Análise

- Estruturada
  - Modelo Essencial ou Lógico constitui-se de dois sub-modelos (Modelo Ambiental e Modelo Comportamental) e um Dicionário de Dados.
  - Linguagens: Fortran, Cobol, C, etc.

- Orientada a Objetos
  - Modelo Funcional, Modelo Estrutural e Modelo Comportamental.
  - Linguagens: Java, C++, etc.

Análise Estruturada

- **Modelo Ambiental:** Define a fronteira entre o sistema e o seu ambiente externo.
  - Diagrama de contexto

- **Modelo Comportamental:** Descreve o comportamento do interior do sistema.

- **Dicionário de Dados**: É uma listagem organizada de todos os elementos de dados do sistema.

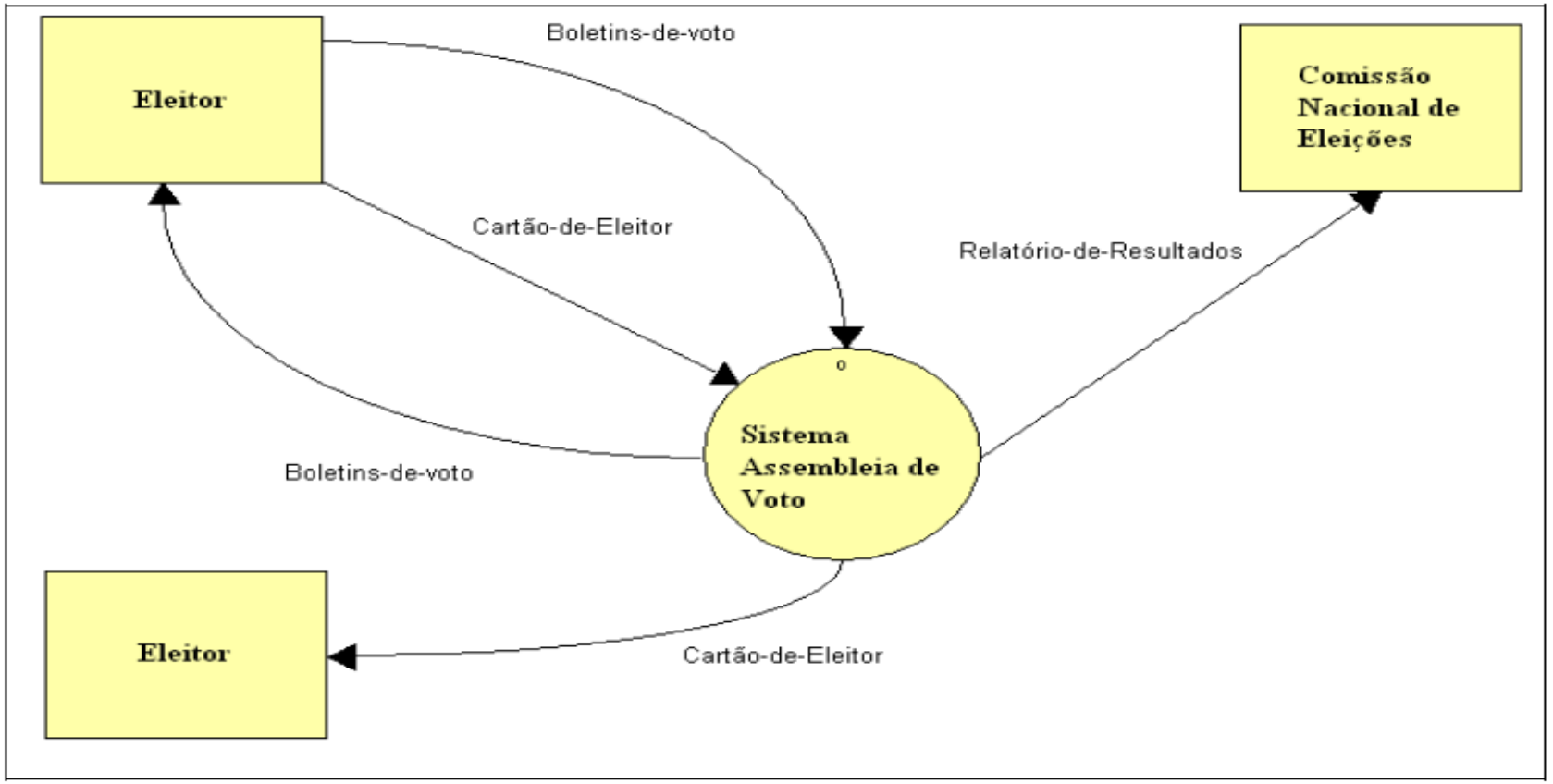
Modelo Ambiental::Diagrama de contexto

- **Processo**. É a parte mais fácil do diagrama de contexto, consiste de um único circulo. O nome do processo é normalmente o nome do sistema. São exemplos: Sistema de gestão da contabilidade e Sistema de gestão da biblioteca.

- **Terminadores/Entidades Externas.** São representados por um rectângulo. Os terminadores comunicam directamente com o sistema através de fluxos de dados ou de fluxos de controlo, ou através de depósitos de dados externos. Os terminadores não podem comunicar entre si.

O diagrama de contexto deve ser construído de modo que as entradas sejam causadas e iniciadas pelos terminadores e que as saídas/respostas sejam causadas e iniciadas pelo sistema.

Modelo Ambiental::Diagrama de contexto



Análise Estruturada::Modelo Comportamental

Depois de modelado e validado o modelo ambiental é necessário passar para a modelação do comportamento do interior do sistema com recurso a:

- Diagramas de Fluxos de Dados (DFD);
- Diagramas de Entidades Relacionamento (ER);
- Diagramas de Transição de Estados (DTE).

Modelo Comportamental::DFD

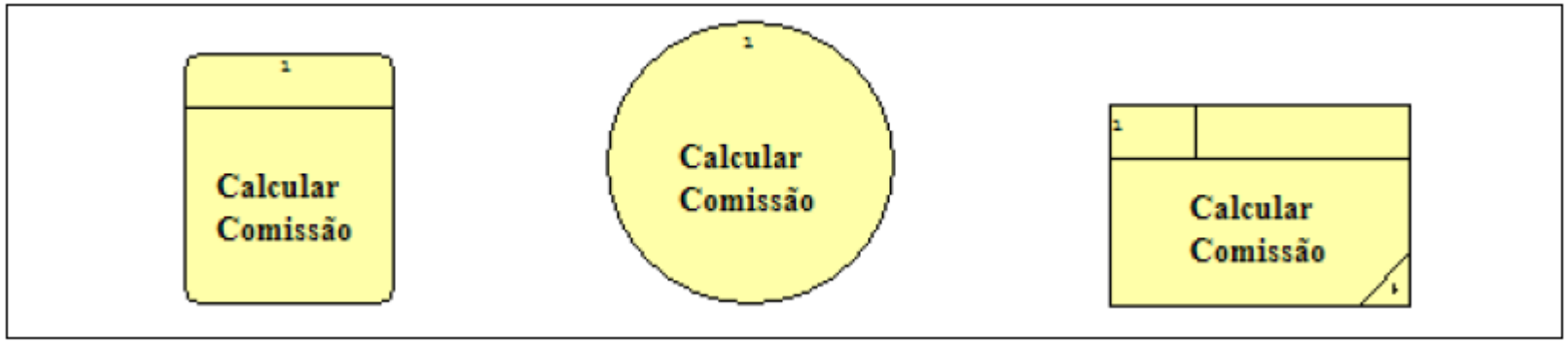
- Os DFDs são a principal técnica de modelação funcional da Análise Estruturada. Estes diagramas modelam o sistema como uma rede de processos ou funções, interligados por fluxos e depósitos de dados.

- Os Diagramas de Fluxos de Dados são composto de Processos, Fluxos de Dados, Depósitos de Dados e Terminadores/Entidades Externas.

- Podem ser usados para descrever quer processos computadorizados quer não computadorizados.

Modelo Comportamental::DFD

Processo

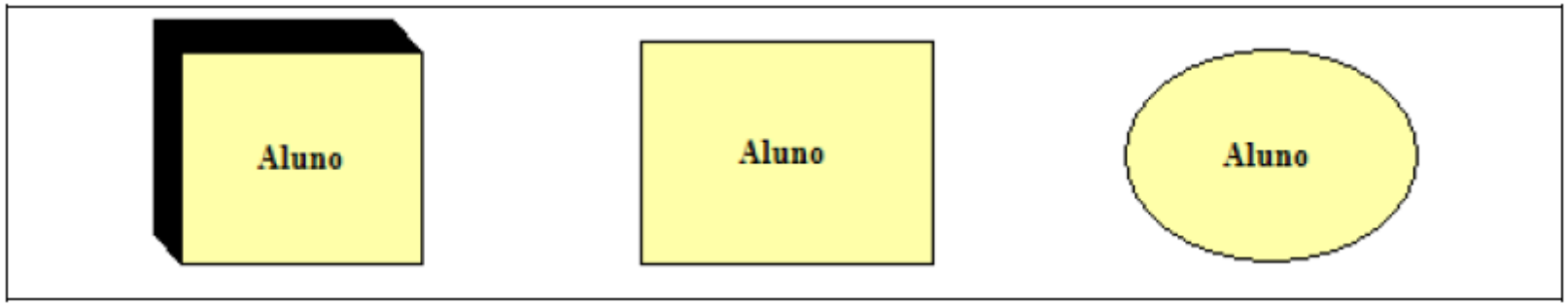


Fluxo de Dados

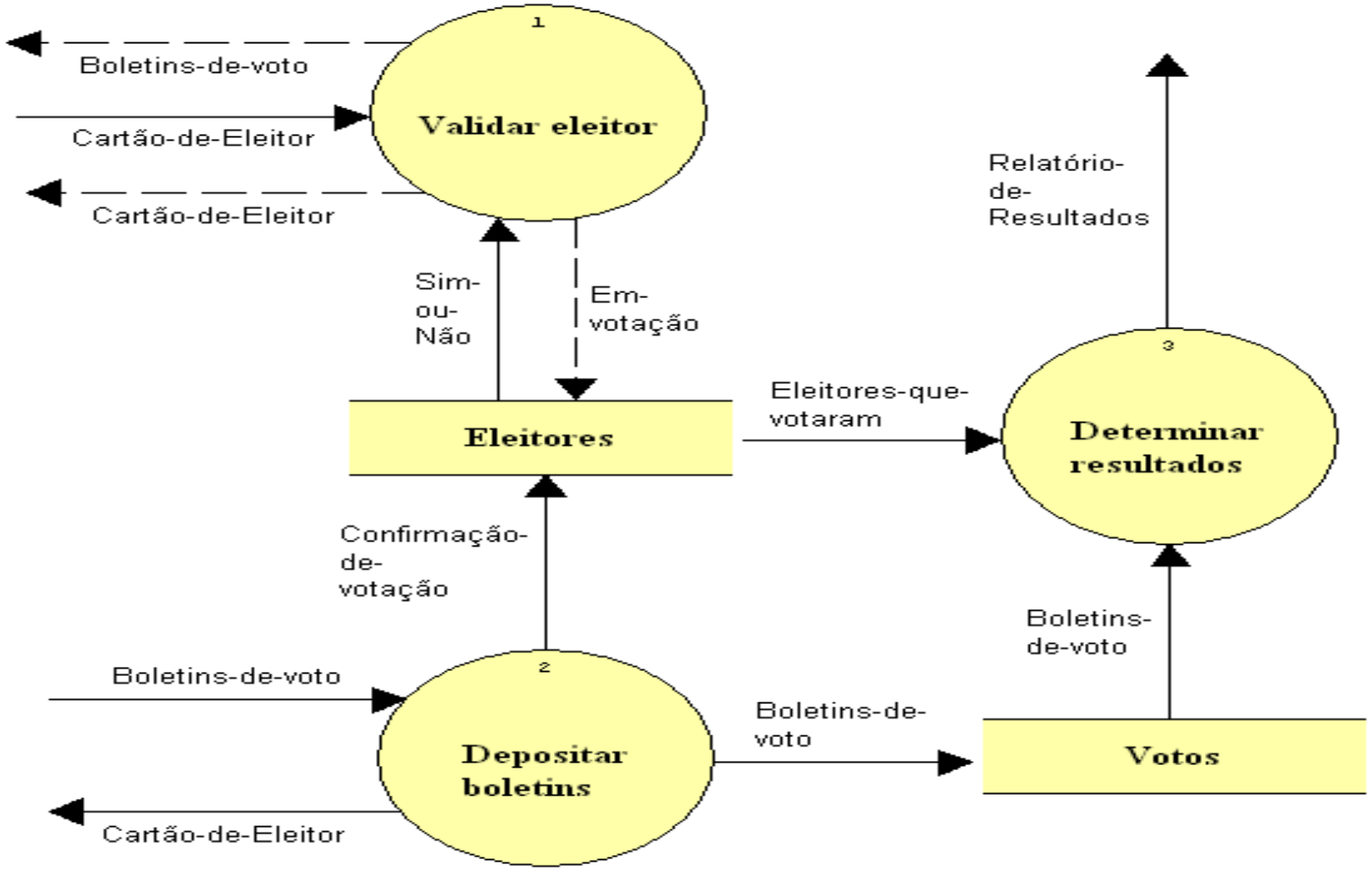
Deposito de Dados

Modelo Comportamental::DFD

Entidades Externas ou Terminadores



Modelo Comportamental::DFD



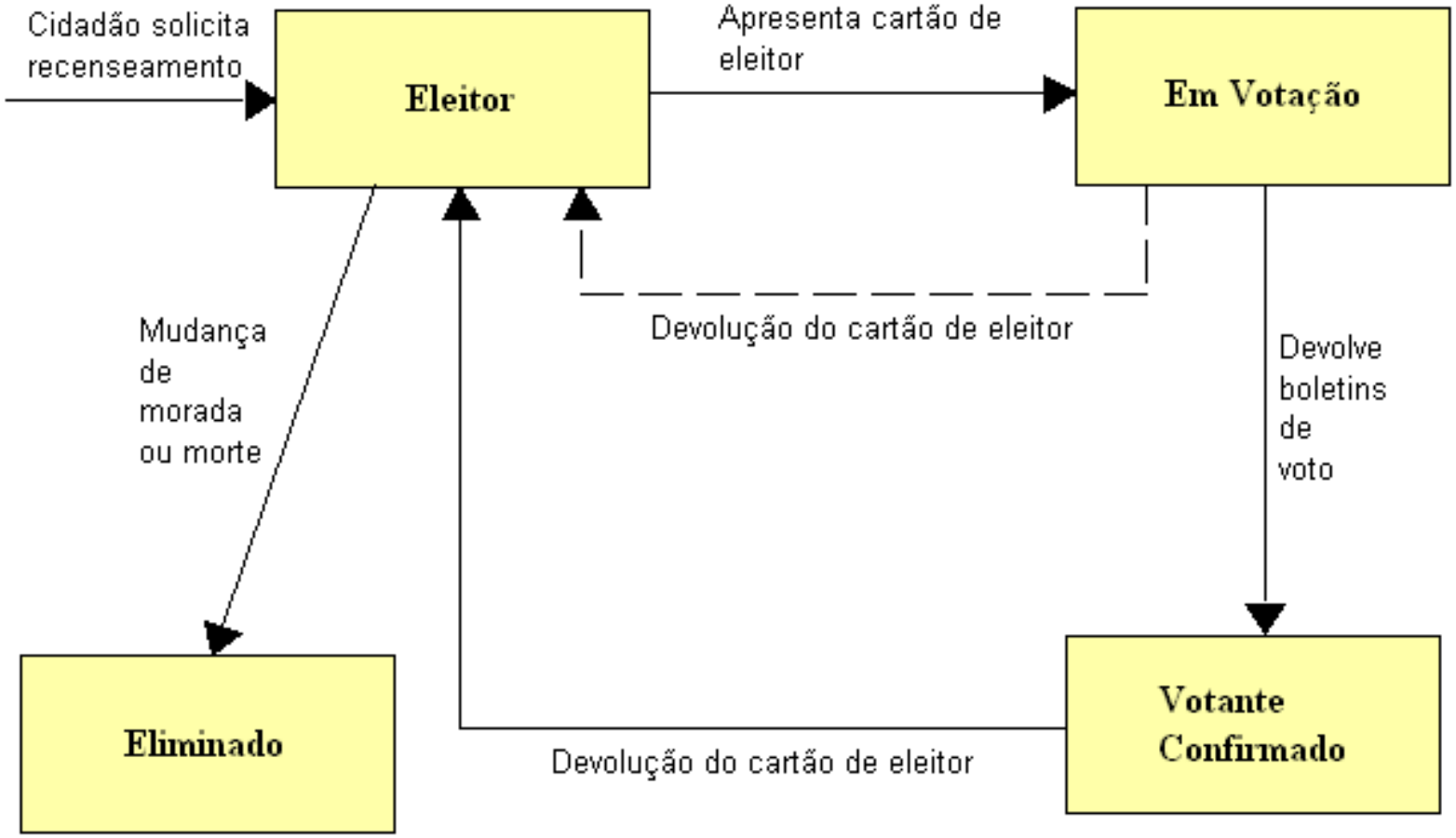
Modelo Comportamental::Diagramas ER

Desenvolvimento do Modelo de Dados Inicial

Modelo Comportamental::DTE

- Um DTE representa todas as seqüência de eventos que podem ocorrer durante o ciclo de vida de uma ocorrência de uma entidade no SI.

Modelo Comportamental::DTE



Análise Estruturada::Dicionário de Dados

- O dicionário de dados é uma listagem organizada de todos os elementos de dados do sistema, com definições precisas e rigorosas.

Análise Estruturada::Dicionário de Dados

- Elementos de dados;
- Estruturas de dados – grupos de elementos de dados.
- Fluxo de dados - um pacote de dados;
- Depósitos – uma coleção de pacotes de dados;

Análise OO

- As metodologias de análise OO procuram sistematizar quer a informação do sistema quer o processamento que manipula essa informação, através de objetos do mundo real.
  - Modelo Funcional,
  - Modelo Estrutural e
  - Modelo Comportamental.

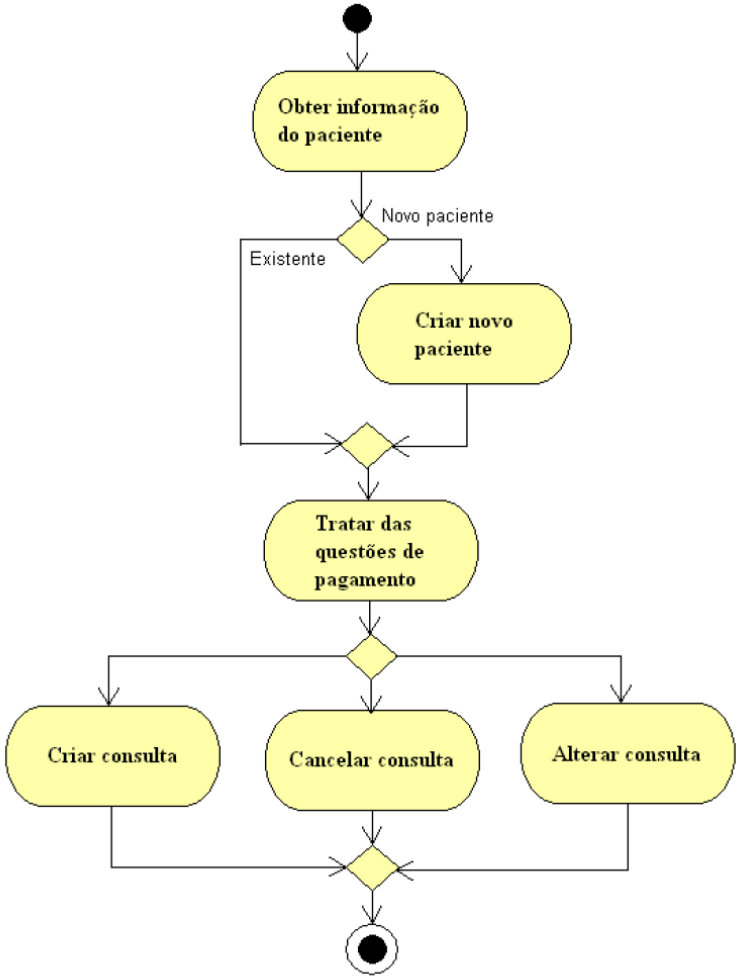
Análise OO::Modelo Funcional

- Descreve os processos de negócio e a interação do SI com o seu ambiente.
  - Diagramas de Atividades;
  - Diagramas de Casos de Uso.

Funcional::Diagrama de Atividades

- Determinar o contexto ou foco do processo a ser modelado de modo a encontrarmos um nome adequado para o diagrama.
- Identificar as atividades, fluxos de controle e fluxos de objeto que ocorram entre atividades.
- Identificar as decisões que fazem parte do processo a ser modelado.
- Identificar eventuais perspectivas de paralelismo no processo.
- Desenhar o diagrama.

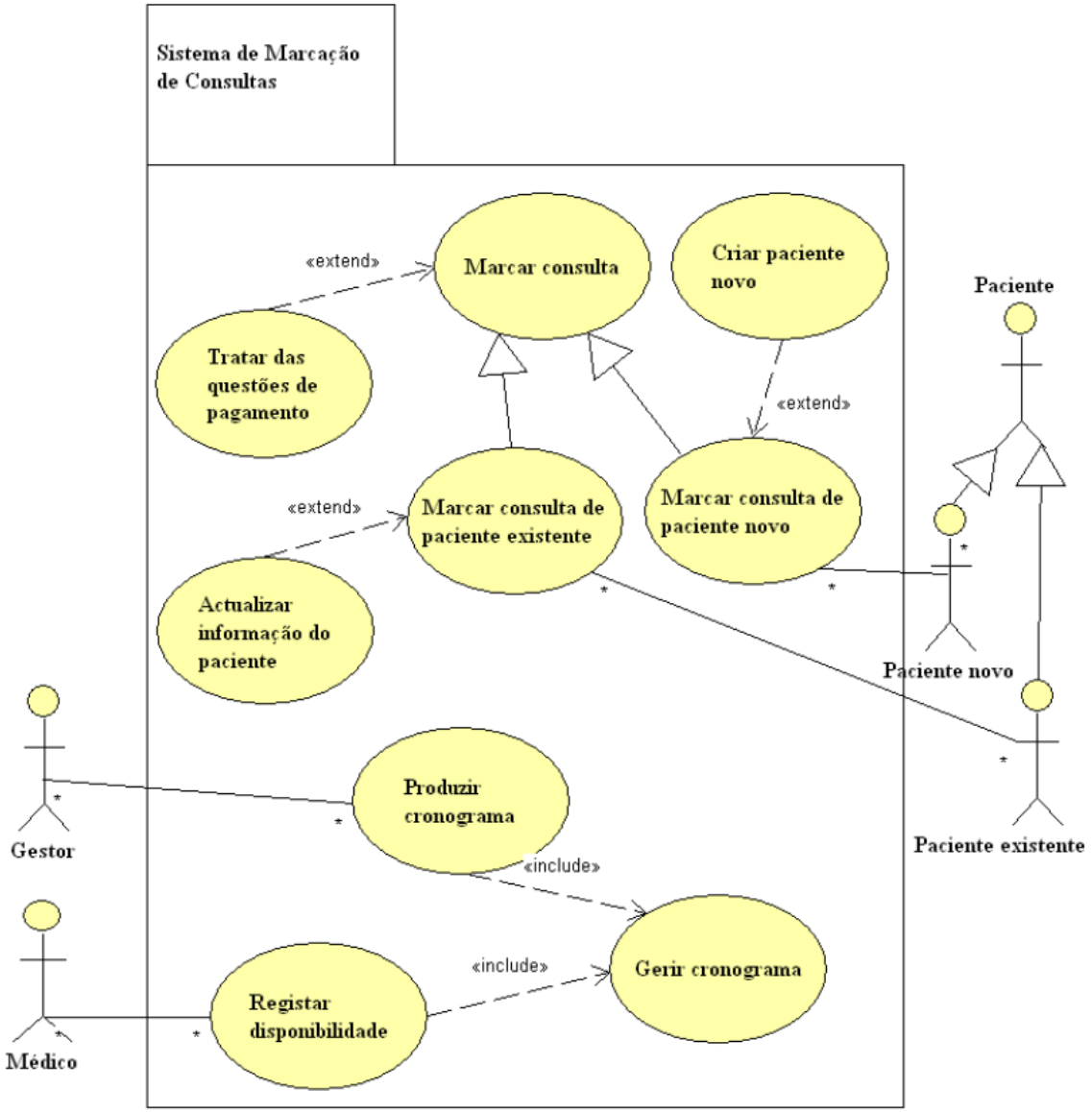
Funcional::Diagrama de Atividades



Funcional::Diagrama de casos de uso

- Identificar os casos de uso principais
- Expandir os casos de uso principais
- Confirmar os casos de uso principais
- Criar o Diagrama.

Modelo Funcional::Diagrama de casos de uso



Análise OO:: Modelo Estrutural

Descreve a estrutura dos dados que suportam os processos de negócio nas organizações.

- Descrição Classes-Responsabilidades-Colaborações (CRC) - deve conter toda a informação necessária para construir um modelo estrutural lógico do domínio do problema em análise;
- Diagramas de Classes;
- Diagramas de Objetos.

Modelo Estrutural:: Descrições CRC

- organização lógica dos dados, sem indicar como os dados são criados, armazenados ou manipulados;
- descrição CRC apresenta os elementos essenciais de uma classe;

Nome da classe: Paciente	**ID:** 3	**Tipo:** Concreta, domínio
Descrição: Um indivíduo que necessita receber cuidados de saúde		**Caso de uso associado:** 2

Responsabilidades	**Colaboradores**
Marcar consulta	Consulta
Determinar última consulta	
Mudar estado	
Fornecer histórico clínico	História clínica

Atributos:

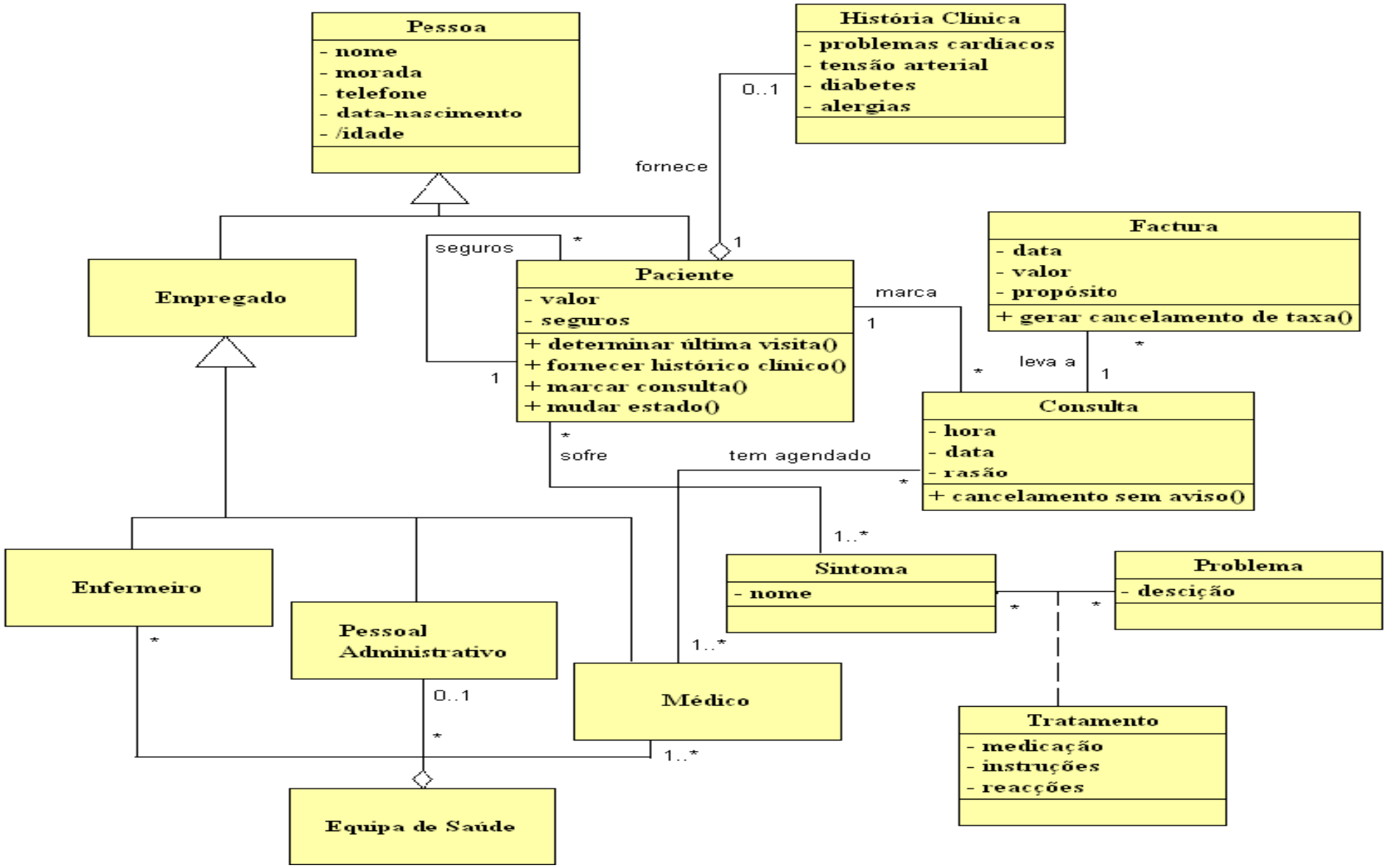
Valor (duplo)
Seguro (texto)

Relacionamentos:

Generalização (um tipo de): Pessoa

Agregação (é parte de): História clínica

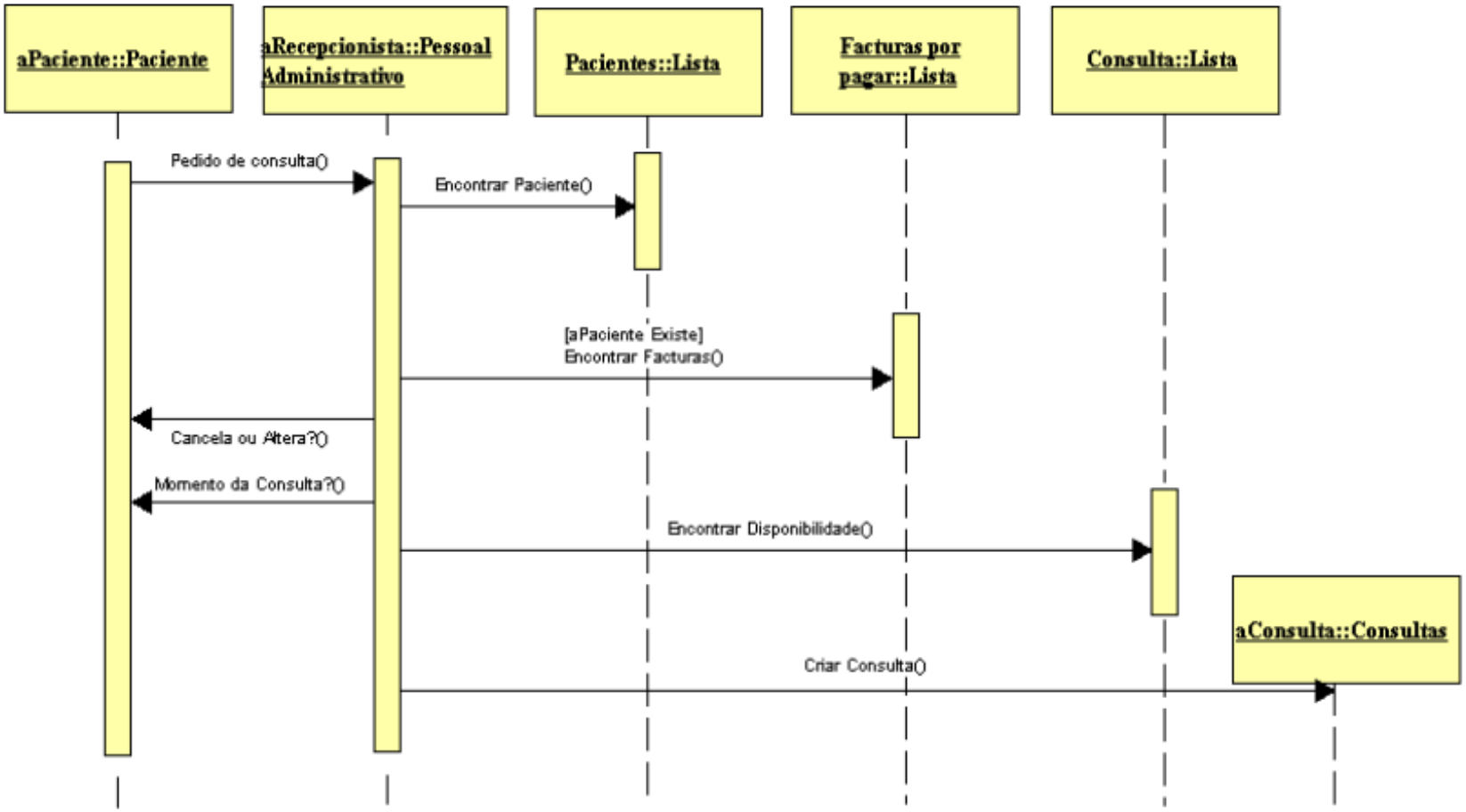
Outras Associações: Consulta


Pessoa
- nome
- morada
- telefone
- data-nascimento
- /idade
História Clínica
- problemas cardíacos
- tensão arterial
- diabetes
- alergias
0..1
fornece
1
Empregado
seguros
*
1
Paciente
- valor
- seguros
+ determinar última visita()
+ fornecer histórico clínico()
+ marcar consulta()
+ mudar estado()
marca
1
*
Factura
- data
- valor
- propósito
+ gerar cancelamento de taxa()
*
leva a
1
Consulta
- hora
- data
- razão
+ cancelamento sem aviso()
*
sofre
*
tem agendado
*
1..*
Enfermeiro
*
Pessoal Administrativo
0..1
*
Médico
1..*
1..*
Sintoma
- nome
*
*
Problema
- descição
Tratamento
- medicação
- instruções
- reacções
Equipa de Saúde

Análise OO::Modelo Comportamental

- Descreve os aspectos da dinâmica interna de um sistema de informação
  - Diagramas de seqüência;
  - Diagramas de comunicação;
  - Diagramas de transição de estados.

Comportamental::Diagrama de seqüência



Comportamental::Diagrama de comunicação

- Os diagramas de comunicação são equivalentes aos diagramas de seqüência, mas enfatizam o fluxo de mensagens ao longo de um conjunto de objetos, enquanto que os diagramas de seqüência focam-se na ordem temporal das mensagens que são passadas.

Comportamental::Diagrama de transição de estados

Projeto::Lógico x Físico

- Projeto lógico (independente de implementação) é executado para produzir um projeto que poderia ser implementado em diferentes plataformas (hardware, linguagem de programação, SGBD)
- Projeto físico (implementação específica) é executado para produzir um projeto que é específico para a plataforma escolhida.
- Algumas vezes se a plataforma é conhecida quando começa o projeto, não haverá o estágio de projeto lógico

Projeto::Compromissos

- É impossível alcançar todos os objetivos - Compromissos devem ser feitos
- Performance X Custo;
  - sistemas mais rápidos, custos mais altos;
- Portabilidade X Padrões X Investimento já realizado;
  - Sistemas Abertos;
  - Sistemas existentes/habilidades de uma linguagem particular;
  - Custos de treinamento e interface com outros sistemas;
- Usabilidade X Custo e Performance;
  - Maior tempo de desenvolvimento;
  - Maior custo de hardware;
  - Melhor aceitação do usuário;

Como fazer um bom projeto?

- Funcional;
- Eficiente ;
- Flexível ;
- Portátil;
- Seguro;
- Confiável;
- Econômico;
- Genérico;
- Possível de ser construído;
- Gerenciável;
- Fácil de Manter;
- Reutilizável;
- Útil.

O que é um padrão?

- Maneira testada ou documentada de alcançar um objetivo qualquer
  - Padrões são comuns em várias áreas da engenharia

- *Design Patterns*, ou Padrões de Projeto
  - Padrões para alcançar objetivos na engenharia de software usando classes e métodos em linguagens orientadas a objeto.

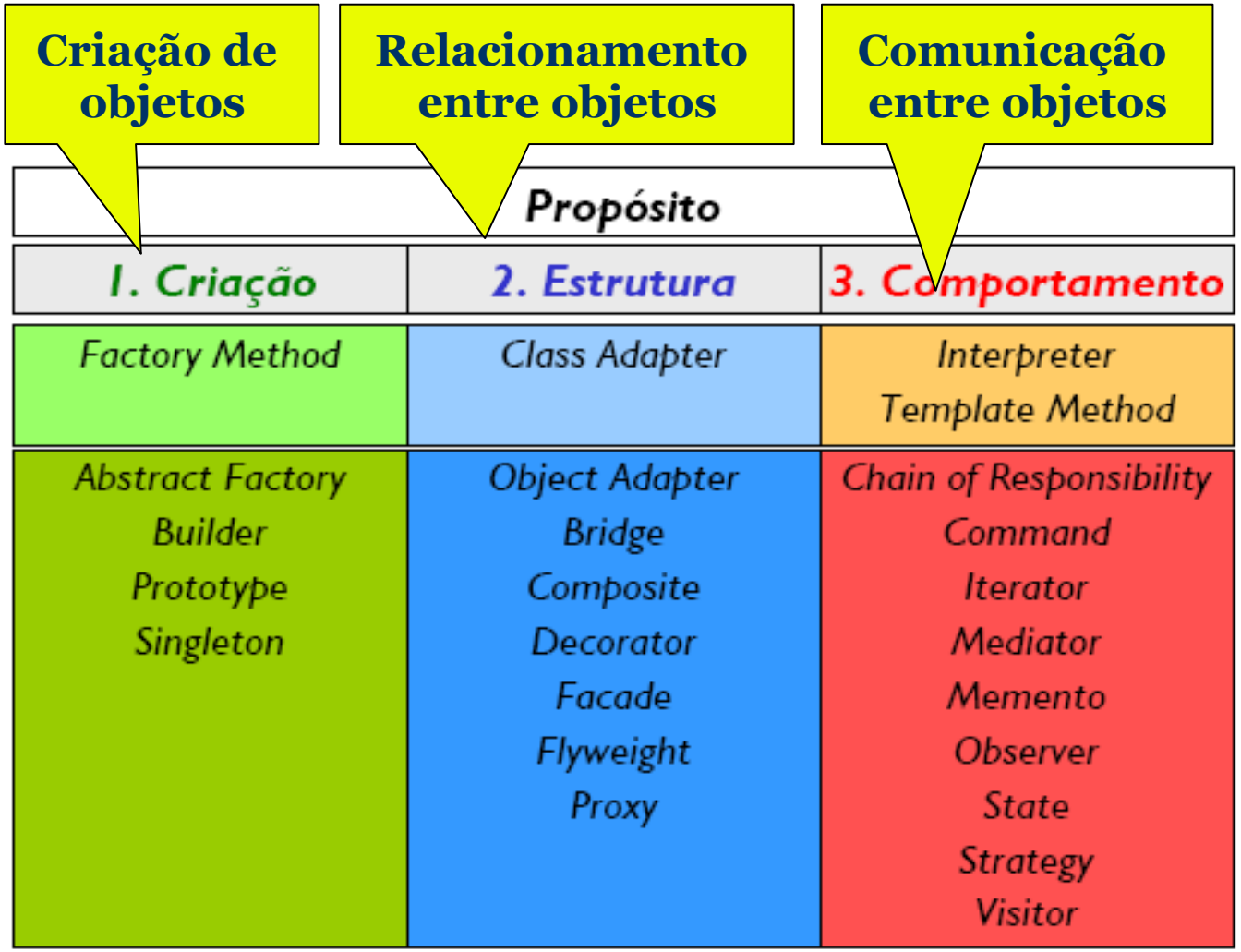
O que é um padrão de projeto?

- Padrões são um repertório de soluções e princípios que ajudam os desenvolvedores a criar software e que são codificados em um formato estruturado consistindo de:
  - **Nome**
  - **Problema que soluciona**
  - **Solução do problema**

Padrões::Classificação

- Padrões de criação
  - Tratam do processo de criação de objetos
- Padrões estruturais
  - Tratam da composição de classes ou objetos.
- Padrões comportamentais
  - Caracterizam interações e distribuição de responsabilidade entre classes e objetos.

Padrões::Classificação



Padrões::Na Prática::Façade

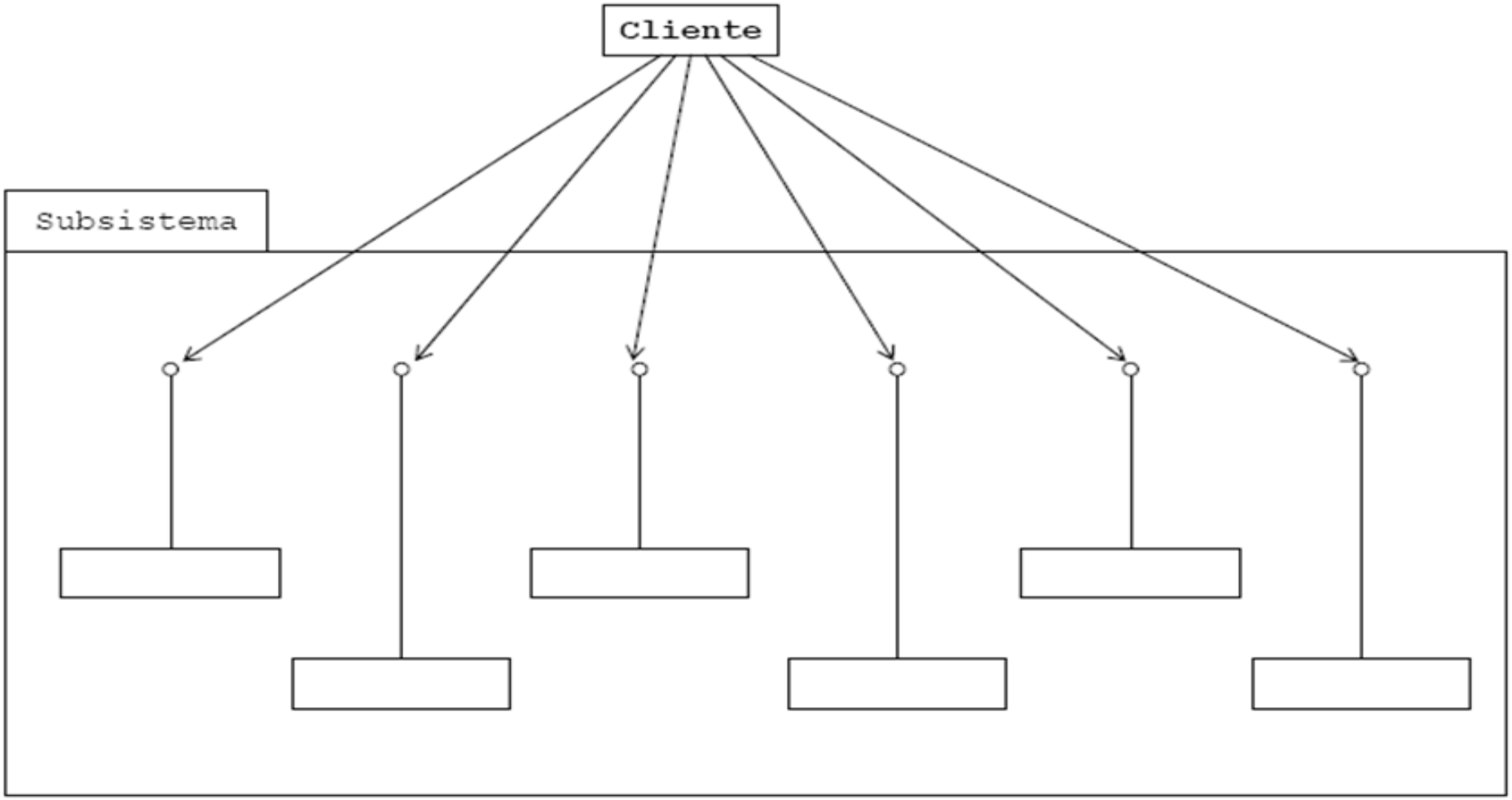
- Problema: Comunicação e a dependência entre os subsistemas forçando o cliente a conhecer muitos detalhes para poder utilizá-lo.

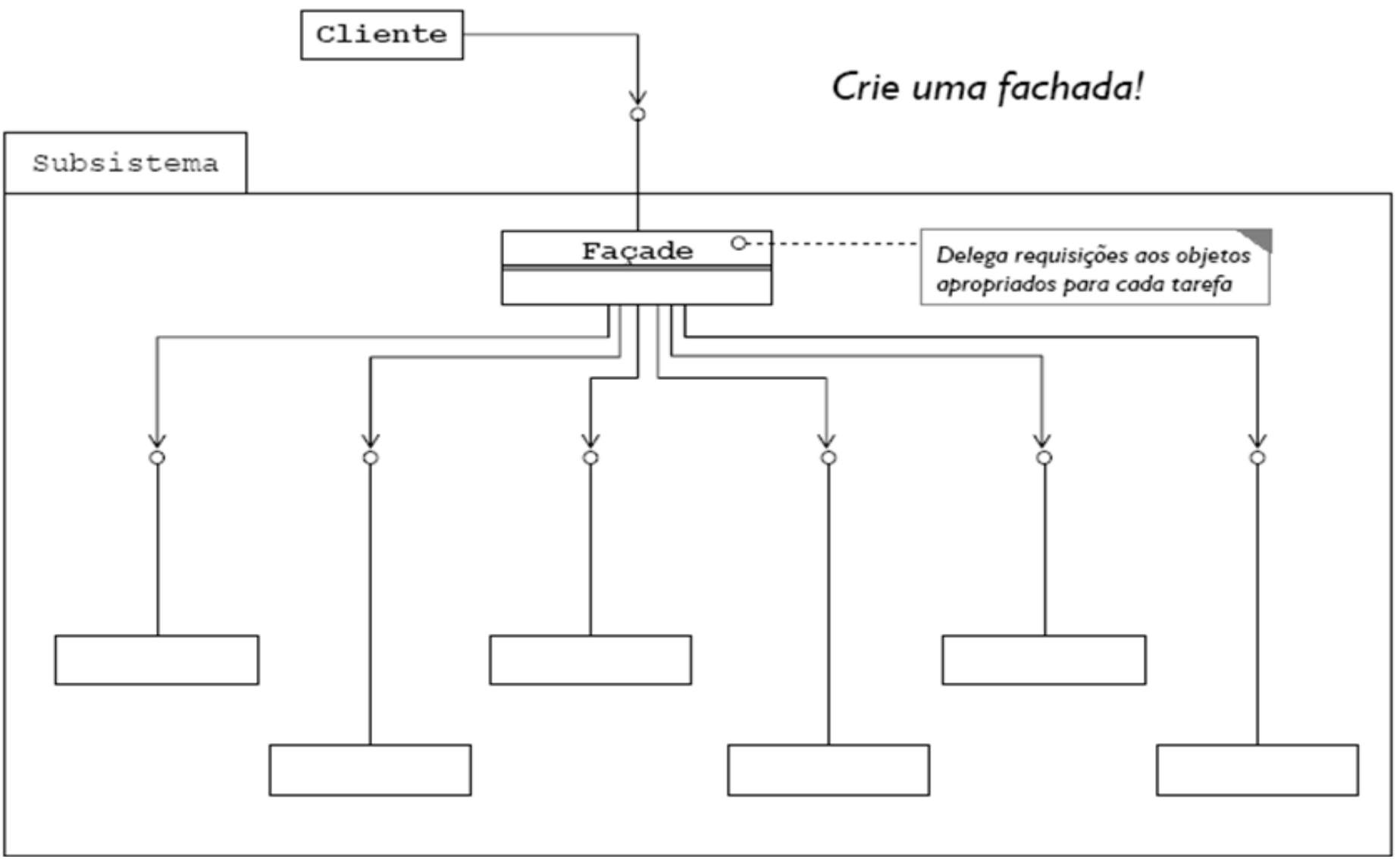
- Aplicações: Quando necessário definir um ponto de entrada para cada nível de subsistema.

- Objetivo: Fornecer uma interface unificada para um conjunto de interfaces em um subsistema. Diminuir o acoplamento entre as classes.

Padrões::Na Prática::Façade

Problema: Cliente precisa conhecer muitos detalhes




Cliente
Crie uma fachada!
Subsistema
Façade
Delega requisições aos objetos apropriados para cada tarefa

```
class Aplicação {
    ...
    Facade f;
    // Obtem instancia f
    f.registrar("Zé", 123);

    f.comprar(223, 123);
    f.comprar(342, 123);

    f.fecharCompra(123);

    ...
}
```

```
public class Facade {
   BancoDeDados banco = Sistema.obterBanco();
   public void registrar(String nome, int id) {
      Cliente c = Cliente.create(nome, id);
      Carrinho c = Carrinho.create();
      c.adicionarCarrinho();
   }
   public void comprar(int prodID, int clienteID) {
      Cliente c = banco.selectCliente(cliente ID);
      Produto p = banco.selectProduto(prodID) {
      c.getCarrinho().adicionar(p);
    }
    public void fecharCompra(int clienteID) {
       Cliente c = banco.selectCliente(clienteID);
       double valor = c.getCarrinho.getTotal();
       banco.processarPagamento(c, valor);
    }
}
```

```
public class Carrinho {
  static Carrinho create() {...}
  void adicionar(Produto p) {...}
  double getTotal() {...}
}
```

```
public class Produto {
  static Produto create(String nome,
                        int id,double preco) {...}
  double getPreco() {...}
}
```

```
public class Cliente {
 static Cliente create(String nome,
                       int id) {...}
 void adicionarCarrinho(Carrinho c) {...}
 Carrinho getCarrinho() {...}
}
```

```
public class BancoDeDados {
 Cliente selectCliente(int id) {...}
 Produto selectProduto(int id) {...}
 void processarPagamento() {...}
}
```

Padrões::Na Prática::Singleton

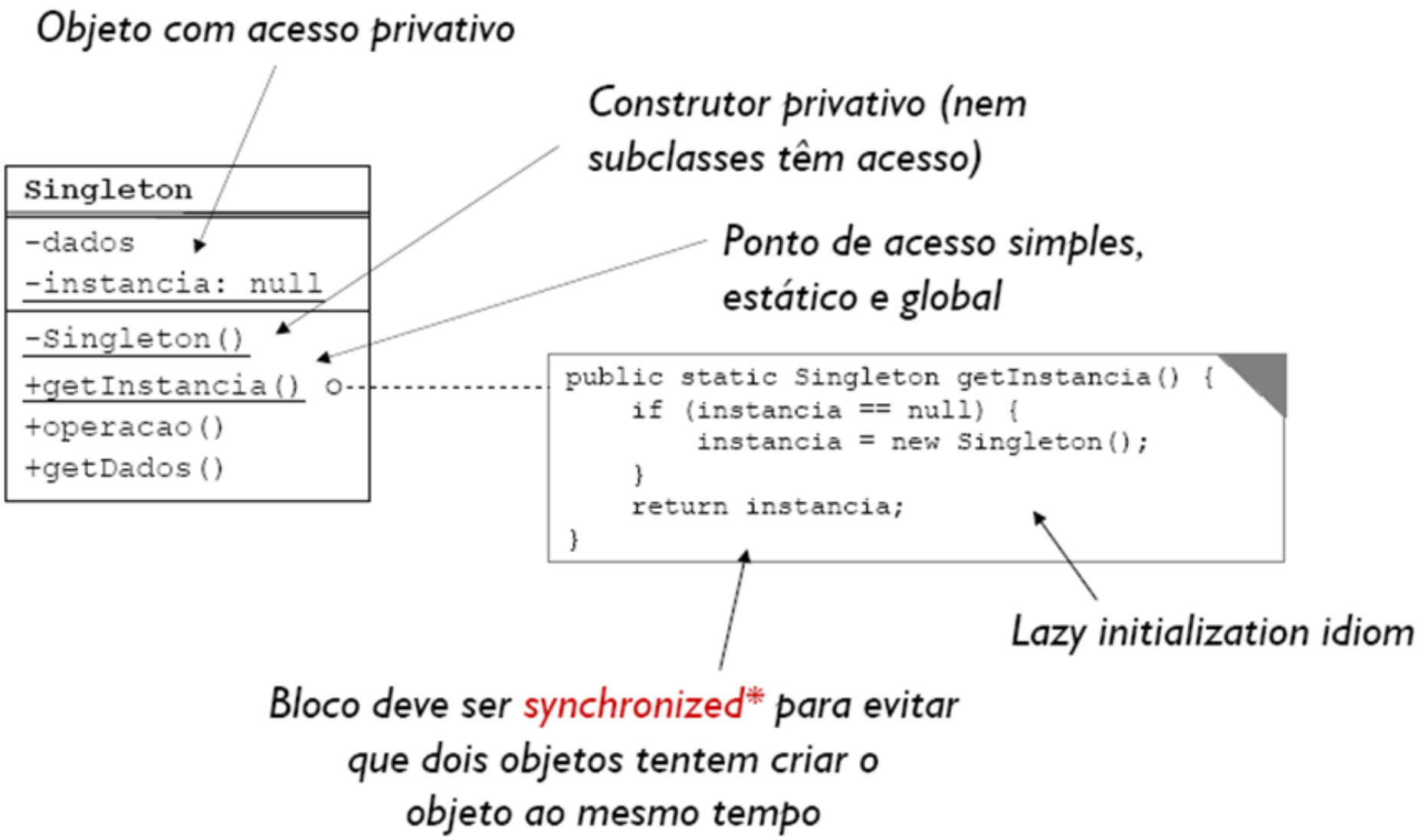
- Problema: Garantir que apenas um objeto exista, independente do número de requisições que se receba para criá-lo.

- Aplicações:
  - Uma única conexão ao banco de dados;
  - Um único acesso ao arquivo de logs.

- Objetivo: Garantir que uma classe tenha apenas uma única instância

Padrões::Na Prática::Singleton

- O padrão de criação de objetos *Singleton* garante que para uma dada classe possa haver somente uma instância. A classe *Singleton* deve:
  - Armazenar a única instância existente;
  - Garantir que apenas uma instância será criada;
  - Prover acesso a tal instância.


Objeto com acesso privativo
Construtor privativo (nem subclasses têm acesso)
Ponto de acesso simples, estático e global
Singleton
-dados
-instancia: null
-Singleton()
+getInstancia()
+operacao()
+getDados()
public static Singleton getInstancia() {
if (instancia == null) {
instancia = new Singleton();
}
return instancia;
}
Lazy initialization idiom
Bloco deve ser synchronized* para evitar que dois objetos tentem criar o objeto ao mesmo tempo

Padrões::Na Prática::Singleton

```
public class Highlander {
  private Highlander() {}
  private static Highlander instancia = new Highlander();
  public static synchronized Highlander obterInstancia() {
        return instancia;
  }
}
```

Esta classe implementa o design pattern Singleton

Esta classe cria apenas um objeto Highlander

```
public class Fabrica {
  public static void main(String[] args) {
    Highlander h1, h2, h3;
    //h1 = new Highlander(); // nao compila!
    h2 = Highlander.obterInstancia();
    h3 = Highlander.obterInstancia();
    if (h2 == h3) {
       System.out.println("h2 e h3 são mesmo objeto!");
    }
  }
}
```

Obrigado!

Apresentação dos Padrões

- Pesquisar sobre os padrões de projetos e apresenta-los em seminário/*workshop* para a turma.

Análise e Projeto de Sistemas de Informação

Andrêza Leite
andreza.lba@gmail.com