# Projeto de Software

Alberto Sales

# As necessidades para projeto de software

- Conhecer uma linguagem orientada a objetos e ter acesso a uma biblioteca é necessário, mas não suficiente para criar um software de objetos.
- Entre uma boa idéia e um software funcional, há muito mais do que programação.
- A análise e o design fornecem “blueprints” de software, ilustrados por uma linguagem de modelagem, como a UML (Unified Modeling Language).
- As plantas servem como uma ferramenta para o pensamento e como uma forma de comunicação com os outros.

# Análise Orientada a objeto

- Uma investigação do problema (em vez de como uma solução é definida)
- Durante a análise de OO, há ênfase na localização e descrição dos objetos (ou conceitos) no domínio do problema.
  - Por exemplo, os conceitos em um Sistema de informações da biblioteca incluem Livro e Biblioteca.

# Análise Orientada a objeto

- Enfatiza uma solução conceitual que atende aos requisitos.
- Precisa definir objetos de software e como eles colaboram para atender aos requisitos.
  - Por exemplo, no Sistema de informações da biblioteca, um objeto de software Book pode ter um atributo title e um método getChapter.
- Os projetos são implementados em uma linguagem de programação.
  - No exemplo, teremos uma classe Book em Java.

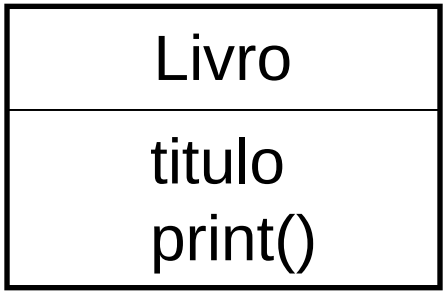
# Do Design à Implementação

Análise
Investigação do problema

Design
Solução lógica

Construção
código

Livro
(conceito)




```
public class Livro {
      public void print();
      private String titulo;
}
```

Conceito de Domínio

Representação em análise de conceitos

Representação em um Orientado a Objeto linguagem de programação.

# Desenvolvimento iterativo e o processo unificado

# O processo unificado (UP)

- Um processo é um conjunto de etapas parcialmente ordenadas destinadas a atingir uma meta.
- Na engenharia de software, o objetivo é fornecer de forma eficiente e previsível um produto de software que atenda às necessidades de seus negócios.
- Um **processo de desenvolvimento de software** é uma abordagem para criar, implantar e manter software.
- O **Processo Unificado (UP)** é um processo para a construção de sistemas orientados a objetos.
- O objetivo da **UP** é permitir a produção de software de alta qualidade que atenda às necessidades dos usuários dentro de cronogramas e orçamentos previsíveis.

# O processo unificado (UP)

- Para sistemas simples, pode ser possível definir seqüencialmente todo o problema, projetar toda a solução, criar o software e testar o produto.
- Para sistemas complexos e sofisticados, essa abordagem linear não é realista.
- O UP promove o desenvolvimento iterativo: a vida de um sistema se estende por uma série de ciclos, cada um resultando em uma liberação do produto.

# Desenvolvimento Iterativo

- O desenvolvimento é organizado em uma série de miniprojetos curtos de tamanho fixo chamados iterações.
- O resultado de cada iteração é um sistema testado, integrado e executável.
- Uma iteração representa um ciclo de desenvolvimento completo: inclui seu próprio tratamento de requisitos, análise, projeto, implementação e atividades de teste.

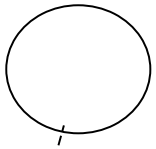
# Desenvolvimento Iterativo

- O ciclo de vida iterativo é baseado na ampliação e refinamento sucessivos de um sistema através de múltiplas iterações com feedback e adaptação.
- O sistema cresce de forma incremental ao longo do tempo, iteração por iteração.
- O sistema pode não ser elegível para implantação de produção até depois de muitas iterações.

# Desenvolvimento Iterativo

[iteração N]
Requisitos - Análise - Projeto - Implementação - Teste

[iteração N+1]
Requisitos - Análise - Projeto - Implementação - Teste



O feedback da iteração N leva a refinamento e adaptação do requisitos e design em iteração N+1.

O sistema cresce incrementalmente.

# Desenvolvimento Iterativo

- A saída de uma iteração não é um protótipo experimental, mas um subconjunto de produção do sistema final.
- Cada iteração aborda novos requisitos e amplia o sistema de forma incremental.
- Uma iteração pode ocasionalmente revisitar o software existente e melhorá-lo.

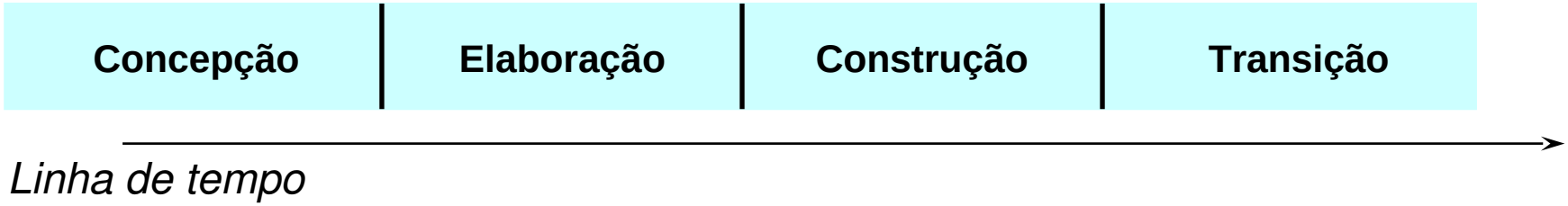
# Abrançando Mudanças

- As partes interessadas geralmente têm requisitos em mudança.
- Cada iteração envolve a escolha de um pequeno subconjunto dos requisitos e rapidamente projetá-los, implementá-los e testá-los.
- Isso leva a um feedback rápido e uma oportunidade para modificar ou adaptar a compreensão dos requisitos ou do projeto.

# Tamanho da Iteração e Timeboxing

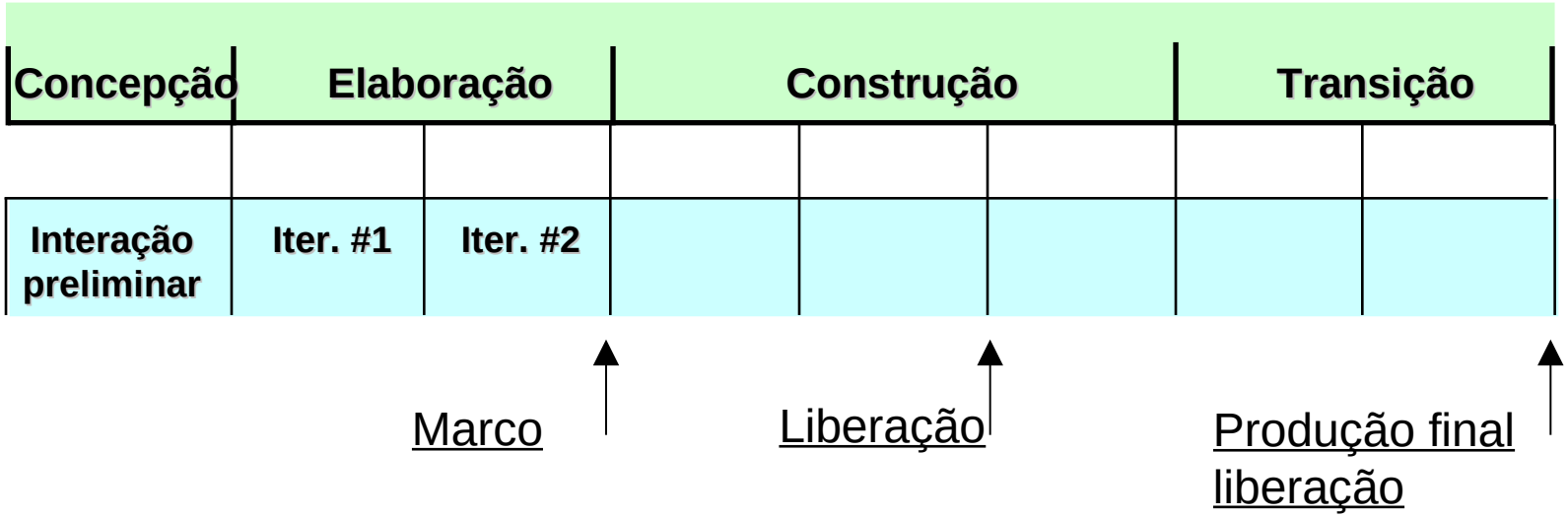
- O UP recomenda comprimentos de iteração curtos para permitir um rápido feedback e adaptação.
- Iterações longas aumentam o risco do projeto.
- As iterações têm duração fixa (timeboxed). Se o cumprimento do prazo parecer difícil, remova tarefas ou requisitos da iteração e inclua-os em uma iteração futura.
- A UP recomenda que uma iteração tenha entre duas e seis semanas de duração.

# Fases do Processo Unificado



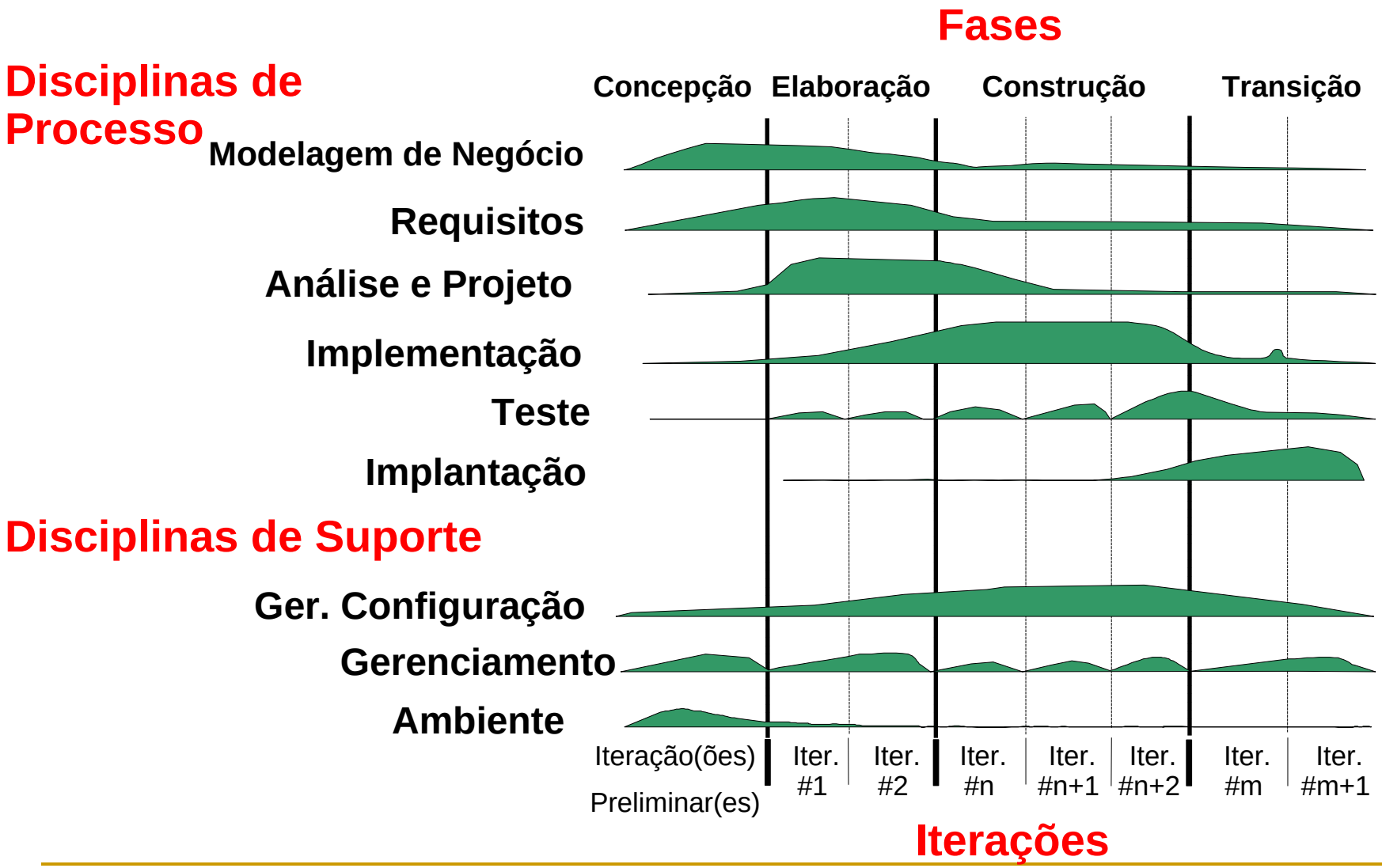

- Um projeto UP organiza o trabalho e as iterações em quatro fases principais:
  - Concepção - Definir o escopo do projeto.
  - Elaboração - Planejar projeto, especificar funcionalidades, arquitetura de linha de base.
  - Construção - Construir o produto
  - Transição - Transição do produto para a comunidade de usuários finais

# Iterações e marcos




- Cada fase e iteração tem algum foco de mitigação de risco e termina com um marco bem definido.
- A revisão de marcos fornece um ponto no tempo para avaliar quão bem os objetivos principais foram alcançados e se o projeto precisa ser reestruturado de alguma forma para prosseguir.
- O final de cada iteração é uma versão secundária, um subconjunto executável estável do produto final.

# As Disciplinas do UP



# Disciplinas e fases

- Embora uma iteração inclua trabalho na maioria das disciplinas, o esforço relativo e a ênfase mudam com o tempo.
  - As primeiras iterações tendem a aplicar maior ênfase aos requisitos e design, e as posteriores, menos.
  - As ilustrações das figuras são sugestivas, não literais.
- Observe que atividades e artefatos são opcionais (exceto código!)
  - Os desenvolvedores selecionam os artefatos que atendem às suas necessidades específicas.

# Vantagens de um processo iterativo

- Reduz riscos
  - Os riscos são identificados cedo, o progresso é mais fácil de ver.
- Obtêm-se uma arquitetura robusta
  - A arquitetura pode ser avaliada e melhorada precocemente.
- Lida com requisitos em evolução
  - Os usuários fornecem feedback aos sistemas operacionais.
  - Responder ao feedback é uma mudança incremental.
- Permite alterações
  - Sistemas podem adaptar aos problemas
- Alcance o aprendizado precoce
  - Todos obtêm uma compreensão dos diferentes fluxos de trabalho desde o início.

# Concepção

# Concepção

- Algumas questões precisam ser exploradas:
  - Qual é a visão e o business case para este projeto?
  - É viável?
  - Comprar e/ou construir?
  - Estimativa aproximada de custo.
  - Devemos prosseguir ou parar?
- A intenção é estabelecer uma visão comum inicial para os objetivos do projeto, determinar se é viável e decidir se vale a pena uma investigação séria na elaboração.

# Quais artefatos podem começar no início?

- Visão e caso de negócios
  - Descreve metas e restrições de alto nível.
- Modelo de caso de uso
  - Descreve os requisitos funcionais e os requisitos não funcionais relacionados.
- Especificação suplementar
  - Descreve outros requisitos
- Glossário
  - Terminologia do domínio chave
- Lista de riscos e plano de gerenciamento de riscos
  - Descreve riscos de negócios, técnicos, de recursos e cronograma e ideias para sua mitigação ou resposta.

# Quais artefatos podem começar no início?

- Protótipos e provas de conceito
- Plano de iteração
  - Descreve o que fazer na primeira iteração de elaboração
- Plano de Fase e Plano de Desenvolvimento de Software
  - Calcule a duração da fase de elaboração. Ferramentas, pessoas, educação e outros recursos.
- Caso de Desenvolvimento
  - Descrição das etapas e artefatos do UP customizados para este projeto.
- Os artefatos serão parcialmente concluídos nesta fase e serão refinados em iterações posteriores.

# Entendendo os Requisitos

# Introdução aos Requisitos

- Requisitos são capacidades e condições do sistema às quais o sistema deve estar em conformidade.
- Requisitos funcionais
  - Recursos e capacidades.
  - Registrado no modelo de Caso de Uso (veja a seguir) e na lista de recursos do sistema do artefato Vision.
- Não funcional (ou requisitos de qualidade)
  - Usabilidade (Ajuda, documentação, …), Confiabilidade (Frequência de falhas, recuperabilidade, …), Desempenho (Tempos de resposta, disponibilidade, …), Suportabilidade (Adaptabilidade, manutenibilidade, …)
  - Registrado no modelo de Caso de Uso ou no artefato de Especificações Suplementares.
- A natureza do UP suporta a mudança de requisitos.

# Modelo de Caso de Uso: Escrevendo Requisitos no Contexto

# Casos de uso e o valor adicionado

- Ator:
  - algo com comportamento, como uma pessoa, sistema de computador ou organização, por exemplo. um caixa.
- Cenário:
  - sequência específica de ações e interações entre os atores e o sistema em discussão, por exemplo. o cenário de compra bem-sucedida de itens com dinheiro.
- Caso de uso:
  - uma coleção de cenários de sucesso e fracasso relacionados que descrevem atores usando um sistema para apoiar uma meta.

# Casos de uso e o valor adicionado

- Lidar com devoluções
  - Principal cenário de sucesso: Um cliente chega a um caixa com itens para devolver. O caixa usa o sistema POS para registrar cada item devolvido…
  - Cenários alternativos:
    - Se a autorização de crédito for rejeitada, informe o cliente e solicite uma forma de pagamento alternativa.
    - Caso o identificador do item não seja encontrado no sistema, notifique o Caixa e sugira a entrada manual do código identificador.

# Casos de uso e o valor adicionado

- Um ponto-chave é focar na questão “como o uso do sistema pode fornecer valor observável para o usuário ou cumprir seus objetivos?”
- Os casos de uso constituem principalmente requisitos funcionais.

# Tipos de Casos de Uso e formatos

- Os casos de uso de caixa preta descrevem as responsabilidades do sistema, ou seja, definem o que o sistema deve fazer.
- Os casos de uso podem ser escritos em três tipos de formalidade
  - Resumo: resumo de um parágrafo, geralmente do principal cenário de sucesso.
  - Casual: formato de parágrafo informal (por exemplo, lidar com devoluções)
  - Completamente vestido: elaborado. Todas as etapas e variações são escritas em detalhes.

# Exemplo COMPLETO: Processar Venda

Caso de uso UC1: Processo de venda
Ator Principal: Caixa
Partes Interessadas e Interesses:
-Caixa: Quer uma entrada precisa e rápida, sem erros de pagamento, …
-Vendedor: Quer comissões de vendas atualizadas.

…
Pré-condições: O caixa está identificado e autenticado.
Garantia de sucesso (pós-condições):
-Venda é salva. Imposto calculado corretamente.

…
Principal cenário de sucesso (ou fluxo básico): [veja o próximo slide]
Extensões (ou fluxos alternativos): [veja o próximo slide]
Requisitos especiais: IU da tela de toque, …
Lista de Variações de Tecnologia e Dados:
-Identificador inserido por scanner de código de barras,…
Questões em aberto: Quais são as variações da legislação tributária? …

# Exemplo COMPLETO: Processar Venda (cont.)

- Cenário de sucesso principal (ou fluxo básico):
  - O Cliente chega a um checkout de PDV com itens para comprar.
  - O caixa registra o identificador de cada item. Se houver mais de um do mesmo item, o Caixa também pode inserir a quantidade.
  - O sistema determina o preço do item e adiciona as informações do item ao a transação de vendas em andamento. A descrição e o preço do atual itens são apresentados.
  - Ao concluir a entrada do item, o Caixa indica ao sistema POS essa entrada de item está concluída.
  - O Sistema calcula e apresenta o total de vendas.
  - O Caixa informa ao cliente o total.
  - O Cliente dá um pagamento em dinheiro (“dinheiro oferecido”) possivelmente maior do que o total da venda.
- Extensões (ou fluxos alternativos):
  - Se for inserido um identificador inválido. Indica erro.
  - Se o cliente não tiver dinheiro suficiente, cancele a transação de vendas.

# Objetivos e escopo de um caso de uso

- Em que nível e escopo os casos de uso devem ser expressos?
- R: Concentre-se em casos de uso no nível do processo de negócios elementar (EBP).
- EBP: uma tarefa executada por uma pessoa em um lugar ao mesmo tempo que adiciona valor comercial mensurável e deixa os dados em um estado consistente.
  - Aprovar ordem de crédito - OK.
  - Negociar um contrato de fornecedor - não OK.
- Geralmente é útil criar "sub" casos de uso separados representando subtarefas dentro de um caso de uso base.
  - por exemplo. Pagamento a crédito

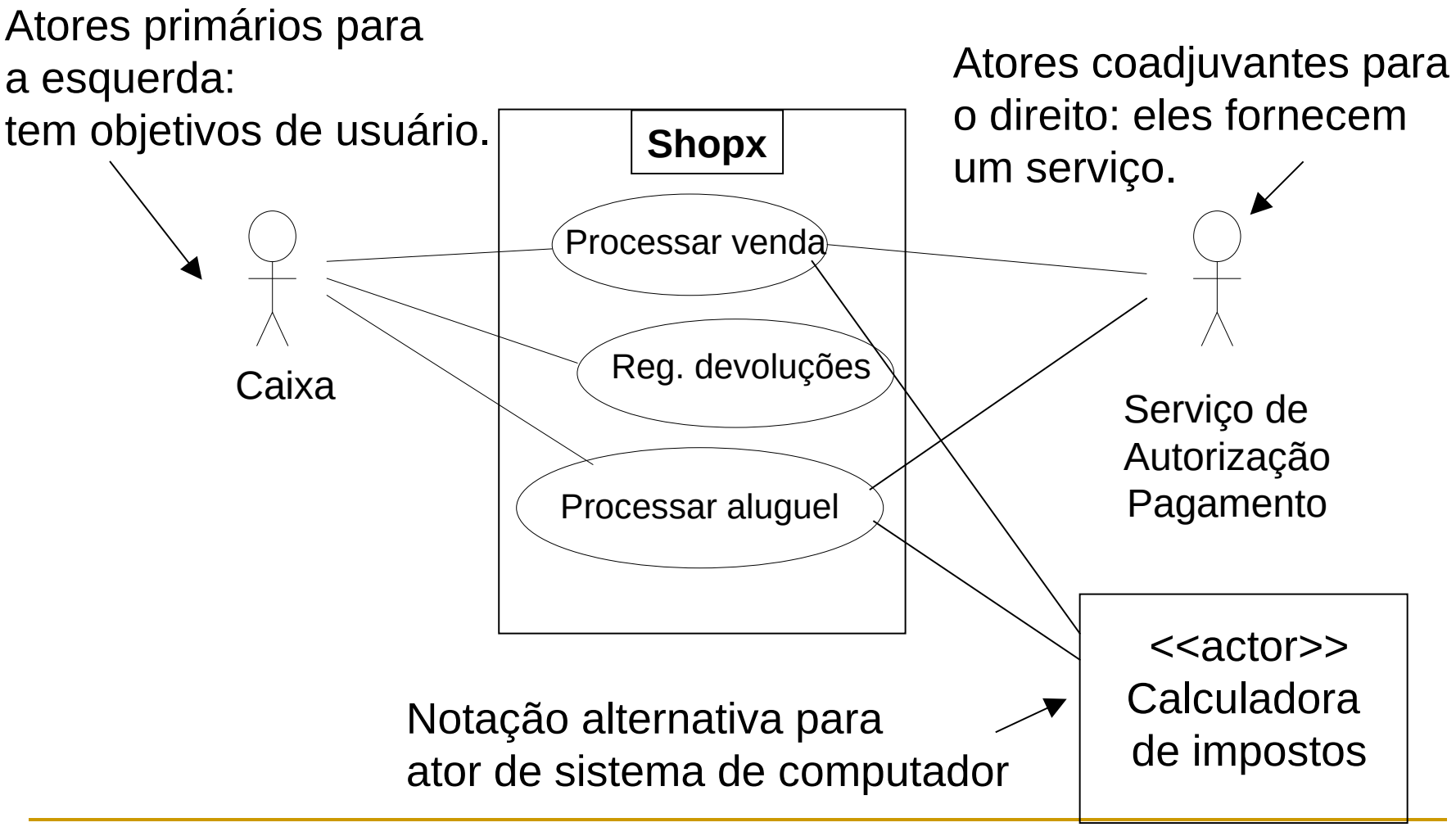
# Encontrando atores primários, objetivos e casos de uso

- Escolha o limite do sistema.
  - Identifique os atores principais.
- Aqueles que têm os objetivos do usuário cumpridos por meio do uso de serviços do sistema
- Para cada ator, identifique seus objetivos de usuário.
  - Tabule as descobertas no artefato Visão.
- Definir casos de uso que satisfaçam os objetivos do usuário; nomeá-los de acordo com seu objetivo.

# Estilo Essencial vs. Concreto

- Essencial: O foco está na intenção.
  - Evite tomar decisões de interface do usuário
- Concreto: as decisões da interface do usuário são incorporadas ao texto do caso de uso.
  - por exemplo. “Admin insere ID e senha na caixa de diálogo (veja a figura X)”
  - Estilo concreto não adequado durante o trabalho inicial de análise de requisitos.

# Diagrama de Casos de Uso



# Da Concepção à Elaboração

# Iteração1 - Requisitos

- Implemente um cenário-chave básico do caso de uso Processar venda: inserir itens e receber um pagamento em dinheiro.
- Sem colaboração com dispositivos externos (como calculadora de impostos ou banco de dados de produtos)
- Nenhuma regra de precificação complexa é aplicada.
- As iterações subsequentes crescerão nesta base.

# Desenvolvimento incremental para o mesmo caso de uso em todas as iterações

- Nem todos os requisitos no caso de uso Processar Venda estão sendo tratados na iteração 1.
- É comum trabalhar em vários cenários ou recursos do mesmo caso de uso em vários cenários e estender gradualmente o sistema para lidar com todas as funcionalidades necessárias.
- Por outro lado, casos de uso simples e curtos podem ser concluídos em uma iteração.

# Modelo de Caso de Uso: Desenhando o Diagrama de Sequência do Sistema

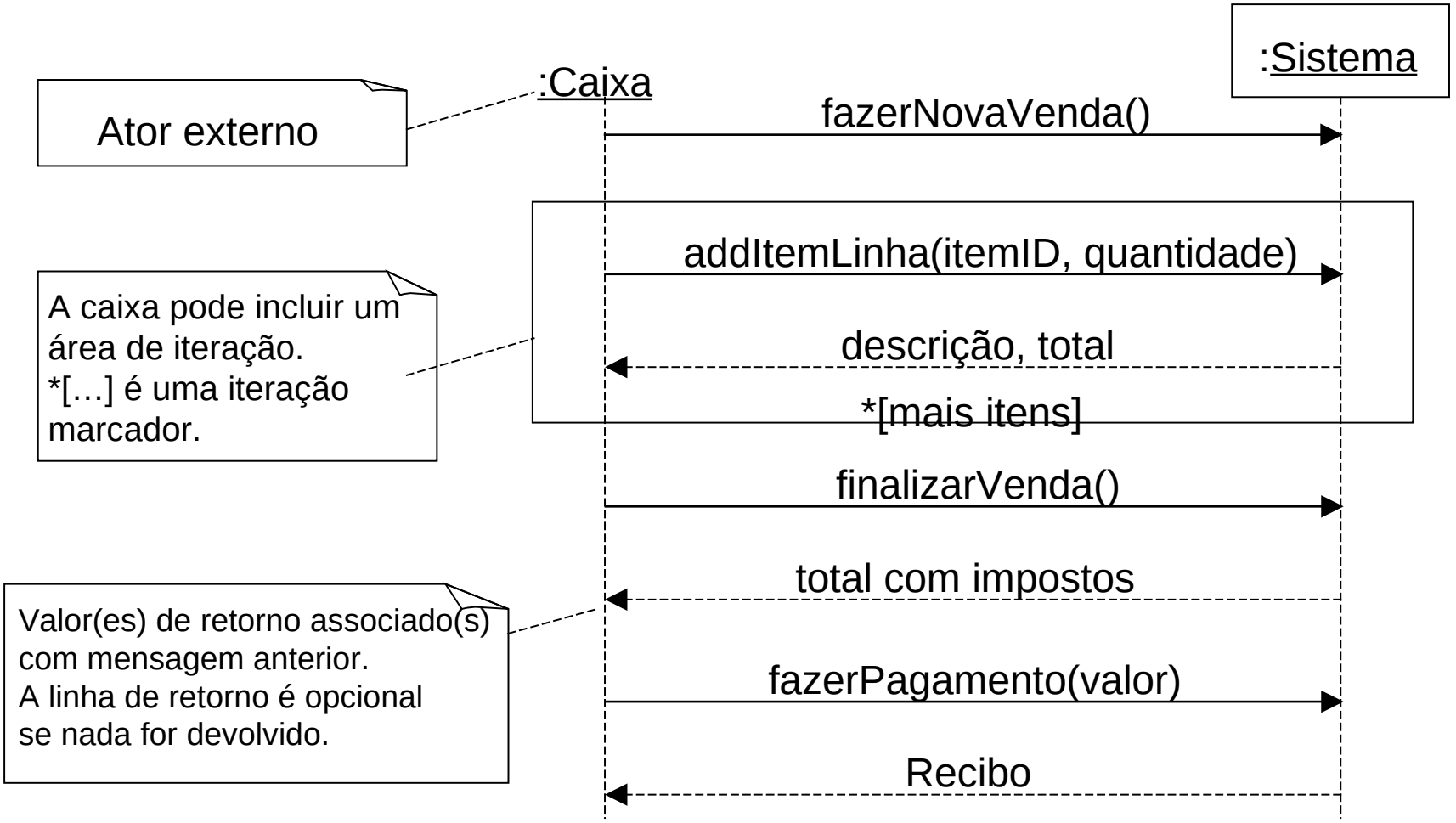
# Comportamento do Sistema e Diagramas de Sequência (UML)

- É útil investigar e definir o comportamento do software como uma “caixa preta”.
- O comportamento do sistema é uma descrição do que o sistema faz (sem uma explicação de como ele o faz).
- Os casos de uso descrevem como os atores externos interagem com o sistema de software. Durante essa interação, um ator gera eventos.
- Um evento de solicitação inicia uma operação no sistema.

# Comportamento do Sistema e Diagramas de Sequência do Sistema (SSDs)

- Um diagrama de sequência é uma imagem que mostra, para um cenário específico de um caso de uso, os eventos gerados por atores externos, sua ordem e possíveis eventos entre sistemas.
- Todos os sistemas são tratados como uma caixa preta; o diagrama enfatiza os eventos que cruzam a fronteira do sistema de atores para sistemas.

# SSDs para Cenário de Venda de Processos

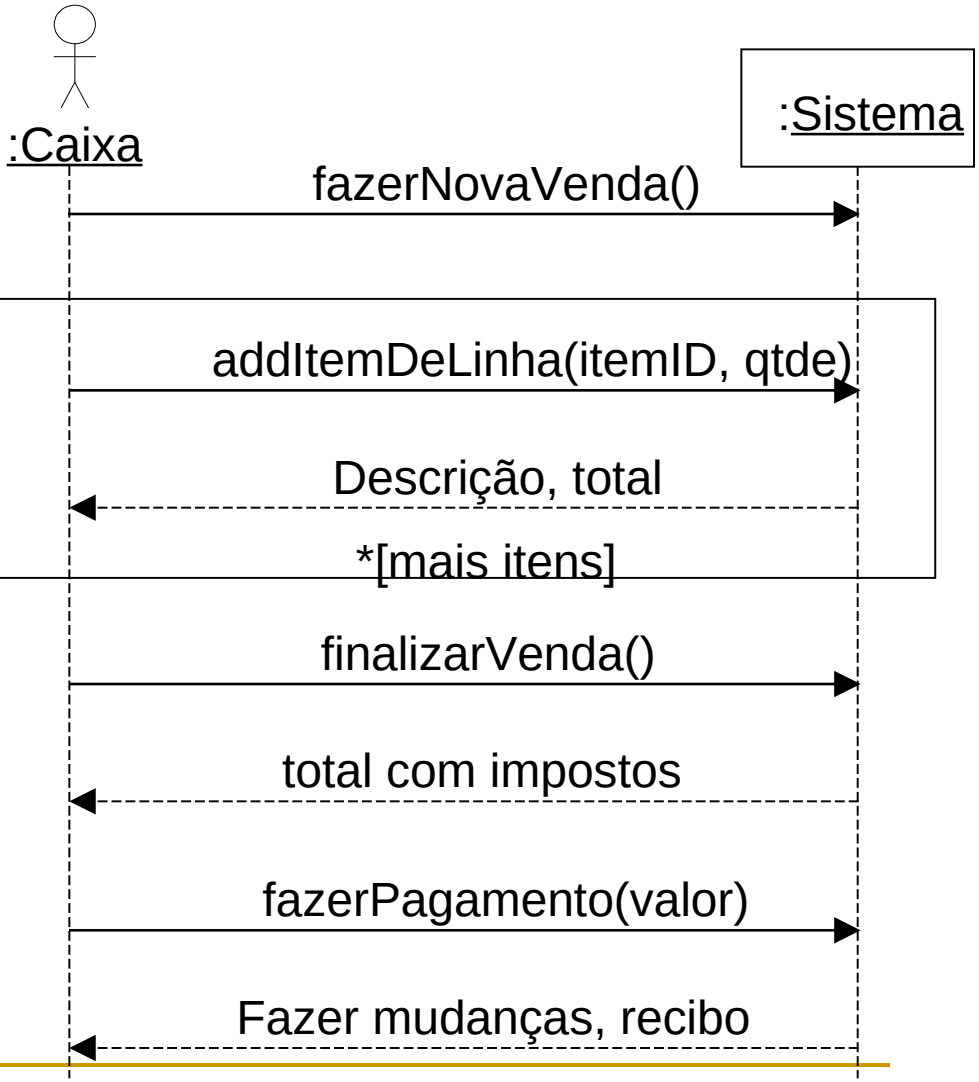


# SSD e Casos de Uso

Cenário de venda de processo simples somente em dinheiro

1. O cliente chega a um checkout de PDV com mercadorias para comprar.
2. O caixa inicia uma nova venda.
3. O caixa insere o identificador do item.
4. O sistema registra o item de linha de venda e apresenta descrição do item, preço e execução total.
   caixa repete os passos 3-4 até indica feito.
5. Sistema apresenta total com impostos calculado.

...



# Nomenclatura de eventos e operações do sistema

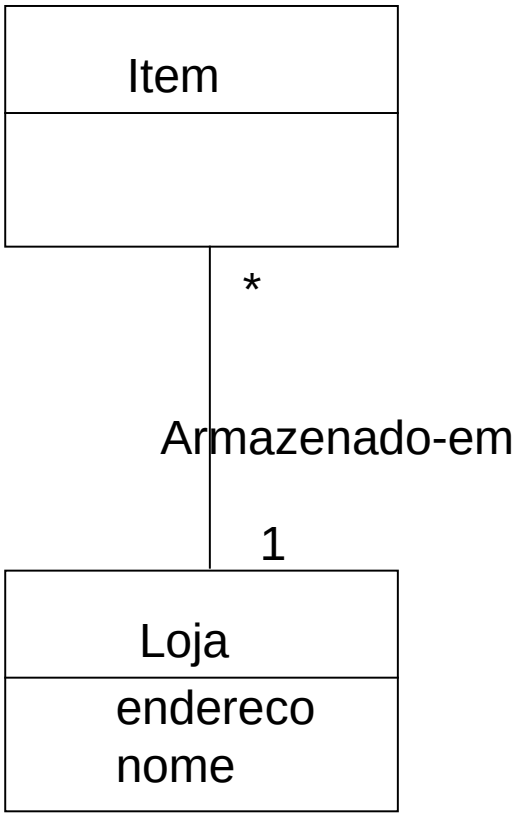
- O conjunto de todas as operações necessárias do sistema é determinado pela identificação dos eventos do sistema.
  - fazerNovaVenda()
  - addLineItem(itemID, quantidade)
  - fimVenda()
  - fazerPagamento(valor)

# Modelo de Domínio: Visualizando Conceitos

# Modelos de domínio

- Um Modelo de Domínio ilustra conceitos significativos em um domínio de problema.
- É uma representação de coisas do mundo real, não de componentes de software.
- É um conjunto de diagramas de estrutura estática; nenhuma operação é definida.
- Pode mostrar:
  - conceitos
  - associações entre conceitos
  - atributos de conceitos

# Modelo de Domínio




- Um Modelo de Domínio é uma descrição de coisas no mundo real.
- Um Modelo de Domínio não é uma descrição do design do software.
- Um conceito é uma ideia, coisa ou objeto.

# Classes conceituais no Domínioi de Vendas


Modelo Parcial do Domínio

- Uma distinção central entre análise orientada a objetos e estruturada: divisão por conceitos (objetos) em vez de divisão por funções.

# Estratégias para Identificar Classes Conceituais

- Use uma lista de categorias de classe conceitual.
- Faça uma lista de conceitos candidatos.
- Use identificação de frase nominal.
- Identifique substantivos (e sintagmas nominais) em descrições textuais do domínio do problema e considere-os como conceitos ou atributos.
- Casos de uso são excelentes descrições para desenhar para esta análise.

# Use uma lista de categorias de classe conceitual

| Categoria do Conceito | Exemplo |
|---|---|
| Objetos físicos ou tangíveis | POS |
| Especificações, projetos, ou descrições de coisas | Especificação do Produto |
| Lugares | Loja |
| Transações | Venda, Pagamento |
| Items de linhas de Transaçõess | Item de linha da vendas |
| Papéis de Pessoas | Caixa |
| Contêineres de outras coisas | Loja, Bin |

(veja lista complete em Larman 3nd. ed., pp. 134-135)

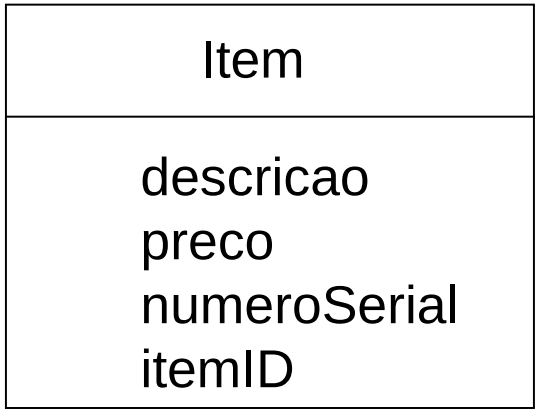
# Encontrando Classes Conceituais com identificação de frase nominal

1. Este caso de uso começa quando um Cliente chega a um checkout de PDV com itens para comprar.
2. O Caixa inicia uma nova venda.
3. O caixa insere o identificador do item.

…

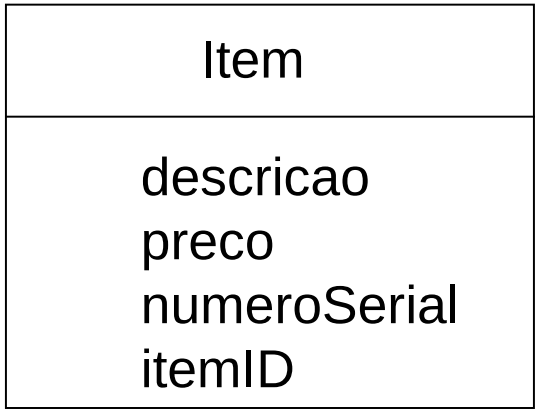
- Os Casos de Uso totalmente abordados são uma excelente descrição a ser desenhada para esta análise.
- Algumas dessas frases nominais são conceitos candidatos; alguns podem ser atributos de conceitos.
- Um mapeamento mecânico de substantivo para conceito não é possível, pois as palavras em uma linguagem natural são (às vezes) ambíguas.

# O Necessário para Especificação ou Descrição de Classes Conceituais



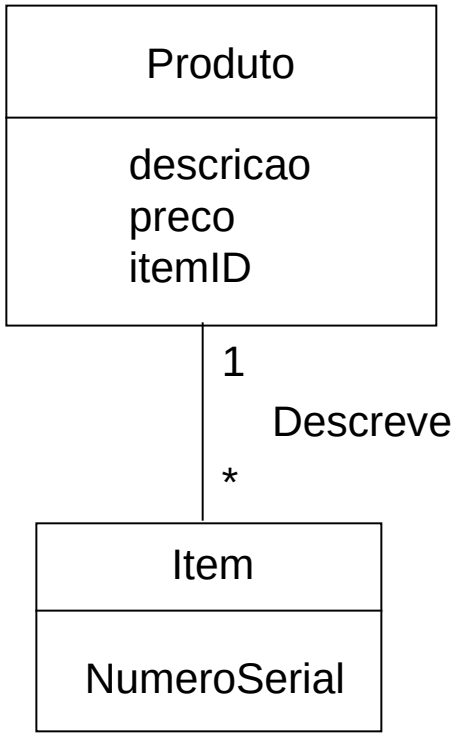

- O que há de errado com esta imagem?
- Considere o caso em que todos os itens são vendidos e, portanto, excluídos da memória do computador.
- Quanto custa um artigo?

# O Necessário para Especificação ou Descrição de Classes Conceituais



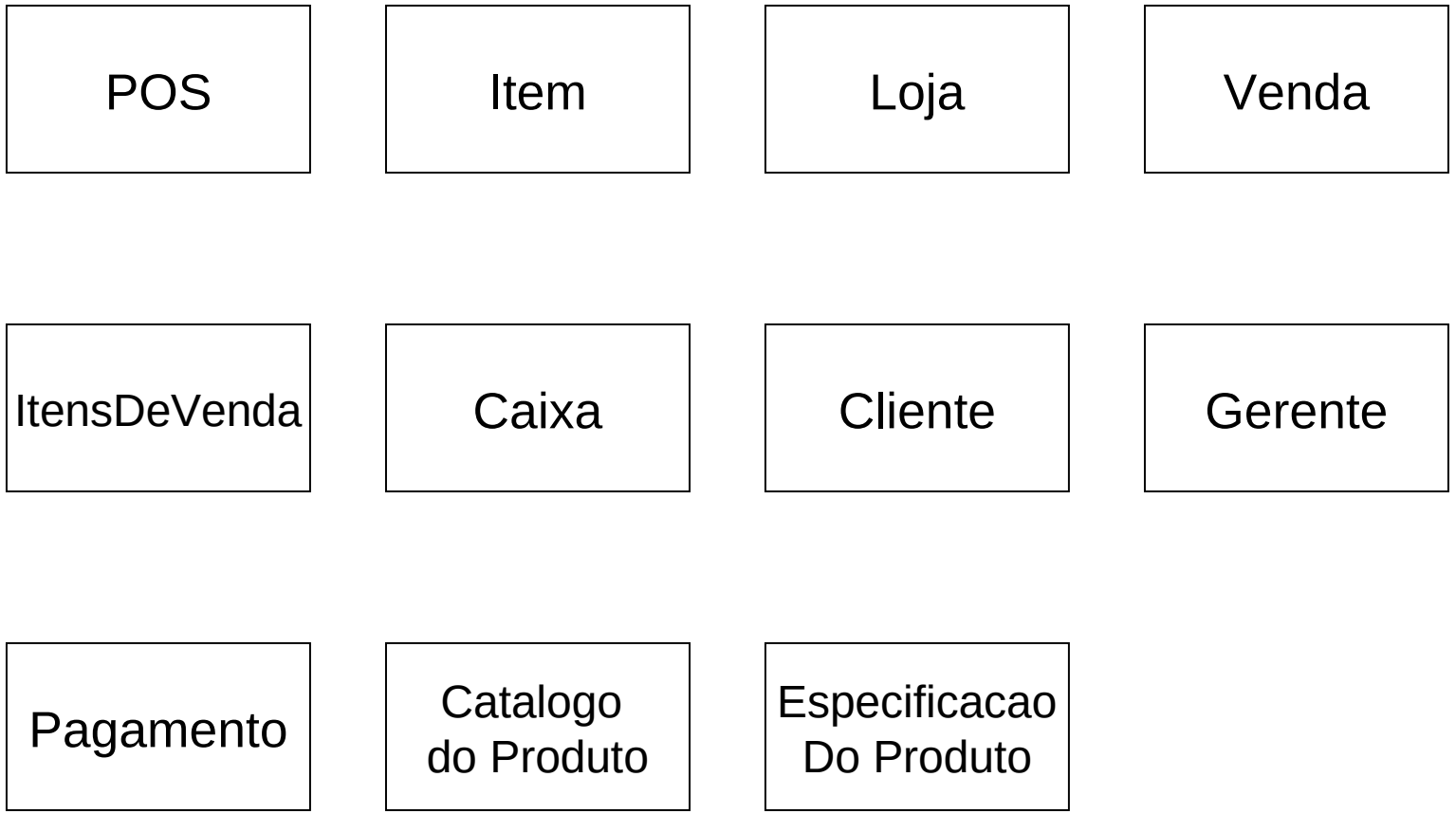

- A memória do preço do item foi anexada às instâncias inventariadas, que foram excluídas.
- Observe também que neste modelo há dados duplicados (descrição, preço, itemID).

# O Necessário para Especificação ou Descrição de Classes Conceituais



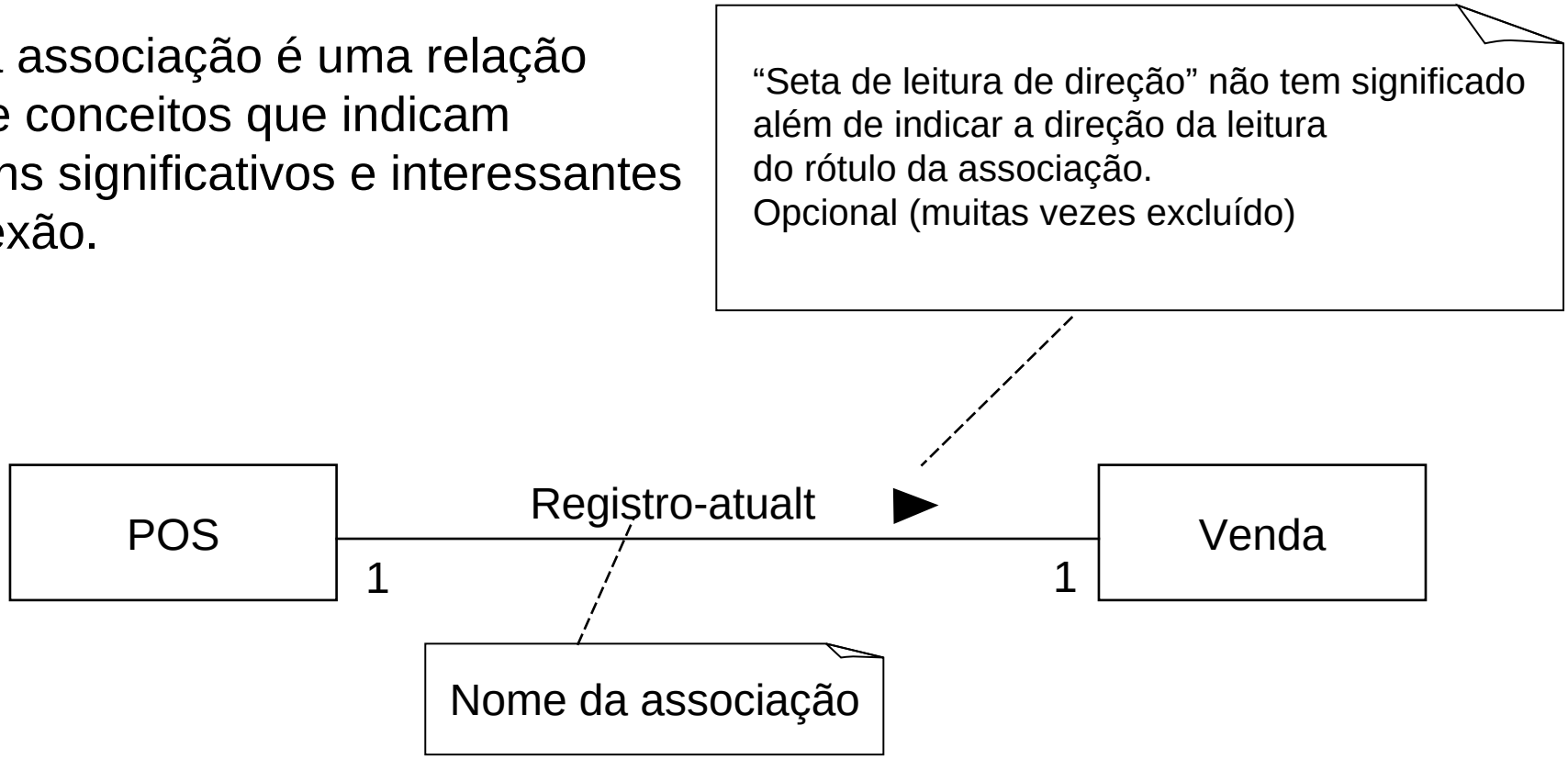

- Adicione uma especificação ou uma descrição do conceito quando:
  - A exclusão de instâncias de coisas que eles descrevem resulta em perda de informações que precisam ser mantidas, devido à associação incorreta de informações com a coisa excluída.
  - Reduz informações redundantes ou duplicadas.

# O modelo de domínio NextGen POS (parcial)



# Adicionando Associações

Uma associação é uma relação entre conceitos que indicam alguns significativos e interessantes conexão.



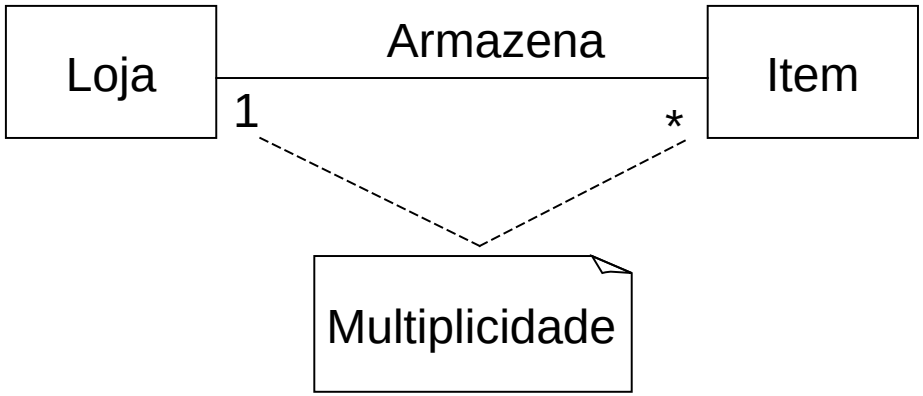
# Encontrando Associações –Lista de Associações Comuns

| **Categoria** | **Exemplos** |
|---|---|
| ***A é parte física de B**** | Gaveta - POS |
| ***A é parte lógica B*** | ItensVenda - Venda |
| ***A está fisicamente contido em/sobre B*** | POS - Store |
| ***A está logicamente contido em B*** | DescricaoItem - Catálogo |
| A é uma descrição de B | DescricaoItem - Item |
| A é um item de linha de uma transação ou informa B | ItensDeVenda - Venda |
| ***A é conhecido/registrado/gravado/ capturado em B*** | Venda - POS |
| A é um membro de B | Caixa - Loja |
| ... | |

(Veja Lista completa em 3nd. ed., pp. 156-157)

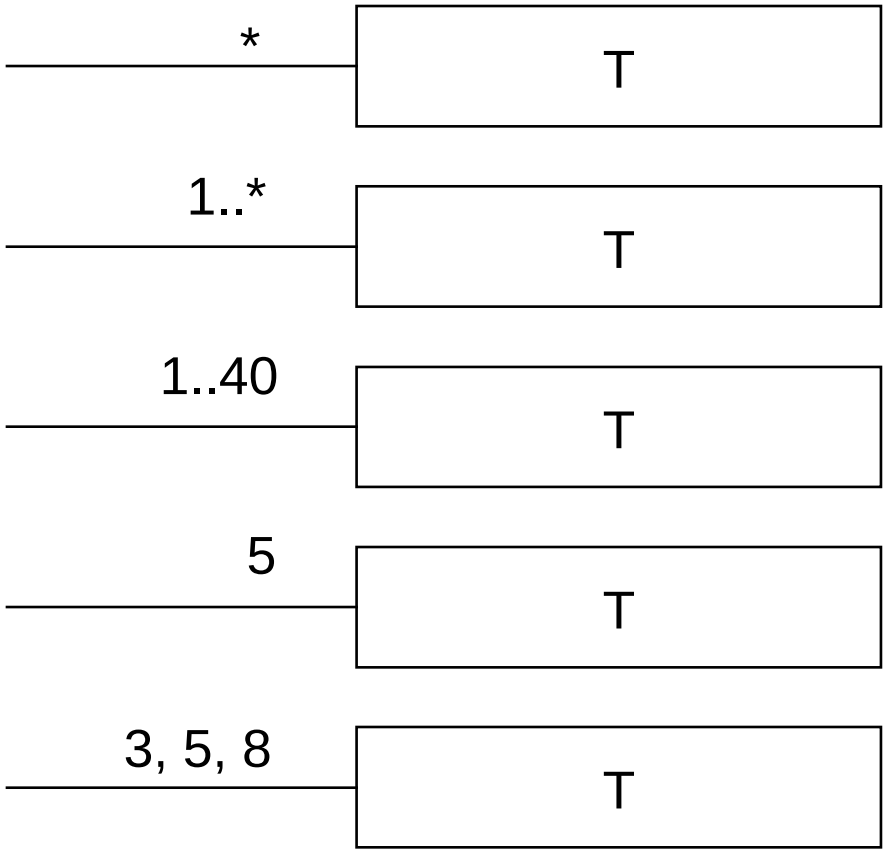
**** Associação de alta prioridade***

# Multiplicidade




- A multiplicidade define quantas instâncias de um tipo A podem ser associadas a uma instância de um tipo B, em um determinado momento.
- Por exemplo, uma única instância de uma Loja pode ser associada a “muitas” (zero ou mais) instâncias de Item.

# Multiplicidade




Zero ou mais; “muitas”

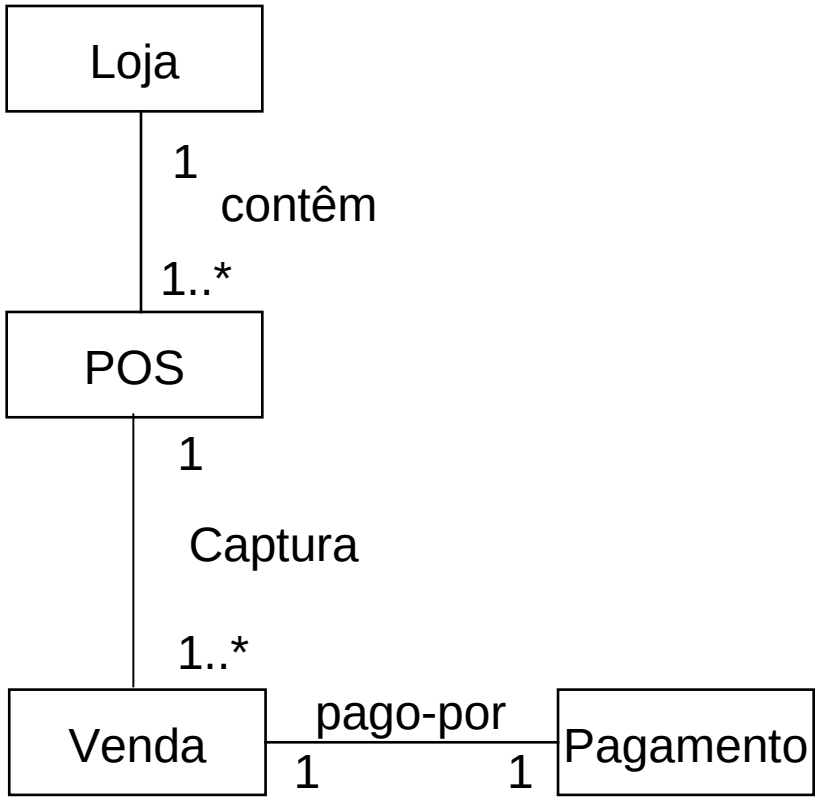
Um ou mais

Um para quarenta

Exatamente cinco

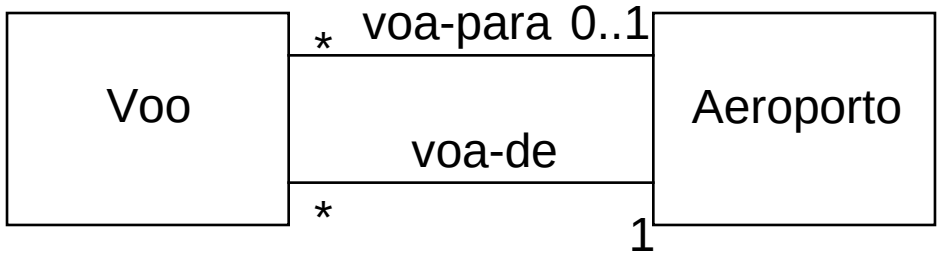
Exatamento, três, cinco ou oito.

# Nameando Associações



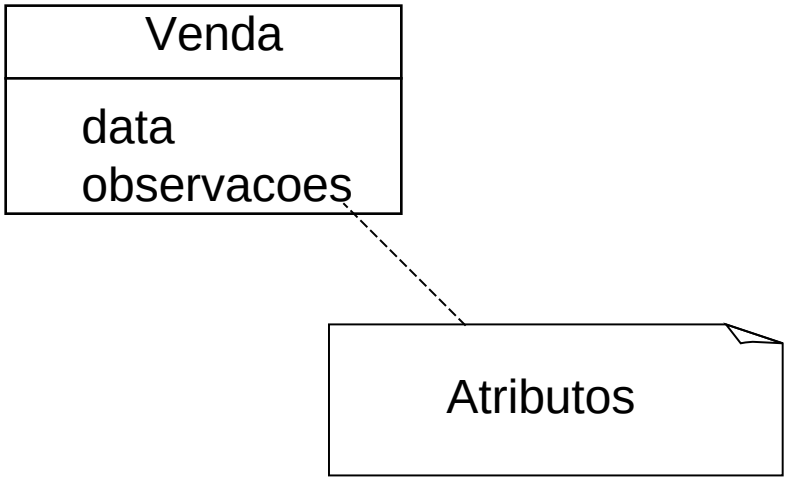

- Nomeie uma associação com base em um formato TipoNome-VerboFrase-TipoNome.
- Os nomes das associações devem começar com letra maiúscula.
- Uma frase verbal deve ser construída com hífens.
- A direção padrão para ler um nome de associação é da esquerda para a direita ou de cima para baixo.

# Múltiplas Associações Entre Dois Tipos



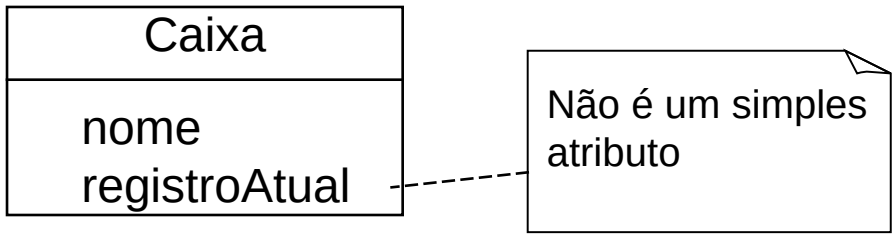

- Não é incomum ter múltiplas associações entre dois tipos.
- No exemplo, nem todo voo tem garantia de pouso em um aeroporto.

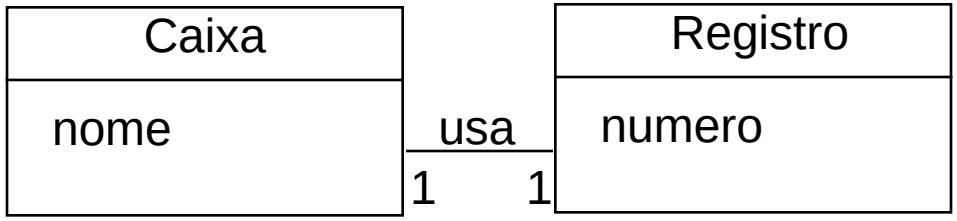
# Adicionando Atributos




- Um atributo é um valor de dados lógicos de um objeto.
- Inclua os seguintes atributos: aqueles para os quais os requisitos sugerem ou implicam a necessidade de lembrar informações.
- Por exemplo, um recibo de vendas normalmente inclui uma data e hora.
- O conceito Venda precisaria de um atributo de data e hora.

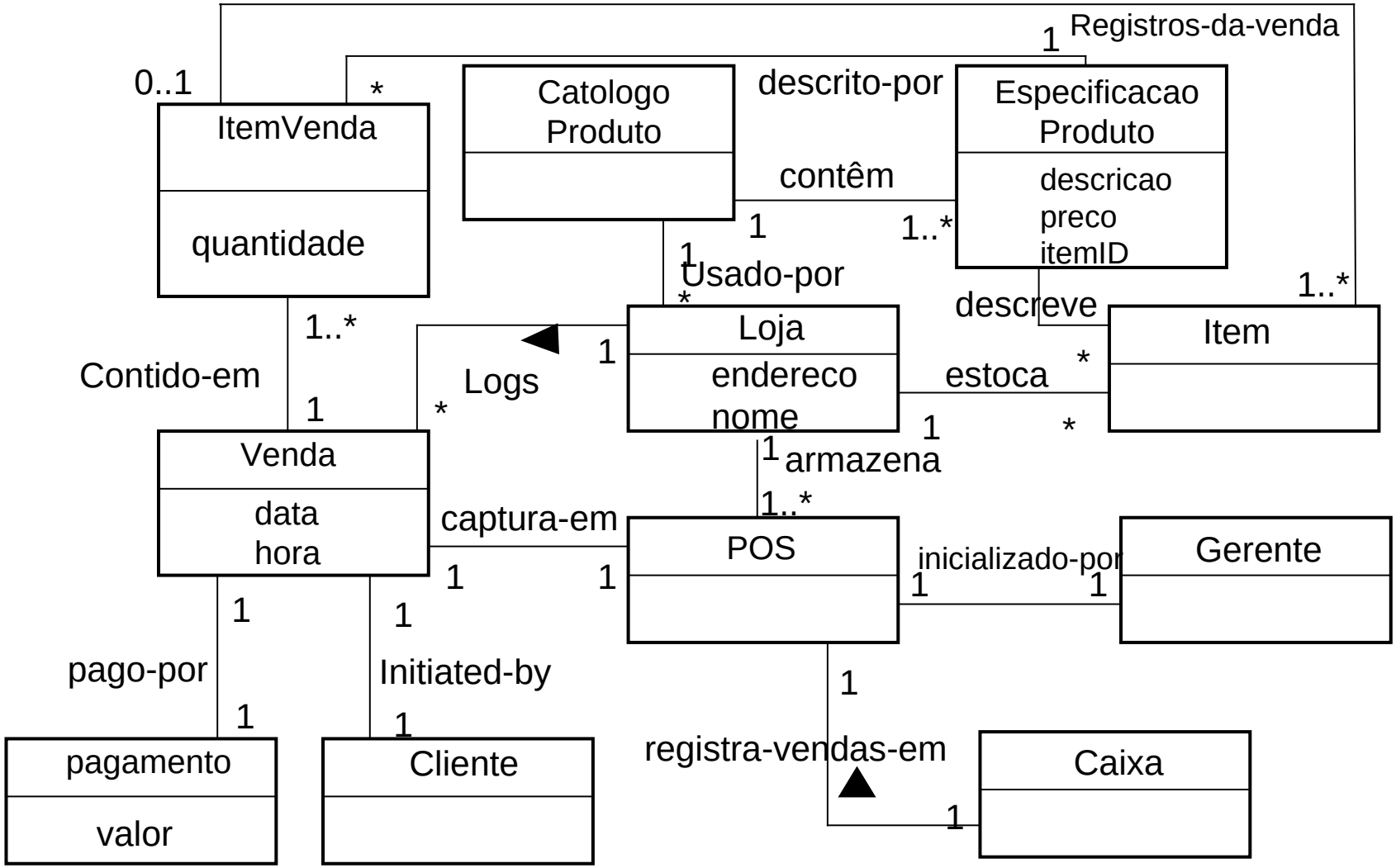
# Tipos de Atributos Válidos




- Mantenha os atributos simples.
- O tipo de um atributo normalmente não deve ser um conceito de domínio complexo, como Venda ou Aeroporto.
- Os atributos em um modelo de domínio devem ser preferencialmente
- Valores de dados puros: Boolean, Date, Number, String, …
- Atributos simples: cor, número de telefone, código postal, código universal do produto (UPC), ...

# Modelo de Domínio Completo

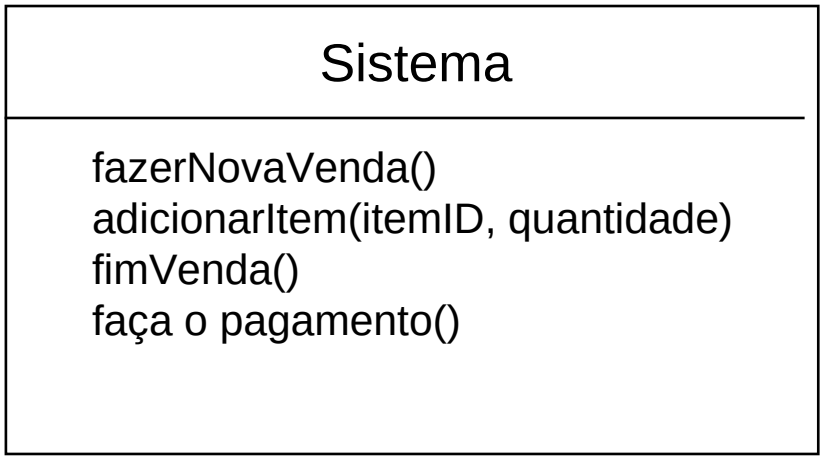


# Modelo de Caso de Uso: Adicionando Detalhes com Contratos de Operação

# Contratos

- Contratos são documentos que descrevem o comportamento do sistema.
- Os contratos podem ser definidos para operações do sistema.
- Operações que o sistema (como uma caixa preta) oferece em sua interface pública para lidar com eventos de sistema de entrada.
- Todo o conjunto de operações do sistema em todos os casos de uso define a interface do sistema público.

# Operações do sistema e a interface do sistema




- Na UML, o sistema como um todo pode ser representado como uma classe.
- Os contratos são escritos para cada operação do sistema para descrever seu comportamento.

# Exemplo de contrato: adicionarItem

**Contrato CO2: adicionarItem**

**Operação:** adicionarItem (itemID: ItemID, qutde: integer)

**Referência cruzada:** Casos de uso: Processar venda.

**Pré-condição:** Há uma venda em andamento.

**Pós-conditições:**

- Uma instância Item **item** foi criada. (criação de instância)
- **item** foi associado à venda. (associação formada)
- **item**.**qtde** foi definido como quantidade. (modificação de atributo)
- **item** foi associado a um objeto EspecificacaoProduto, com base na correspondência itemID (associação formada)

# Pre- e Pós-condições

- As pré-condições são suposições sobre o estado do sistema antes da execução da operação.
- Uma pós-condição é uma suposição que se refere ao estado do sistema após a conclusão da operação.
  - As pós-condições não são ações a serem executadas durante a operação.
  - Descrever as alterações no estado dos objetos no Modelo de Domínio (instâncias criadas, associações sendo formadas ou quebradas e atributos alterados)

# Pós-condições do adicionarItem

- Criação e exclusão de instância
- Depois que o itemID e a quantidade de um item foram inseridos pelo caixa, quais novos objetos deveriam ter sido criados?
  - Uma instância Item item foi criada.

# Pós-condições adicionarItem

- Modificação de atributo
- Depois que o itemID e a quantidade de um item foram inseridos pelo caixa, quais atributos de objetos novos ou existentes devem ser modificados?
- item.qtde foi definido como quantidade (modificação de atributo).

# Pós-condições adicionarItem

- Associações formadas e quebradas
- Depois que o itemID e a quantidade de um item foram inseridos pelo caixa, quais associações entre objetos novos ou existentes deveriam ter sido formadas ou quebradas?
  - item foi associado à venda atual (associação formada).
  - item foi associado a um EspecificacaoProduto, com base na correspondência itemID (associação formada).

# Escrever contratos leva a atualizações do modelo de domínio

- Também é comum descobrir a necessidade de registrar novos conceitos, atributos ou associações no Modelo de Domínio.

# Diretrizes para Contratos