

# Casos de Uso

# Objetivo da Aula

- Modelagem de Casos de Uso
  - Conceitos
  - Como trabalhar
  - Diagrama de Casos de Uso

# Processo Unificado

- É dirigido por casos de uso
- É centrado na arquitetura
- É iterativo e incremental
- É focado em riscos


# Fases do Processo Unificado

- Concepção
- Elaboração
- Construção
- Transição

# Fases do Processo Unificado

- Concepção
  - Análise dos requisitos
  - **Elaboração dos casos de uso**
  - Estudo de viabilidade

# Elaboração

- Requisitos já foram levantados 
- Próximo passo:
  - Organizá-los em grupos correlacionados, de forma a abordá-los nos ciclos iterativos.

# O que é um caso de uso?

- O caso de uso é um processo compreendido do **ponto de vista** do usuário
- O conjunto de casos de uso deve definir e esgotar toda **funcionalidade** possível do sistema
  - Funcionalidade: Comportamento ou ação que possa ser executada do início ao fim

## Caso de Uso

- Um caso de uso conta uma **história** sobre como um **usuário final** interage com o sistema sob um conjunto de circunstâncias específicas
- São processos de interação com o sistema que têm início e fim sem interrupções



# Caso de Uso

- **Sistematiza e organiza** os requisitos
- Levanta informações sobre como o sistema **interage** com os possíveis **usuários**
- Identifica quais **consultas** e **transformações** de informação são necessárias para que os processos sejam executados

# Caso de Uso

- Cada caso de uso será associado a um conjunto de requisitos funcionais do sistema

## Vamos lá

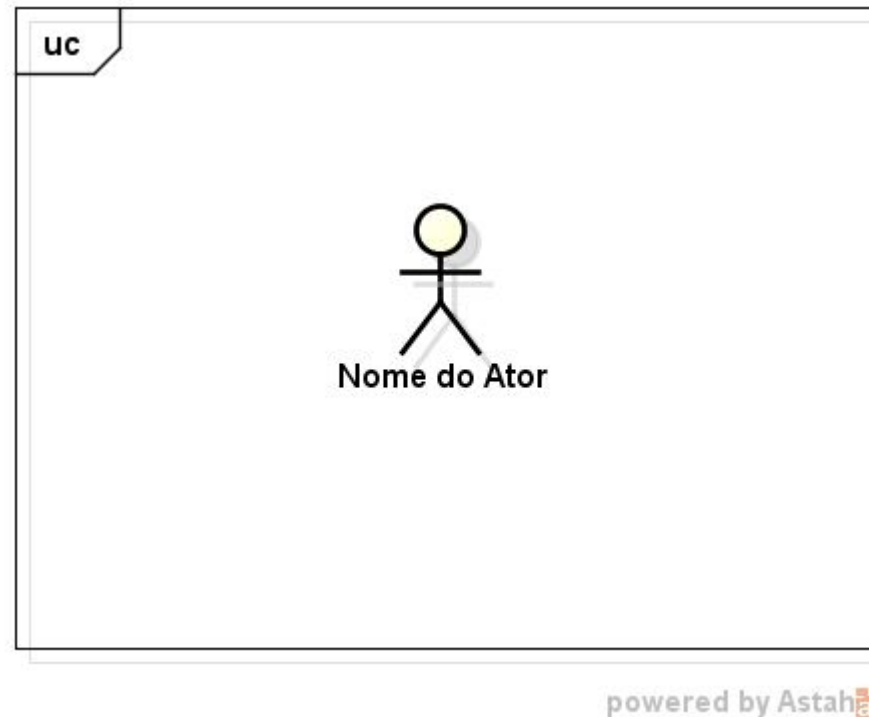
- Identificar os **atores** envolvidos com o sistema
- Identificar os principais **processos de negócio**

## O que são atores?

- **Interagem** diretamente com o sistema
- **Trocam** informações
- Não fazem **parte** do sistema
- Um ator representa um **papel**
  - conjunto de comportamentos e responsabilidades (Uma mesma pessoa pode ter N papéis)

# Na UML

- Na UML os atores são representados por bonecos



# Tipos de Atores

1. Pessoas (usuários)
2. Organização (Administradora de Cartões)
3. Outros Sistemas (Sistema de RH, Sistema Financeiro)
4. Equipamentos (Sensores, Leitores de digital ou código de barras)

## Como identificar?

- Quem irá interagir com o sistema?
- Quais outros sistemas irão se comunicar com o sistema a ser desenvolvido?
- Quem está interessado em um certo requisito funcional do sistema?

# Exemplo

## Sistema de vendas de livros

O **comprador** deve realizar um **cadastro** para ter acesso ao sistema. Com permissão de acesso ao sistema o comprador pode **comprar** quantos livros desejar. Os livros à venda devem estar disponíveis em estoque. Deverá existir uma opção de “guardar meu carrinho de compras”.

O **vendedor** deve **cadastrar** todos os livros disponíveis para venda. O controle de estoque fica por responsabilidade do sistema. O sistema avisará ao vendedor quais livros já estão esgotados. Fica por conta do vendedor **encomendar** ou não novos livros.



## Vamos lá

- Identificar os **atores** envolvidos com o sistema
- Identificar os principais **processos de negócio**

# Passos para identificar Casos de Uso

- Identificar os **objetivos** de cada **ator** do sistema
- Identificar os principais **processos** de negócio em que os atores participam

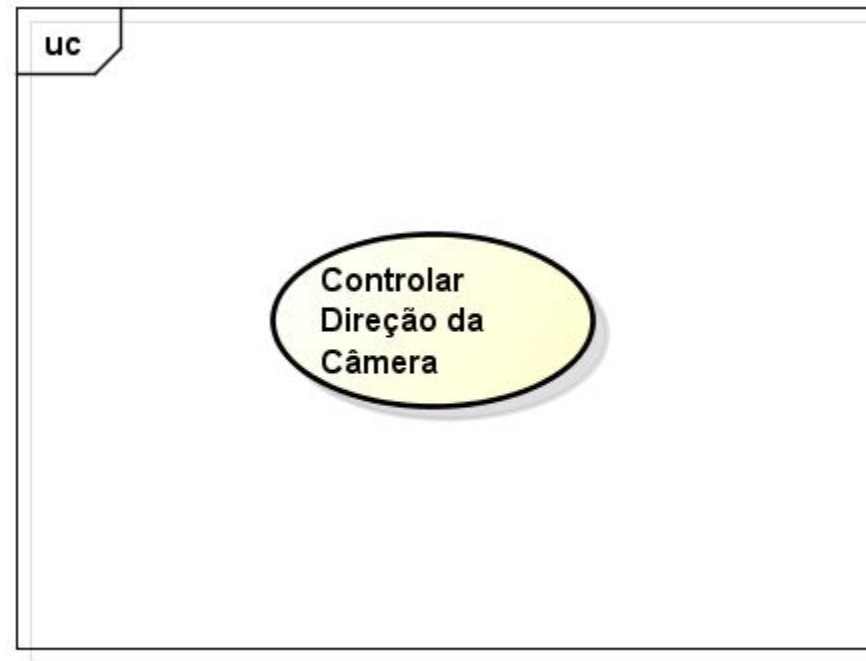
**Dos processos  
identificados surgirão  
os Casos de Uso**

# Como identificar os processos?

- Ocorrer isoladamente
- Tem um início e fim bem definidos
- Ocorre em um intervalo de tempo sem interrupções
- Produz um resultado consistente

# Na UML

Na UML os casos de uso são representados por elipses



powered by Astah

# Pergunta

- Para cada requisito funcional, existe um ou mais casos de uso para atendê-lo?
  - Se algum requisito não está associado a um Caso de Uso então algum Caso de Uso **não foi identificado**

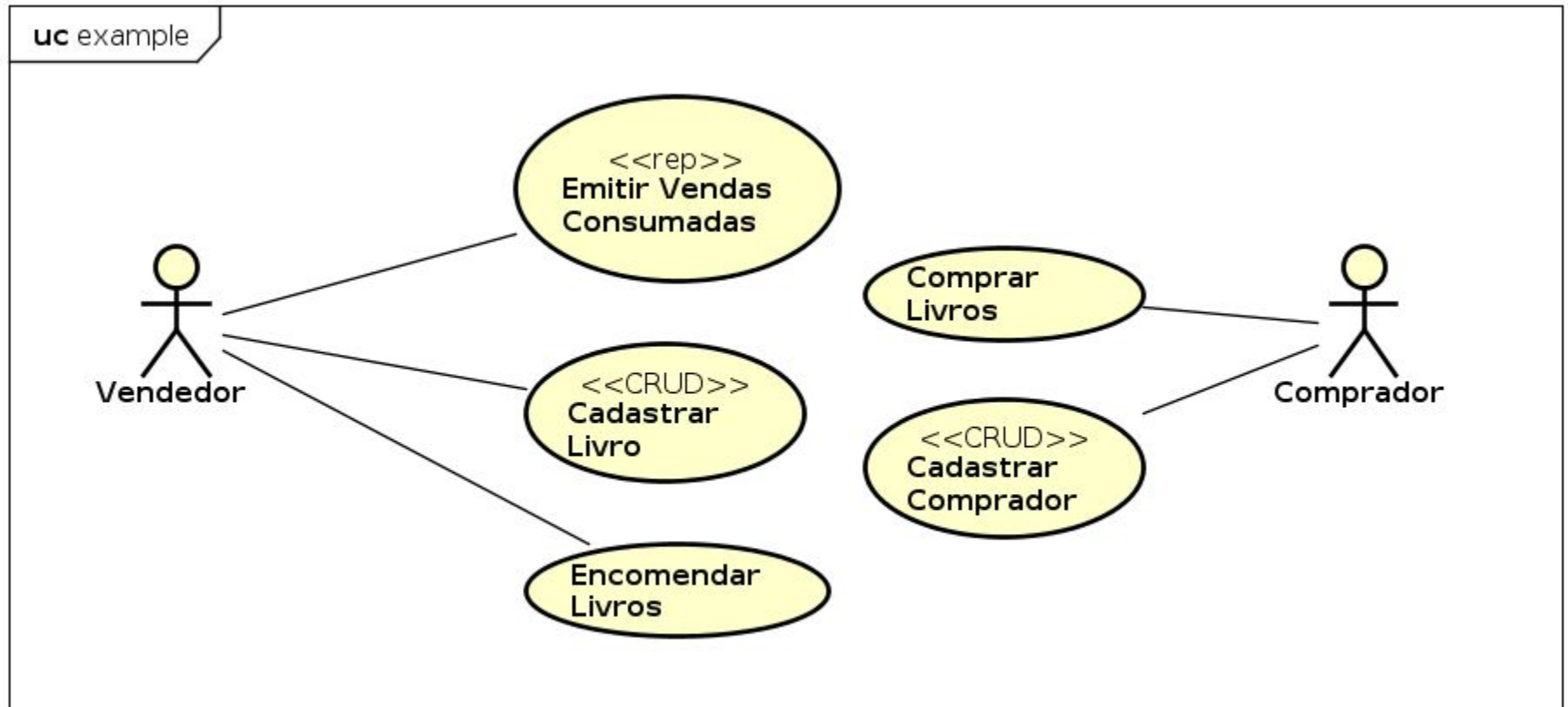
# Exemplo

## Sistema de vendas de livros

O **comprador** deve realizar um **cadastro** para ter acesso ao sistema. Com permissão de acesso ao sistema o comprador pode **comprar** quantos livros desejar. Os livros à venda devem estar disponíveis em estoque. Deverá existir uma opção de “guardar meu carrinho de compras”.

O **vendedor** deve **cadastrar** todos os livros disponíveis para venda. O controle de estoque fica por responsabilidade do sistema. O sistema avisará ao vendedor quais livros já estão esgotados. Fica por conta do vendedor **encomendar** ou não novos livros.

# Exemplo



# Tipos de casos de uso

- Categorias de Casos de Uso
  - Processos de Negócio
  - CRUD
  - Relatórios



# Processos de Negócio

Fluxos principais e estratégicos do sistema.

- Características:

- Integram múltiplas funcionalidades.
- Requerem validações e decisões de negócio.

Exemplo: Realizar venda.

- Importância: Representam processos essenciais da organização.

# CRUD (Create, Read, Update, Delete)

Operações básicas de persistência de dados.

- Características:

- Estrutura simples e repetitiva.
- Manipulam entidades específicas.

Exemplo: Cadastrar cliente, Consultar produto, Atualizar endereço, Excluir registro.

- Importância: Fundamentais para gerenciamento de dados.

# Relatórios

Definição: Operações básicas de persistência de dados.

- Características:

- Estrutura simples e repetitiva.
- Manipulam entidades específicas.

Exemplo: Cadastrar cliente, Consultar produto, Atualizar endereço, Excluir registro.

- Importância: Fundamentais para gerenciamento de dados.

<b>Categoria</b>	<b>Foco Principal</b>	<b>Complexidad e Típica</b>	<b>Relação com o Negócio</b>
<b>Processos de Negócio</b>	Fluxos estratégicos que representam atividades-chave da organização	Alta	Direta e essencial
<b>CRUD</b>	Manipulação básica de dados	Baixa a média	Suporte operacional
<b>Relatórios</b>	Geração e apresentação de informações consolidadas	Variável (baixa a alta)	Apoio à decisão

# Relacionamento em Casos de uso

- Comunicação
- Include
- Extends

# Comunicação

- Vínculo entre um ator e um caso de uso indicando que há interação entre eles.
- O ator participa do caso de uso (inicia ou é requisitado). Não implica sucesso.
- Notação UML: Linha contínua entre ator e caso de uso.

# Comunicação

- Quando usar: Sempre que o ator interage para que o caso de uso aconteça.
- Erros comuns: Ligar atores a todos os casos sem interação real; representar recursos internos como atores.

# Comunicação

- Ator Cliente — “Finalizar Compra”
- Ator Gateway de Pagamento — “Processar Pagamento”



## «include» (Inclusão)

- Reuso obrigatório; um caso de uso sempre executa o incluído.
- Toda vez que o caso base ocorre, o incluído também ocorre.
- Notação UML: Seta tracejada com seta vazada do caso base para o incluído, rotulada com «include».

## «include» (Inclusão)

- Quando usar: Subfluxo comum a vários casos de uso, como validações e cálculos.
- Benefícios: Redução de redundância, manutenção e consistência.
- Erros comuns: Usar para opcionais; fragmentar casos de uso excessivamente.

## «include» (Inclusão)

- “Finalizar Compra” —▷ «include» “Autenticar Usuário”
- “Finalizar Compra” —▷ «include» “Calcular Frete”

Ambos sempre ocorrem quando se finaliza uma compra.

## «extend» (Extensão)

- Comportamento opcional ou condicional; o extensor adiciona passos ao caso base se condição ocorrer.
- Caso base define pontos de extensão; extensor executa se condição for verdadeira.
- Notação UML: Seta tracejada com seta vazada do extensor para o caso base, rotulada com «extend».

## «extend» (Extensão)

- Quando usar: Comportamentos opcionais, promoções, variações ou exceções.
- Benefícios: Clareza ao isolar variações e exceções.
- Erros comuns: Usar para partes obrigatórias ou desvios frequentes; não documentar condições.

## «extend» (Extensão)

- “Emitir Cupom Promocional” —▷ «extend» “Finalizar Compra”

Condição: cliente elegível a promoção;

Extension point (em “Finalizar Compra”): após calcular total.

## Decisão rápida: «include» vs «extend»

- Acontece sempre? → «include»
- Só às vezes / sob condição? → «extend»
- É reuso comum a vários casos? → «include»
- É variação ou exceção? → «extend»

## Boas práticas

- Nomear casos de uso com verbo + objeto.
- Documentar condições e pontos de extensão em «extend».
- Reaproveitar com «include» apenas o que é estável.
- Evitar fragmentação excessiva.



## Referências

PRESSMAN, R. S. Software Engineering - A Practitioner's Approach. 7th ed. McGraw-Hill, 2010.

WAZLAWICK, Raul Sidnei. Engenharia de Software: conceitos e práticas. 2013.