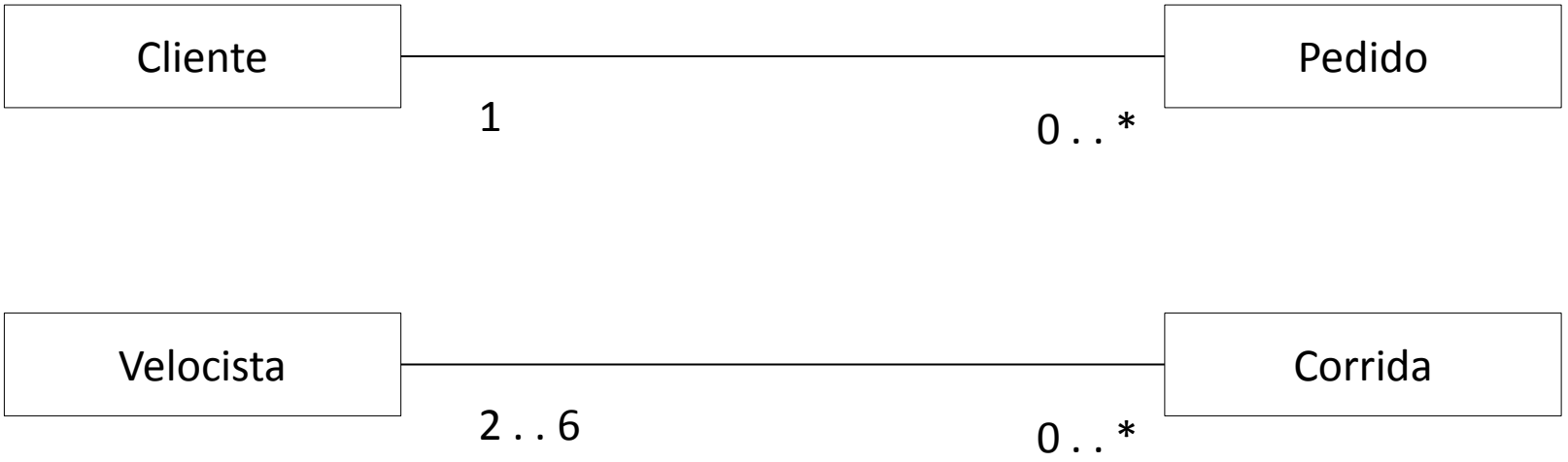
Associações



Multiplicidade

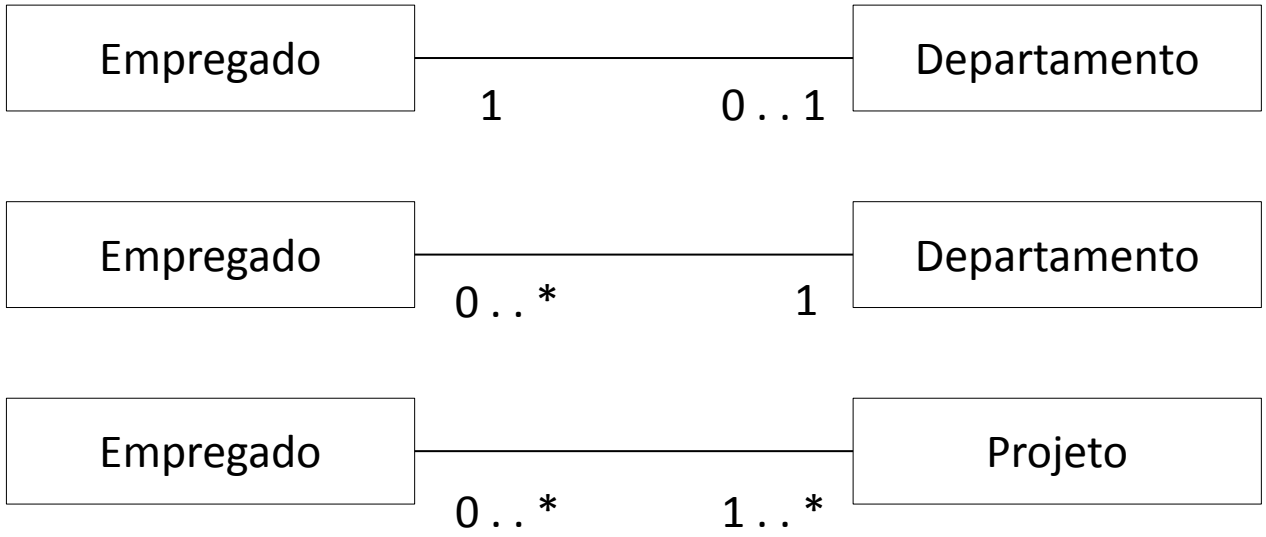
Nome	Simbologia
Apenas Um	1
Zero ou Muitos	0 . . *
Um ou Muitos	1 . . *
Zero ou Um	0 . . 1
Intervalo Específico	i . . j

Multiplicidade

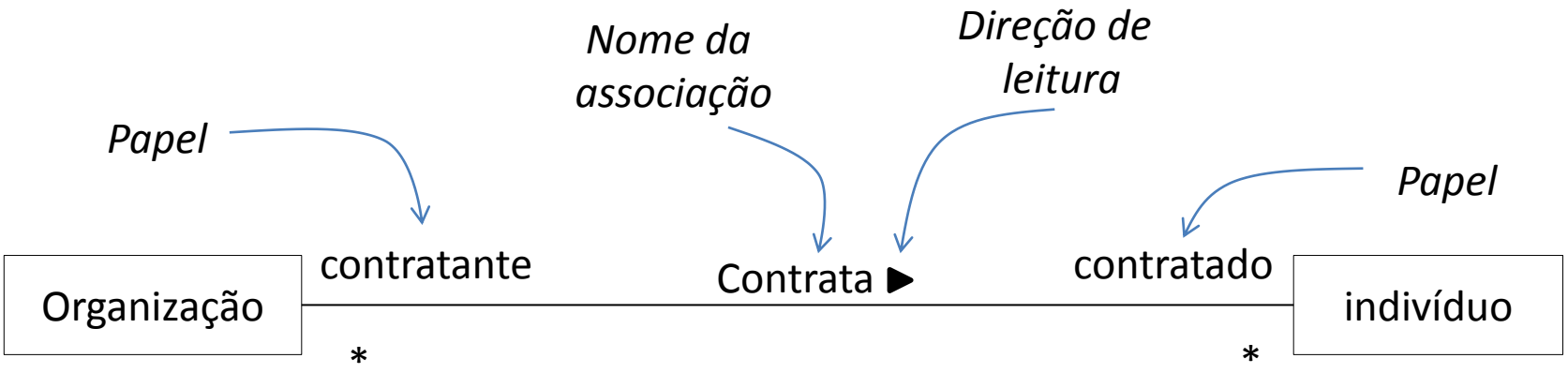


Conectividade x Multiplicidade

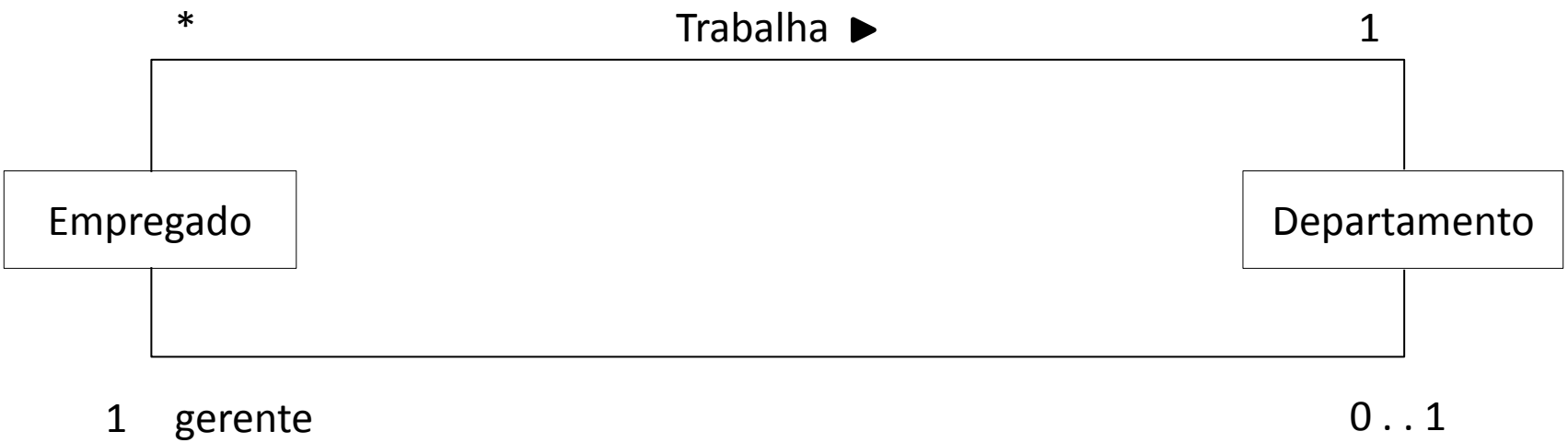
Conectividade	Multiplicidade de um extremo	Multiplicidade de outro extremo
Um para um	0 . . 1 ou 1	0 . . 1 ou 1
Um para muitos	0 . . 1 ou 1	* ou 1 . . * ou 0 . . *
Muitos para muitos	* ou 1 . . * ou 0 . . *	* ou 1 . . * ou 0 . . *



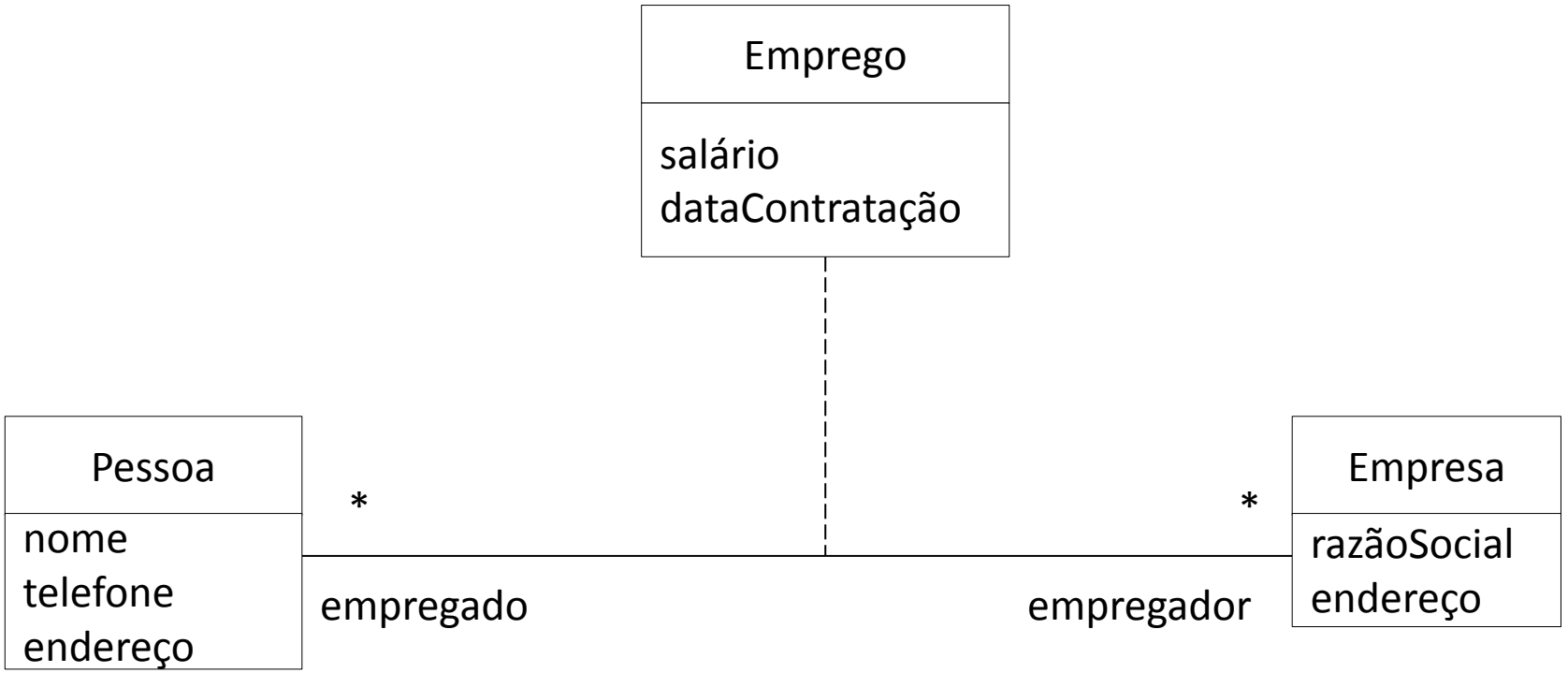
Nome de associação, direção de leitura e papéis



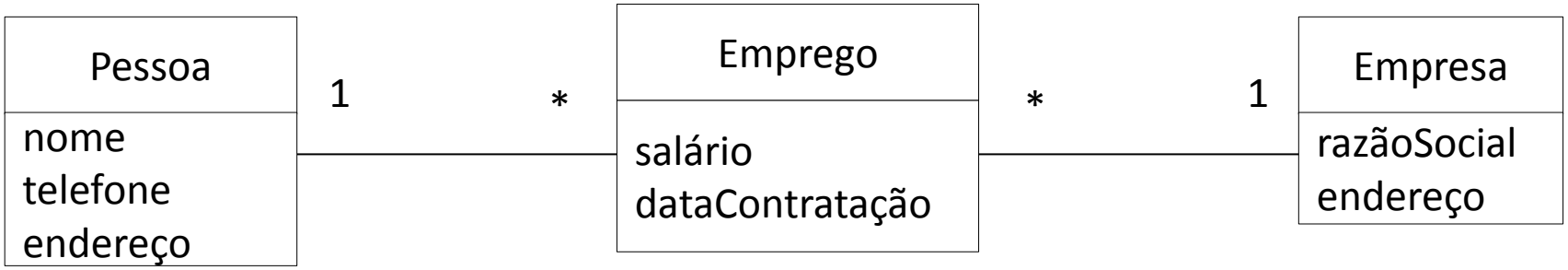
Nome de associação, direção de leitura e papéis



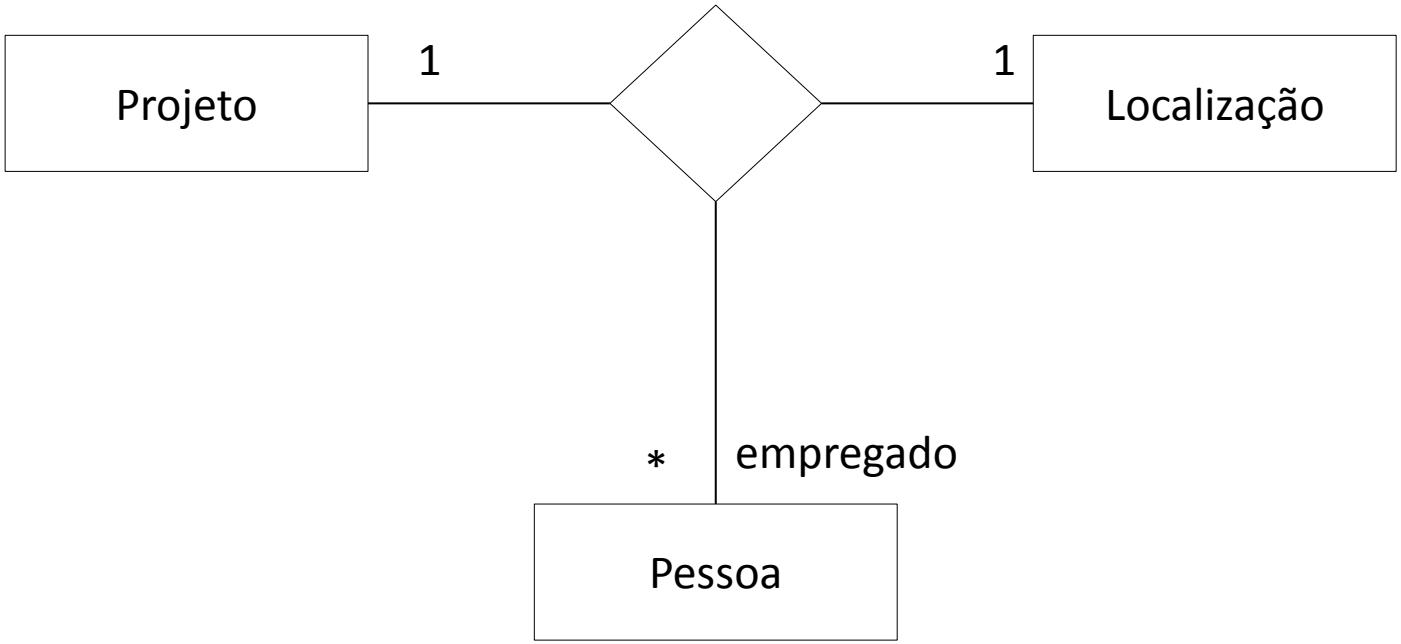
Classe Associativa



Substituição de Classe Associativa por Classe Ordinária

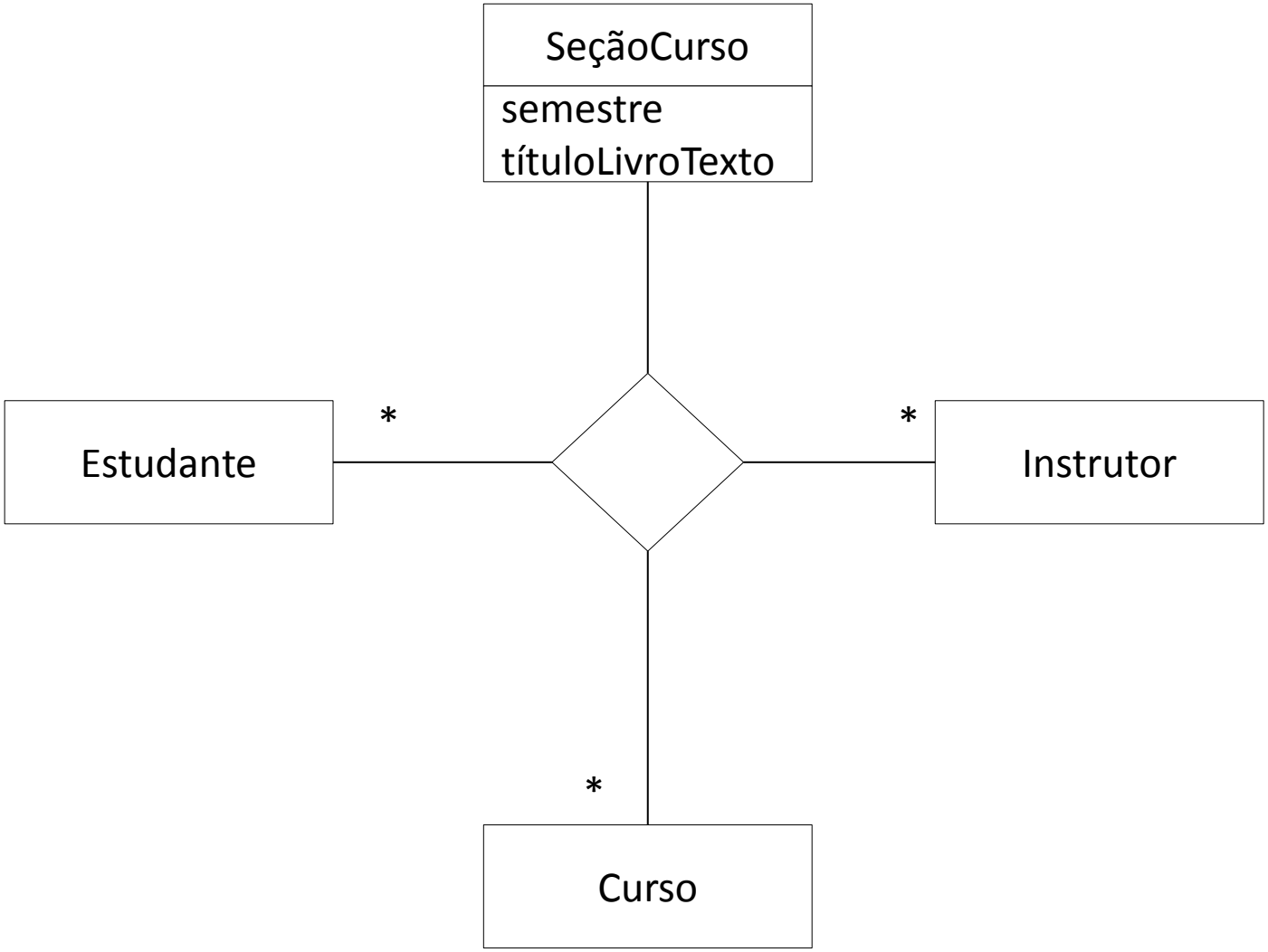


Associação Ternária

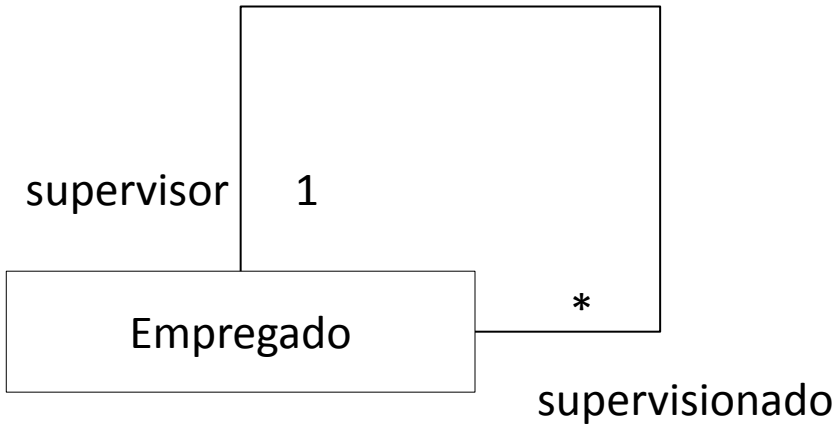


Cada pessoa empregada em um projeto trabalha em somente uma localização, mas pode estar em diferentes localizações em diferentes projetos.

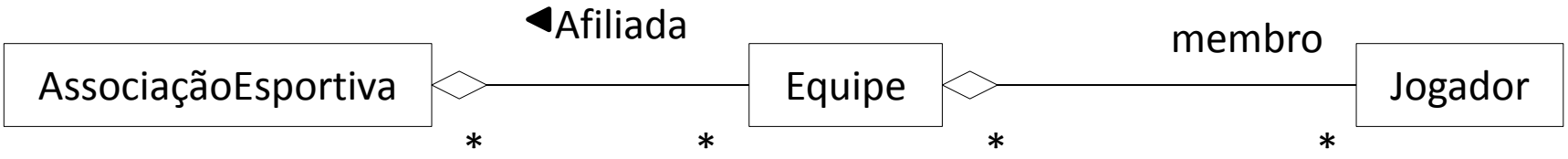
Classe Associativa Ternária



Associação Reflexiva

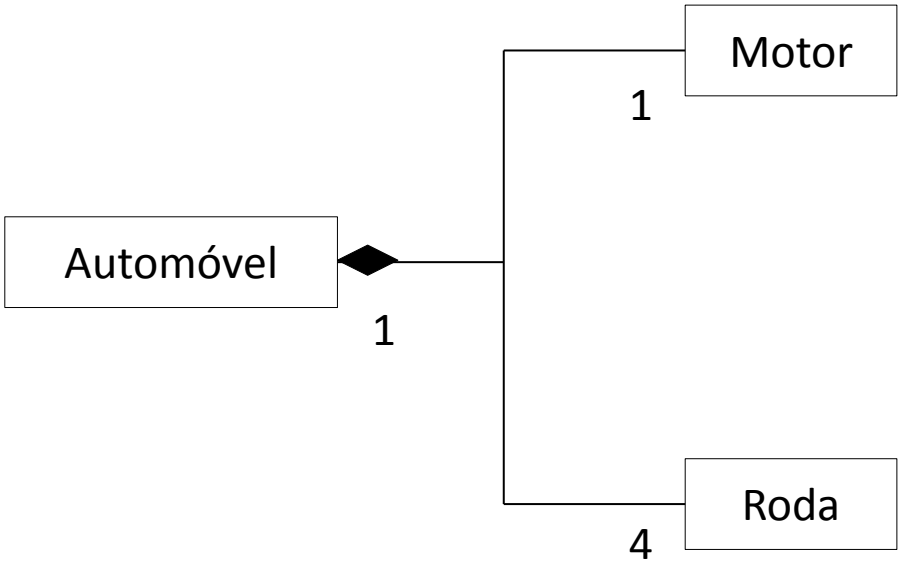


Agregação



Este diagrama indica que uma associação esportiva é formada por diversas equipes. Cada equipe é formada por diversos jogadores. Por outro lado, um jogador pode fazer parte de diversas equipes.

Composição

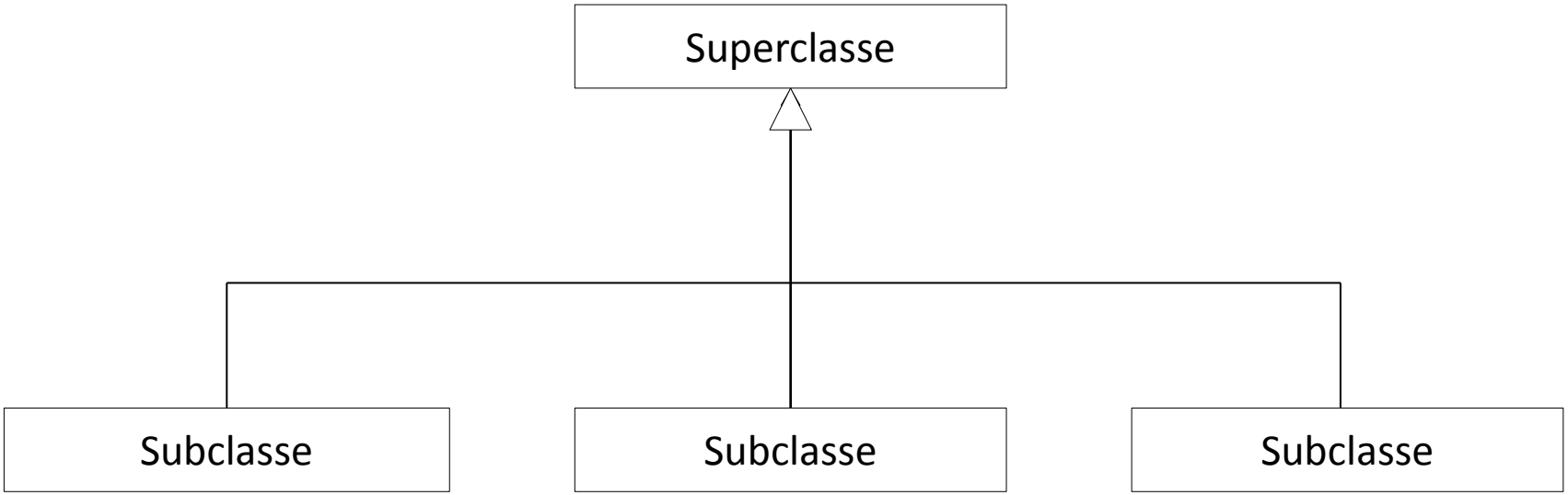


Todo:automóvel
Partes: Motor e Roda

Composição x Agregação

- Se um projetista constata que uma associação tem uma semântica todo-parte, ele deve decidir que relacionamento utilizar:
- Dica
  - Na agregação, a destruição de um objeto todo não implica necessariamente a destruição do objeto parte
  - Na composição, os objetos *parte* pertencem a um único *todo*

Generalização
(*herança*)



Generalização (*herança*)

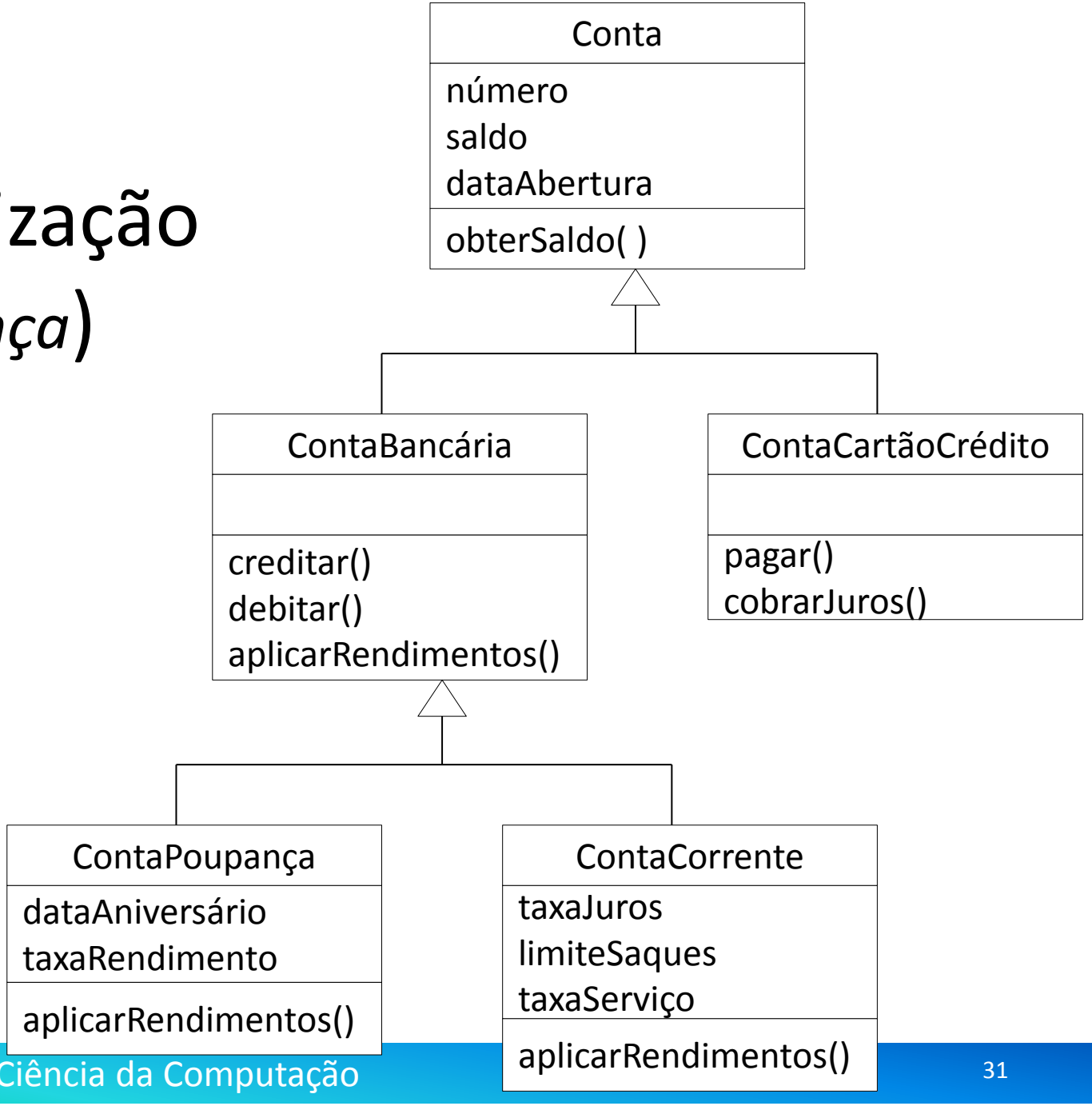
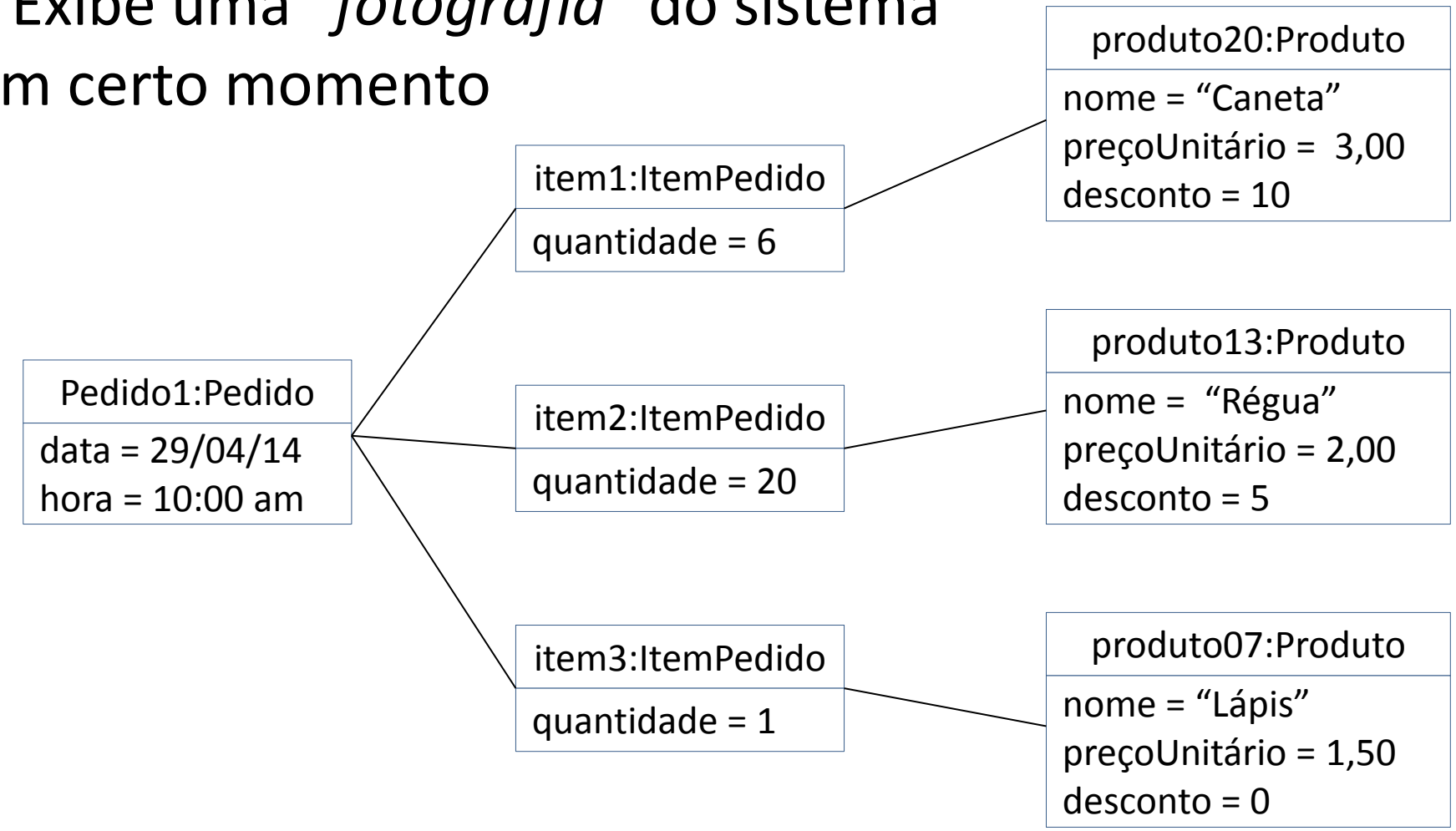


Diagrama de Objetos

- Exibe uma "*fotografia*" do sistema em certo momento



Técnicas para identificação de classes

- Análise textual de Abbot
- Análise de casos de uso
  - Categorização BCE
- Identificação dirigida a responsabilidades
  - Modelagem CRC
- Padrões de análise

Análise textual de Abbot

- Fonte: documentos de requisitos, modelos de negócio, glossários, conhecimentos sobre o domínio
  - Destacar os nomes (substantivos e adjetivos – locuções equivalentes a substantivos)
  - Remover os sinônimos
  - O que sobra se enquadra em:
    - O termo se torna uma classe (são candidatas a classe)
    - O termo se torna um atributo
    - O termo não tem relevância com relação aos requisitos do sistema

Análise textual de Abbot

- Identificação de operações e associações
  - Destacar os verbos no texto
    - Verbos com sentido de ação (calcular, confirmar, cancelar, fechar, estimar, depositar, sacar, etc)
      - são operações em potencial
    - Verbos com sentido de ter
      - São agregações ou composições em potencial
    - Verbos com sentido de ser
      - São generalizações (herança) em potencial
    - Demais verbos
      - São associações em potencial

Análise textual de Abbot

Parte do texto	Componente	Exemplo
Nome próprio	Objeto	José Silva
Nome simples	Classe	aluno
Verbos de ação	Operação	registrar
Verbo ser	Herança	é um
Verbo ter	Todo-parte	tem um

Análise textual de Abbot

- Vantagem
  - Simplicidade
- Desvantagem
  - Resultado depende das fontes analisadas estarem completas
  - Dependendo do estilo de texto utilizado para escrever o texto , pode levar a identificar diversas classes candidatas que não gerarão classes
  - Pior que isso, a análise do textopode não deixar explícita uma classe importante
- Solução
  - Aplicar outras técnicas

Análise de casos de uso

- Caso particular da Análise Textual Abbot
- Fonte: MCU
  - Caso de uso corresponde a um comportamento específico do sistema
  - Em um sistema esse comportamento pode ser produzido por objetos que compõem o sistema
  - Tendo isto em mente, o modelador tenta identificar as classes necessárias para produzir o comportamento que está documentado na descrição do caso de uso

Análise de casos de uso

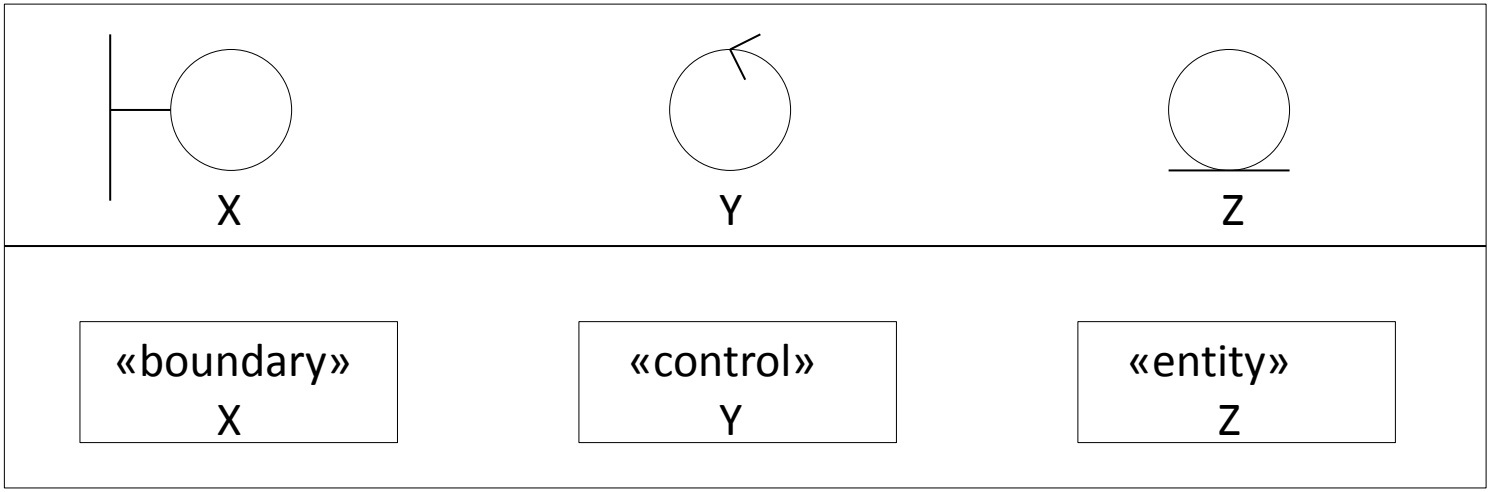
- Passo a passo
  1. O modelador estuda a descrição textual de cada caso de uso para identificar as classes candidatas
  2. Para cada caso de uso, seu texto (cenário, exceção, pós-condições e pré-condições, etc) é analisado
  3. Na análise de certo caso de uso, o modelador tenta identificar classes que possam fornecer o comportamento do mesmo
  4. Na medida em que os casos de uso são analisados um-a-um, as classes do sistema são identificadas
  5. Quando todos os casos de uso tiverem sido analisados, todas as classes (ou pelo menos a maioria delas) terão sido identificadas
- Justificativa
  - A existência de uma classe só pode se justificar se ela participar de alguma forma do comportamento externamente visível do sistema

Análise de casos de uso (resumindo)

1. Complemente as descrições de caso de uso.
    - Esse passo envolve complementar a descrição dos casos de uso com o objetivo de torná-los completos e facilitar a identificação de todas as classes envolvidas
2. Para cada caso de uso:
    a. Identifique classes a partir do comportamento de caso de uso
    b. Distribua o comportamento do caso de uso pelas classes identificadas
3. Para cada classe de análise resultantes:
    a. Descreva suas responsabilidades
    b. Descreva atributos e associações
4. Unifique as classes de análise identificadas em um ou mais diagramas de classes

Categorização BCE
(boundary, control, entity)

- Objetos podem ser divididos em 3 categorias
  - Objetos de fronteira
  - Objetos de controle
  - Objetos de entidade



Notação UML para objetos, segundo BCE

Objetos de Fronteira

- Realizam a comunicação do sistema com os atores
- 3 tipos principais
  - Realizam a interface com o usuário (atores humanos)
  - Realizam a interface com sistemas externos
  - Realizam comunicações com dispositivos atrelados ao sistema

Objetos de Fronteira

- Responsabilidades
  1. Notificar aos demais objetos os eventos gerados pelo ambiente do sistema
  2. Notificar aos atores o resultado de interações entre os demais objetos
- Não devem conter verbos nem sugerir ações
- Nome deve lembrar que é canal de comunicação
- Se comunicam com atores e controladores
- Existem somente durante a realização do caso de uso
- Modelador deve abstrair os detalhes (como vais ser feita a comunicação) dessa categoria de objetos
- Objetivo é identificar classes e suas responsabilidades em alto nível de abstração sem comprometer soluções técnicas

Objetos de Controle

- Ponte de comunicação entre objetos de fronteira e objetos de entidade
- Responsáveis por coordenar a execução de alguma funcionalidade específica do sistema
  - Normalmente, (mas não necessariamente), corresponde a um caso de uso
- Decidem o que o sistema deve fazer quando ocorre um evento externo relevante
- Controlam o processamento, i.e., são "*gerentes*" de outros objetos para a realização de um ou mais cenários de um caso de uso
- Em geral tem vida curta

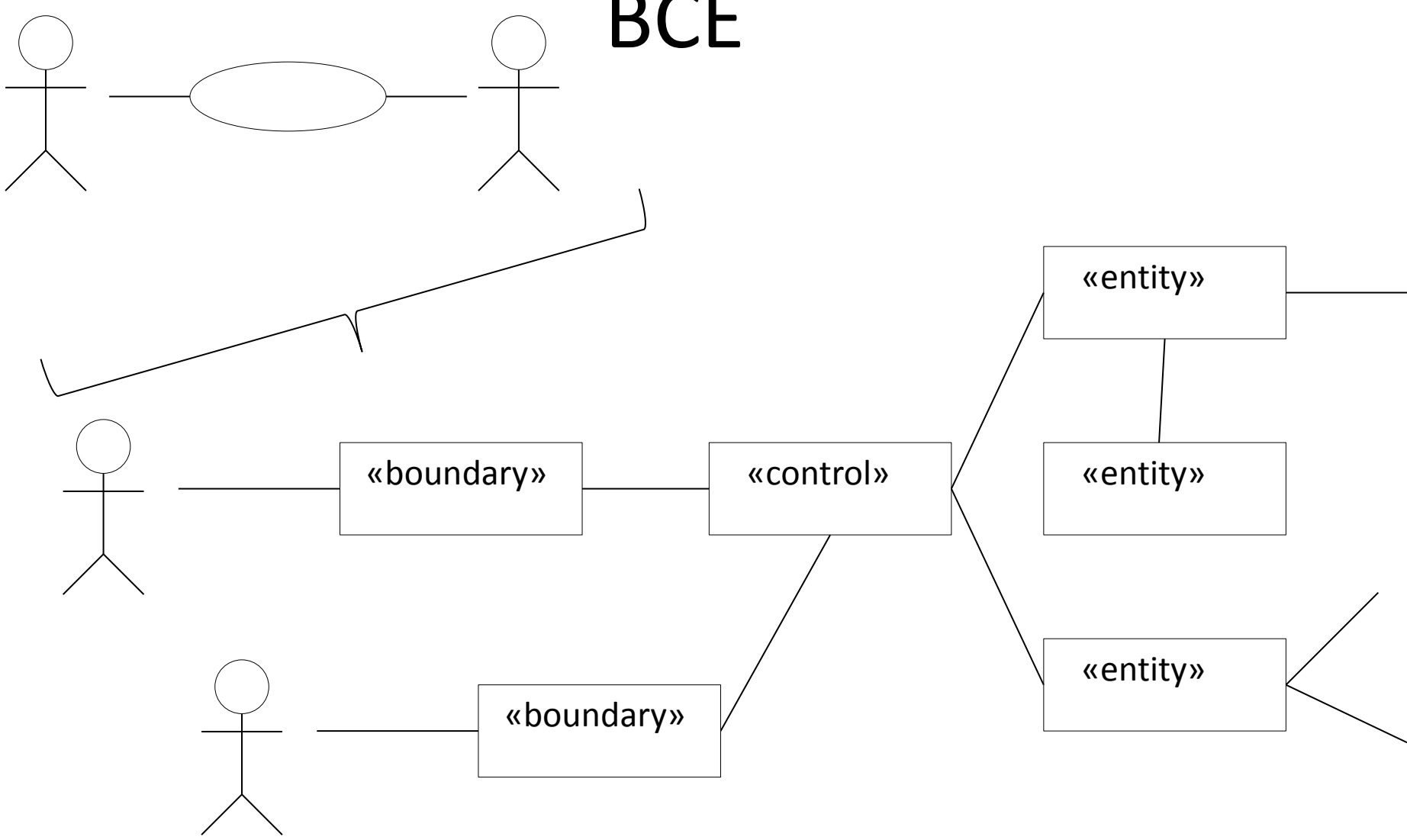
Objetos de Entidade

- Representa um conceito encontrado no domínio do problema
- Normalmente servem como repositório para alguma informação manipulada pelo sistema
- Possuem as responsabilidades mais importantes do sistema, que dizem respeito à lógica do negócio
- É comum existirem várias instâncias de uma mesma classe entidade coexistindo no sistema
  - Ex: Sistema de venda de produtos
    - Têm milhares de objetos da classe Produto
- Participam de vários casos de uso
- Têm um ciclo de vida longo (armazenam informações persistentes)

BCE – regras aplicáveis

1. Adicionar um objeto de fronteira para cada ator participante do caso de uso
   - Dessa forma as particularidades de comunicação com cada ator de caso de uso ficam encapsuladas no objeto de fronteira correspondente
2. Adicionar um objeto de controle para cada caso de uso
   - Isso porque um caso de uso representa um determinado processo de negócio
   - O controlador do caso de uso tem então a atribuição de coordenar a realização desse processo de negócio

BCE



Identificação dirigida a responsabilidades

- Ênfase na identificação de classes a partir de seus *comportamentos* relevantes para sistema
- Esforço do modelador recai sobre a identificação das responsabilidades que cada classe deve ter dentro do sistema
- Um objeto cumpre com suas responsabilidades a partir das informações que ele possui ou das informações que ele pode derivar de colaborações com outros objetos
- Modelagem CRC

CRC

(Classes, Responsabilidades e Colaboradores)

- Se baseia fortemente no paradigma de orientação a objetos, em que objetos colaboram uns com os outros para que uma tarefa seja realizada
- Idéia básica é de que as responsabilidades do sistema devem ser atribuídas aos objetos componentes do mesmo

$R_1, R_2, \ldots, R_i, \ldots, R_k, \ldots, R_n$

Responsabilidades identificadas para o sistema

$R_1, R_2, \ldots, R_i$

Objeto

$R_{i+1}, \ldots, R_k$

Objeto

$R_{k+1}, \ldots, R_n$

Objeto

CRC - início

- Reunir profissionais em uma sala para uma sessão CRC
  - Cerca de seis pessoas
    - Especialistas de domínio
    - Projetistas
    - Analistas
    - Moderador da sessão
- Cada pessoa recebe um cartão CRC que corresponde a uma classe do sistema
- Seleciona-se um conjunto de cenários de determinado caso de uso é selecionado
- Para cada cenário do conjunto, faz-se uma sessão CRC

Cartão CRC

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

CRC - execução

- Algum dos participantes simula o ator primário que dispara a realização do caso de uso
- À medida que esse participante simula a interação do ator com o sistema, os demais participantes encenam a colaboração entre objetos que ocorre internamente ao sistema para que o objetivo do ator seja alcançado
- Graças a essa encenação desempenhada pelos participantes, as classes, responsabilidades e colaboradores são identificados

CRC
Dicas para atribuir responsabilidades

- Associe responsabilidades com base na especialidade da classe
  - Fronteira, controle, entidade
- Distribua a inteligência do sistema
  - Uma classe não deve ser sobrecarregada com responsabilidades demais
  - Se isso acontecer, considere a criação de mais classes
  - Uma quantidade pequena de classes complexas demais significa que as mesmas encapsulam mais conhecimento do que o desejado e são menos reutilizáveis
  - Uma quantidade maior de classes mais simples significa que cada classe encapsula menos inteligência e, assim, são mais reutilizáveis

CRC
Dicas para atribuir responsabilidades

- Agrupe as responsabilidades conceitualmente relacionadas
  - responsabilidades conceitualmente relacionadas devem ser mantidas em uma única classe (mantenha a coesão da classe)
- Evite responsabilidades redundantes
  - Se duas ou mais classes precisam de alguma informação analise as seguintes alternativas para manter a informação :
    - Criar uma nova classe
    - Escolher uma das classes preexistentes
  - Em ambos os casos, a informação fica em um único lugar, e os outros objetos devem enviar mensagens para o objeto que possui tal informação para obtê-la.

CRC

- Técnica útil para
  - identificar novas classes
  - Validar classes identificadas mediante outras técnicas

Padrões de Análise

- Análogo aos Padrões de Projeto (Design Patterns)
  - Identifica caracterísitcas comuns aos diversos modelos
  - Ao reconhecer processos e estruturas comuns em um domínio de problema, o modelador pode descrever (catalogar) a parte essencial dos mesmos, dando origem a um padrão de software
- Área em desenvolvimento, com muita pesquisa e catalogação
- Fora do escopo da disciplina

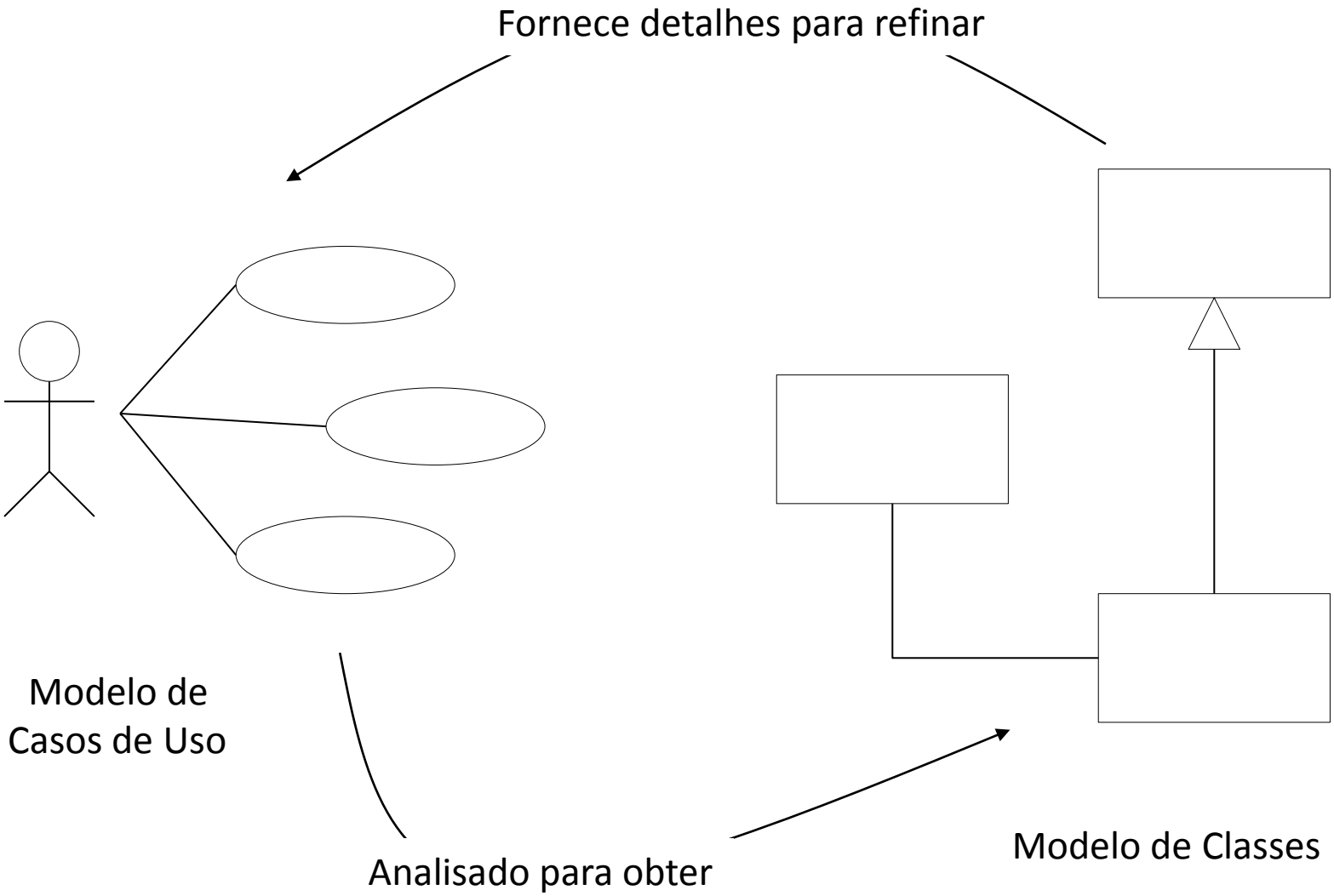
Modelos de Dados x Modelos de Classes

- Modelos de classes são perigosamente similares a modelos de dados. A maioria dos princípios que resultam em um bom modelo de dados também resultam em um bom modelos de classes. O perigo é justamente desenvolver um modelo de classes orientados a dados, em vez de orientado a responsabilidades
- Outra diferença significativa é que o modelo de objetos representa o comportamento dos objetos, além dos seus dados. Em domínios em que têm um viés mais comportamental dificilmente se identificam com a modelagem de dados

Modelos de Classes

- Após a identificação das classes e responsabilidades, o modelador deve verificar a consistência entre as classes para eliminar as incoerências e redundâncias
- Em seguida o analista deve começar a definir o mapeamento das responsabilidades e dos colaboradores de cada classe para os elementos do Diagrama de Classes

Modelos de Classes



Construção de Modelos de Classe

- Definição de propriedades
- Definição de associações
- Organização da documentação

Definição de Propriedades

- Propriedade
  - Nome genérico aplicado aos atributos e operações da classe
- Questão chave
  - Como mapear responsabilidades atribuídas a uma classe para propriedades da mesma
- Distinguindo as responsabilidades:
  - de fazer
  - de conhecer

Definição de Propriedades

- Responsabilidades *de fazer*
  - Correspondem às operações
  - Algumas operações são facilmente identificáveis, mas uma definição completa e detalhada só pode ser feita após a construção dos diagramas de interação
    - Representam detalhes sobre a ordem das colaborações entre os objetos que participam de um caso de uso
    - Será visto mais adiante, i.e., agora veremos apenas o mapeamento de responsabilidades em atributos

Definição de Propriedades

- Normalmente uma responsabilidade *de conhecer* é mapeada para atributos ou para associações
- Tarefa relativamente fácil, desde que estejam corretamente identificadas as responsabilidades *de fazer* e os colaboradores da classe

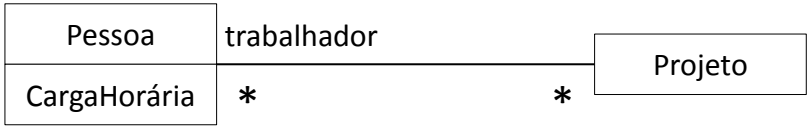
Definição de Propriedades

- Após mapear responsabilidades em atributos , o modelador deve verificar se cada atributo possui as seguintes propriedades
  1. Guarda um valor atômico
     - R$ 1,00, “João Silva”, 3
  2. Não contém estrutura interna
     - Propriedade derivada da anterior
  3. Aplica-se a todos os objetos da classe
     - Cada atributo de uma classe deve fazer sentido para todo e qualquer objeto da classe
     - Ex: atributo salário não faz sentido em uma classe Pessoa, pois nem toda pessoa tem salário

Definição de Associações

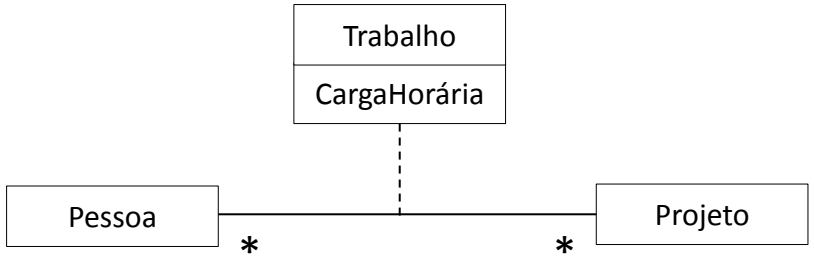
- Um objeto precisa conhecer o outro para fazer requisições
- Classes de associações surgem a partir de responsabilidades de saber que o modelador não conseguiu atribuir a alguma classe
  - Solução: criar uma classe associativa

Inadequado



Viola propriedades 2 e 3 do atributo

Adequado



Definiu-se classe associativa para cumprir responsabilidades órfãs

Documentação

- Após a identificação das classes (e suas responsabilidades) e colaboradores, esses elementos devem ser organizados em diagramas de classes. O conjunto de diagramas de classe forma o modelo de classes de análise
- Na diagramação, classes devem ser posicionadas de tal forma que associações sejam lidas da esquerda para a direita ou de baixo para cima
- Para cada caso de uso criar uma VCP (Visão de Classes Participantes). Uma VCP é um diagrama de classes cujos objetos participam de um caso de uso
- Classes também devem ser documentadas textualmente

Referências

- Ferramentas de modelagem visual
  - Rational Rose (www.rational.com)
  - ASTAH Community (astah.net/editions/community)
- Livros
  - The Unified Modeling Language User Guide, Grady Booch et al
  - Engenharia de software – uma abordagem profissional, Roger S. Pressman
  - Princípios de análise e projetos de sistemas com UML, Eduardo Bezerra
- Especificações
  - www.omg.org

Dúvidas

?