UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
ESCOLA AGRÍCOLA DE JUNDIAÍ
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CARLA FERNANDES CURVELO
CARLAFCF@GMAIL.COM

TAD0004 - ANÁLISE E PROJETO ORIENTADO A OBJETOS

AULA 2
INTRODUÇÃO

O QUE VEREMOS NESTA AULA...

- Paradigma da orientação a objetos
- Definir análise e projeto OO (A/POO)
- Dar uma visão geral da UML

> Coisas simples devem ser simples, e coisas complexas devem ser possíveis.
>
> **– ALAN KAY**

PARADIGMA DA ORIENTAÇÃO A OBJETOS

O QUE É PARADIGMA?

Forma de abordar um problema

- Idealizado por Alan Kay
- **Substitui o paradigma estruturado** (dados e processos)
- Sistema funciona como um **ser vivo**
- Cada "célula" se comunica com outra "célula" através de mensagens
- As "células" se comportam como uma unidade autônoma

O PARADIGMA DA ORIENTAÇÃO A OBJETOS

O paradigma da OO visualiza um sistema de software como uma coleção de agentes interconectados chamados **objetos**

Cada objeto é responsável por realizar **tarefas** específicas

Para cumprir com algumas das tarefas sob sua responsabilidade, um objeto pode ter que **interagir** com outros objetos

É pela interação entre objetos que uma **tarefa computacional** é realizada

(BEZERRA, 2015, pg. 6)

EXEMPLO

- Suponha que alguém queria comprar uma pizza. Chame este alguém de João. Ele está muito ocupado, e então pede a pizza por telefone. João liga para a pizzaria e faz o pedido, informando ao atendente (digamos, José) seu nome, as características da pizza e seu endereço.

- José, que só tem a função de atendente, informa a Maria (responsável por preparar as pizzas) qual pizza deve ser feita. Quando Maria termina a pizza, José chama Antônio, o entregador, que leva a pizza para a casa de João e entrega a ele 30 minutos depois do pedido.

PRINCÍPIOS DA ORIENTAÇÃO A OBJETOS

1. Qualquer coisa é um objeto
2. Objetos realizam tarefas por meio da requisição de serviços a outros objetos
3. Cada objeto pertence a uma determinada *classe*. Uma classe agrupa objetos similares
4. A classe é um repositório para comportamento associado ao objeto
5. Classes são organizadas em hierarquias

OBJETOS

- O ser humano se relaciona com o mundo através do conceito de **objetos**
- Estamos sempre **identificando** objetos ao nosso redor
  - Atribuindo nomes
  - Classificando em grupos (**classes**)
  - Enviando mensagens a outros objetos para realizar serviços

DEFINIÇÃO

Um objeto é a representação computacional de um elemento ou processo do mundo real

Cada objeto possui um conjunto de **características** ou **comportamentos**

PRINCÍPIOS DA ORIENTAÇÃO A OBJETOS

1. Qualquer coisa é um objeto (João, José, Maria e Antônio)
2. Objetos realizam tarefas por meio da requisição de serviços a outros objetos (Todos colaboram com uma parte e o objetivo é alcançado)
3. Cada objeto pertence a uma determinada *classe*. Uma classe agrupa objetos similares (Antônio é um objeto da classe Entregador)
4. A classe é um repositório para comportamento associado ao objeto (Comportamento de Antônio é o mesmo de todos os entregadores)
5. Classes são organizadas em hierarquias (Antônio é Entregador, Humano, Mamífero, ...)

CARACTERÍSTICAS E COMPORTAMENTOS

- Uma **característica** descreve uma **propriedade** de um objeto
- Um **comportamento** representa uma **ação** ou **resposta** de um objeto a uma ação do mundo real

OBJETO: CARRO

Características: cor, marca, número de portas, ano de fabricação, tipo de combustível

Comportamento: acelerar, parar, andar, estacionar

CARACTERÍSTICAS E COMPORTAMENTOS

OBJETO: CACHORRO

Características:

Comportamento:

OBJETO: BICICLETA

Características:

Comportamento:

CARACTERÍSTICAS E COMPORTAMENTOS

OBJETO: CACHORRO

Características: nome, cor, raça

Comportamento: latir, correr

OBJETO: BICICLETA

Características: marcha atual, velocidade atual

Comportamento: trocar marcha, aplicar freio

ORIENTADO A OBJETOS

- Um programa é visto como um **conjunto de objetos**
- **Elementos que fazem parte da solução de um problema**
- Possuem **atributos** (características) e **funcionalidades** (comportamento)

EXEMPLOS: SISTEMA DE GESTÃO ACADÊMICA (SIGAA)

- Que classes podemos identificar?
- Exemplos de objetos?
- Que mensagens podem ser trocadas?

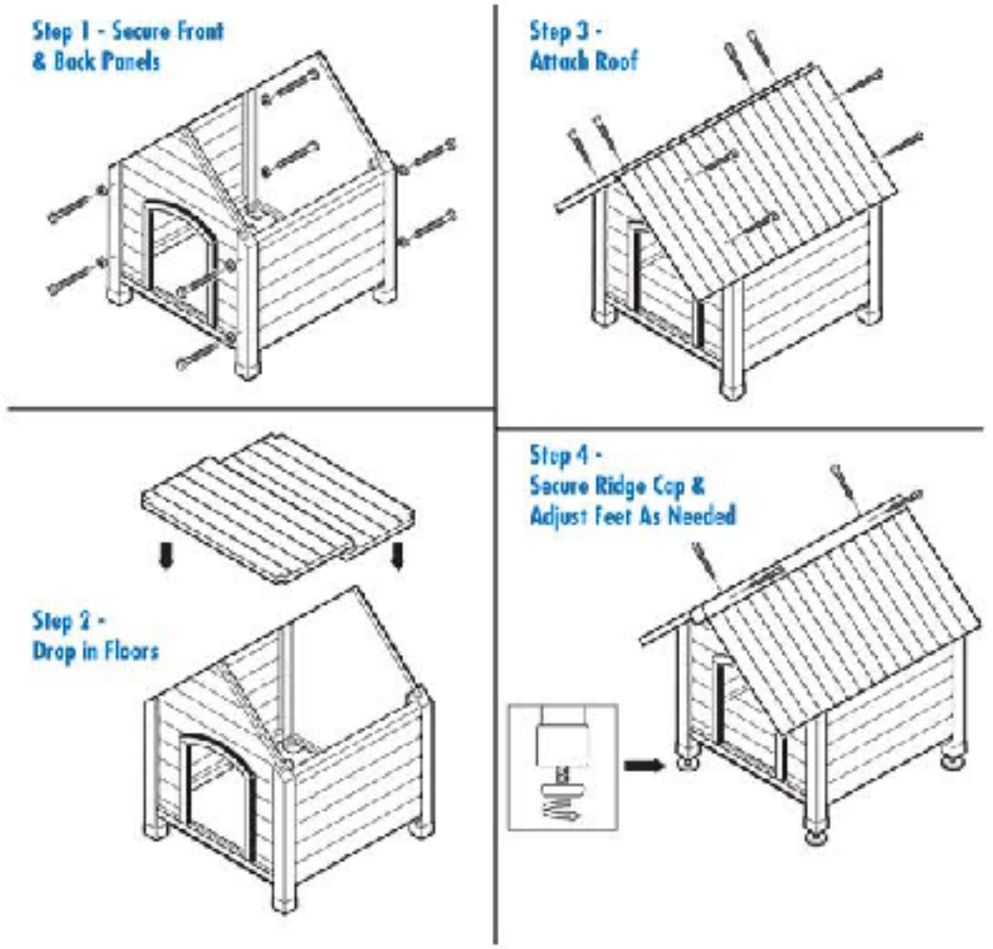
EXEMPLOS: SISTEMA DE GESTÃO ACADÊMICA (SIGAA)

- **Que classes podemos identificar?**
  - Alunos, Professores, Pessoas, Turma, Disciplina, Curso, ...
- **Exemplos de objetos?**
  - Professora Carla, Aluno João, Disciplina TAD0004, ...
- **Que mensagens podem ser trocadas?**
  - Matricular aluno em uma disciplina, Definir professor da turma, ...

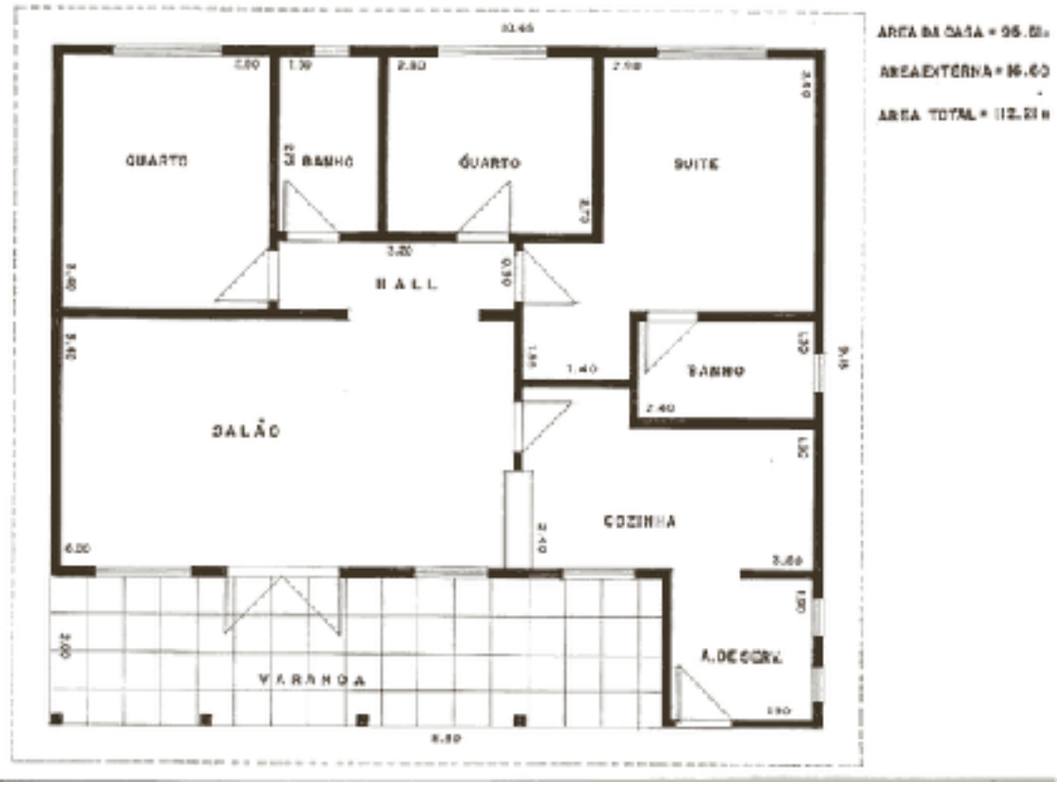
MODELAGEM DE SISTEMAS

MODELAGEM DE SISTEMAS

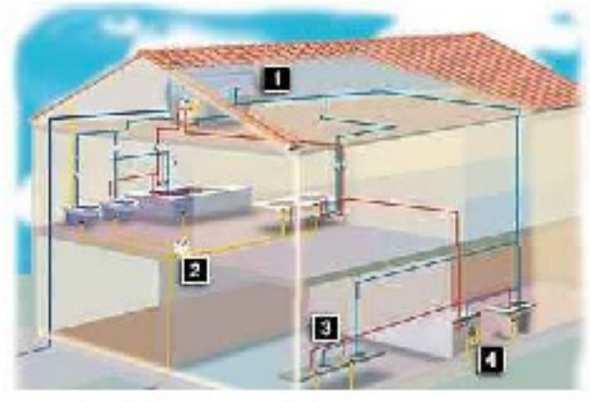
- A complexidade do desenvolvimento de um sistema cresce a medida que o tamanho do sistema cresce



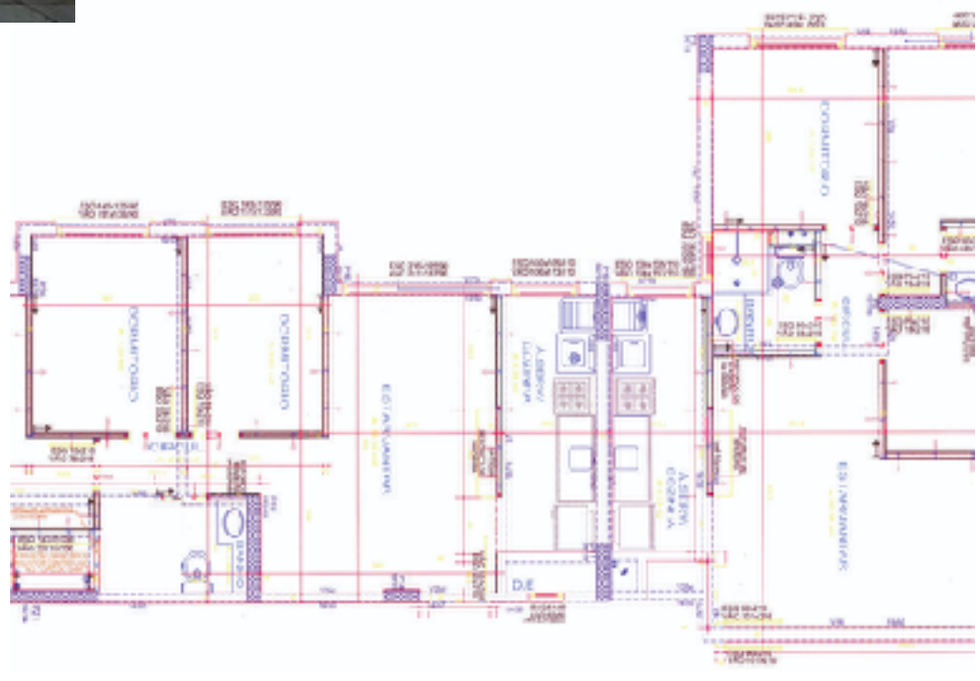
MODELAGEM DE SISTEMAS



Através das paredes



esgoto
água fria
água quente



MODELAGEM DE SISTEMAS

- Na construção de um casa ou edifício, engenheiros e arquitetos constroem plantas dos diversos elementos da habitação antes do início da construção propriamente dita
  - Plantas hidráulicas, elétricas, de fundação etc. são projetadas e devem manter consistência entre si
- Para construção de sistemas de software, é necessário também um planejamento inicial

MODELOS

- Forma de **representar** a realidade (problema, situação..)
- Os modelos em geral empregam **abstração** de modo a diminuir a complexidade do problema, retirando detalhes de menor importância para um determinado contexto
- Todos nós já lidamos com modelos
  - Quando explicamos algo a uma pessoa usando fotografias ou gráficos, ou quando representamos planos ou sólidos usando equações matemáticas. Estamos interpretando a realidade através de **metáforas de substituição** (modelos)
- Como é feita a modelagem em *softwares*?
  - Utilizando **UML**

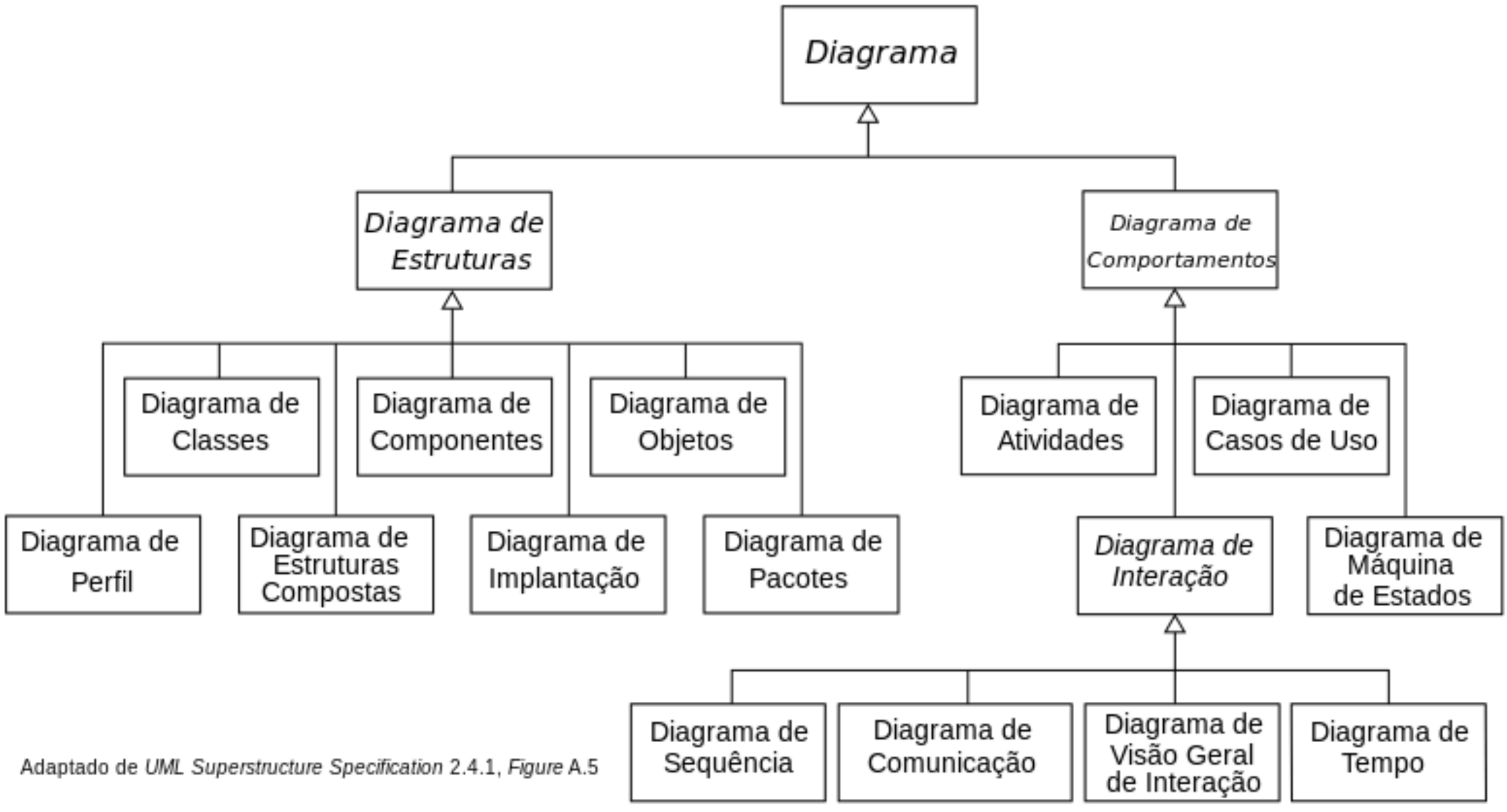
O QUE É UML?

UML: LINGUAGEM DE MODELAGEM UNIFICADA

Linguagem visual para especificar, construir e documentar artefatos dos sistemas OO

- Notação diagramática **padrão**
- Idealizada e desenvolvida pelos *três amigos*: Booch, Rumbaugh e Jacobson
  - UML 1.0 foi lançada em 1997
  - Atualmente: UML 2.5
- Focaremos nos diagramas mais utilizados e nos seus recursos mais comuns

DIAGRAMAS UML



Adaptado de *UML Superstructure Specification* 2.4.1, *Figure* A.5

UML: MODOS DE APLICAÇÃO

- **UML como rascunho:** diagramas incompletos e informais
- **UML como planta de *software*:** diagramas detalhados
  - Engenharia reversa: gerado a partir do código
  - Geração de código: fornecem uma base para a geração de códigos, manualmente ou automaticamente
- **UML como linguagem de programação:** grande número de diagramas detalhados
  - Código completamente gerado através de digramas de classes, sequência, estado, entre outros
- **Modelagem Ágil:** UML como rascunho

UML: PERSPECTIVAS DE APLICAÇÃO

- **Perspectiva conceitual:** descrever coisas do mundo real

- **Perspectiva de especificação (*software*):** descrever abstrações de um *software*, sem nenhum comprometimento com uma implementação particular

- **Perspectiva de implementação (*software*):** descrever implementações de *software* com uma tecnologia específica

POR QUE UTILIZAR MODELOS EM SISTEMAS DE SOFTWARE?

1. Gerenciamento da complexidade
2. Comunicação entre as pessoas envolvidas
3. Redução dos custos no desenvolvimento
   - Modelos de sistemas são mais fáceis (menos custosos) para construir e para alterar
4. Previsão do comportamento futuro do sistema

FERRAMENTAS CASE

- **CASE (*Computer Aided Software Engineering*)**
  - Criação de diagramas
  - Garantia de interoperabilidade entre ferramentas CASE
  - Manutenção de consistência
  - Engenharia direta: geração de códigos através de diagramas
  - Engenharia reversa: geração de diagramas através de códigos

OBJETIVO DA DISCIPLINA

- Ao final desta disciplina o aluno deverá ser capaz de **realizar análises de problemas do mundo real, bem como projetar soluções para esses problemas utilizando UML para produzir abstrações orientadas a objetos**

INTRODUÇÃO À A/POO

- Não se refere apenas à UML
  - UML: notação padrão de diagramação
  - IMPORTANTE: criar um excelente projeto OO ou avaliar e melhorar um existente
- **A/POO**
  - *"Faça a coisa certa e faça certo a coisa"*

INTRODUÇÃO À A/POO

- **Análise (*"Faça a coisa certa"*)**
  - Investigação do problema e dos requisitos
  - Análise de requisitos: investigação dos requisitos
  - Análise OO: investigação dos objetos do domínio
  - Exemplo: novo sistema online de comercialização
    - Como será usado?
    - Quais as suas funções?
- **Projeto (*"Faça certo a coisa"*)**
  - Solução conceitual (e não sua implementação)
  - Projeto de objetos
  - Projeto de banco de dados

INTRODUÇÃO À A/POO – EXEMPLO

- **Sistema de informação de voos**
  - Análise de objetos:
    - *avião*, *voo*, *piloto*
  - Projeto de objetos:
    - *avião* → atributo: *numDaCauda*
    - *avião* → método: *obterHistoricoDoVoo*

INTRODUÇÃO À A/POO – EXEMPLO

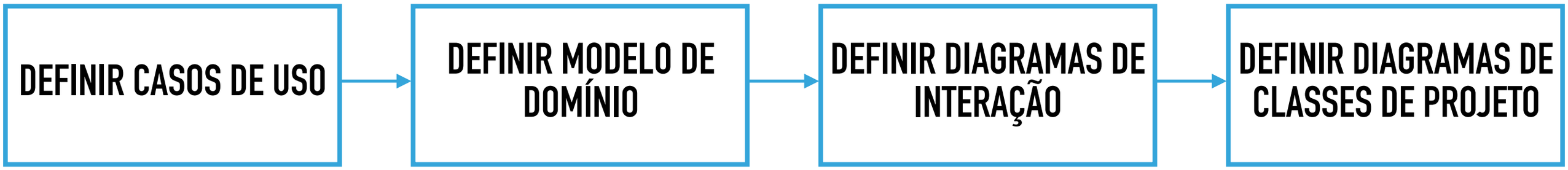
- **Sistema de informação de voos**
  - Análise de objetos:
    - *avião*, *voo*, *piloto*
  - Projeto de objetos:
    - *voo*
    - *piloto*

INTRODUÇÃO À A/POO – EXEMPLO

- **Sistema de um hospital**
  - Análise de objetos:
  - Projeto de objetos:

INTRODUÇÃO À A/POO: EXEMPLO

- **Jogo de dados**
  - O jogador joga dois dados, e se a soma dos valores for 7 ele vence; caso contrário, perde.

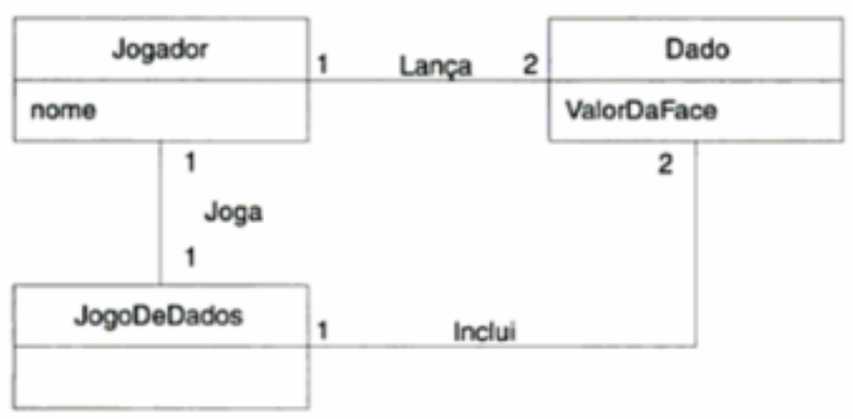


INTRODUÇÃO À A/POO: EXEMPLO

DEFINIR CASOS DE USO → DEFINIR MODELO DE DOMÍNIO → DEFINIR DIAGRAMAS DE INTERAÇÃO → DEFINIR DIAGRAMAS DE CLASSES DE PROJETO

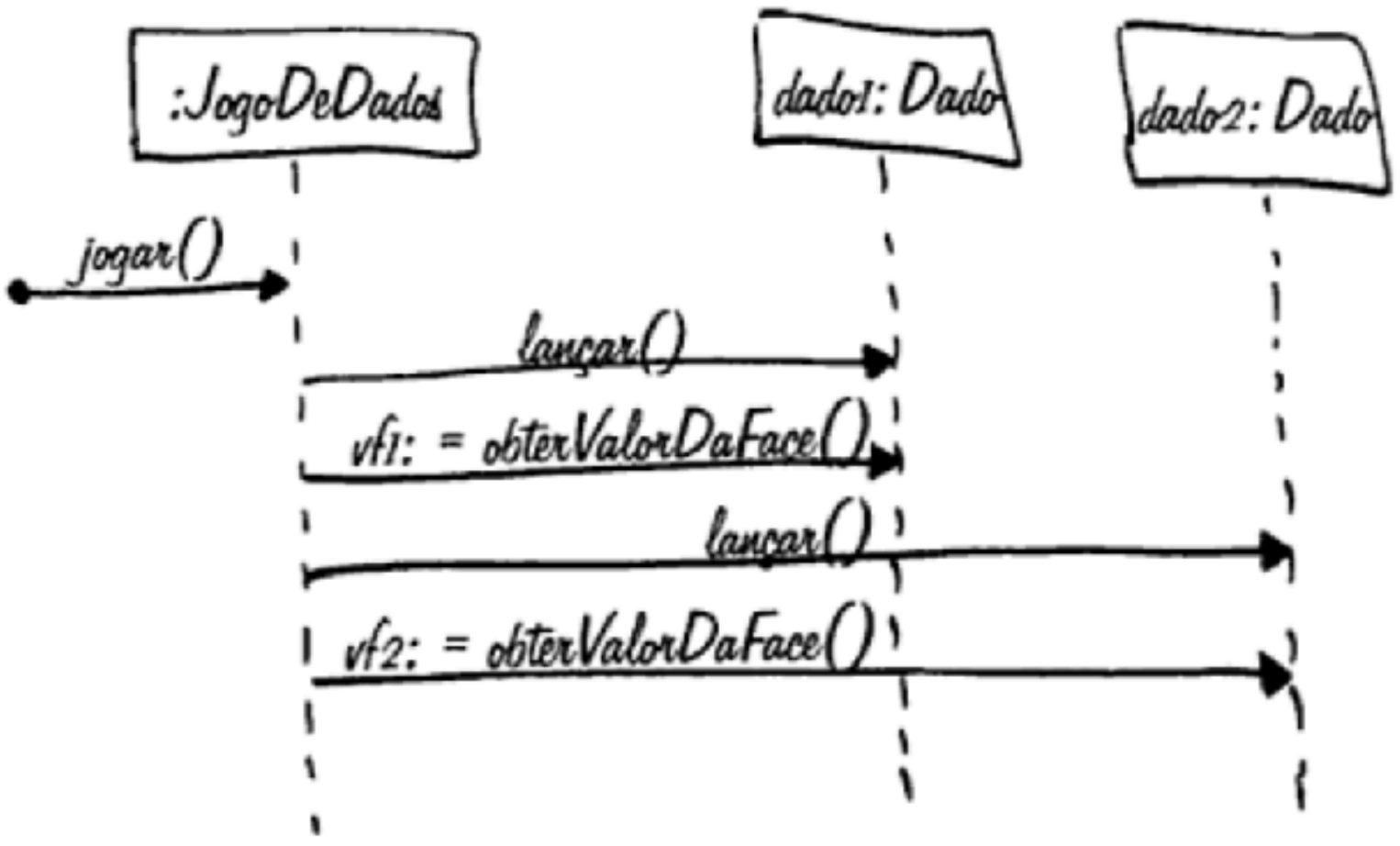
- Narrativas escritas

DEFINIR CASOS DE USO → **DEFINIR MODELO DE DOMÍNIO** → DEFINIR DIAGRAMAS DE INTERAÇÃO → DEFINIR DIAGRAMAS DE CLASSES DE PROJETO



INTRODUÇÃO À A/POO: EXEMPLO

DEFINIR CASOS DE USO → DEFINIR MODELO DE DOMÍNIO → **DEFINIR DIAGRAMAS DE INTERAÇÃO** → DEFINIR DIAGRAMAS DE CLASSES DE PROJETO



INTRODUÇÃO À A/POO: EXEMPLO

DEFINIR CASOS DE USO → DEFINIR MODELO DE DOMÍNIO → DEFINIR DIAGRAMAS DE INTERAÇÃO → **DEFINIR DIAGRAMAS DE CLASSES DE PROJETO**

BIBLIOGRAFIA

- LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimeto interativo.** 3.ed. Porto Alegre: Bookman, 2007. xiv, 695 p. [cap. 1]

- BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 3. ed. rev atual. Rio de Janeiro, RJ: Elsevier, 2015. 393 p. [cap. 1]