

# FATEC - OURINHOS

**Profa. Me. Viviane de Fatima Bartholo Potenza**  
**vbartholo@gmail.com**

# **UML - Unified Modeling Language**

- Construimos modelos para compreender melhor o sistema que estamos desenvolvendo.
- Um **modelo** é uma simplificação da realidade.

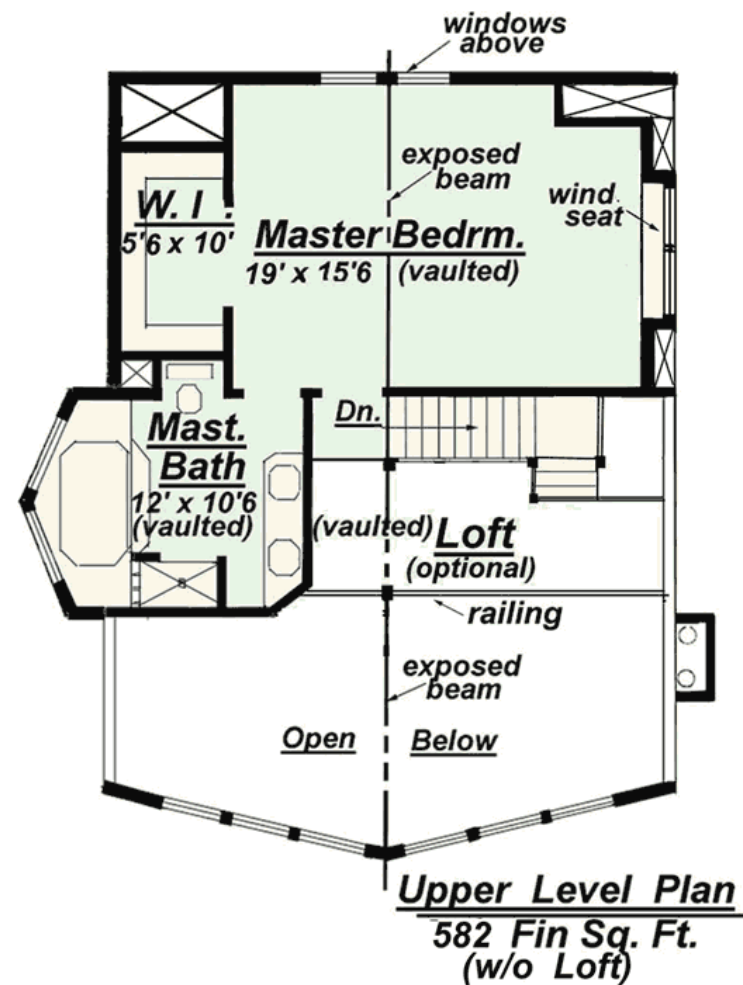
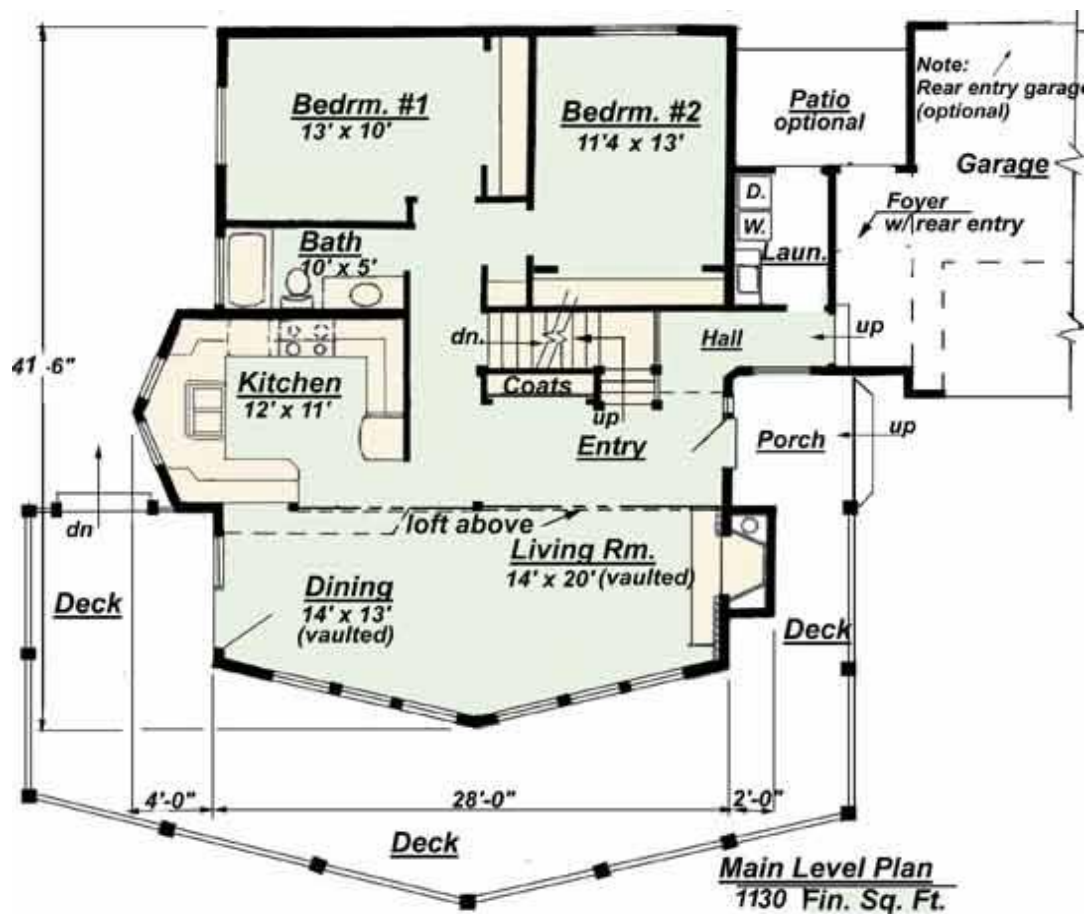
# O que é um modelo?

Fatec  
Ourinhos/SP



# O que é um modelo?

Fatec  
Ourinhos/SP



- Ajuda a ter uma visão geral do sistema
- Permite especificar a estrutura e o comportamento do sistema
- Proporciona um guia para a construção do sistema
- Documenta as decisões tomadas

- **Unified Modeling Language (UML)** é...
  - ... uma linguagem gráfica para **visualizar, especificar, construir e documentar** os artefatos de um sistema de software.
  - ... as primeiras versões do UML foram criados por **Três Amigos** - **Grady Booch** (criador do método de Booch), **Ivar Jacobson** (Object-Oriented Software Engineering, OOSE), e **Jim Rumbaugh** (Object Modeling Technique-, OMT).
  - ... resultado da unificação das notações utilizadas nos métodos **Booch**, **OMT** (Object Modeling Technique) e **OOSE** (Object-Oriented Software Engineering).
  - ... **UML ® especificação** (standard) é atualizado e gerenciado pelo **Object Management Group** (OMG <sup>TM</sup>) [OMG UML](http://www.omg.org) .

- **Unified Modeling Language (UML)** é...
  - ... adotada por grande parte da indústria de software e por fornecedores de ferramentas CASE como linguagem padrão de modelagem.
  - ... utilizada com qualquer processo de desenvolvimento já que é independente dele.



- Uma linguagem fornece um **vocabulário** e as **regras** para a combinação de "palavras" desse vocabulário, com o objetivo de comunicar algo.
- Uma **linguagem de modelagem** é uma linguagem cujo vocabulário e regras têm seu foco voltado para a representação conceitual e física de um sistema.
- O vocabulário e as regras de uma linguagem de modelagem indicam como **criar** e **ler** modelos bem formados, mas não apontam quais modelos devem ser criados e nem em que seqüência.
- Facilita a **comunicação** entre membros da equipe de desenvolvimento.

**...visualização,  
especificação,  
construção e  
documentação.**



**UNIFIED MODELING LANGUAGE™**



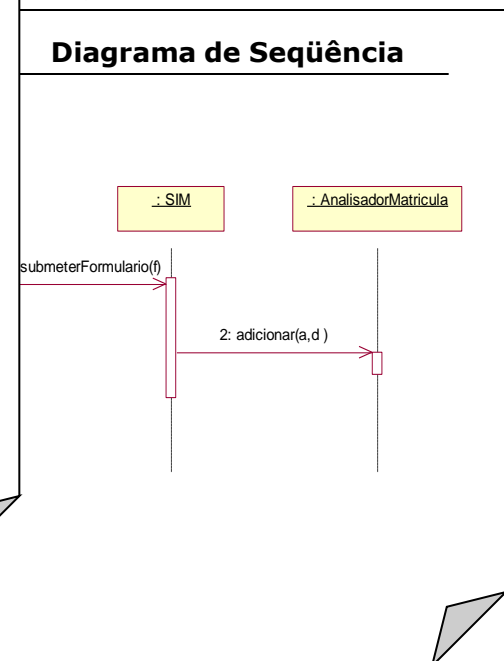
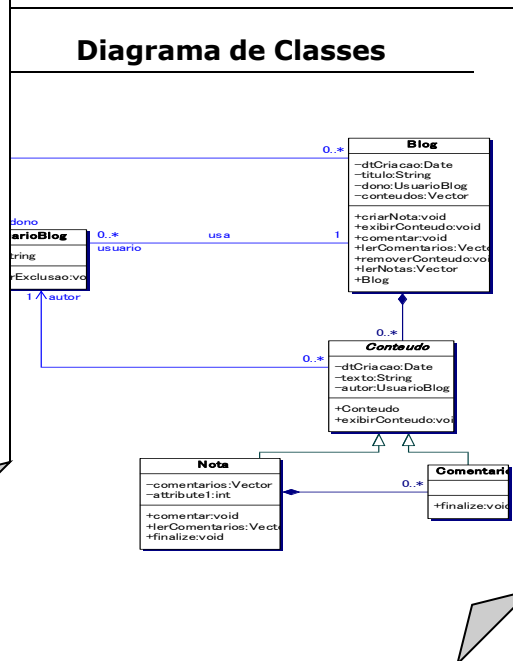
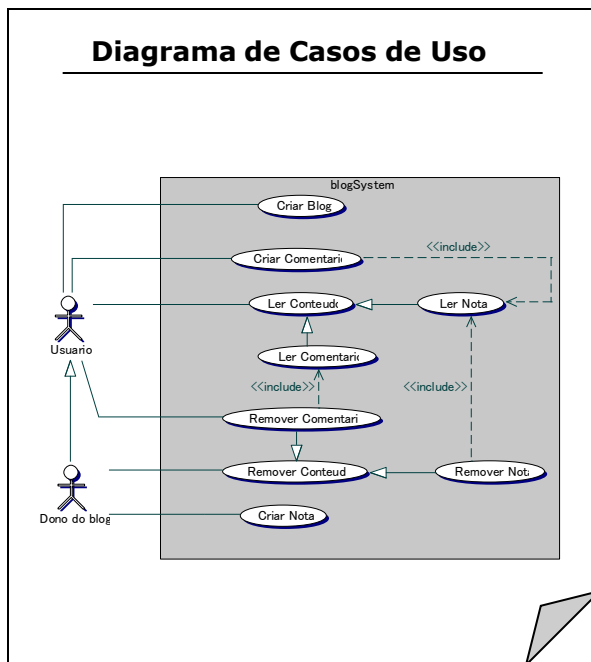
<http://www.uml.org/>  
<http://www.omg.org/>

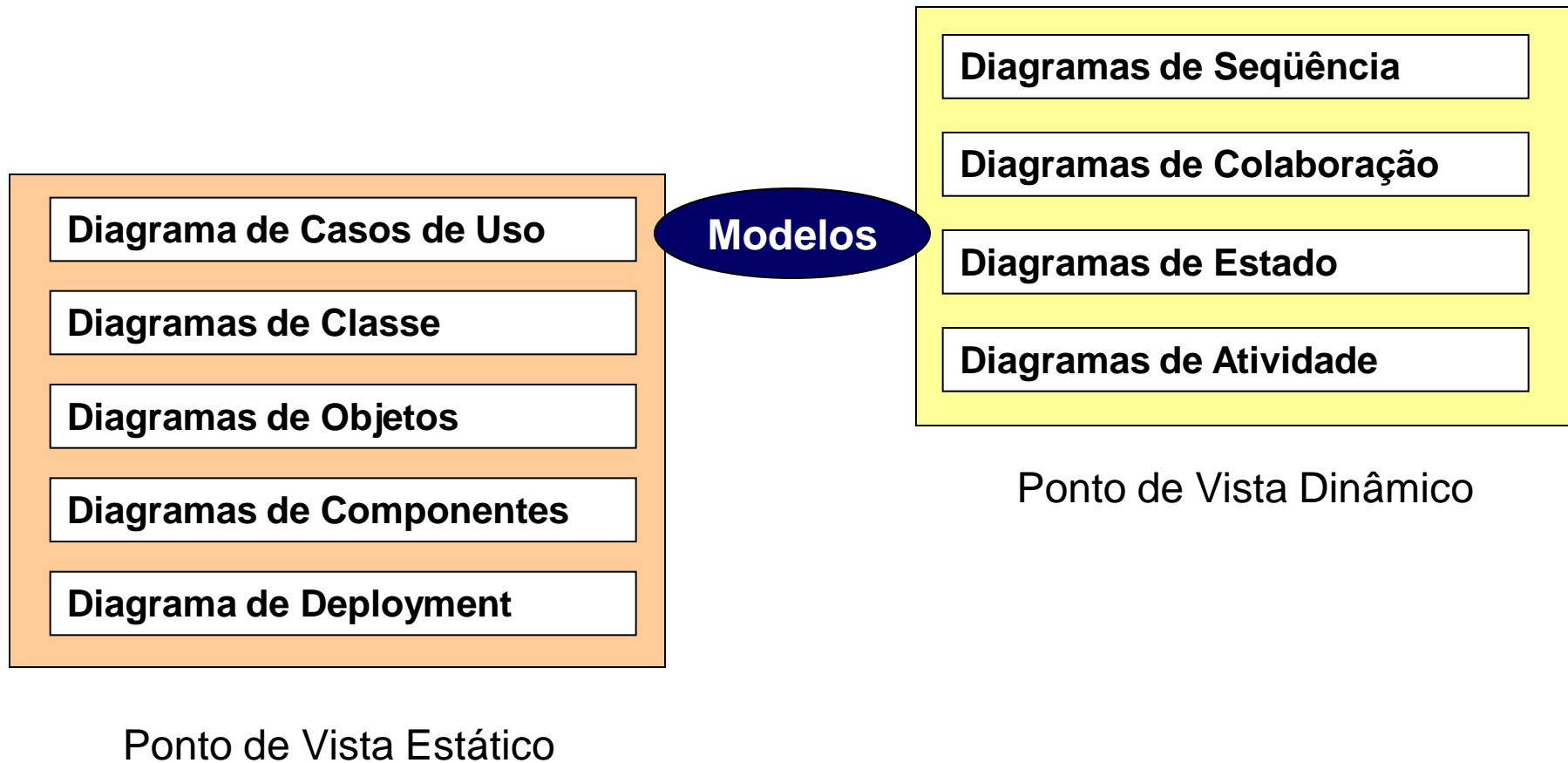
- No processo de desenvolvimento de sistemas de software, é quase impossível a visualização de toda a **estrutura de um sistema** sem o uso de modelos que a represente.
- A UML fornece os símbolos gráficos para a **representação** de artefatos de software.
- Por trás de cada símbolo empregado na notação da UML, existe uma **sintaxe** e uma **semântica** bem-definidas.
- Dessa maneira, um desenvolvedor poderá usar a UML para escrever seu modelo, **diminuindo a ambigüidade** em sua interpretação.

- No presente contexto, **especificar** significa construir modelos precisos, completos e sem ambigüidades.
- A UML atende a todas as decisões importantes em termos de **análise, projeto e implementação**, que devem ser tomadas para o desenvolvimento e implantação de sistemas complexos de software.

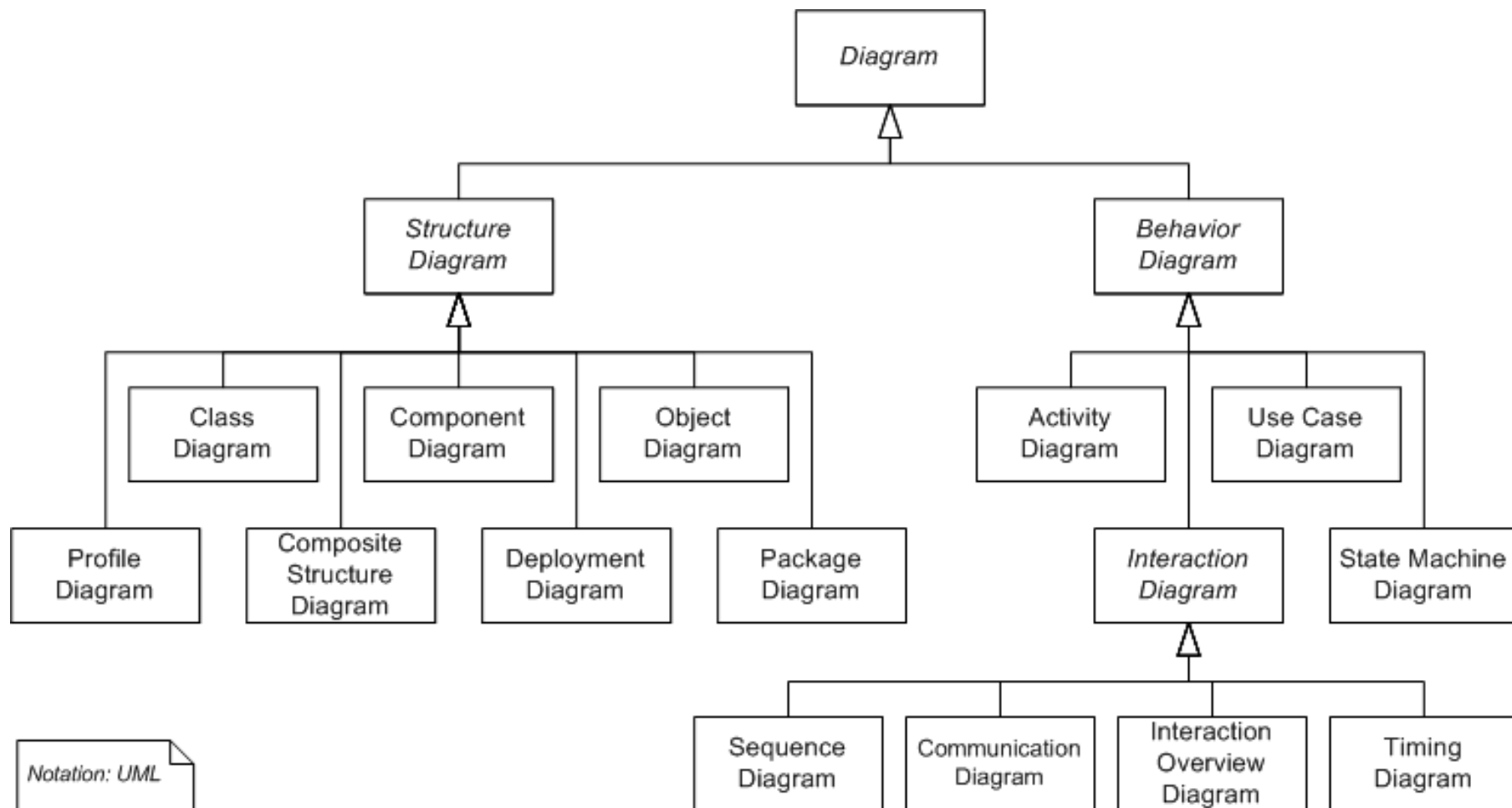
- Os modelos de UML podem ser diretamente "traduzidos" para várias linguagens de programação.
  - Isso significa que é possível mapear os modelos da UML para linguagens de programação tais como, **Java**, **C++** e **Visual Basic**.
  - Esse mapeamento permite a realização de uma **engenharia de produção**: geração de código a partir de um modelo em UML.
  - O processo inverso, a **engenharia reversa**, também é possível, com a reconstrução de um modelo a partir de sua implementação.

- Cada modelo criado é um artefato do software



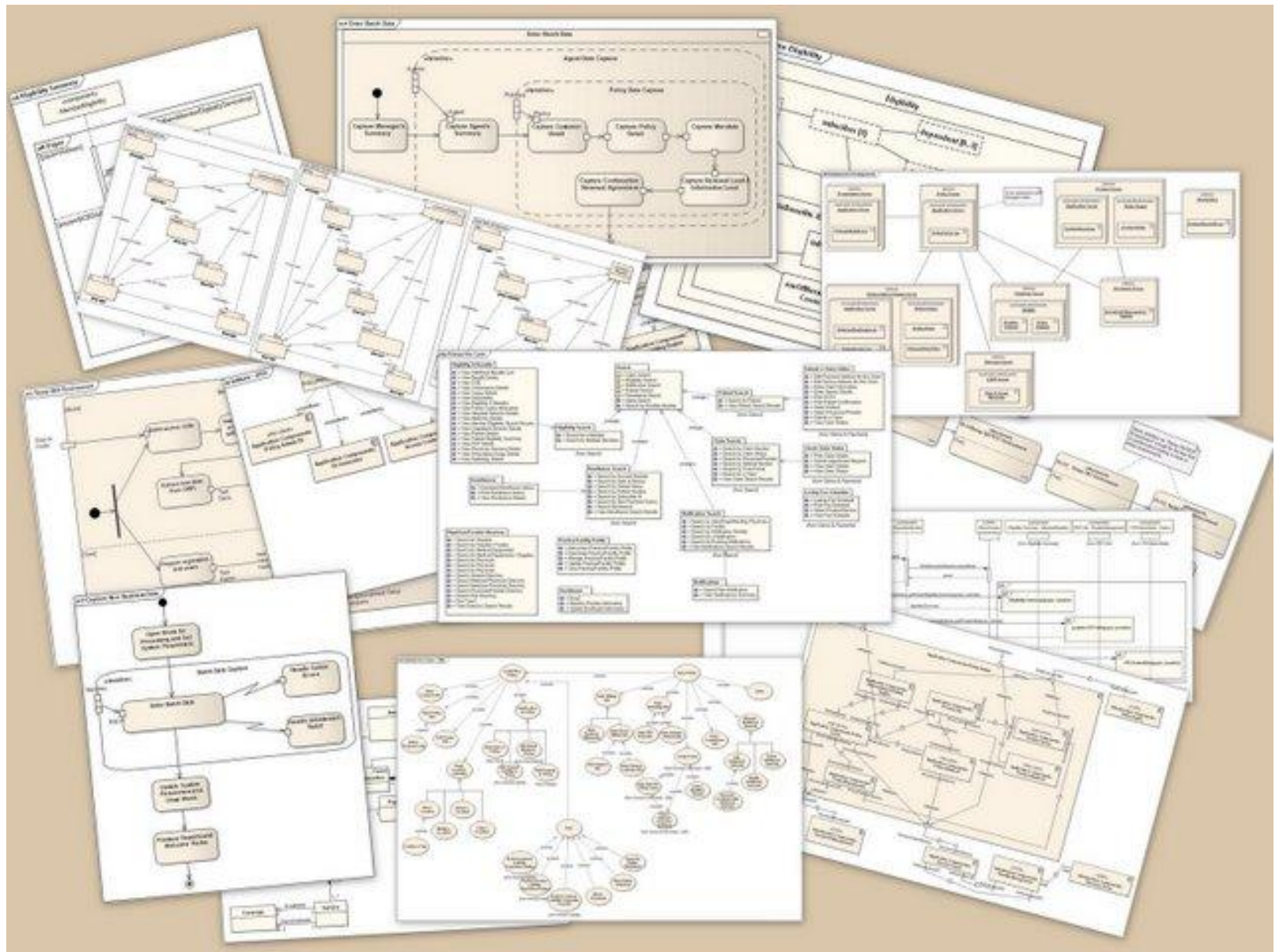


# UML 2.2 tem 14 tipos de diagramas

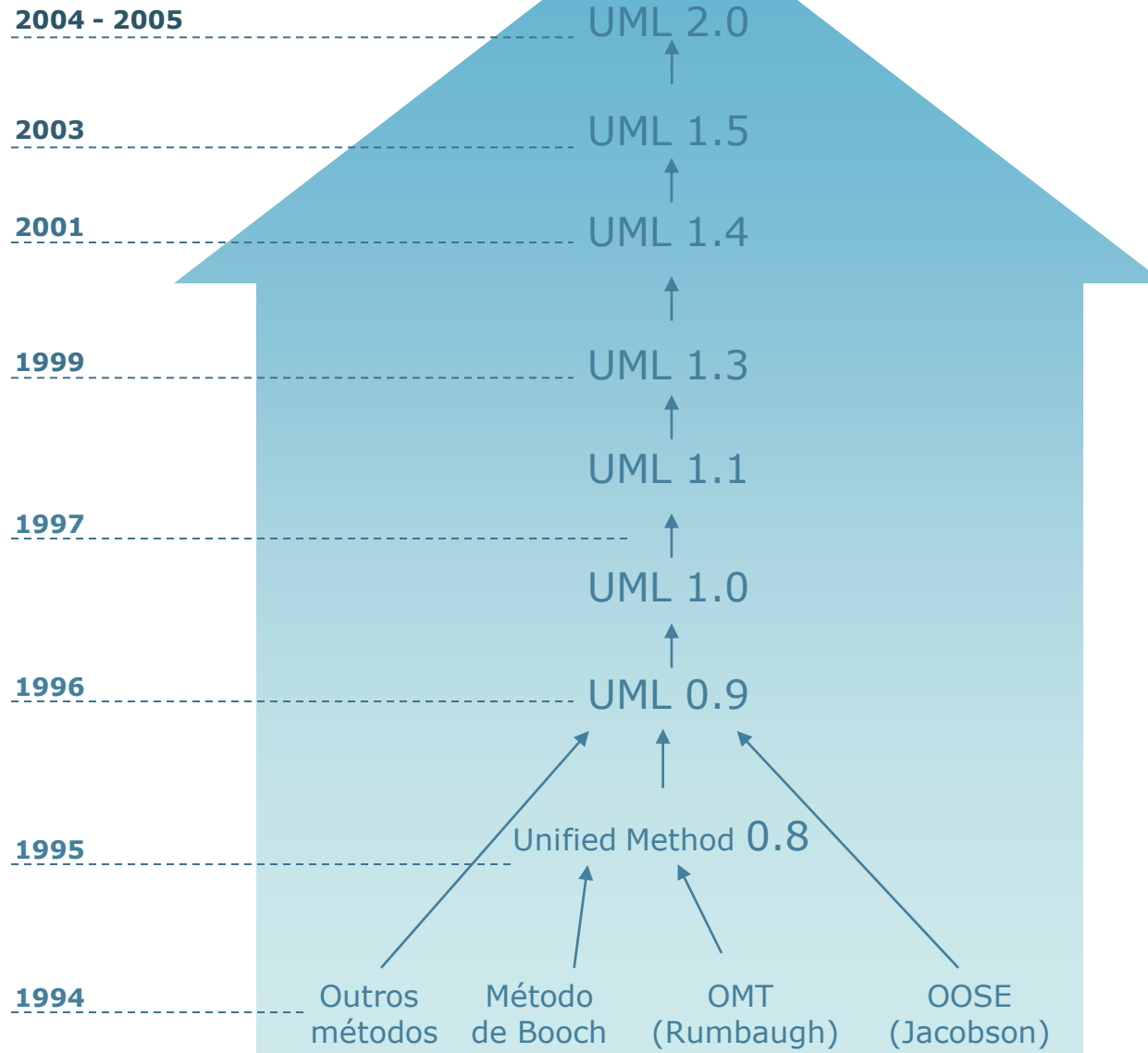




# Uma colagem de diagramas UML



- Padrão aberto e não proprietário.
- Extensível.
- Independência do processo de desenvolvimento.
- Aplicável a todas as fases do ciclo de desenvolvimento.
- Independência de linguagem de implementação.



## UML 2.0

2004 - 2005

2003

2001

1999

1997

1996

1995

1994

versão	data	conteúdo
<b>2.1</b>	04-2006	Minor revision to UML 2.0 - corrections and consistency improvements.
<b>2.1.1</b>	02-2007	Minor revision to the UML 2.1
<b>2.1.2</b>	11-2007	Minor revision to the UML 2.1.1
<b>2.2</b>	02-2009	Fixed numerous minor consistency problems and added clarifications to UML 2.1.2
<b>2.3</b>	05-2010	Minor revision to the UML 2.2, clarified <u>associations</u> and association classes, added <u>final classifier</u> , updated <u>component diagrams</u> , composite structures, actions, etc.
<b>2.4.1</b>	08-2011	Current version of UML - minor revision to the UML 2.3, few fixes and updates to classes, packages - added <u>URI package attribute</u> ; updated actions; removed creation event, execution event, send and receive operation events, send and receive signal events, renamed destruction event to <u>destruction occurrence specification</u> ; <u>profiles</u> - changed stereotypes and applied stereotypes to have upper-case first letter - <u>«Metaclass»</u> and <u>stereotype application</u> .

- **Diagramas Estruturais ou Estáticos**

- Pacotes
- Classes
- Objetos
- Estrutura Composta
- Componentes
- Instalação
- Perfil

- **Diagramas Comportamentais**

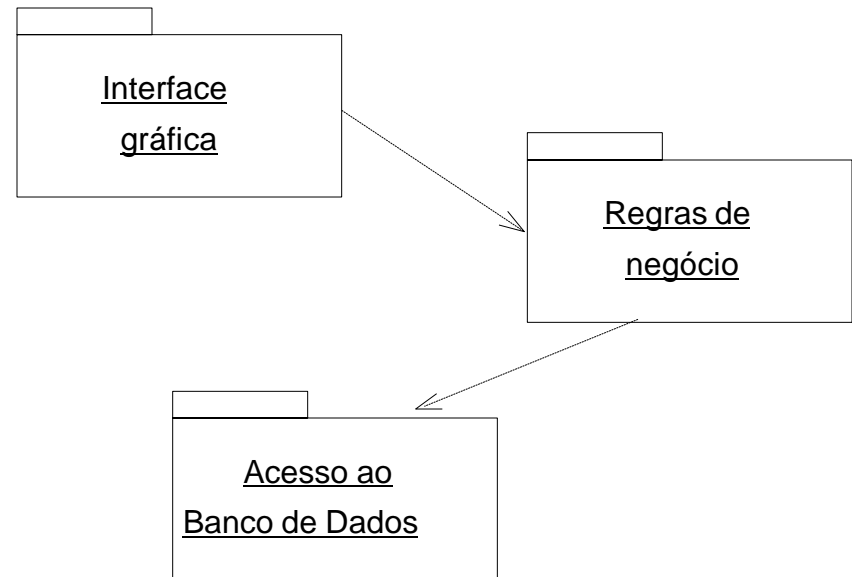
- Casos de uso
- Atividades
- Máquina de estado
- Colaboração ou Comunicação
- Seqüência
- Tempo
- Interatividade

- Diagramas estáticos ou estruturais definem estaticamente a arquitetura de um modelo.
- São usados para modelar classes, objetos, relações e componentes físicos que compõem um modelo.
- Além disso, também são usados modelar os relacionamentos e as dependências entre elementos.

# Diagramas Estruturais ou Estáticos

## DIAGRAMA DE PACOTES

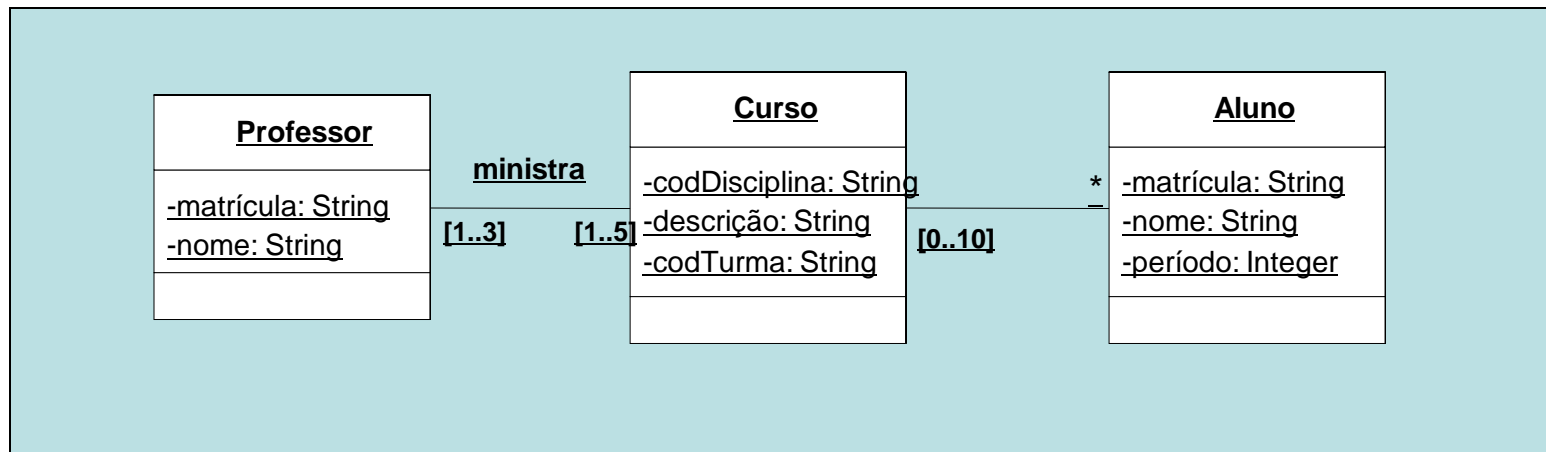
- O Diagrama de pacotes ou diagrama de módulos descreve os pacotes ou pedaços do sistema divididos em agrupamentos lógicos, mostrando as interações entre eles em alto nível (já que pacotes podem depender de outros pacotes).
- Esse diagrama é muito utilizado para ilustrar a arquitetura de um sistema mostrando o agrupamento de suas classes.



# Diagramas Estruturais ou Estáticos

## DIAGRAMA DE CLASSES

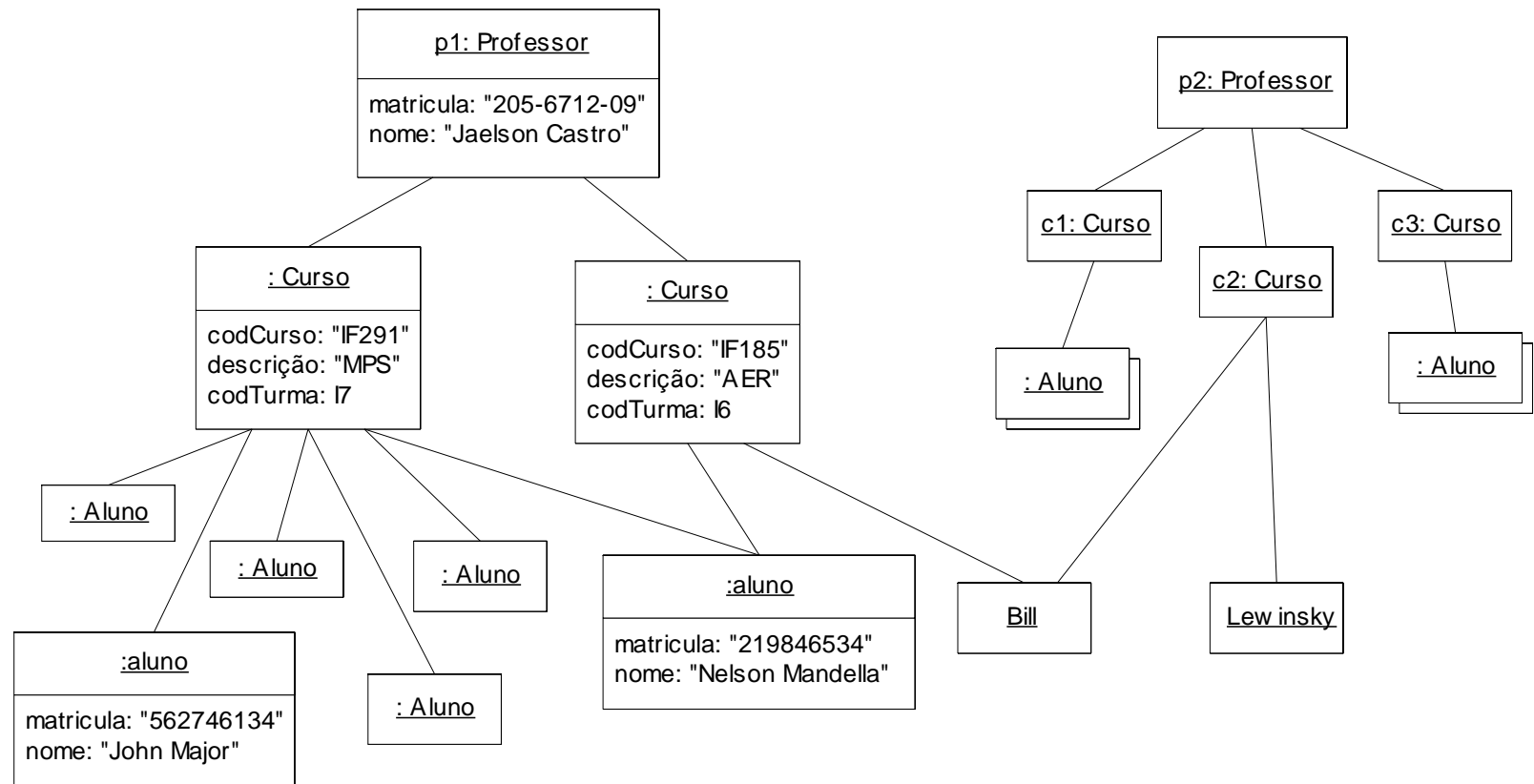
- Define os elementos básicos de um modelo, ou seja, os tipos, as classes e os relacionamentos usados para construir um modelo completo.
- Em programação, um diagrama de classes é uma representação da estrutura e relações das classes que servem de modelo para objetos.
- É uma modelagem muito útil para o sistema, definindo todas as classes que o sistema necessita possuir e é a base para a construção dos diagramas de comunicação, seqüência e estados.





- É uma variação do diagrama de classes e utiliza quase a mesma notação, com duas exceções: os objetos são escritos com seus nomes sublinhados e todas as instâncias num relacionamento são mostradas.
- A diferença é que o diagrama de objetos apresenta os atributos com valores e mostra os objetos que foram instanciados das classes. É como se fosse o perfil do sistema em um certo momento de sua execução.
- Os diagramas de objetos não são tão importantes como os diagramas de classes, mas eles são muito úteis para exemplificar diagramas complexos de classes ajudando muito em sua compreensão.
- Também são usados como parte dos diagramas de colaboração, onde a colaboração dinâmica entre os objetos do sistema é mostrada.

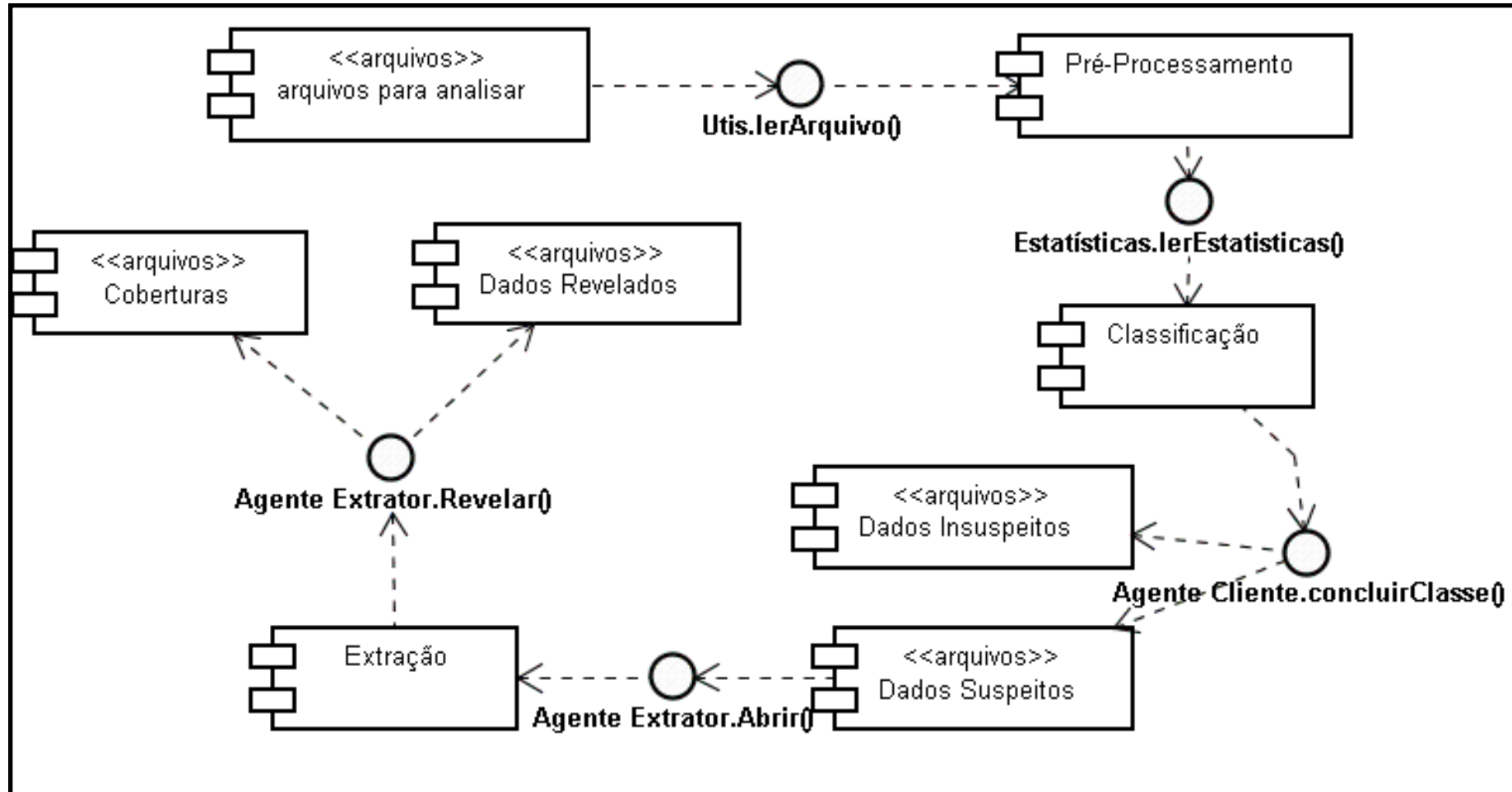
# DIAGRAMAS DE OBJETOS



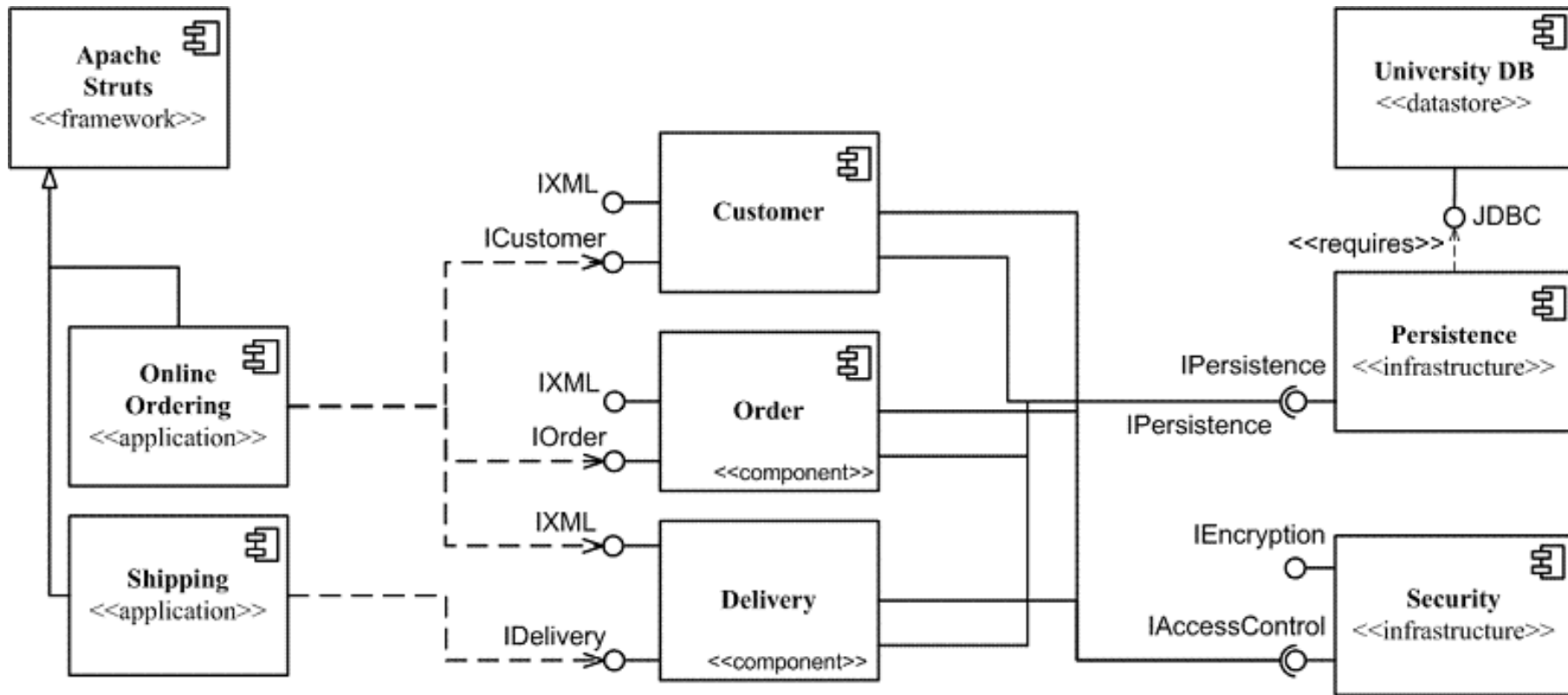
- Ilustra como as classes deverão se encontrar, organizadas através da noção de componentes de trabalho, ou seja, apresenta as dependências entre componentes de software, incluindo implementação de classes, arquivos de código fonte, arquivo de código binário, arquivos executáveis, scripts.
- Utilizado para:
  - \* Modelar os componentes do código-fonte, do código executável do software;
  - \* Destacar a função de cada módulo para facilitar a sua reutilização;
  - \* Auxiliar no processo de engenharia reversa, por meio da organização dos módulos do sistema e seus relacionamentos.

Componente: Peça física distribuível e substituível de código que contém elementos que apresentam um conjunto de interfaces requeridas e fornecidas.

## DIAGRAMA DE COMPONENTES - EXEMPLO



# DIAGRAMA DE COMPONENTES – EXEMPLO UML 2.0

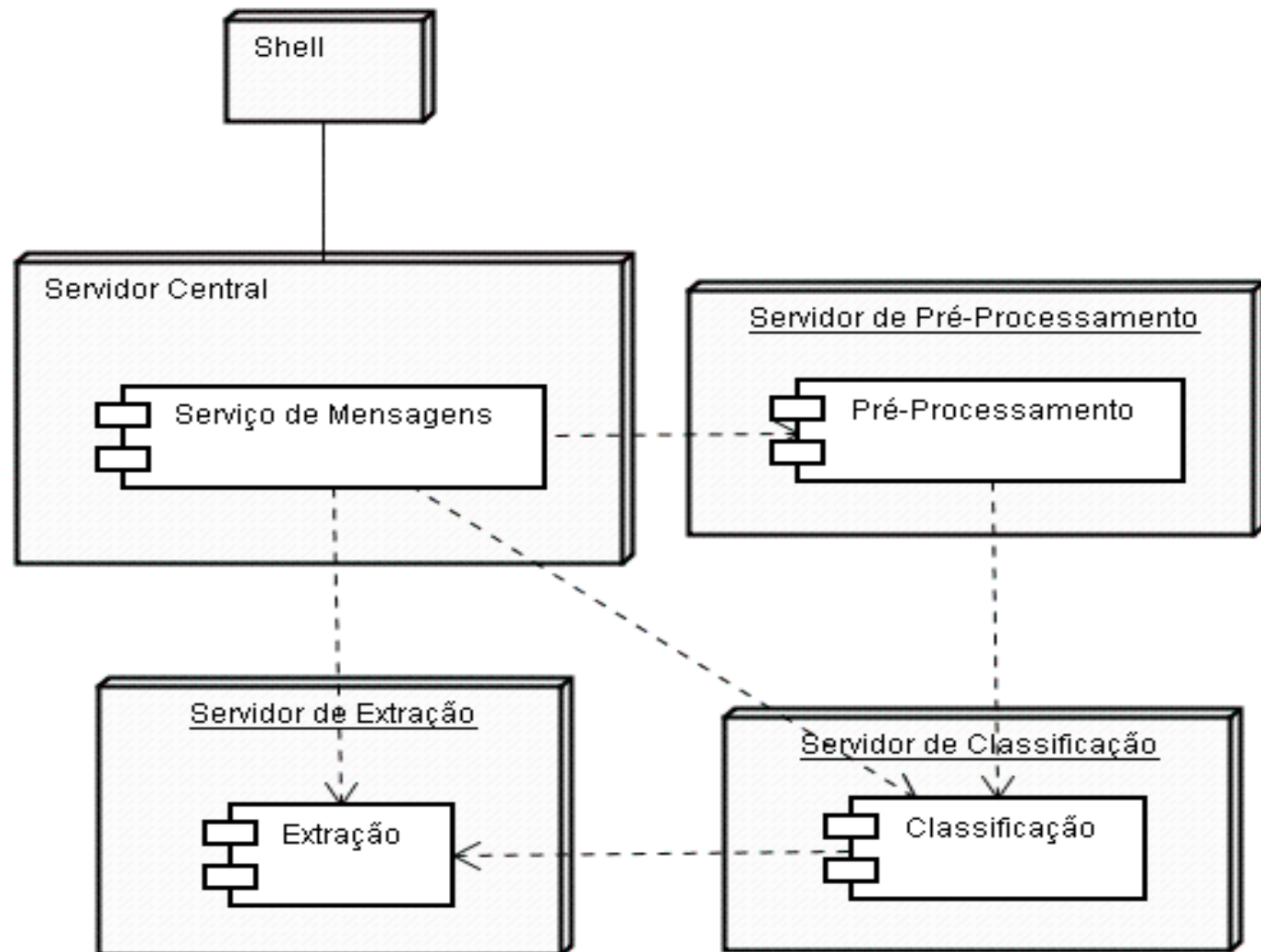


# Diagramas Estruturais ou Estáticos

## DIAGRAMA DE INSTALAÇÃO OU EXECUÇÃO

- Mostra a configuração dos elementos de processamento em tempo de execução, além dos componentes, processos e objetos do sistema neles existentes, sendo usado para modelar a arquitetura de um sistema de informação da perspectiva dos seus componentes físicos/hardware (computadores, adaptadores de rede, impressoras, routers etc), explicitando as suas dependências de comunicação, assim como, que componentes são instaladas em cada nó computacional, ilustrando a configuração dos elementos de processamento e dos componentes de software, processos e objetos neles suportados. Modela também sistemas cliente-servidor e sistemas distribuídos.

## DIAGRAMA DE INSTALAÇÃO



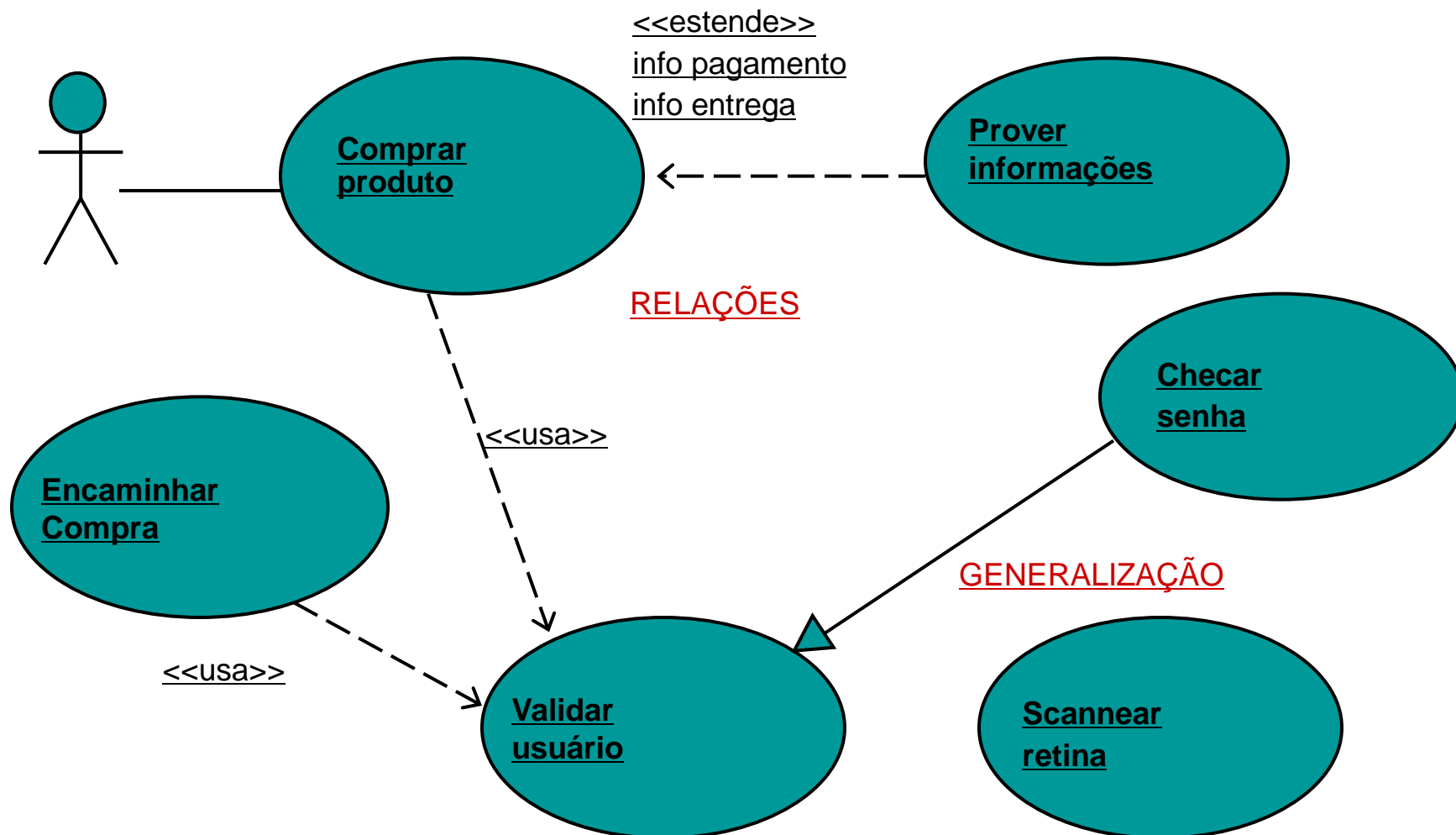
# DIAGRAMA COMPORTAMENTAIS OU DINÂMICOS

- Os diagramas dinâmicos ou de comportamento apresentam as variedades da interação e do estado instantâneo dentro de um modelo enquanto é “executado”.



- Descreve a funcionalidade proposta para o novo sistema, sendo usado para modelar interações do usuário/sistema.
- Define o comportamento, as exigências e o resultado esperado de uma funcionalidade.
- Segundo Ivar Jacobson, idealizador do modelo, pode-se dizer que um Caso de Uso é um "documento narrativo que descreve a seqüência de eventos de um ator que usa um sistema para completar um processo".
- Um Caso de Uso pode "usar" outra funcionalidade de Caso de Uso ou "estender" outro Caso de Uso com seu próprio comportamento.

# DIAGRAMA DE CASOS DE USO

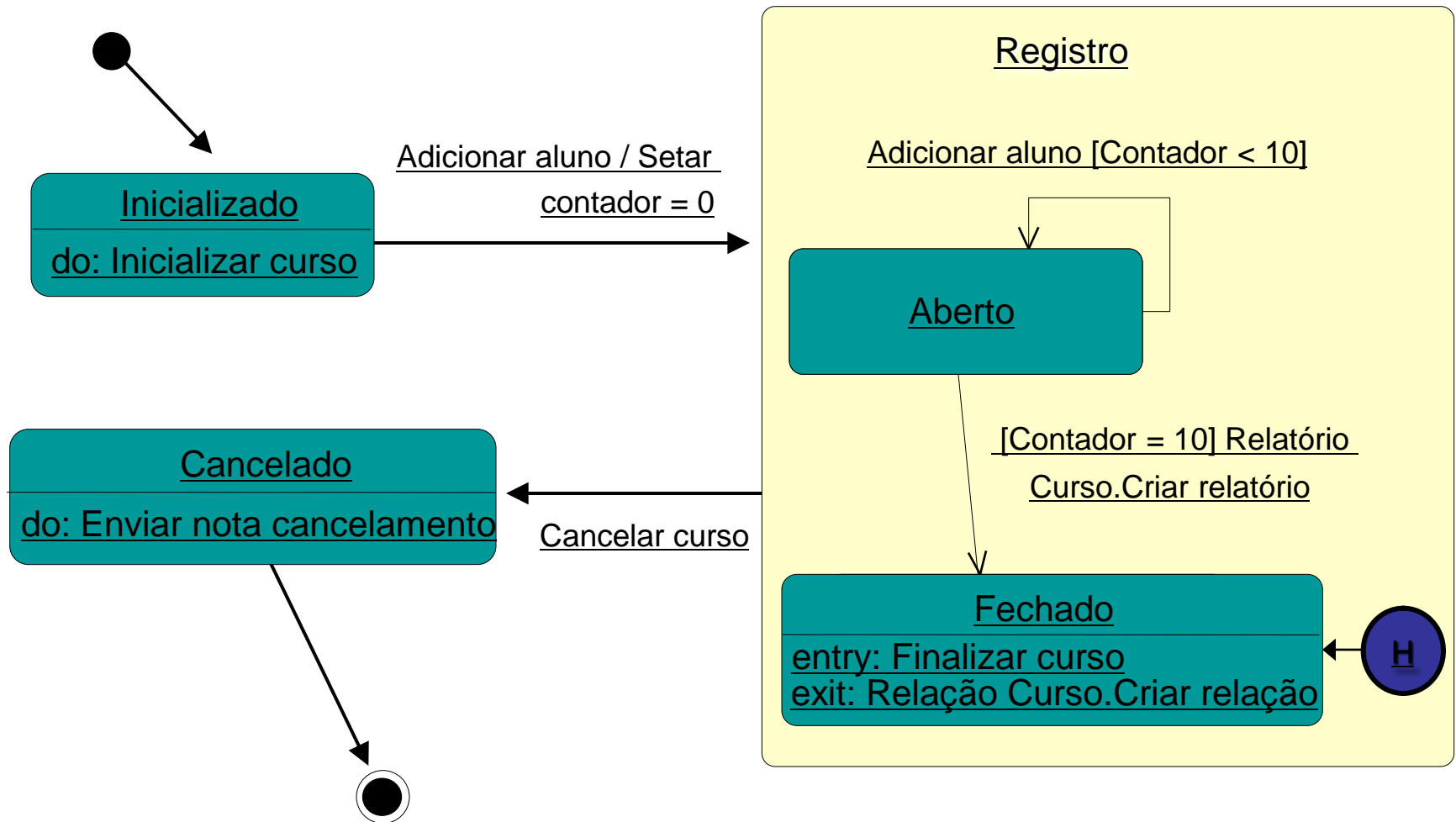


# Diagramas Comportamentais ou Dinâmicos

## DIAGRAMA DE MÁQUINA DE ESTADOS

- Em engenharia de software e eletrônica digital, um diagrama de transição de estados é uma representação do estado ou situação em que um objeto pode se encontrar no decorrer da execução de processos de um sistema.
- Com isso, o objeto pode passar de um estado A (estado inicial) para um estado B (estado final) através de uma transição.
- Representa as ações ocorridas em resposta ao recebimento de um evento, onde cada ponto de parada representa um estado da aplicação.

# DIAGRAMA DE MÁQUINA DE ESTADOS



- É uma variação do diagrama de máquina de estados que representa a execução das ações e as transições que são acionadas pela conclusão de outras ações ou atividades, ou seja, um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra.
- Comumente, isso envolve a modelagem das etapas seqüenciais em um processo computacional.
- Os diagramas de atividade não são importantes somente para a modelagem de aspectos dinâmicos de um sistema ou um fluxograma, mas também para a construção de sistemas executáveis por meio de engenharia de produção reversa.

# DIAGRAMA DE ATIVIDADES

SWIMLANE

INÍCIO

SINCRONIZAÇÃO  
(FORK)

FLUXO DE AÇÃO

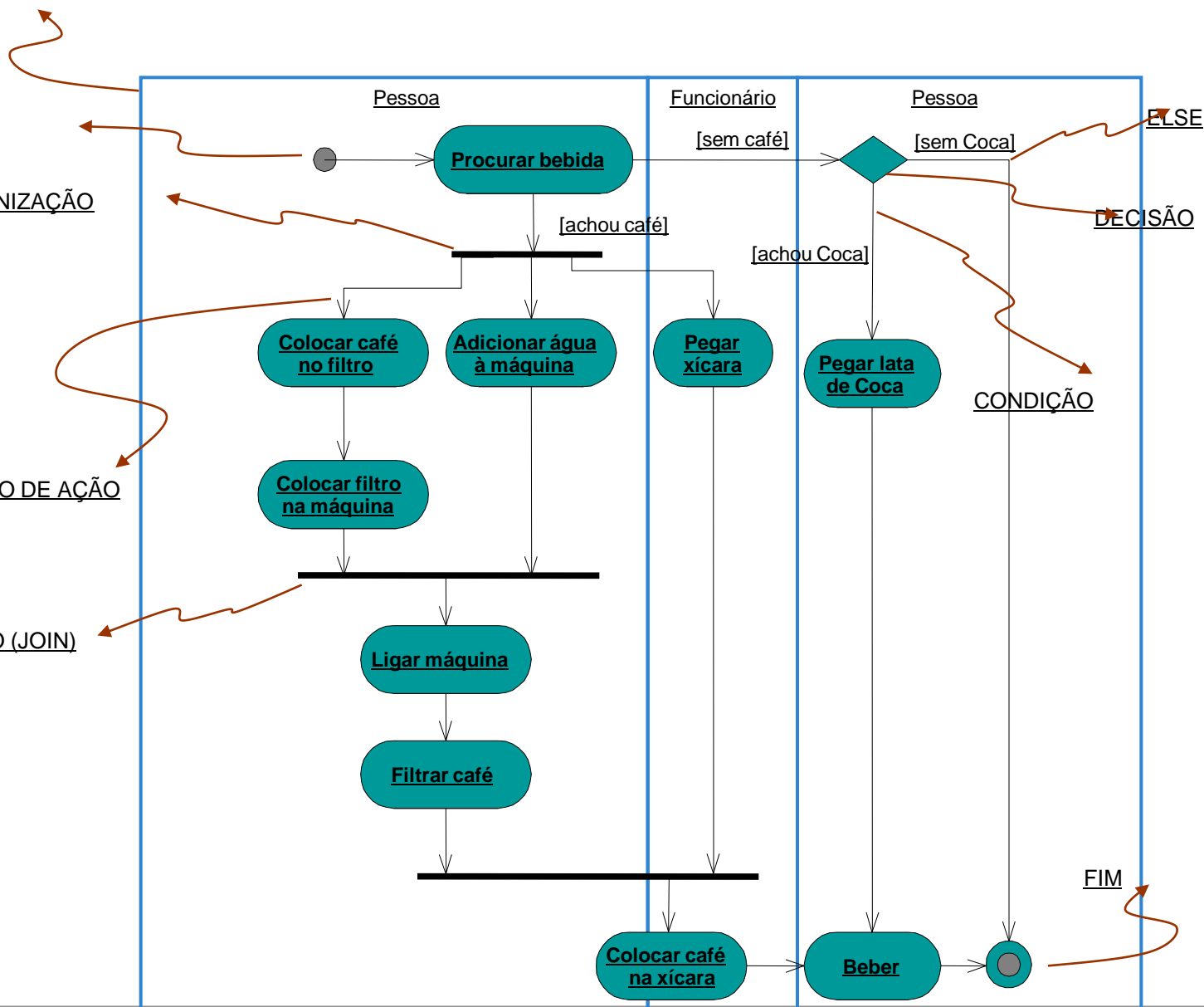
JUNÇÃO (JOIN)

ELSE

DECISÃO

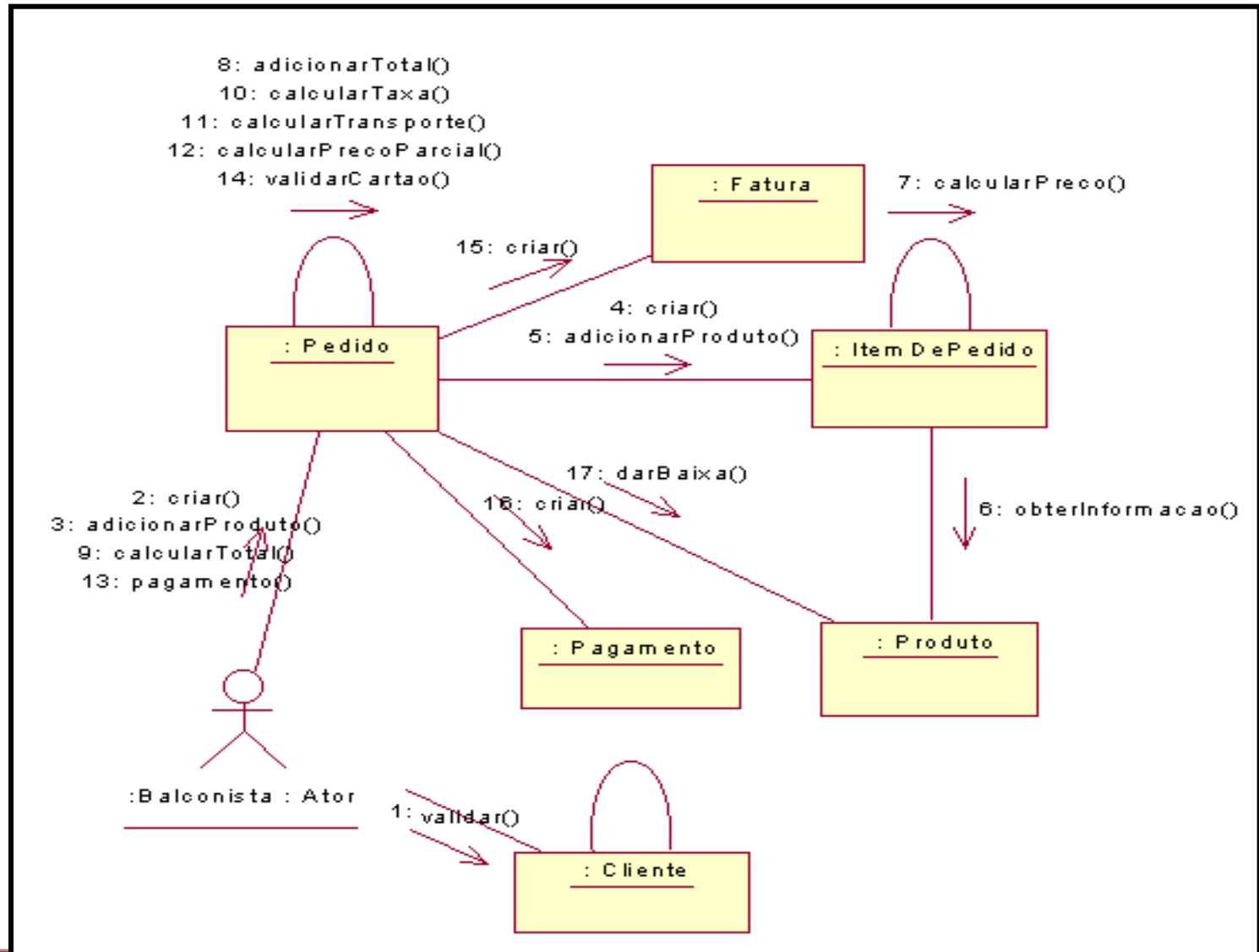
CONDIÇÃO

FIM



- Mostra, de maneira semelhante ao diagrama de seqüência, a colaboração dinâmica entre os objetos, ou seja, exibe uma interação, consistindo de um conjunto de objetos e seus relacionamentos, incluindo as mensagens que podem ser trocadas entre eles.
- Se a ênfase do diagrama for o decorrer do tempo, o ideal é o diagrama de seqüência, mas se a ênfase for o contexto do sistema, a prioridade será o diagrama de colaboração que dá ênfase à ordenação estrutural em que as mensagens são trocadas entre os objetos de um sistema.

# DIAGRAMA DE COLABORAÇÃO OU COMUNICAÇÃO



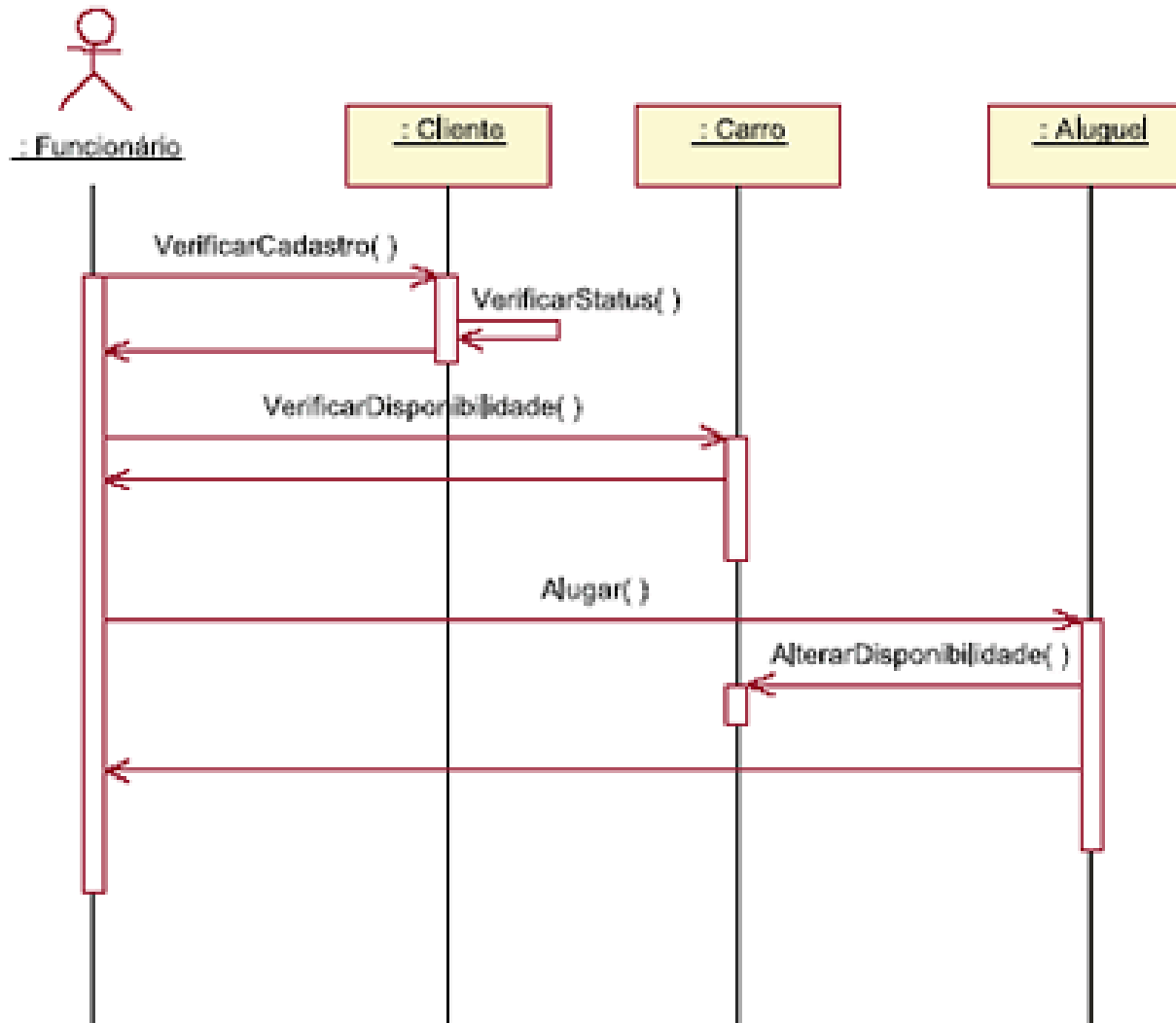


# Diagramas Comportamentais ou Dinâmicos

## DIAGRAMA DE SEQÜÊNCIA DE MENSAGENS

- Representa a seqüência de processos (mais especificamente de mensagens passadas entre objetos) num programa de computador. É usado para representar o comportamento das operações, métodos (procedimentos ou funções) entre classes de um cenário.
- Ele registra o comportamento de um único caso de uso (interações entre objetos de um cenário, realizadas através de operações ou métodos (procedimentos ou funções)). Exibe os objetos e as mensagens passadas entre esses objetos no caso de uso, descrevendo a maneira como os grupos de objetos colaboram em algum comportamento ao longo do tempo.
- O diagrama de seqüência dá ênfase à ordenação temporal em que as mensagens são trocadas entre os objetos de um sistema. Entende-se por mensagens os serviços solicitados de um objeto a outro, e as respostas desenvolvidas para as solicitações.

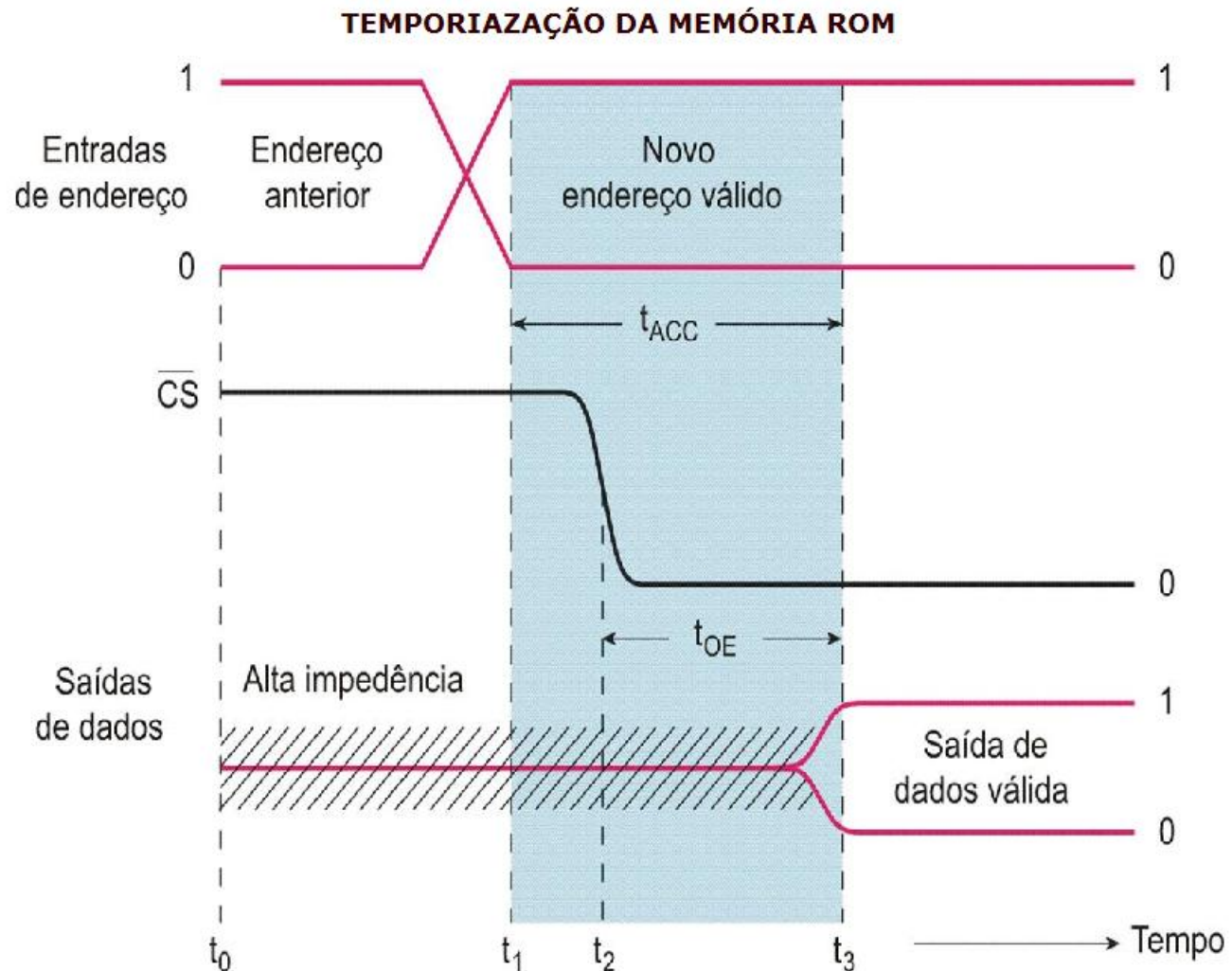
# DIAGRAMA DE SEQUÊNCIA DE MENSAGENS



# DIAGRAMA DE TEMPO OU TEMPORAL

- O Diagrama de tempo (Timing Diagram), incluído a partir da UML 2.0, apresenta o comportamento dos objetos e sua interação em uma escala de tempo, focalizando as condições que mudam no decorrer desse período.
- Modo alternativo de mostrar um diagrama de seqüência, ou melhor, a fusão dos diagramas de seqüência e de estado, apresentando o estado dos objetos em relação ao tempo e as mensagens que modificam esse estado, usando para isso métrica de tempo na linha de vida.
- Pode ser útil em aplicações de tempo real.

# DIAGRAMA DE TEMPO (EXEMPLO)



# DIAGRAMA DE INTERATIVIDADE

- Diagramas de interatividade são variações de "Diagrama de atividades", ou melhor, a fusão do diagrama de atividades e seqüência. Ele permite que fragmentos das interações sejam facilmente combinados aos pontos e fluxos de decisão.

Nele, seqüências formam um fluxo de atividades, mostrando como elas trabalham em uma seqüência de eventos.

# **UML: Casos de Uso**

Projeto de Sistemas de Software

- Os casos de uso:
  - ✓ Descrevem como os usuários interagem com o sistema (as funcionalidades do sistema)
  - ✓ Facilitam a organização dos requisitos de um sistema
  - ✓ Dão uma visão externa do sistema
  - ✓ O conjunto de casos de uso deve ser capaz de comunicar a funcionalidade e o comportamento do sistema para o cliente
  - ✗ Descrevem **o que** o sistema faz, mas NÃO especificam **como** isso deve ser feito

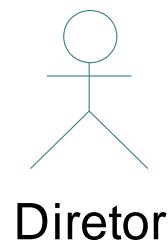
- Elementos do diagrama:
  - Atores
  - Casos de uso
  - Relacionamentos
    - Associação
    - Generalização
    - Dependência: Extensão e Inclusão
  - Fronteira do sistema



- Elementos do diagrama
  - **Atores**
  - Casos de uso
  - Relacionamentos
    - Associação
    - Generalização
    - Dependência: Extensão e Inclusão
  - Fronteira do sistema

- Atores
  - Representam os papéis desempenhados por elementos externos ao sistema
    - Ex: humano (usuário), dispositivo de hardware ou outro sistema (cliente)
  - Elementos que interagem com o sistema

Notação:



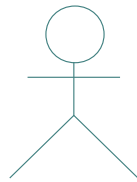
## Exemplo: Loja de CDs

### Identificando os atores

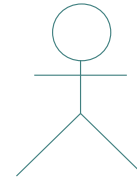
- Uma loja de CDs possui discos para venda. Um cliente pode comprar uma quantidade ilimitada de discos para isto ele deve se dirigir à loja. A loja possui um **atendente** cuja função é atender os clientes durante a venda dos discos. A loja também possui um **gerente** cuja função é administrar o estoque para que não falem discos. Além disso é ele quem dá folga ao atendente, ou seja, ele também atende os clientes durante a venda dos discos.

## Exemplo: Loja de CDs

Identificando os atores



Gerente



Atendente

- E o **Cliente**:
  - Não é ator pois ele não interage com o sistema!

- Elementos do diagrama
  - Atores
  - **Casos de uso**
  - Relacionamentos
    - Associação
    - Generalização
    - Dependência: Extensão e Inclusão
  - Fronteira do sistema

- Caso de Uso
  - Representa uma funcionalidade do sistema (um requisito funcional)
  - É iniciado por um ator ou por outro caso de uso

## Dicas:

- ✓ Nomeie os casos de uso iniciando por um verbo

## Notação:



Nome do Caso de Uso

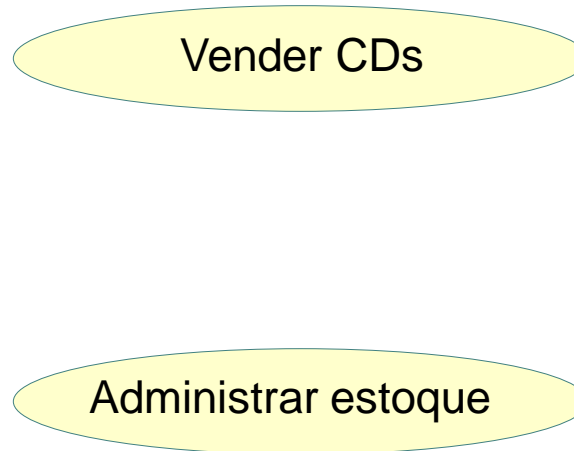
## Exemplo: Loja de CDs

### Identificando os casos de uso

- Uma loja de CDs possui discos para venda. Um cliente pode comprar uma quantidade ilimitada de discos para isto ele deve se dirigir à loja. A loja possui um atendente cuja função é atender os clientes durante a **venda dos discos**. A loja também possui um gerente cuja função é **administrar o estoque** para que não falem discos. Além disso é ele quem dá folga ao atendente, ou seja, ele também atende os clientes durante a **venda dos discos**.

## Exemplo: Loja de CDs

Identificando os casos de uso





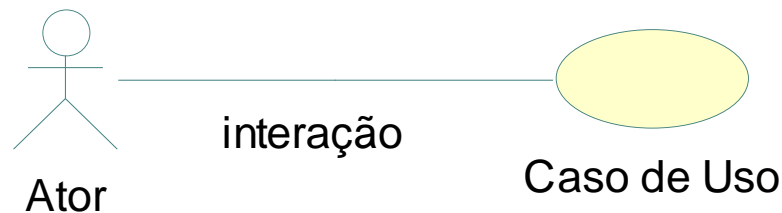
- Elementos do diagrama
  - Atores
  - Casos de uso
  - **Relacionamentos**
    - Associação
    - Generalização
    - Dependência: Extensão e Inclusão
  - Fronteira do sistema

- Relacionamento de associação
  - Indica que há uma interação (comunicação) entre um caso de uso e um ator
  - Um ator pode se comunicar com vários casos de uso

## Dicas:

- ✗ NÃO use setas nas associações
- ✗ Associações NÃO representam fluxo de informação

## Notação:



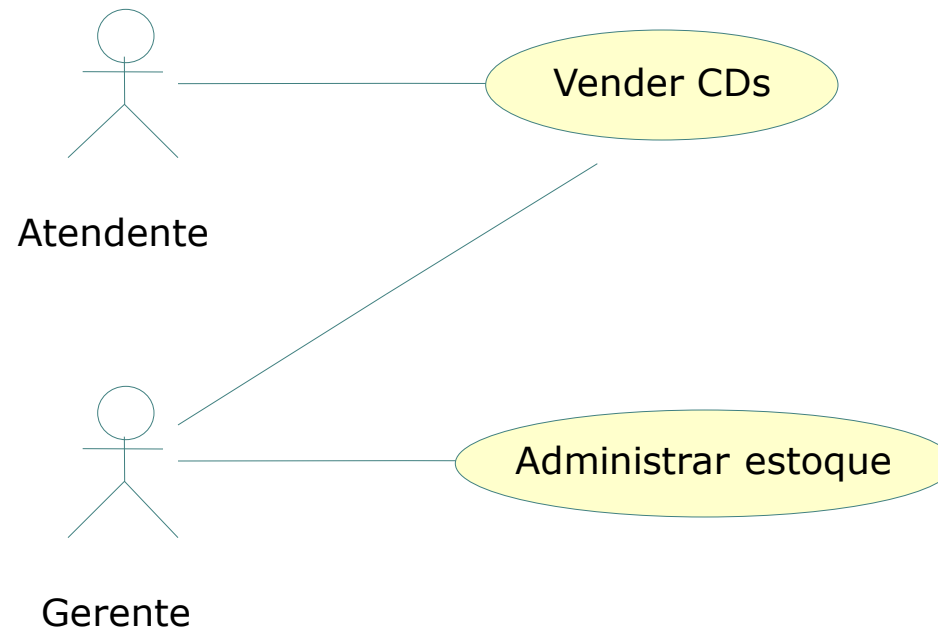
## Exemplo: Loja de CDs

### Identificando os relacionamentos de associação

- Uma loja de CDs possui discos para venda. Um cliente pode comprar uma quantidade ilimitada de discos para isto ele deve se dirigir à loja. A loja possui um **atendente** cuja função é atender os clientes durante a **venda dos discos**. A loja também possui um **gerente** cuja função é **administrar o estoque** para que não falem discos. Além disso é ele quem dá folga ao atendente, ou seja, ele também atende os clientes durante a **venda dos discos**.

## Exemplo: Loja de CDs

Identificando os relacionamentos de associação



- Elementos do diagrama
  - Atores
  - Casos de uso
  - **Relacionamentos**
    - Associação
    - Generalização
    - Dependência: Extensão e Inclusão
  - Fronteira do sistema

- Relacionamento de generalização

## Generalização de atores

- Quando dois ou mais atores podem se comunicar com o mesmo conjunto de casos de uso
- Um filho (herdeiro) pode se comunicar com todos os casos de uso que seu pai se comunica.

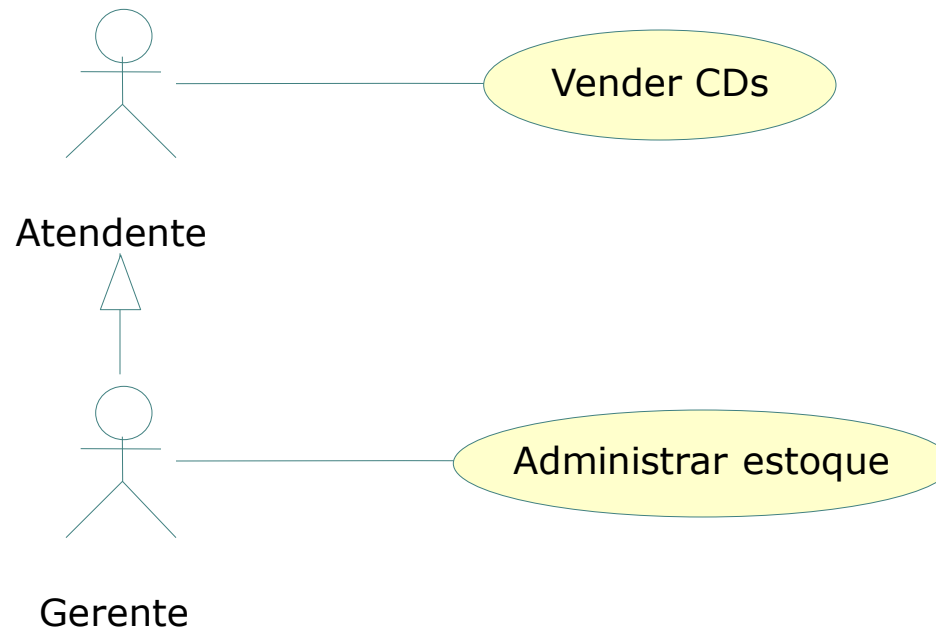
**Dica:** coloque os herdeiros embaixo

**Notação:**



## Exemplo: Loja de CDs

Identificando generalização de atores



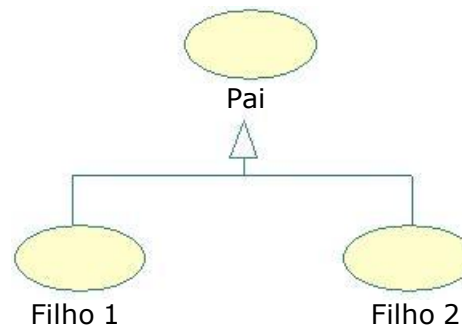
- Relacionamento de generalização

## Generalização de casos de uso

- O caso de uso filho herda o comportamento e o significado do caso de uso pai
- O caso de uso filho pode incluir ou sobrescrever o comportamento do caso de uso pai
- O caso de uso filho pode substituir o caso de uso pai em qualquer lugar que ele apareça

**Dica:** deve ser aplicada quando uma condição resulta na definição de diversos fluxos alternativos.

**Notação:**





## Exemplo: Loja de CDs

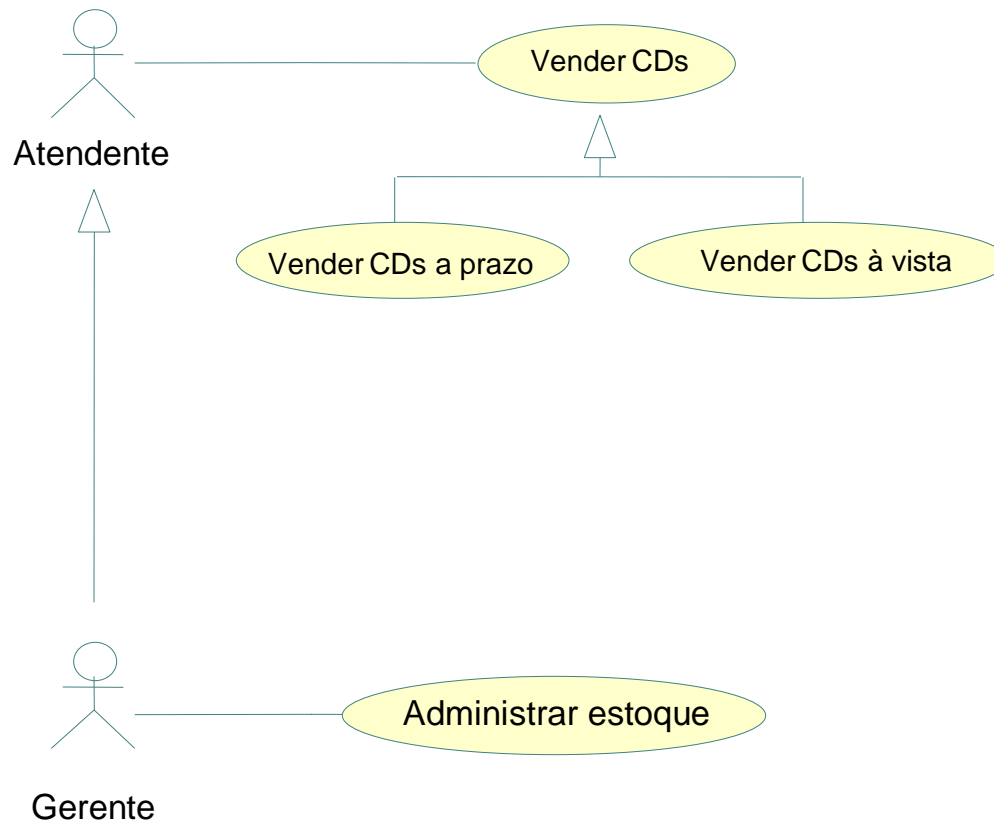
Identificando generalização de casos de uso

### Novos requisitos:

- As vendas podem ser **à vista** ou **a prazo**. Em ambos os casos o estoque é atualizado e uma nota fiscal, entregue ao consumidor.
  - No caso de uma venda à vista, clientes cadastrados na loja e que comprem mais de 5 CDs de uma só vez ganham um desconto de 1% para cada ano de cadastro.
  - No caso de uma venda a prazo, ela pode ser parcelada em 2 pagamentos com um acréscimo de 20%. As vendas a prazo podem ser pagas no cartão ou no boleto. Para pagamento com boleto, são gerados boletos bancários que são entregues ao cliente e armazenados no sistema para lançamento posterior no caixa. Para pagamento com cartão, os clientes com mais de 10 anos de cadastro na loja ganham o mesmo desconto das compras a vista.

## Exemplo: Loja de CDs

Identificando generalização de casos de uso



## Exemplo: Loja de CDs

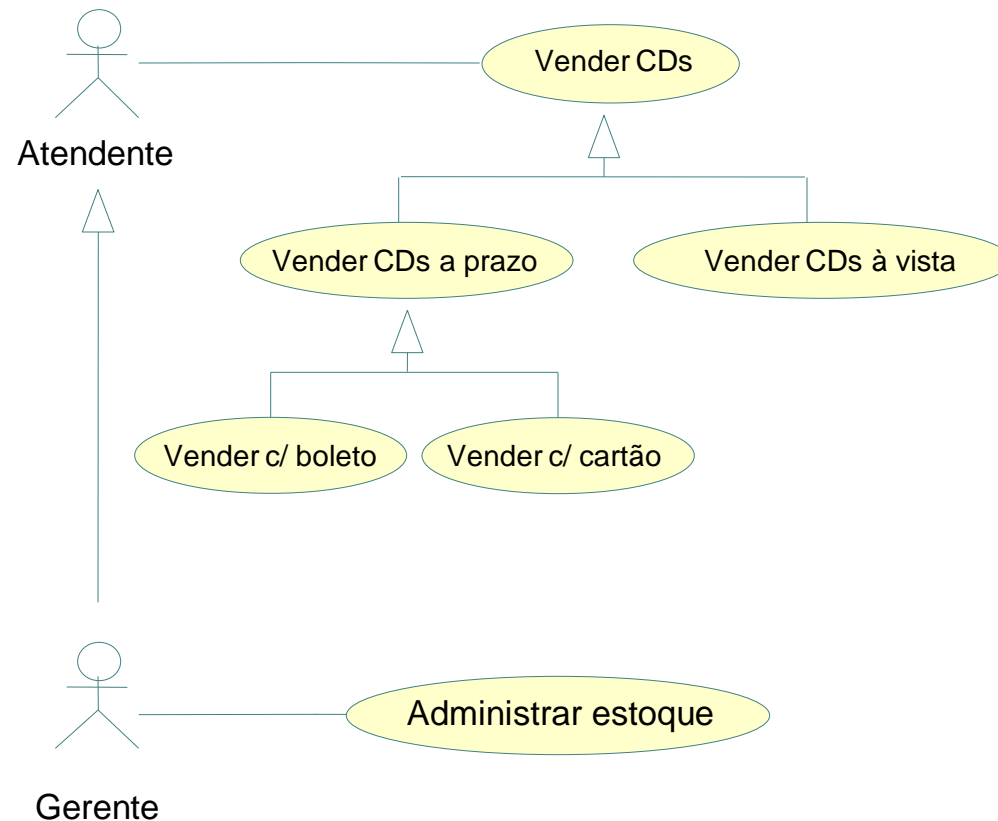
Identificando mais generalização de casos de uso

### Novos requisitos:

- As vendas podem ser **à vista** ou **a prazo**. Em ambos os casos o estoque é atualizado e uma nota fiscal, entregue ao consumidor.
  - No caso de uma venda à vista, clientes cadastrados na loja e que comprem mais de 5 CDs de uma só vez ganham um desconto de 1% para cada ano de cadastro.
  - No caso de uma venda a prazo, ela pode ser parcelada em 2 pagamentos com um acréscimo de 20%. As vendas a prazo podem ser pagas no **cartão** ou no **boleto**. Para pagamento com boleto, são gerados boletos bancários que são entregues ao cliente e armazenados no sistema para lançamento posterior no caixa. Para pagamento com cartão, os clientes com mais de 10 anos de cadastro na loja ganham o mesmo desconto das compras a vista.

## Exemplo: Loja de CDs

Identificando generalização de casos de uso



- Elementos do diagrama
  - Atores
  - Casos de uso
  - **Relacionamentos**
    - Associação
    - Generalização
    - Dependência: Extensão e Inclusão
  - Fronteira do sistema

- Relacionamento de dependência:

## Extensão:

- Representa uma variação/extensão do comportamento do caso de uso base
- O caso de uso estendido só é executado sob certas circunstâncias
- Separa partes obrigatórias de partes opcionais
  - Partes obrigatórias: caso de uso base
  - Partes opcionais: caso de uso estendido
- Fatorar comportamentos variantes do sistema (podendo reusar este comportamento em outros casos de uso)

Notação: <<extends>>



## Exemplo: Loja de CDs

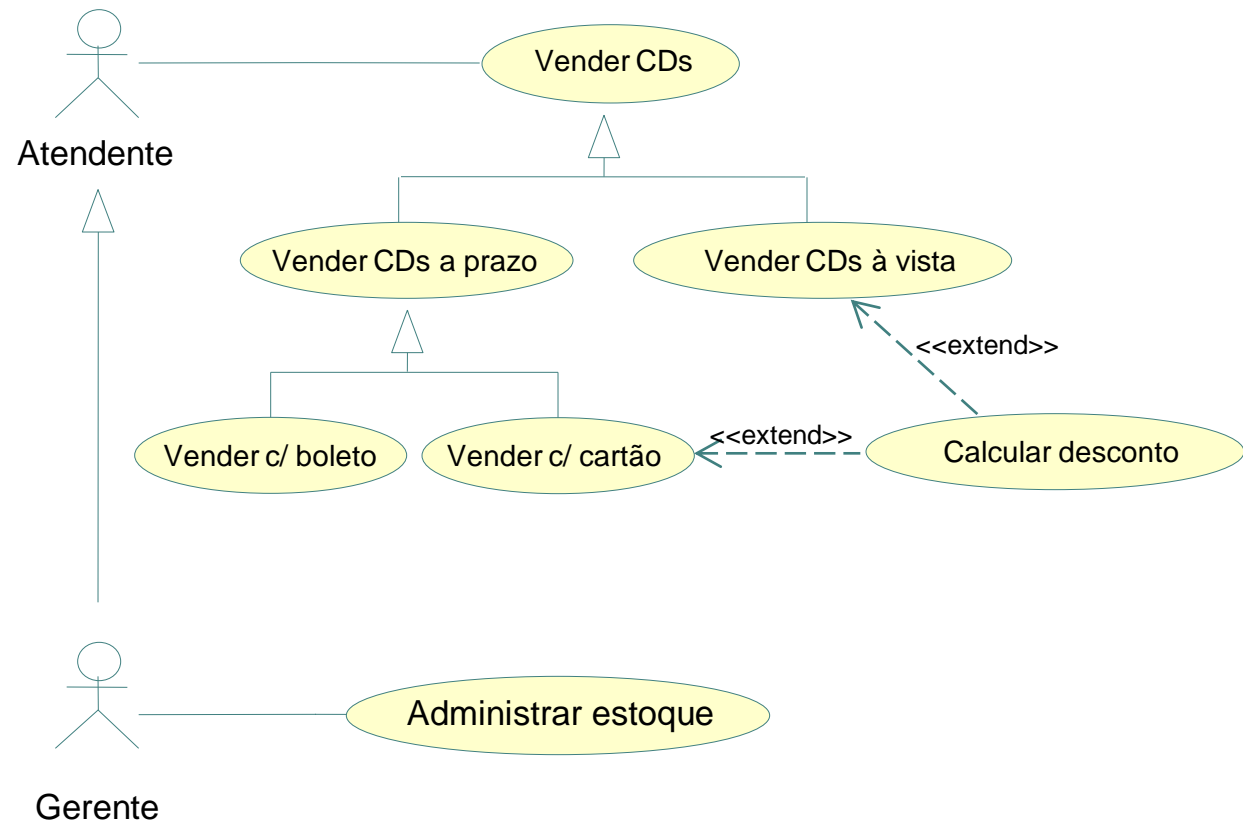
Identificando dependência: extensão

### Novos requisitos:

- No caso de uma venda à vista, clientes cadastrados na loja e que comprem mais de 5 CDs de uma só vez ganham um **desconto** de 1% para cada ano de cadastro.
- No caso de uma venda a prazo...  
...Para pagamento com cartão, os clientes com mais de 10 anos de cadastro na loja ganham o mesmo **desconto** das compras à vista.

## Exemplo: Loja de CDs

Identificando dependência: extensão





- Relacionamento de dependência:

## Inclusão:

- Evita repetição ao fatorar uma atividade comum a dois ou mais casos de uso
- Um caso de uso pode incluir vários casos de uso

Notação: <<includes>>



## Exemplo: Loja de CDs

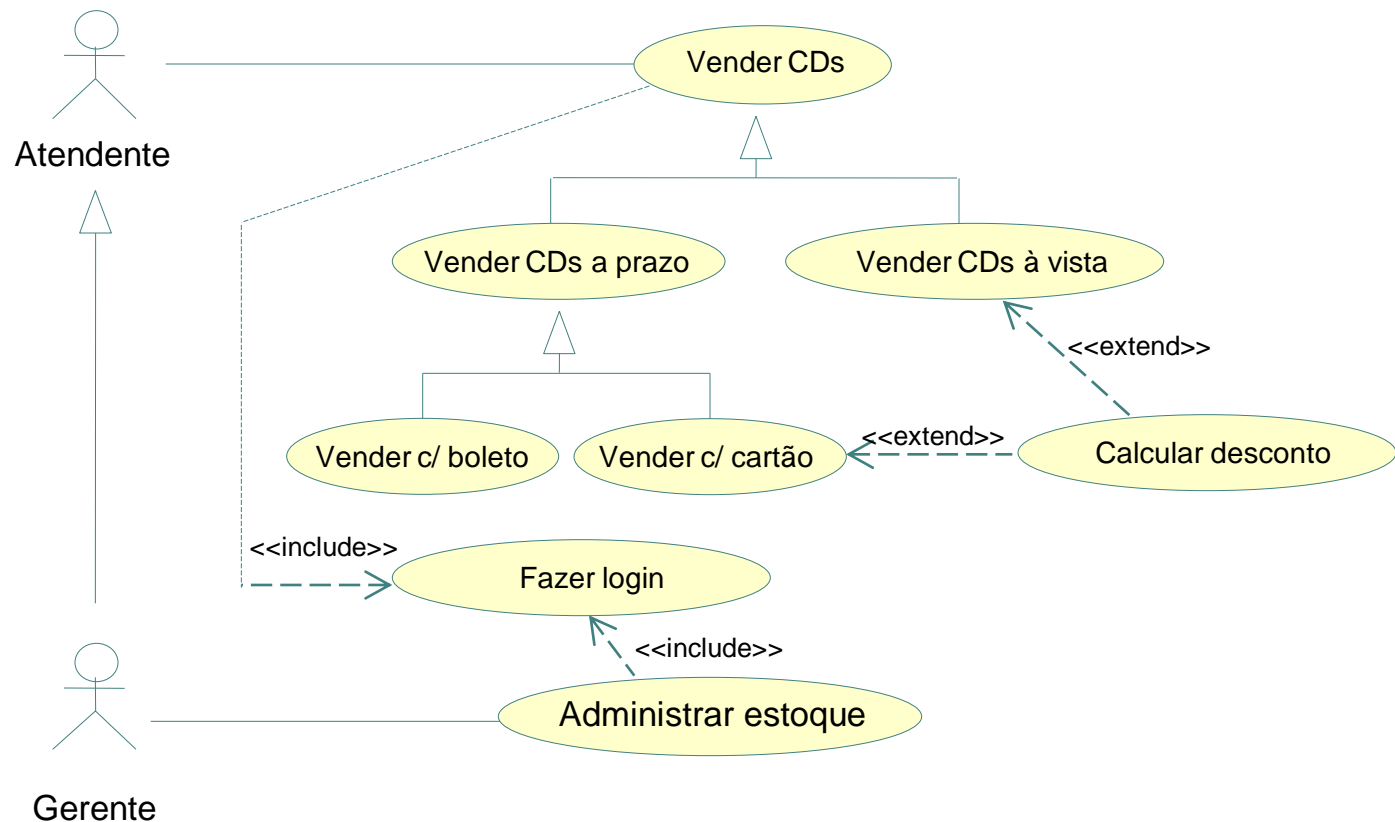
Identificando dependência: inclusão

### Novos requisitos:

- Para efetuar vendas ou administrar estoque, atendentes e gerentes terão que **validar** suas respectivas senhas de acesso ao sistema.

## Exemplo: Loja de CDs

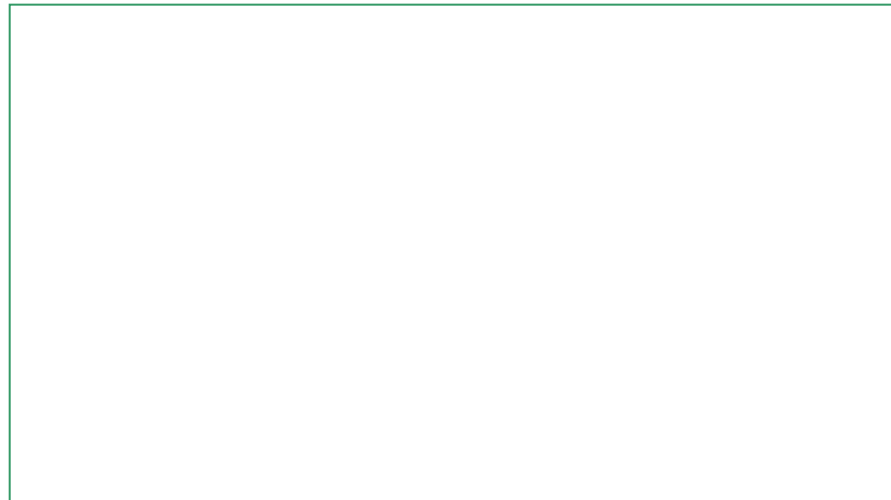
Identificando dependência: inclusão



- Elementos do diagrama
  - Atores
  - Casos de uso
  - Relacionamentos
    - Associação
    - Generalização
    - Dependência: Extensão e Inclusão
  - **Fronteira do sistema**

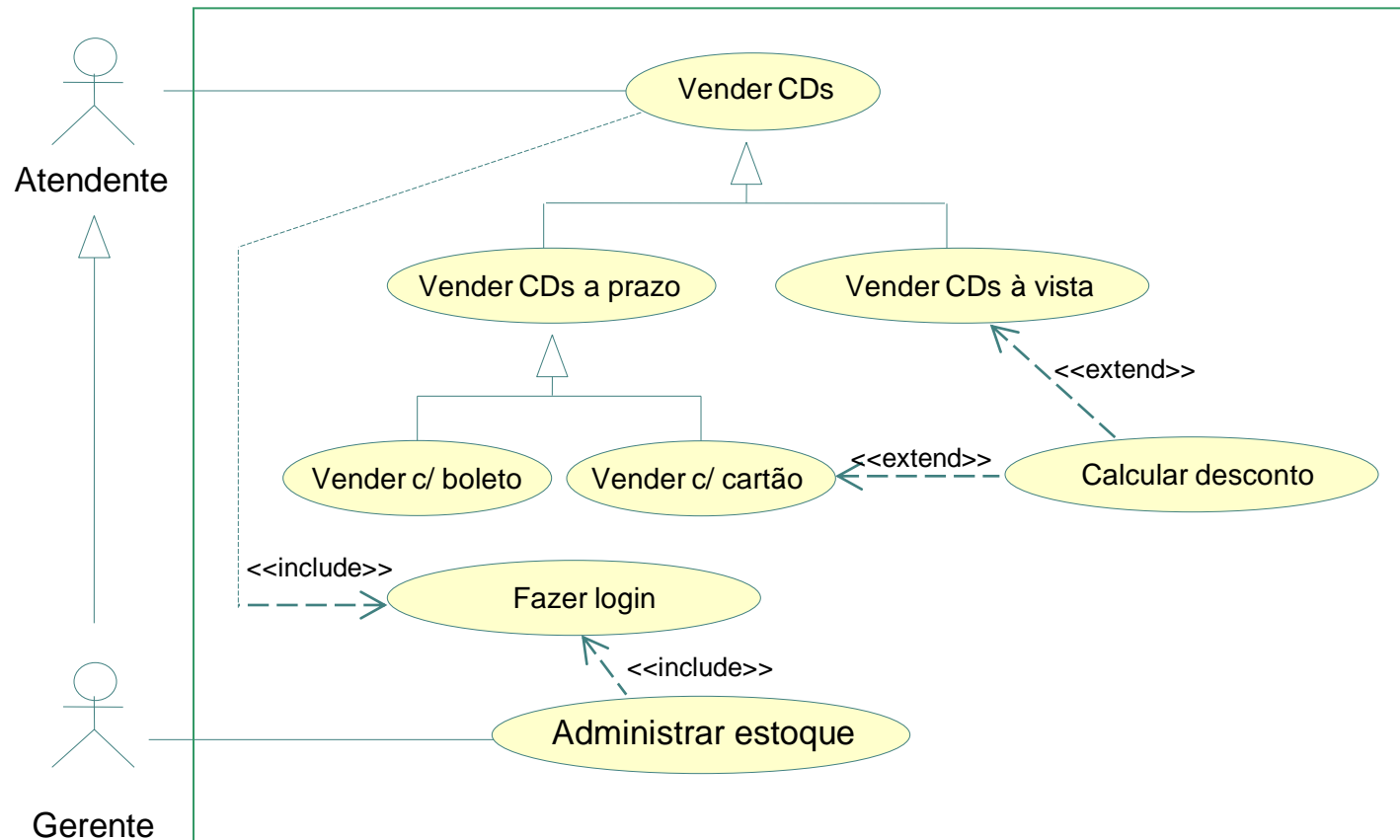
- Fronteira do Sistema
  - Elemento opcional (mas essencial para um bom entendimento)
  - Serve para definir a área de atuação do sistema

Notação:



## Exemplo: Loja de CDs

Identificando a fronteira do sistema



- A descrição é mais importante do que o diagrama
- UML não especifica padrão
- Pode ser:
  - Informal
  - Típica
  - Detalhada

- Descrição Informal
  - Contém o nome do caso de uso e uma descrição textual de sua funcionalidade

## Exemplo:

### **Caso de Uso 01 – Cadastrando Cliente (descrição informal)**

O Cliente inicia o cadastro preenchendo a ficha cadastral e enviando a documentação necessária para o dep. de Cadastro. O Assistente de Cadastro examina a documentação enviada. Estando a documentação em ordem, o Gerente de Cadastro valida os dados da ficha cadastral e marca o cliente como aprovado.

Se houverem problemas com os documento enviados, o Assistente de Cadastro informa documentação irregular. O Cliente envia a documentação regularizada para o Assistente de Cadastro.

Se houverem problemas com os dados da ficha cadastral, o Gerente de Cadastro informa irregularidade dados cadastrais. O Cliente corrige os dados cadastrais.



- Descrição Típica
  - Contém:
    - Identificação do ator que iniciou o caso de uso
    - Pré-requisitos (se houver) do caso de uso
    - Descrição textual do:
      - Fluxo normal
      - Fluxos alternativos (se houver)

## Exemplo:

### Caso de Uso 01 – Cadastrando Cliente (descrição típica)

**Ator Primário:** Cliente

**Precondições:** Nenhuma

#### **Fluxo Normal**

- 1 – Cliente preenche ficha cadastral.
- 2 – Assistente de Cadastro informa recebimento documentação cadastral
- 3 – Gerente de Cadastro informa aprovação de Cliente

**Fluxo Alternativo:** documentação incompleta ou com erro

- 2a – Assistente de Cadastro Informa documentação irregular.
  - 2b – Cliente envia documentação corrigida para cadastro.
- Retorna ao passo 2.

**Fluxo Alternativo:** irregularidade nos dados cadastrais

- 3a – Gerente de Cadastro informa irregularidade dados cadastrais
- 3b – Cliente atualiza dados cadastrais.
- 3c – Retorna ao passo 3.

- Descrição Detalhada (Ex.1)
  - Contém:
    - Identificação do ator que iniciou o caso de uso
    - Objetivo
    - Nível
    - Pré-requisitos (se houver) do caso de uso
    - Condições de disparo (triggers)
    - Descrição textual do:
      - Fluxo normal
      - Fluxos alternativos (se houver)

**Caso de Uso 01 – Cadastrando Cliente (descrição detalhada)****Ator Primário:** Cliente**Objetivo:** Este caso de uso tem por objetivo controlar o processo de cadastro de um novo cliente no Investidor OnLine. Ao final desse caso de uso o cliente estará cadastrado no Sistema de Carteiras, sua documentação estará completa e estará aprovado para operar.**Nível:** Negócio (Summary)**Precondições:** Nenhuma**Condição de disparo (Trigger):** Cliente decide operar através do Investidor OnLine.**Fluxo Normal**

1 – Cliente <u>preenche ficha cadastral</u> , envia documentação para Assistente de Cadastro.	Sistema emite relação de clientes pendentes de documentação para Assistente de Cadastro. Sistema envia aviso de documentação pendente para Cliente
2 – Assistente de Cadastro recebe documentação do Cliente e <u>informa recebimento documentação cadastral</u>	Sistema emite relação de clientes pendentes de aprovação para Gerente de Cadastro. Sistema envia aviso de documentação recebida para Gerente de Cadastro.
3 – Gerente de Cadastro <u>informa aprovação de Cliente</u>	Sistema gera Número da conta. Sistema envia pedido de criação de conta para o Sistema de Carteiras Sistema emite relação de clientes aprovados para Gerente de Cadastro. Sistema envia aviso de aprovação de cadastro para Cliente

**Fluxo Alternativo:** documentação incompleta ou com erro

2a – Assistente de Cadastro <u>informa documentação irregular</u> .	Sistema emite relação de clientes com documentação irregular para Assistente de Cadastro Sistema envia aviso de documentação irregular para Cliente.
2b – Cliente envia documentação corrigida para cadastro. Retoma ao passo 2.	

**Fluxo Alternativo:** irregularidade nos dados cadastrais

3a – Gerente de Cadastro identifica. <u>Informa irregularidade dados cadastrais</u>	Sistema emite relação de clientes com cadastro irregular para Gerente de Cadastro. Sistema envia aviso de irregularidade dados cadastrais para Cliente.
3b – Cliente <u>atualiza dados cadastrais</u> .	Sistema emite relação de clientes pendentes de aprovação para Gerente de Cadastro. Sistema envia aviso de alteração de dados para Gerente de Cadastro.
3c – Retorna ao passo 3	

**Prioridade:** -**Versão:** -**Tempo de Resposta:** -**Frequência de Uso:** 10/dia**Canal para Ator primário:** navegador de internet, sistema de email ou equivalente.**Atores secundários:** Assistente de Cadastro, Gerente de Cadastro, Sistema de Carteiras.**Canal para Atores secundários:** navegador de internet, sistema de email ou equivalente, servidor de fila de mensagens.**Questões em aberto:** -

- Descrição Detalhada (Ex.2)
  - Contém:
    - Nome
    - Descrição sucinta
    - Atores
    - Pré-condições
    - Pós-condições
    - Fluxo básico
    - Fluxos alternativos
    - Fluxos de exceção
    - Estruturas de dados
    - Regras de negócio
    - Observações

## VENDER CDS - CASO DE USO

**NOME**

Vender CDs

**DESCRIÇÃO SUCINTA**

Atendente vende um ou mais CDs a um usuário.

**ATORES**

1. Atendente

**PRÉ-CONDIÇÕES**

1. Ter executado o caso de uso "CDU000\_Validar Senha"

**FLUXO BÁSICO**

1. O Atendente seleciona a opção "Vender CDs".
2. O Sistema exibe a lista de CDs.
3. O Atendente seleciona os CDs, informando as respectivas quantidades.
4. O Sistema exibe a lista de clientes.
5. O Atendente seleciona o cliente.
6. O Atendente seleciona a opção "Vender".
7. O Sistema exibe as informações da venda: CDs, quantidades e o cliente.
8. O Atendente confirma as informações da venda.
9. O Sistema efetua a venda, verificando a regra RN1.
  - 9.1. O Atendente seleciona o tipo de venda "A Prazo" ou "À Vista"
  - 9.2. O Sistema deve executar o caso de uso "CDU001a\_Vender CDs a prazo" ou o caso de uso "CDU001b\_Vender CDs à vista", de acordo com a opção selecionada pelo atendente no passo anterior.
  - 9.3. O Sistema atualiza o estoque de acordo com a regra RN2.
10. O Sistema emite a Nota Fiscal conforme ED1.
11. O caso de uso é encerrado.

**FLUXOS ALTERNATIVOS****(A1) Alternativa ao Passo 4 – Cliente não cadastrado**

- 1.a O Atendente seleciona a opção "Cadastrar Cliente"
- 1.b O Sistema executa o caso de uso "CDU002\_Cadastrar Cliente"
- 1.c O Sistema retoma ao Passo 4.

**(A2) Alternativa ao Passo 8 – Informações Incorretas**

- 2.a O Atendente não confirma as informações da venda.
- 2.b O Sistema retoma ao Passo 2.

**(A3) Alternativa ao Passo 9 – A regra RN1 não é atendida**

- 3.a O Sistema exibe a mensagem "Não há produtos disponíveis em estoque."
- 3.b O caso de uso é encerrado.

**ESTRUTURA DE DADOS****(ED1) Nota Fiscal**

- 1.1. CPF do cliente
- 1.2. Nome do cliente
- 1.3. Endereço do cliente
- 1.4. CNPJ da loja
- 1.5. Razão social da loja
- 1.6. Endereço da loja
- 1.7. Data da compra
- 1.8. Código dos produtos comprados
- 1.9. Descrição dos produtos comprados
- 1.10. Valores dos produtos comprados
- 1.11. Valor total da compra
- 1.12. Valor do desconto
- 1.13. Valor final da compra

**REGRAS DE NEGÓCIO****(RN1)** O produto deve estar disponível em estoque.**(RN2)** O sistema deve atualizar o estoque de produtos, i.e., para cada produto selecionado para venda, o sistema deve subtrair a quantidade vendida da quantidade disponível em estoque.

### VENDER CDS - CASO DE USO

#### NOME

Vender CDs

#### DESCRIÇÃO SUCINTA

Atendente vende um ou mais CDs a um usuário.

#### ATORES

1. Atendente

#### PRÉ-CONDIÇÕES

1. Ter executado o caso de uso "CDU000\_Validar Senha"

#### FLUXO BÁSICO

1. O Atendente seleciona a opção "Vender CDs".
2. O Sistema exibe a lista de CDs.
3. O Atendente seleciona os CDs, informando as respectivas quantidades.
4. O Sistema exibe a lista de clientes.
5. O Atendente seleciona o cliente.
6. O Atendente seleciona a opção "Vender".
7. O Sistema exibe as informações da venda: CDs, quantidades e o cliente.
8. O Atendente confirma as informações da venda.
9. O Sistema efetua a venda, verificando a regra RN1.
  - 9.1. O Atendente seleciona o tipo de venda "À Prazo" ou "À Vista"
  - 9.2. O Sistema deve executar o caso de uso "CDU001a\_Vender CDs a prazo" ou o caso de uso "CDU001b\_Vender CDs à vista", de acordo com a opção selecionada pelo atendente no passo anterior.
  - 9.3. O Sistema atualiza o estoque de acordo com a regra RN2.
10. O Sistema emite a Nota Fiscal conforme ED1.
11. O caso de uso é encerrado.

### ESTRUTURA DE DADOS

#### (ED1) Nota Fiscal

- 1.1. CPF do cliente
- 1.2. Nome do cliente
- 1.3. Endereço do cliente
- 1.4. CNPJ da loja
- 1.5. Razão social da loja
- 1.6. Endereço da loja
- 1.7. Data da compra
- 1.8. Código dos produtos comprados
- 1.9. Descrição dos produtos comprados
- 1.10. Valores dos produtos comprados
- 1.11. Valor total da compra
- 1.12. Valor do desconto
- 1.13. Valor final da compra

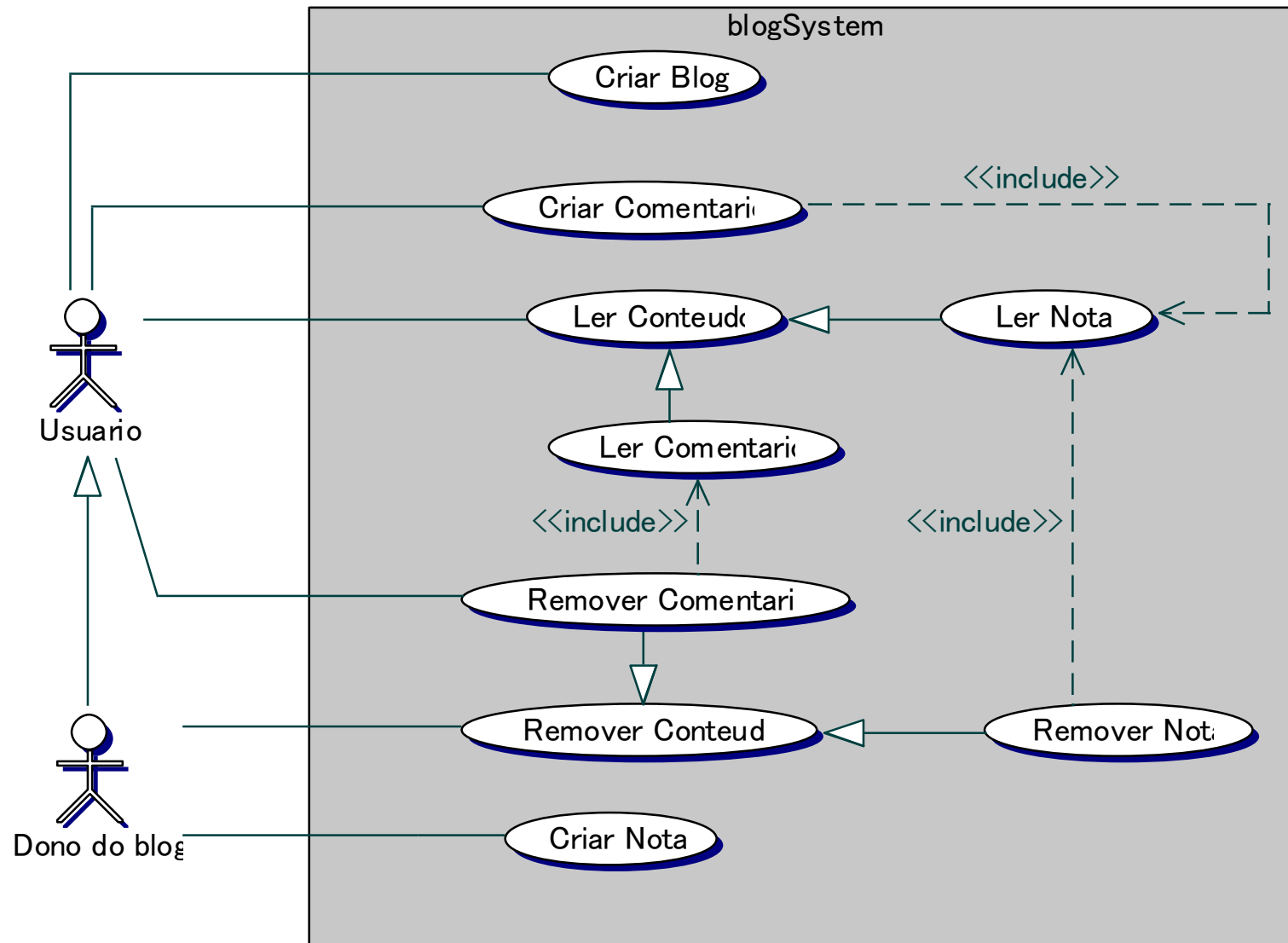
### REGRAS DE NEGÓCIO

(RN1) O produto deve estar disponível em estoque.

(RN2) O sistema deve atualizar o estoque de produtos, i.e., para cada produto selecionado para venda, o sistema deve subtrair a quantidade vendida da quantidade disponível em estoque.

- Um *blog* é uma ferramenta de colaboração
- Um *blog* é formado por um conjunto de conteúdos:
  - notas
  - comentários sobre as notas
- Os conteúdos possuem as seguintes informações: texto, data de criação e autor
- Os usuários de um blog podem ser:
  - **Usuário:** pode ler conteúdos de um blog, comentar uma nota, remover comentários, e pode criar um blog.
  - **Dono do blog:** além de todas as funcionalidades de um usuário comum, o dono do blog pode criar notas e remover notas
- Para remover um conteúdo o usuário precisa ler o conteúdo antes

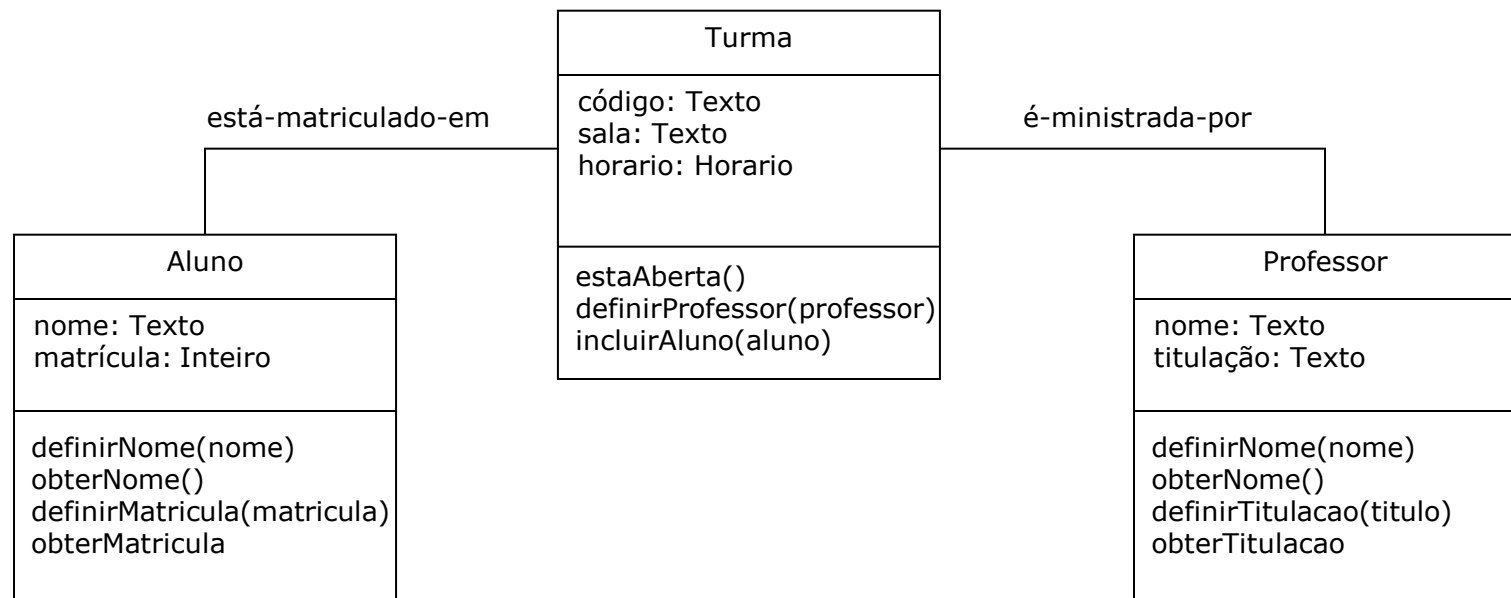




# **UML: Diagrama de Classes**

Projeto de Sistemas de Software

- Mostra um conjunto de classes e seus relacionamentos.
  - Modelo Conceitual
  - Modelo de Domínio
- É o diagrama central da modelagem orientada a objetos.

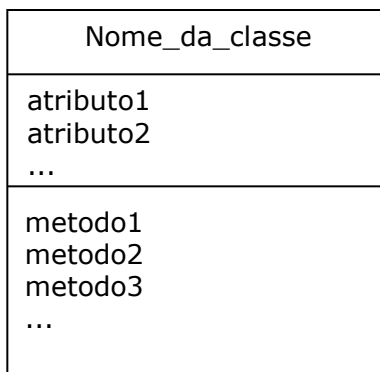


- Elementos de um diagrama de classes
  - Classes
  - Relacionamentos
    - Associação
      - Agregação
      - Composição
    - Generalização
    - Dependência

- Elementos de um diagrama de classes
  - **Classes**
  - Relacionamentos
    - Associação
      - Agregação
      - Composição
    - Generalização
    - Dependência

## Classes

- Graficamente, as classes são representadas por retângulos incluindo nome, atributos e métodos.



- Devem receber nomes de acordo com o vocabulário do domínio do problema.
- É comum adotar um padrão para nomeá-las  
**Ex:** todos os nomes de classes serão substantivos singulares com a primeira letra maiúscula

## Classes

- Atributos
  - Representam o conjunto de características (estado) dos objetos daquela classe
  - Visibilidade:
    - + público: visível em qualquer classe de qualquer pacote
    - # protegido: visível para classes do mesmo pacote
    - privado: visível somente para classe

### Exemplo:

+ nome : String

## Classes

- Métodos
  - Representam o conjunto de operações (comportamento) que a classe fornece
  - Visibilidade:
    - + público: visível em qualquer classe de qualquer pacote
    - # protegido: visível para classes do mesmo pacote
    - privado: visível somente para classe

### Exemplo:

- getNome() : String

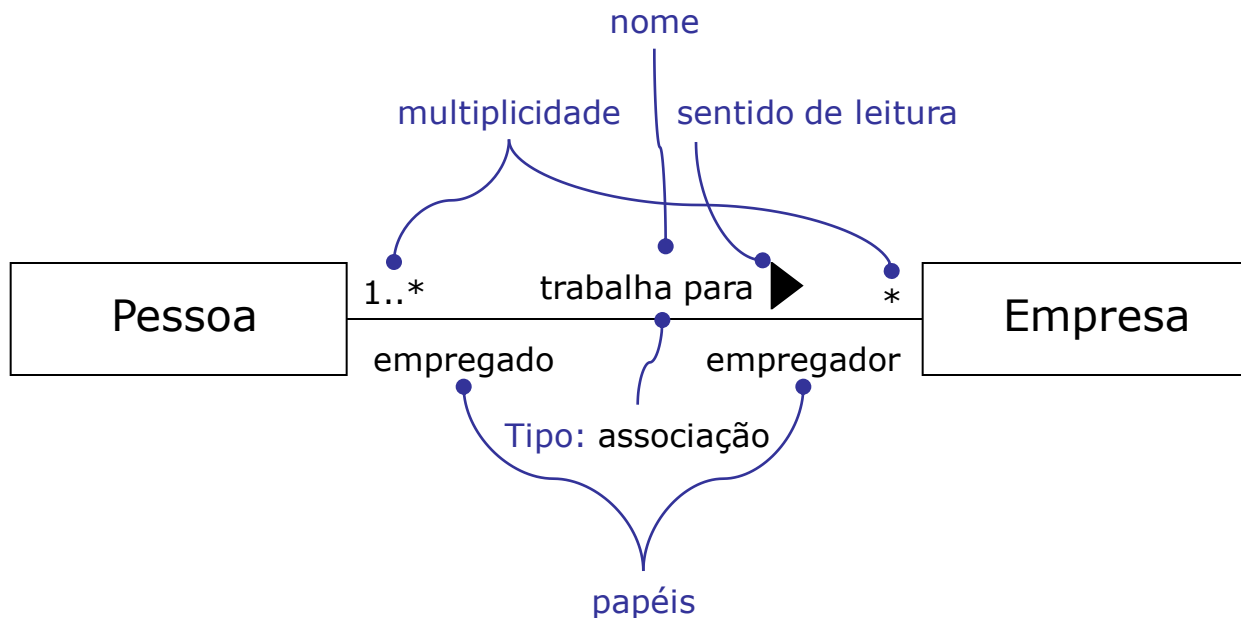


- Elementos de um diagrama de classes
  - Classes
  - **Relacionamentos**
    - Associação
      - Agregação
      - Composição
    - Generalização
    - Dependência

## Relacionamentos

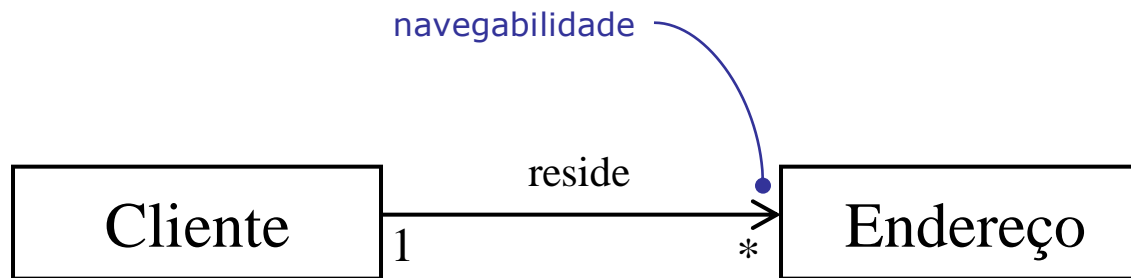
- Os relacionamentos possuem:
  - **Nome:** descrição dada ao relacionamento (faz, tem, possui,...)
  - **Sentido de leitura**
  - **Navegabilidade:** indicada por uma seta no fim do relacionamento
  - **Multiplicidade:** 0..1, 0..\*, 1, 1..\*, 2, 3..7
  - **Tipo:** associação (agregação, composição), generalização e dependência
  - **Papéis:** desempenhados por classes em um relacionamento

- Relacionamentos



E a navegabilidade?

- Relacionamentos

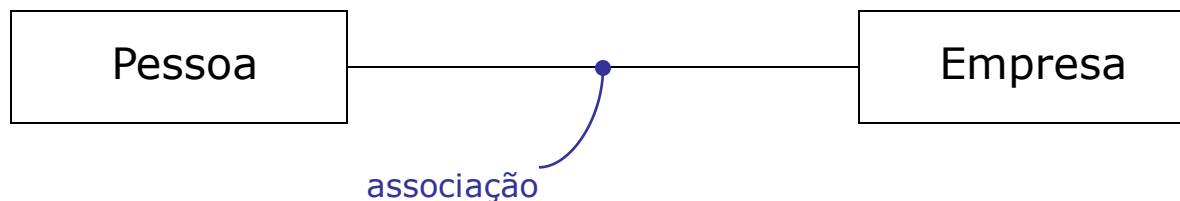


- O cliente sabe quais so seus endereos, mas o endereo no sabe a quais clientes pertence

- Elementos de um diagrama de classes
  - Classes
  - **Relacionamentos**
    - Associação
      - Agregação
      - Composição
    - Generalização
    - Dependência

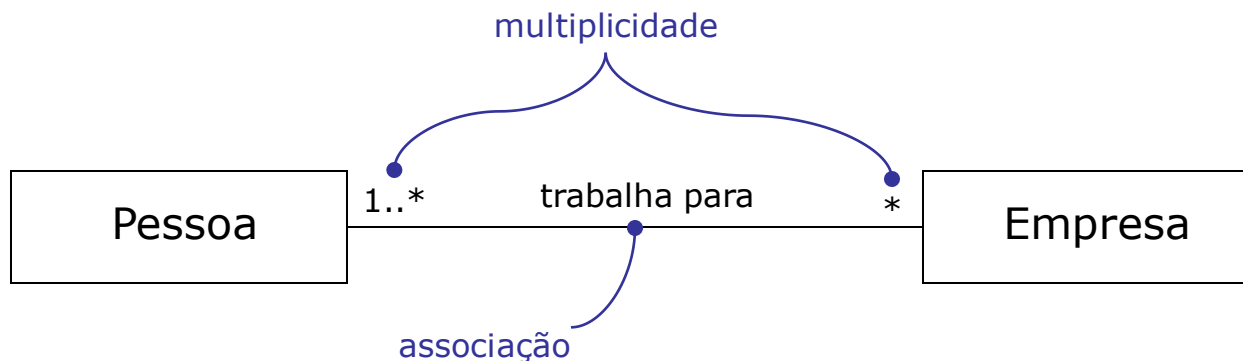
## Relacionamentos: Associação

- Uma **associação** é um relacionamento estrutural que indica que os objetos de uma classe estão vinculados a objetos de outra classe.
- Uma associação é representada por uma linha sólida conectando duas classes.



## Relacionamentos: Associação

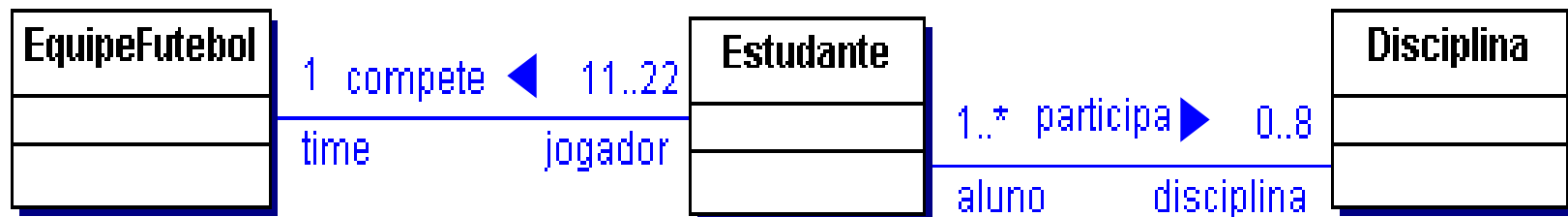
- Indicadores de multiplicidade:
  - 1 Exatamente um
  - 1..\* Um ou mais
  - 0..\* Zero ou mais (muitos)
  - \* Zero ou mais (muitos)
  - 0..1 Zero ou um
  - m..n Faixa de valores (por exemplo: 4..7)



## Relacionamentos: Associação

### Exemplo:

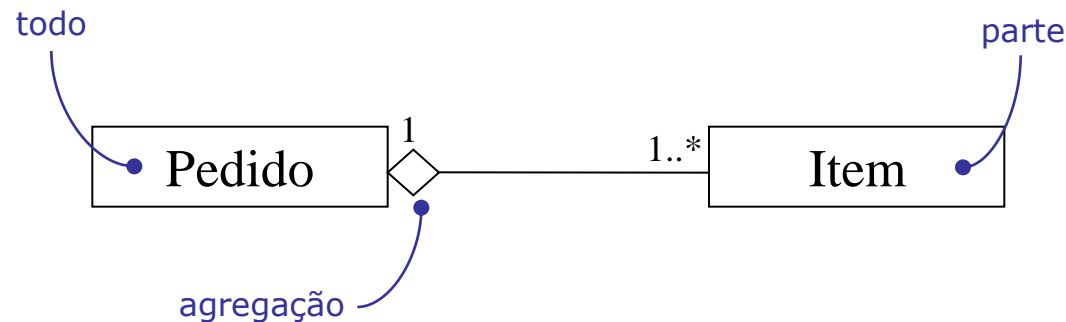
- Um **Estudante** pode ser  
um **aluno** de uma Disciplina e  
um **jogador** da Equipe de Futebol
- Cada Disciplina deve ser cursada por no mínimo 1 aluno
- Um aluno pode cursar de 0 até 8 disciplinas





- Elementos de um diagrama de classes
  - Classes
  - **Relacionamentos**
    - Associação
      - **Agregação**
      - Composição
    - Generalização
    - Dependência

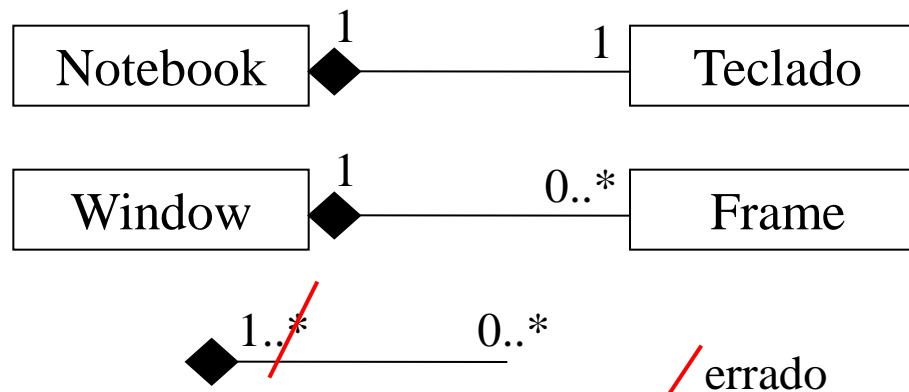
- Relacionamento: Agregação
  - É um tipo especial de associação
  - Utilizada para indicar “todo-parte”



- um objeto “parte” pode fazer parte de vários objetos “todo”

- Elementos de um diagrama de classes
  - Classes
  - **Relacionamentos**
    - Associação
      - Agregação
      - **Composição**
    - Generalização
    - Dependência

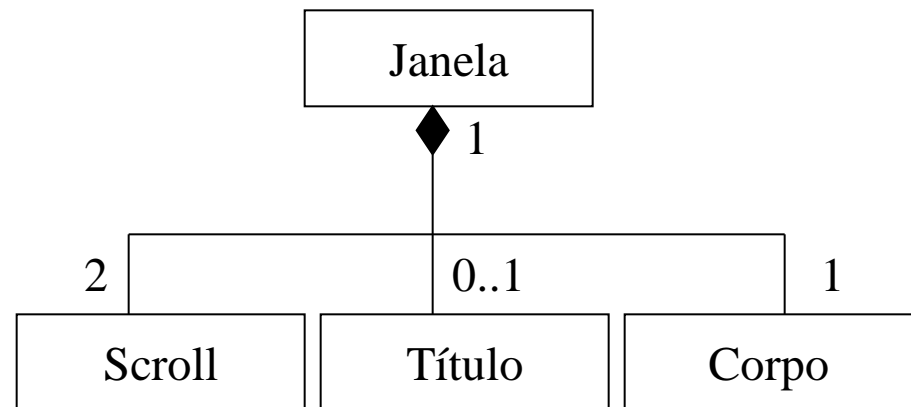
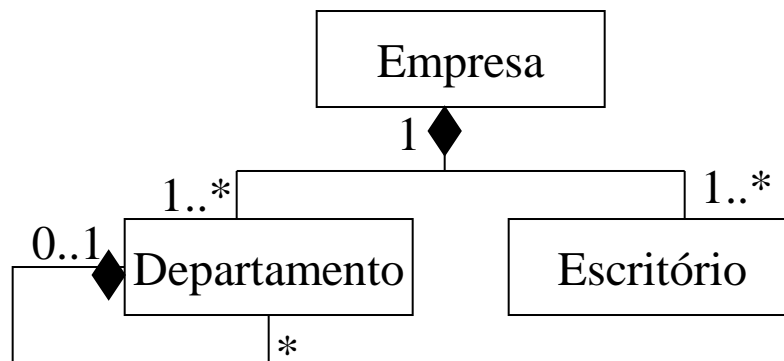
- Relacionamento: Composição
  - É uma variante semanticamente mais “forte” da agregação
  - Os objetos “parte” só podem pertencer a um único objeto “todo” e têm o seu tempo de vida coincidente com o dele



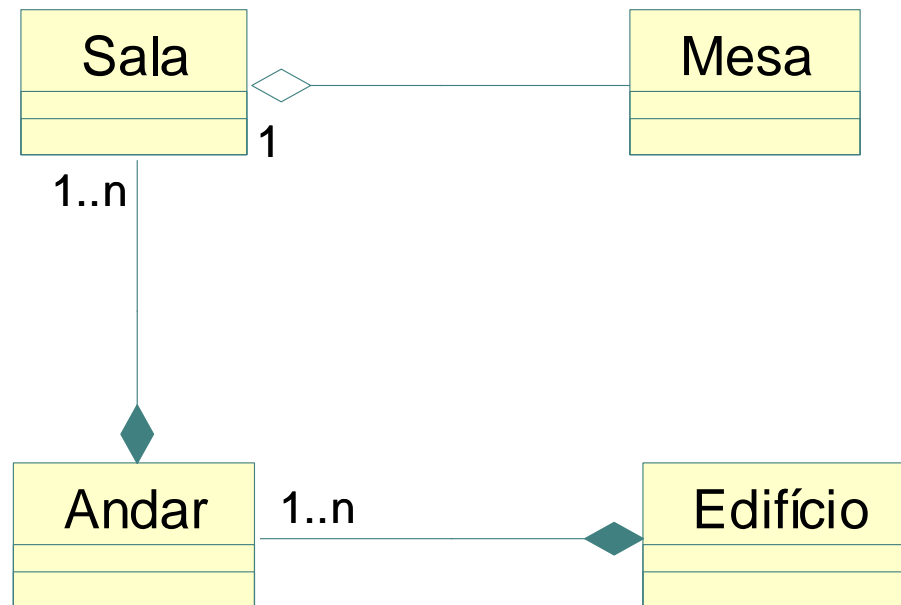
- Quando o “todo” *morre* todas as suas “partes” também *morrem*

- Relacionamento: Composição

Ex:

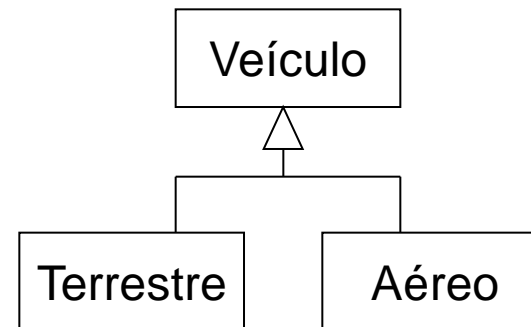
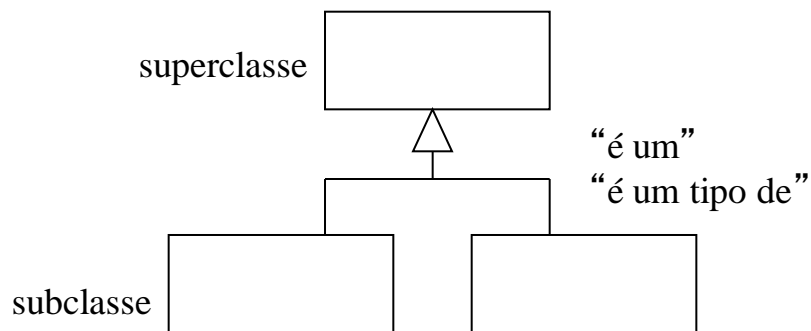


- Agregação X Composição



- Elementos de um diagrama de classes
  - Classes
  - **Relacionamentos**
    - Associação
      - Agregação
      - Composição
    - Generalização
    - Dependência

- Relacionamento: Generalização
  - É um relacionamento entre itens gerais (superclasses) e itens mais específicos (subclasses)

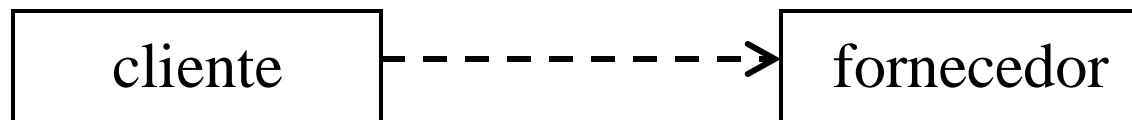




- Elementos de um diagrama de classes
  - Classes
  - **Relacionamentos**
    - Associação
      - Agregação
      - Composição
    - Generalização
    - Dependência

- Relacionamento: Dependência
  - Representa que a alteração de um objeto (o objeto independente) pode afetar outro objeto (o objeto dependente)

Ex:



Obs:

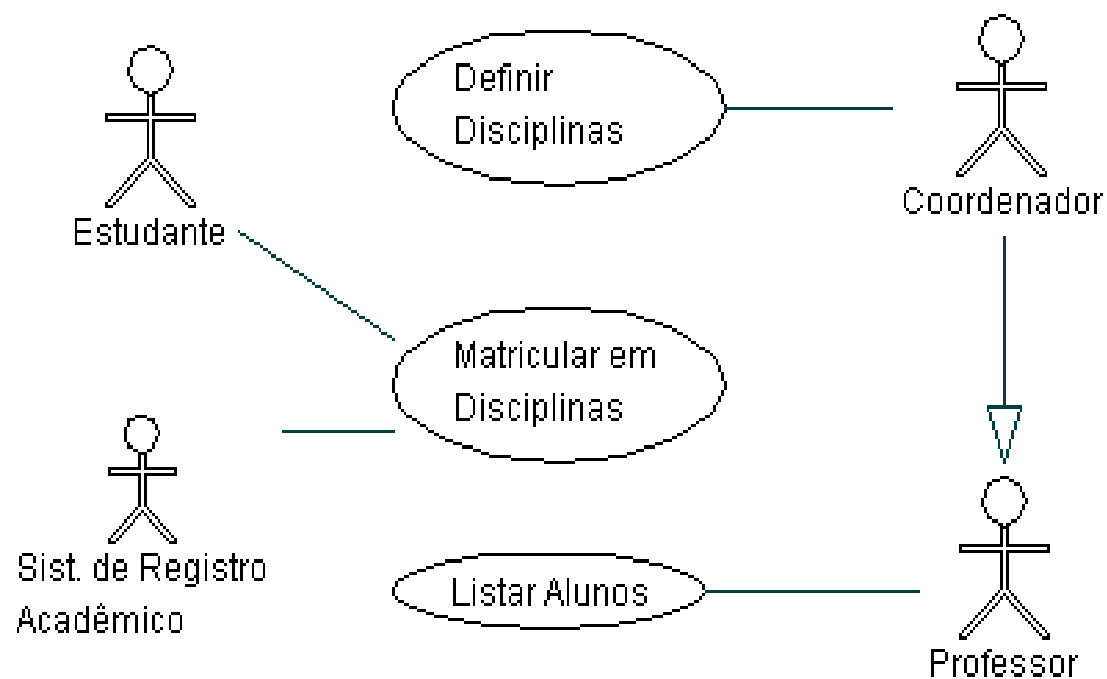
- A classe cliente depende de algum serviço da classe fornecedor
- A mudança de estado do fornecedor afeta o objeto cliente
- A classe cliente não declara nos seus atributos um objeto do tipo fornecedor
- Fornecedor é recebido por parâmetro de método

## Descrição

A Universidade XYZ deseja informatizar seu sistema de matrículas:

- A universidade oferece vários cursos.
- O **Coordenador** de um curso define as disciplinas que serão oferecidas pelo seu curso num dado semestre.
- Várias disciplinas são oferecidas em um curso.
- Várias turmas podem ser abertas para uma mesma disciplina, porém o número de estudantes inscritos deve ser entre 3 e 10.
- **Estudantes** selecionam 4 disciplinas.
- Quando um estudante matricula-se para um semestre, o **Sistema de Registro Acadêmico (SRA)** é notificado.
- Após a matrícula, os estudantes podem, por um certo prazo, utilizar o sistema para adicionar ou remover disciplinas.
- **Professores** usam o sistema para obter a lista de alunos matriculados em suas disciplinas.
- Todos os usuários do sistema devem ser validados.

## Diagrama de Casos de Uso



## Descrição do Caso de Uso “Matricular em Disciplina”

- Esse caso de uso se inicia quando o Estudante de Curso inicia uma sessão no sistema e apresenta suas credenciais.
- O sistema verifica se a credencial é válida.
- O sistema solicita que o estudante realize sua matrícula, selecionando 4 disciplinas.
- O estudante preenche um formulário eletrônico de matrícula e o submete para uma análise de consistência.
- O sistema analisa as informações contidas no formulário.
  - Se as informações são consistentes, o estudante é incluído em turmas abertas de 4 disciplinas, iniciando pelas preferenciais.
  - Se as informações não são consistentes, o sistema informa o motivo da inconsistência e solicita que o formulário seja alterado.

Diagrama de Classes: identificando as classes

Professor

Coordenador

Estudante

Universidade

Disciplina

Turma

Curso

FormularioMatricula

AnalizadorMatricula

SistemaRegistroAcademico

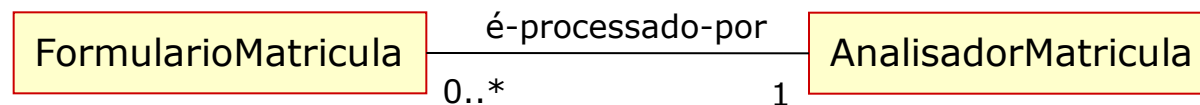
ListaAlunos

## Diagrama de Classes: identificando os relacionamentos

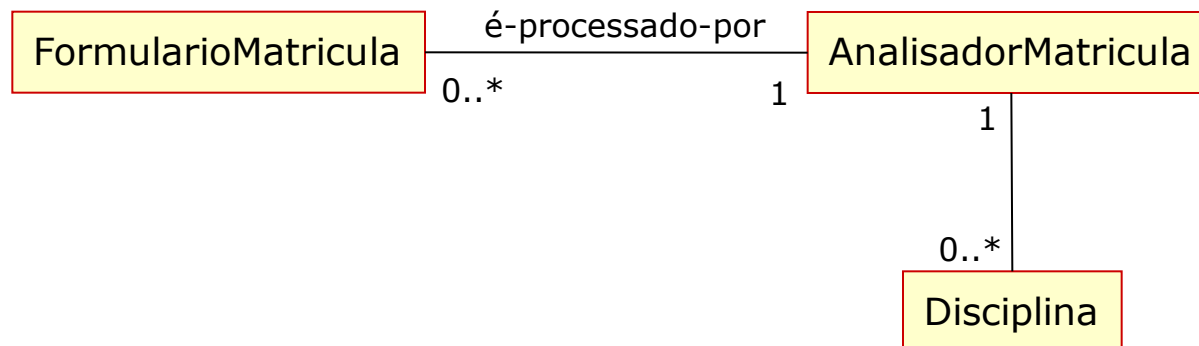
- Exemplos de candidatos a relacionamentos:
  - **A** é parte física ou lógica de **B**.
  - **A** está contido fisicamente ou logicamente em **B**.
  - **A** é uma descrição de **B**.
  - **A** é membro de **B**.
  - **A** é subunidade organizacional de **B**.
  - **A** usa ou gerencia **B**.
  - **A** se comunica/interage com **B**.
  - **A** está relacionado com uma transação **B**.
  - **A** é possuído por **B**.
  - **A** é um tipo de **B**.

Diagrama de Classes: identificando os relacionamentos

- O formulário de matrícula é processado por um analisador de matrícula

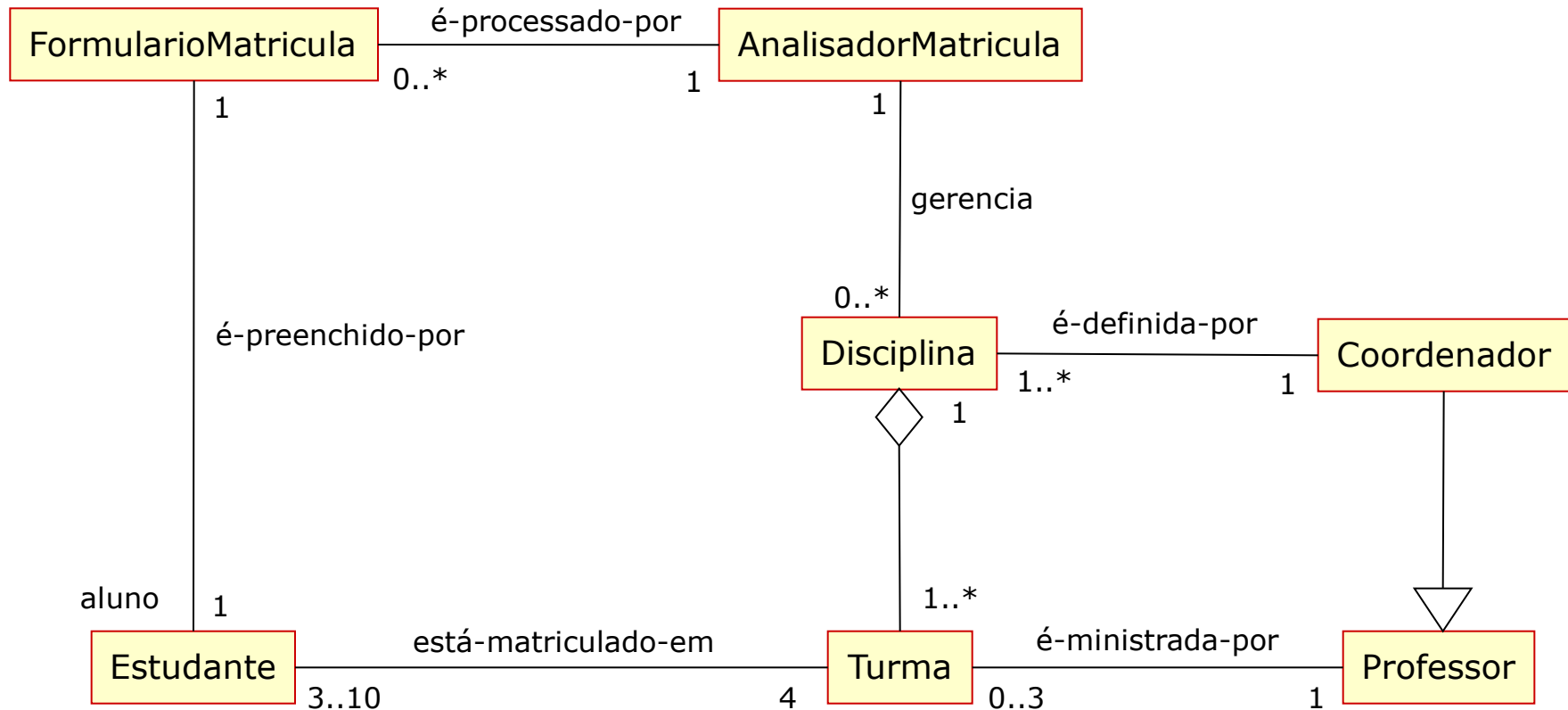


- O analisador de matrícula gerencia a disciplina





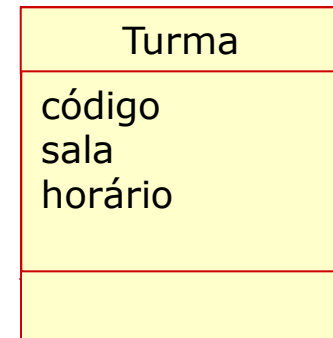
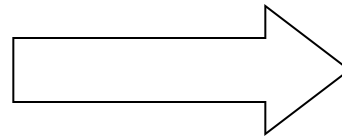
## Diagrama de Classes



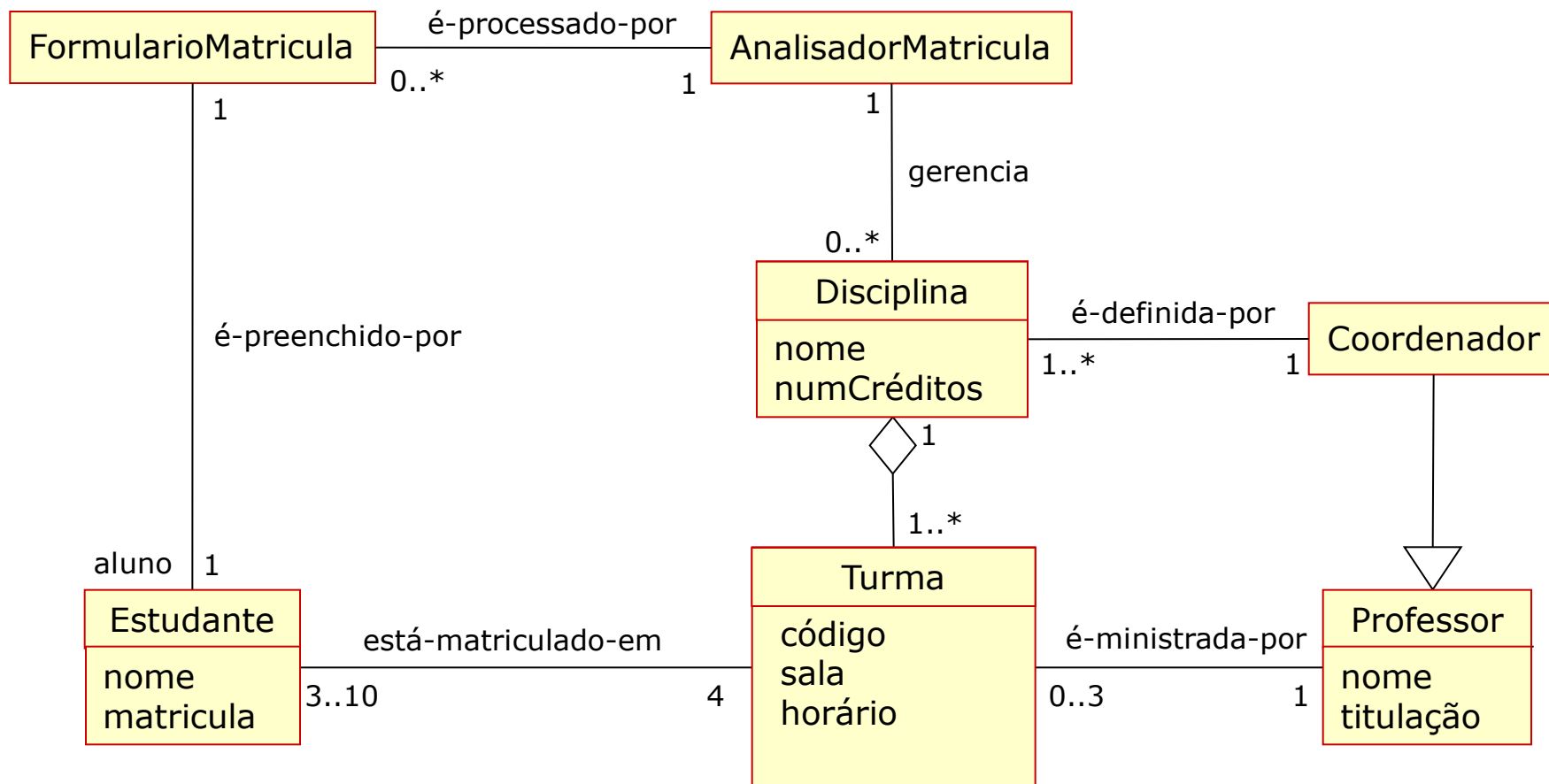
## Diagrama de Classes: identificando os atributos

- Os atributos podem ser encontrados examinando-se as descrições dos casos de uso e também pelo conhecimento do domínio do problema.

- Cada turma oferecida possui um código, uma sala e um horário.

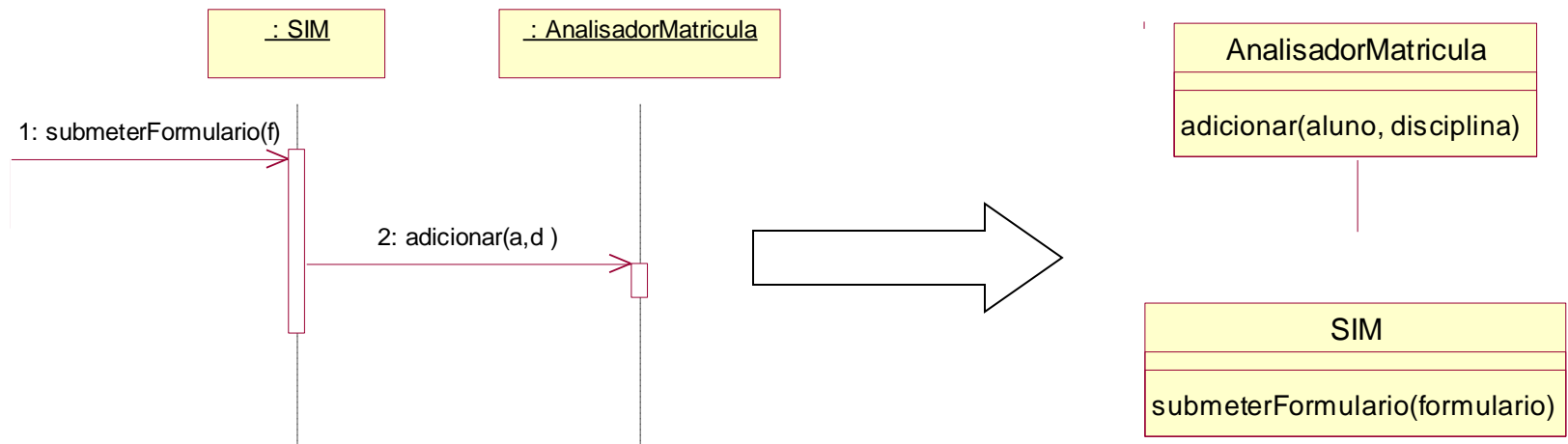


## Diagrama de Classes



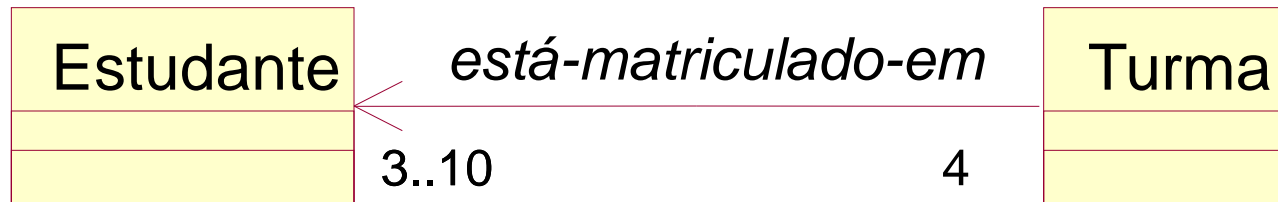
## Diagrama de Classes: identificando os métodos

- Somente depois de modelar os diagramas de seqüência



## Diagrama de Classes:

- E a navegabilidade?



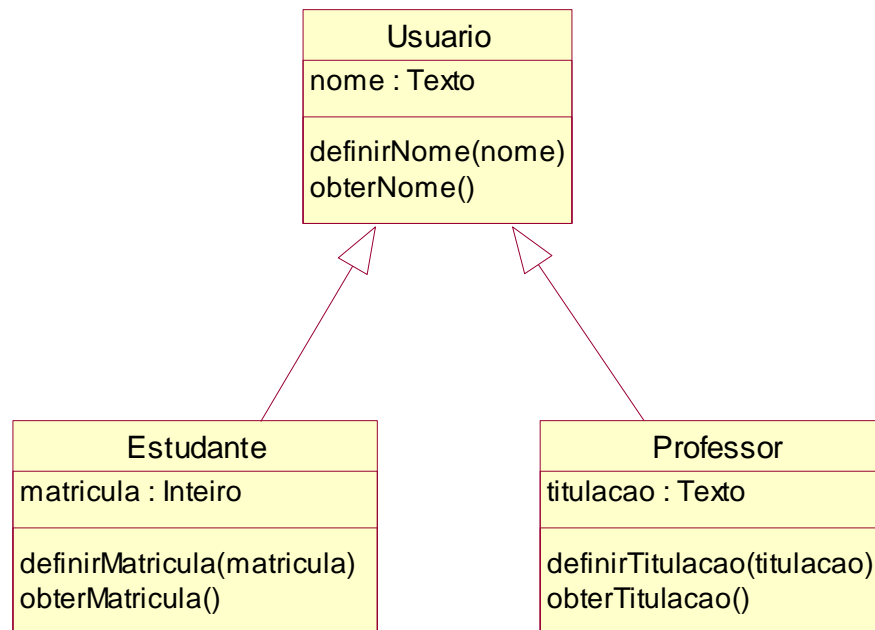
```
public class Estudante {
    private String nome;
    private String matricula;
    ...
}
```

```
public class Turma {
    private String codigo;
    private String sala;
    private Estudante alunos[];
    ...
}
```

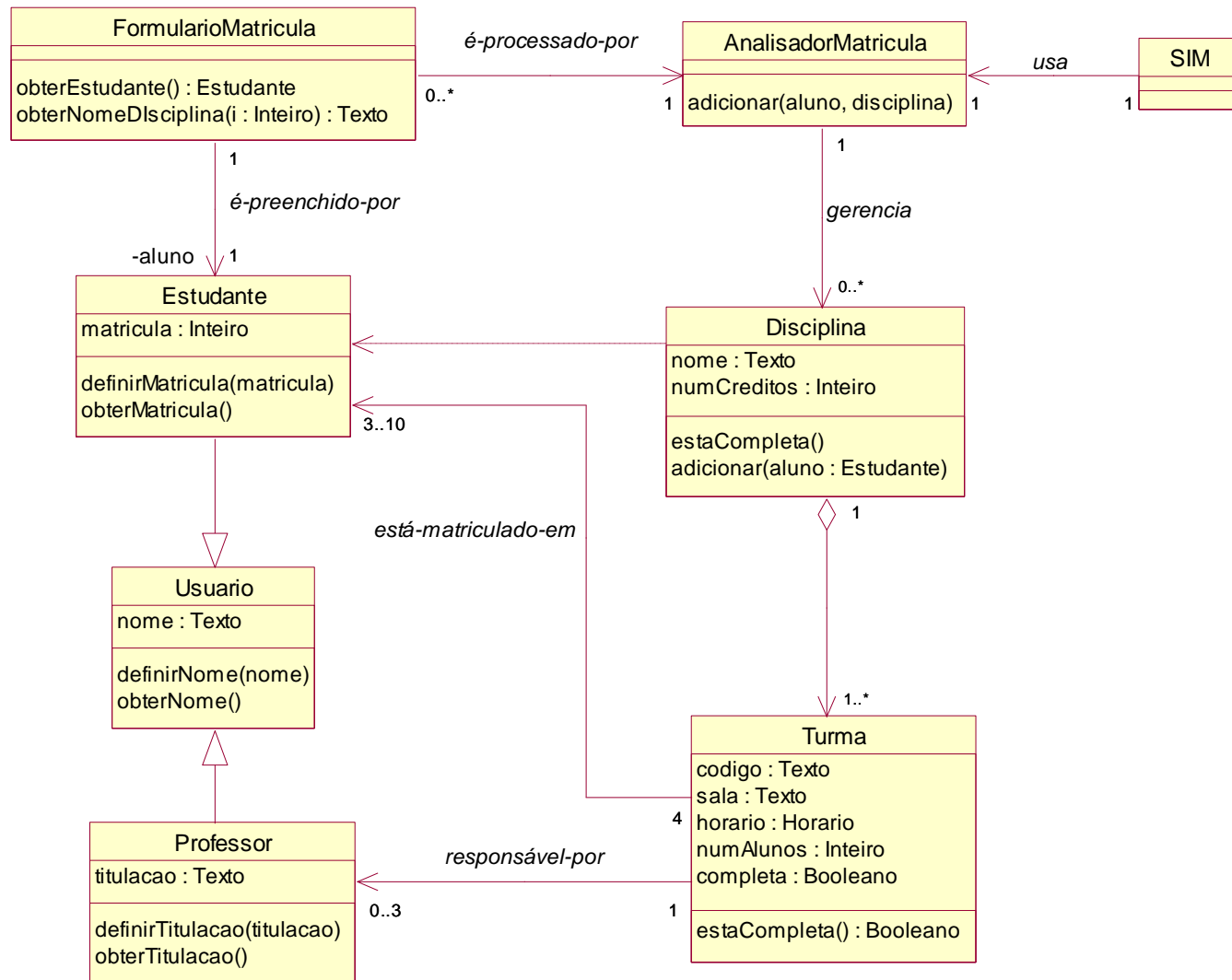
OBS: Turma não aparece como atributo de Estudante!

## Diagrama de Classes:

- Acrescentando generalizações:
  - Atributos, operações e/ou relacionamentos comuns podem ser movidos para uma classe mais geral.



# Exemplo: Sistema de Matrícula

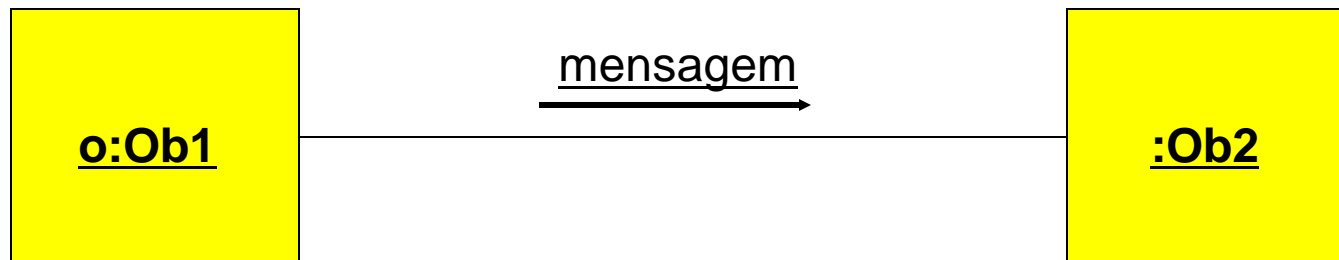


# **Diagramas de Seqüência**

Projeto de Sistemas de Software



- Comportamento que
  - Envolve conjunto de mensagens trocadas entre objetos dentro de um determinado contexto
  - Objetiva atingir resultado específico
- Acontecem em função da troca de mensagens entre objetos
- Usadas para a modelagem dos aspectos dinâmicos de um sistema



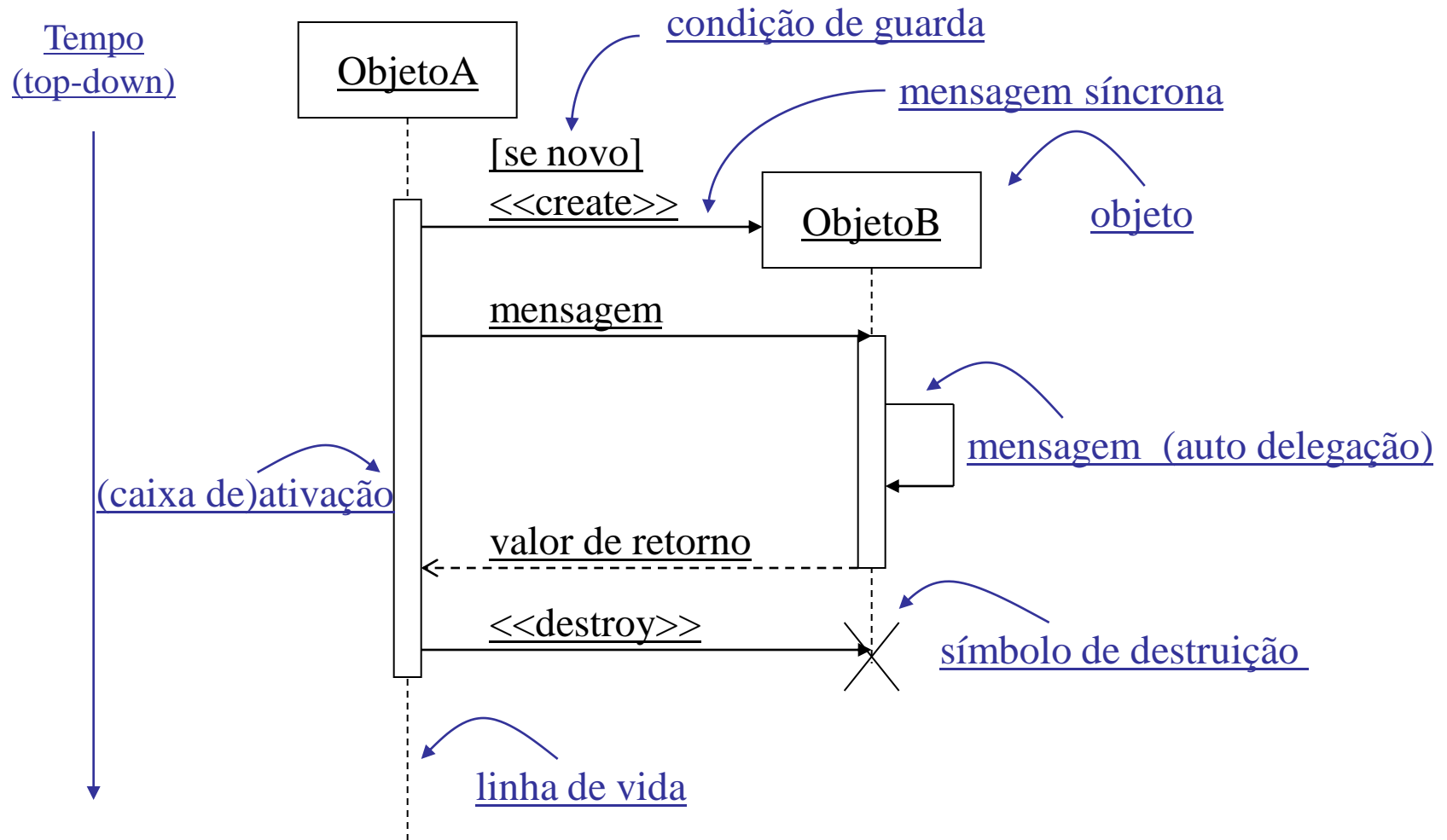
Mensagem =

<u>Ident. Objeto</u>	<u>Ident. Operação</u>	<u>Parâmetros</u>
----------------------	------------------------	-------------------

- Mensagem
  - Recepção de mensagem por um objeto
    - Considerado instância de evento
  - Decorrência da passagem de uma mensagem
    - Repercute ação representada por um comando executável
    - Comando Executável: abstração de procedimento computacional

- Deseja-se **representar o comportamento** de vários objetos
  - Dentro de um único caso de uso
  - A partir das **mensagens** que são passadas entre eles
- Objetivo
  - Definir um **contexto** de caso de uso
  - Estabelecer os **objetos** que interagem e seus **relacionamentos**
- Termo genérico que se aplica a dois tipos de diagramas que enfatizam interações entre objetos
  - **Diagrama de Seqüência**
  - **Diagrama de Colaboração**

- Informações bastante similares mas de maneira diferente
  - Diagrama de Seqüência
    - Interação enfatizando o **tempo de seqüência**
    - Mostra objetos participando em interações de acordo com suas linhas de vida e as mensagens que trocam
  - Diagrama de Colaboração
    - Interação enfatizando o **relacionamento** entre os objetos



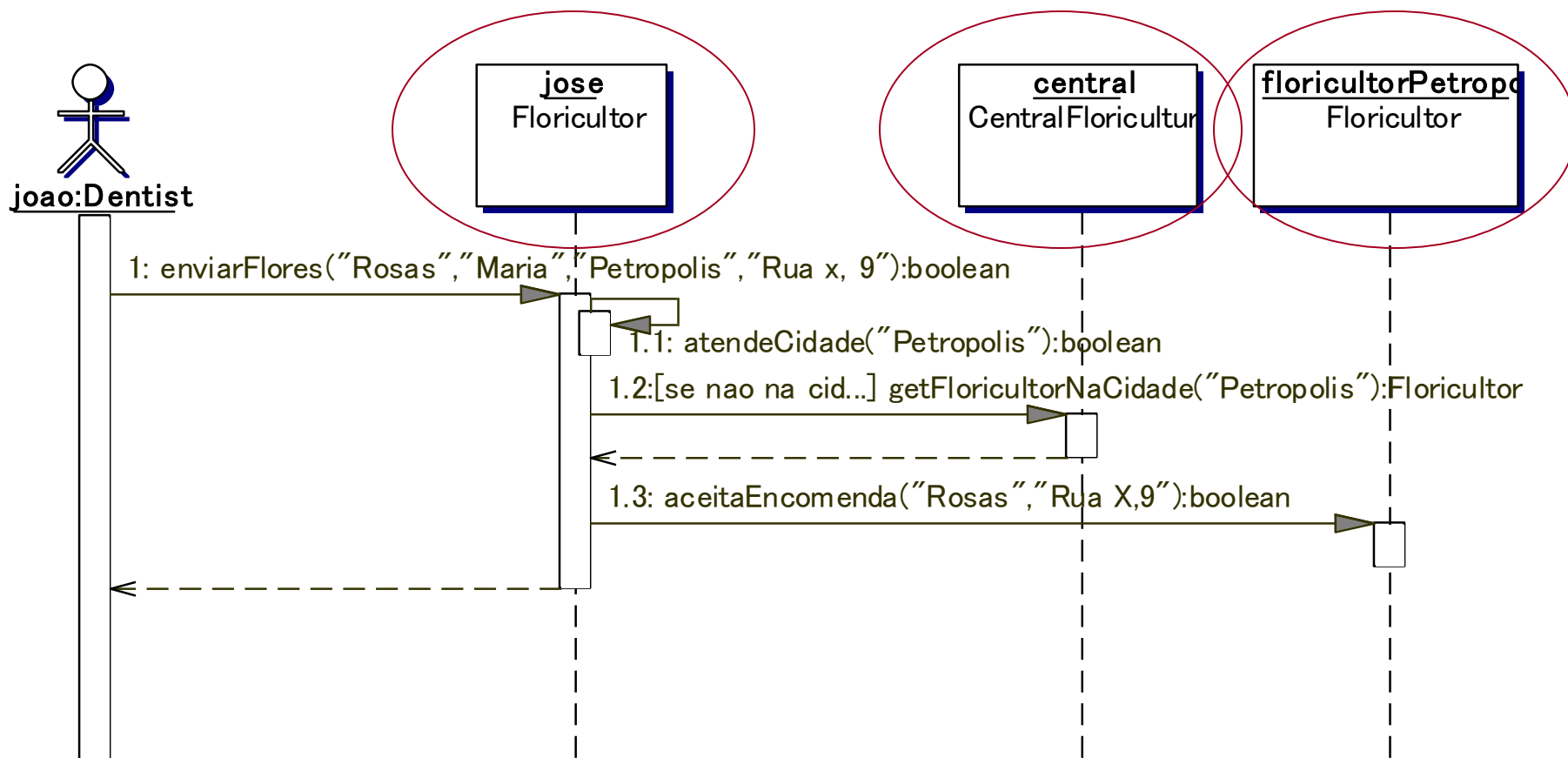
- Objetos
- Linhas de vida
- Mensagens
- Focos de controle

- Apresentados na **dimensão horizontal** do diagrama
- **Ordem** dos objetos não é considerada
  - Dispô-los de forma a tornar o diagrama “mais legível”
- Objetos tem nomes
  - **obj:Classe**

Ex.: joão:Dentista

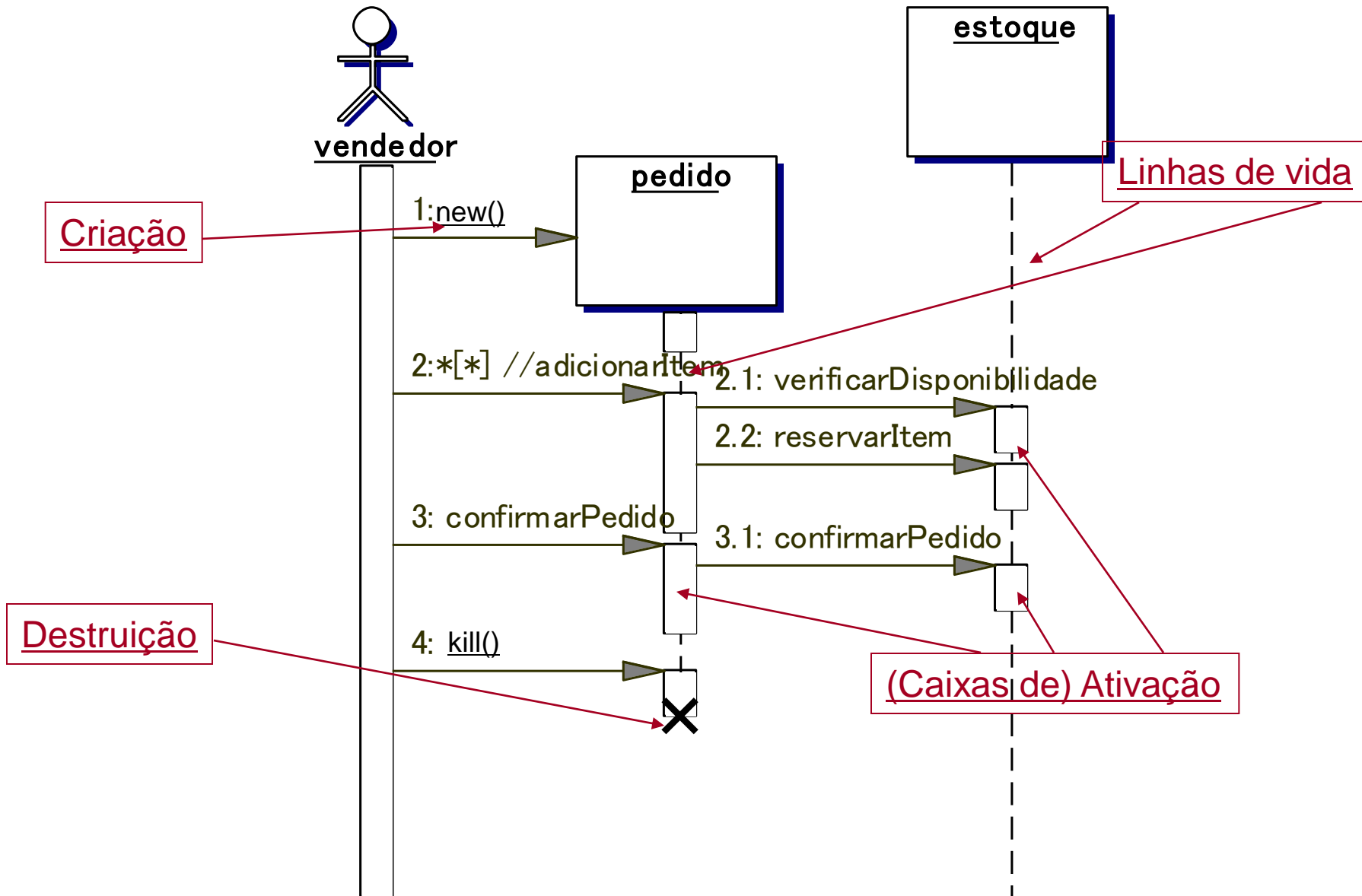
:Floricultor (um objeto floricultor não identificado)

obj1: (um objeto obj1 sem classe definida)





- **Dimensão vertical** do diagrama
- Apresentam o **tempo de vida** dos objetos
- Pode apresentar a **ativação** ou a **desativação** dos objetos
  - Indicam que os objetos estão executando algo
    - Foco de controle
  - Caixas de ativação podem ser empilhadas
    - Indica chamada de método do próprio objeto
    - Objeto jose no slide anterior
- Podem representar a **criação** e a **destruição** de objetos

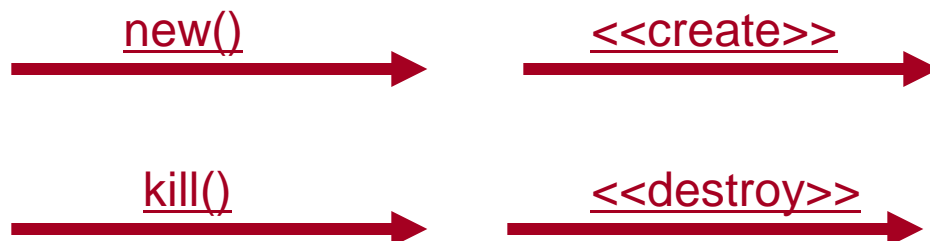





- Objetos interagem através da troca de mensagens
  - Setas sólidas que vão do objeto solicitante para o solicitado
    - Para o próprio objeto: auto-delegação
  - Rotulados com os nomes dos estímulos mais os argumentos (ou valores dos argumentos) do estímulo
- Sintaxe

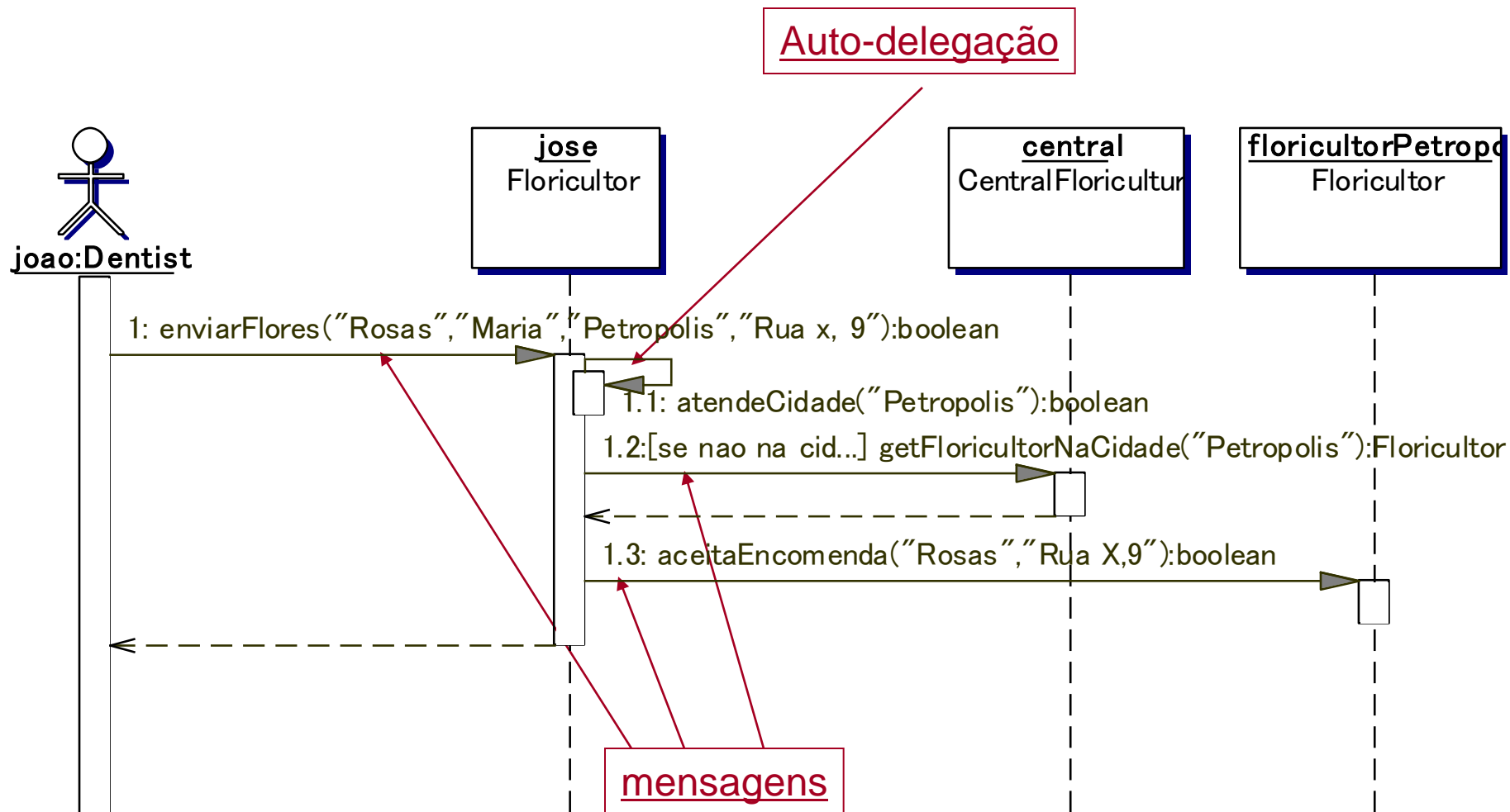
return := message(parameter:parameterType):returnType

- onde
  - **return** é o nome do valor de retorno
  - **message** é o nome da mensagem
  - **parameter** é o nome de um parâmetro da mensagem
  - **parameterType** é o nome do tipo desse parâmetro
  - **returnType** é o tipo do valor de retorno

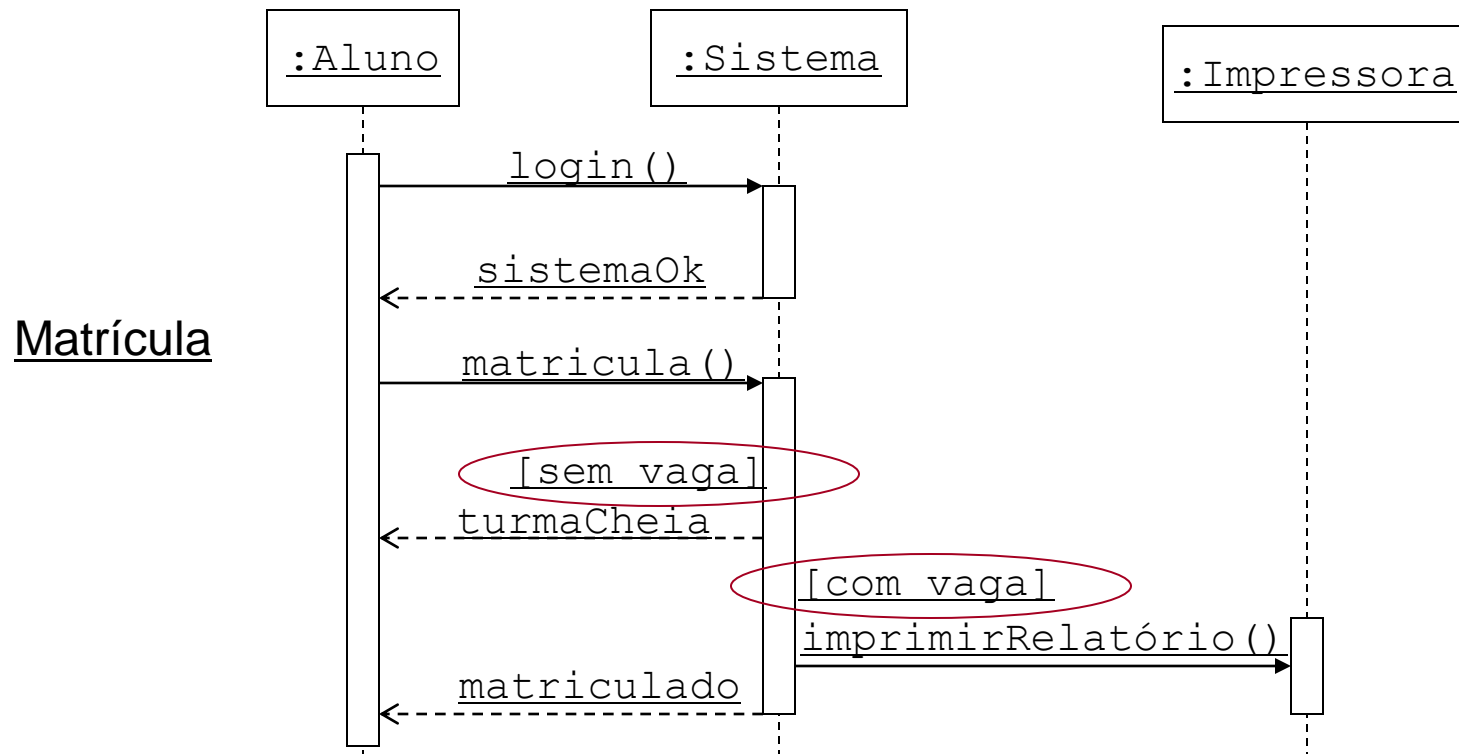
- Tipos de ação que uma mensagem pode representar
  - call
    - Invoca uma operação sobre um objeto
      - Objeto pode mandar uma chamada para si próprio
        - » Resultando na execução local de uma operação
  - return
    - Representa o retorno de um valor para o objeto que chamou a operação
    - Opcional
  - create
    - Criação de um objeto
  - destroy
    - Eliminação de um objeto



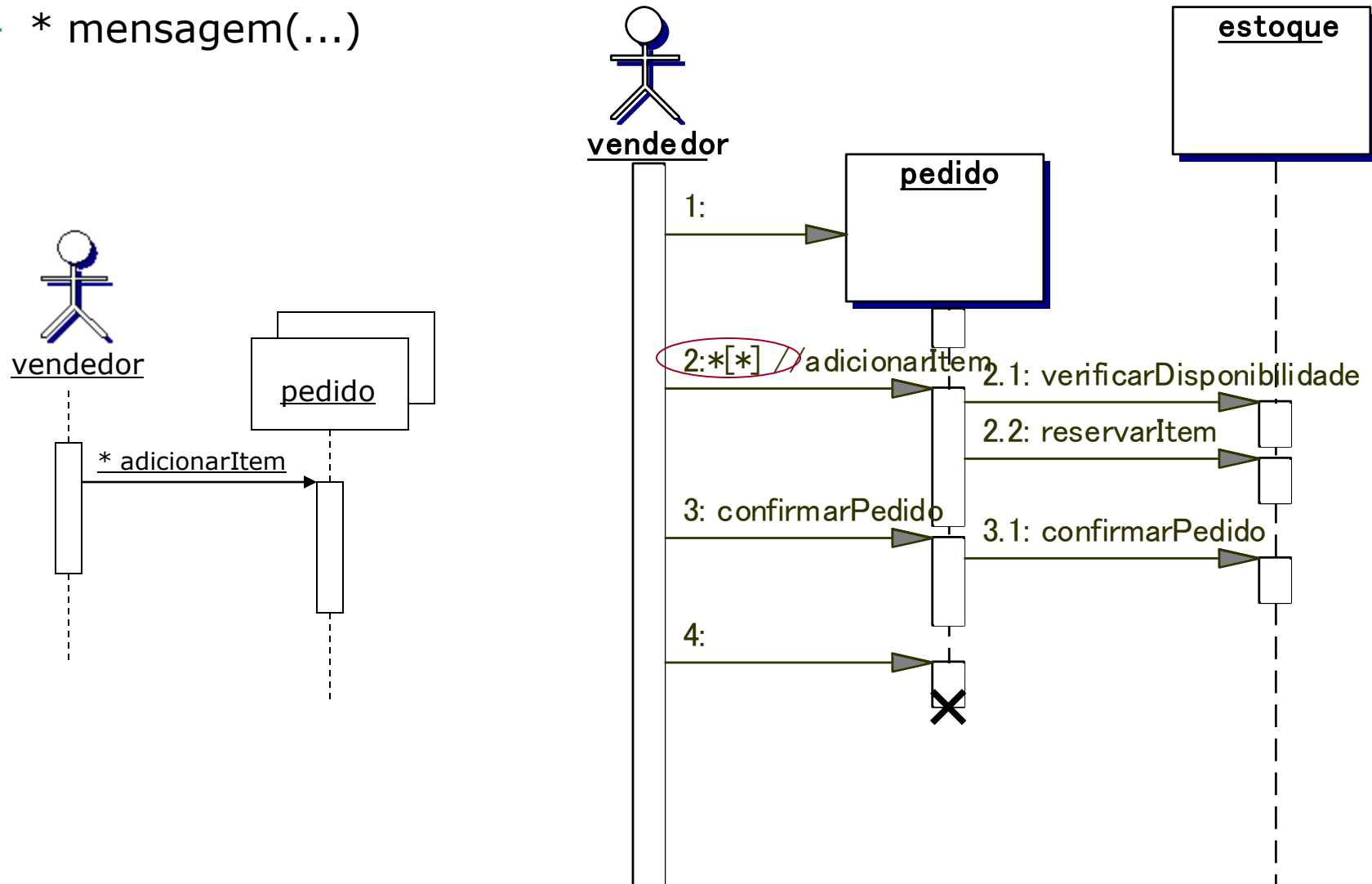
Símbolo	Significado
	Mensagem síncrona
	Mensagem assíncrona
	Mensagem de retorno (opcional)



- Mensagens podem apresentar condições de guarda
  - condições em que a mensagem é enviada
  - [condição de guarda]



- Uma mensagem pode ser enviada repetidas vezes
  - \* mensagem(...)





- Período de tempo que o objeto executa uma ação
- Relação de controle entre ativação e o responsável pela sua invocação

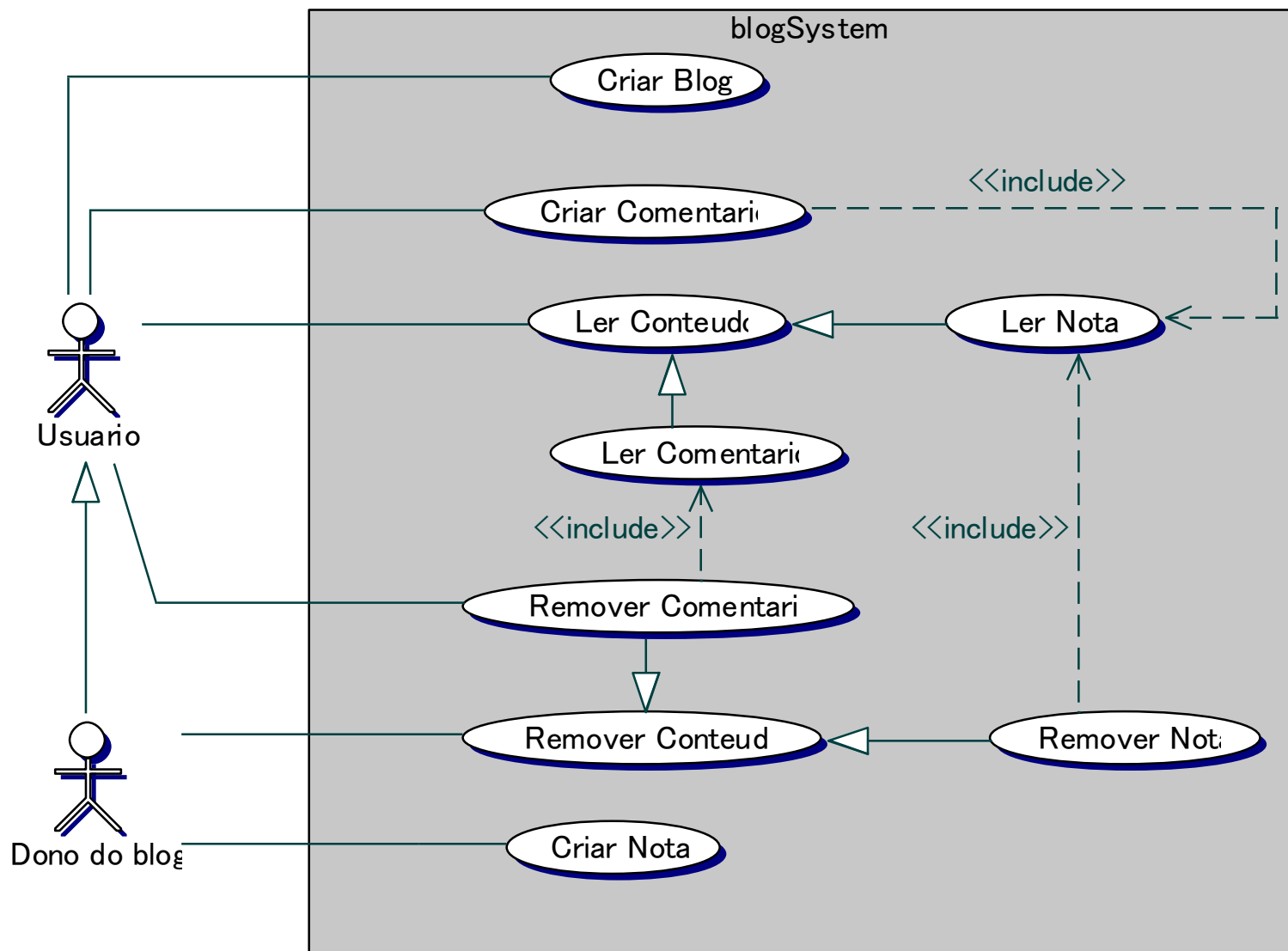
- Escolher um **caso de uso**
- Identificar os **objetos** que fazem parte da **interação**
- Identificar o objeto que **começa** a interação
- Identificar as **mensagens** trocadas entre os objetos
- Identificar a **seqüência** destas mensagens

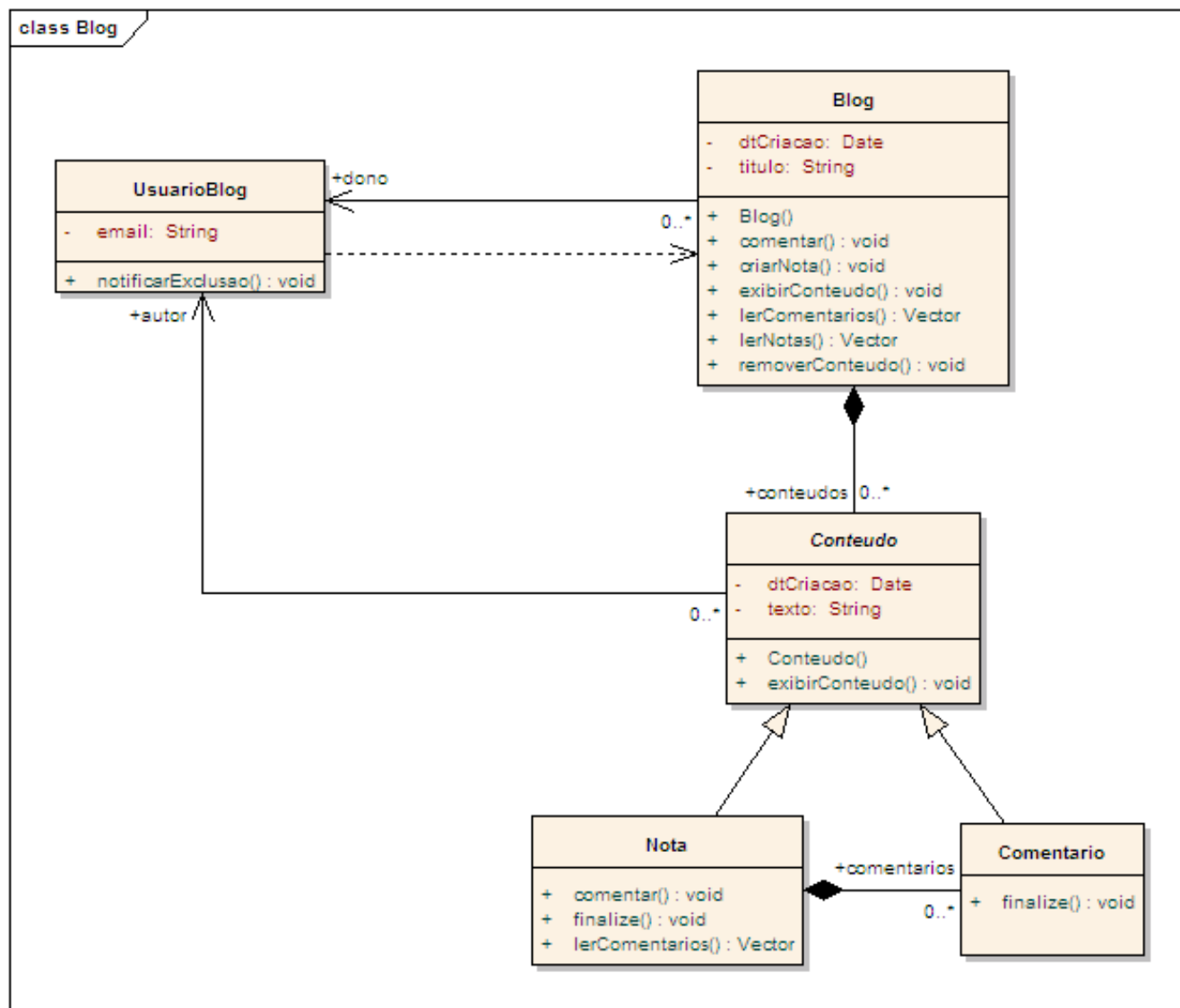
## ESTUDO DE CASO

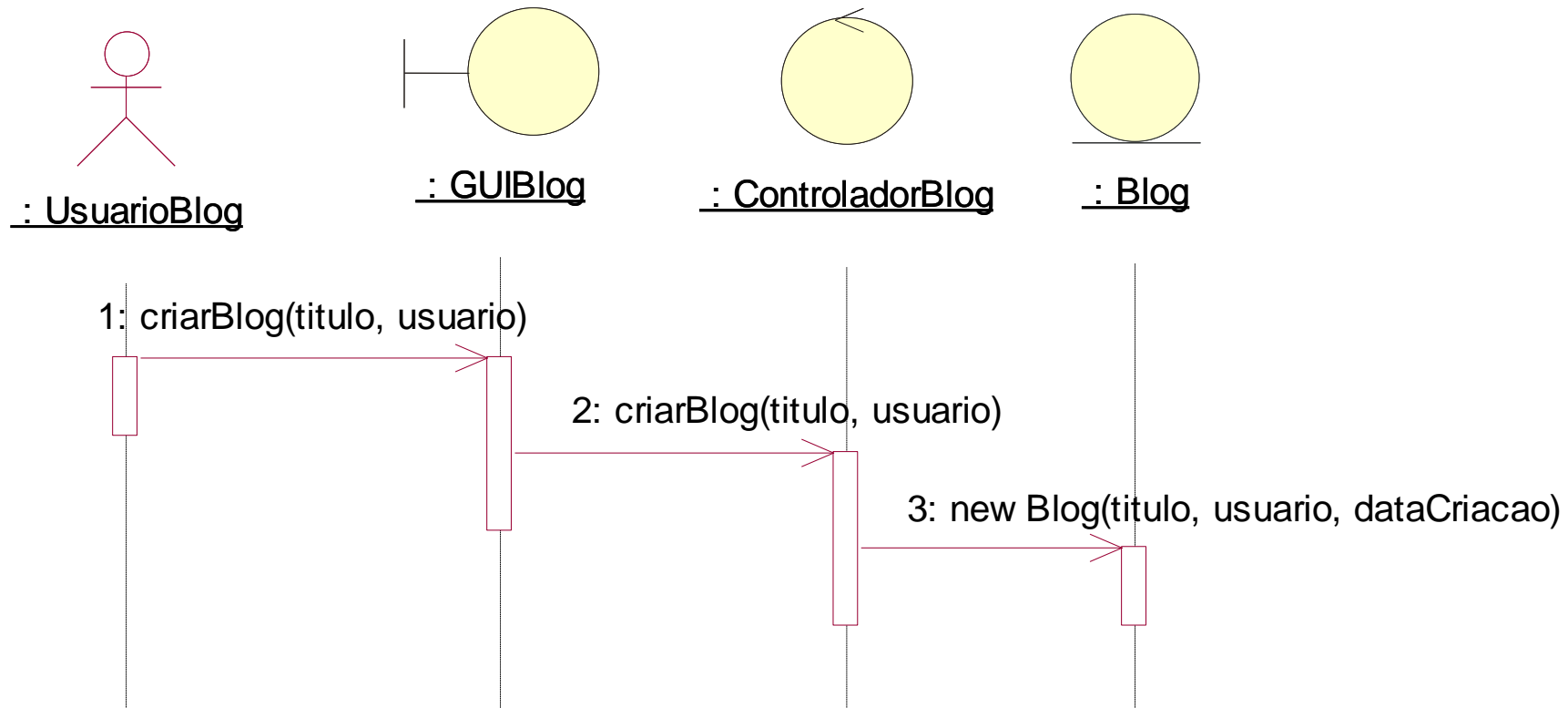
- Um *blog* tem um título e uma data de criação e além disso é um conjunto de conteúdos.
- Estes conteúdos (mensagens) podem ser notas ou comentários sobre as notas. Tanto notas quanto comentários têm características comuns como o texto e a data de sua criação.
- Todo usuário possui:
  - E-mail (deve ser único, ou seja, não há mais de um usuário com o mesmo e-mail)

- Um *blog* tem um título e uma data de criação e além disso é um conjunto de conteúdos.
- Estes conteúdos (mensagens) podem ser notas ou comentários sobre as notas. Tanto notas quanto comentários têm características comuns como o texto e a data de sua criação.
- Todo usuário possui:
  - E-mail (deve ser único, ou seja, não há mais de um usuário com o mesmo e-mail)
  - Permitir a criação de blogs
- Permitir a utilização de blogs
  - Qualquer usuário pode ler conteúdos
  - Somente o dono do blog pode criar notas
  - Qualquer usuário pode criar comentários. Para criar um comentário o usuário precisa ler as notas.
  - Somente o dono do blog pode remover conteúdos. Para remover um conteúdo ele precisará ler o conteúdo. Caso ele remova um comentário, o autor do comentário deve ser notificado por e-mail.

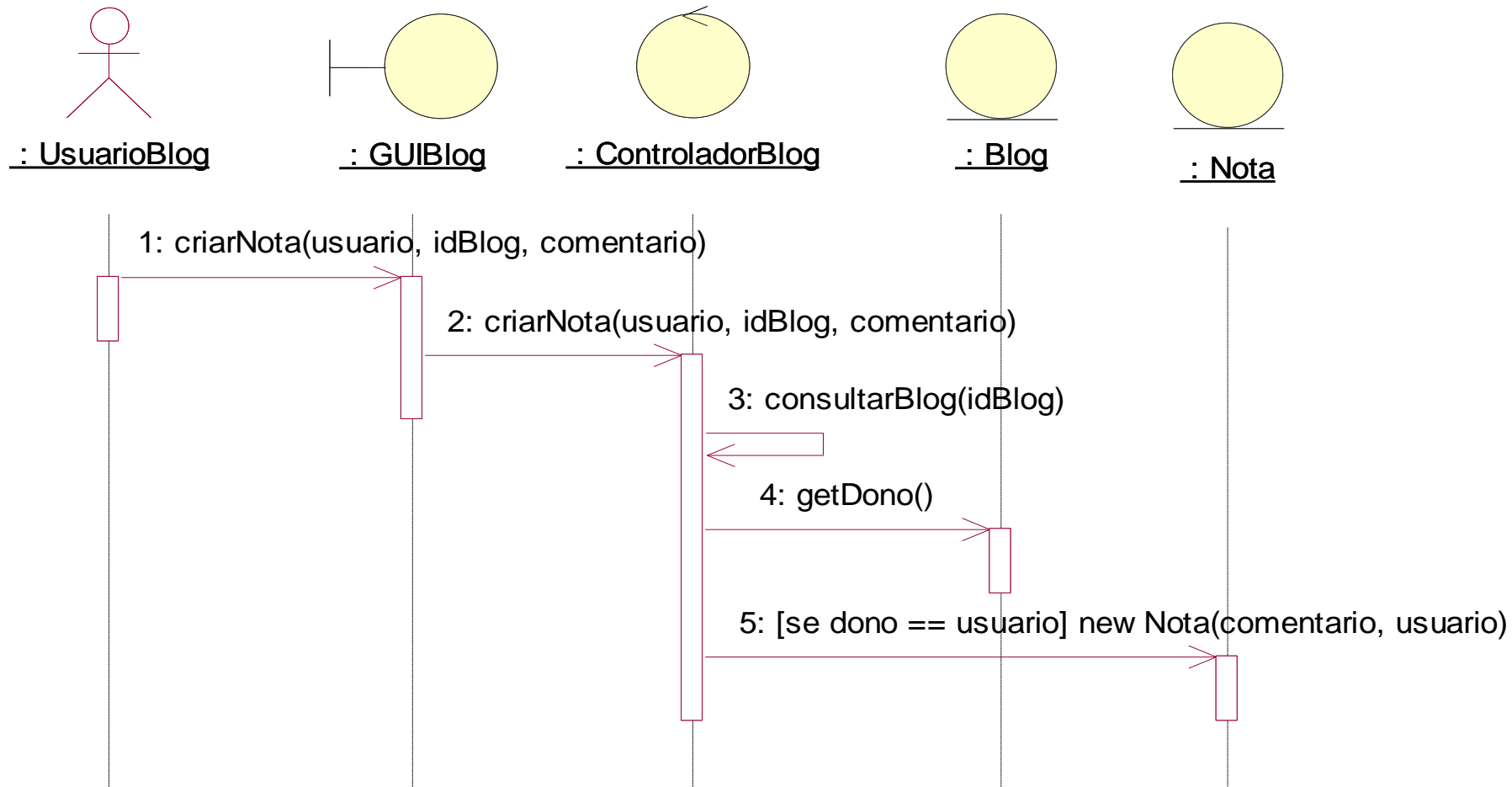
- Desenvolva:
  - CASO DE USO
  - DIAGRAMA DE CLASSES
  - DIAGRAMA DE SEQUÊNCIA





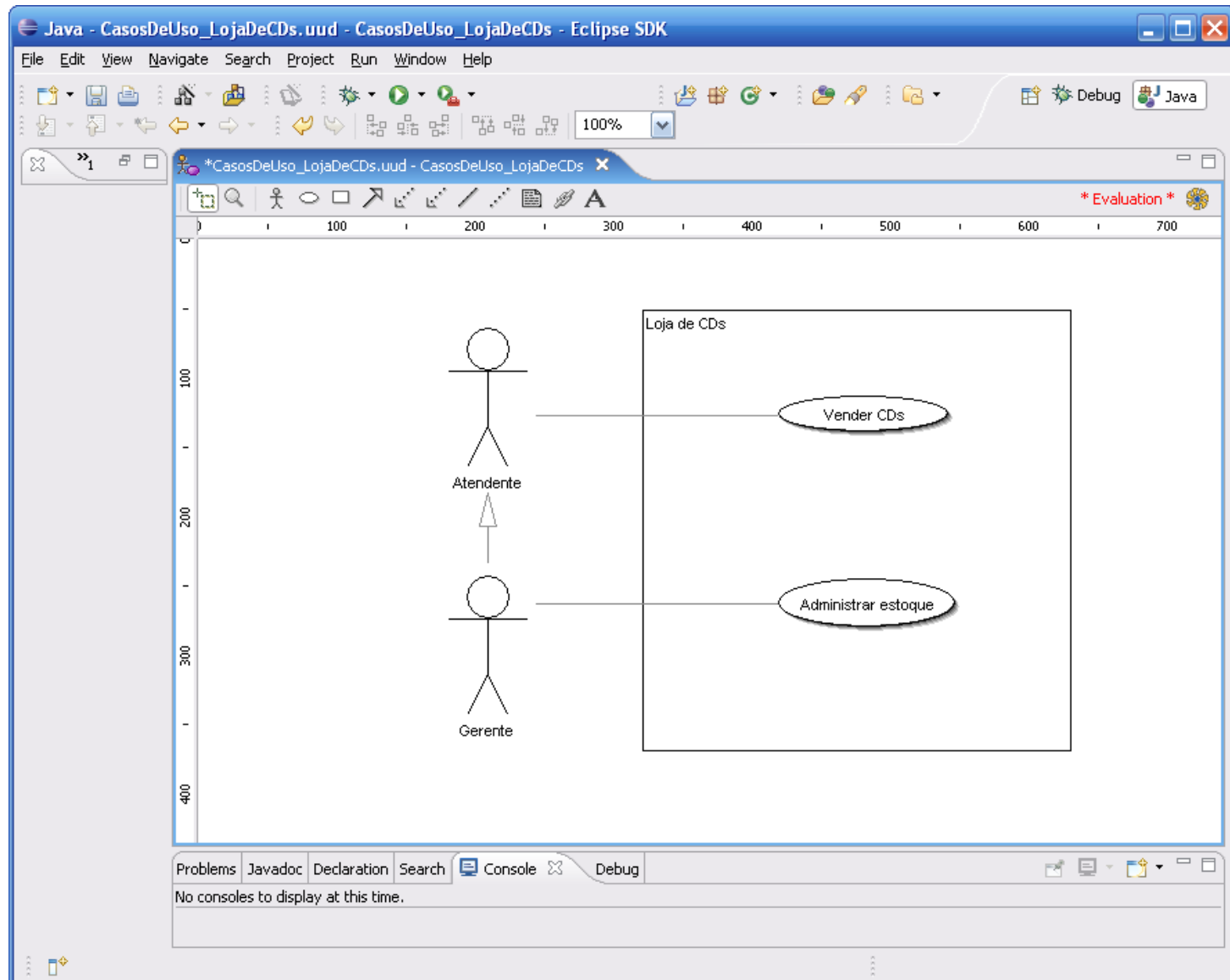


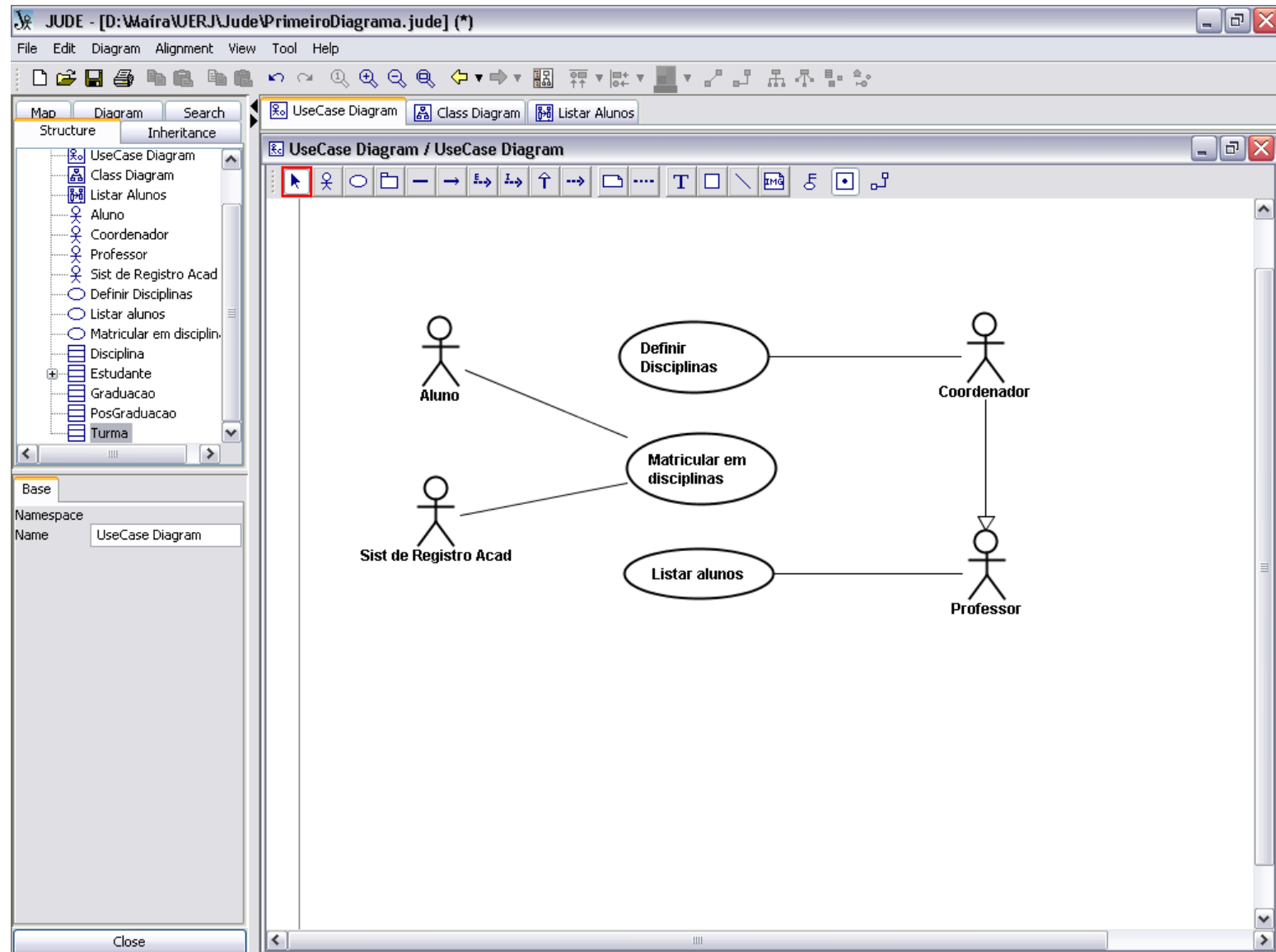




## Exemplo de Ferramentas

- Omondo – Plugin para Eclipse - <http://www.omondo.com/>
- Astah – <http://astah.net>
- Together - [http://www.borland.com/products/downloads/download\\_together.html](http://www.borland.com/products/downloads/download_together.html)
- IBM Rational Rose - <http://www.ibm.com/software/rational>
- ...





- Slides adaptado das Notas de aula do Laboratório de Engenharia de Software da PUC – Rio, disponível em <http://wiki.les.inf.puc-rio.br/index.php/>
- **Guedes, G., UML 2.0 - Uma Abordagem Prática . Rio de Janeiro : Novatec /2009.**
- Cockburn, A., *Writing Effective Use Cases*, Addison-Wesley, 2001.
- Fowler, M e Scott, K., *UML Distilled – A Brief Guide to the standard Object Modeling Language*, Addison Wesley Longman, 2002
- Booch, G., Rumbaugh, J. and Jacobson, I., *Unified Modeling Language User Guide*, 2<sup>nd</sup> Edition, Addison-Wesley Object Technology Series, 2005.