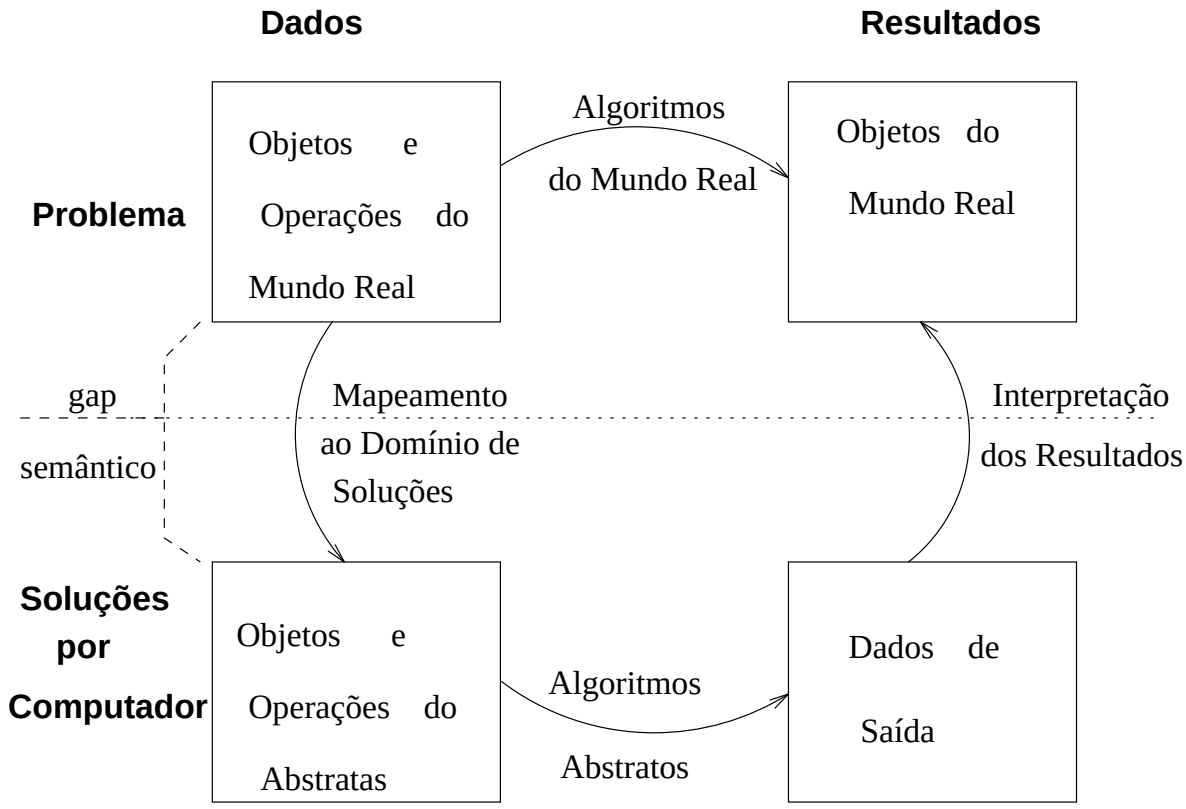
Análise e Projeto Orientados a Objetos: Introdução a UML

Silvia Regina Vergilio

Análise Orientada a Objetos



Quanto mais próximo estiver o espaço de soluções do espaço de problemas mais fácil será o desenvolvimento.

> compreensão , > confiabilidade, > facilidade de manutenção

UML: Unified Modeling Language (97)

$*$ método de Booch e OMT (Object Modeling Technique) de Rumbaugh

$*$ pretende descrever modelos do sistema baseados em objetos, que envolvem:

1. Escopo Geral; Identificação dos clientes; Objetivos

2. Funções do Sistema: o que o sistema tem que fazer

3. Atributos do Sistema: qualidades não funcionais tais como: facilidades de uso, plataformas de SO, tempo de resposta, tolerância a falhas, etc.

Categorias de Funções do Sistema

- evidente: precisa ser realizada e o usuário deve saber que ela é executada. Ex: calcular total de uma compra

- oculta: precisa ser realizada e o usuário não deve saber que ela é executada. Ex: armazenar dados em um arquivo.

- opcional: adicioná-la não afeta custos ou outras funções. Ex: exibir subtotais enquanto calcula total

Categorias de Atributos do Sistema

$***$ necessário, desejado

- tempo de resposta: 5 segundos
- interface: é necessário usar menus
- tolerância a falhas: pagamento precisa ser realizado em 24 horas
- plataformas de SO: WindowsNT

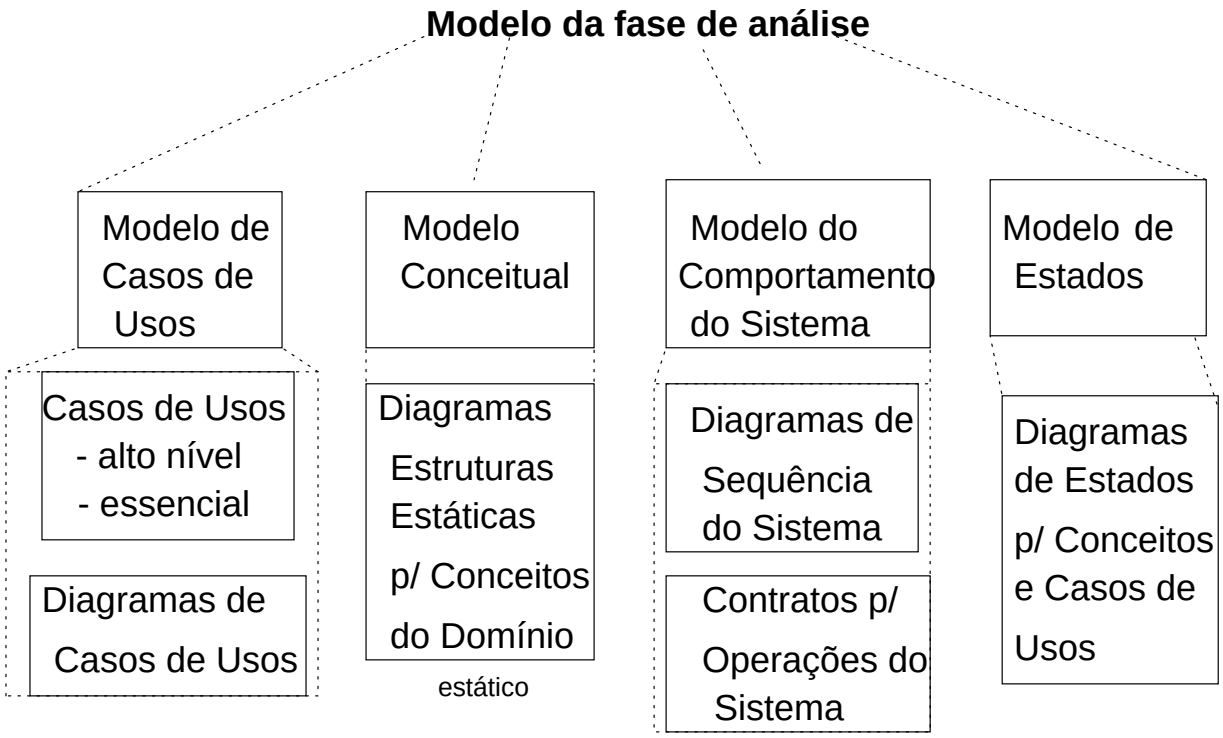
Exemplo: Funções e Atributos do Sistema

Ref#	Função	Categoria	Atributo	Detalhes e Restrições	Categoria
R1.9	Mostra descrição e preço de item registrado	evidente	tempo de resposta	5 segundos	necessário
R 2.4					

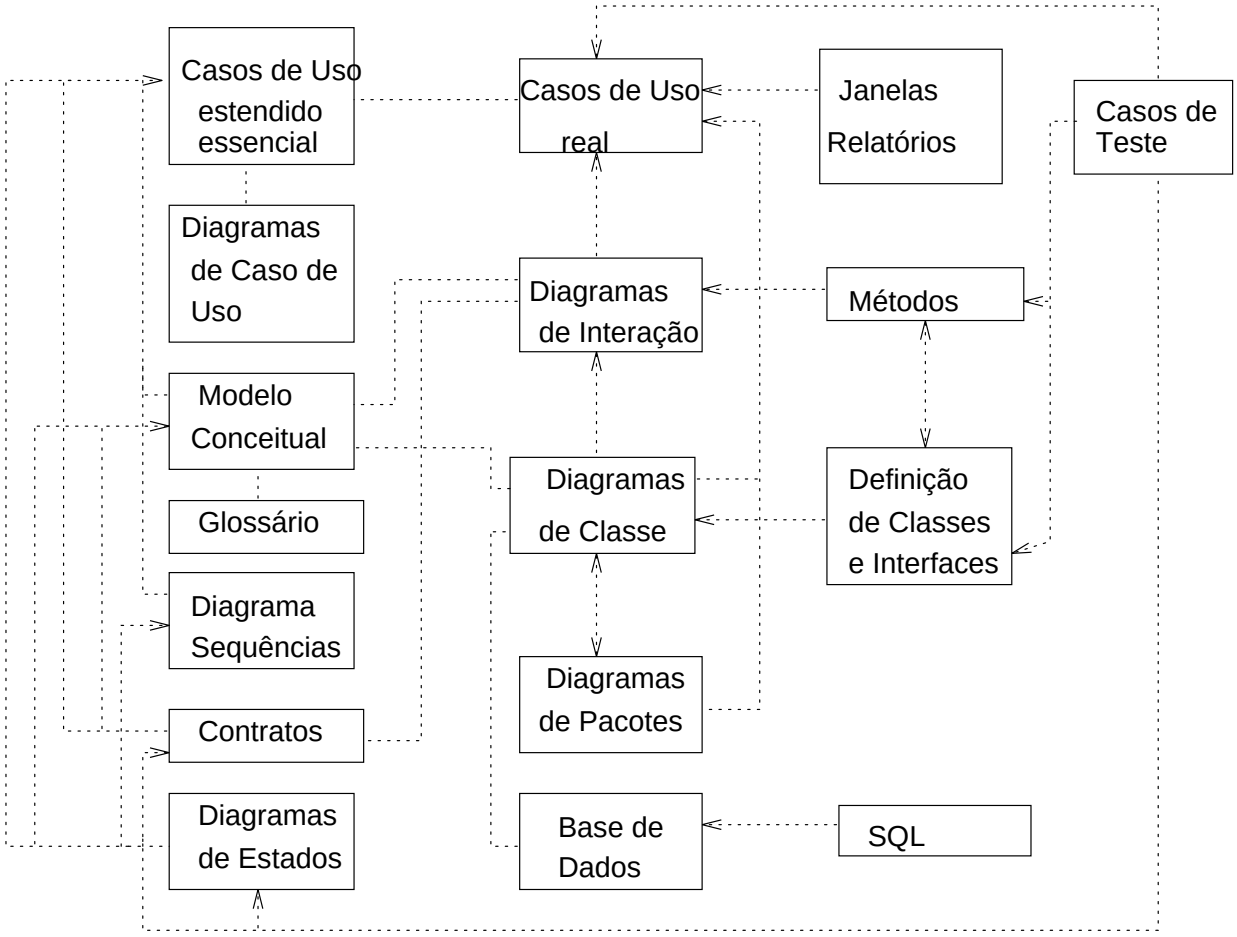
Modelos do Sistema

Organizam e comunicam detalhes importantes do problema do mundo real.

São compostos de artefatos: modelos, diagramas e documentos



Dependências entre Artefatos



Dependências entre Artefatos

I. Casos de Usos e Diagramas de Casos de Usos

Caso de Uso: é uma narrativa que descreve a sequência de enventos de um ator (agente externo) usando o sistema para completar um processo. São estórias de usos de um sistema.

Caso de Uso :	Comprar Itens
Atores:	Cliente, Caixa
Tipo:	principal
Descrição:	Um Cliente chega com itens para compra. O Caixa registra os itens de compra e recebe o pagamento. Ao final o Cliente vai embora com os itens

Casos de Usos podem ser:

- Alto Nível: descrição breve
- Estendido: descrição detalhada passo a passo dos eventos

- Principal: comum e bastante importante. Ex. comprar itens
- Secundário: ocorre mais raramente. Ex: cadastrar um produto
- Opcional: pode ou não acontecer. Ex: pagamento com cartão

- Essencial: livre de detalhes de tecnologia e implementação, decisões de projeto não estão presentes.
- Real: descrições em termos do projeto.

Caso de Uso Estendido

Caso de Uso :	Comprar Itens à Vista
Atores:	Cliente, Caixa
Propósito:	Registrar uma Venda
Descrição:	Um Cliente chega com itens para compra. O Caixa registra os itens de compra e recebe o pagamento. Ao final o Cliente vai embora com os itens
Tipo:	principal e essencial
Referências:	Funções: R1.1, R1.2, R2.1

Sequência de Eventos Ação do Ator	Resposta do Sistema
1. Este caso de uso comeca quando um cliente chega ao caixa com itens de compra	
2. O caixa registra cada item	3. Determina o preco do item e adiciona o item a venda.
Se tiver mais de um mesmo item o Caixa entra a quantidade.	Descrição e preço do item são apresentados
....	.....

Notação: Caso de Uso Estendido

Caso de Uso :	nome do caso de uso
Atores:	lista de atores que iniciam o caso de uso agentes externos
Propósito:	objetivo geral
Descrição:	resumo
Tipo:	1. principal, secundário ou opcional 2. real ou essencial
Referências:	funções do sistema relacionadas

Sequência de Eventos

Ações de Atores	Respostas do Sistema

Sequências Não Típicas

Identificando um Caso de Uso

Primeiro Método: baseado em atores

- identificar atores
- para cada ator, identificar os processos que ele inicia ou participa

Segundo Método: baseado em eventos

- identificar eventos externos aos quais o sistema deve responder
- relacionar os eventos a atores e casos de usos

Um processo descreve do início ao fim, uma sequência de eventos, ações ou transações necessárias para produzir ou completar alguma coisa para a organização ou ator.

Um caso de uso é uma descrição de um processo, e inclui muitos passos. Não utilizá-lo para descrever um passo ou atividade de um processo. Ex. Imprimir recibo é um passo de comprar itens.

Diagrama de Casos de Usos

Ilustra um conjunto de casos de usos para um sistema, atores e o relacionamento entre atores e casos de usos

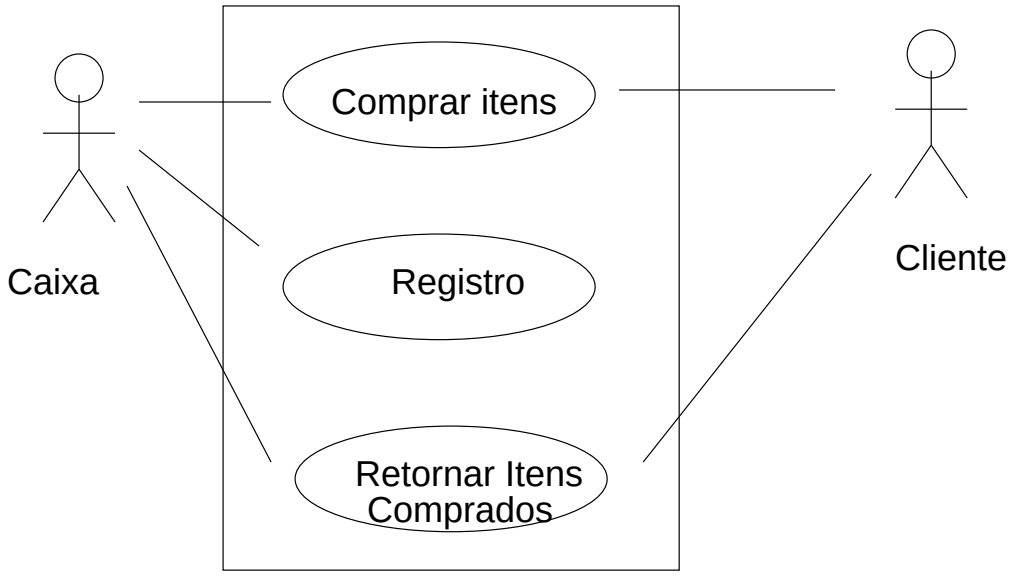
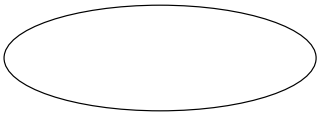
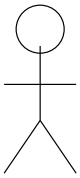


Diagram de caso de Uso



caso de uso



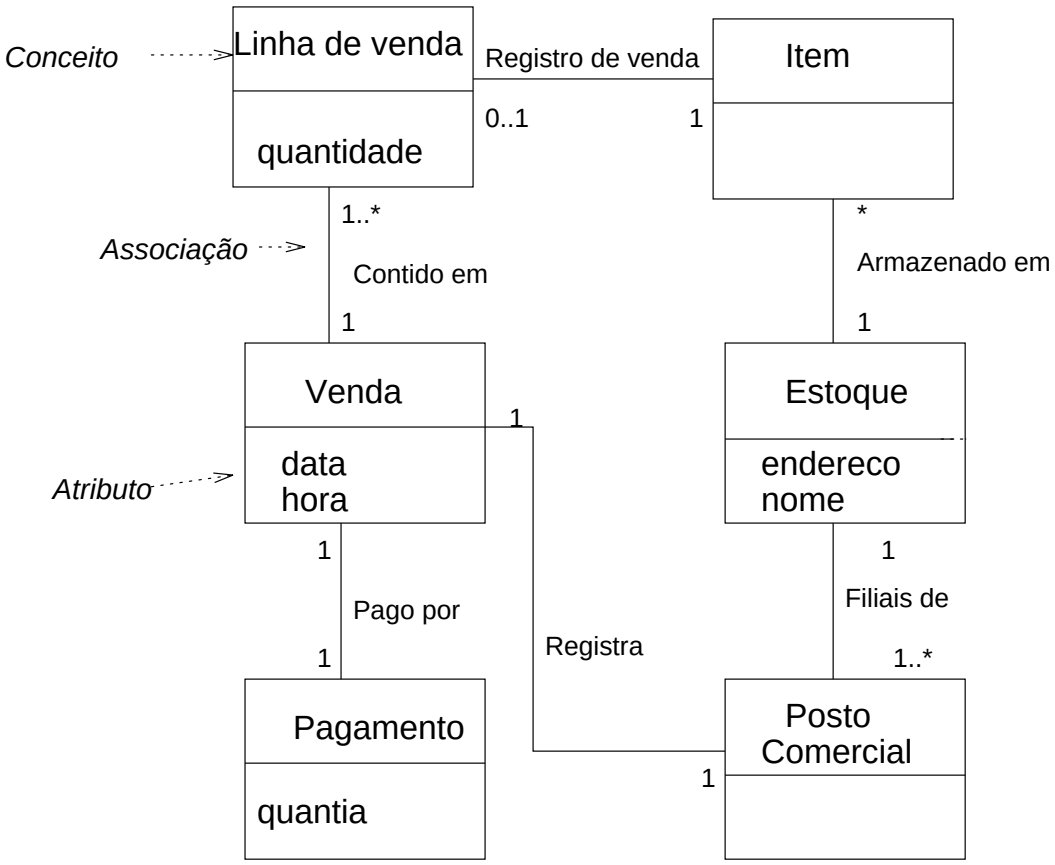
ator

fluxo de informação

II. Modelo Conceitual

Representação de conceitos no domínio do problema

Deve mostrar: conceitos, associações entre conceitos e atributos de conceitos.



Modelo Conceitual - Exemplo

1. Conceitos

É uma idéia, coisa ou objeto do mundo real.

Um bom modelo conceitual deve superestimar o número de conceitos.

Para identificar conceitos:

a) Buscar por nomes em descrições textuais:

A **descrição** e o **preço** de cada **item** são apresentados.

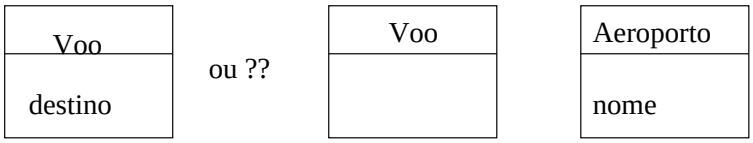
Cuidado: Imprecisão da linguagem natural

b) Relatórios, recibos, etc não devem aparecer no modelo conceitual a menos que sejam derivados de outras fontes.

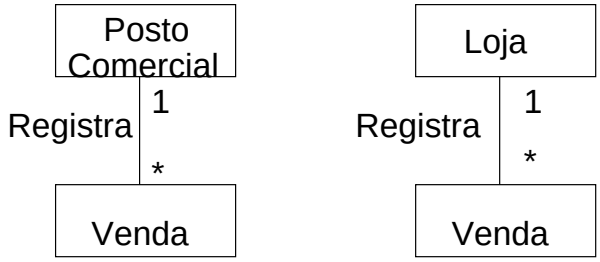
c) Lista de Categorias de Conceitos

- objetos físicos: avião
- especificações, descrição de coisas: especificação do produto, descrição do produto
- lugares: loja, aeroporto
- transações: reserva, pagamento
- função de pessoas: caixa, piloto
- recipiente de coisas: estoque, avião
- coisas em recipientes: itens, passageiros
- outros sistemas: autorização de cartao de crédito, controle de tráfico aéreo
- conceitos abstratos: fome
- organizações: cia aéreas, departamento de vendas
- eventos: voo, acidentes, venda, roubo
- processos (podem ser): vendendo um produto, reservando um assento
- regras e políticas: retorno de mercadoria, cancelamento
- catálogo: produtos de catálogo, partes de catálogo
- contratos legais, de finanças: contrato de trabalho
- serviços: linha de crédito
- manuais, livros: manual do empregado, manual de reparo

d) Se você não descreve X com um número ou texto no mundo real, X provavelmente será um conceito

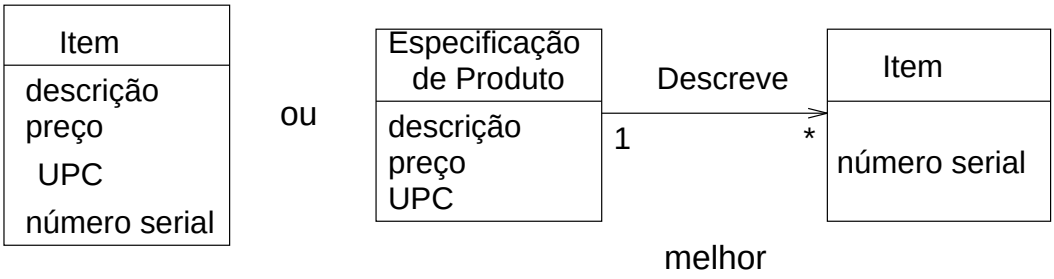


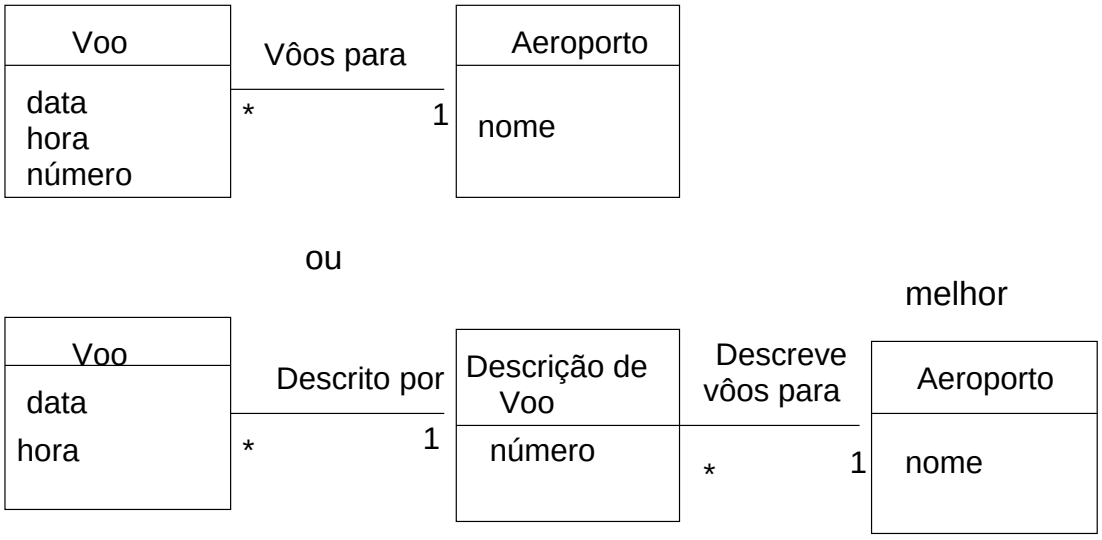
e) Um conceito pode não estar incorreto, pode apenas não ser útil



Conceitos Similares com Diferentes Nomes

f) Conceitos do tipo especificação/descrição de coisas: adicioná-los para reduzir redundância e informação duplicada ou quando informação é perdida se a coisa descrita for removida





Exemplo- Conceitos do Tipo Especificação/ Descrição

2. Associações

São relacionamentos entre conceitos que indicam alguma poderosa e interessante conexão.

Multiplicidade: define quantas instâncias de um tipo A podem ser associadas a uma instância de um tipo B

Critério para escolha das associações:

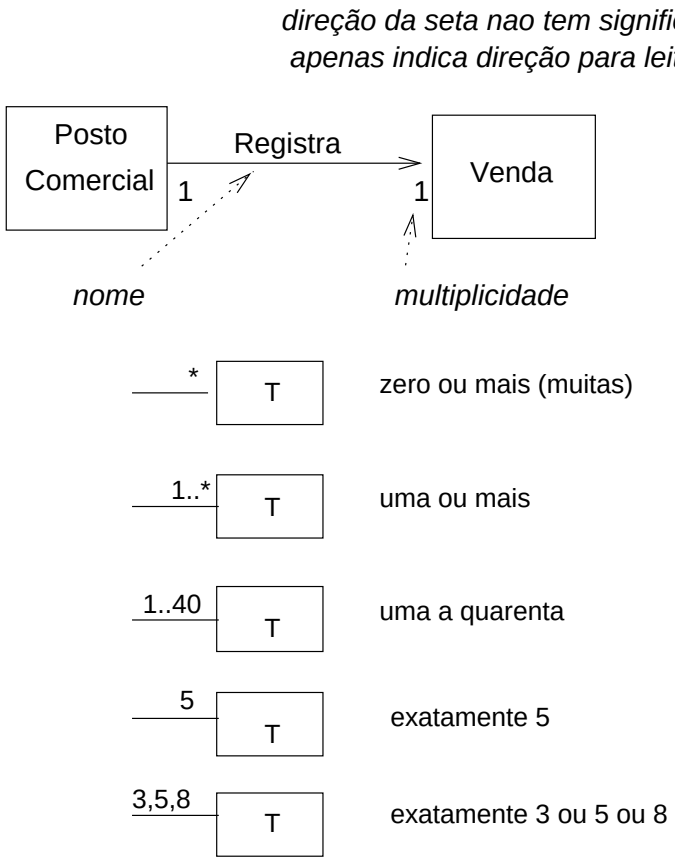
a) o conhecimento de um relacionamento precisa ser preservado (regra necessidade de saber)

b) são importantes para compreender conceitos importantes do domínio

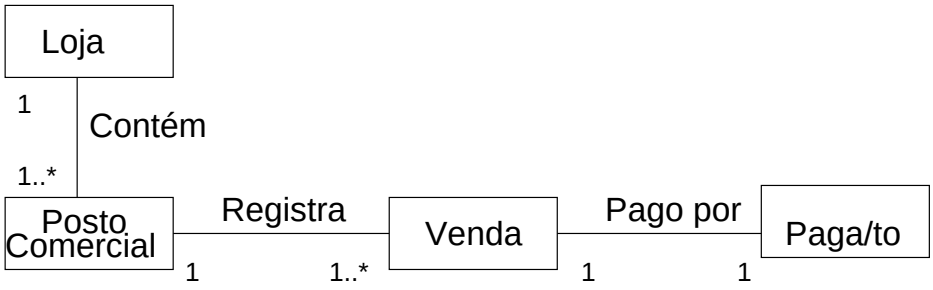
c) é melhor identificar conceitos que associações. Muitas delas podem tornar o modelo conceitual confuso. Evite associações redundantes e deriváveis.

d) através de uma lista de associações

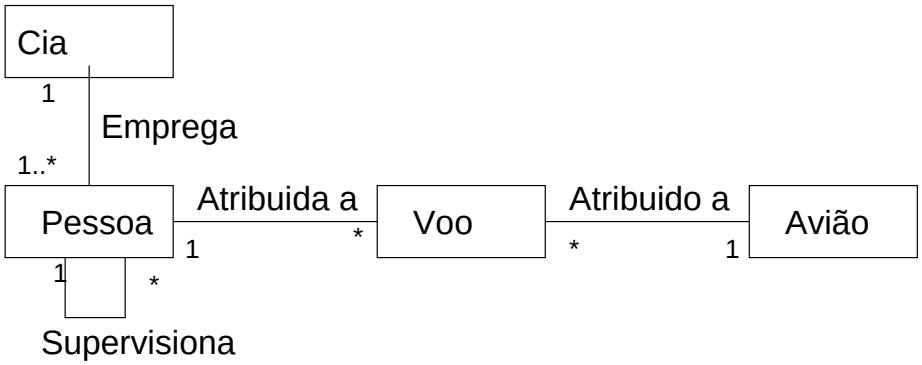
- A é parte física de B: asa - avião
- A é parte lógica de B: itens de venda - venda
- A está fisicamente contida em B: passageiro - avião
- A está logicamente contida em B: descrição de item - cátalogo
- A é descrição de B: descrição de voo - voo
- A é uma linha de transação ou relatório de B: trabalho de manutenção - relatório de manutenção
- A é conhecido, registrado, relatado em B: venda - posto comercial
- A é um membro de B: piloto - cia aérea
- A é uma sub-unidade operacional de B : departamento - loja
- A usa ou gerencia B: piloto - avião
- A se comunica com B: cliente - caixa
- A está relacionado a uma transação de B: passageiro - ticket
- A é uma transação relacionada a uma outra transação B: pagamento - venda
- A está proximo a B: cidade - cidade
- A é posse de B: avião - cia aérea



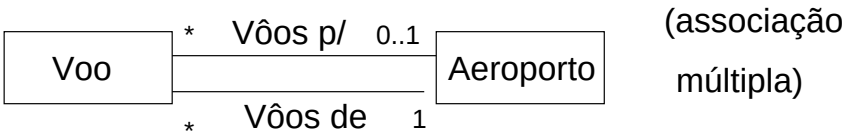
Notação para associações



(1)



(2)



(3)

Exemplos de Associações

3. Atributos

Um atributo é um valor lógico para um objeto.

Critério para escolha de atributos

a) aqueles que sugerem ou implicam necessidade de lembrança.

b) devem ser simples ou valores puros (primitivos)

c) não represente um atributo como chave estrageira

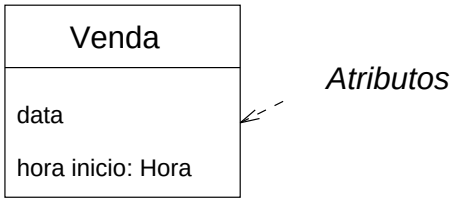
d) representar atributos primitivos como não primitivos se:

* são compostos: nome de pessoa, número de telefone

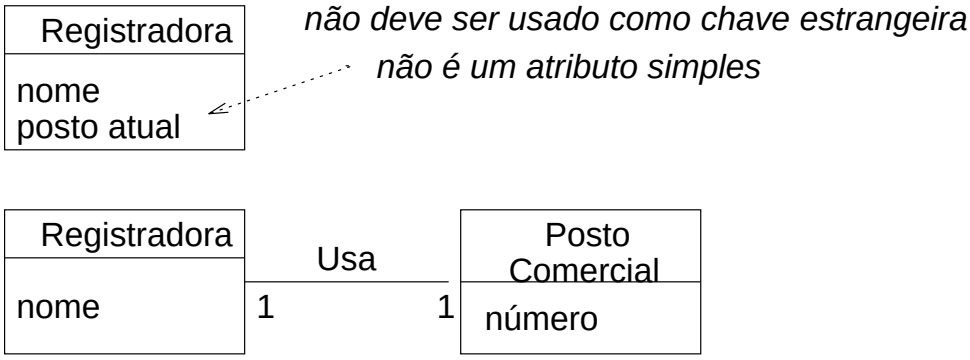
* sobre eles ocorrerem operações de validação: CPF

* eles têm outros atributos: preços promocionais com data de início e término

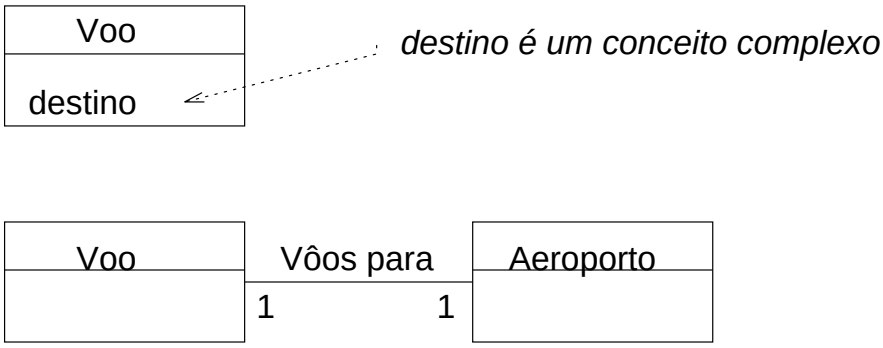
* representam quantidade com uma unidade: pagamento e moeda corrente



Notação para Atributos

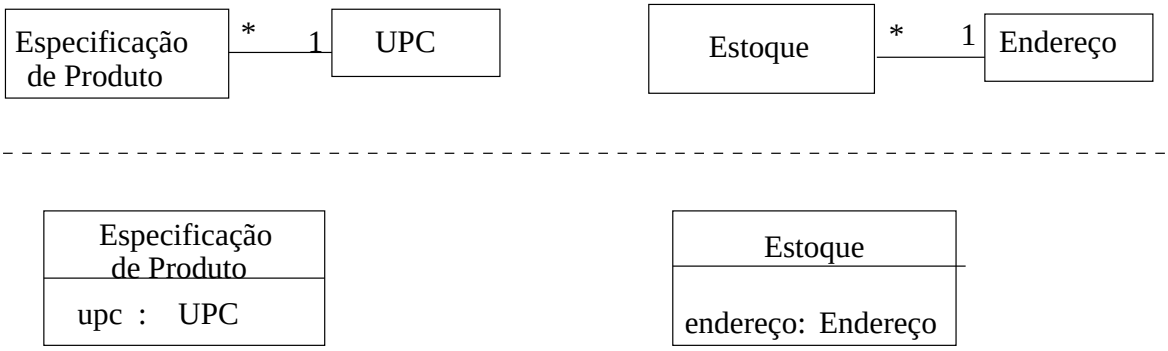


(1)

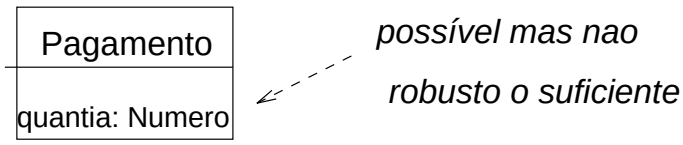


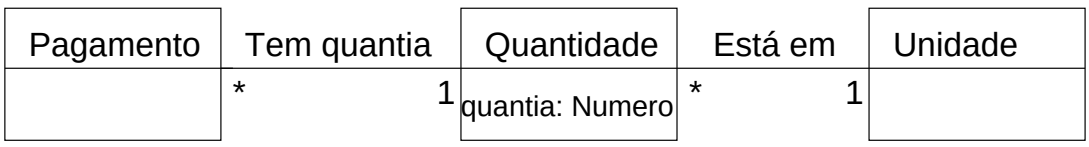
(2)

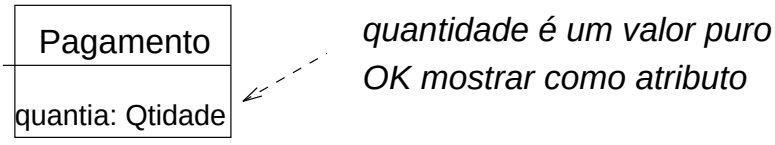
Exemplos de Usos de Atributos



Exemplo: Uso de Atributos Puros







Para modelar quantidades - valores de dados puros

Como fazer um modelo conceitual

- Listar conceitos candidatos - lista de conceitos e identificação de nomes nas frases que descreve os requisitos
- Desenhar o modelo conceitual
- Adicionar as associações necessárias para relacionamentos que são necessários preservar
- Adicionar os atributos necessários para preencher informações sobre os requisitos

3. Glossário

Lista todos os termos que requerem alguma explicação para melhorar a comunicação e reduzir mal-entendidos

Termo	**Categoria**	**Comentários**
Comprar Itens	caso de uso	descrição do processo de um cliente realizar uma compra
Venda	conceito	uma transação
Pagamento	conceito	pagamento a vista
Pagamento.quantia: Qtidade	atributo	A quantia apresentada pelo cliente para o pagamento
....	....	....

III Comportamento Dinâmico

Descrever o que o sistema faz sem se preocupar como

1. Diagrama de Sequências do Sistema

É um dado cenário (instância ou caminho percorrido no mundo real) de um caso de uso. Mostra os eventos que os atores externos geram, a ordem que ocorrem e eventos entre sistemas

Eventos do Sistema: é um evento de entrada externa gerado por um ator para o sistema. Ele inicia uma operação como uma resposta

Operação do Sistema: operação executada como resposta a um evento do sistema

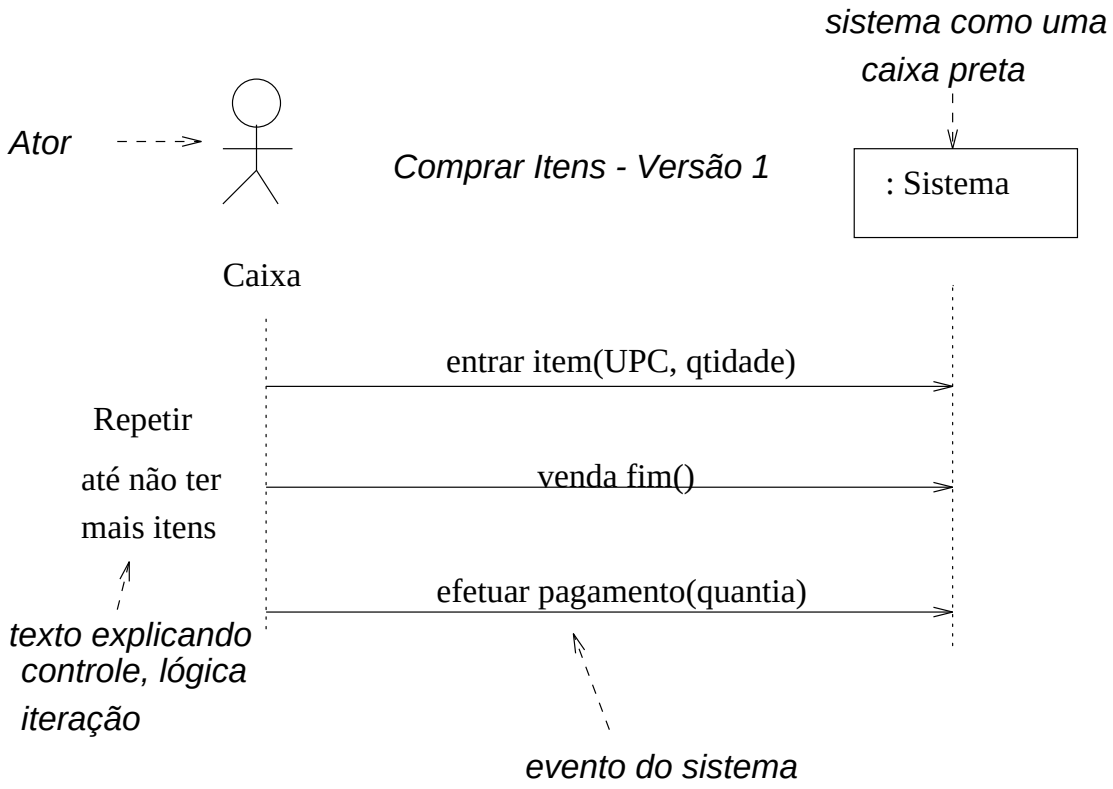
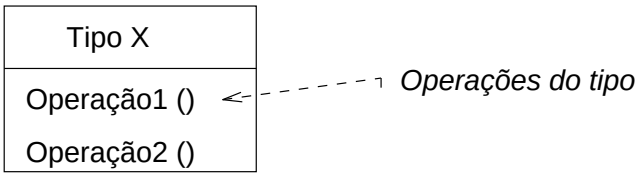


Diagrama de Sequência de Sistemas



Notação para Operações

Como fazer um diagrama de sequências do sistema

Fazer um diagrama para uma sequência típica de eventos para um caso de uso:

1. desenhe uma linha representando o sistema como uma caixa preta

2. identifique cada ator que diretamente opera o sistema. Desenhe uma linha a partir de cada ator.

3. do texto de eventos típicos (caso de uso estendido) identifique eventos que são gerados por cada um dos atores. Ilustre-os no diagrama.

4. Opcionalmente, inclua o texto do caso de uso ao lado do diagrama.

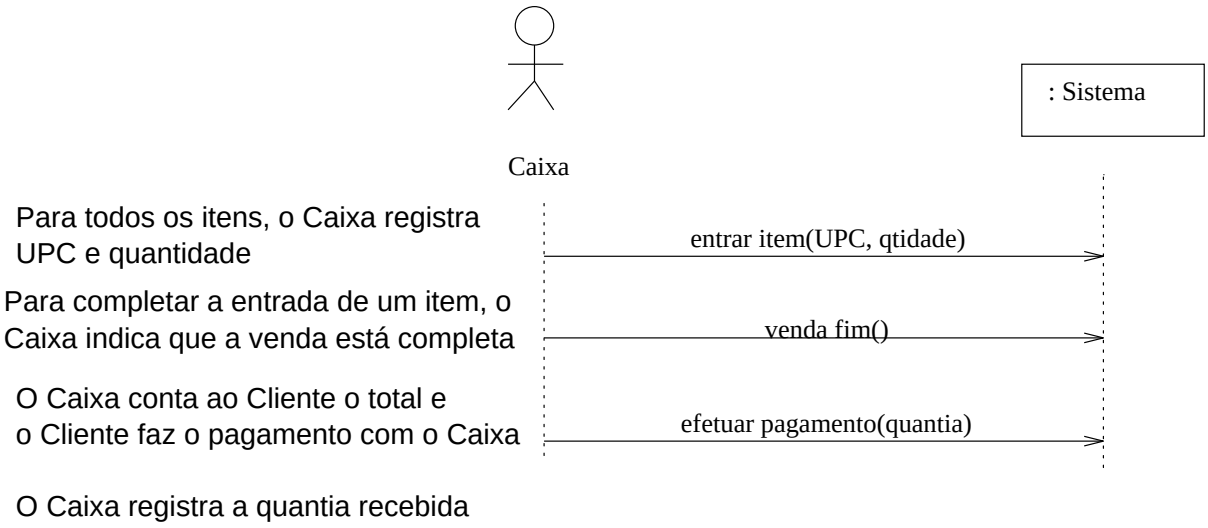


Diagrama de Sequências do Sistema com Texto de Caso de Uso

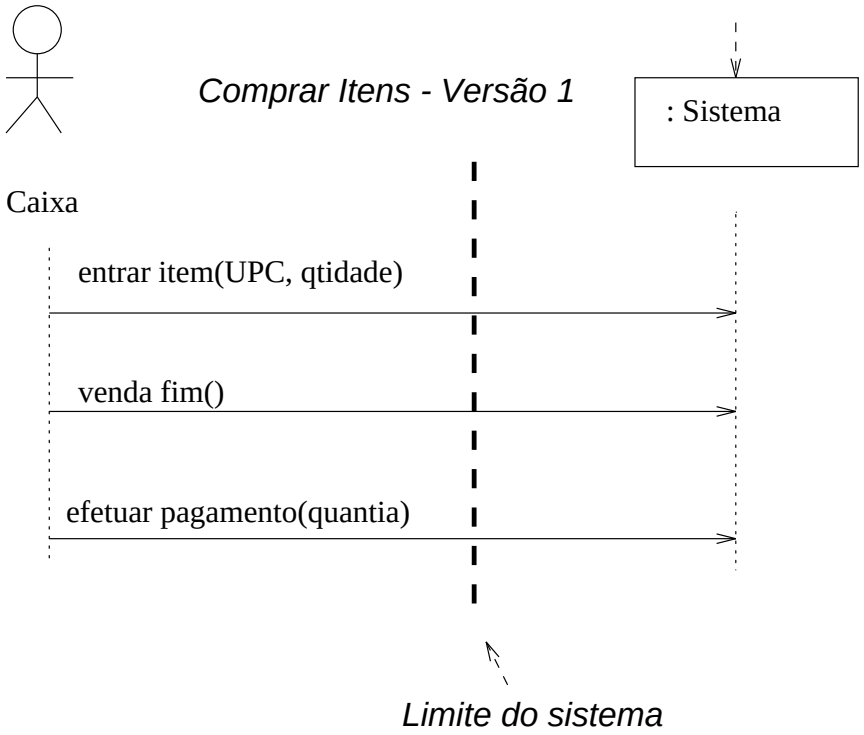


Diagrama de Sequências Mostrando o Limite do Sistema

2. Contratos

Contratos descrevem o efeito das operações do sistema e ajudam a definir o seu comportamento.

É declarativo, porque o interesse está no que acontecerá e não como isso será realizado. São expressos em termos de pré e pós condições.

Contrato	
Nome :	nome da operação e parâmetros
Responsabilidades:	responsabiliades a serem cumpridas
Tipo:	nome do tipo (conceito, classe de software, interface)
Referências:	funções dos sistema, casos de usos etc
Notas:	notas de projeto, algoritmos, detalhes de implementação
Excessões :	casos excepcionais
Saída:	mensagens e registros enviados para fora do sistema (outros sistemas)
Pré-Condições:	suposições sobre o estado sistema antes da execução da operação
Pós-Condições:	o estado do sistema após o término da operação.

Contrato - Um Exemplo

Nome :	entrar item (upc:number; quantidade:inteiro)
Responsabilidades:	registrar um item de venda e adicioná-lo à venda; mostrar preço e descrição do item
Tipo:	sistema
Referências:	funções: R1.1, R1.3, R1.9 casos de usos: Comprar Itens
Notas:	acesso a uma base de dados
Excessões :	se UPC não for válida mostrar msg de erro
Saída:	
Pré-Condições:	deve-se conhecer valor de UPC
Pós-Condições:	

1. se a compra é nova, uma nova Venda (instância) foi criada
2. se a compra é nova, a nova Venda foi associada com o Posto comercial (uma associação foi estabelecida)
3. um Item da linha de Venda foi criado (instância criada)
4. Item da linha de Venda foi associado com Compra (asossiação estabelecida)
5. O atributo Item da linha de Venda.quantidade recebe um valor (modificação de um atributo)
6. Um Item da linha de Venda foi associado com uma Especificação de Produto, baseado no valor da UPC (asossiação estabelecida)

Como fazer um Contrato

1. identificar as operações do sistema do diagrama de sequências do sistema

2. para cada operação encontrada, construir um contrato

3. comece escrevendo a seção de responsabilidades, o propósito da operação

4. complete a seção de pós-condições, descrevendo mudanças de estado em objetos no modelo conceitual

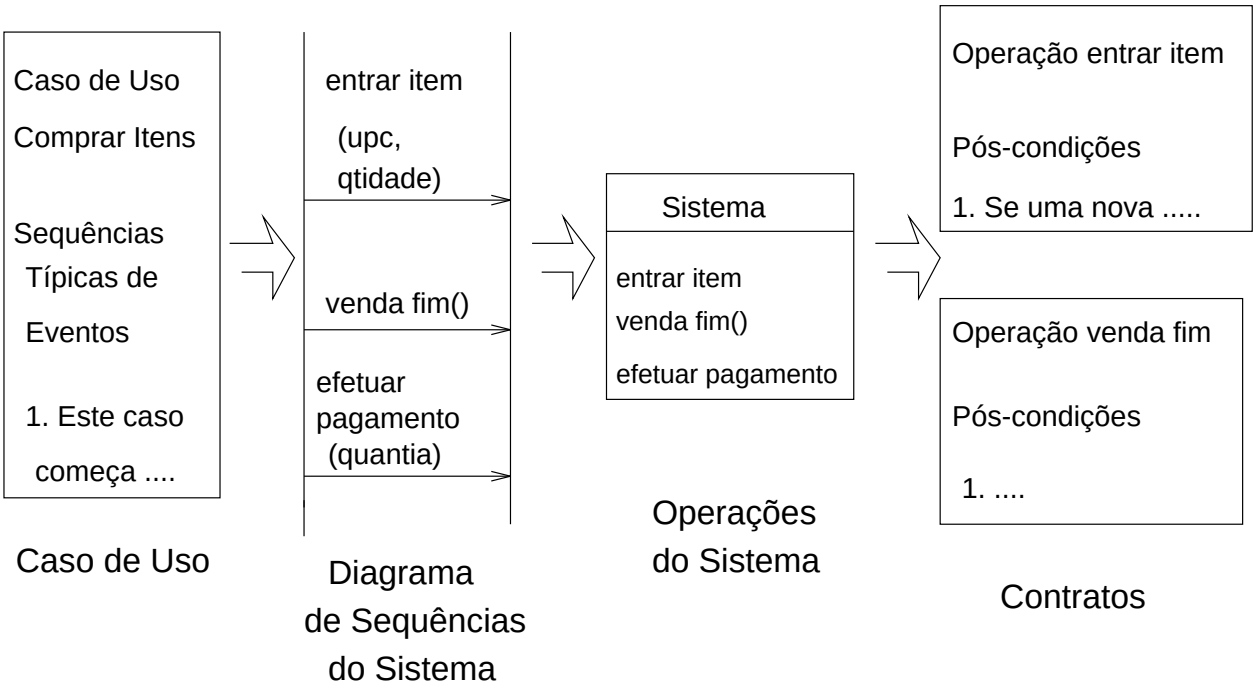
5. descreva pós-condições nas seguintes categorias: criação e remoção de instâncias, modificação de atributos, associações estabelecidas e rompidas.

   * pós-condições são declarações do estado do sistema; não são ações a tomar, devem estar no passado.

6. descreva pré-condições

   * pré-condições são fatos importantes que precisam ser testados.

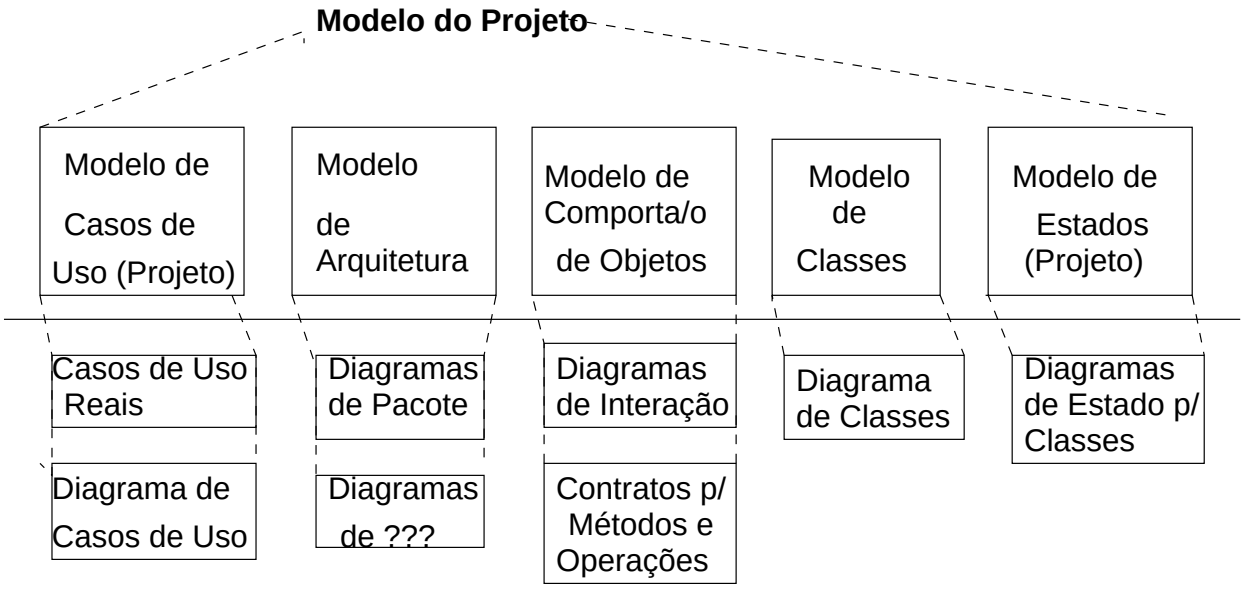
7. além de criar instâncias é importante estabelecer associações.



Contratos e Outros Artefatos

Da Análise para o Projeto

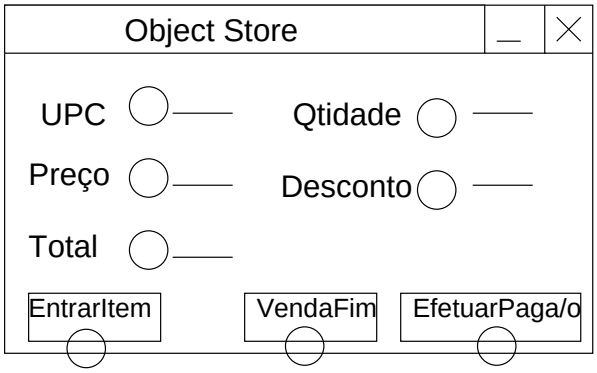
Artefato da Análise	Questões Respondidas
Casos de Uso	Quais os processos do domínio?
Modelo Conceitual	Qauis os conceitos, termos?
Diagrama de Sequencias do Sistema	Quais os eventos do sistema e as operações?
Contratos	O que as operações do sistema fazem?



I. Casos de Uso Reais

Descrevem algum detalhe de implementação do caso de uso, em termos de tecnologia utilizada.

Caso de Uso	**Compra Itens (versão 1)**
Atores:	Cliente, Caixa
Propósito	Registrar uma venda e seu pagamento
Resumo:	Um Cliente chega para compra. O Caixa registra os itens de compra e recebe o pagamento. Ao final o Cliente vai embora
Tipo:	pincipal e real
Referências	Funções: R1.1, R1.3, R2.1



Sequência Típica de Eventos

Ação dos Atores	Resposta
.....	.....

II. Diagramas de Interação

Ilustram como objetos interagem via mensagens e como realizam tarefas. As pós-condições dos contratos são cumpridas.

Diagrama de Colaboração

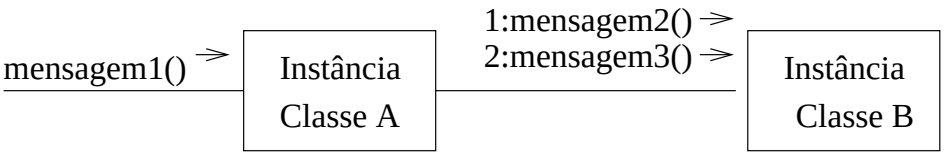
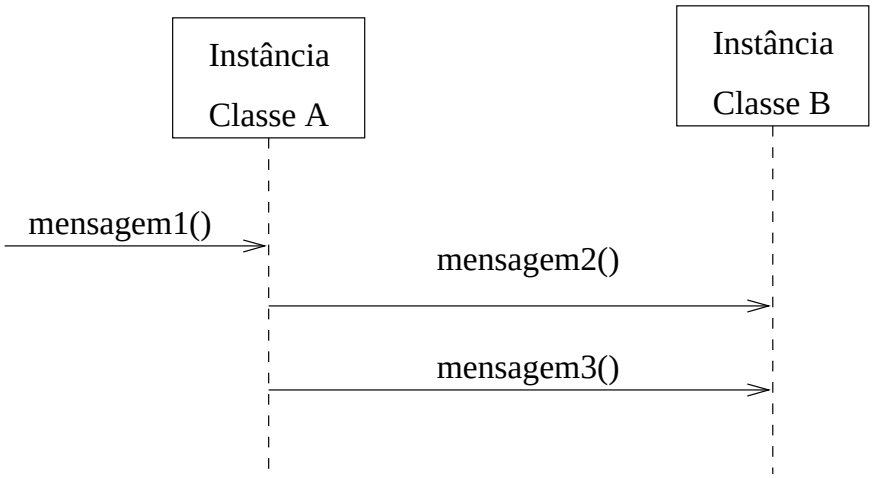
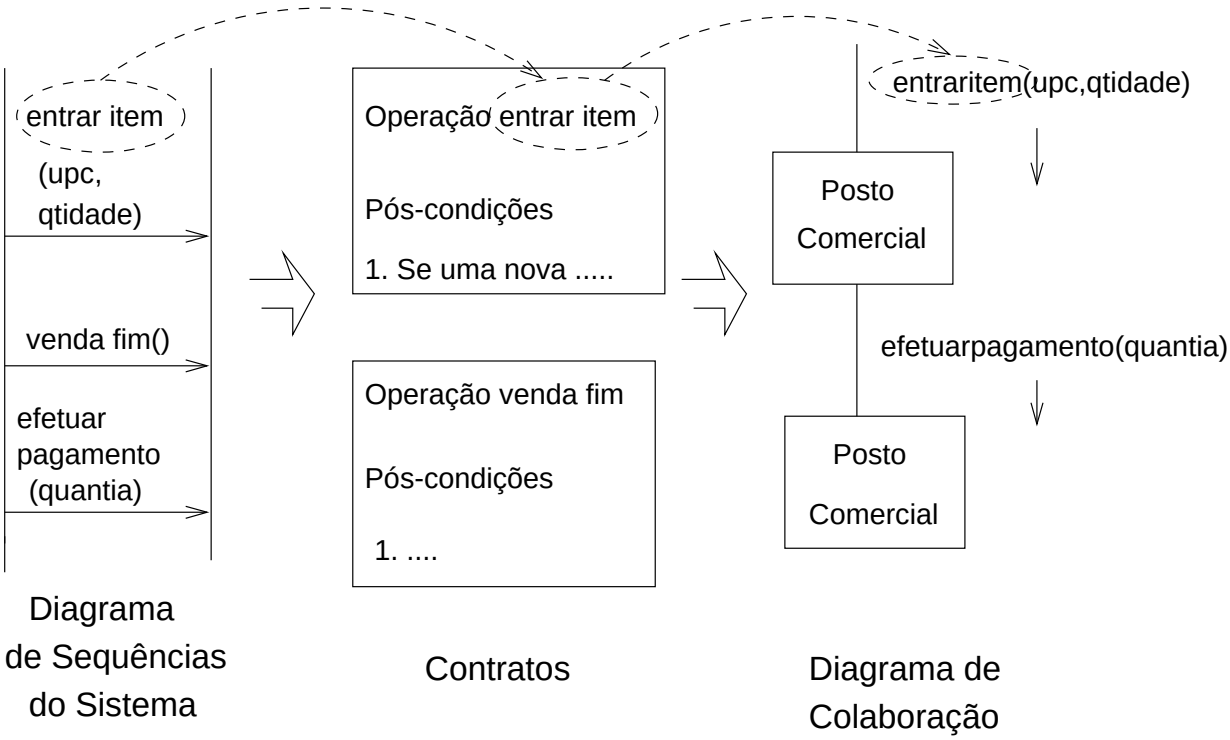


Diagrama de Sequência



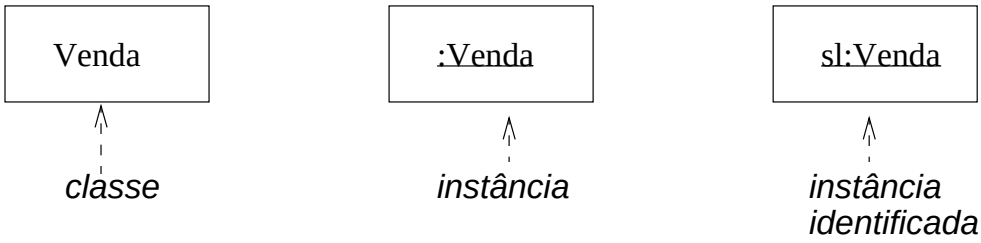
Como Fazer um Diagrama de Colaboração

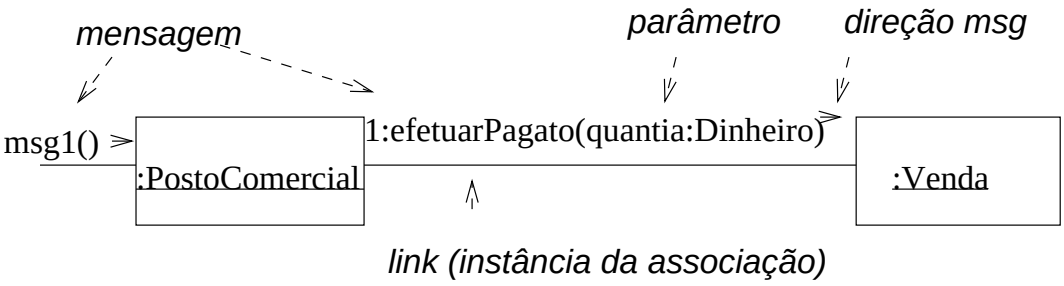
- Atribuir responsabilidades: parte difícil e importante
- Auxílio: uso de "patterns" que são diretrizes e princípios a serem aplicados
- Criar um diagrama para cada operação do sistema; para cada msg de operação do sistema, fazer um diagrama com ela sendo a msg inicial
- Se o diagrama ficar complexo, divida-o em diagramas menores
- Usando responsabilidades e pós condições dos contratos e, a descrição de caso de uso como ponto inicial, projete objetos interagindo para completar as tarefas.
- Aplique padrões para fazer um bom projeto e para estabelecer os métodos e operações de cada objeto.

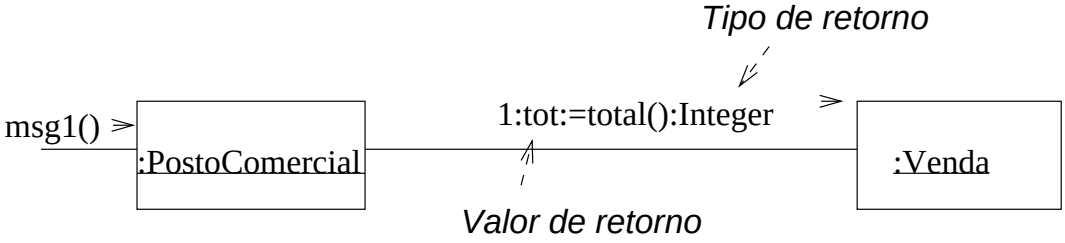


Diagramas de Colaboração e Outros Artefatos

Notação para o Diagrama de Colaboração

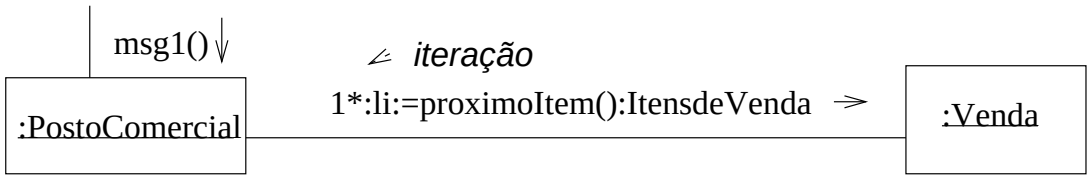


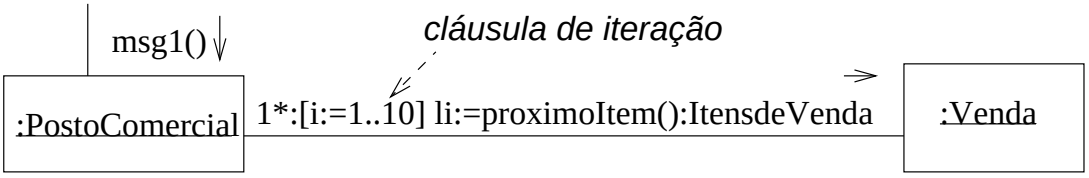




retorno :=mensagem(parâmetro:TipoParâmetro):TipoRetorno

Ilustrando Iterações em Diagramas de Colaboração





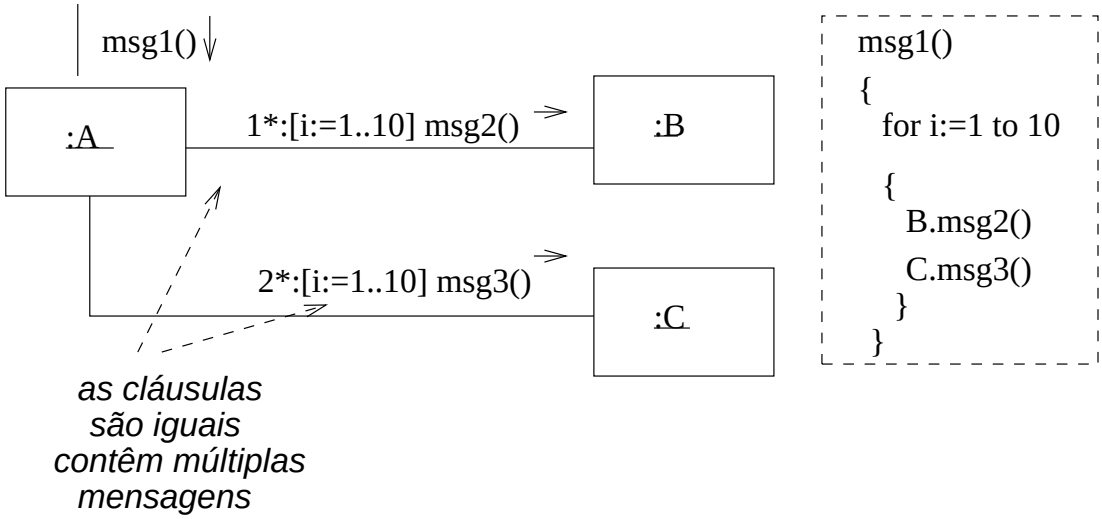
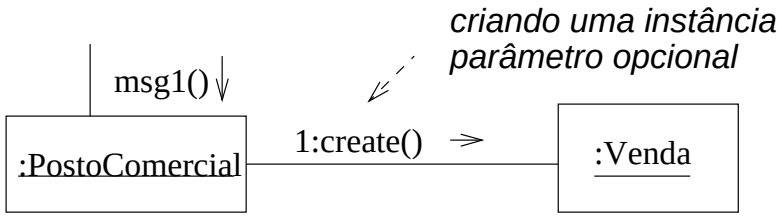
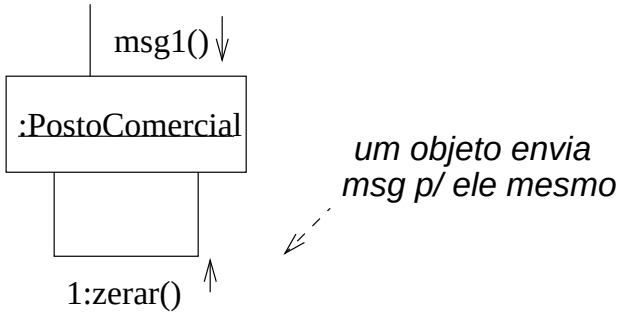
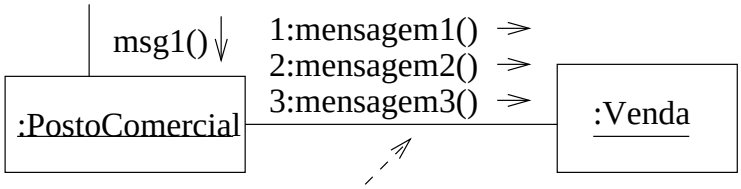


Diagrama de Colaboração - Notações/Exemplos

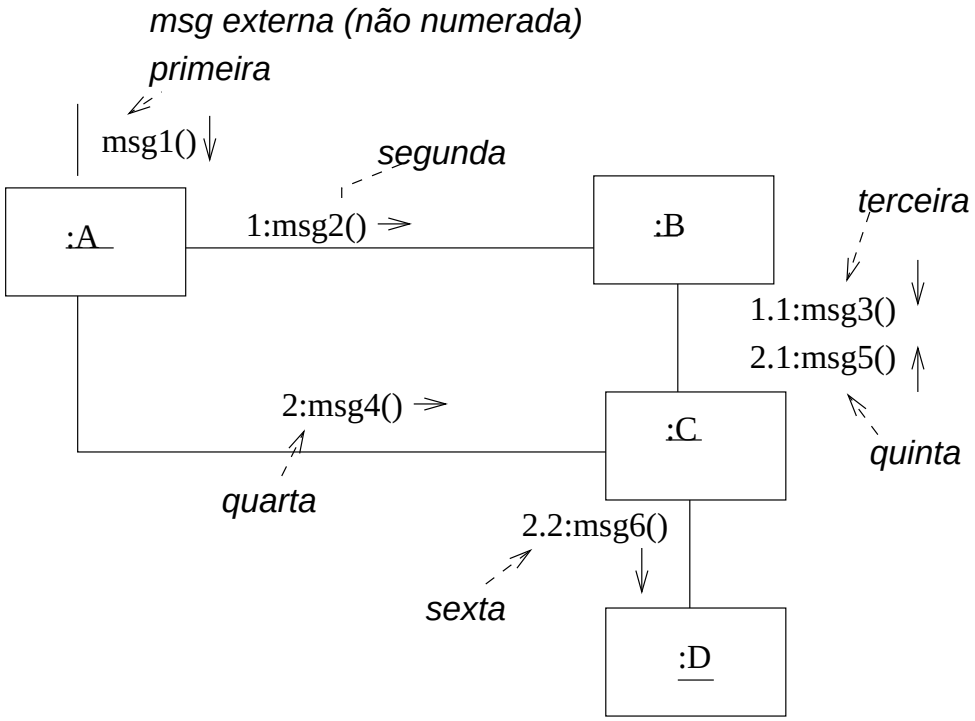




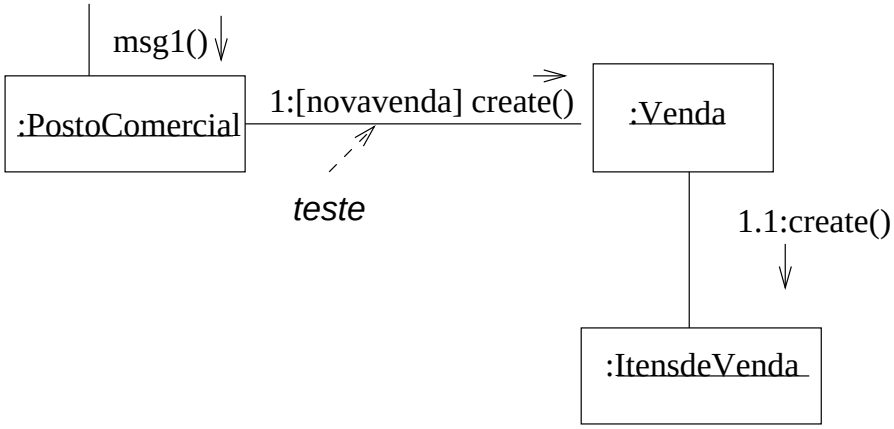


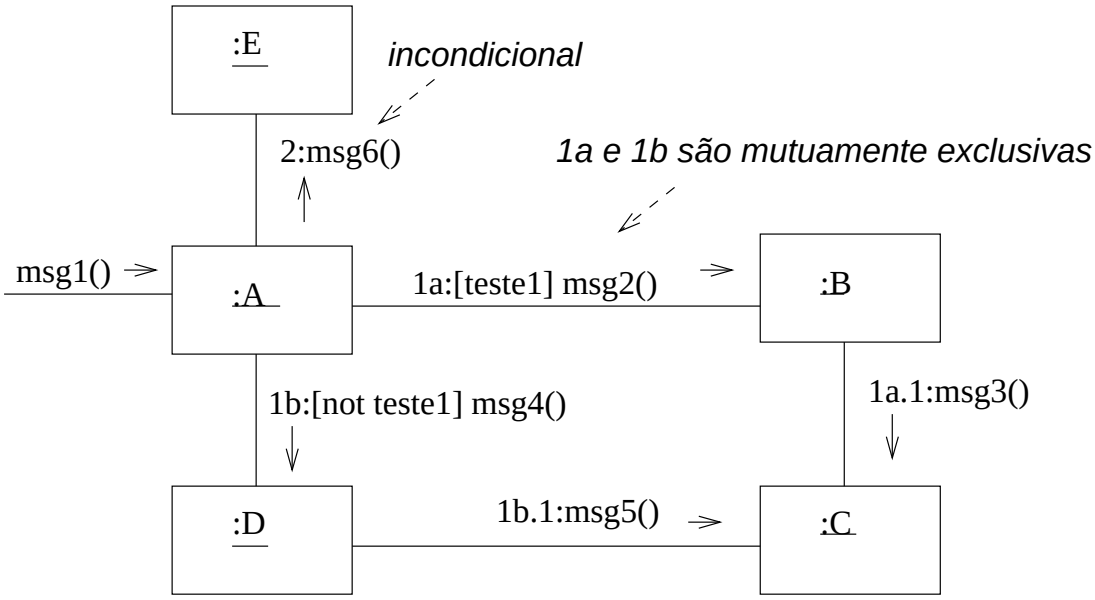
várias mensagens pelo mesmo link

Ilustrando Sequenciamento de Mensagens

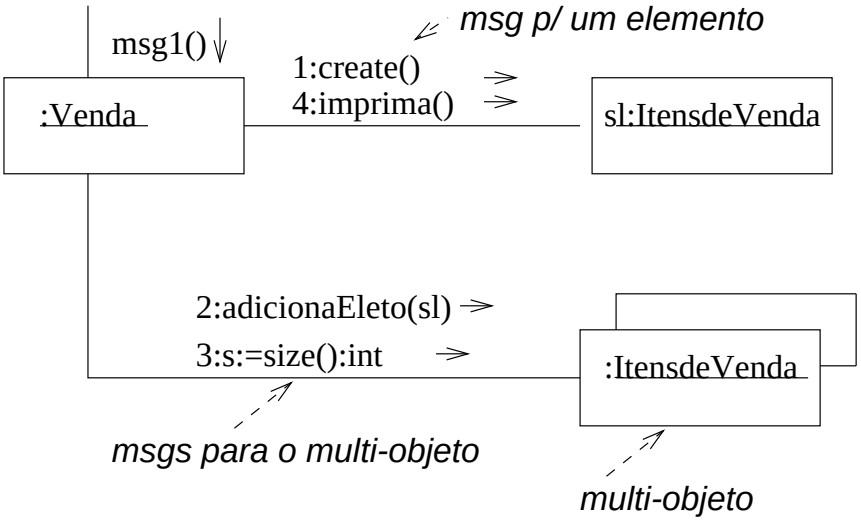


Ilustrando Mensagens Condicionais

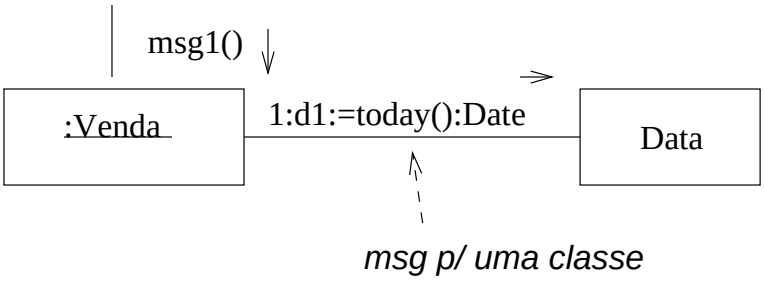




Ilustrando Mensagens para MultiObjetos



Ilustrando Mensagens para Classes



Padrões GRASP

Responsabilidades: Contrato ou obrigação de uma classe:

saber: saber sobre os seus dados, sobre objetos relacionados, sobre coisas que ele pode calcular ou derivar.

fazer: iniciar ações entre outros objetos, controlar e coordenar atividades em outros objetos

Métodos: cumprem as responsabilidades.

Padrões: contêm descrições de um $problema$ e uma $solução$ que pode ser utilizada em diferentes contextos.

Codificam idéias e heurísticas existentes para atribuir responsabilidades a objetos.

Padrões GRASP básicos: especialista, criador, alta coesão, baixo acoplamento, controle.

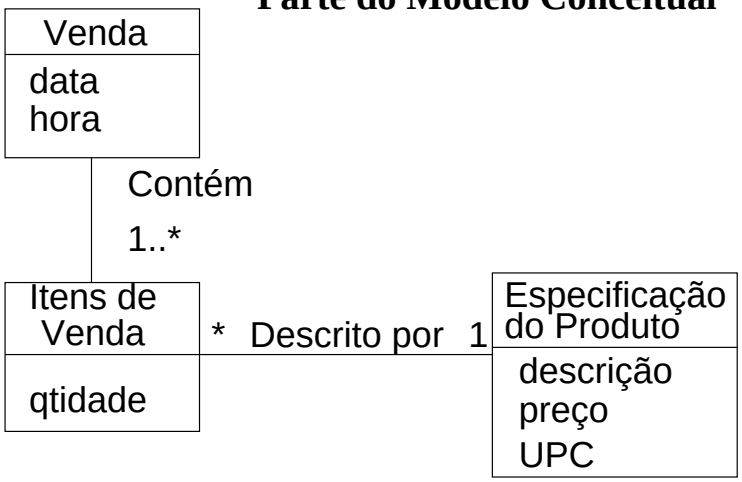
Padrão Especialista (Expert)

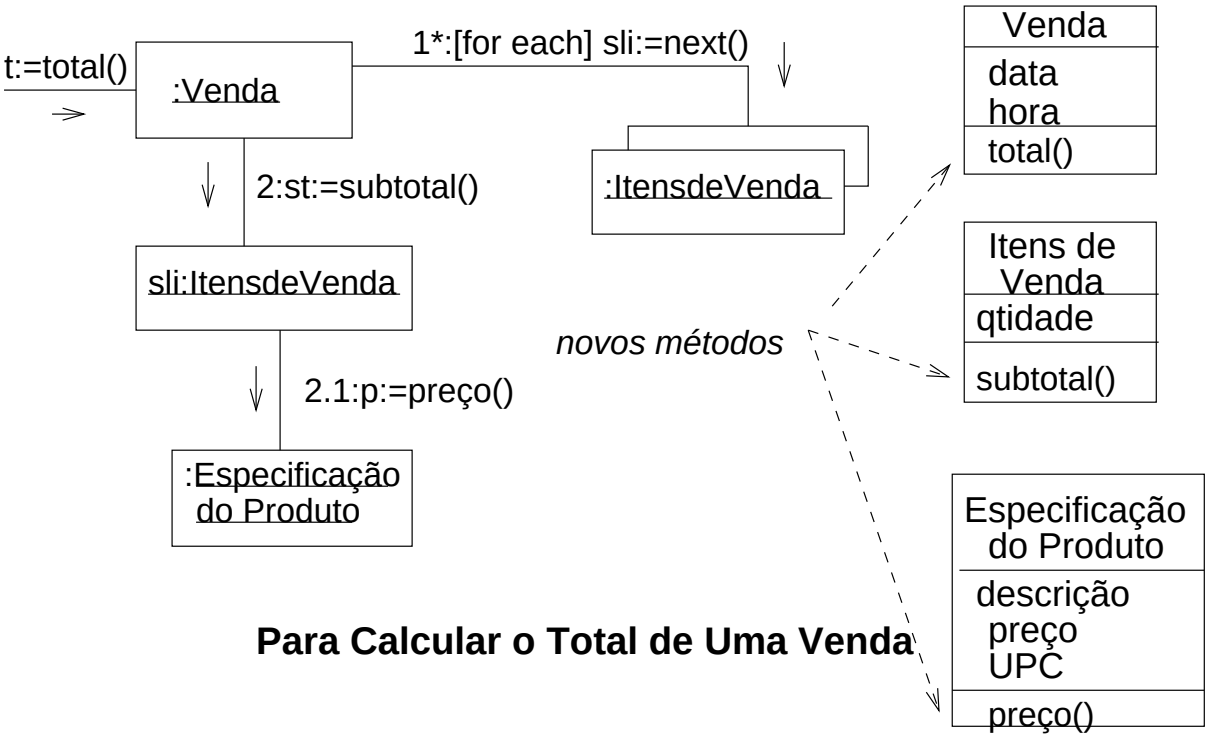
Solução: Atribua uma responsabilidade a uma classe que possui a informação necessária para cumprí-la.

Problema: Quem deveria ser responsável por calcular o total-geral de uma venda? *a classe Venda possui a informação para isso*

Classe	**Responsabilidade**
Venda	sabe total geral da venda
Item de Venda	sabe sub-total de cada item
Especificação do Produto	sabe preço do produto

Parte do Modelo Conceitual





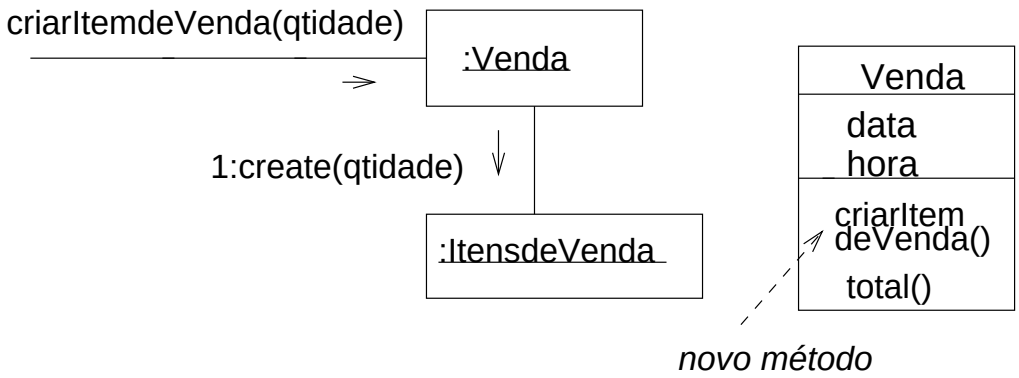
Para Calcular o Total de Uma Venda

Padrão Criador (Creator)

Solução: Atribua a classe B a responsabilidade de criar uma instância da classe A se:

1. B agrega objetos de A
2. B contém A
3. B armazena instâncias de A
4. B usa objetos de A
5. B possui informação necessária a criação de A (B é um especialista para criar A)

Problema: Quem deveria ser responsável por criar a instância da classe Item de Venda? *classe Venda agrega muitos objetos da classe Itens de Venda*



Para Criar uma Linha de Item de Venda

Padrão Baixo Acoplamento e Padrão Alta Coesão

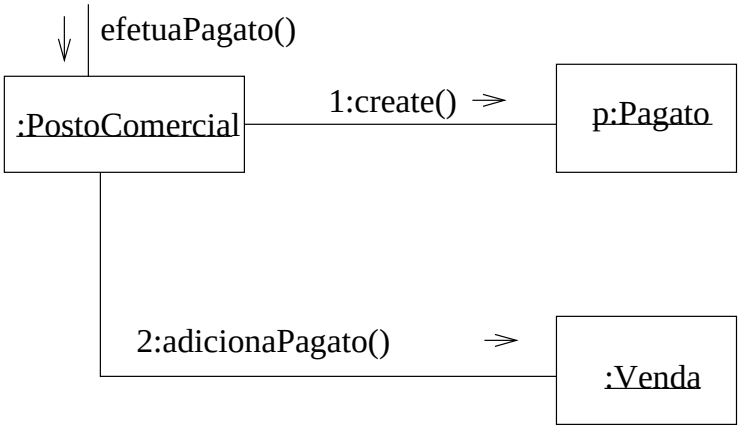
(Low Coupling e High Cohesion)

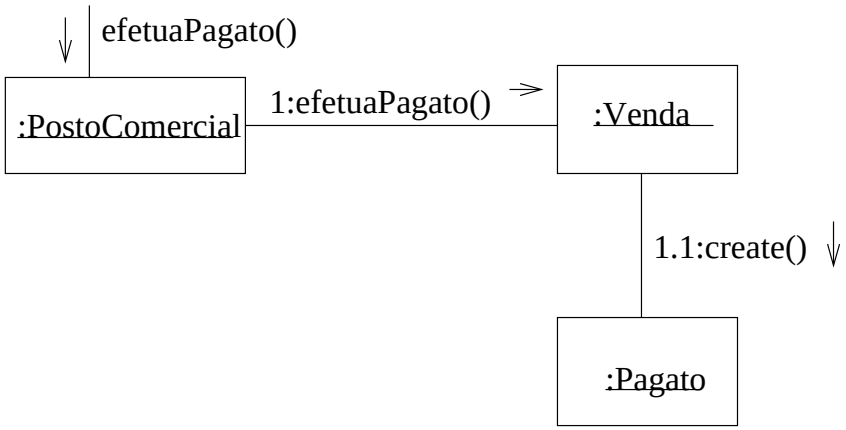
Solução: Atribua responsabilidades para garantir baixo acoplamento (mede a independência da classe em relação a outras) e alta coesão (medida de relacionamento entre as responsabilidades de uma classe)

Problema: Para evitar que a classe seja:

1. constantemente afetada por mudanças em outras classes
2. difícil de compreeder quando isolada
3. difícil de ser reusada sem as outras classes
4. difícil de manter

que classe poderia criar uma instância da classe Pagato?





Ilustrando Alta Coesão e Baixo Acoplamento

Padrão Controle (Controller)

Solução: Atribua responsabilidades para lidar com mensagens de eventos do sistema a uma classe que:

1. represente o sistema como um todo
2. represente a organização
3. represente algo ativo no mundo real envolvido na tarefa (por exemplo o papel de uma pessoa)
4. represente um controlador artificial dos eventos de sistema de um caso de uso (controlador de caso de uso)

Problema: Quem deveria ser responsável por lidar com um evento do sistema?

um controlador é um objeto de interface responsável por gerenciar um evento do sistema, definindo métodos para operações do sistema

Quem deveria ser o controlador para eventos do sistema tais como entrarItem e Vendafim?

Padrão Controle

* Use o mesmo controlador para todos os eventos do sistema no mesmo caso de uso

* Classes do tipo janela, applet, aplicações, documento não deveriam realizar tarefas associadas a eventos do sistema. Elas apenas recebem e delegam os eventos ao controlador

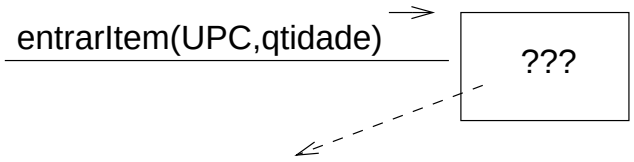
* Um evento do sistema é gerado por um ator

* Um controlador não deve ter muitos atributos e nem manter informação sobre o domínio do problema

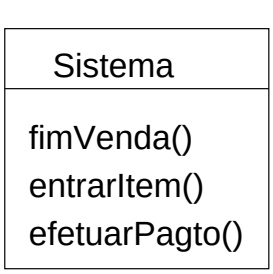
* Um controlador não deve realizar muitas tarefas, apenas delegá-las

* Um bom projeto deve dar vida aos objetos, atribuindo-lhes responsabilidades, até mesmo se eles forem seres inanimados no mundo real

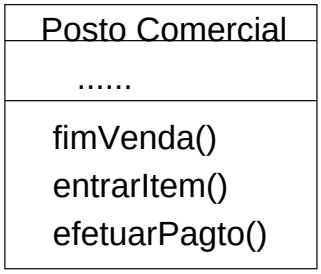
* A camada de apresentação (interface com o usuário) não deve tratar eventos do sistema



posto comercial, loja, caixa ....

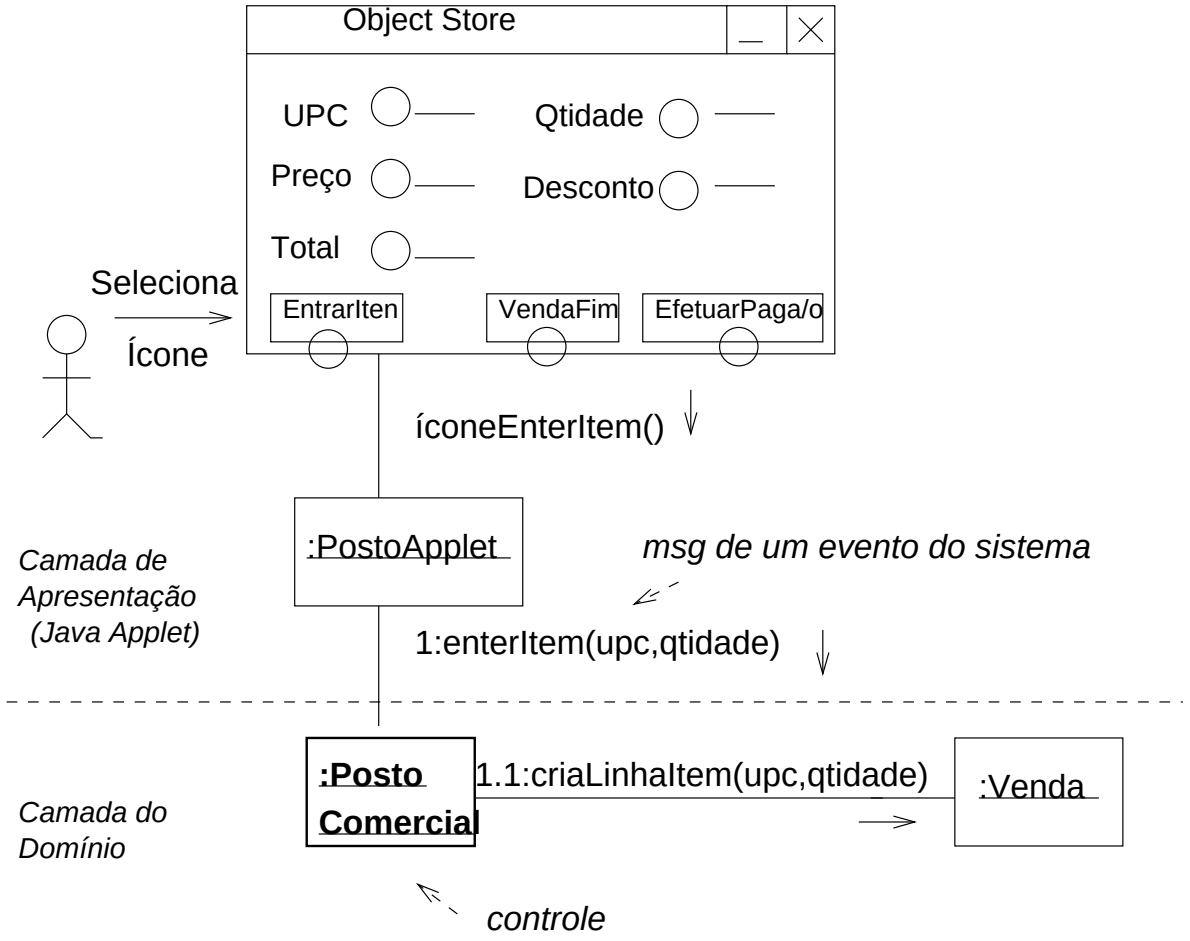


operações encontradas durante a análise do comportamento do sistema



alocação das operações do sistema durante projeto usando o padrão Controle

Exemplificando o Uso do Padrão Controle

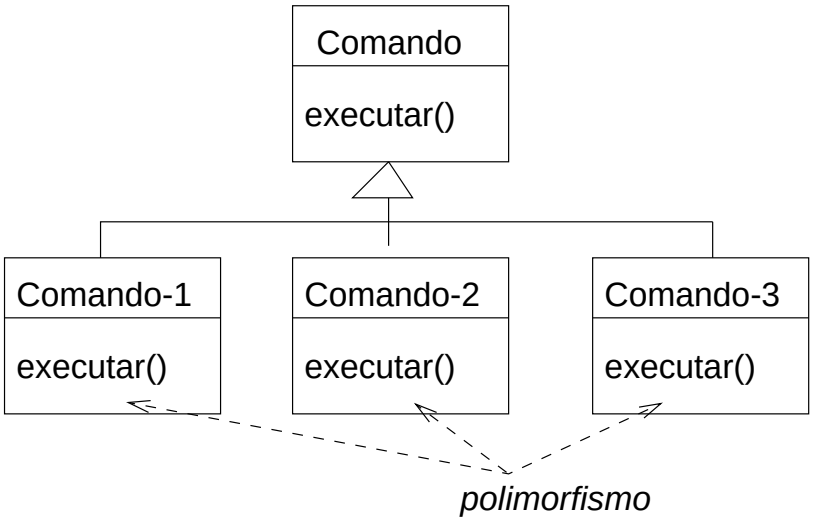


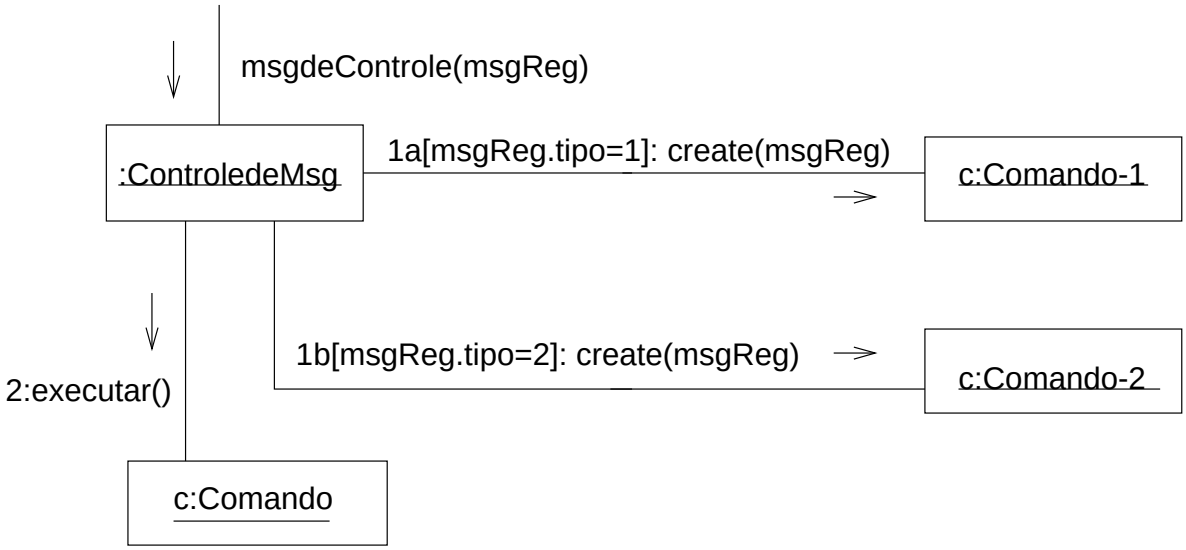
Acoplando a Camada de Apresentação à do Domínio

Padrão Comando

Problema: Aplicações que recebem msg de um sistema externo (não existe interface com o usuário). Ex: sistemas de telecomunicações

Solução: definir uma classe para cada comando, com o método executar. O controlador criará uma instância da classe correspondente a msg do evento do sistema e enviará a mensagem executar à classe comando.





Usando os Padrões Comando e Controlador

Exemplificando o Uso de Padrões

Para fazer um Diagrama de Interação:

- utilize as responsabilidades e pós-condições dos contratos e use casos de usos como ponto de partida

- escolha a classe que controlará o sistema, aplicar padrão controle

- para cada operação existe um contrato, uma operação vai ser uma mensagem.

- separação do modelo de visão: não é responsabilidade dos objetos do domínío se comunicarem com o usuário (ignorar responsabilidades de apresentação dos dados no display, mas toda informação do domínio de objetos tem que ser mostrada)

- para um objeto enviar uma msg a outro é necessário visibilidade: habilidade de um objeto ver ou fazer referência a outro objeto

Diagrama de Colaboração para EntrarItem

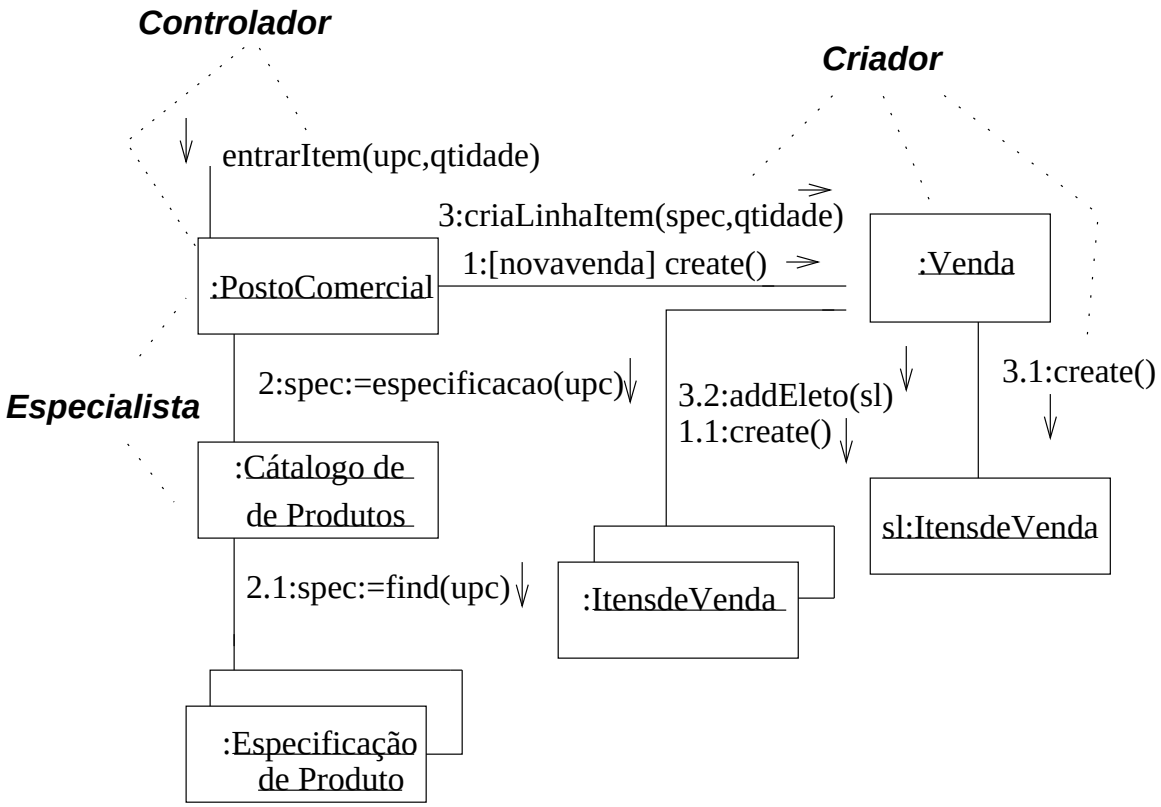


Diagrama de Colaboração para VendaFim

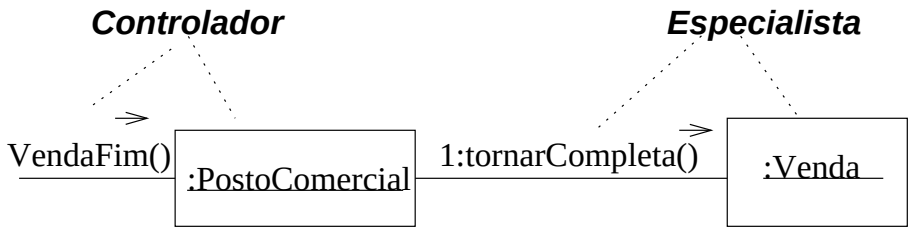


Diagrama de Colaboração para Calcular o Total de Venda

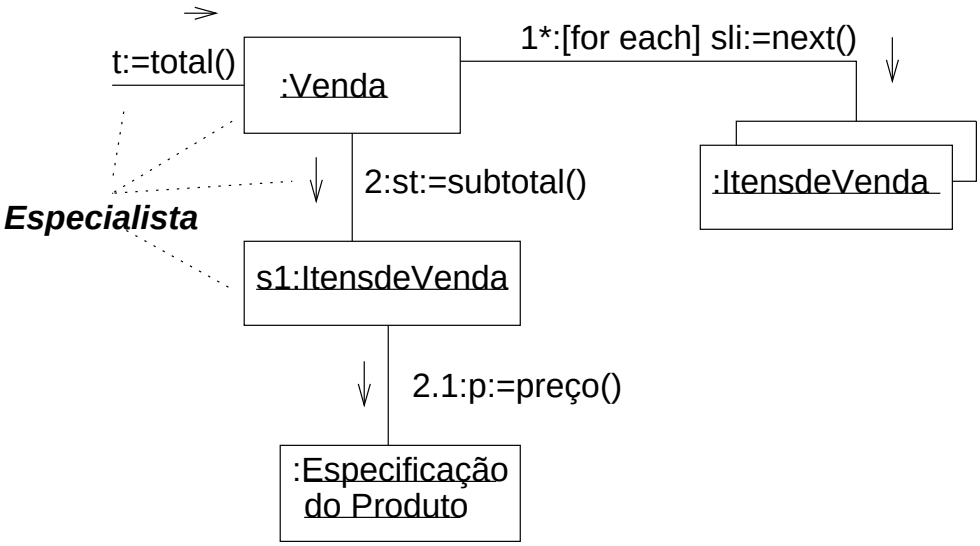


Diagrama de Colaboração para Efetuar Pagamento

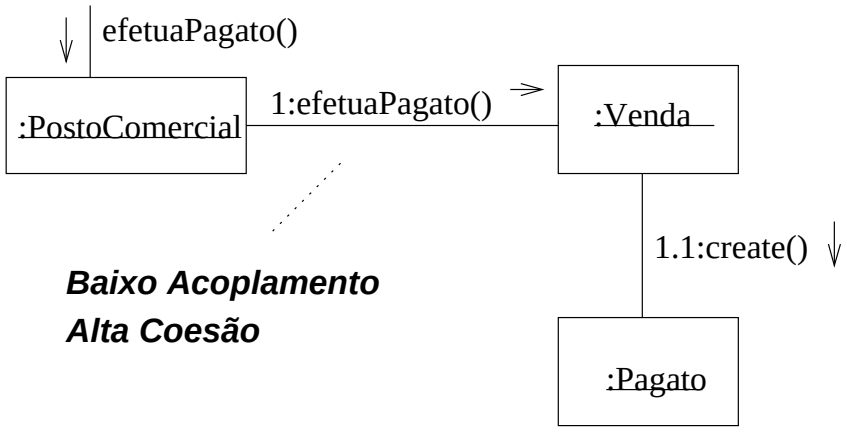


Diagrama de Colaboração para Registrar Venda

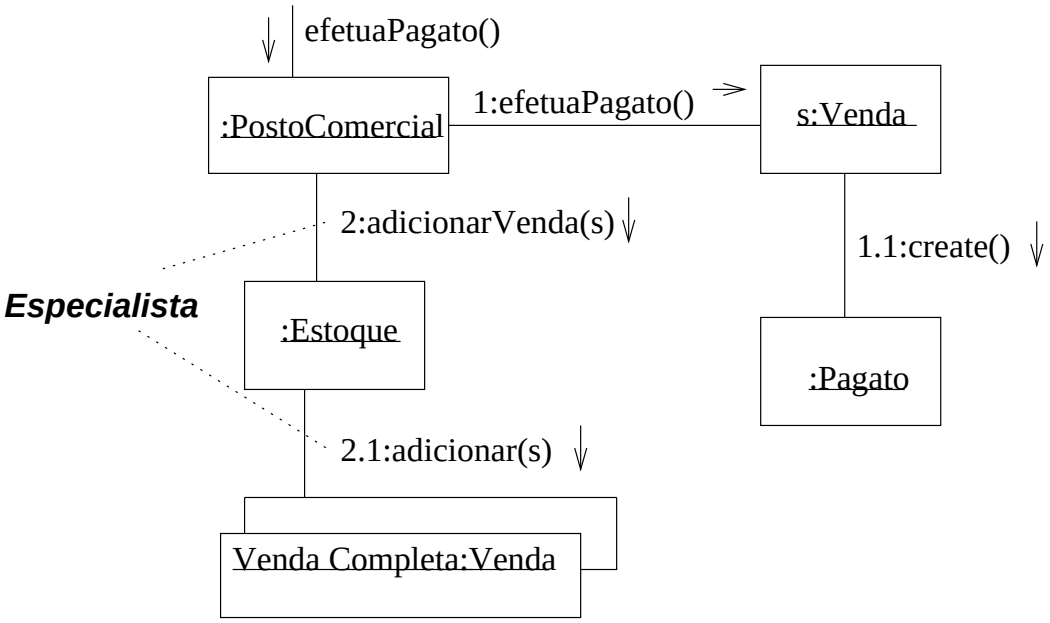
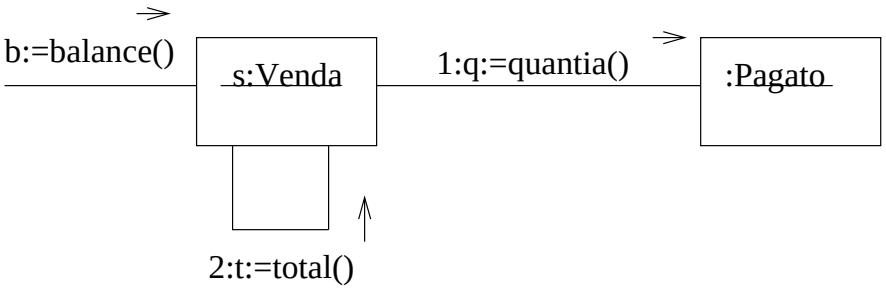


Diagrama de Colaboração para Efetuar Balanço



Como se realizam as operações iniciais da aplicação?

Cria-se um caso de uso startUp.

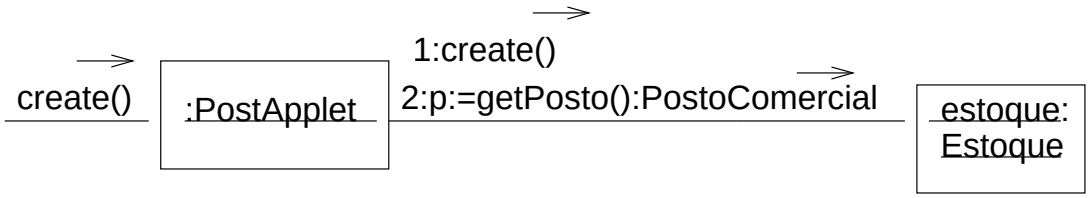
Seu diagrama de colaboração deve ser criado por último, representando o que acontece quando o objeto inicial do problema é criado.

Quem deveria ser o objeto inicial do sistema?

- classe representando toda a informação lógica do sistema

- classe representando a organização

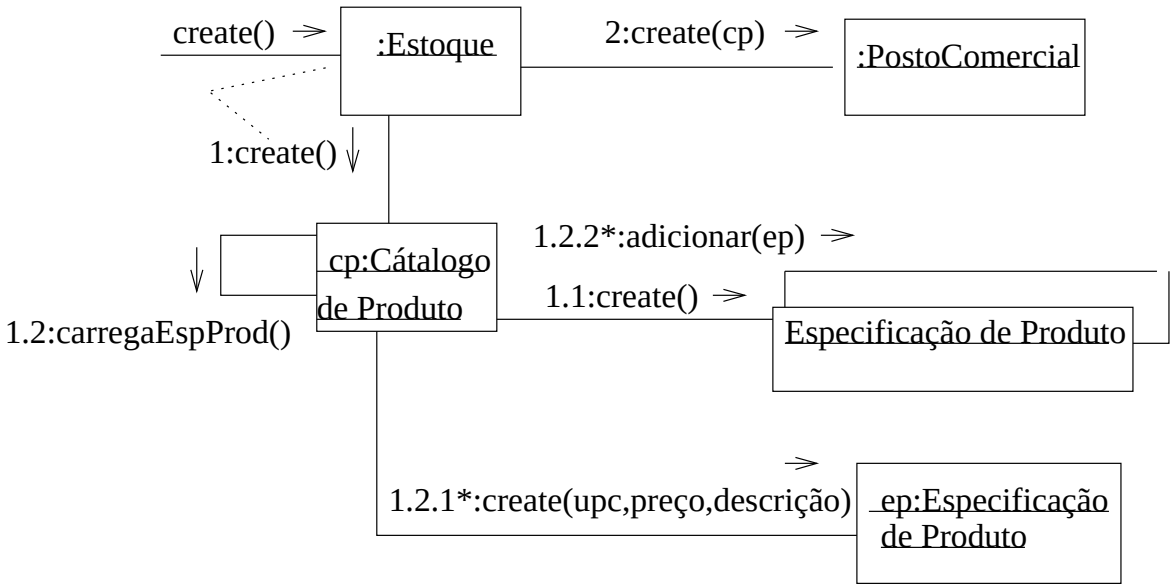
- usar Padrões Alta Coesão e Baixo Acoplamento




```
public class PostApplet extends Applet

public void init()

{
    p=estoque.getPosto
}
```

estoque é o objeto inicial do domínio
e o construtor Store cria os outros objetos.

Criação dos Objetos Iniciais do Domínio



Conectando a camada de apresentação com a do domínio

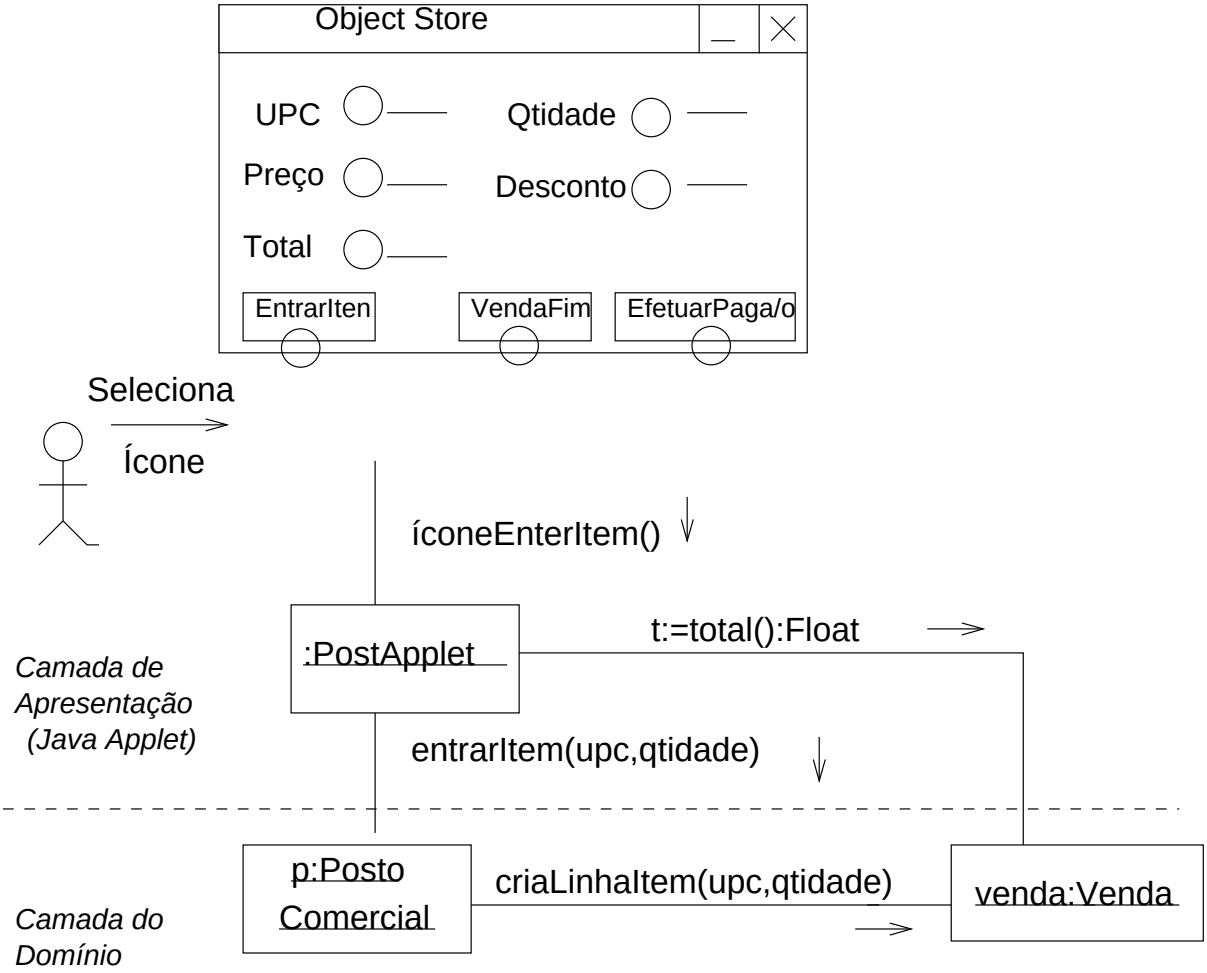
Uma operação de startUp pode ser:

- uma mensagem "create" para o objeto inicial;
- se o objeto inicial é o controlador, uma mensagem "run" para um objeto inicial é enviada.

Se uma interface do usuário estiver envolvida ela é responsável por iniciar a criação do objeto inicial e outros associados.

Objetos da camada de apresentação não devem ter responsabilidades lógicas. Das nossas escolhas resultarão extensibilidade, claridade e manutenibilidade

Conectando a Camada de Apresentação a do Domínio



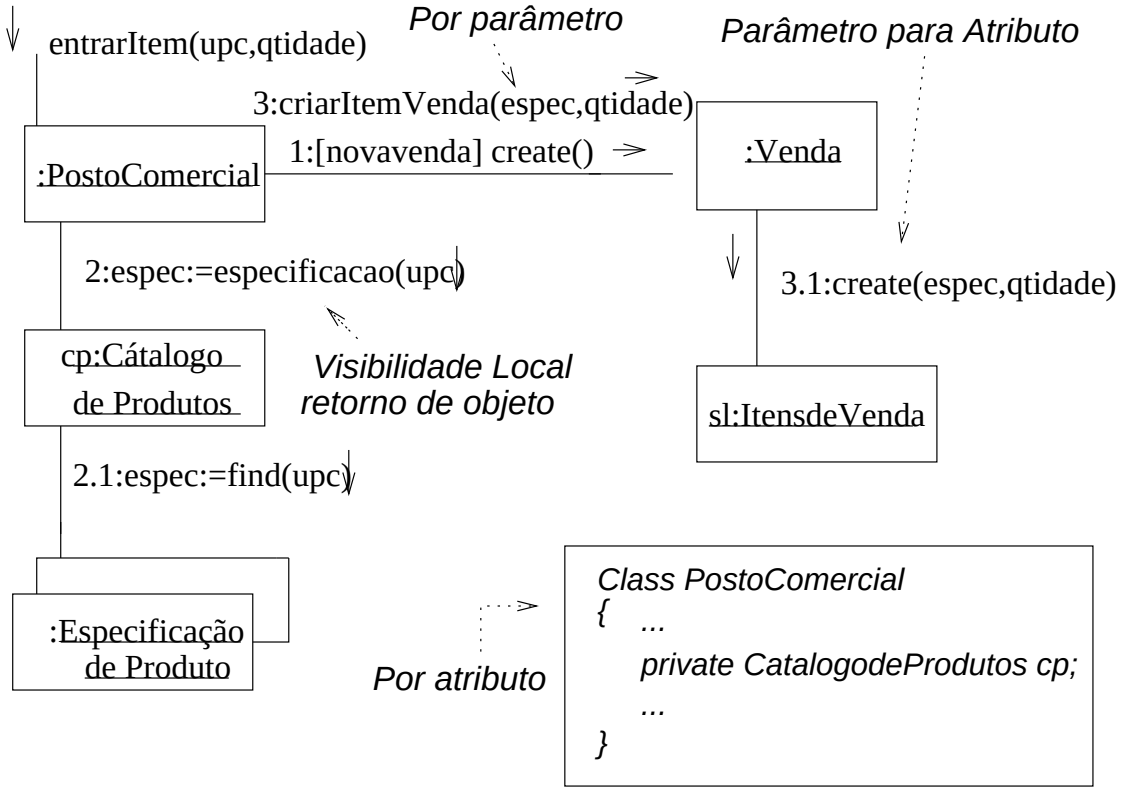
Uma vez que PostApplet tem uma conexão com uma instância de posto p ele pode enviar uma mensagem, por exemplo entrarItem

Visibilidade

Habilidade de um objeto A ver ou fazer referência a um outro objeto B. Para A enviar uma msg para B, B deve ser visível a A.

- por atributo: B é um atributo de A
- por parâmetro: B é um parâmetro de um método de A
- localmente declarada: B é um objeto local de um método de A; uma instância local é criada, ou ele será um valor de retorno
- global: B é visível globalmente; usa-se uma variável global para armazenar uma instância - pouco recomendada

Exemplos de Visibilidade




```
PostoComercial .... entrarItem(upc,qtididade)
 {
    ...
   espec=cp.especificacao(upc);
   ...
 }
```

```
Venda ... criarItemVenda(espec,qtidade)
{
    ...
  sl=new ItensdeVenda(espec,qtidade)
    ...
 }
```

```
construtor ...ItensdeVenda(espec,qtidade)
  {
     ...
     prod=espec;
      ...
  }
```

III. Diagrama de Classes

Modelo Conceitual Estendido – acréscimo de métodos, tipos dos atributos, visibilidade e navegabilidade, inclui a visão de projeto além da visão de domínio

Como fazer um diagrama de classes:

- identificar todas as classes participantes da solução através dos diagramas de colaboração
- desenhar o diagrama
- copiar os atributos
- adicionar os métodos dos diagramas de interação

- adicionar tipos de atributos, parâmetro e retornos de métodos.

- adicionar as associações necessárias a visibilidade

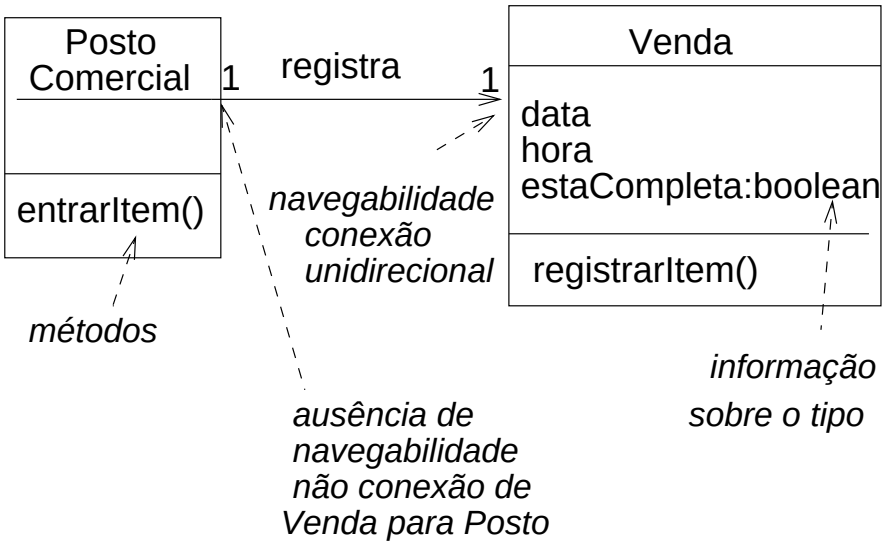
- adicionar navegabilidade que indica a direção de visibilidade por atributo

- adicionar setas pontilhadas indicando visibilidade que não é por atributo

- Métodos "create" e de acessos aos atributos podem ser omitidos.

- Os tipos podem ser mostrados opcionalmente; classes podem ser detalhadas ao máximo pensando na implementação.

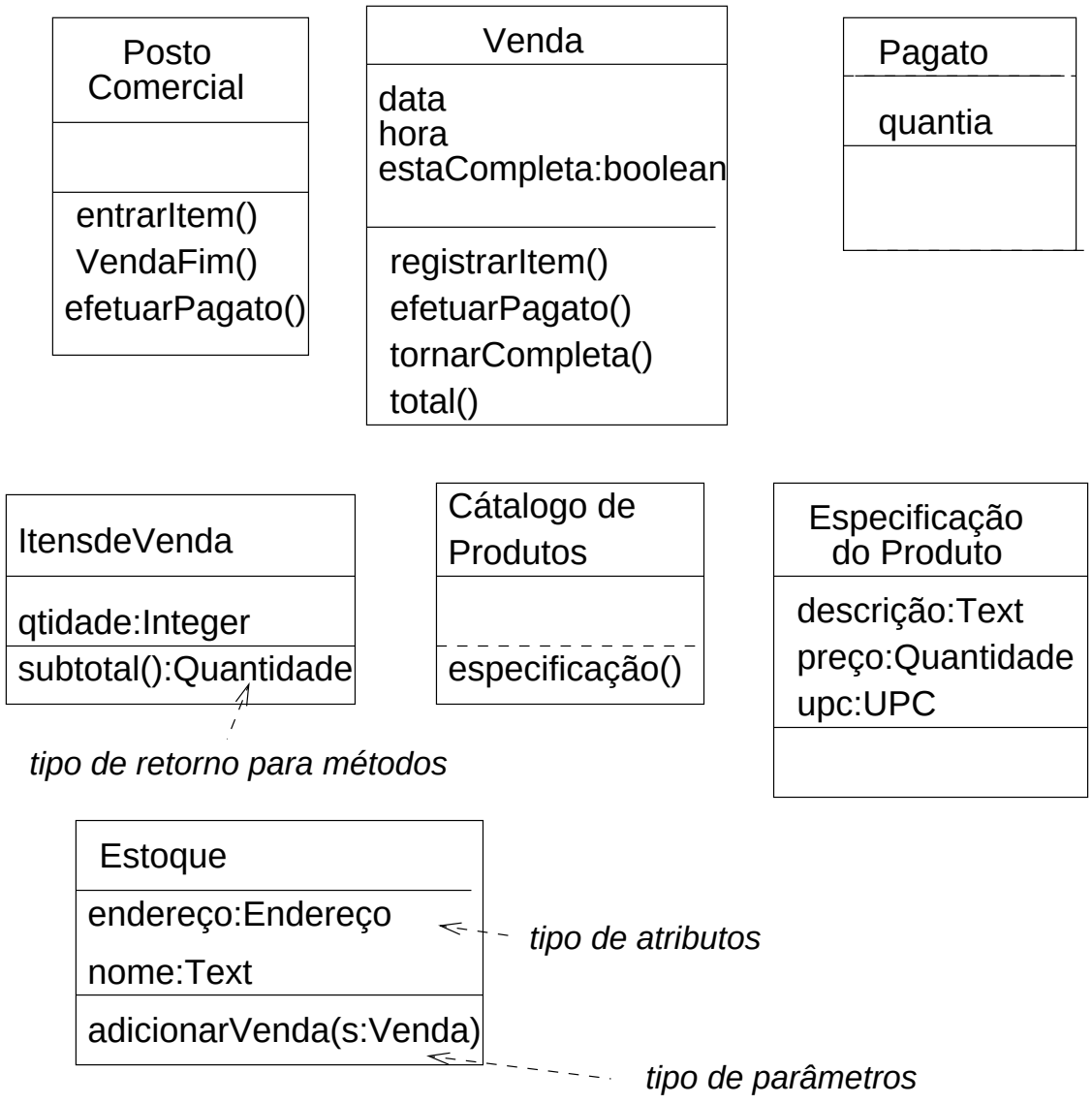
Exemplo - Diagrama de Classes



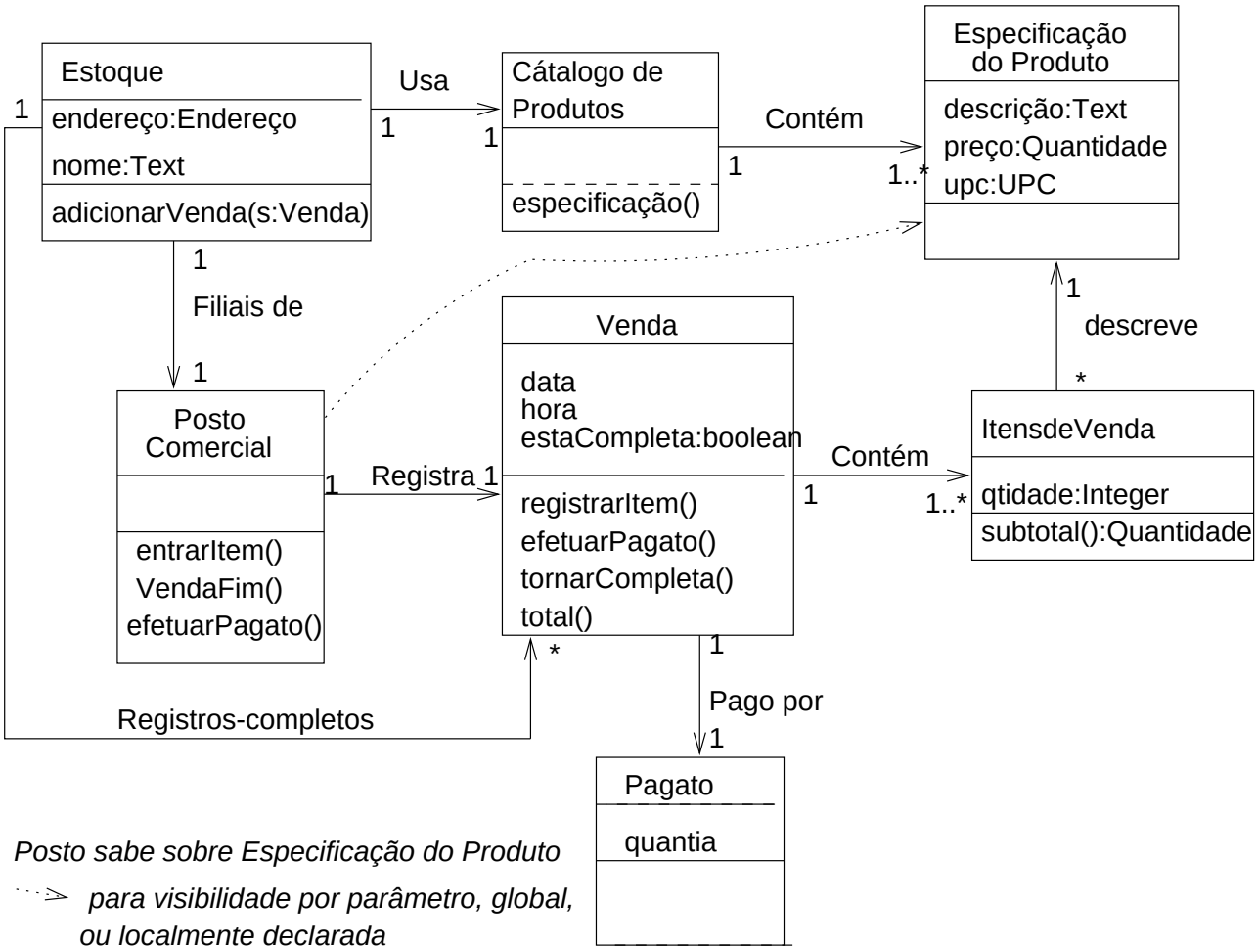
Modelo Conceitual apenas apresenta abstrações.

Não apresenta componentes de software

Métodos são derivados do diagrama de colaboração



Exemplo - Visibilidades que não são por Atributo



Notação para Generalização/Especialização

Quando particionar em Sub-Classes

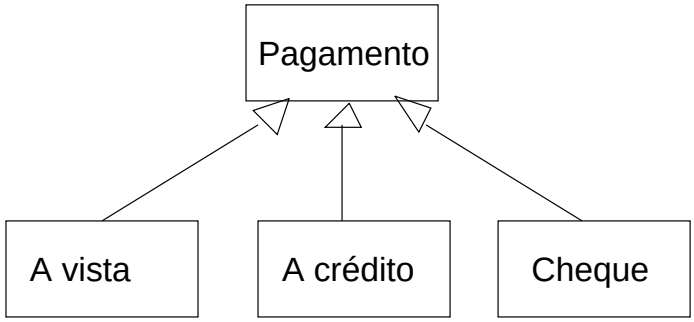
- a sub-classe tem atributos adicionais de interesse
- a sub-classe tem associações adicionais de interesse
- a sub-classe será manipulada ou usada de maneira diferente da super-classe ou das outras sub-classes
- a sub-classe se comporta diferente da super-classe ou das outras sub-classes

Notação para Generalização/Especialização

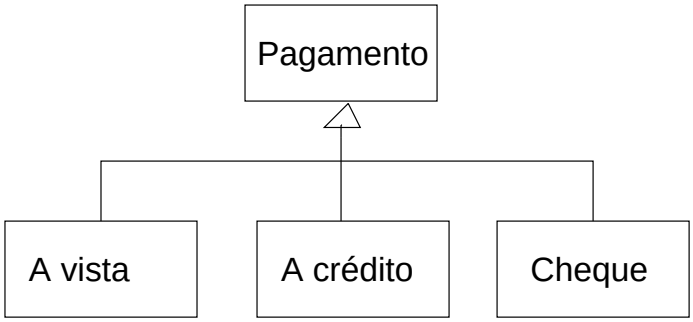
Quando criar uma Super-Classe

- as potenciais sub-classes representam variações de um conceito similar
- as sub-classes satisfazem 100% a regra "is-a"
- todas as sub-classes possuem um atributo comum que poderá ser expresso na super-classe
- todas as sub-classes possuem uma associação comum que poderá ser relacionada à super-classe

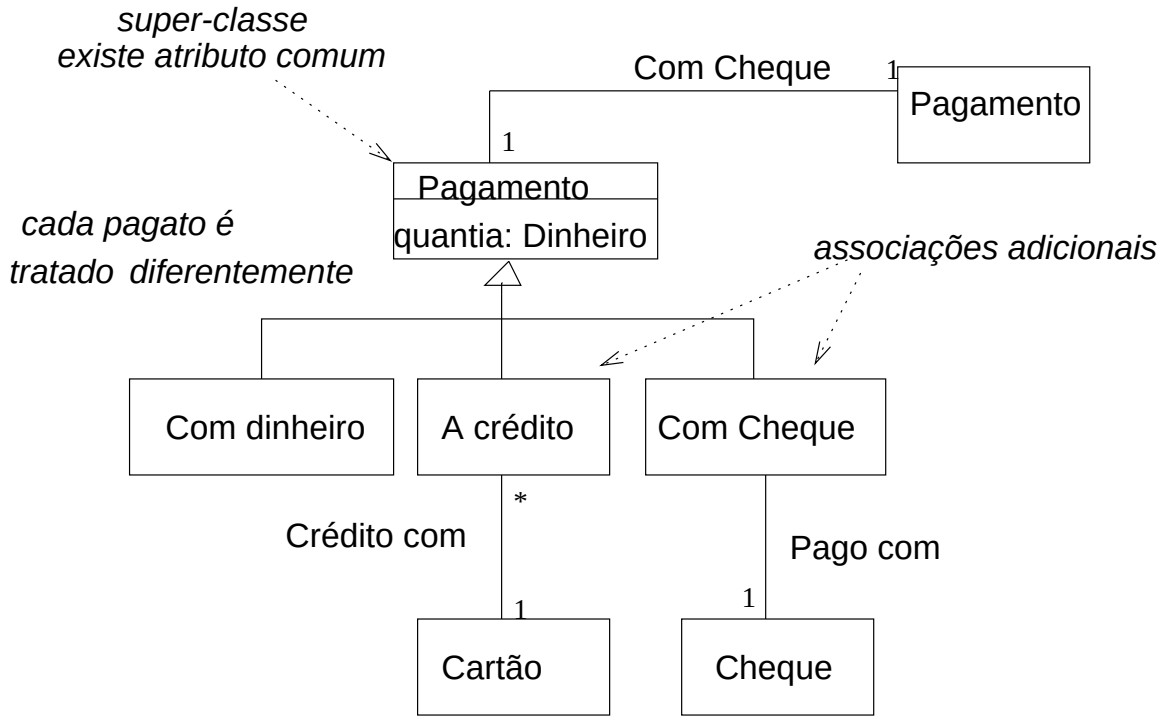
Notação para Generalização/Especialização - UML



ou



Justificativa para Sub-classes de Pagamento



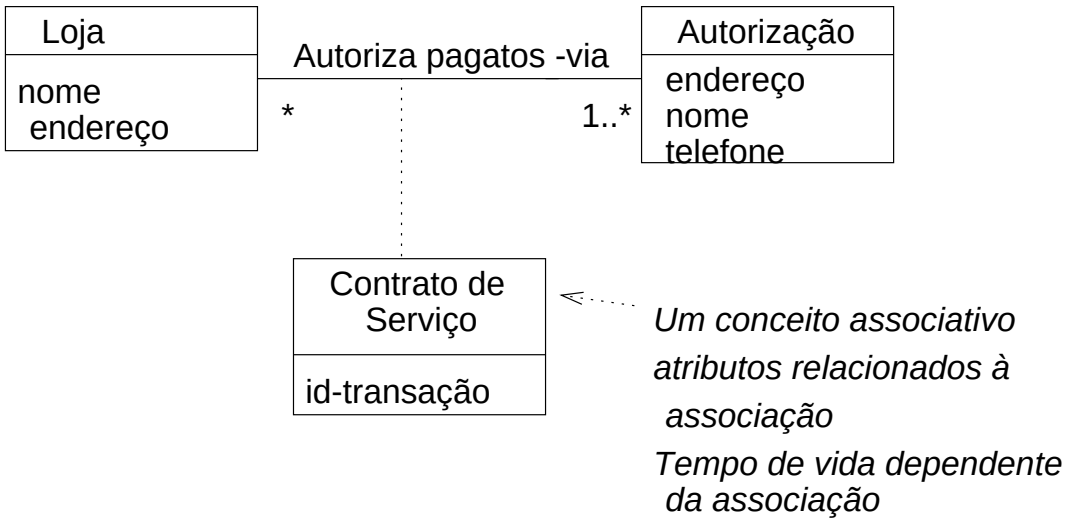
Um conceito Associação - ou Associativo

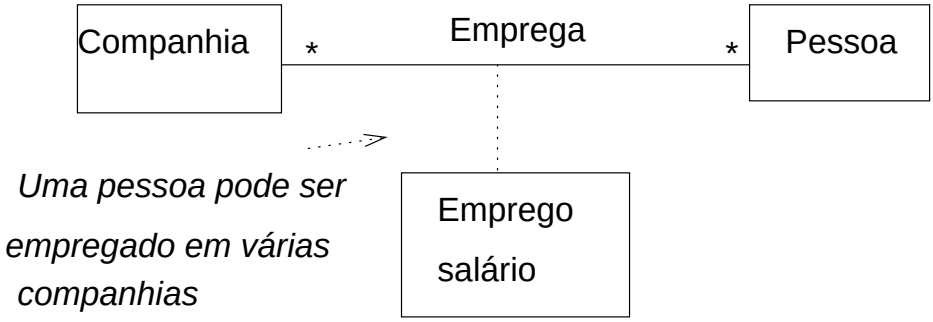
Seus atributos estão relacionados a uma associação entre outros dois conceitos, da qual seu tempo de vida é dependente.

Existe uma associação muitos para muitos entre os dois conceitos envolvidos.

Geralmente somente uma instância de um conceito associativo existe entre os dois objetos participantes da associação.

Conceito Associativo - Exemplo





Agregação

É um tipo de associação utilizada para modelar relacionamentos todo-parte

Composição: a parte pertence apenas a um único composto

Compartilhamento: a parte pode pertencer a mais de um composto

Como identificar:

- o tempo de vida da parte é limitado ao tempo de vida do composto
- existe uma composição física ou lógica
- algumas propriedades e/ou operações do composto se aplicam às partes

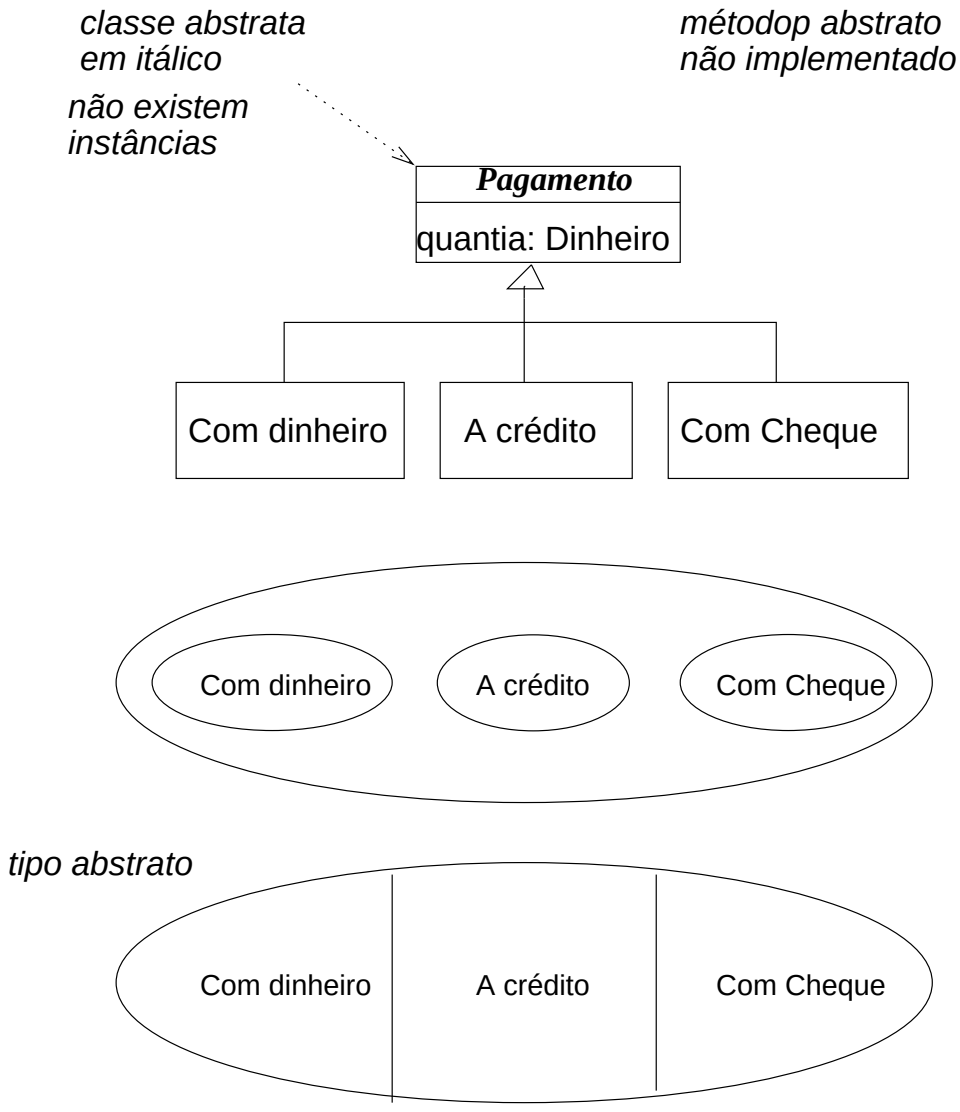
*** operações de "create", "copy", "delete" devem também ser aplicadas às parte durante a fase de projeto.

Agregação/ Composição

agregação de composição

agregação de compartilhamento

Classe Abstrata



cada membro T é tmabém membro de um sub-tipo

IV. Modelo de Arquitetura do Sistema

Divisão do Sistema em Camadas. Por exemplo, três camadas:

Apresentação: janelas, relatórios

Aplicação Lógica: registrar vendas, autorizar pagamentos

Armazenamento: Banco de Dados

Possibilita reuso; pode-se separ a camada de Aplicação Lógica em diferentes componentes a serem reusados.

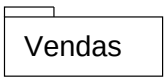
As diferentes camadas podem ser distribuídas em diferentes processos; aumenta desempenho e informação compartilhada num sistema cliente servidor.

Diferentes equipes (especialistas) desenvolvedoras em diferentes camadas

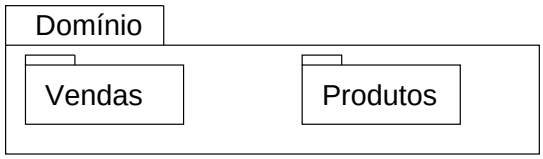
Pacotes do Sistema

Um pacote é um conjunto de elementos do modelo tais como classes, casos de usos, diagramas de colaboração e outros pacotes.

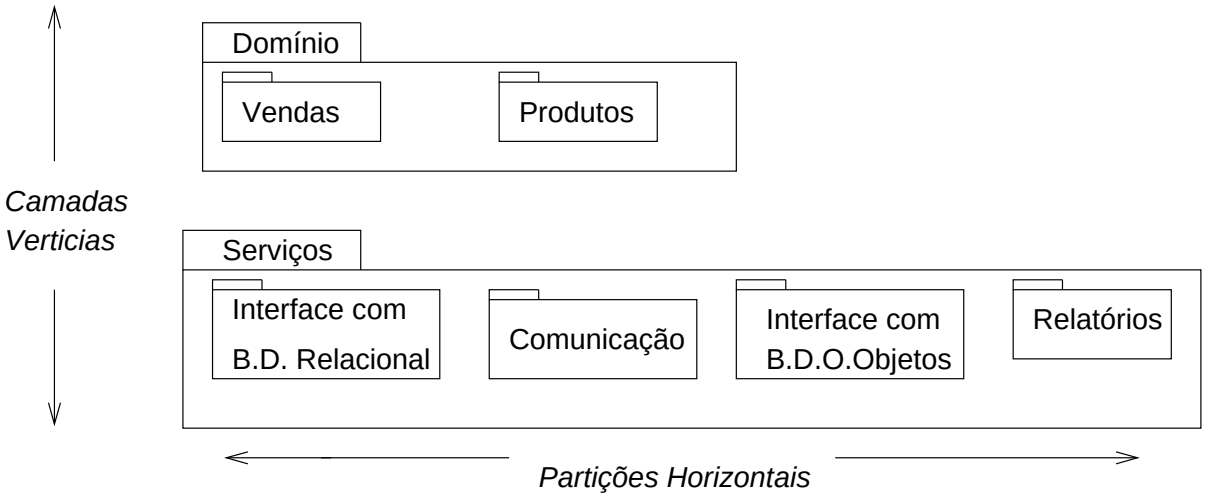
Notação para Pacotes em UML



Exemplo



Camadas e Partições



Pacotes da Camada do Domínio

Os pacotes são agrupados logicamente num diagrama

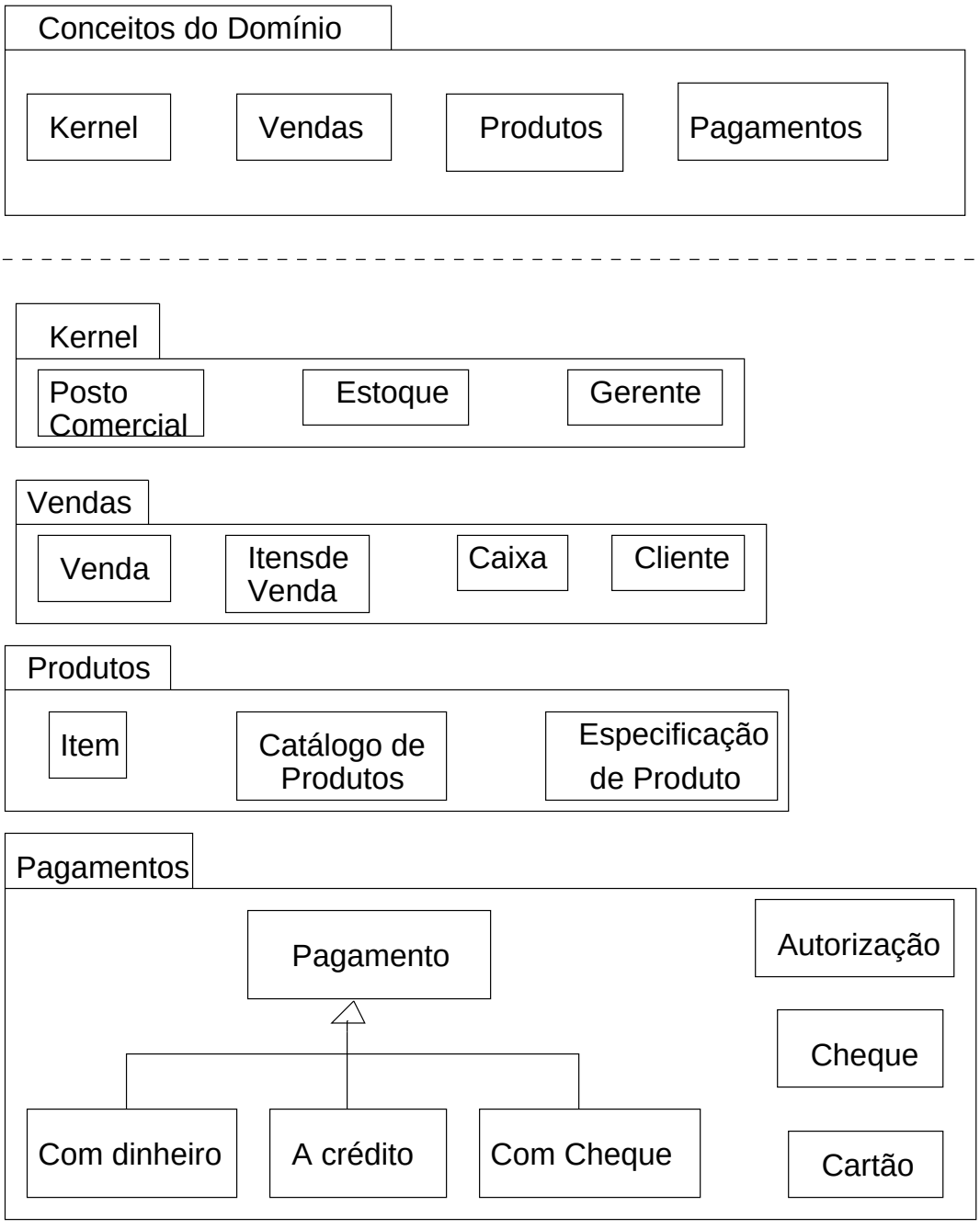
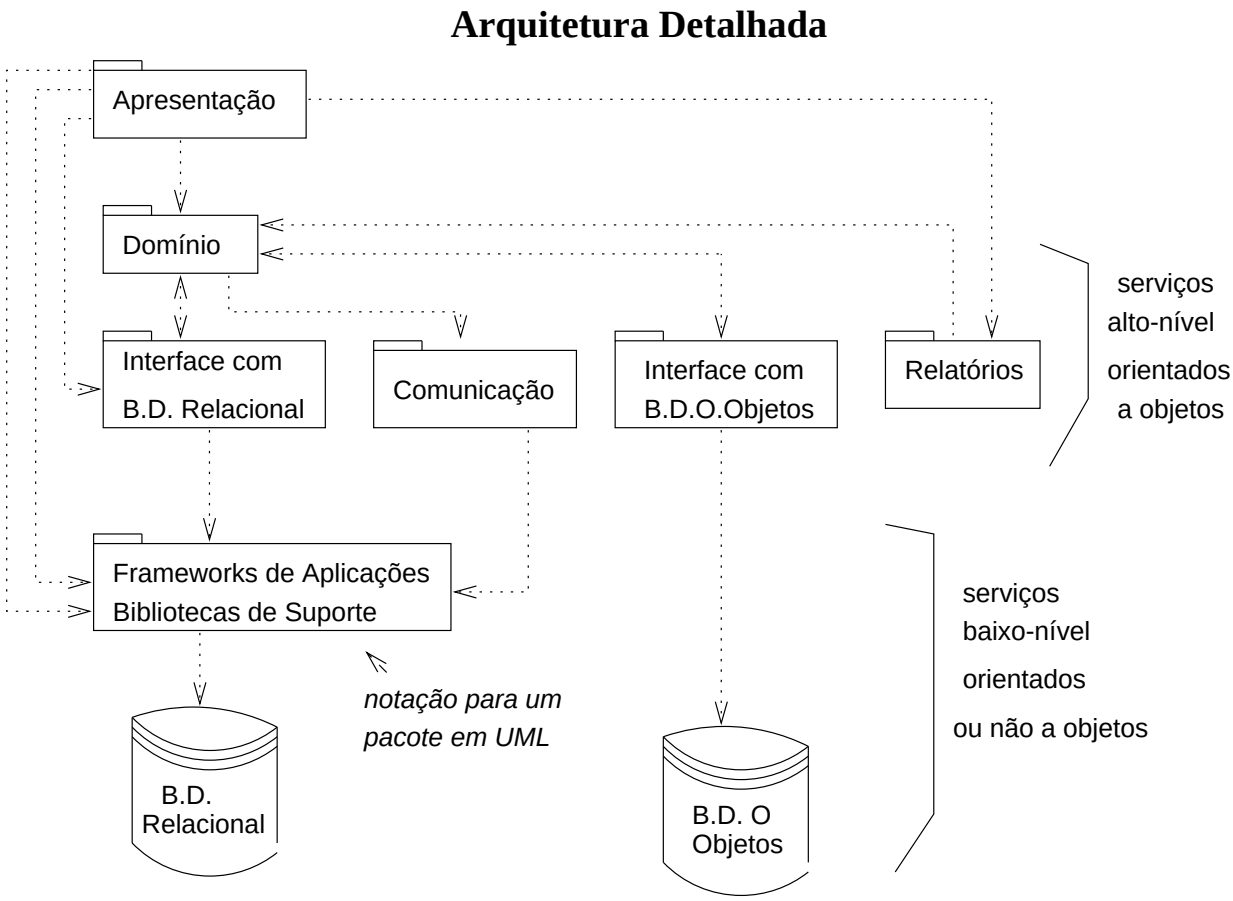


Diagrama de Pacotes de Arquitetura do Sistema

Uma arquitetura mais detalhada mostra a visibilidade entre os direntes pacotes



Padrões para Construir o Diagrama de Pacotes

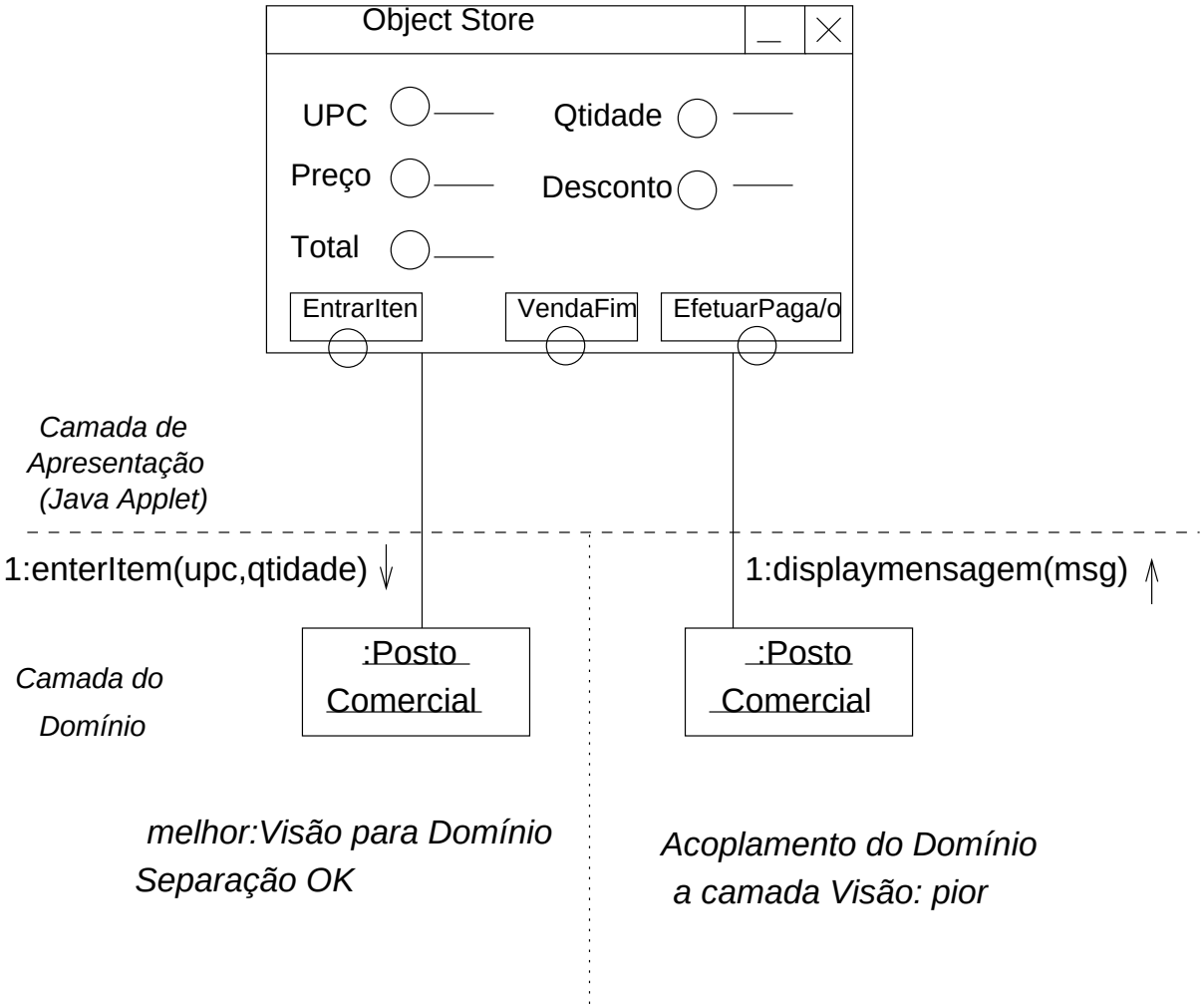
Padrão Domínio-Apresentação Separados (Modelo-Visão Separados)

Problema: Separar apresentação da camada do domínio para aumentar reuso e diminuir impacto de mudanças.

Solução: As classes da apresentação não mantêm dados, apenas realizam E/S, acoplamento mínimo de objetos do domínio com janelas da apresentação.

Permite múltiplas e novas visões para a apresentação e desenvolvimento separado das camadas, aumentando coesão.

Conectando a Camada de Apresentação a de Domínio



V. Diagramas de Estado

evento é uma ocorrência significante. Pode ser:

interno: uma msg gerada dentro dos limites do sistema

externo: gerado fora dos limites do sistema, ex: msg enviada por um ator

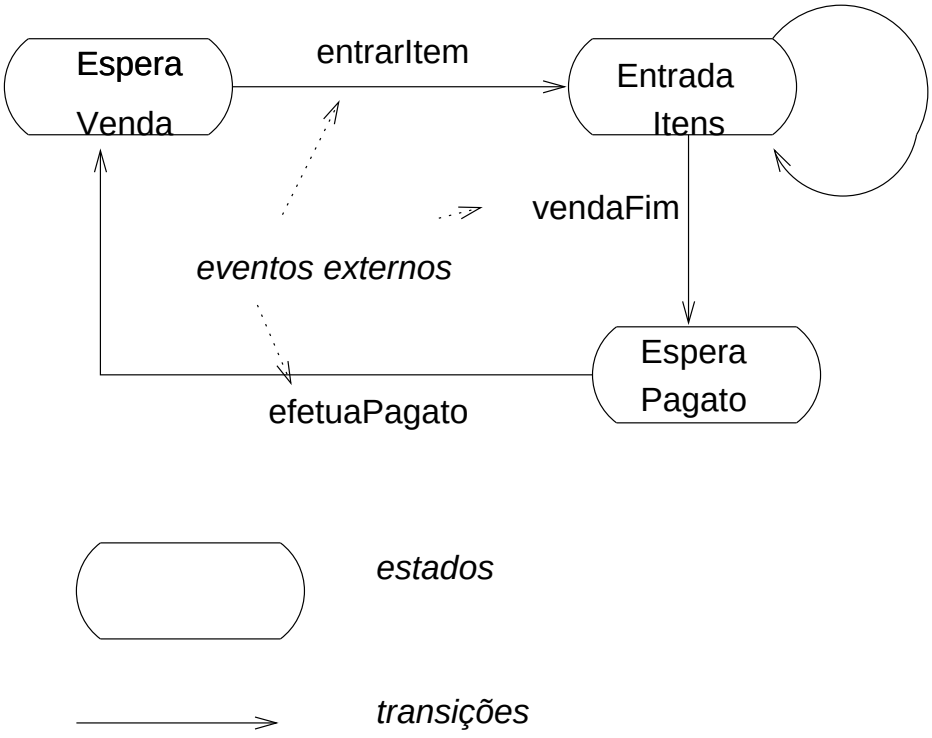
temporal: gerado com o passar do tempo, ou por hora ou data específica

estado: é a condição de um objeto num dado momento, ou tempo entre eventos

diagrama de estados: representa eventos, estados de um objeto e o comportamento de um objeto como resposta a um evento

Pode ser aplicado a casos de usos, classes, conceitos

Diagrama de Estados para Casos de Usos

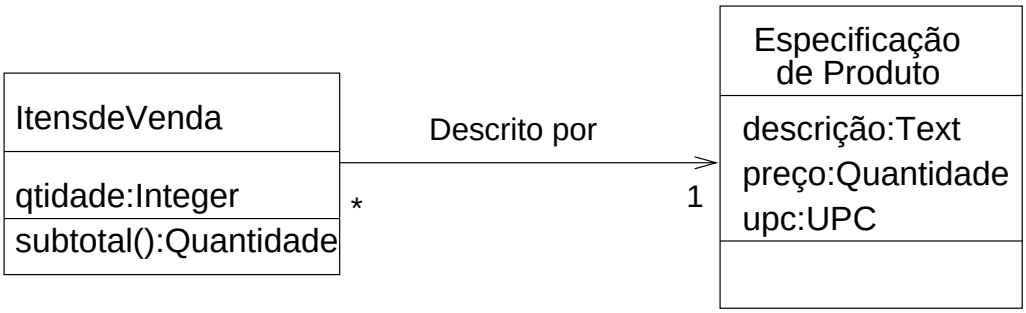


Lista de Conceitos/Classes em geral Dependentes de Estados

casos de usos, sistemas, janelas, aplicação (applets in Java, ApplicationModels in Smalltalk, etc), controladores, transações, dispositivos: modens, etc.

Do projeto para a Codificação

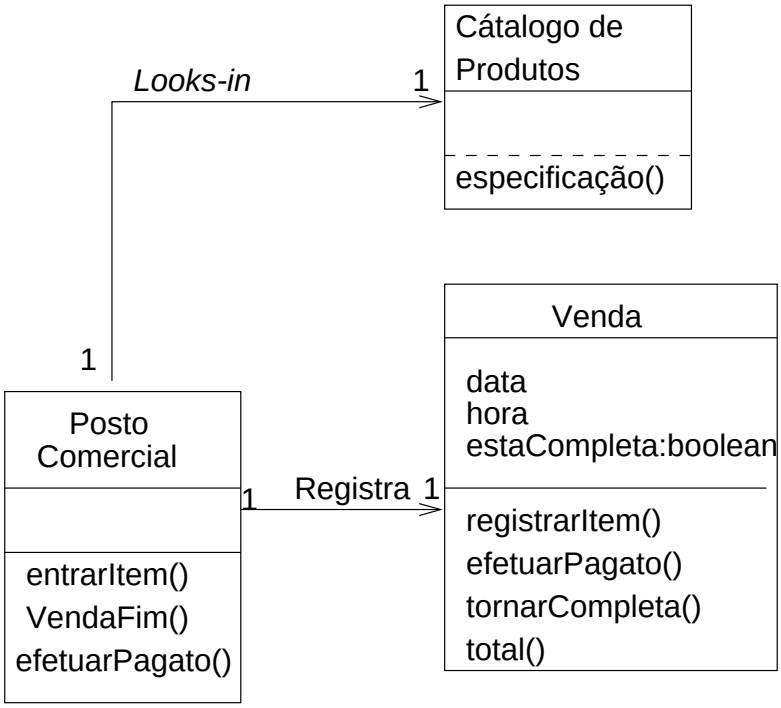
Exemplo - Atributos




```
public class ItensdeVenda
 {   ...
      public float subtotal();
      private int qtidade;    ——  atributo simples
      private EspecificacaodeProduto espec;      ——   atributo referência
  }
```

Diagramas de classes e diagramas de colaboração são utilizados para gerar código

A classe Posto Comercial e o Método entrarItens

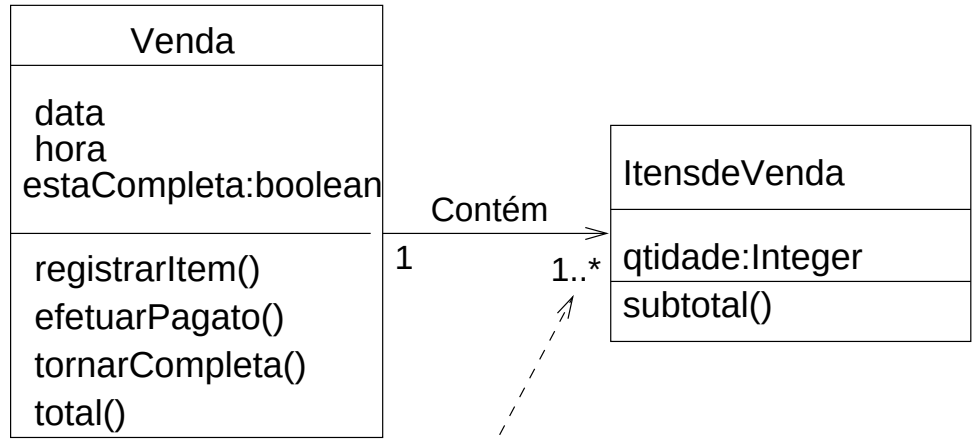



```
public class PostoComercial
{ public PostoComercial (CatalogodeProdutos cp);

    public void VendaFim();
    public void entrarItem(int upc, int qtidade);
    public efetuaPagato(float cash);

    private CatalogodeProdutos cp;
    private Venda venda;
}
```

```
public void entrarItem(int upc, int qtidade)
{
    if (isNovaVenda())
        { venda = new Venda; }
    EspecificacaodeProduto espec =
            cp.espec(upc);
    venda.registrarItem(espec,qtidade);
}
```

Código com Coleção de Classes - Visibilidade

Cuidar Navegabilidade e Visibilidade




```
public class Venda
{
   ...
    private Vector itens;      ——   visibilidade para todos os itens
}
```

Possível Ordem de Implementação para as Classes