

# Apêndice

# UML e Metodologias de Desenvolvimento de Sistemas de Informação Orientadas a Objetos

•  
•  
•  
•  
•  
•  
•

•  
•  
•

# Metodologia de Desenvolvimento

- ❖ Visão geral - metodologias e UML
- ❖ Técnicas de Análise
- ❖ Projeto: Objetos, Interface e Sistema
- ❖ Método Integrado
- ❖ Considerações Gerais

•  
•  
•

# Metodologia

Coleção de **técnicas e diretrizes** para construção, manutenção e melhoria em produtos de software.

Fornece uma base de comunicação, um conjunto de técnicas e uma base para engenharia de software.

• • • • • • • • • •

- # Fases Clásicas

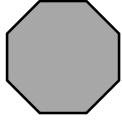
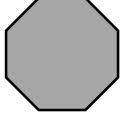
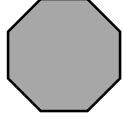
# Fases Clásicas

- # Modelo Espiral

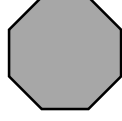
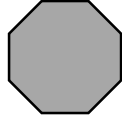
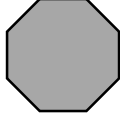
-

•  
•  
•

# Metodologia OO



**O que faz com que uma metodologia de desenvolvimento de sistemas seja considerada Orientada a Objetos?**

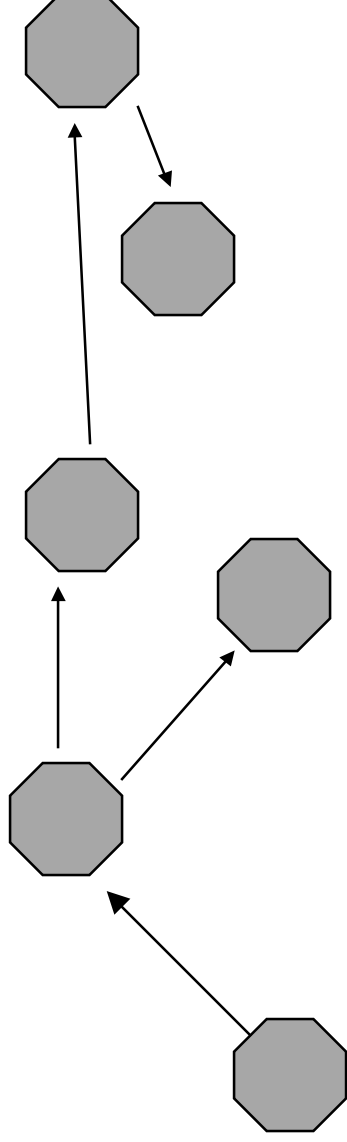


• • • • • • • • • •

•  
•  
•

# Metodologia OO

Uma metodologia de desenvolvimento de sistemas é considerada Orientada a Objetos se ela orienta a construção de sistemas a partir do entendimento do mundo real como um conjunto de objetos que comunicam-se entre si de forma coordenada



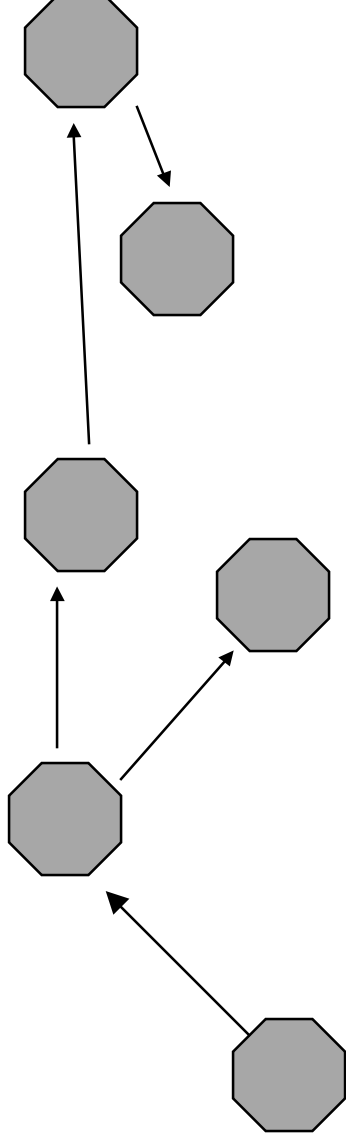
• • • • • • • • • •

•  
•  
•

# Metodologia OO

## *Quais são as principais atividades ?*

- ❖ Entender quais são os **objetos** envolvidos no domínio do problema
- ❖ Entender como se **comunicam** no mundo real
- ❖ **Projetar** a forma como devem ser **implementados**



• • • • •

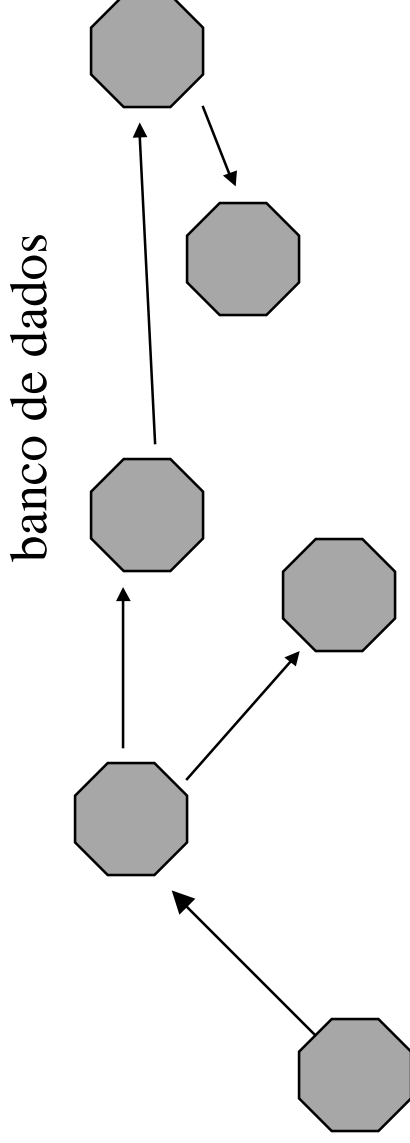


• • •

# Metodologia OO

# Quais as principais técnicas utilizadas?

- ❖ Entendimento do mundo real - Revisão de processos, Use cases
- ❖ Objetos e seus relacionamentos - Modelo de Objetos, CRC, DTE, DI
- ❖ Projeto - Padrões de projeto, frameworks, componentes
- ❖ Implementação - ambientes de desenvolvimento, middleware (RMI),

[illegible]

•  
•  
•

# Métodos de Desenvolvimento OO

**Booch** - Object-Oriented Design with Applications

**Wirfs-Brock** - Designing Object-Oriented Software (**CRC**)

**Rumbaugh** - Object-Oriented Modeling and Design (**OMT**)

**Coad-Yourdon** - Object-Oriented Analysis

**Jacobson** - OO Software Engineering - A **Use Case** Driven Approach

**Shlaer-Mellor** - Object **Lifecycles**-Modeling the World in States

**Coleman et al:** Fusion - OO Development: The **Fusion** Method

• • • • • • • • • •

•  
•  
•

# Métodos de Desenvolvimento OO

## *Estratégia*

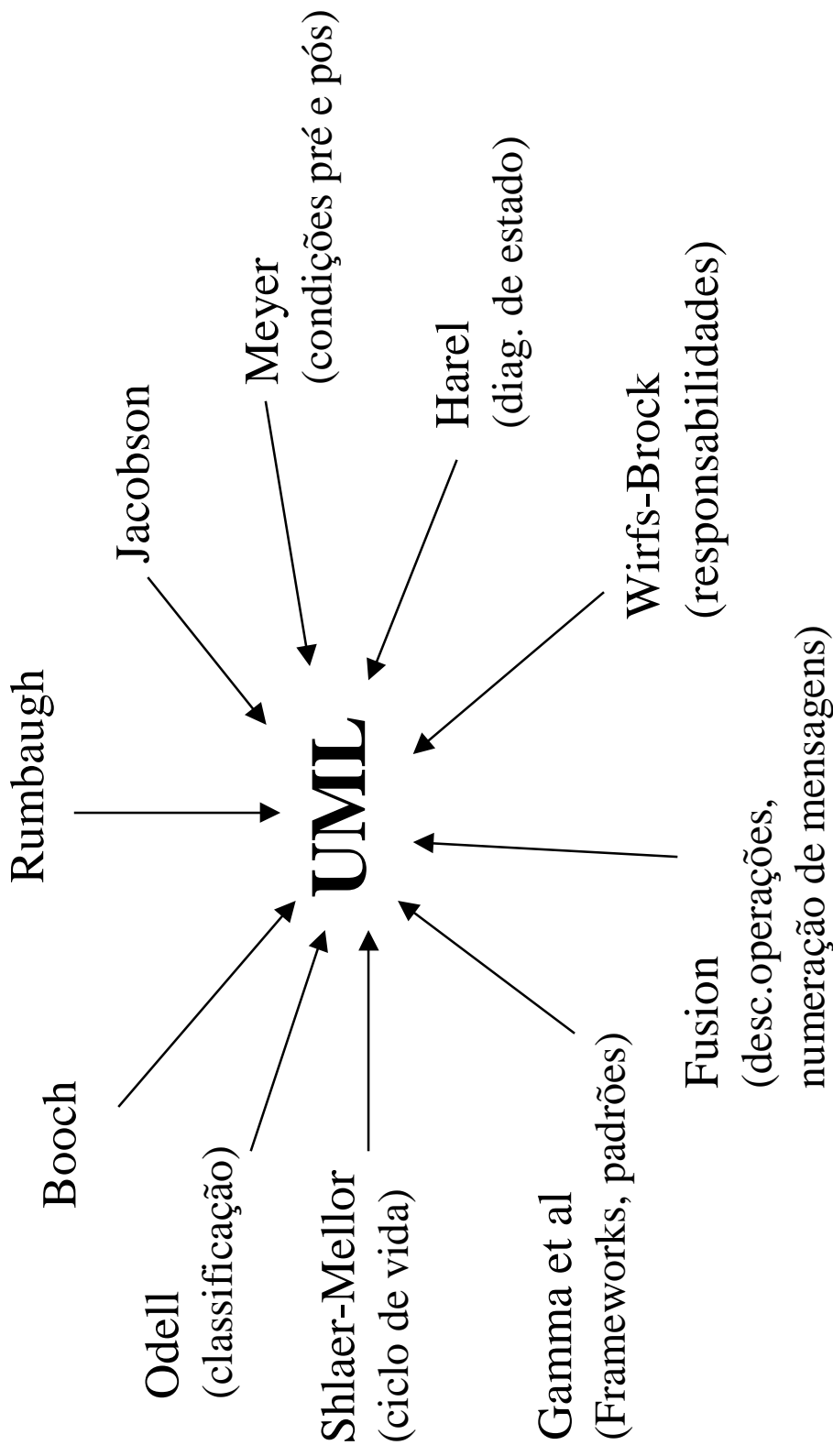
Escolher um Método como sendo o método **principal** para ser seguido e ater-se à sua notação para todo o ciclo de vida.

Usar **técnicas de outros métodos** para dar suporte aos esforços de modelagem e desenvolvimento.

• • • • • • • • • •

•  
•  
•

# UML - Unified Modeling Language



• • • • • • • • • •

•  
•  
•

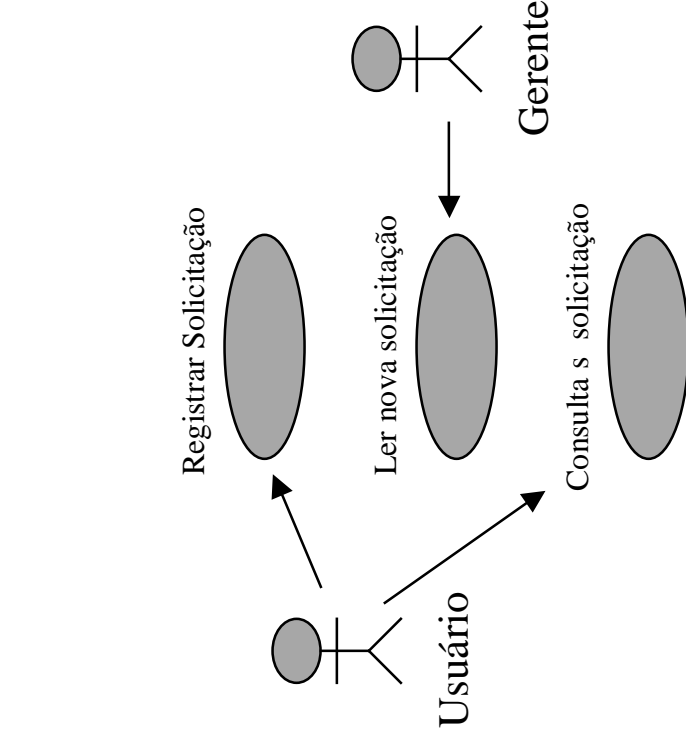
# Análise Orientada a Objetos

•  
•  
•  
•  
•  
•  
•  
•  
•

- # Análise Dinâmica vs Análise Estática

A **Análise Dinâmica** descreve o comportamento dos objetos em termos de suas mudanças ao longo do tempo

# Análise - Use Case



Ator: Usuário

Atividade: Registrar solicitação no sistema

Evento: necessidade de execução de um serviço

Curso básico de ação:

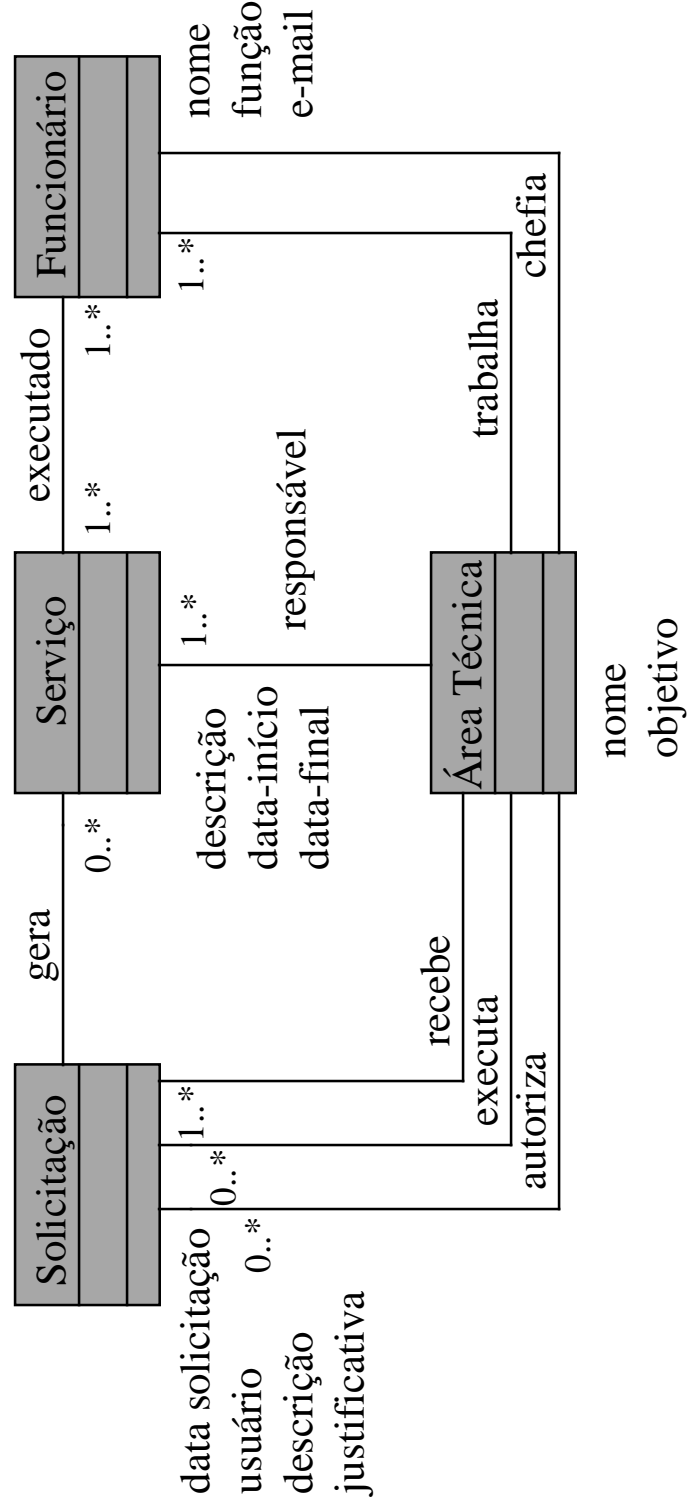
- 1) cliente informa nome, e-mail, telefone,....
- 2) cliente informa área destino dentro do CPD
- 3) cliente escolhe produto na lista de produtos existentes
- 4) o sistema gera um número de solicitação
- 5) apresentar opção de vincular solicitação a outra já existente
- 6) avisar área destino que existe uma nova solicitação para ela

**Diagrama do Use Case**

**Descrição do Use Case**

- 
- 
- 

# Análise - Modelo de Classes



**Diagrama de classes extraído a partir da análise dos *Use Cases***



- # Análise - CRC

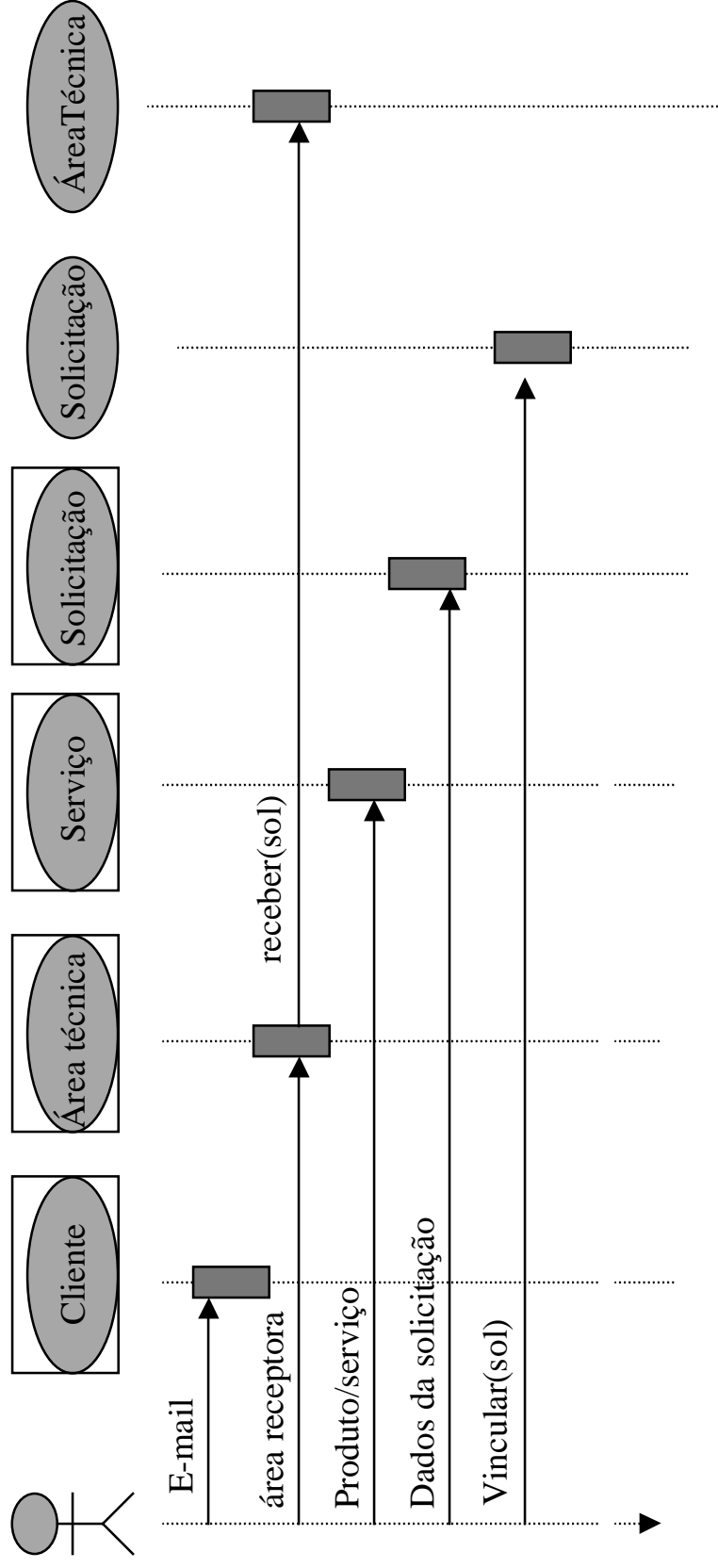
# Clases/responsabilidades/colaboradores

# Feed-back para validar os métodos da classe

•  
•  
•

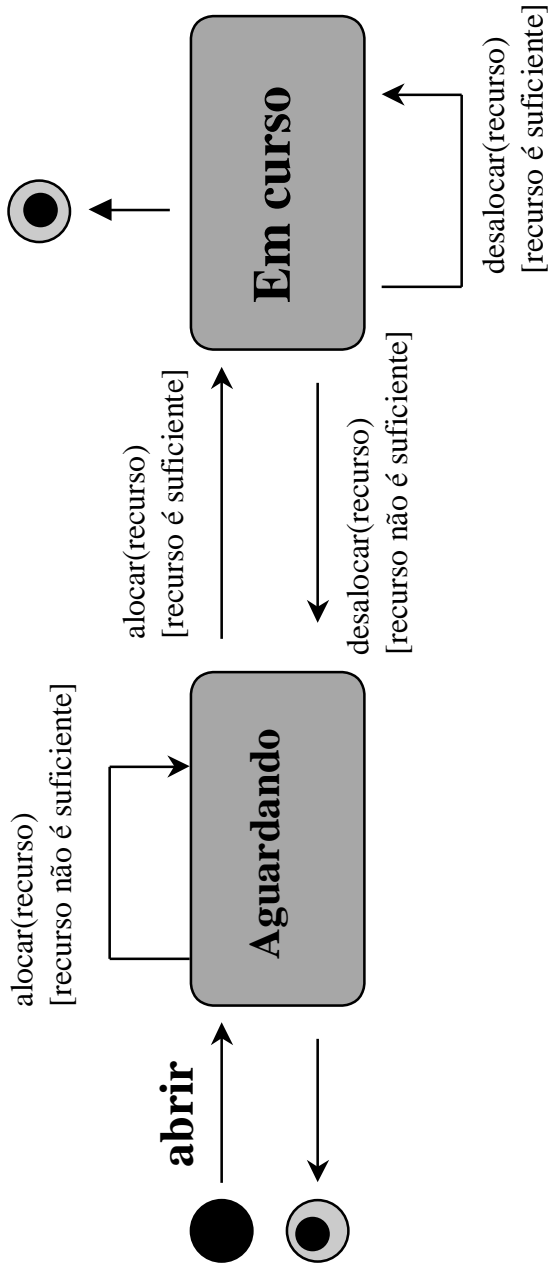
# Análise - Diagrama de Interação (DI)

## Registrar Solicitação no sistema (pelo usuário)



•  
•  
•

# Análise - Diagrama Transição de Estado

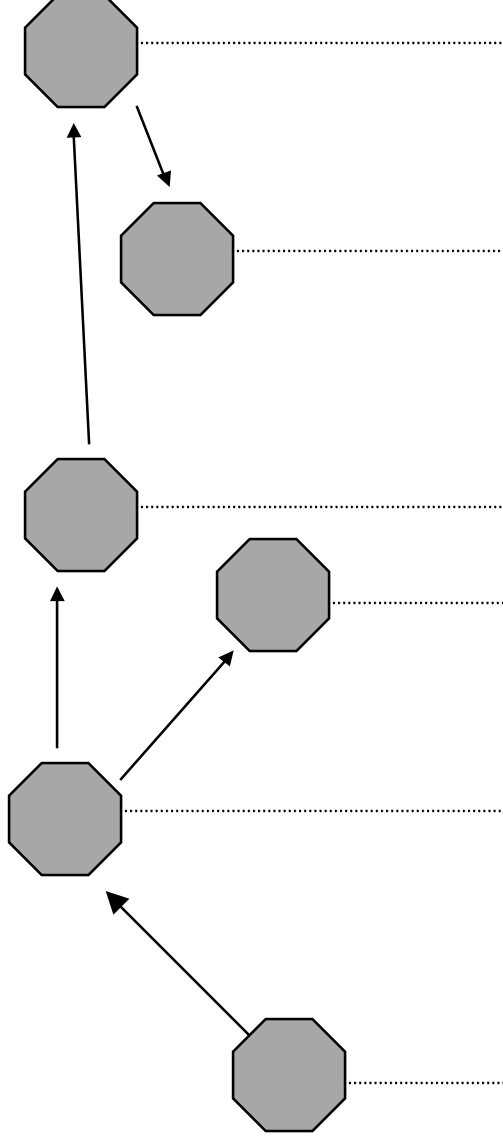




•  
•  
•

# Análise

No final da análise, já sabemos quais os objetos do “negócio”  
envolvidos e como eles devem interagir de forma geral,  
através do **modelo de classes** e do **diagrama de interação**



•

•

•

•

•

•

•

•

•

•

•

•

•

•

•  
•  
•

# Projeto Orientado a Objetos

•  
•  
•  
•  
•  
•  
•  
•  
•

•  
•  
•

# Projeto

**Será que as classes de negócio são suficientes para a implementação do sistema ?**

- ❖ O modelo de objetos pode ser otimizado ?
- ❖ Como o usuário vai interagir com o sistema ?
- ❖ Quem irá controlar o fluxo de mensagens ?
- ❖ Quem vai interagir com o banco de dados ?

• • • • • • • • • •

•  
•  
•

# Projeto

Tanto o domínio do problema, como o domínio da solução  
devem ser entendidos como um conjunto de  
classes de objetos

- ❖ **Classes de interface** - interação com o usuário
- ❖ **Classes de controle** - sequência das mensagens
- ❖ **Classes de banco de dados** - persistência dos dados
- ❖ **Classes de negócio** - relacionadas no modelo de análise

• • • • • • • • • • •



- # Projeto

# Principais atividades de projeto

- Otimização e refinamento do modelo de classes da análise
- Conversão para o modelo relacional de banco de dados

Defina a forma de interação dos usuários com a aplicação sendo construída

Produce uma arquitetura de aplicação a qual inclui decisões sobre a organização do sistema e a alocação de módulos em componentes de hardware e software

- # Projeto

# Tecnologias de apoio

- Design Patterns (padrões de projeto)
- Packages

Utilização de um Guia de Estilo que oriente a utilização correta de recursos gráficos (GUIs), formulários, frames e outros, considerando o ambiente de implementação (por exemplo, Web)

## Arquiteturas de aplicação, frameworks, componentes

• • •

# Projeto de Interface

# Etapas principais

- ❖ Análise de usuários e tarefas
- ❖ Projeto inicial da interface
- ❖ Avaliação da interface sem o usuário
- ❖ Teste da interface com os usuários
- ❖ Construção da interface
- ❖ Acompanhar a utilização da interface pelos usuários
- ❖ Adaptar o projeto de interface

[illegible]

• • •

# Projeto de Interface

# Princípios básicos

- ◆ **Transparência**
- ◆ **Robustez**
- ◆ **Orientação**
- ◆ **Produtividade**
- ◆ **Integridade**

•  
•  
•  
•  
•  
•  
•

•  
•  
•

# Projeto de Interface

## *Princípios básicos (cont)*

- ❖ Controle
- ❖ Clusterização
- ❖ Visibilidade
- ❖ Consistência Inteligente
- ❖ Utilização de cores como complemento

• • • • • • • • • •

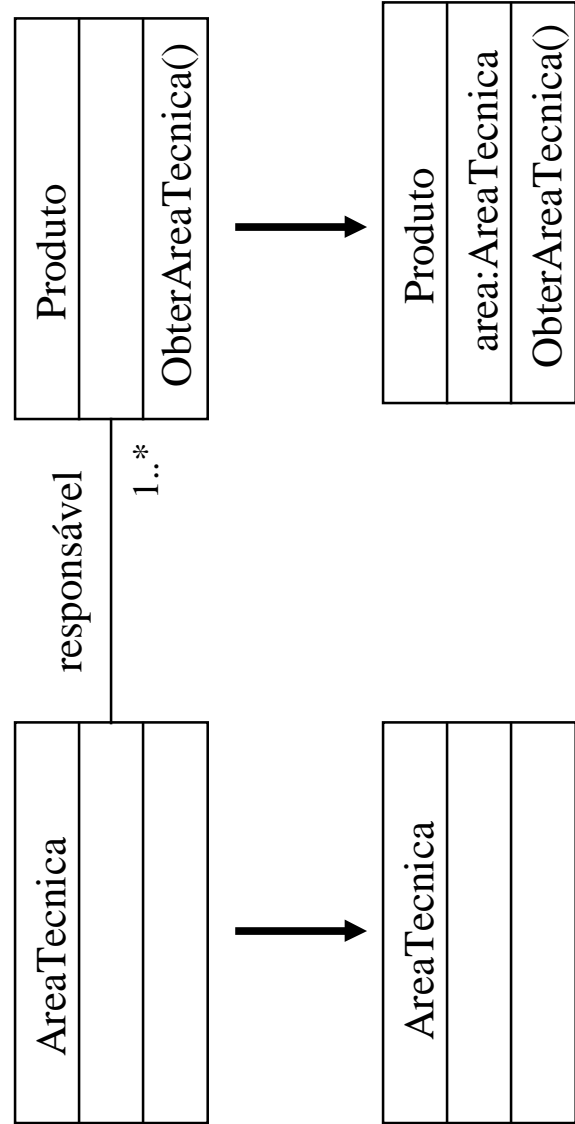
- # Projeto de Objetos

- ❖ Projetar algoritmos para implementar os métodos
- ❖ Otimizar caminhos de acesso aos dados
- ❖ Implementar controle para interações externas
- ❖ Ajustar a estrutura de classes para aumentar a herança
- ❖ Projetar associações
- ❖ Determinar a representação de objetos
- ❖ Empacotar classes e associações em módulos

- 
- 
- 

# Projeto de Objetos

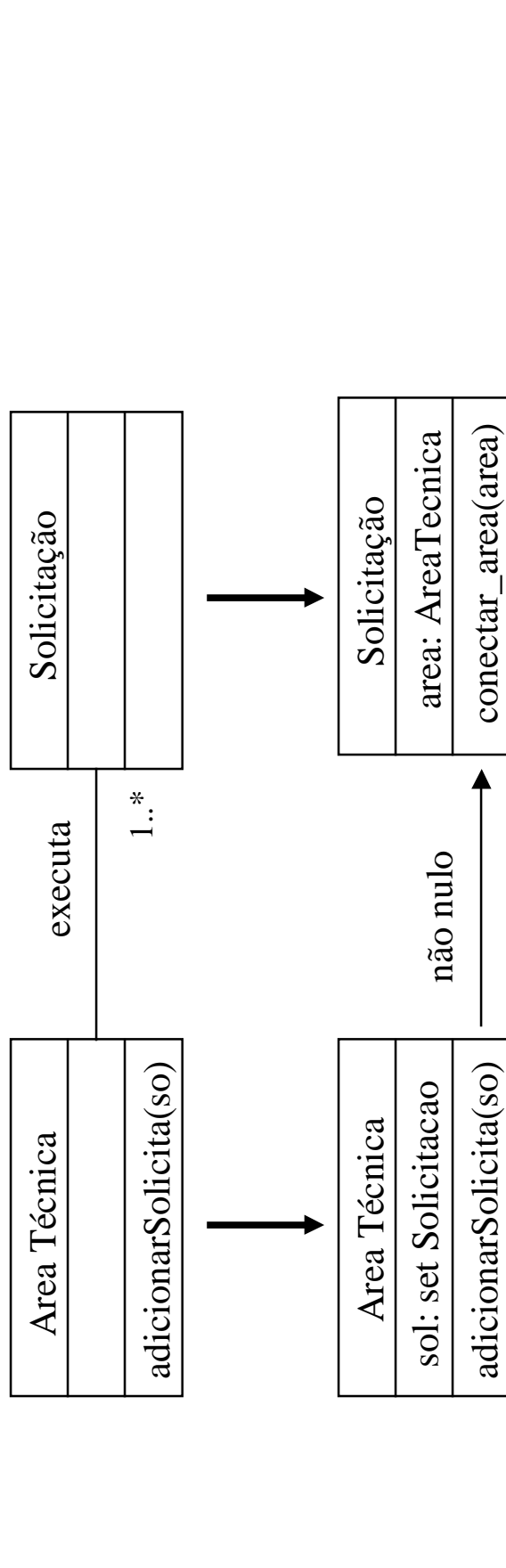
*Exemplo de refinamento do modelo de classes da análise*



Associações de mão única são atravessadas apenas em uma direção  
 Regra Geral: implementada como atributos de ponteiro

# Projeto de Objetos

## *Exemplo de refinamento do modelo de classes da análise*



Associação bidirecional: execução bem-sucedida de *adicionarSolicitação* requer execução bem-sucedida de *conectar\_area* de maneira a ter um vínculo bidirecional consistente



# Projeto de Sistemas

# Questões básicas para a Arquitetura da Aplicação

- ❖ Organização geral e estrutura de controle global
- ❖ Atribuição de funcionalidade aos módulos
- ❖ Composição dos módulos
- ❖ Distribuição física dos módulos
- ❖ Protocolos para comunicação
- ❖ Sincronização e acesso aos dados
- ❖ Expansão e desempenho



•  
•  
•

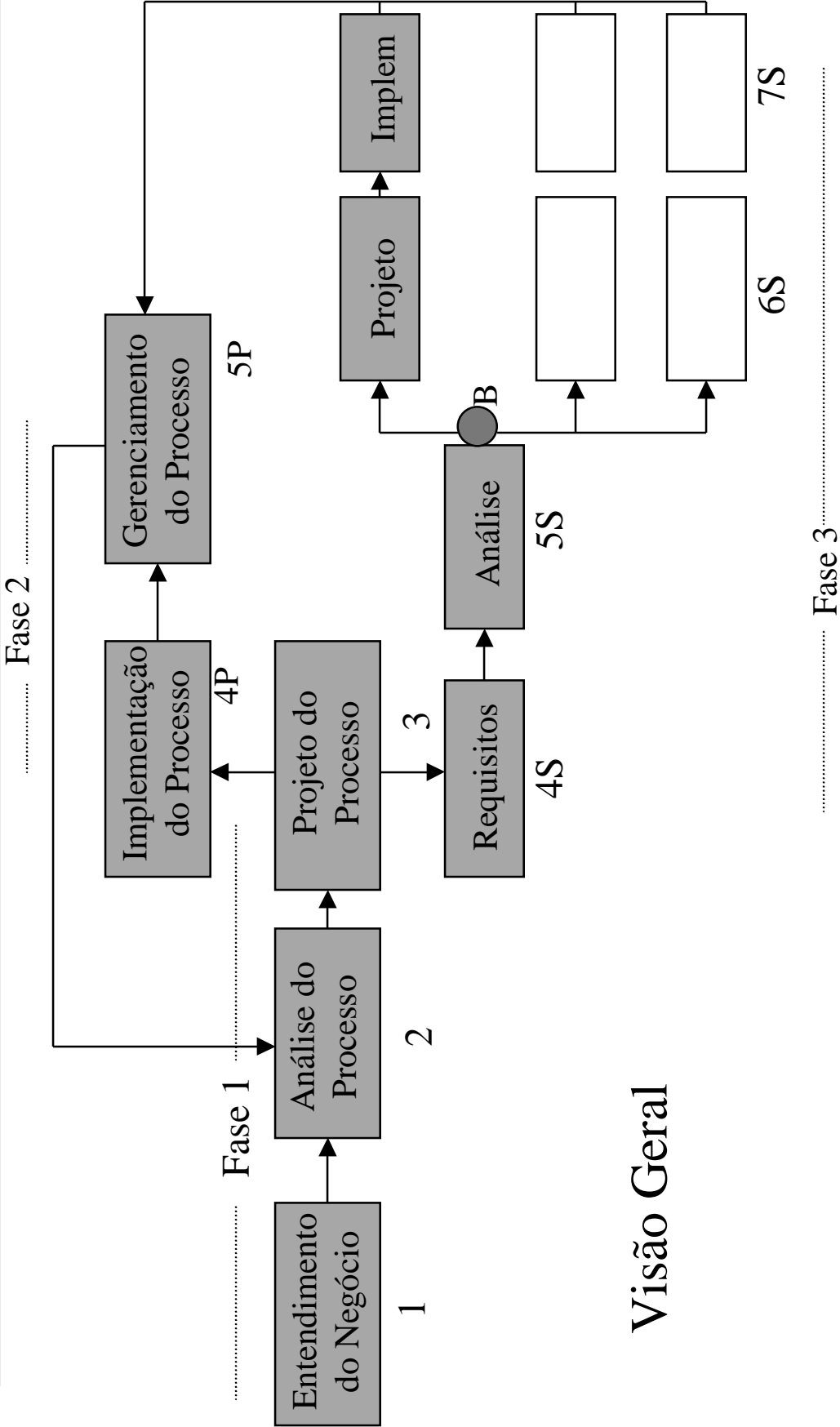
# Tecnologias de Implementação

- ❖ Linguagens Orientadas a Objetos
  - Java, C++, Smalltalk
- ❖ Ambientes de Desenvolvimento
  - VisualAge for Java(IBM)
  - Visual Café (Symantec)
  - Jbuilder 3 (Borland)
- ❖ Banco de Dados Orientado a Objetos
  - O2 (O2 Technology)
  - Objectstore (Object Design Inc.)
  - Gemstone (Servio Logic)
  - Jasmine (Computer Associates)

• • • • • • • • • •

- 
- 
- 

# Método Integrado



# Aprendizado

# Aprendizado

# Implementação de uma metodologia OO

- ❖ Mudança de paradigma - treinamento intensivo
- ❖ Protótipos sem compromissos
- ❖ Primeiros sistemas devem ser livres...
- ❖ Grupo formal de metodologia - proposta e treinamento
- ❖ Acerto do ambiente de desenvolvimento - suporte, padrões
- ❖ Administração de classes/objetos - Biblioteca de Classes
- ❖ Ferramenta CASE

• • •

# Orientação a Objetos

# Benefícios

- ❖ Reaproveitamento
- ❖ Estabilidade
- ❖ Projetista pensa em termos de comportamento dos objetos e não em detalhes de baixo nível
- ❖ Construção de objetos cada vez mais complexos
- ❖ Confiabilidade
- ❖ Verificação de precisão
- ❖ Novos mercados de software
- ❖ Desenvolvimento acelerado
- ❖ Integridade

[illegible]

- # Orientação a Objetos

## Benefícios

-