Análise e Projeto Orientados a Objetos

Professor José Azanha Neto

jose.azanha@uninove.br

Sumário

- Apresentação do Professor
- Apresentação da Disciplina
- Bibliografia

Apresentação da Disciplina

- A importância da modelagem no desenvolvimento de sistemas;
- Histórico da evolução das metodologias de modelagem, modelagem tradicional, estruturada e essencial;
- Modelagem orientada a objeto, padrões e normas, notação da interface, subtipos, composição, agregação e diagramas da UML.

Bibliografia

- BLAHA, Michael; RUMBAUGH, James. Modelagem e projetos baseados em objetos com UML 2 – 2ª Edição. Rio de Janeiro: Campus, 2006.
- BOOCK, Grady; JACOBSON, Ivar; RUMBAUGH, James. UML: guia do usuário. Rio de Janeiro: Campus, 2000.
- CARLSON, David. Modelagem de aplicações XML com UML: aplicações práticas de e-business. São Paulo: Pearson, 2002.

BIBLIOGRAFIA COMPLEMENTAR:

- BEZERRA, Eduardo. Princípios de análise e projeto de sistemas com UML. 2ª Edição. Rio de Janeiro: Campus, 2007.
- LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientado a objetos – 2ª Edição. Porto Alegre: Bookman, 2006.

Introdução a Modelagem de Sistemas

Atividades do Desenvolvimento de Sistemas

- Análise;
- Projeto;
- Implementação e Testes Unitários;
- Testes Integrados;
- Homologação e Aceite;
- Treinamento;
- Implantação e;
- Acompanhamento pós-implantação.

Introdução a Modelagem de Sistemas

Análise

- A análise enfatiza a investigação do problema.
- O objetivo da análise é levar o analista a investigar e a descobrir a causa e a solução do problema.
- Para que esta etapa seja realizada em menos tempo e de forma mais precisa, deve-se ter um bom método de trabalho.

Introdução a Modelagem de Sistemas

Projeto

- A fase de projeto enfatiza a proposta de uma solução que atenda os requisitos da análise.
- Logo, se a analise é uma investigação para tentar descobrir o que o cliente quer, o projeto consiste em propor uma solução com base no conhecimento adquirido na análise.

Introdução a Modelagem de Sistemas

Implementação e Testes Unitários

- A utilização de técnicas sistemáticas nas fases de análise e projeto faz com que o processo de geração de código possa ser automatizado.
- Neste caso, cabe ao programador dominar as características específicas das linguagens, ferramentas, frameworks e estruturas de dados para adaptar o código gerado aos requisitos indicados.

Introdução a Modelagem de Sistemas

Testes Integrados

- Cabe ao programador executar os testes integrados após a implementação total do sistema.
- Os testes integrados devem ser feitos em todo os sistema com o cruzamento das informações e funcionalidades.
- O principal objetivo é validar o resultado das informações geradas e regras de negócio.

Introdução a Modelagem de Sistemas

Homologação

- Nesta fase, o projeto de sistema é implantado no Ambiente de Testes do cliente e os mesmos testes executados na fase de Testes Integrados são feitos na Homologação.
- Todos os erros encontrados são documentados e enviados para resolução.
- Caso contrário, o cliente assina o termo de aceite.

Introdução a Modelagem de Sistemas

Implantação e Pós-Implantação

- Nesta fase, o projeto de sistema é implantado no Ambiente de Produção do cliente e parte dos testes executados na fase de Homologação.
- Após a implantação, é feito um acompanhamento do sistema no período de uma ou duas semanas junto aos usuários.

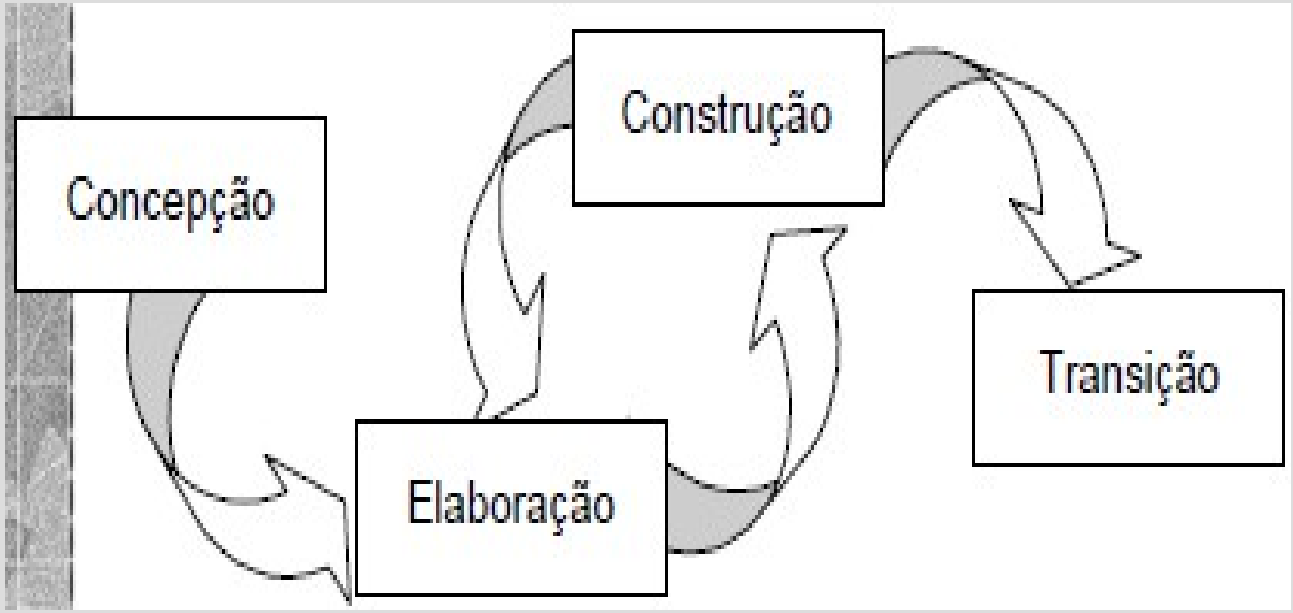
Introdução a Modelagem de Sistemas

Fases do Processo Unificado - RUP

- A fase de concepção incorpora o estudo de viabilidade e uma parte da análise de requisitos.
- A fase de elaboração incorpora a maior parte da análise de requisitos e o projeto.
- A fase de construção corresponde à programação e testes.
- A fase de transição consiste na instalação e manutenção do sistema.

Introdução a Modelagem de Sistemas

Fases do Processo Unificado - RUP



Introdução a Modelagem de Sistemas

Fases do SCRUM

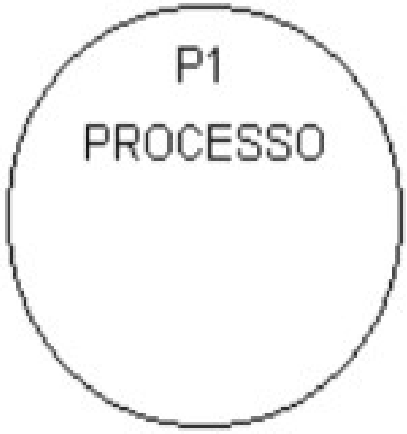
Análise Estruturada

- Enfatiza a perspectiva das funções, com ênfase nos processos.
- Utiliza as seguintes ferramentas:
  - Diagrama de Fluxo de Dados.
  - Dicionário de Dados.
  - Especificação da Lógica de Processos.
- A análise estruturada clássica não modela o comportamento temporal, nem complexos relacionamentos de dados.

Análise Estruturada

- O sistema especificado pelo DFD pode ser representado pelos seguintes objetos:
  - Processo (Pn)
  - Entidade Externa
  - Depósito de Dados
  - Fluxo de Dados

Análise Estruturada



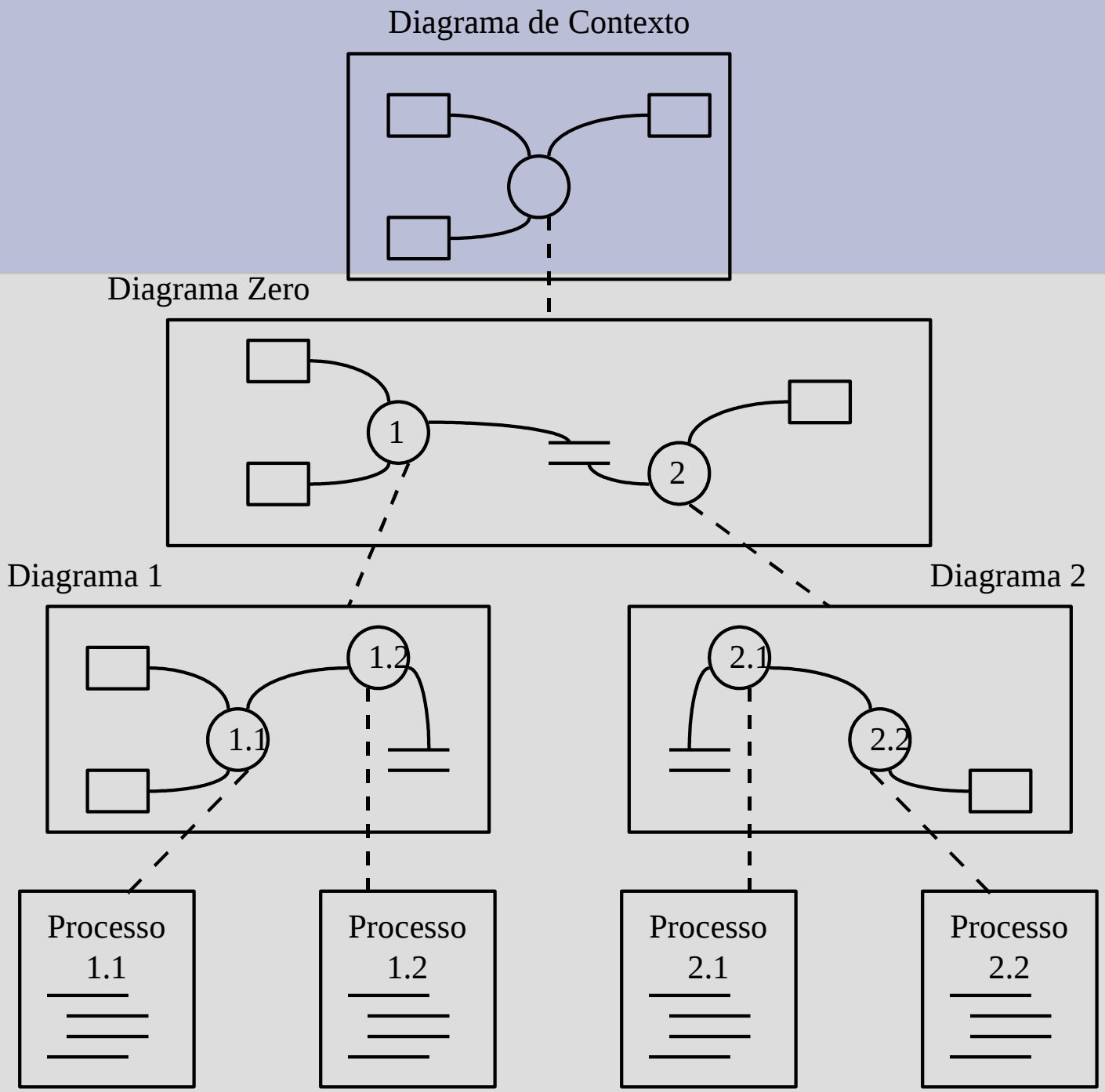
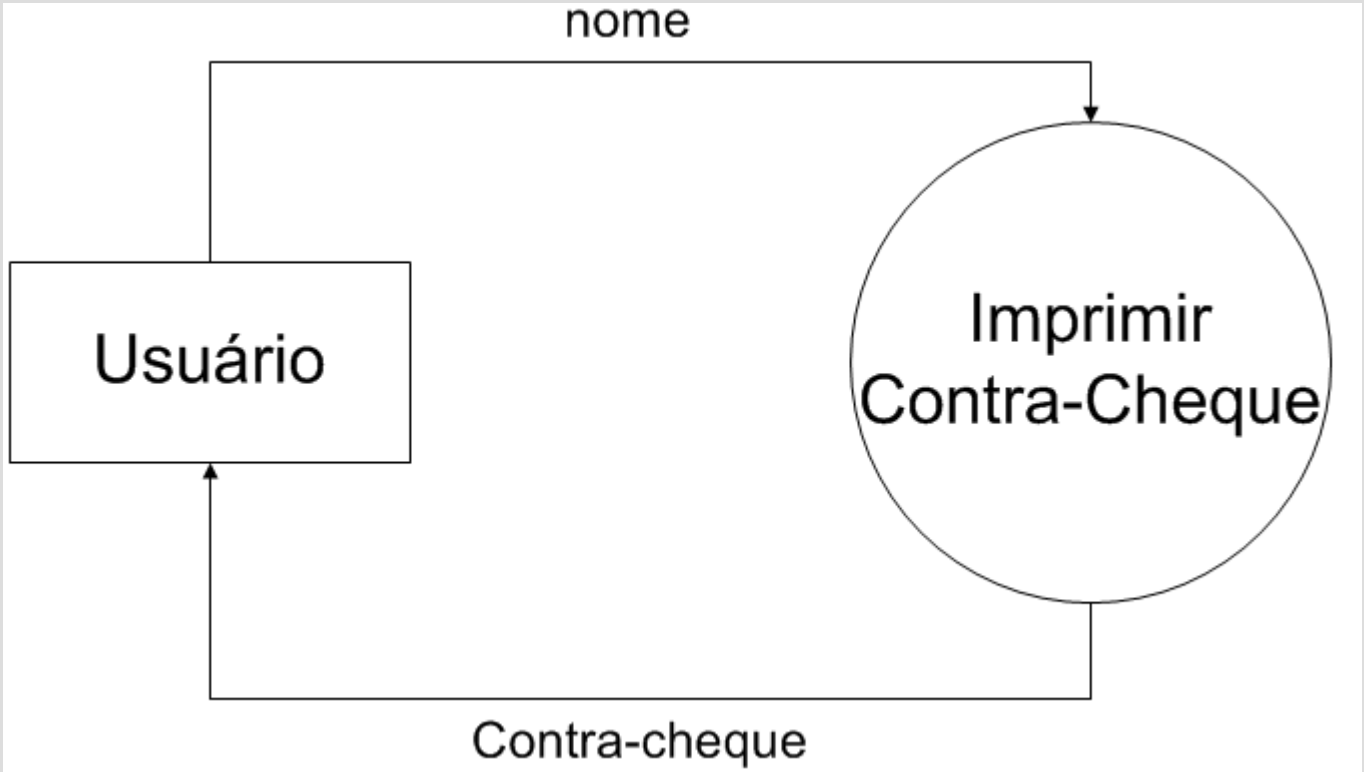


Diagrama de Contexto
Diagrama Zero
1
2
Diagrama 1
Diagrama 2
1.1
1.2
2.1
2.2
Especificação da lógica dos processos
Processo 1.1
Processo 1.2
Processo 2.1
Processo 2.2

Análise Estruturada

Diagrama de Contexto – Emissão de Contra-Cheque



Análise Estruturada - Exemplo

DFD Nível 0 – Solicitação de Pedido

Análise Essencial

- Foi uma evolução da Análise Estruturada por adicionar a preocupação com o **controle**.
- Usa uma lista de eventos externos como base para o particionamento do sistema.
- O modelo essencial é construído sem considerar restrições de implementação.

Análise Essencial

- O modelo Essencial é formado pelo:
  - Modelo Ambiental – define a fronteira entre o sistema e o ambiente.
  - Modelo Comportamental – descreve o comportamento interno do sistema.
  - Modelo de Informação – modela os dados necessários às atividades essenciais do sistema.

Análise Essencial

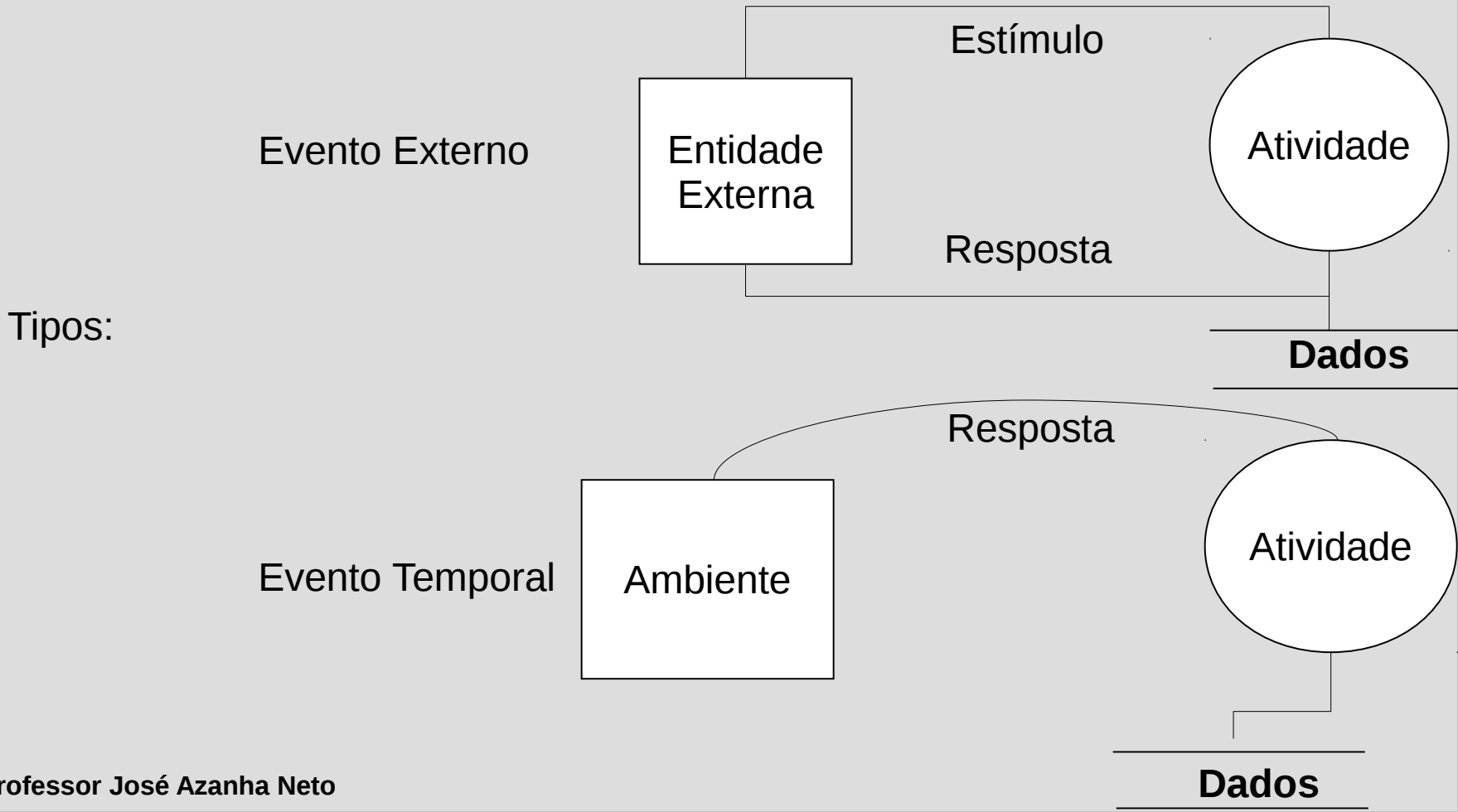
- Modelo Ambiental
  - Diagrama de Contexto - Define as interfaces entre o sistema e o ambiente. São identificadas informações externas e as produzidas como saída.
  - Lista de Eventos - Identifica os eventos que ocorrem no ambiente e como o sistema deve reagir.

Análise Essencial

- Modelo Ambiental – Diagrama de Contexto.

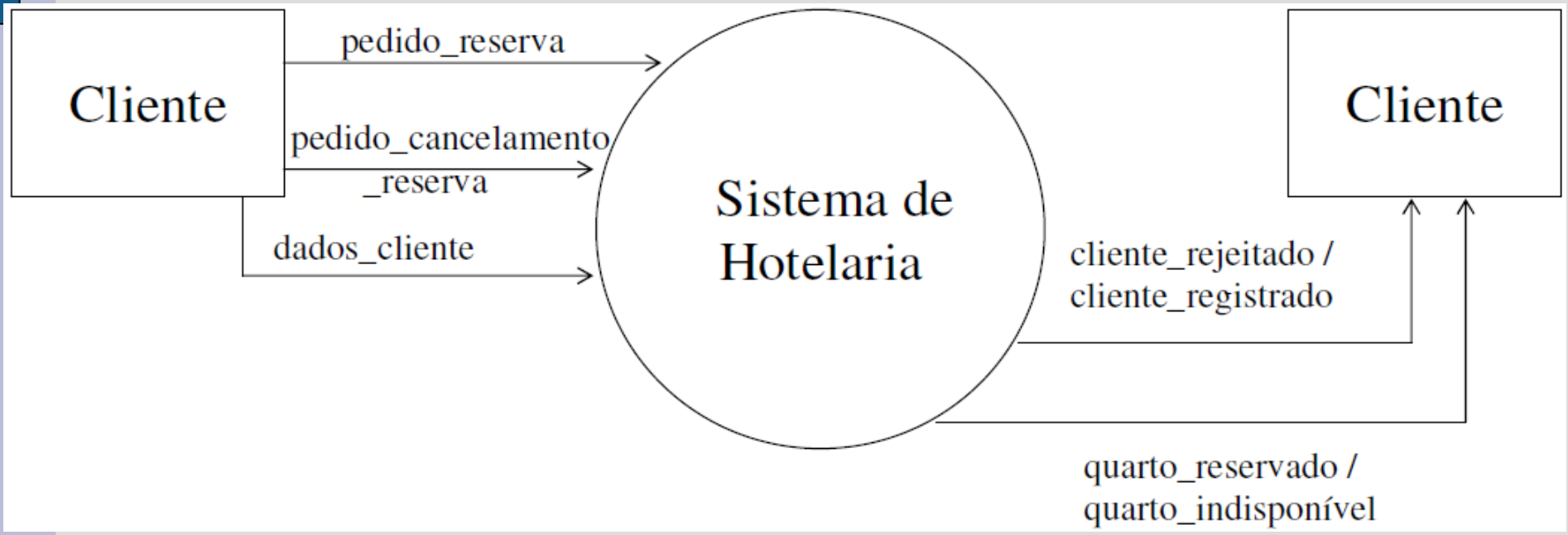
Análise Essencial

- Modelo ambiental: Lista de Eventos



Análise Essencial

- Modelo Ambiental: Lista de Eventos no Diagrama de Contexto

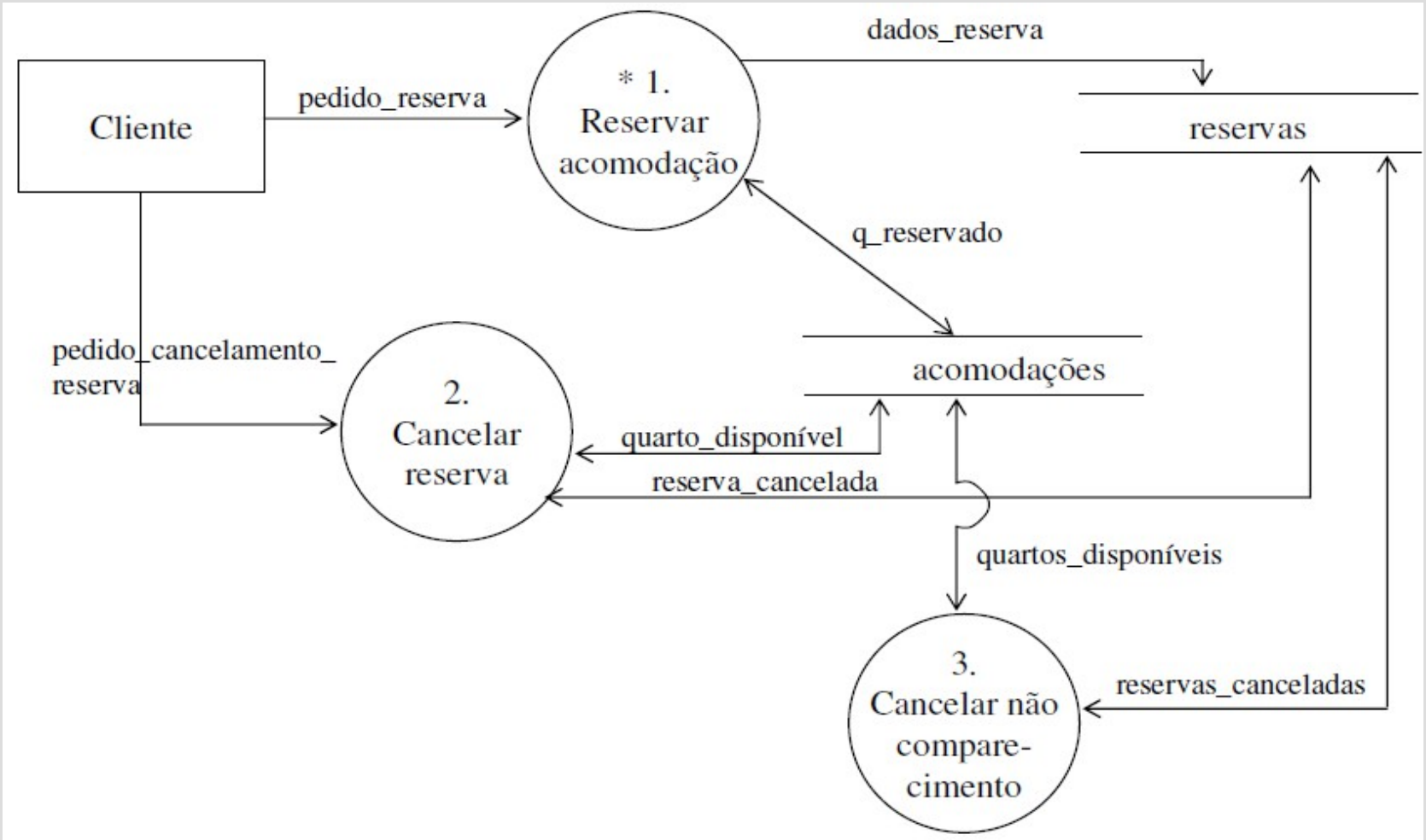


Análise Essencial

- Modelo Comportamental
  - Mostra o comportamento interno do sistema.
  - Usa como ferramenta DFD com abordagem diferente.
  - Constrói um DFD para cada evento (DFD de resposta a eventos). A partir dele é feito o agrupamento para formar os diagramas superiores e inferiores.
  - Dicionário de Dados e Especificação de processos.

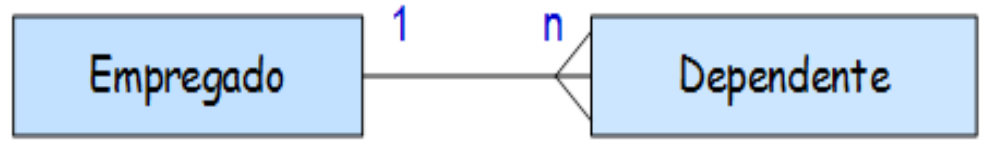
Análise Essencial

- Modelo Comportamental – DFD Nível 0



Análise Essencial

- Modelo de Informação
  - Representa os dados necessários ao sistema.
- Ferramentas utilizadas são:
  - Diagrama de entidade e relacionamento
  - Deriva da lista de eventos
  - Representa a estrutura estática dos dados
  - Dicionário de Dados



Orientação a Objetos

- A orientação a objetos representa melhor o mundo real, uma vez que a percepção e o raciocínio estão relacionados diretamente com objetos.
- Objetivo principal da utilização do paradigma da Orientação a Objetos, na construção de software:
  - Rápido
    - Não perder tempo no desenvolvimento
  - Barato
    - Reutilização de código
  - Flexível
    - Fácil modificar ou estender

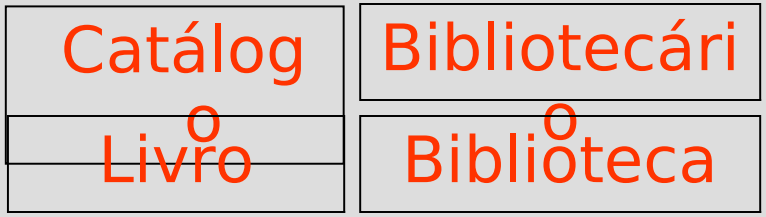
Estruturada x OO

- Com a orientação a objetos procura-se eliminar as diferenças entre as etapas de análise, projeto e implementação, reabilitando a difamada tarefa de implementação
- O segredo é fazer com que os conceitos de programação, e as notações para programação, sejam suficientemente de alto nível para que possam servir apropriadamente como ferramentas de modelagem.

Estruturada x OO

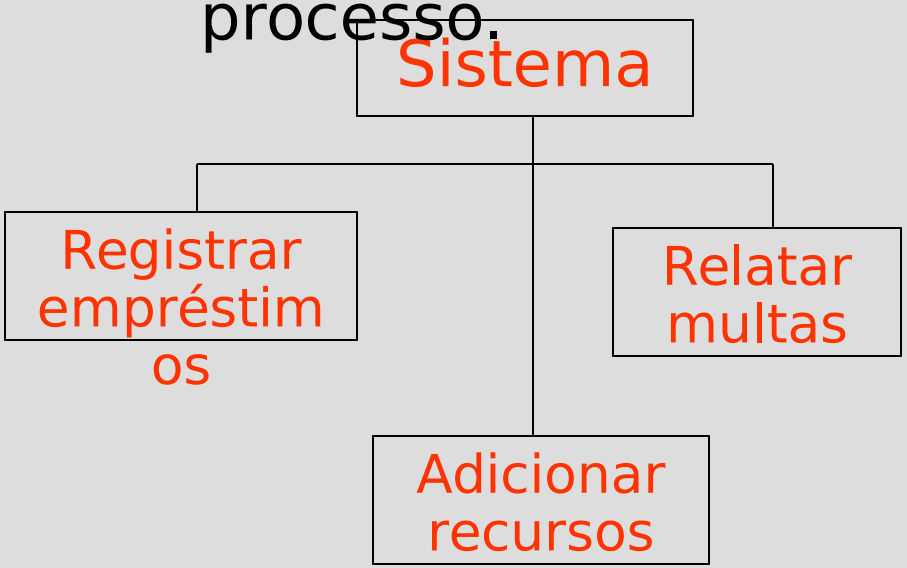
A/P orientados a objetos

Decompor por objetos ou conceitos.



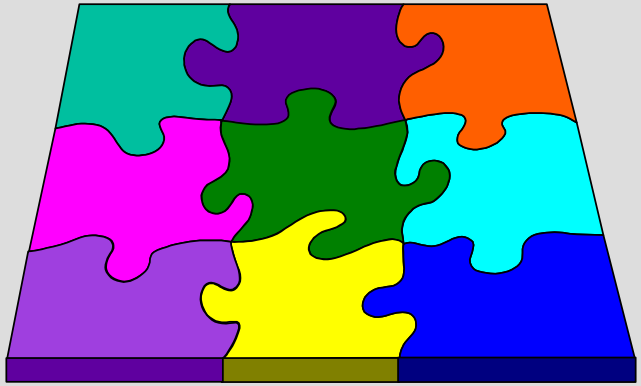
A/P estruturados

Decompor por funções ou processo.



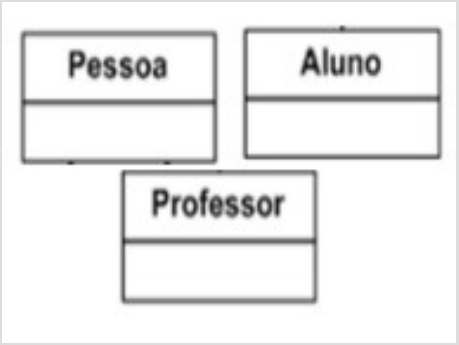
Elementos de Modelagem OO

- Classes e Objetos Persistentes
- Associações e Ligações
- Multiplicidade e Cardinalidade
- Herança
  - Generalização e Especialização
- Agregação
- Entidades Associativas
- Atributos de Ligação
- Associações Qualificadas
- Polimorfismo
- Abstrações



Conceitos sobre OO

- **Objeto**: Representa algo do mundo real que possui características (valores) e comportamentos (ações ou eventos).
- A orientação a objetos é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.
  - Exemplos: Carro, Pessoa, Mesa, Cadeira, etc.

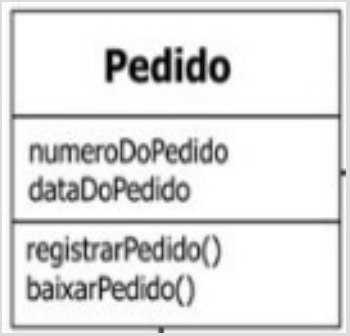


Conceitos sobre OO

- **Classe**: Representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos.
- Exemplo: Classe Pessoa; Classe Veículo

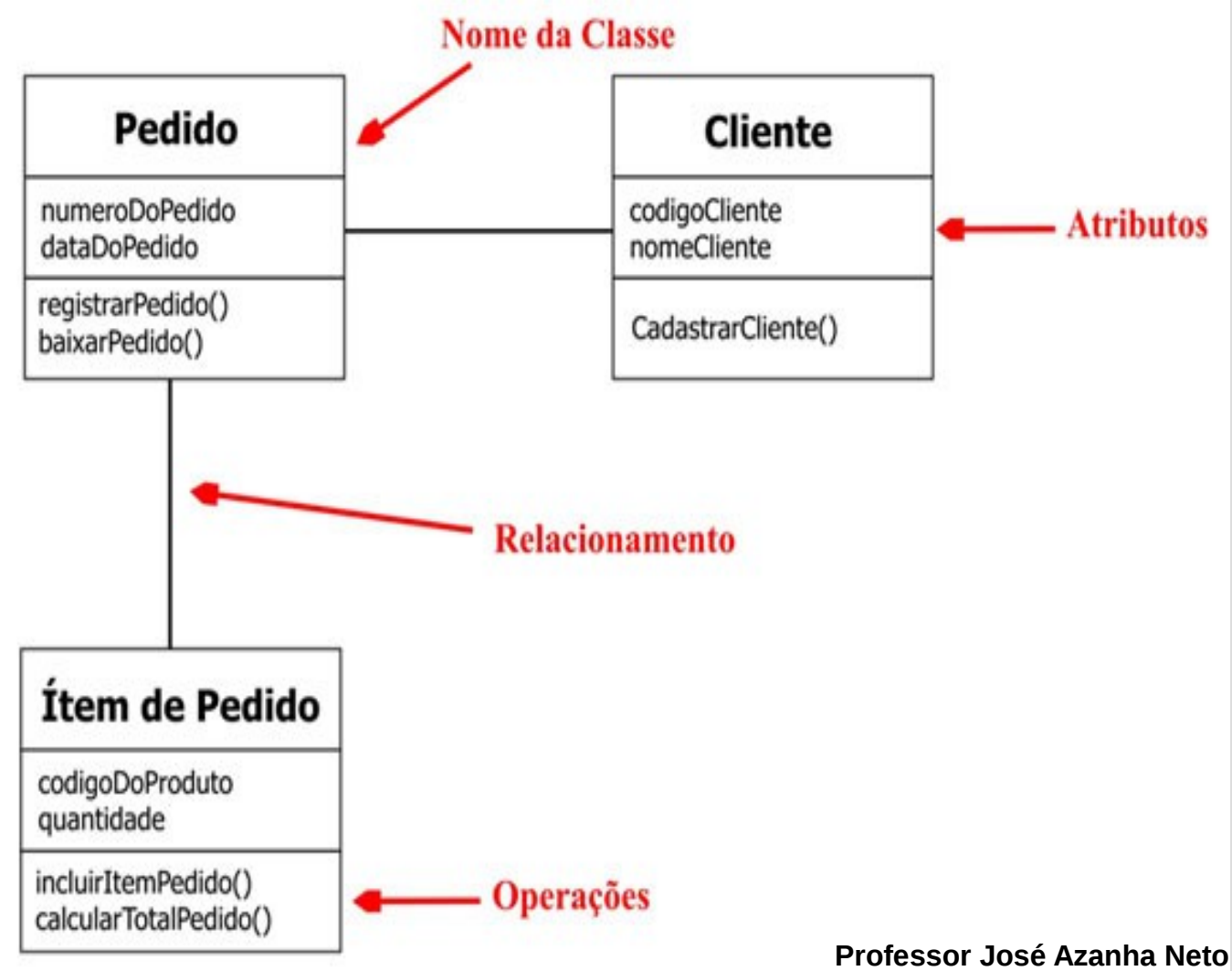
Conceitos sobre OO

- **Nome da Classe**: Determina o nome do objeto na classe.
- **Atributos da Classe**: Determinam as características de um objeto na classe.
- **Operações (métodos) da Classe**: Determinam o comportamento e/ou ações que um objeto pode sofrer.



Conceitos sobre OO

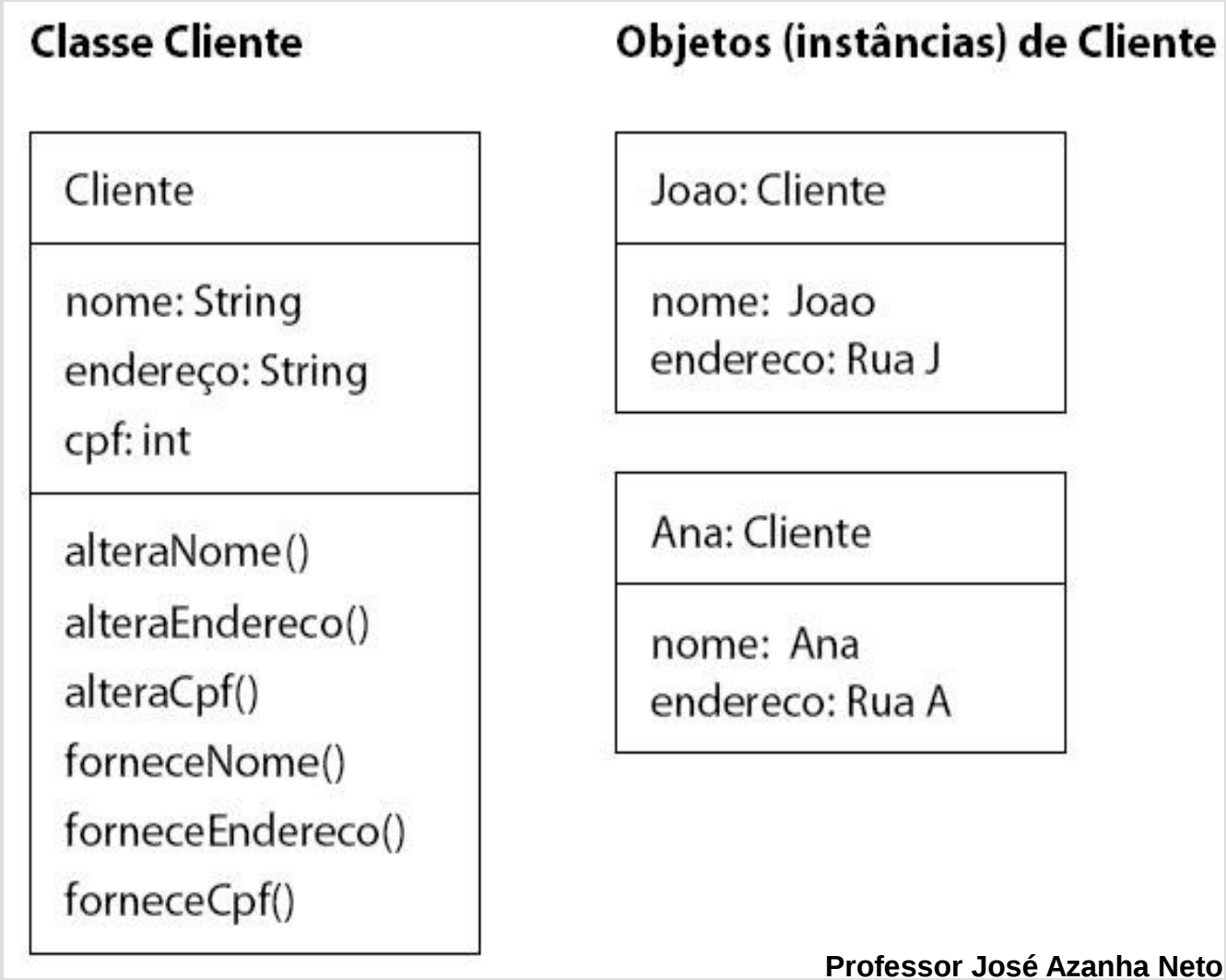
Objeto e Classe



Conceitos sobre OO

Objeto, Classe e Instâncias

As instâncias representam o número de ocorrências de um objeto específico. Veja o exemplo do objeto Cliente.

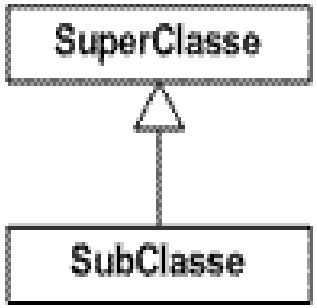


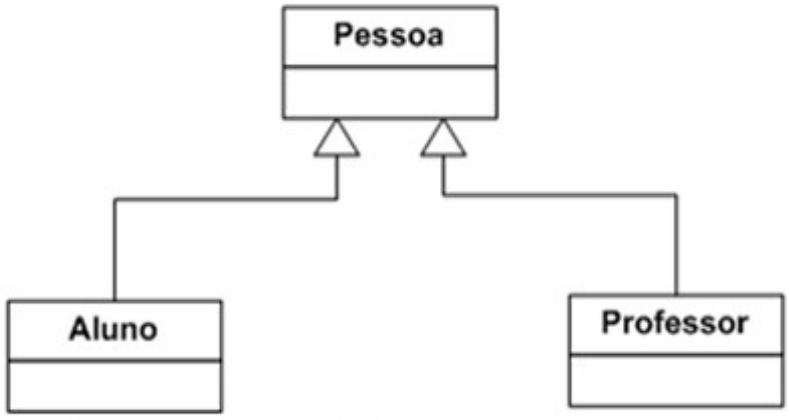
Conceitos sobre OO

- **Herança**: Representa que um determinado objeto-filho herda (recebe) todas as caracteristicas e comportamentos do objeto-pai.

  Exemplos:

Astra	← Astra Sport	Pessoa	← PF
Médico	← Ortopedista	Computador	← Notebook

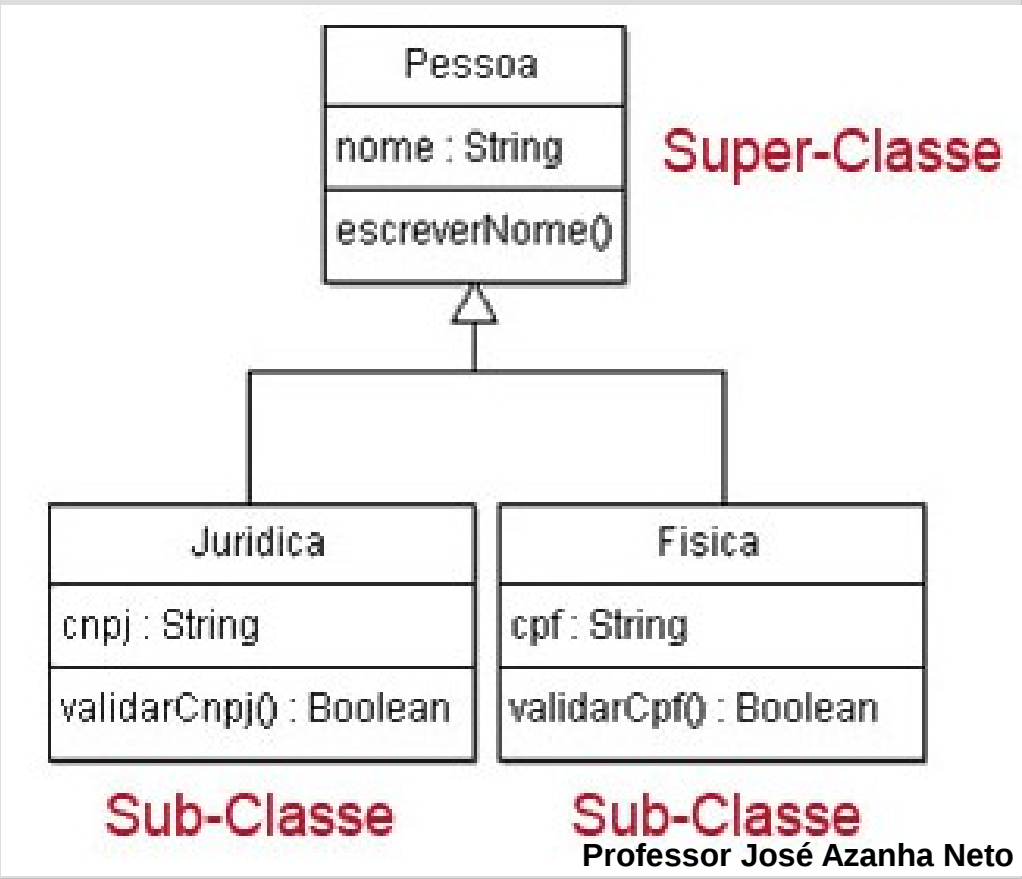




Herança: Aluno e Professor herdam de Pessoa

Conceitos sobre OO

- **Generalização**:
Representa o nome dado para as Super-Classes, ou seja, objetos cuja finalidade é genérica.
- **Especialização**:
Representa o nome dado para as Sub-Classes, ou seja, objetos cuja finalidade é específica.



Conceitos sobre OO: Polimorfismo

- Em OOP, polimorfismo é uma facilidade que permite que dois ou mais objetos diferentes respondam a mesma mensagem.

  O objeto emissor não precisa saber como o objeto receptor implementa a mensagem. Apenas os objetos receptores devem se preocupar com isso.

  Uma analogia ao polimorfismo é o sinal dos colégios. Muito embora seja um único sinal, ele significa coisas distintas para cada aluno: uns vão para casa, outros para biblioteca e terceiros voltam para sala de aula. Logo, todos respondem ao sinal, mas cada um do seu jeito.

Polimorfismo

```
package Polimorfismo;

public class Cliente {

    public int nu_matricula;
    public String nm_cliente;

    void imprimirCliente() {

        System.out.println("Início dos Atributos da Classe Cliente...");
        System.out.println("Nome Cliente: " + this.nm_cliente);
        System.out.println("Matrícula Cliente: " + this.nu_matricula);
        System.out.println("Fim dos Atributos da Classe Cliente!!!\n");
    }
}
```

Polimorfismo

```
package Polimorfismo;

public class ClientePF extends Cliente {

    public String nu_cpf;

    @Override
    void imprimirCliente() {
        System.out.println("Início dos Atributos da Classe Cliente PF...");
        System.out.println("CPF Cliente: " + nu_cpf);
        System.out.println("Fim dos Atributos da Classe Cliente PF !!!\n");
    }
}
```

Polimorfismo

```
package Polimorfismo;

public class Aplicativo {

    static public void main(String args[]) {

        Cliente manuel = new Cliente();

        ClientePF maria = new ClientePF();

        manuel.nu_matricula = 1010;
        manuel.nm_cliente = "Manuel da Silva";

        maria.nu_cpf = "123.456.789-00";

        manuel.imprimirCliente();
        maria.imprimirCliente();

        manuel = maria;
        manuel.imprimirCliente();
    }

}
```

A UML

- A Unified Modeling Language (Linguagem de Modelagem Unificada) é uma técnica de análise baseada em objetos e com grande aplicabilidade em projetos de software por todo o mundo.
- Empresas (especialistas em TI ou não) de pequeno, médio e grande porte

  utilizam esta técnica em seus

  processos de Engenharia de

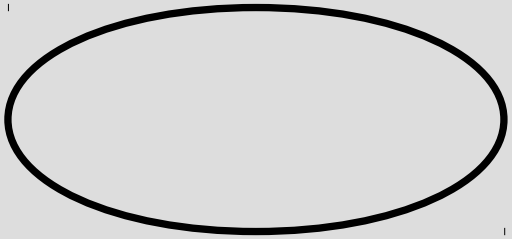
  Software.

Caso de Uso: Objetivos

- Introduzir conceitos de Caso de Uso (*use case)*, ator e fluxo de eventos
- Apresentar sub-fluxos de eventos
- Discutir sobre identificação, evolução e organização de *use cases*
- Apresentar notação UML para reusar atores e *use cases*

Caso de Uso

- Sequência de ações, executada pelo sistema, que gera um resultado
  - De valor observável
  - E para ator particular

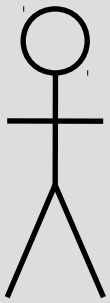


Função

Procedimento computacional/algorítmico específico.

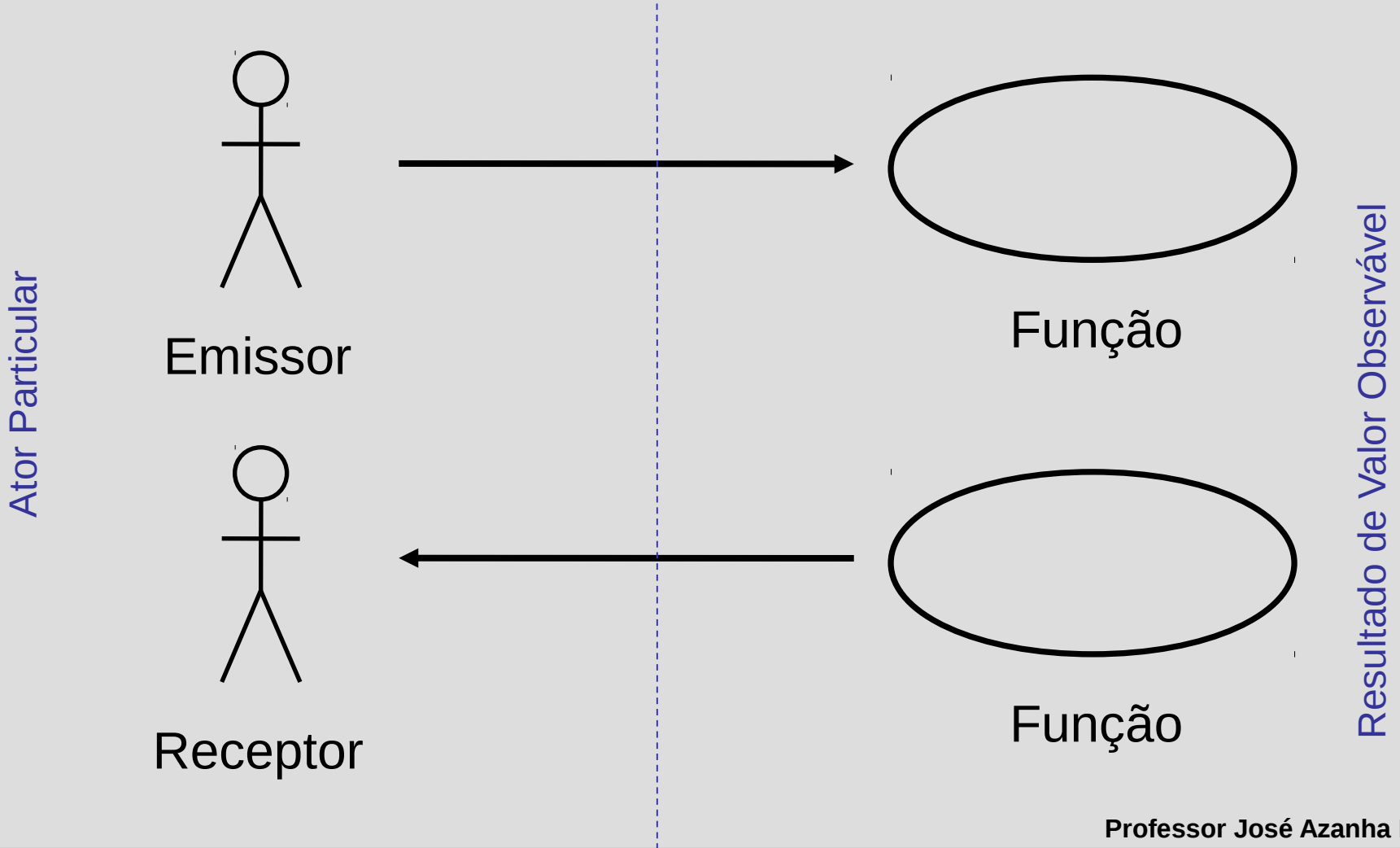
Ator

- Usuário ou Sistema (**externo**) que interage como emissor ou receptor com o sistema.



Ator

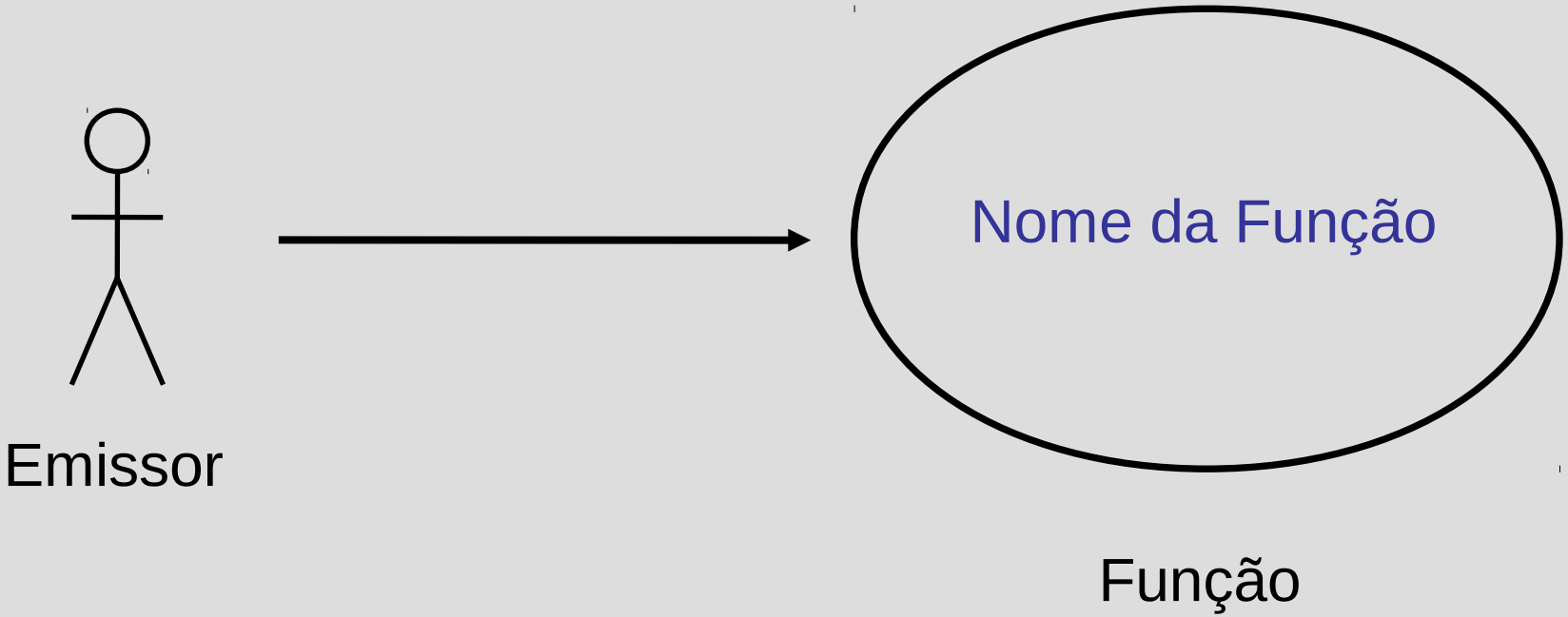
Caso de Uso e Ator



Caso de Uso e Ator

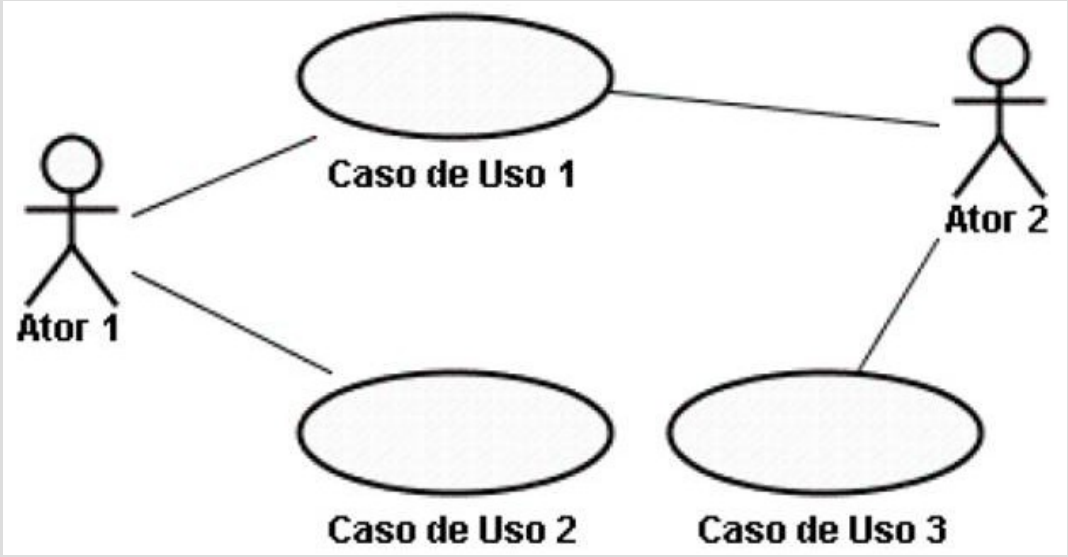
- A descrição de um *use case* define o que o sistema faz quando o *use case* é realizado
- A funcionalidade do sistema é definida por um conjunto de *use cases*, cada um representando um fluxo de eventos específico

Caso de Uso e Ator



A UML

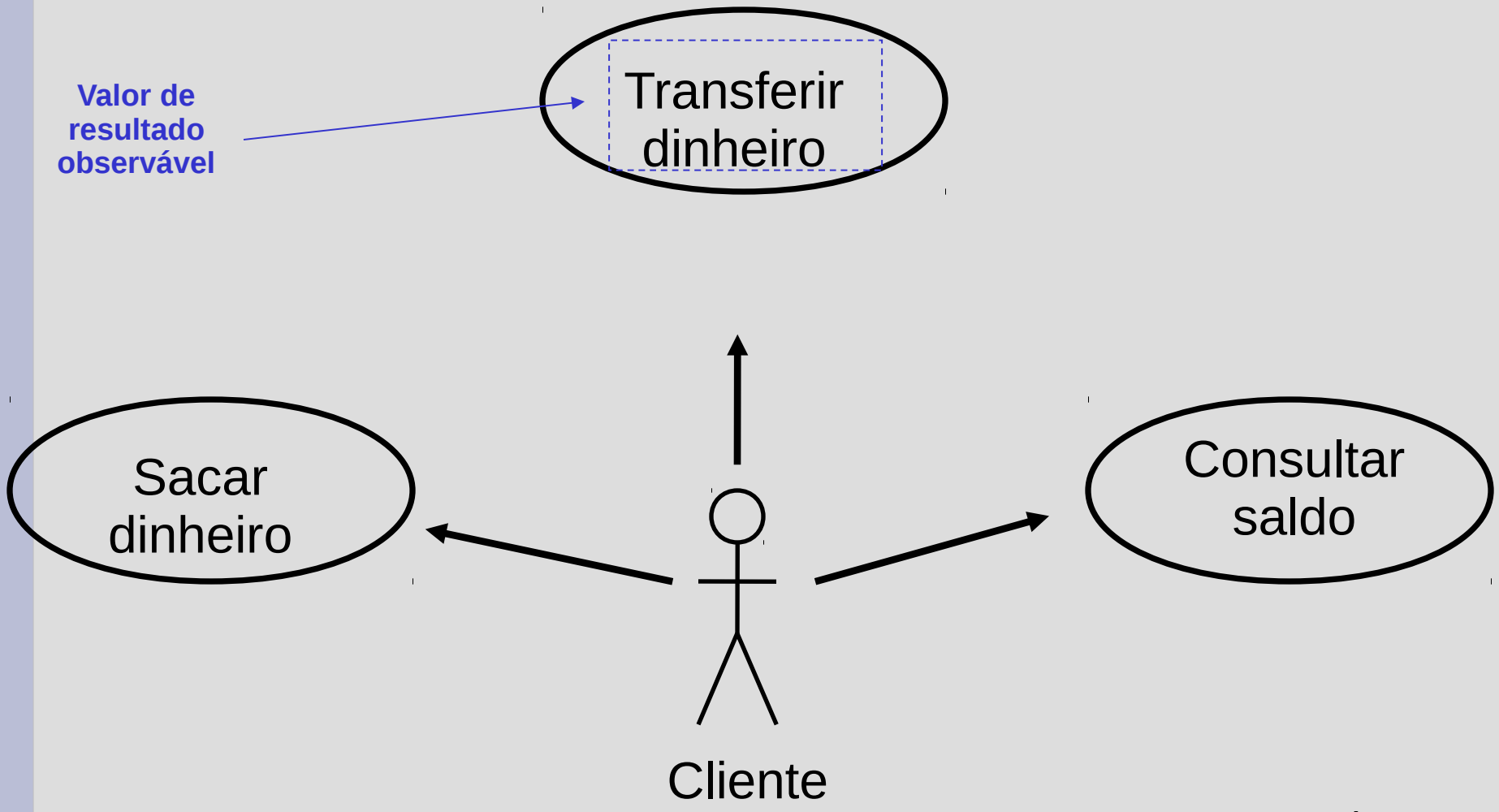
- Estrutura básica do Diagrama de Caso de Uso



Exemplo de Use Case e Ator

- Cliente do banco pode usar um caixa automático para:
  - sacar dinheiro, transferir dinheiro ou consultar o saldo da conta
- Ator: Cliente
- Use cases: Sacar dinheiro, transferir dinheiro e consultar saldo

Exemplo de Use Case e Ator



Identificando Use Cases

- Em geral, difícil decidir entre um ou vários use cases
- Por exemplo, seriam use cases
  - Inserir cartão em um Caixa Automático?
  - Entrar com a senha?
  - Receber o cartão de volta?

Identificando Use Cases

- Representar valor observável para ator
- Pode-se determinar
  - De interações (seqüência de ações) com o sistema que resultam valores para atores
  - Satisfaz um objetivo particular de um ator que o sistema deve prover

Identificando Use Cases

- Facilitar gerenciamento durante ciclo de desenvolvimento
  - A razão para agrupar funcionalidades e chamá-las de *use cases*

Evolução de Use Cases

- Inicialmente *use cases* são simples
  - Apenas esboço sobre funcionamento é suficiente
- Mas com a sedimentação da modelagem
  - Descrição mais detalhada do fluxo de eventos faz-se necessária
- Fluxo de eventos deve ser refinado
  - Todos os *stakeholders* envolvidos devem estar de acordo com a descrição

Organizando Use Cases

- Sistema pequeno não demanda estruturação
  - Exemplo, seis *use cases*, com dois/três atores
- Já sistemas maiores requerem princípios de estruturação e organização
  - Caso contrário, planejamento, atribuição de prioridades, etc., podem se tornar difíceis.

Exercício 1

1) Identifique os atores e casos de uso.

O Portal Geral é um site de conteúdo sobre educação, política, negócios, esportes, revistas, jornais, etc.

O acesso ao conteúdo será gerenciado por meio do controle de usuário, onde cada um terá um ID de sessão. Quando o ID de sessão expirar, o usuário perderá a conexão com o portal, tendo que logar novamente.

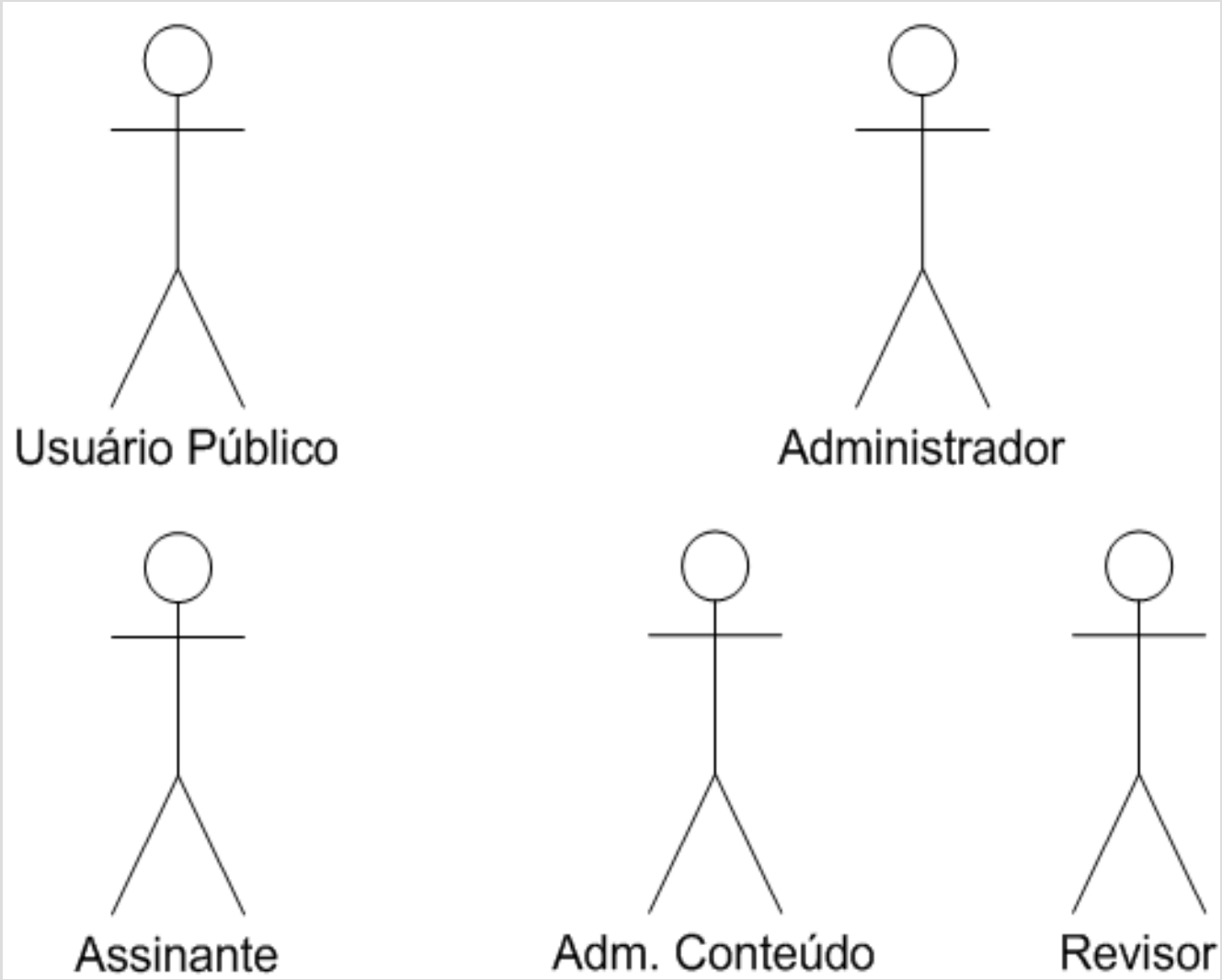
Somente os usuários cadastrados e logados poderão acessar o conteúdo informativo.

Somente os administradores de conteúdo poderão gerenciar o conteúdo do site (banners, notícias, links, etc).

Antes do conteúdo ser publicado, os revisores analisam o conteúdo e submetem à aprovação ou correção.

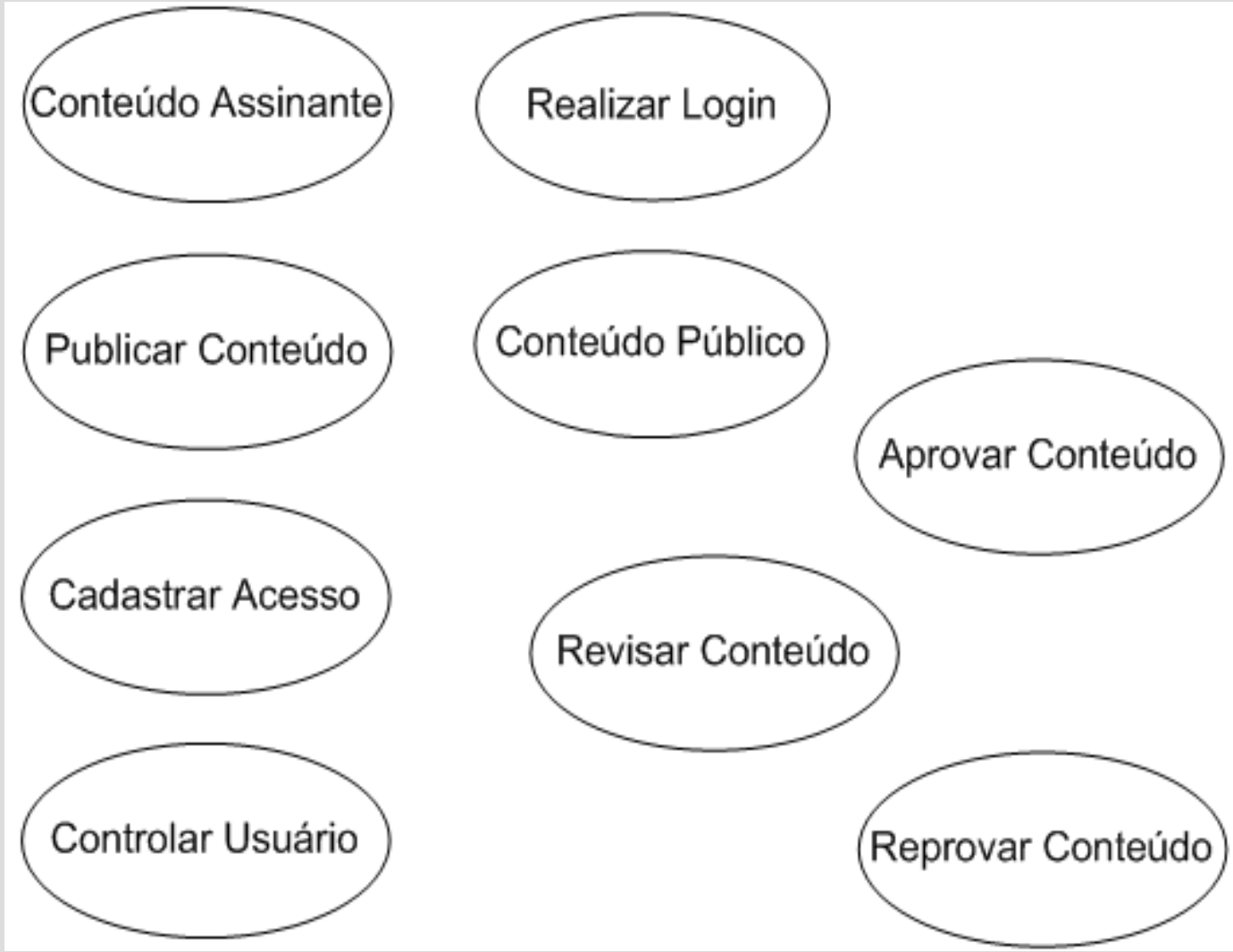
Exercício 1 – Identificando Atores

Quais são os atores deste contexto?



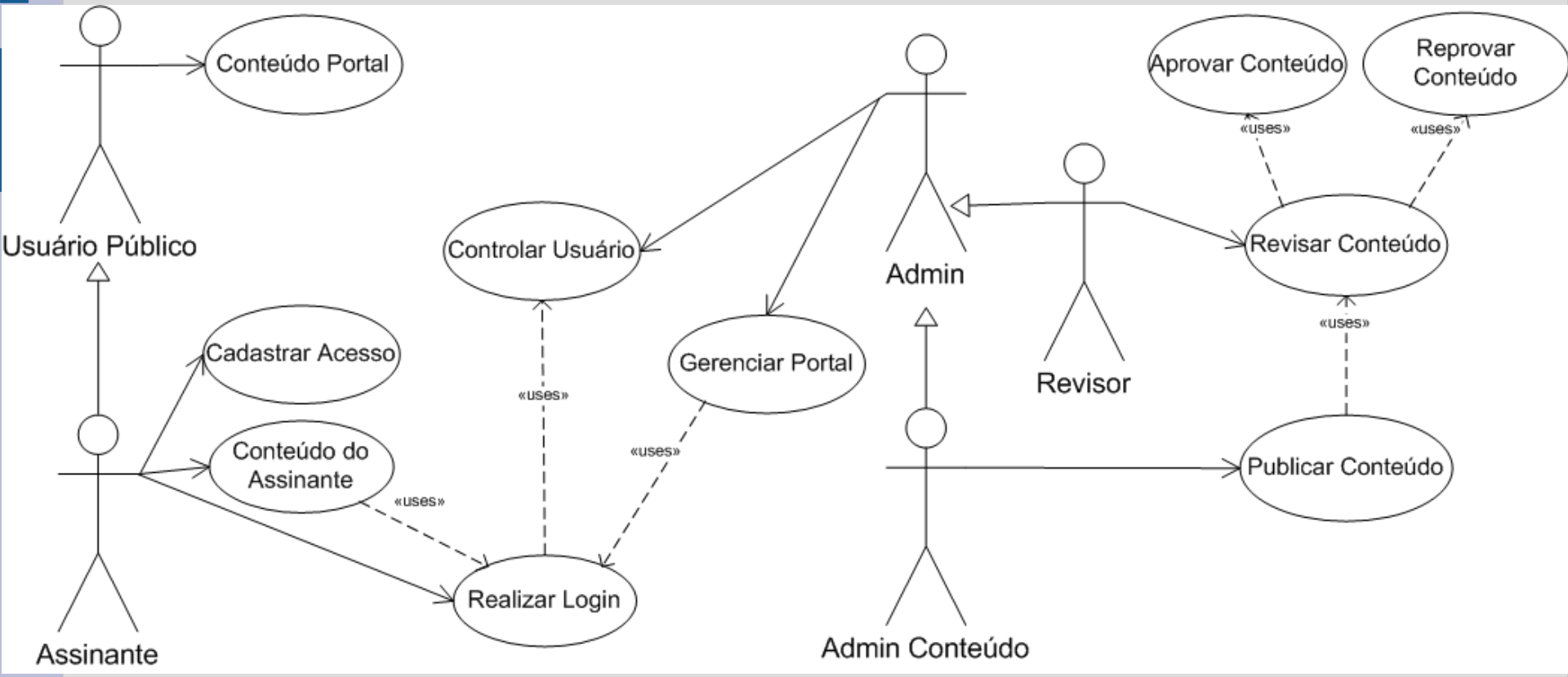
Exercício 1 – Identificando UC

Quais são os casos de uso deste contexto?



Exercício 1 – Diagrama UC

Quais são os relacionamentos entre Ator e UC?

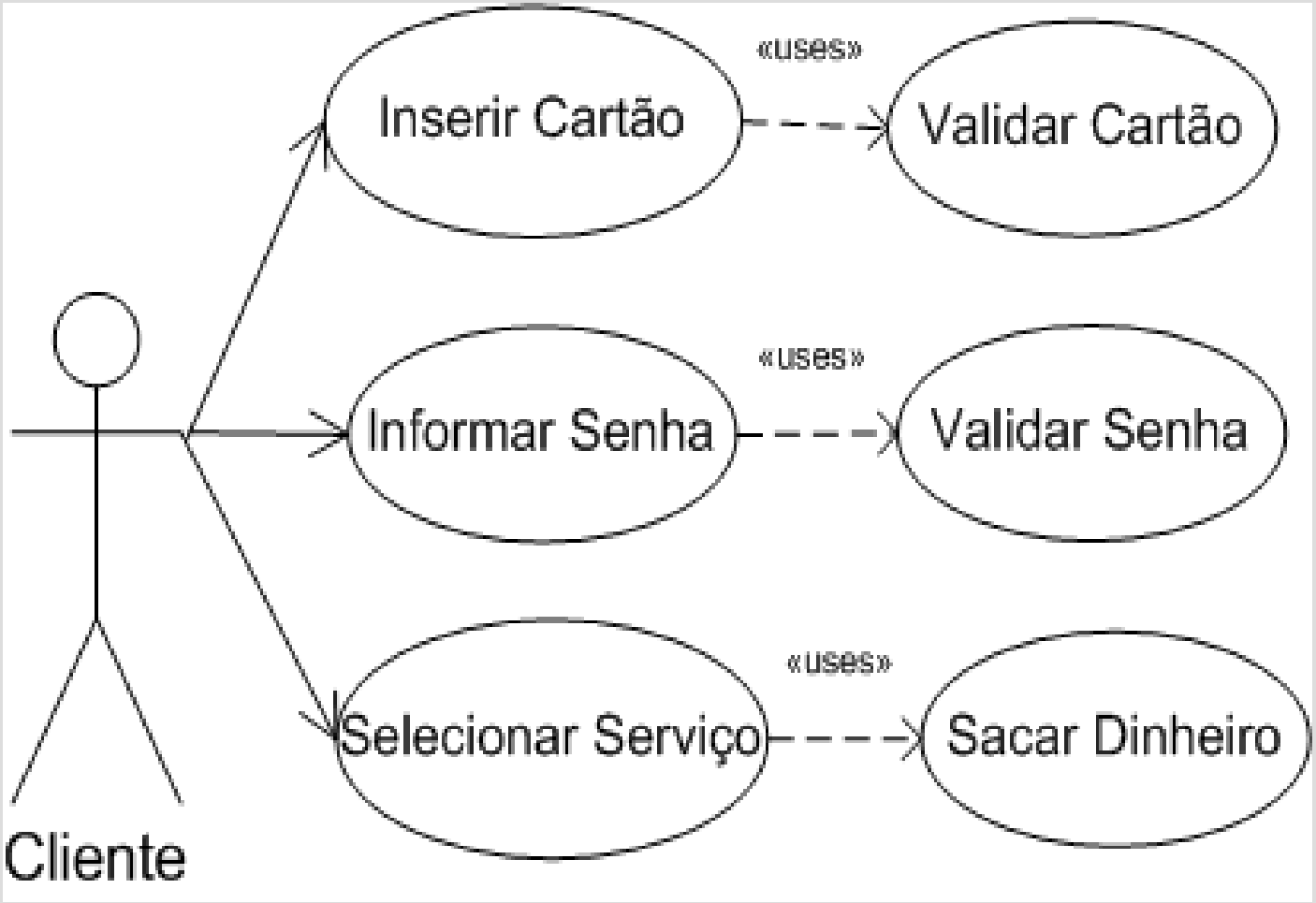


Exercício 2 – Diagrama de UC

– Um esboço inicial sobre Sacar dinheiro seria:

- O processo inicia quando o Cliente insere um cartão no CA (Caixa Automático).
  - Sistema lê e valida informação do cartão.
  - Sistema pede a senha.
  - Cliente entra com a senha.
  - Sistema valida a senha.
  - Sistema pede seleção do serviço.
  - Cliente escolhe “Sacar dinheiro”

Exercício 2 – Diagrama de UC



Finalizamos o *Use Case?*

- No que concerne à introdução, sim, mas veremos mais detalhes e conceitos avançados sobre *Use Case* mais adiante.

- Agora vamos ver quais são os demais diagramas que fazem parte da UML, pois a versão 2.0 trouxe novos diagramas.

A UML 2.0

- A UML 2.0 está em conformidade com a OMG (Object Management Group), que é um consórcio internacional da indústria de software fundado em maio de 1989 com o propósito de criar padrões para possibilitar a interoperabilidade e portabilidade das aplicações distribuídas utilizando a tecnologia de objetos. Diferentemente de outras organizações como a Open Software Foundation (OSF), o OMG não visa a produção de software, mas somente especificações.

A UML 2.0

- Em sua versão 2.0, são oferecidos 13 diagramas:
  - Atividade
  - Classe
  - Comunicação (Colaboração)
  - Componentes
  - Estruturas Compostas
  - Implantação
  - Panorama de Interação
  - Objeto

A UML 2.0

- (Continuação...)
  - Pacote
  - Máquina de Estado
  - Sequência
  - Temporização
  - Caso de Uso

Diagrama de Use Case

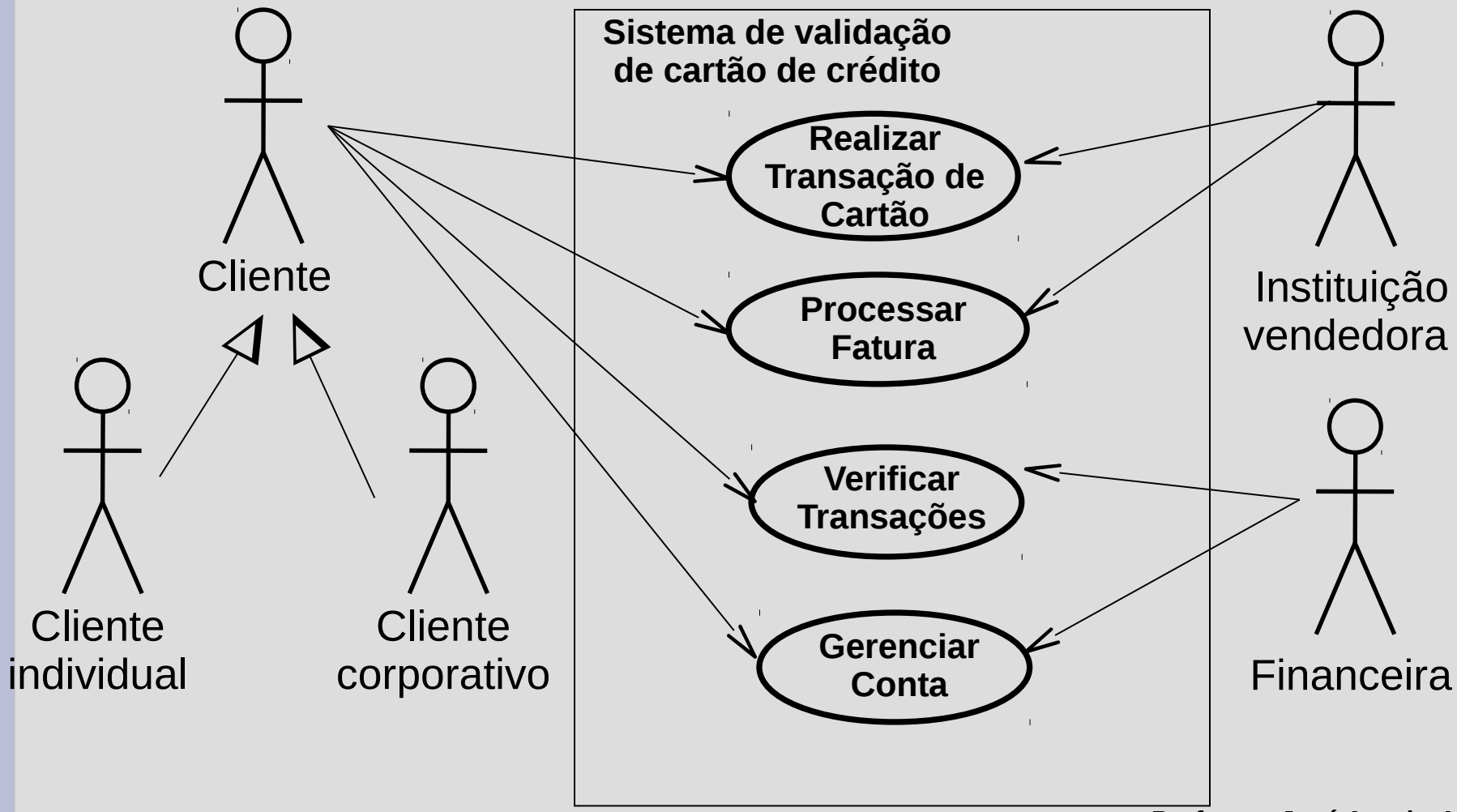
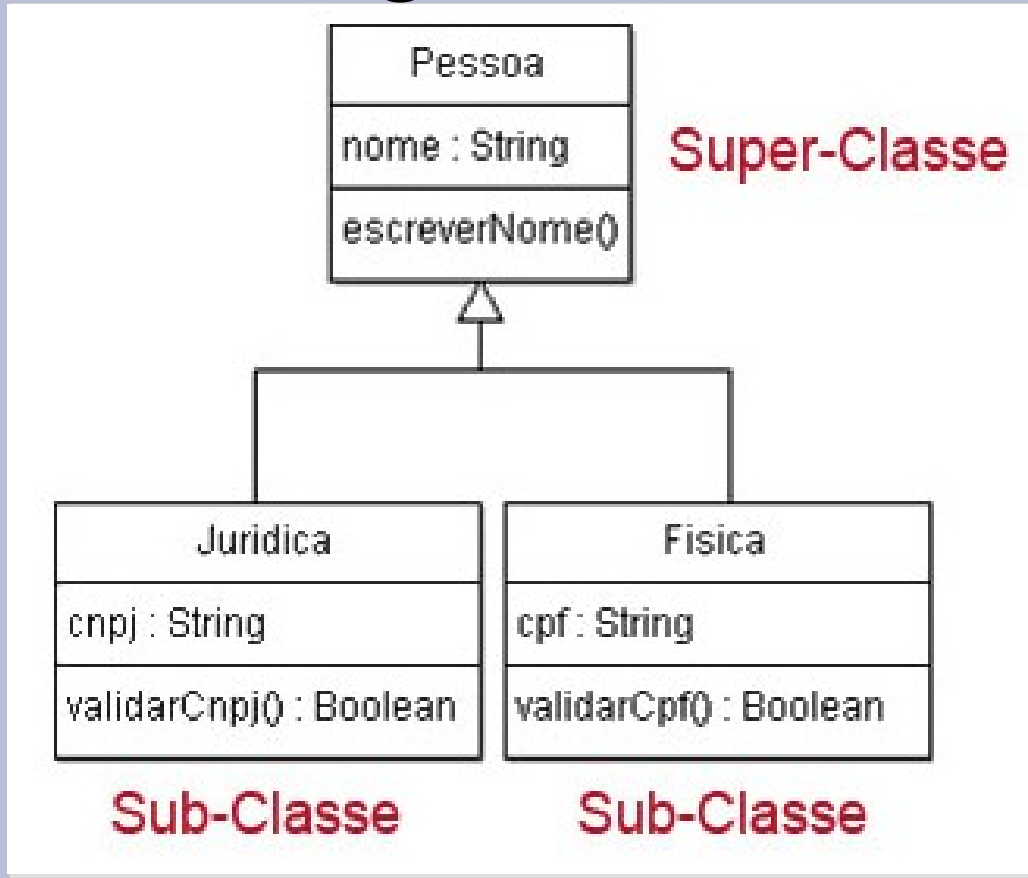


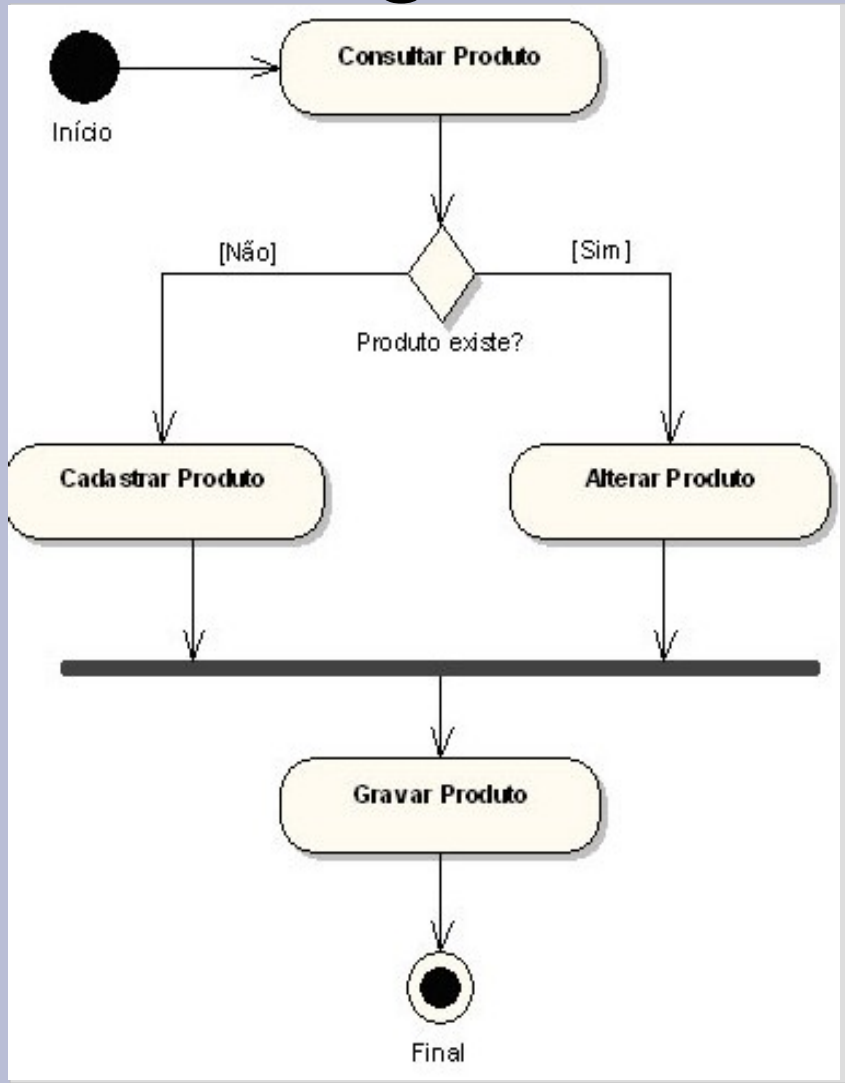
Diagrama de Classe



Uma Classe define a quantidade de instâncias ou ocorrências de um determinado objeto.

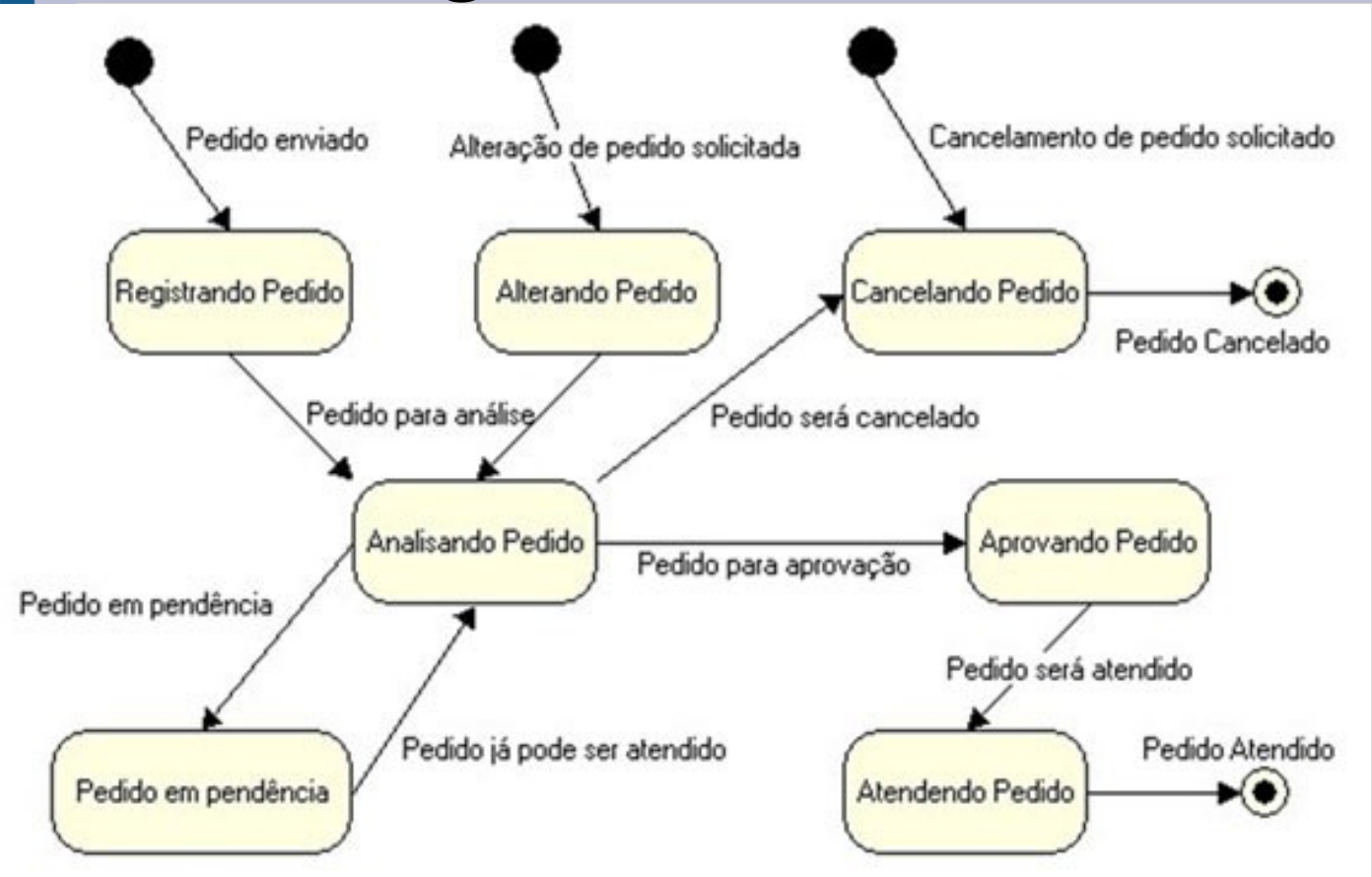
Define também as características e comportamentos de um determinado objeto.

Diagrama de Atividade



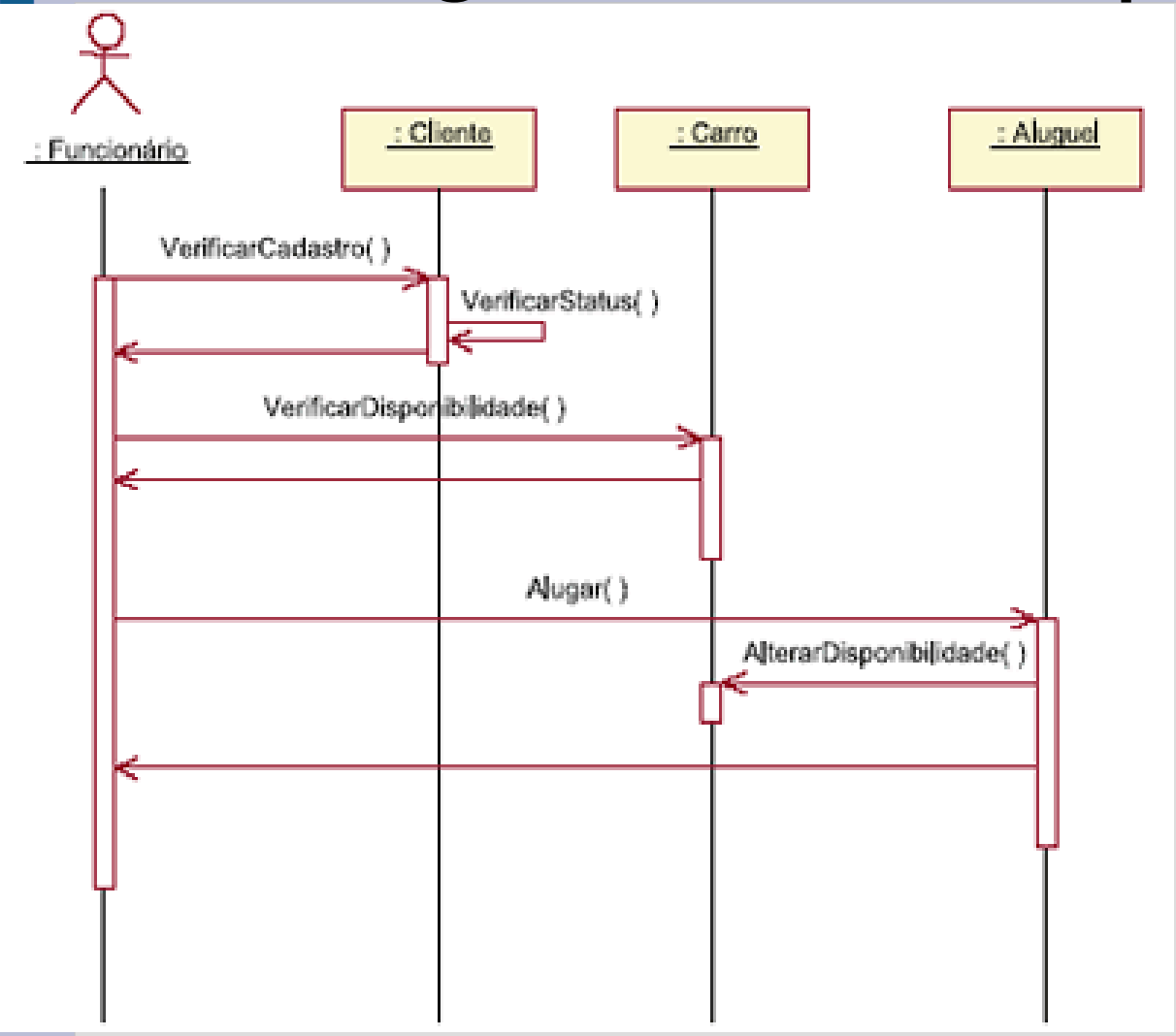
Uma Atividade é essencialmente parte de um fluxo, que interage com uma outra atividade, podendo ser paralela a ela ou não.

Diagrama de Estado



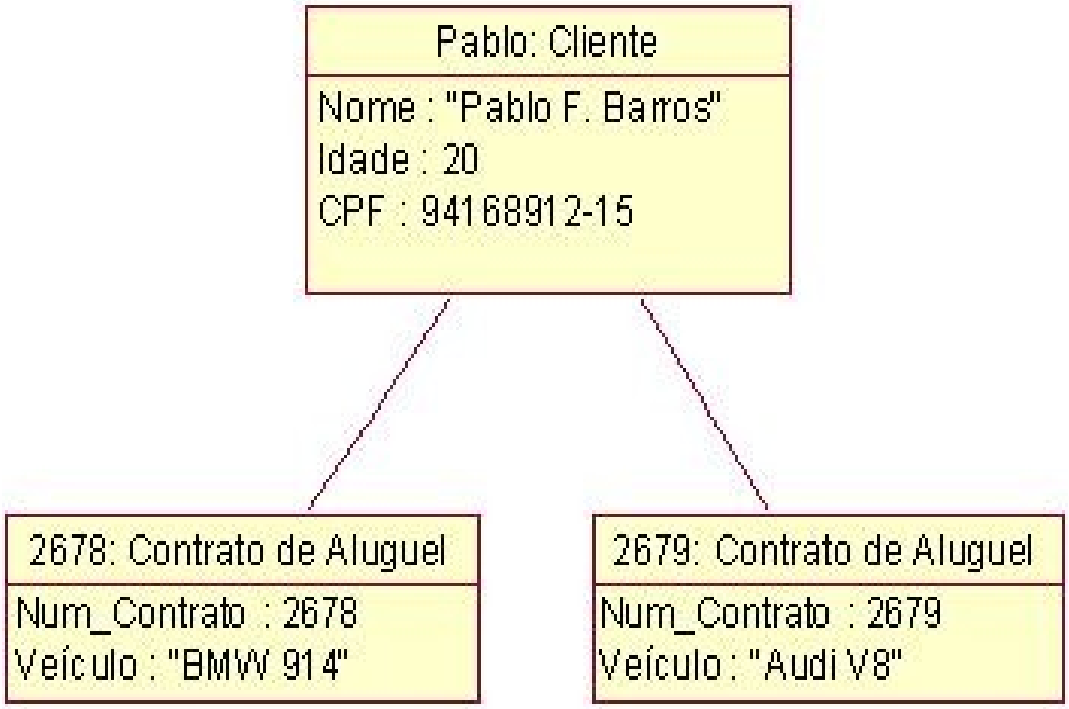
Um estado é uma condição ou situação na vida de um objeto durante a qual o objeto satisfaz alguma condição, realiza uma atividade ou aguarda um evento.

Diagrama de Sequência



O diagrama de sequência descreve a maneira como os grupos de objetos colaboram com algum comportamento ao longo do tempo.

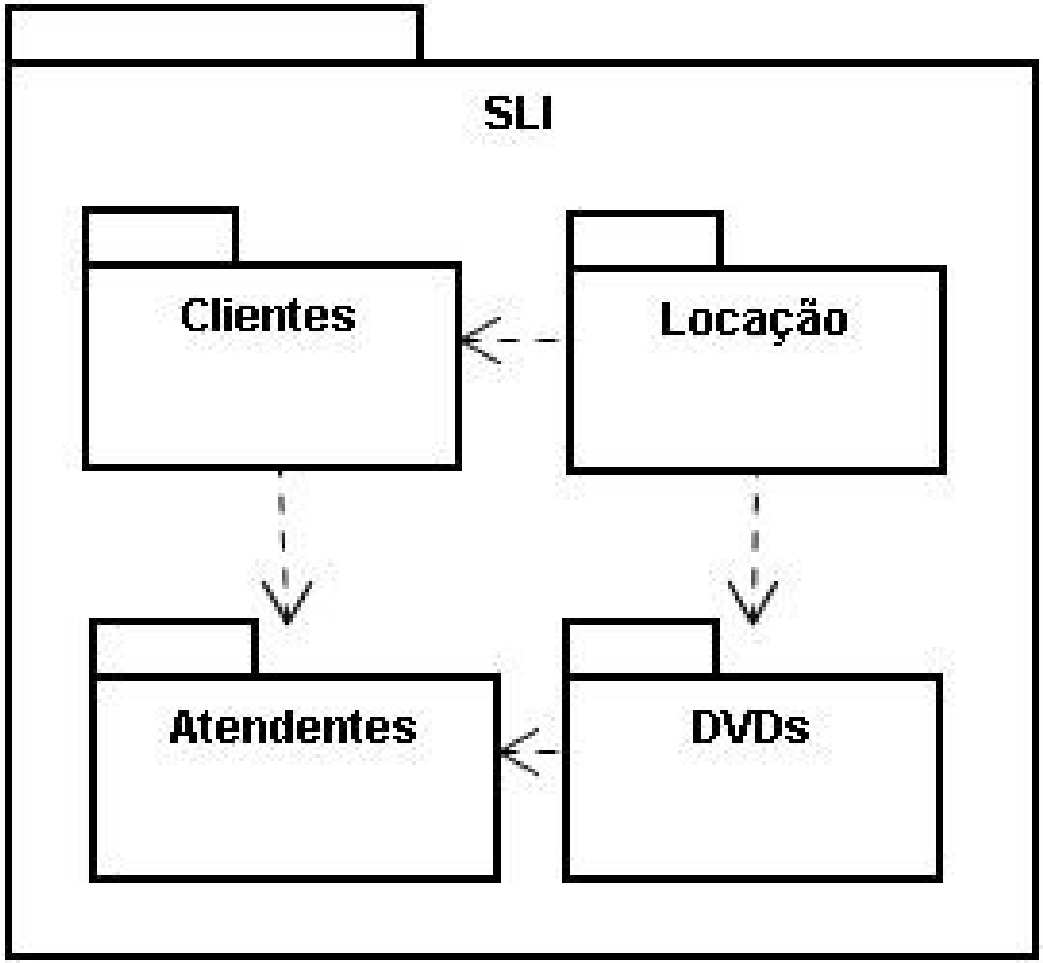
Diagrama de Objetos



O diagrama de objetos mostra uma visão do relacionamento entre as instâncias (ocorrências) entre os objetos.

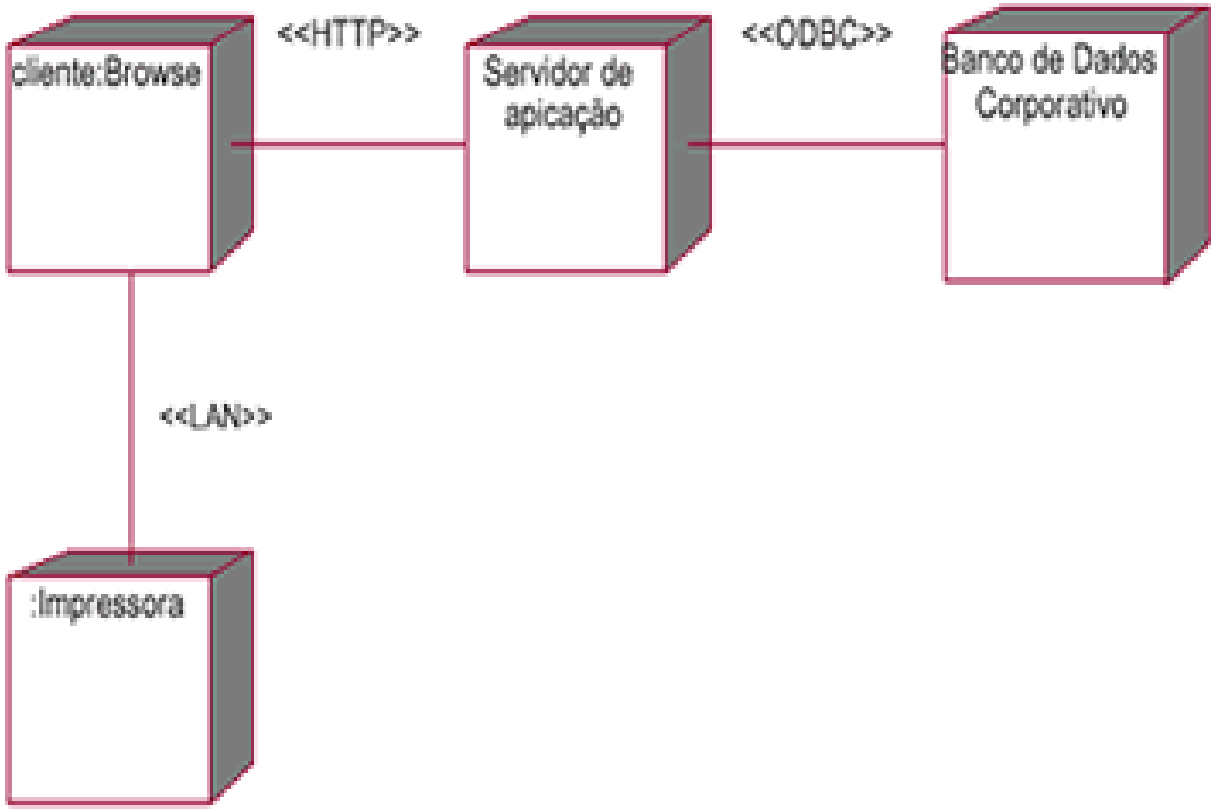
A visão ao lado mostra todos os objetos “contrato” do objeto “Pablo”.

Diagrama de Pacotes



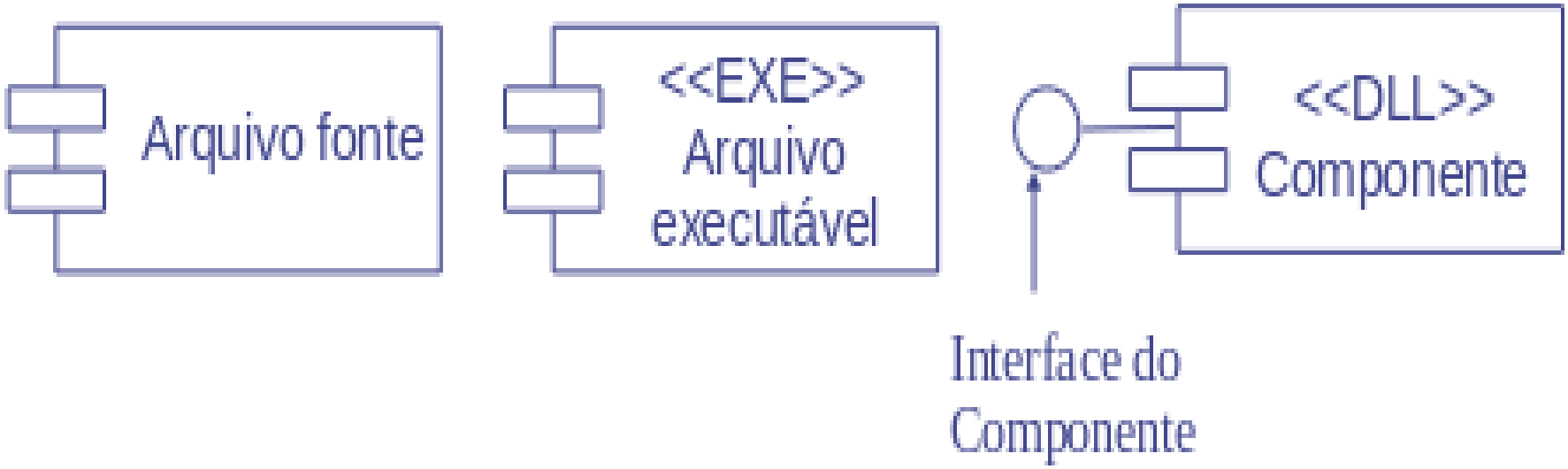
O diagrama de pacotes ou módulos descreve os pedaços do software que estão relacionados e suas dependências. Um sistema de ERP (Enterprise Resource Planning) possui diversos módulos de gestão corporativa.

Diagrama de Implantação



Um diagrama de implantação modela o inter-relacionamento entre recursos de infra-estrutura, de rede ou artefatos de sistemas. Normalmente representamos servidores neste diagrama.

Diagrama de Componentes



O diagrama de componentes ilustra a organização e dependências entre componentes de software.

É um diagrama que provê a realização de (uma/um conjunto de) interface(s).

Diagrama de Componentes

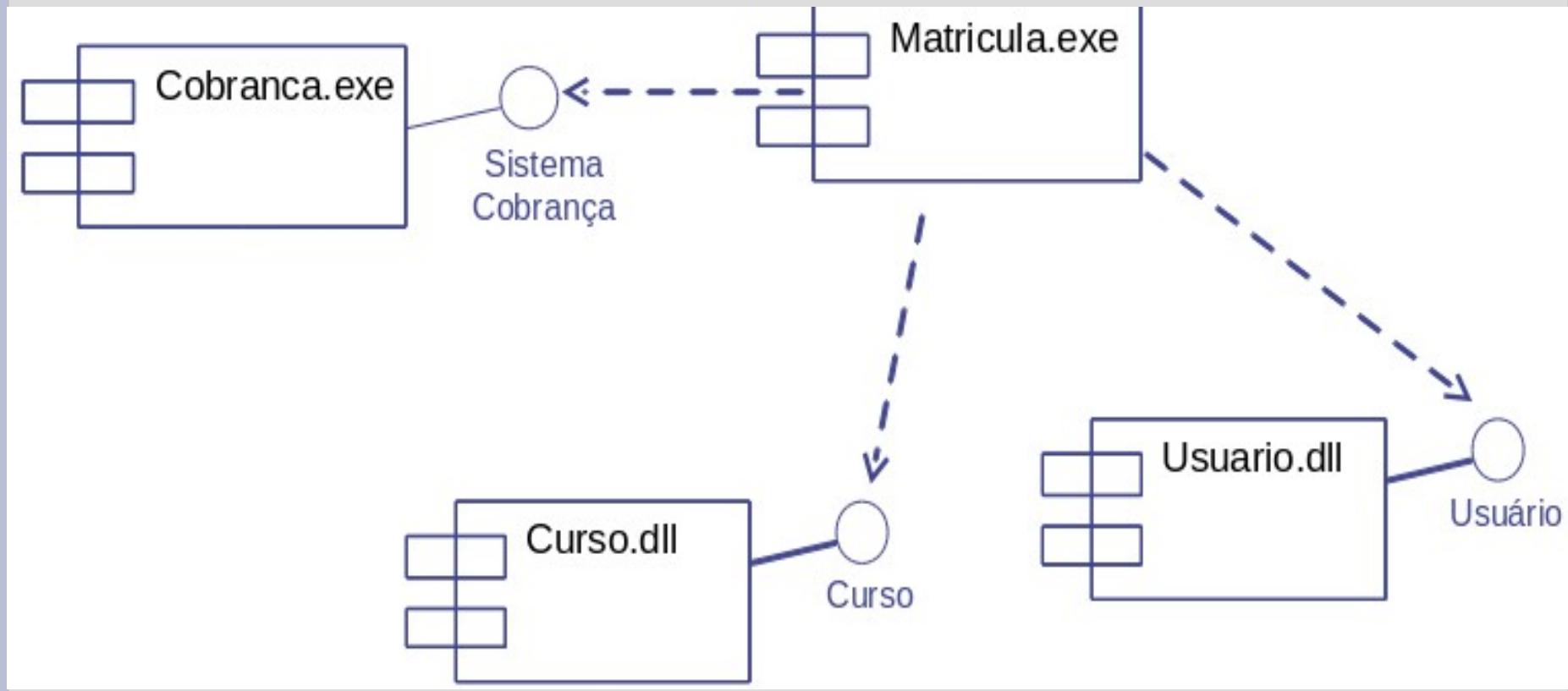
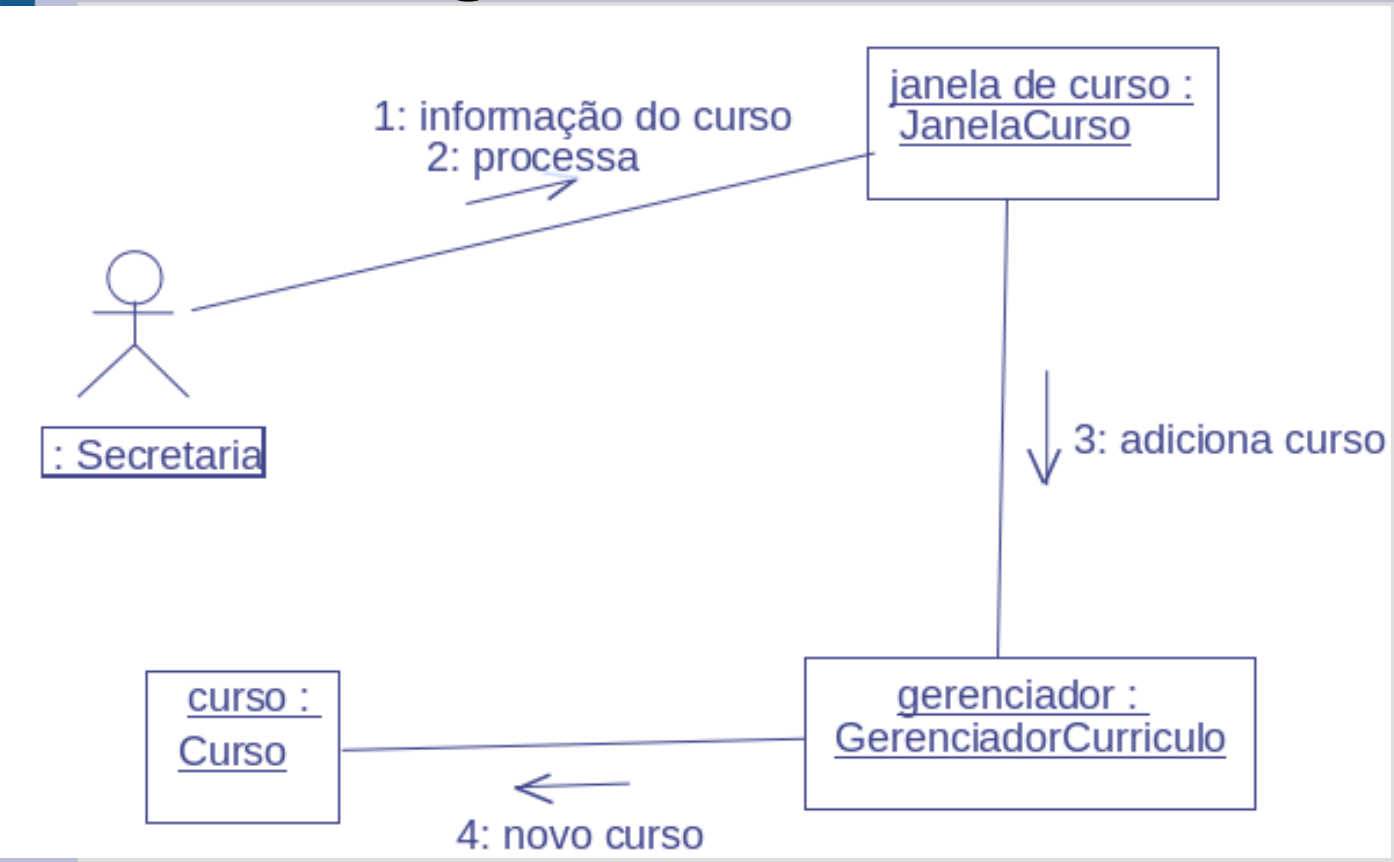


Diagrama de Comunicação



O diagrama de comunicação (colaboração) ilustra todo o processo de comunicação (troca de mensagens) entre os objetos de software.

Exercício de Use Case - 1

Para o enunciado abaixo, faça:

1-Definir quem é (são) o(s) ator(es).
2-Definir quais são os casos de uso (requisitos funcionais).
3-Elabore o diagrama de caso de uso.

O usuário Professor de Modelagem de Sistemas deseja abrir a sua planilha eletrônica, onde relaciona as notas e faltas dos alunos, para registrar presença dos alunos, salvar as atualizações e imprimir.

Mas antes, o professor terá que informar a senha para poder abrir a planilha eletrônica, pois a mesma está protegida por senha.

Atividade em grupo (jogo de 40s)

1 – É o elemento da UML que recebe as características e comportamentos.
Classe-filha ou sub-classe
2 – É o diagrama usado na UML que especifica os principais requisitos do projeto.
Use case
3 – Representam as características de um objeto.
Atributos
4 – Representam o número de ocorrências de um determinado objeto.
Instâncias.
5 – Ele possui características e comportamentos muito claros na modelagem.
Objeto.
6 – Pode representar um sistema externo ao projeto de software.
Ator.
7 – Representa o desenvolvimento do código-fonte no processo RUP.
Construção
8 – É o elemento utilizado para representar uma base de dados na análise essencial.
Depósito de Dados.
9 – É a técnica de levantamento de requisitos conhecida como chuva de ideias.
Brainstorming.
10 – É o nome dado aos principais interessados no projeto.
Stakeholders.

Atividade em grupo (jogo de 40s)

11 – É o documento utilizado para formalizar que o cliente está de acordo com o projeto entregue.
Documento de aceite do projeto.
12 – É o nome dado aos requisitos que têm foco em usabilidade.
Requisitos não-funcionais.
13 – É o diagrama na UML comparado ao antigo fluxograma.
Diagrama de Atividades.
14 – É o nome dado à classe que ilustra um tipo específico de objeto.
Especialização.
15 – É o nome dado ao diagrama na UML responsável pela estrutura física dos objetos e o relacionamentos das instâncias.
Diagrama de classes.
16 – É o documento utilizado para registrar as decisões e discussões da reunião.
Ata de reunião.

Exercício de Use Case - 2

- Modelar o diagrama de caso de uso para:

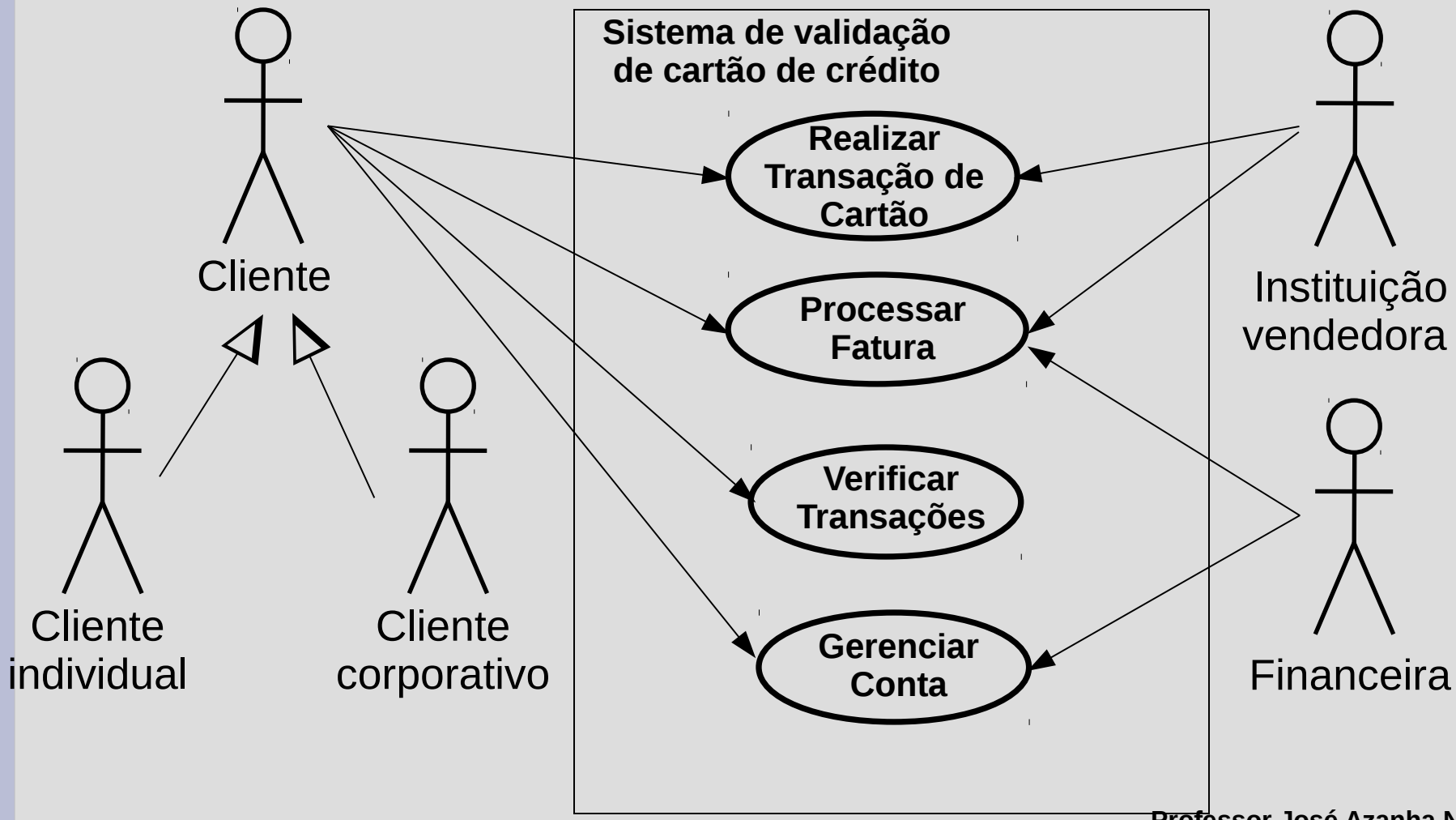
  Projeto: Portal de Material Didático

  O professor deverá se identificar para incluir os arquivos referentes ao material didático.

  O aluno deverá se identificar para ter acesso a listagem de materiais disponíveis. Ao escolher o material, o aluno poderá ver o material diretamente na página, mas poderá também, a partir desta página, fazer o download e imprimir.

Retornando ao...

Diagrama de Use Case

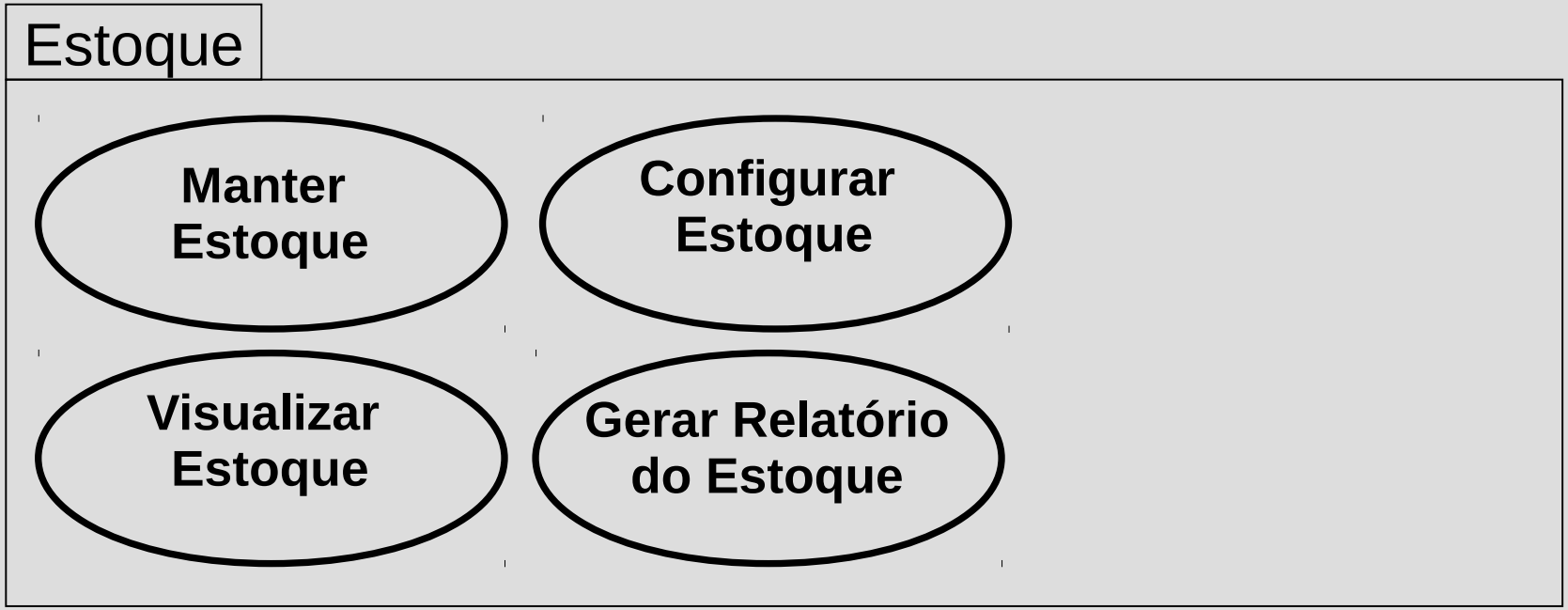


Pacote de *Use Case*

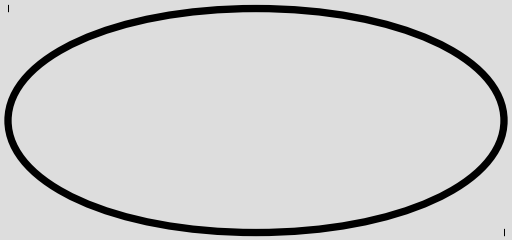
- Primeiro esforço de estruturação
- Agrupam-se use cases relacionados em único *container*

Pacote de *Use Cases*

- Pode ser útil para distribuir trabalho para sub-grupos de trabalho.



Pacote de *Use Cases*



Clientes :: Atendimento

Reuso em *Use Cases*

- Comportamento comum a mais de dois *use cases* (ou forma parte independente)
  - Pode-se modelar como *use case* para ser reusado
- Há três possibilidades
  - Inclusão (uso)
  - Extensão
  - Generalização/Especialização

Inclusão <<include>> estereótipo

- O estereótipo «include» indica que um caso inclui o outro.

- Permite fatorar comportamento comum a vários casos.
  - Comum/idêntico
  - Sempre usado

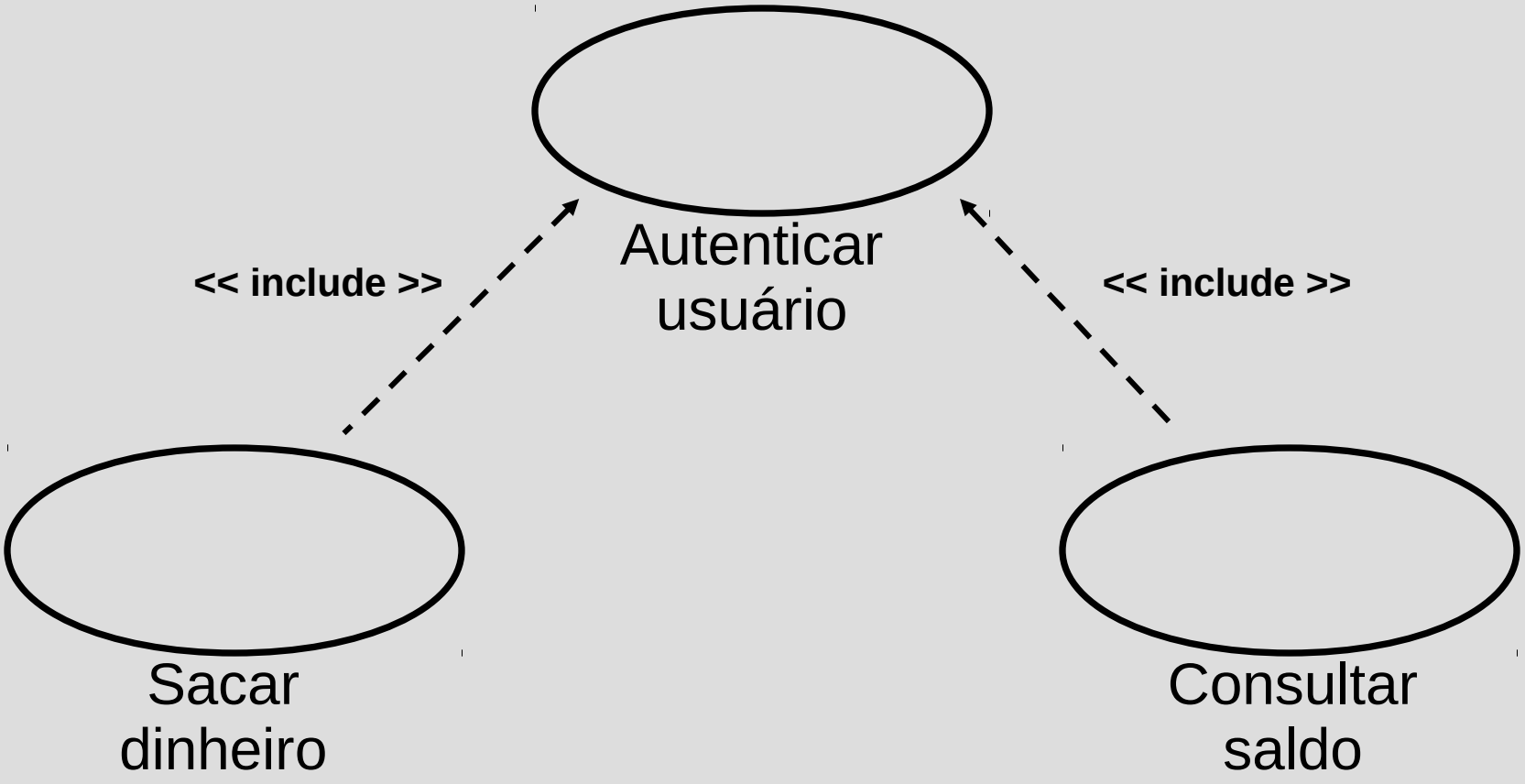
Inclusão <<include>> estereótipo

- Equivalente a fatoração feita em programação e SOs através de sub-programas
  - #include a linguagem C, C++, Java, entre outras. Utilizado também em arquivos de configuração de Servidores Linux.

Inclusão <<include>> estereótipo

- Como exemplo, tanto “Sacar dinheiro” quanto “Consultar saldo” necessitam da senha
  - Pode-se criar novo use case “Autenticar usuário” e incluí-lo
- Mas Atenção!
  - NÃO SE DEVE CRIAR USE CASES MÍNIMOS
  - Deve haver ganho no reuso

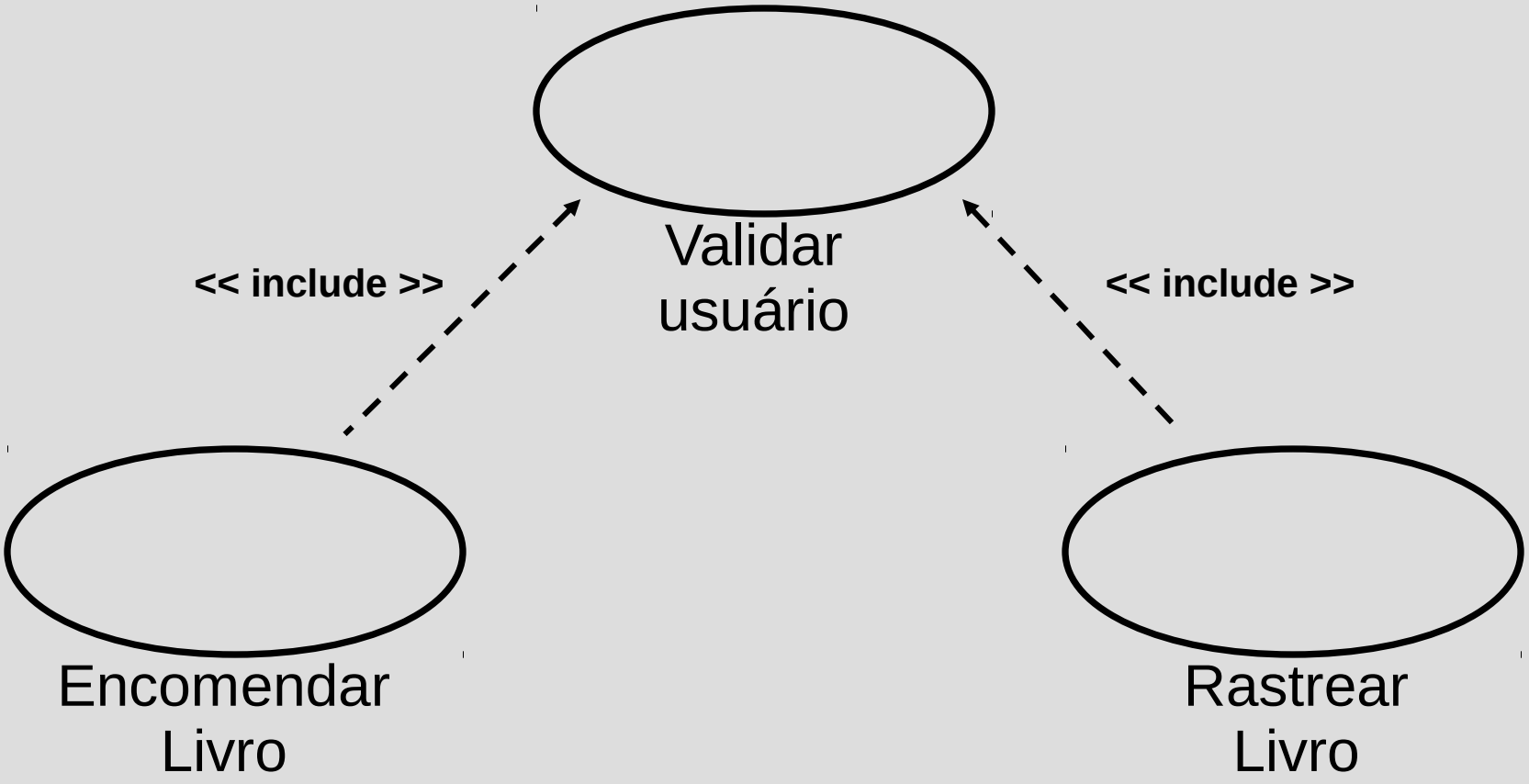
Inclusão <<include>> estereótipo



Inclusão <<include>> estereótipo

- Descrição de Consultar saldo
  - Fluxo de Eventos Principal:
    - include (Autenticar usuário). Sistema pede ao Cliente que selecione o tipo de conta (corrente, etc).

Inclusão <<include>> estereótipo



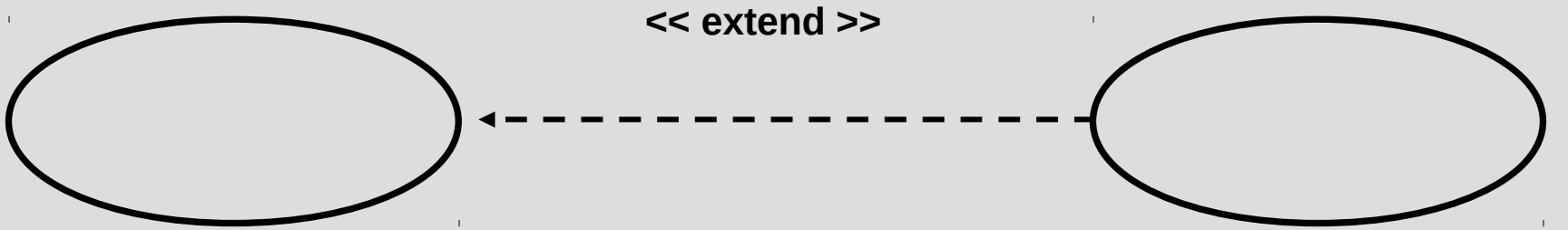
Extensão <<extend>> estereótipo

- Pode-se usar o estereótipo «extend» para indicar que um caso estende o outro.
- Útil para fatorar comportamento incomum/não-padrão.
- Use case pode ser estendido por outro
  - Extensão de funcionalidade/Caso excepcional

Extensão <<extend>> estereótipo

- Extensão ocorre em pontos específicos
  - Pontos de extensão
- Há também inclusão de texto (fluxo de eventos)
  - Porém sob condições particulares
- Pode ser usada para
  - Simplificar fluxos de eventos complexos
  - Representar comportamentos opcionais
  - Lidar com exceções

Extensão <<extend>> estereótipo



Fazer Atendimento

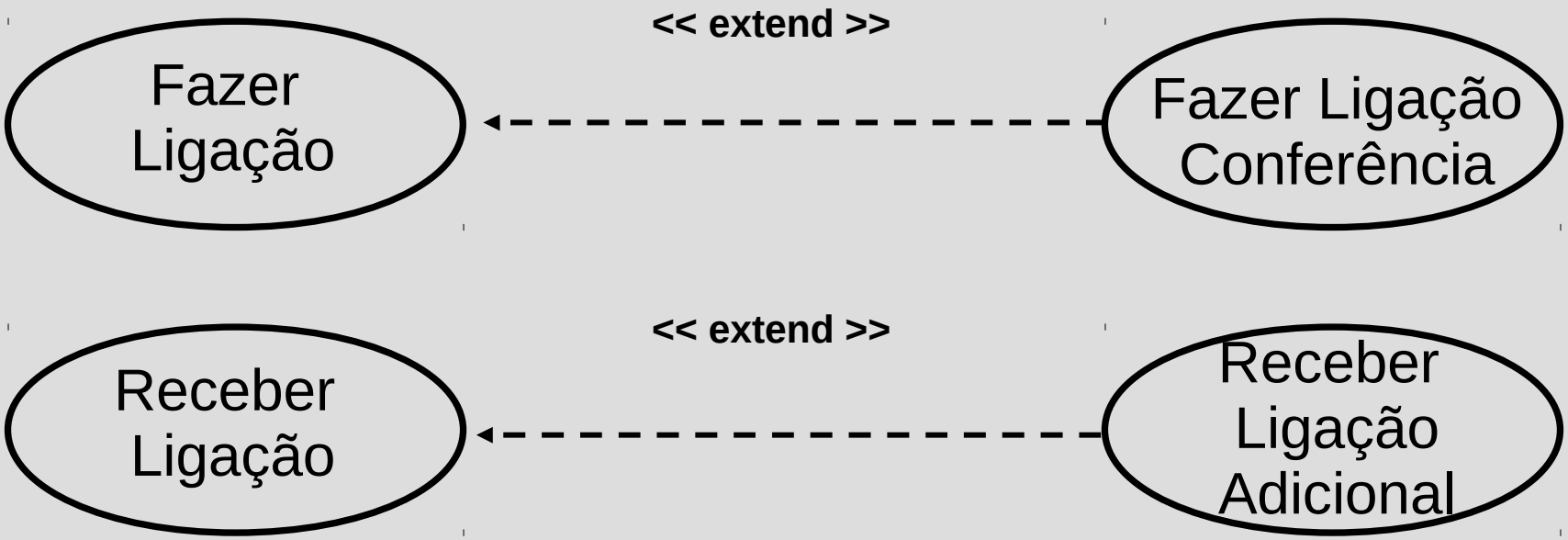
Fazer Atendimento de Urgência

Pontos de extensão urgente

Extensão <<extend>> estereótipo

- Descrição de Atendimento
  - Fluxo de Eventos Principal:
    - Colete os itens do pedido (urgente). Submeta pedido para processamento.

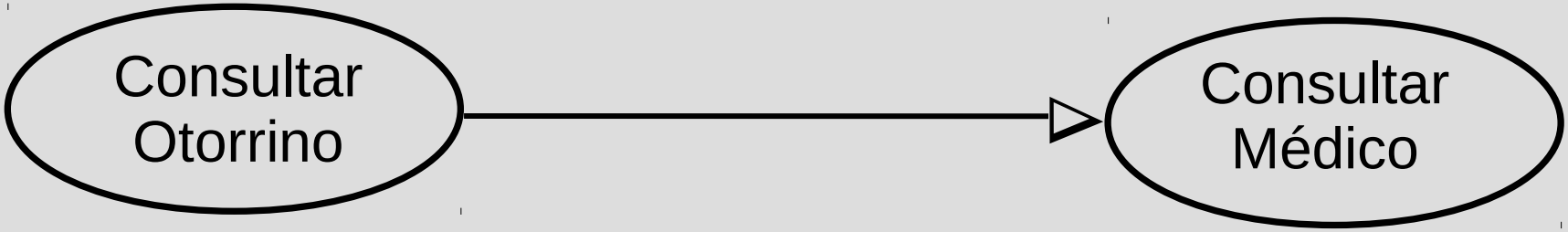
Extensão <<extend>> estereótipo

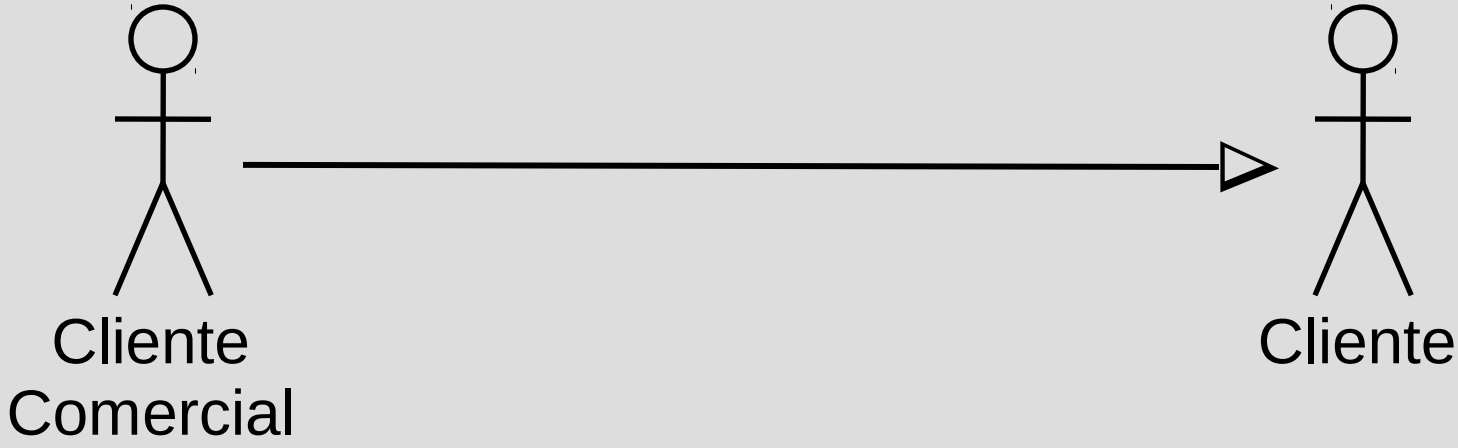


Especialização

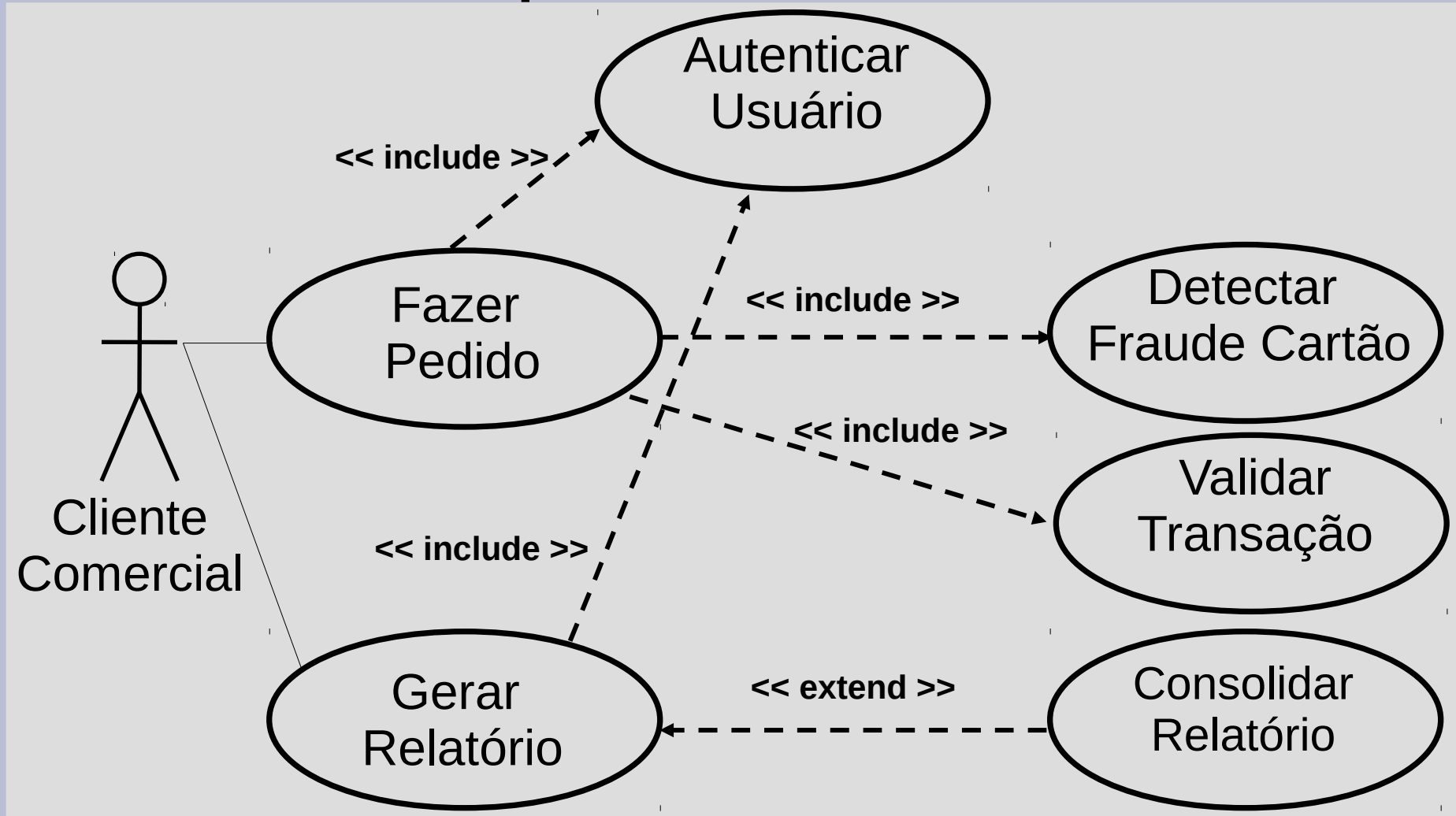
- Use case pode especializar outro
  - Adição/refinamento do fluxo de eventos original
- Especialização permite modelar comportamento de estruturas de aplicação em comum

Especialização





Caso de Uso com Estereótipos



Classes e Relacionamentos

- Os relacionamentos são conexões entre os elementos da análise. Na orientação a objetos, os relacionamentos mais importantes são 3 e conhecidos como:
  - Associação
    - Multiplicidade;
    - Agregação;e
    - Composição.
  - Dependência
  - Generalização

Classes e Relacionamentos

- Associação
  - É um relacionamento estrutural que especifica objetos de um item conectados a objetos de outro item. A partir de duas classes, você é capaz de navegar do objeto de uma classe até o objeto de outra classe e vice-versa.
  - A associação é representada por uma linha sólida conectando duas classes.

Classes e Relacionamentos

- Associação - Simples

Exemplo 1

Exemplo 2

Classes e Relacionamentos

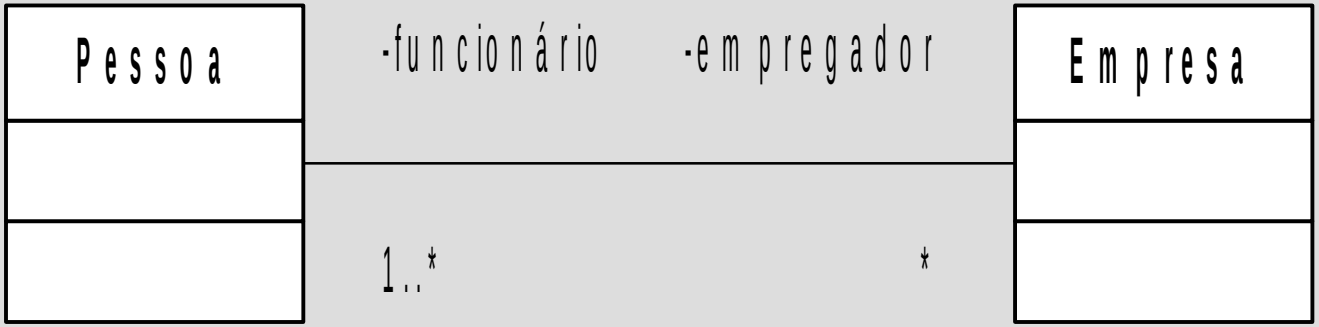
- Associação – Multiplicidade
  - Representa um relacionamento estrutural entre objetos. Em muitas situações na modelagem, é importante determinar a quantidade de objetos que podem ser conectados pela instância de uma associação.
  - Essa "*quantidade*" é chamada de multiplicidade do papel e é expresso em intervalo de valores.

Classes e Relacionamentos

- Associação – Multiplicidade
  - O intervalo de valores máximo para o relacionamento de associação – multiplicidade é relacionado a seguir:
    - → 1
    - → *
    - → 0..*
    - → 0..1
    - → 1..1
    - → 1..*

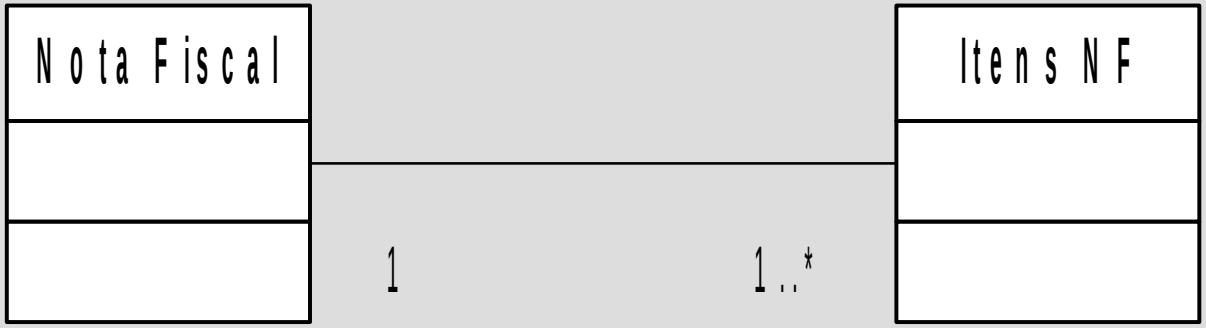
Classes e Relacionamentos

- Associação – Multiplicidade



Exemplo 1

Exemplo 2

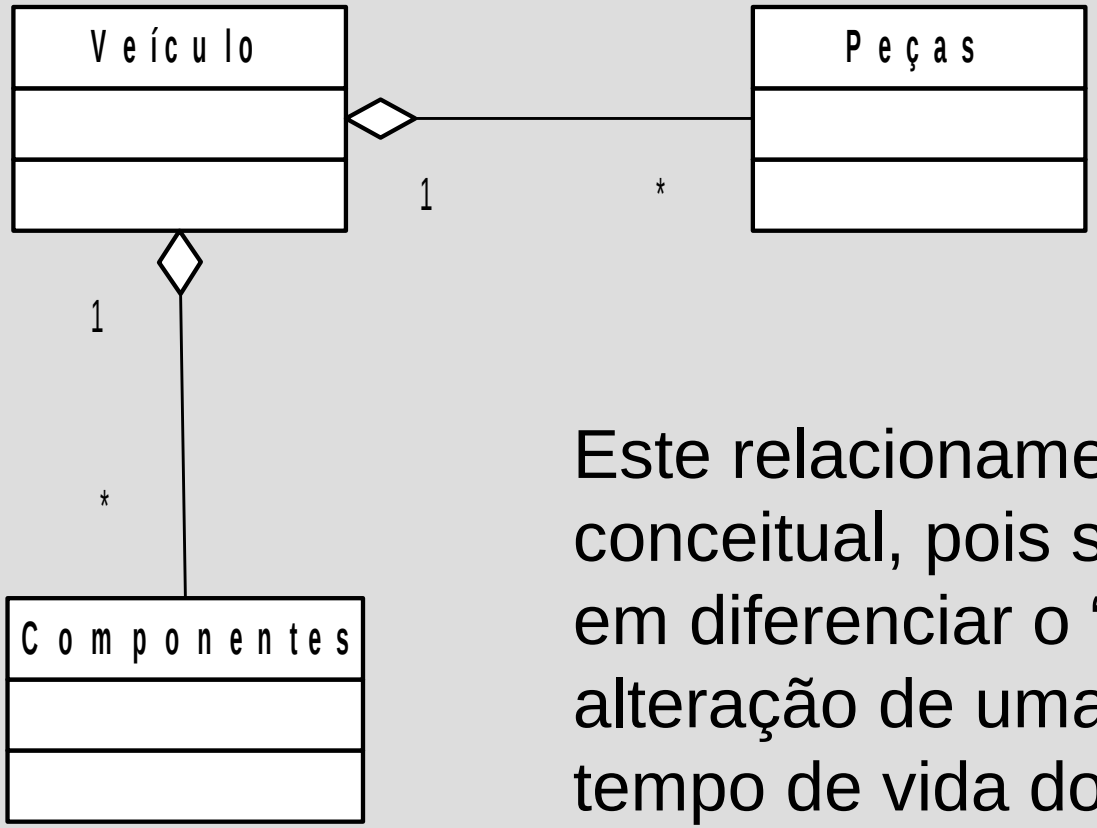


Classes e Relacionamentos

- Associação – Agregação
  - Representa uma pura associação entre duas classes onde uma delas representa “o todo” e outra “as partes” ou todo-parte, ou seja, a parte *agrega* ao todo.
  - É um tipo de associação simples com um diamante aberto na extremidade do todo.
  - Exemplo: Família Silva representa “o todo” e o João Silva representa “a parte”.

Classes e Relacionamentos

- Associação – Agregação



Este relacionamento é apenas conceitual, pois se preocupa apenas em diferenciar o “todo” da “parte” e a alteração de uma classe não altera o tempo de vida do todo e suas partes.

Classes e Relacionamentos

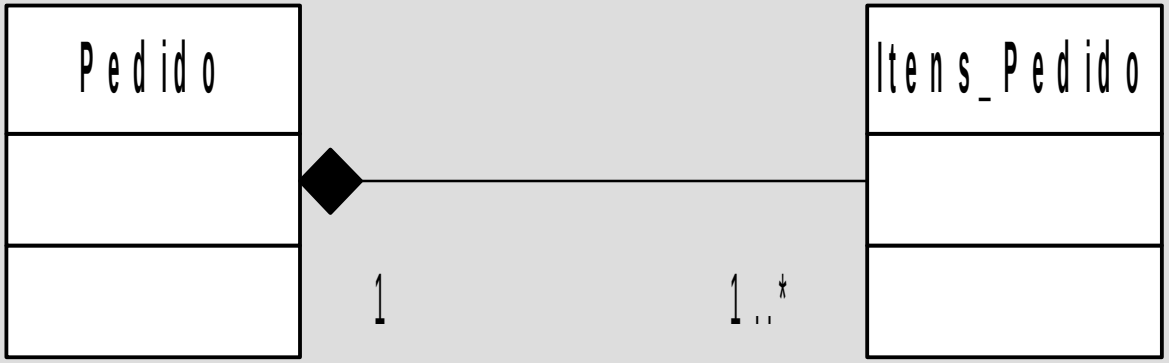
- Associação – Composição
  - Representa uma forma de agregação com propriedade bem-definida e tempo de vida coincidente como parte do todo. Uma vez a multiplicidade criada entre as classes, as partes vivem e morrem com ela.
  - Essas partes também podem ser removidas antes da morte do objeto composto.

Classes e Relacionamentos

- Associação – Composição

Exemplo 1

Exemplo 2

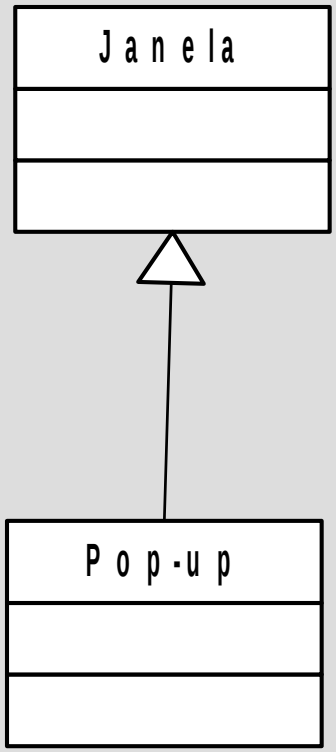


Classes e Relacionamentos

- Especialização
  - Representa um tipo de relacionamento de generalização e especialização, onde a classe-mãe representa a classe genérica e as classes-filhas representam as classes específicas.
  - É importante destacar que as classes-filhas herdam atributos e operações da classe-mãe.

Classes e Relacionamentos

- Especialização



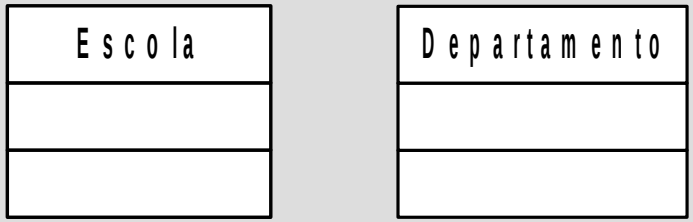
A Janela de Pop-up não pode ser ajustada quanto ao seu tamanho e obriga a resposta do usuário para continuar utilizando a aplicação. Mas a janela Pop-up possui características e comportamentos de uma Janela comum, mas vice-versa nunca ocorre.

Classes e Relacionamentos

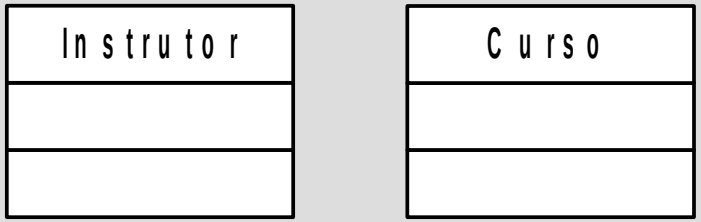
- Exercícios

  Fazer o relacionamento de associação (multiplicidade, agregação e composição) das classes abaixo.

1)



2)



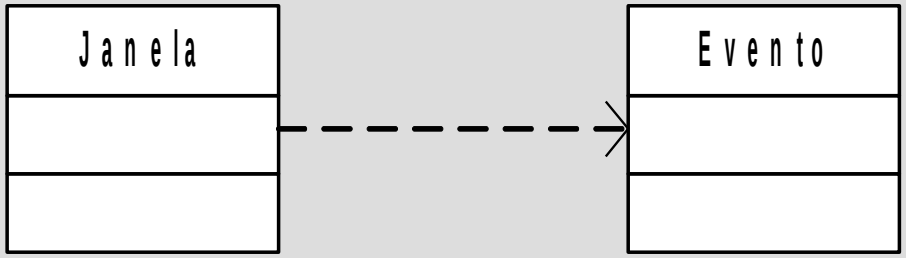
Classes e Relacionamentos

- Dependência
  - Representa um relacionamento de utilização determinando as modificações na especificação de um item. A dependência é representada graficamente com uma linha tracejada.
  - Usamos a dependência quando uma classe usa outra como argumento na assinatura de uma operação.

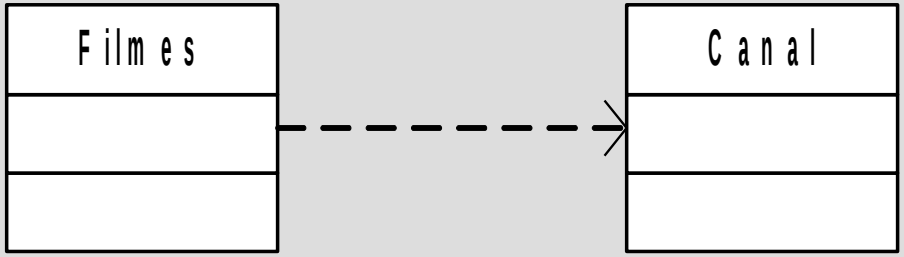
Classes e Relacionamentos

- Exemplos

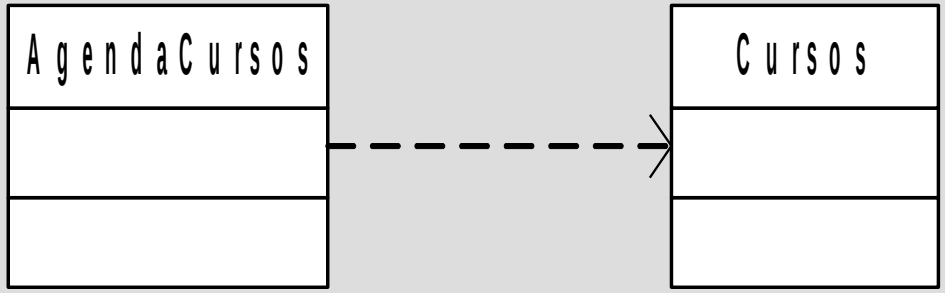
1)



2)



3)

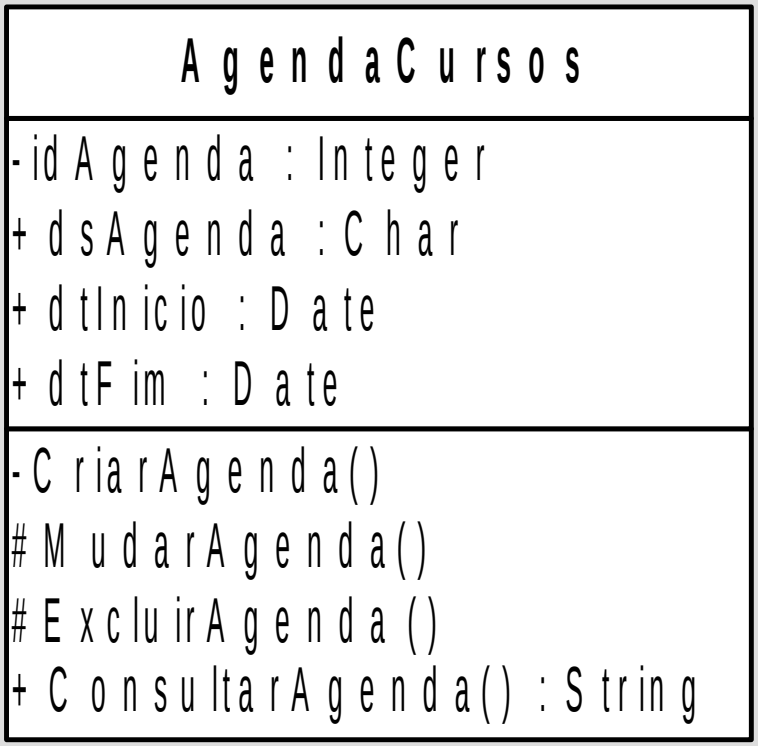


Estrutura das Classes

- As classes são estruturadas da seguinte forma:
  - Atributo: determina as características do objeto. O atributo receberá um valor que deve ser especificado (int, char, etc).
  - Operação: determina as operações (ações) do objeto. A operação pode receber um valor que deve ser especificado (int, char, etc).
  - Visibilidade: determina o modo de visualização das operações e atributos do objeto/classe.

Estrutura das Classes

- Exemplo



Estrutura das Classes

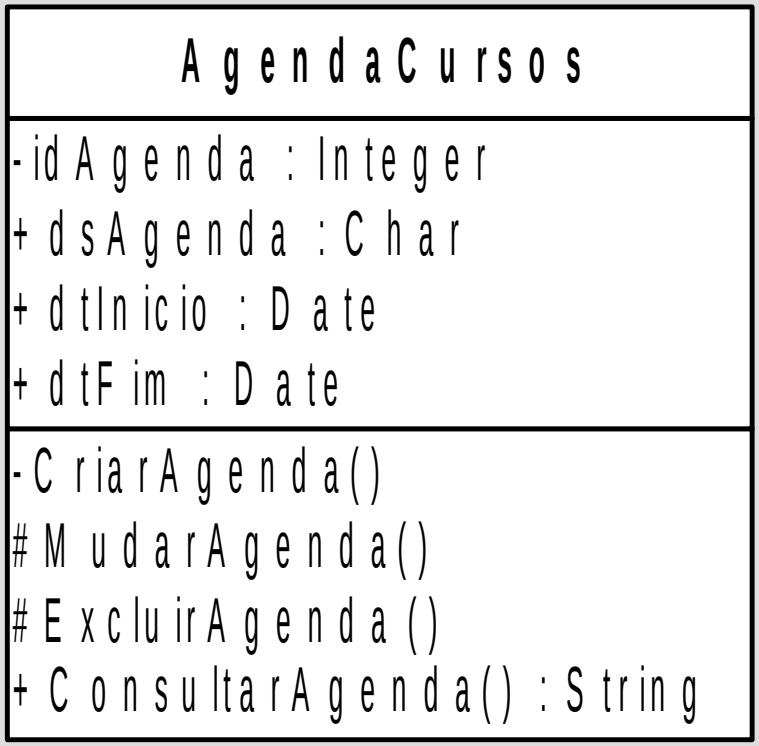
- Visibilidade
  - É uma enumeração de valores (public, protected e private) que indicam se o elemento da modelagem ao qual se referem podem ser vistos fora de seu espaço de nome (domínio).
    - Público (public): qualquer classe externa com relacionamento com a classe que possui a característica, será capaz de usar. Neste caso, a característica é antecedida pelo sinal de '+'.

Estrutura das Classes

- Visibilidade
  - Protegido (protected): qualquer classe que seja descendente da classe cujo a característica esteja protegida, poderá visualizar. Neste caso, a característica é antecedida pelo sinal de '#'.
  - Privado (private): somente a classe dona da característica poderá visualizar. Neste caso, a característica é antecedida pelo sinal de '-'.

Estrutura das Classes

- Exemplo



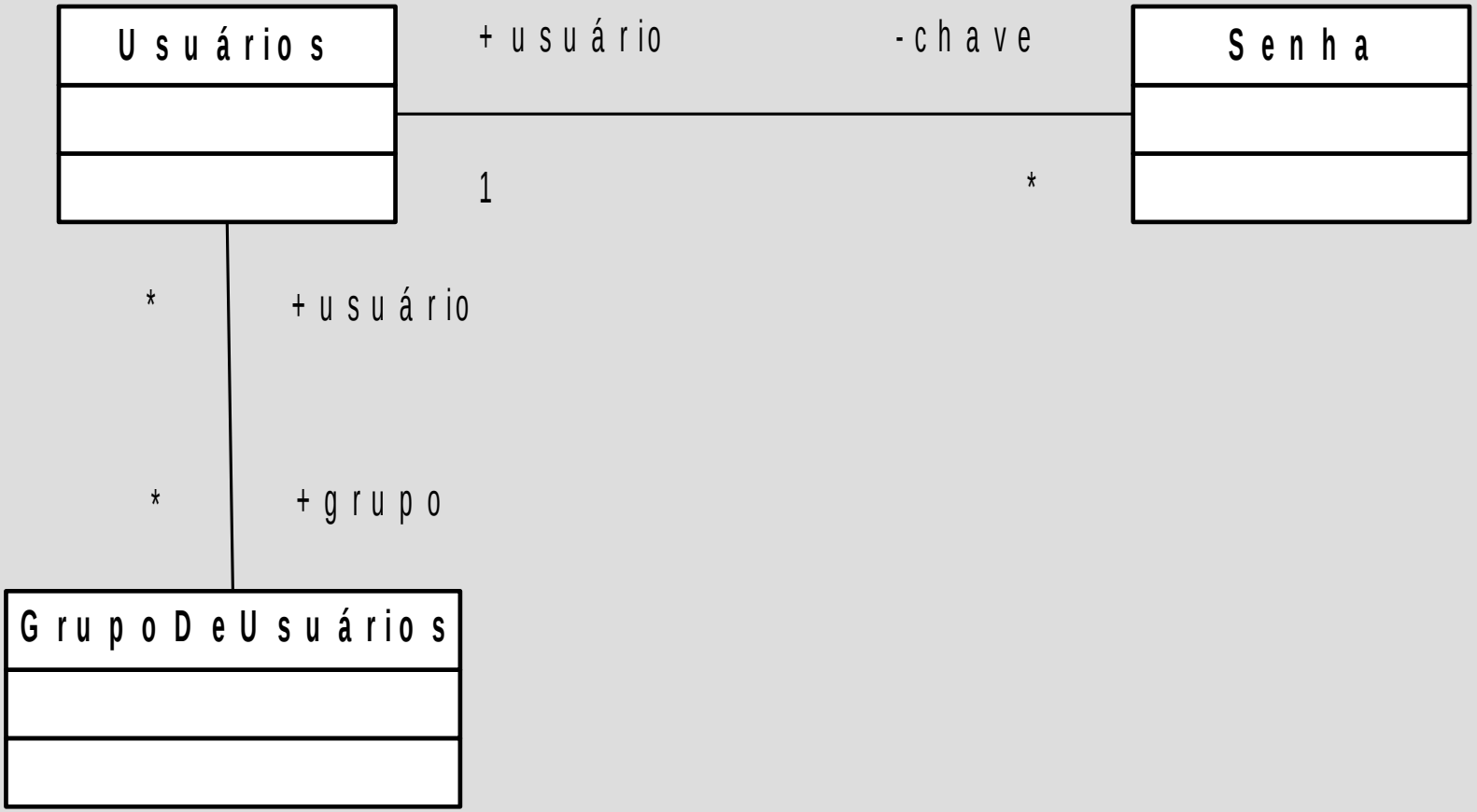
A visibilidade de um elemento do modelo define se ele poderá ser referenciado por um elemento fora de seu espaço de nome.

Estrutura das Classes

- Vale destacar que a **visibilidade** é usada para realizar o ***encapsulamento*** de uma abstração, ou seja, expor somente as características que são necessárias para a cumprimento das responsabilidades de um elemento de modelagem.

Estrutura das Classes

- Exemplo 1



Estrutura das Classes

- Exemplo 2

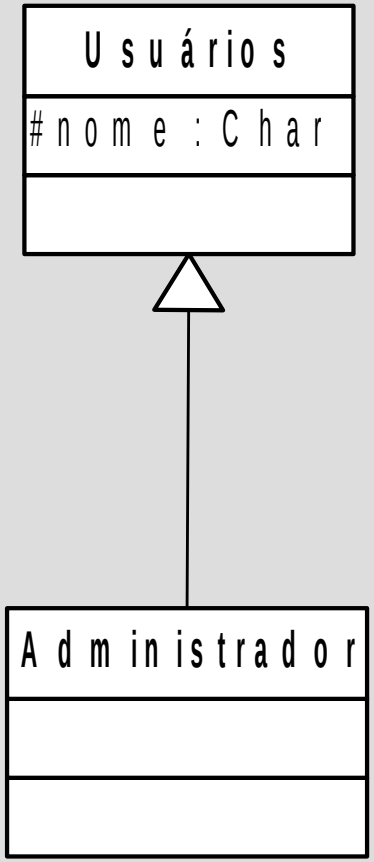
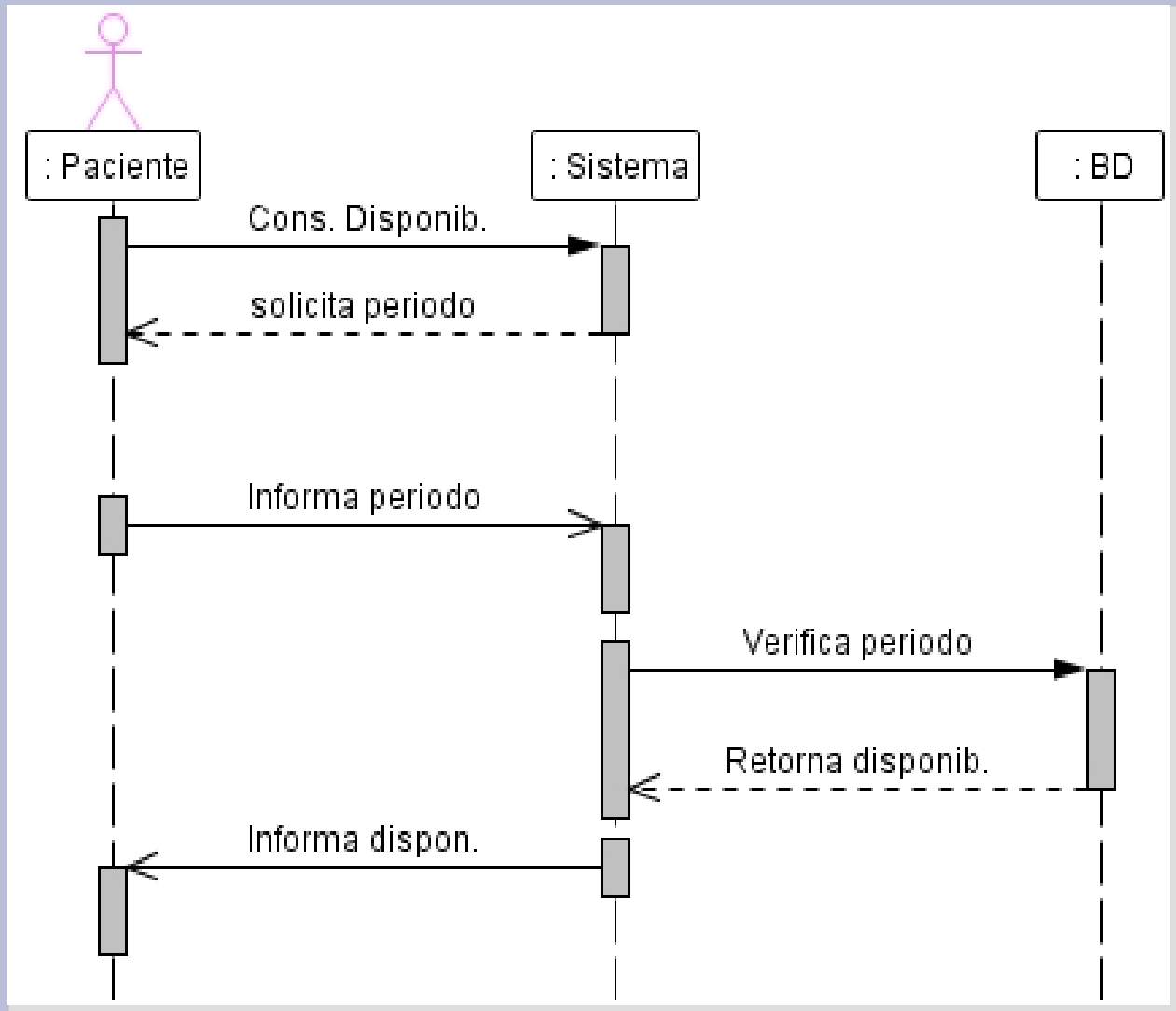


Diagrama de Sequência

- Determinam as colaborações e trocas de mensagens entre os objetos da modelagem. O diagrama também pode especificar os atores do contexto da aplicação.

Diagrama de Sequência
Exemplo

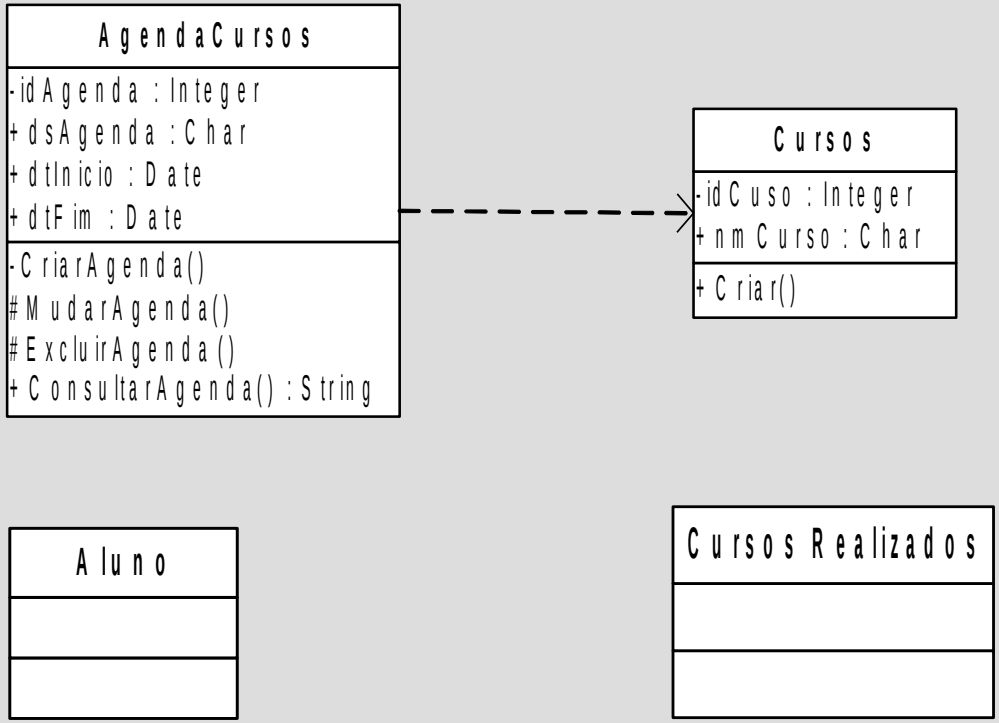


Lista de Exercícios

Resolução

1) No diagrama de classes abaixo onde temos as classes AgendaCursos, Cursos e Aluno, com quais classes a classe Aluno se relacionaria? Quais são os relacionamentos?

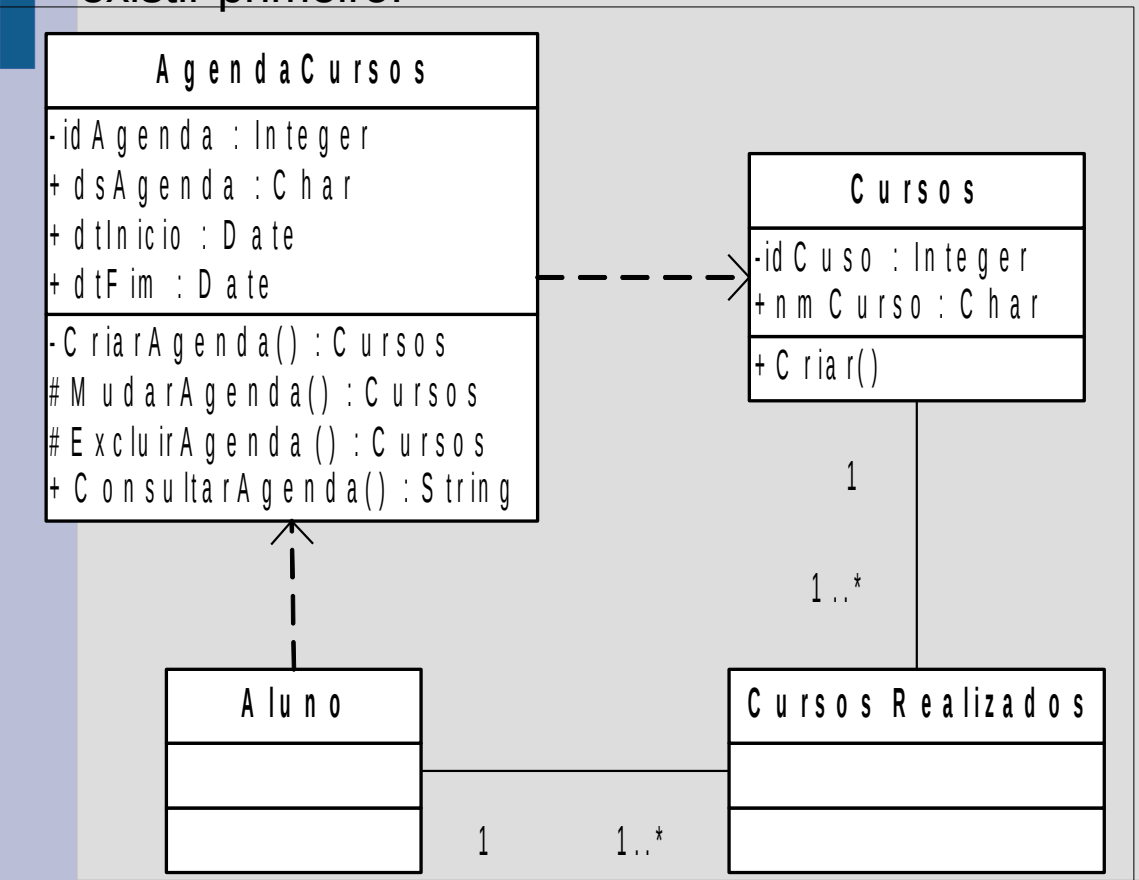
Faça o desenho para completar o diagrama de classes.



Lista de Exercícios

Resolução

Resposta: Verificamos que a classe *AgendaCursos* possui uma dependência com a classe *Cursos.* Isso quer dizer que existe uma dependência na assinatura das operações, ou seja, um objeto *AgendaCursos* jamais existirá se o objeto *Curso* não existir primeiro.



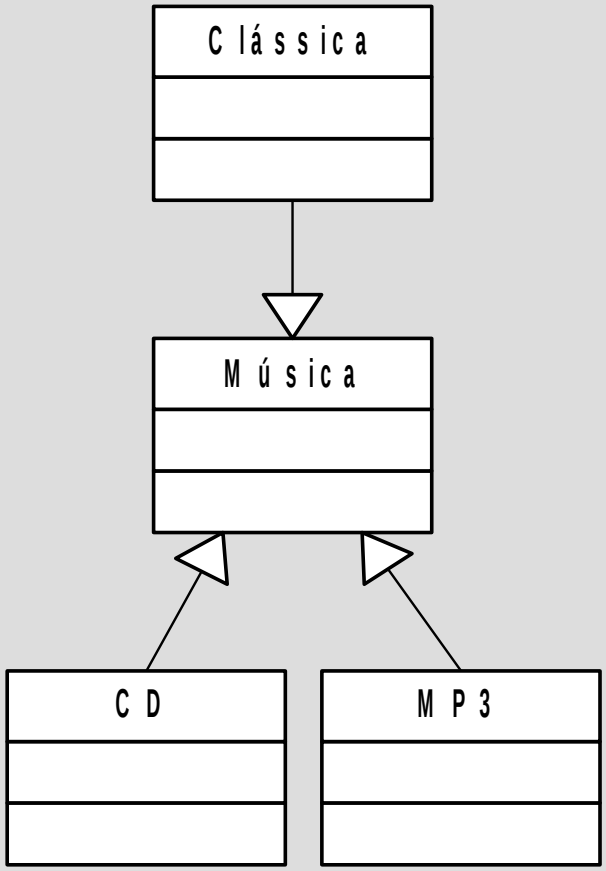
Verificamos que a classe *Aluno* possui uma dependência da classe *AgendaCursos*, pois sem ela não é possível realizar a inscrição de um *Aluno* em *Cursos*.
Verificamos que a classe *Aluno* se relaciona por Associação e Multiplicidade com a classe *Cursos* e esta relação é de (1..* e 1..*), logo surge uma classe *Cursos Realizados* que chamamos de Classe de Associação.

Lista de Exercícios

Resolução

2) O diagrama de classes abaixo está correto? Existe algo que está correto?

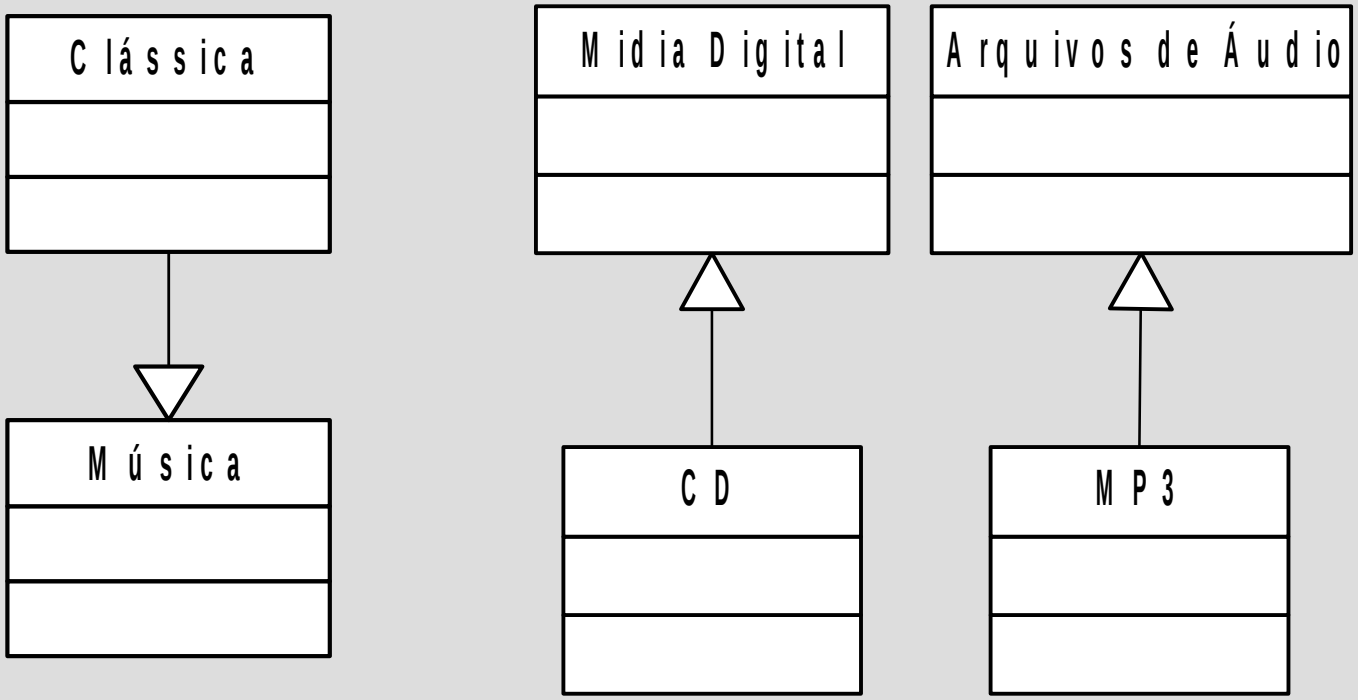
Se não concorda com a modelagem abaixo, faça um novo diagrama.



Lista de Exercícios

Resolução

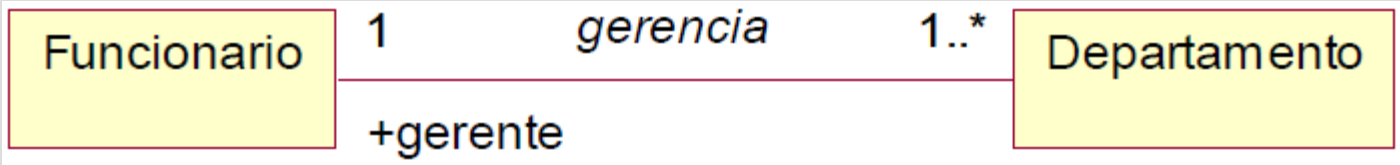
Resposta: O relacionamento de especialização (herança) entre as classes Música (superclasse/classe-mãe) e Clássica (subclasse/classe-filha) está correto, pois a música clássica é um tipo específico de música. Os relacionamentos das classes CD e MP3 estão errados, visto que CD e MP3 são tipos específicos de mídias e arquivos de áudio, respectivamente.



Lista de Exercícios

Resolução

3) Analise a modelagem realizada no diagrama de classes abaixo. O relacionamento por associação e multiplicidade está correto? Justifique sua resposta.

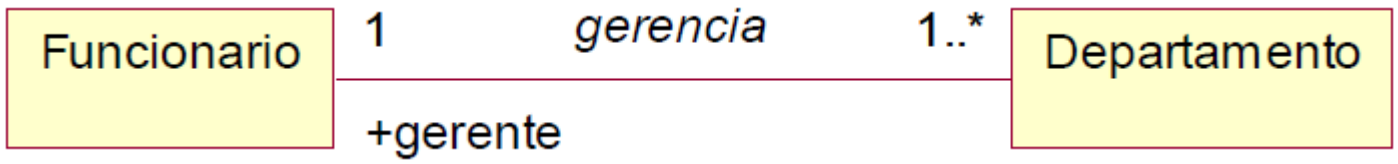


Lista de Exercícios

Resolução

Resposta: Sim, o relacinamento de associação por multiplicidade está correto. A classe **Funcionário** possui *um gerente* que *gerencia um ou muitos* **Departamentos**.

Vemos no cotidiano das empresas que é cada vez mais comum esse tipo de atribuição funcional.



Lista de Exercícios
Resolução

4) Desenhe o Diagrama de Caso de Uso para o enunciado abaixo:

"O *cliente* deseja que um

sistema de software solicite o

Login para Identificação do Usuário,

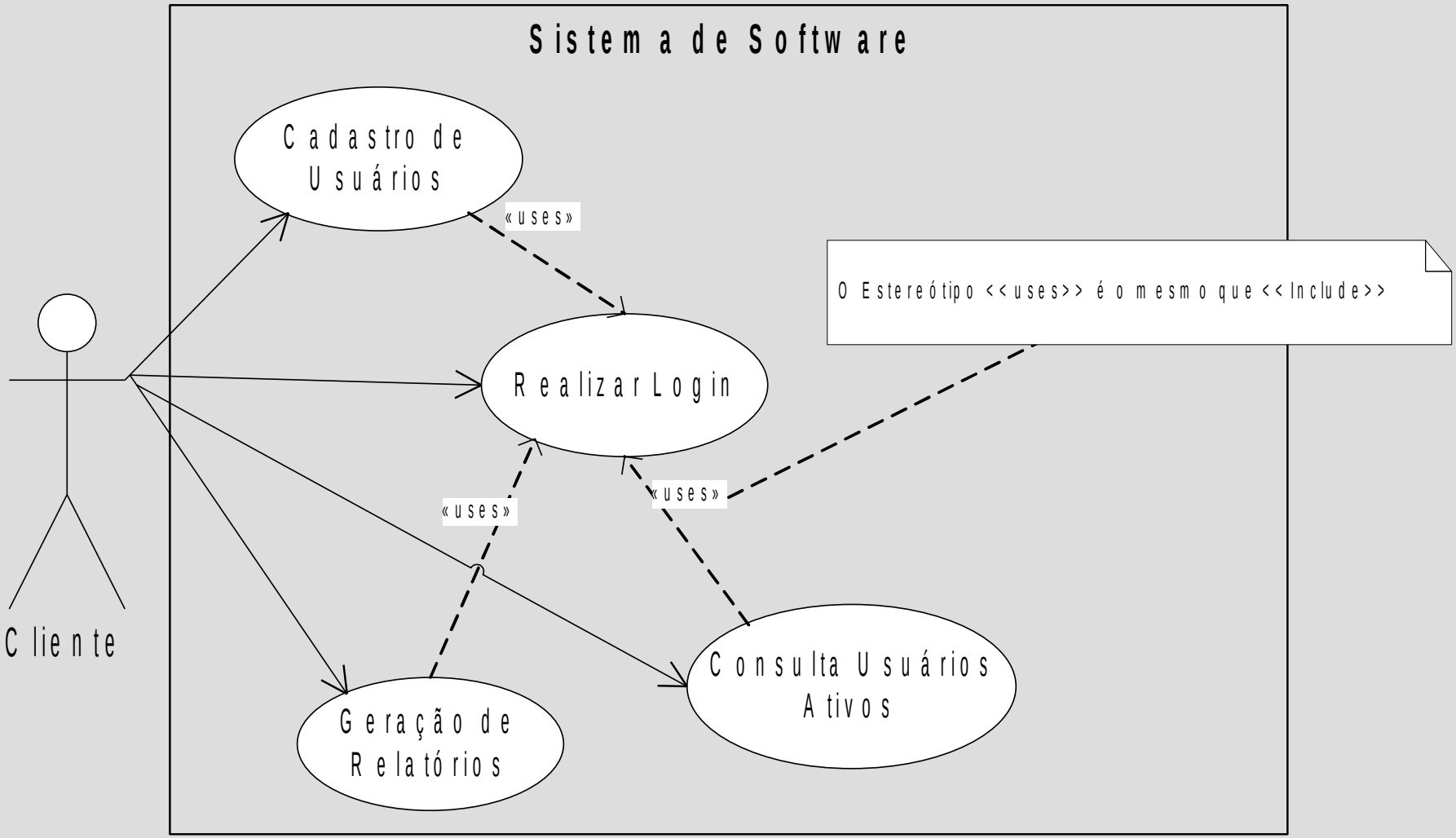
realize o *Cadastro de Usuários*,

Consulta de Usuários Ativos e a

Geração de Relatórios."

Lista de Exercícios

Resolução



Trabalho em Grupo (peso 2 – Máx. 5 part.)

1) É o diagrama que representa a troca de mensagens entre os objetos.

2) É o diagrama que especifica uma linha tracejada saindo de um objeto.

3) É o nome dado ao único relacionamento que possui visibilidade da característica definida como privada.

4) É o nome dado, no diagrama de sequência, a toda seta que devolve uma mensagem ao solicitante.

5) É o nome dado, no diagrama de sequência, a seta que envia uma mensagem ao destinatário.

6) São os estereótipos mais comuns nos relacionamentos de dependência.

7) É o relacionamento que especifica o todo-parte, mas influi no tempo de vida do objeto.

Trabalho em Grupo (peso 2 – Máx. 5 part.)

08) É o nome dado ao ato de reaproveitar casos de uso já existentes.

09) É o relacionamento da UML que nos remete à lembrança do relacionamento de cardinalidades.

10) É o nome dado ao uso da visibilidade para expor somente as características da classe que são necessárias, logo, ocultando das demais.

11) É o relacionamento que especifica o todo-parte, mas o faz apenas conceitualmente.

12) São os intervalos mínimo e máximo para a associação por multiplicidade.

13) É o elemento da modelagem capaz de separar outros elementos em sub-grupos de trabalho.

Muito Obrigado

Professor José Azanha Neto