

# Análise e Projeto de Sistemas

Universidade Federal do Ceará – UFC

Campus de Quixadá

Curso de Sistemas de Informação

Prof. Marcos Antonio de Oliveira (deoliveira.ma@gmail.com)

“Coisas simples devem ser simples e coisas complexas devem ser possíveis.” (ALAN KAY )

# **VISÃO GERAL**

Esses slides são uma adaptação das notas de aula do professor Eduardo Bezerra autor do livro Princípios de Análise e Projeto de Sistemas com UML

# Índice

- Introdução
- Modelagem de sistemas de software
- Evolução histórica da modelagem de sistemas
- Paradigma Orientado a Objetos
- A Linguagem de Modelagem Unificada (UML)

# INTRODUÇÃO

# Sistemas de Informações

- *“A necessidade é a mãe das invenções”*
  - Em consequência do crescimento da importância da informação, surgiu a necessidade de gerenciar informações de uma forma adequada e eficiente e, desta necessidade, surgiram os denominados ***sistemas de informações***.

# Sistemas de Informações

- Um SI é uma combinação de pessoas, dados, processos, interfaces, redes de comunicação e tecnologia que interagem com o objetivo de dar suporte e melhorar o processo de negócio de uma organização com relação às informações
  - Vantagens do ponto de vista competitivo
- Objetivo principal e final da construção de um SI: *adição de valor*

# Sistemas de Software

- Um dos componentes de um SI é denominado **sistema de software**
- Compreende os módulos funcionais computadorizados que interagem entre si para proporcionar a automatização de diversas tarefas
- Característica intrínseca do desenvolvimento de sistemas de software: **complexidade**

# Sistemas de Software

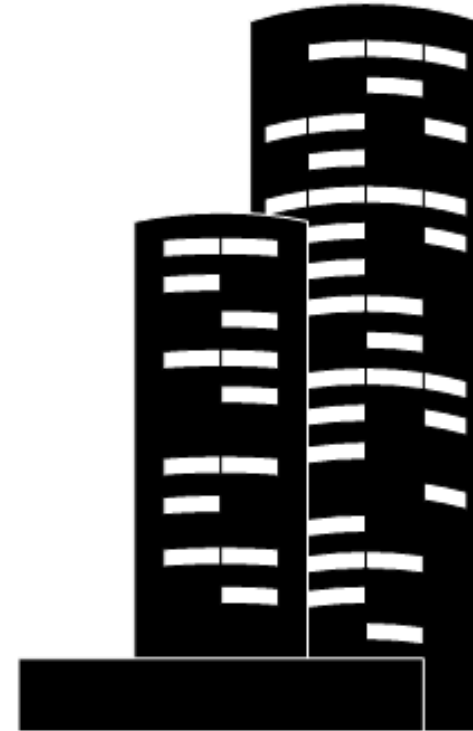
## ■ Uma analogia...



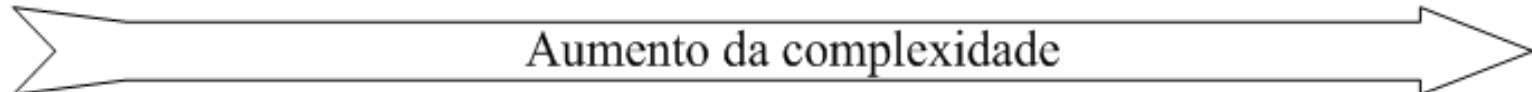
Casa de Cachorro



Casa



Arranha-céu





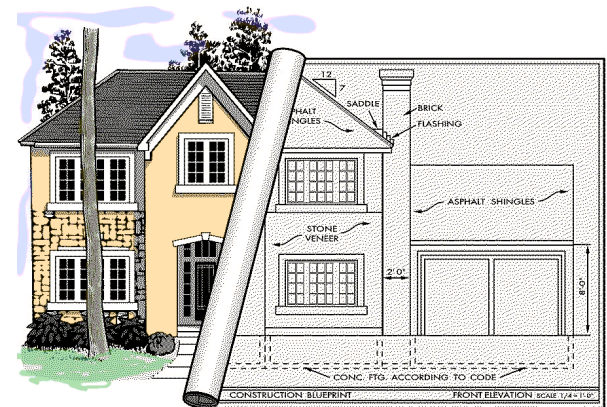
# Sistemas de Software

- O que é um *software*?

# **MODELAGEM DE SISTEMAS**

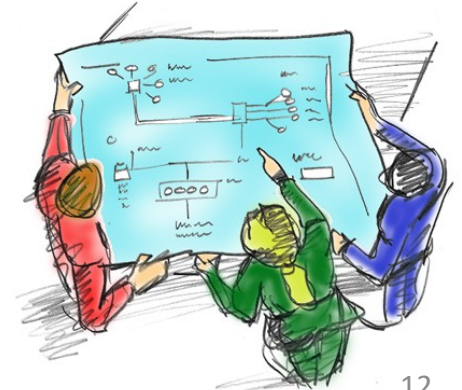
# Complexidade

- Na construção de sistemas de software, assim como na construção de sistemas habitacionais, também há uma gradação de complexidade
  - A construção desses sistemas necessita de um planejamento inicial



# Modelos

- De uma perspectiva mais ampla, um modelo pode ser visto como uma representação idealizada de um sistema a ser construído
- Maquetes de edifícios e de aviões e plantas de circuitos eletrônicos são apenas alguns exemplos de modelos



# Razões para Construção de Modelos

- Gerenciamento da complexidade inerente ao desenvolvimento de software
- Comunicação entre as pessoas envolvidas
- Redução dos custos no desenvolvimento
- Predição do comportamento futuro do sistema

# Razões para Construção de Modelos

- Alguém aqui já precisou “desenhar” um software antes de começar a programar?
- Quais ferramentas vocês usaram? Ou “fez na mão”?

# Diagramas

- No contexto de desenvolvimento de software, correspondem a desenhos gráficos que seguem algum padrão lógico (***diagramas***)
- Um diagrama é uma apresentação de uma coleção de ***elementos gráficos*** que possuem um significado predefinido

# Diagramas

- Diagramas fornecem uma representação concisa do sistema. “**uma figura vale por mil palavras**”



- **Cuidado!!!** Modelos também são compostos de informações textuais
- Dado um modelo de uma das perspectivas de um sistema, diz-se que o seu diagrama, juntamente com a informação textual associada, formam a **documentação** deste modelo



# Modelagem de Software

“A modelagem de sistemas de software consiste na utilização de notações gráficas e textuais com o objetivo de construir modelos que representam as partes essenciais de um sistema, considerando-se diversas perspectivas diferentes e complementares”

# **EVOLUÇÃO HISTÓRICA DA MODELAGEM DE SISTEMAS**

# Evolução do Hardware

- A ***Lei de Moore*** é bastante conhecida na computação

*“A densidade de um transistor dobra em um período entre 18 e 24 meses” (Lei de Moore)*

- Essa lei foi declarada em 1965 pelo engenheiro Gordon Moore, co-fundador da Intel

# Evolução do Hardware

- A ***Lei de Moore*** (na verdade, um chute) vem se verificando há vários anos
- O rápido crescimento da capacidade computacional das máquinas resultou na demanda por sistemas de software cada vez mais complexos

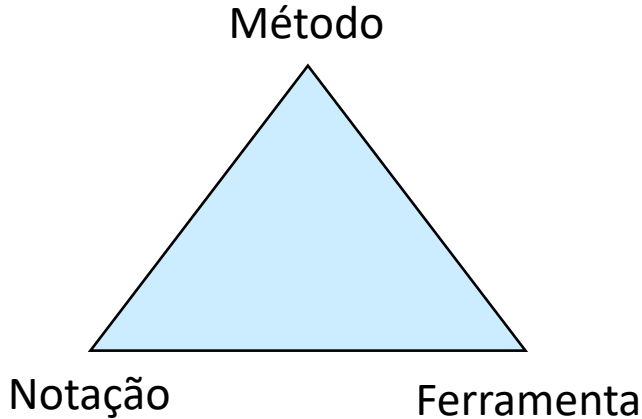
# Evolução do Software

- O surgimento de sistemas de software mais complexos resultou na necessidade de reavaliação da forma de se desenvolver sistemas
- Consequentemente as técnicas utilizadas para a construção de sistemas computacionais têm evoluído de forma impressionante, notavelmente no que tange à modelagem de sistemas

# Evolução do Software

- Na primeira metade da década de 90 surgiram várias propostas de técnicas para modelagem de sistemas segundo o **paradigma orientado a objetos**
- Houve uma grande proliferação de propostas para modelagem orientada a objetos
  - Diferentes notações gráficas para modelar uma mesma perspectiva de um sistema
  - Cada técnica tinha seus pontos fortes e fracos

# A Tríade do Desenvolvimento



## Exemplos

Método: RUP, OpneUP, XP, ...

Notação: UML, DER, ...

Ferramenta: Astah,  
StarUML,...

- O **método** é essencial
- A **notação** padroniza a comunicação
- A **ferramenta** agiliza automatizando
- Dá para trabalhar sem uma **ferramenta**!
- Fica muito ruim sem uma **notação** conhecida
- É improdutivo sem **método**

# A Tríade do Desenvolvimento

- Qual **método** você já usou?
- Teve alguma **notação** no processo de desenvolvimento?
- Que **ferramentas** foram usadas?



# **PARADIGMA ORIENTADO A OBJETOS**

# Paradigma?

- *Um paradigma é uma forma de abordar um problema*
- O paradigma da OO surgiu no fim dos anos 60
- Hoje em dia, praticamente suplantou o paradigma anterior
  - *O paradigma estruturado...*

# O Paradigma da Orientação a Objetos

- Alan Kay, um dos pais do paradigma da orientação a objetos, formulou a chamada **analogia biológica**.
- “Como seria um sistema de software que funcionasse como um ser vivo?”

# Analogia Biológica

- Cada “célula” interagiria com outras células através do envio de mensagens para realizar um objetivo comum
- Adicionalmente, cada célula se comportaria como uma unidade autônoma

# Analogia Biológica

- De uma forma mais geral, Kay pensou em como construir um sistema de software a partir de agentes autônomos que interagem entre si
- Com isso, ele estabeleceu os princípios da **orientação a objetos**

# Princípios da Orientação a Objetos

- Qualquer “coisa” é um objeto
- Objetos realizam tarefas através da requisição de serviços a outros objetos
- Cada objeto pertence a uma determinada classe. Uma classe agrupa objetos similares
- A classe é um “repositório” para comportamento associado ao objeto
- Classes são organizadas em hierarquias

# Orientação a Objetos

“O paradigma da orientação a objetos visualiza um sistema de software como uma coleção de agentes interconectados chamados objetos. Cada objeto é responsável por realizar tarefas específicas. É através da interação entre objetos que uma tarefa computacional é realizada.”

# Orientação a Objetos

“Um sistema de software orientado a objetos consiste de objetos em colaboração com o objetivo de realizar as funcionalidades deste sistema. Cada objeto é responsável por tarefas específicas. É através da cooperação entre objetos que a computação do sistema se desenvolve.”



# Conceitos da Orientação a Objetos

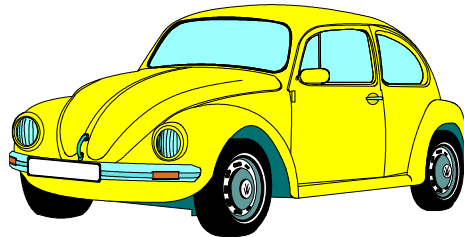
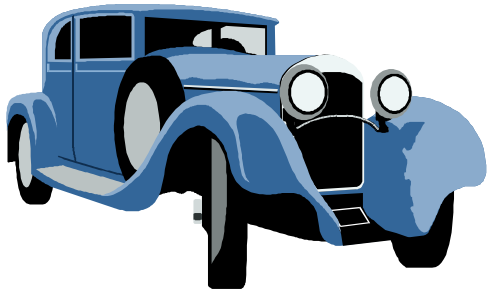
- Classes e objetos
- Mensagens
- Encapsulamento
- Polimorfismo
- Herança

# Classes e Objetos

- O mundo real é formado de coisas...
- ... em OO os objetos representam estas coisas do mundo real
- Os seres humanos costumam agrupar os objetos para entendê-los...
- ... a descrição de um grupo de objeto é denominada **classe de objetos**, ou simplesmente de **classe**

# O que é uma Classe?

- Uma classe é um “molde” para objetos. Diz-se que um objeto é uma “instância” de uma classe



# Classes e Objetos

- **Importante!**
  - Uma classe é uma abstração das características relevantes de um grupo de “coisas” do mundo real
- **Cuidado!**
  - Na maioria das vezes, um grupo de objetos do mundo real é muito complexo para que todas suas características sejam representadas em uma classe

# Objetos como Abstrações

- Uma abstração é uma representação das características relevantes de um conceito do mundo real para um determinado problema
  - Carro (para uma transportadora de cargas)
  - Carro (para uma fábrica de automóveis)
  - Carro (para um colecionador)
  - Carro (para uma empresa de kart)
  - Carro (para um mecânico)

# Classes X Objetos

- Classes são definições estáticas, que possibilitam o entendimento de um grupo de objetos
- Objetos são abstrações de entidades que existem no mundo real
- **Cuidado!**
  - Estes dois termos muitas vezes são usados indistintamente



# Mensagens

- Para que um objeto realize alguma tarefa, deve haver um estímulo enviado a ele
- Pense em um objeto como uma entidade ativa que representa uma abstração de algo do mundo real
  - Então faz sentido dizer que tal objeto pode responder a estímulos a ele enviados
  - Assim como faz sentido dizer que seres vivos reagem a estímulos que eles recebem

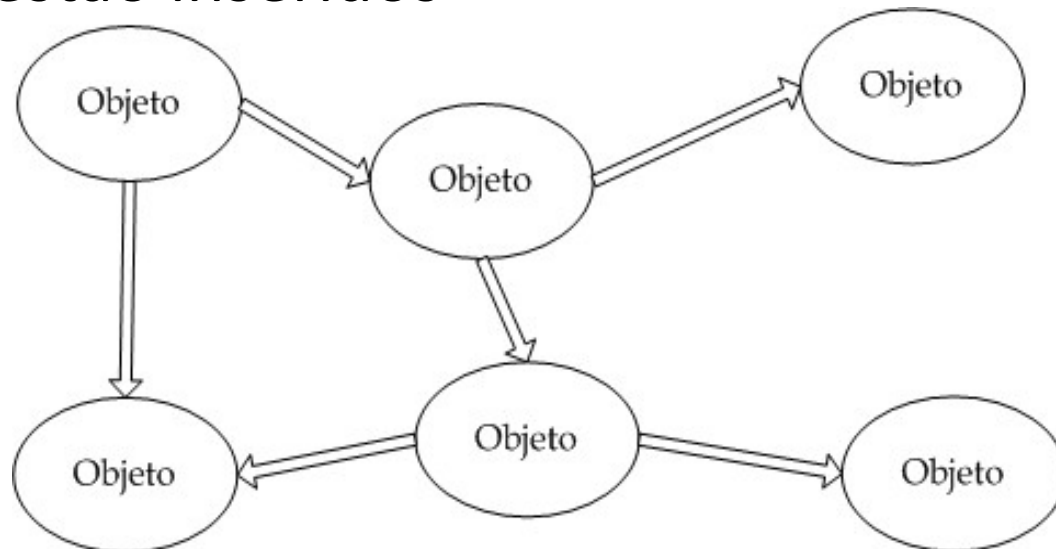
# Mensagens

- Independentemente da origem do estímulo, quando ele ocorre, diz-se que o objeto em questão está recebendo uma **mensagem**
- Uma mensagem é uma requisição enviada de um objeto a outro para que este último realize alguma operação



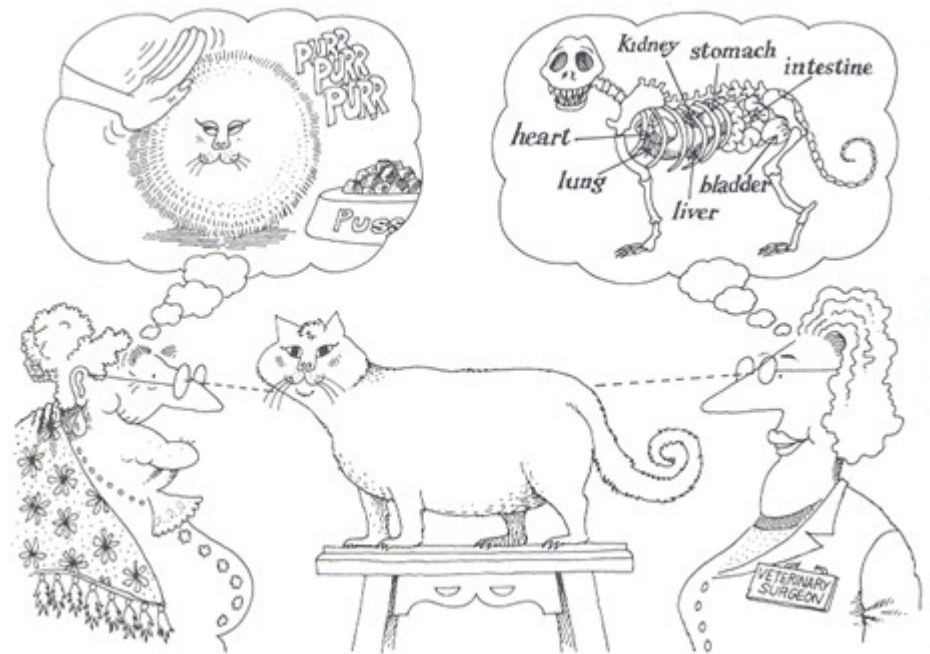
# Mensagens

- Objetos de um sistema trocam mensagens
  - Isto significa que estes objetos estão enviando mensagens uns aos outros com o objetivo de realizar alguma tarefa dentro do sistema no qual eles estão inseridos



# Abstração

- Uma abstração é qualquer modelo que inclui os aspectos relevantes de alguma coisa, ao mesmo tempo em que ignora os menos importantes



# Abstração

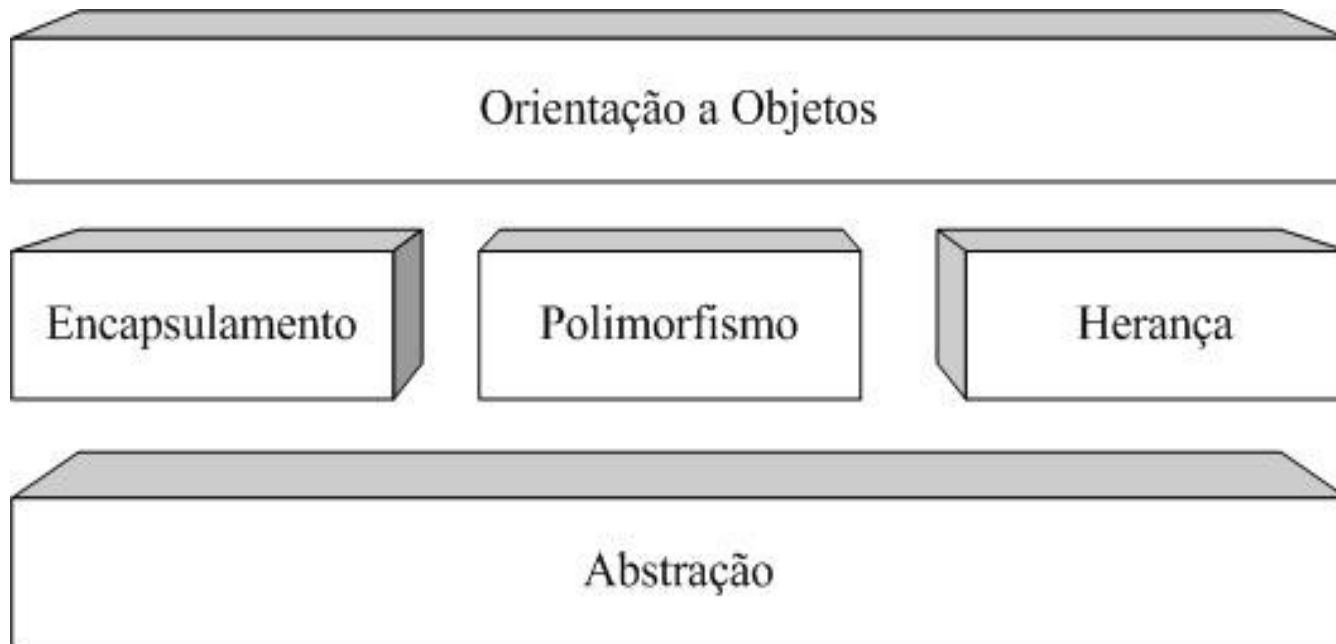
- A abstração depende do observador...



# Abstração na Orientação a Objetos

- A orientação a objetos faz uso intenso de abstrações
  - Os princípios da orientação a objetos podem ser vistos como aplicações do Princípio da Abstração
- Princípios
  - Encapsulamento
  - Polimorfismo
  - Herança

# Abstração na Orientação a Objetos



# Encapsulamento

- Objetos possuem “comportamento”
  - O termo comportamento diz respeito a que operações são realizadas por um objeto e também de que modo estas operações são executadas

# Encapsulamento

- O *encapsulamento* é uma forma de restringir o acesso ao comportamento interno de um objeto
  - Um objeto que precise da colaboração de outro objeto para realizar alguma tarefa simplesmente envia uma mensagem a este último
  - O método (maneira de fazer) que o objeto requisitado usa para realizar a tarefa não é conhecido dos objetos requisitantes

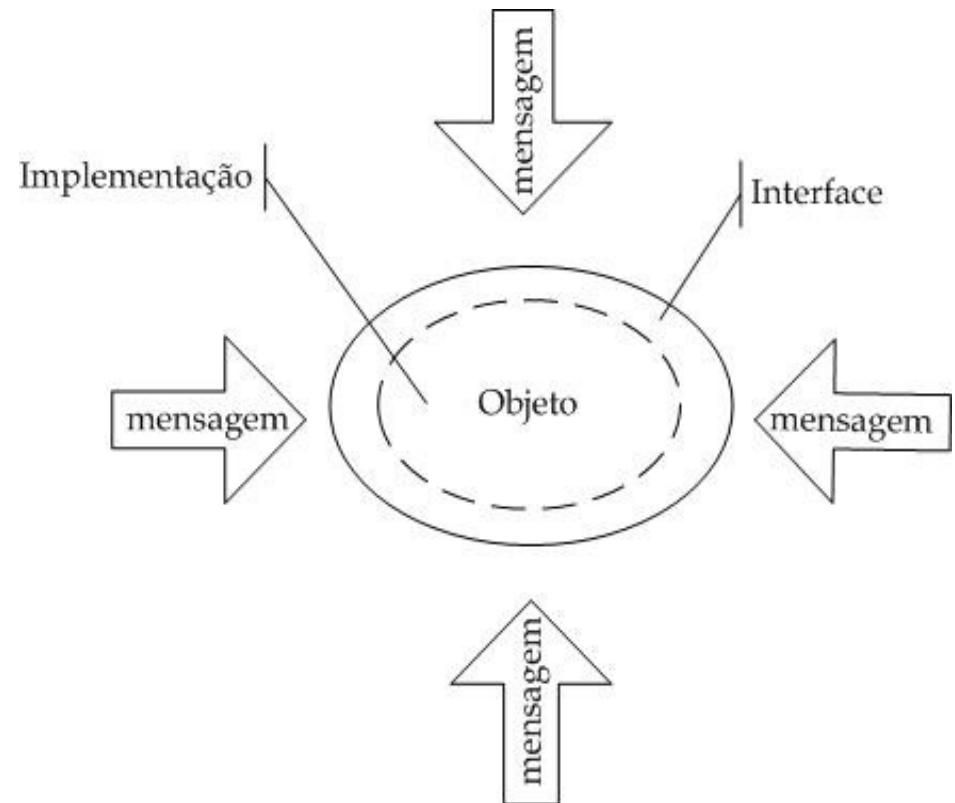
# Encapsulamento

- Na OO, diz-se que um objeto possui uma **interface**
  - A interface de um objeto é o que ele conhece e o que ele sabe fazer, sem descrever como o objeto conhece ou faz
  - A interface de um objeto define os serviços que ele pode realizar e conseqüentemente as mensagens que ele recebe



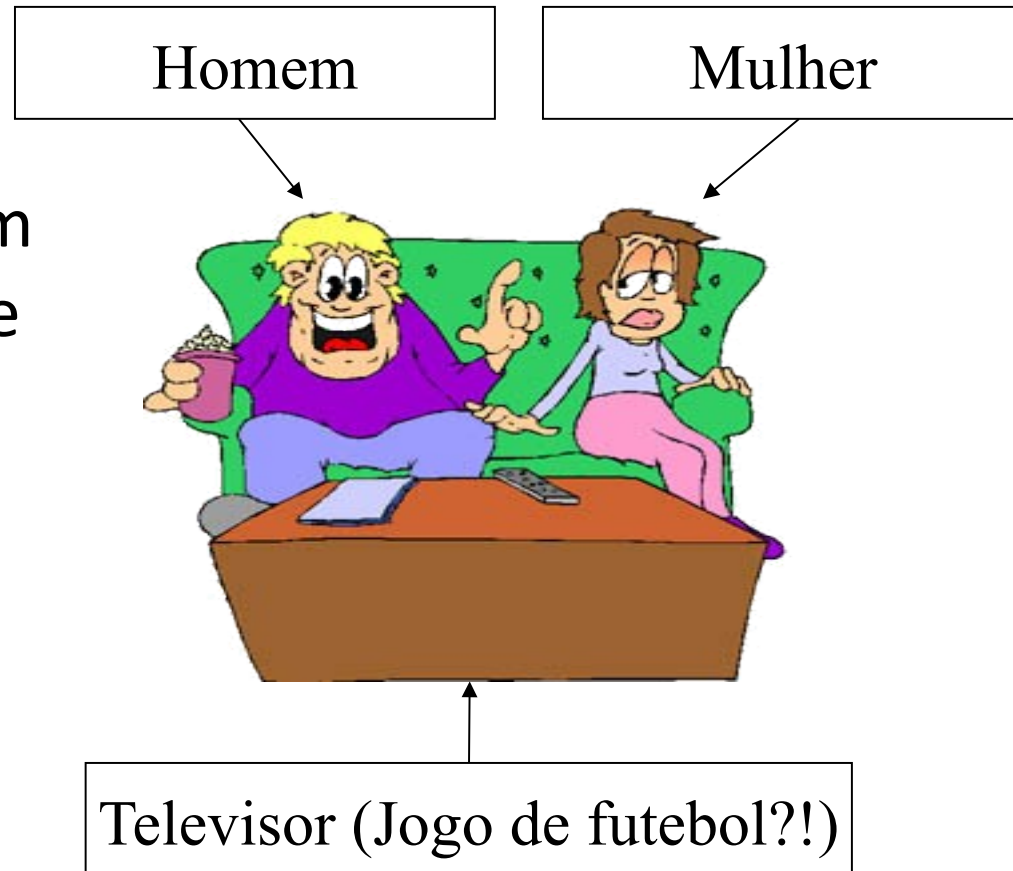
# Encapsulamento

- Uma interface pode ter várias formas de **implementação...**
- Mas, pelo Princípio do Encapsulamento, a implementação de um objeto requisitado não importa para um objeto requisitante



# Polimorfismo

- É a habilidade de objetos de classes diferentes responderem a mesma mensagem de diferentes maneiras



# Polimorfismo

- Em uma linguagem orientada a objetos, C++ ou Java:

```
for(i = 0; i < poligonos.tamanho(); i++)  
    poligonos[i].desenhar();
```

# Herança

- A herança pode ser vista como um nível de abstração acima da encontrada entre classes e objetos
- Na herança, classes semelhantes são agrupadas em hierarquias.
  - Cada nível de uma hierarquia pode ser visto como um nível de abstração.
  - Cada classe em um nível da hierarquia herda as características das classes nos níveis acima

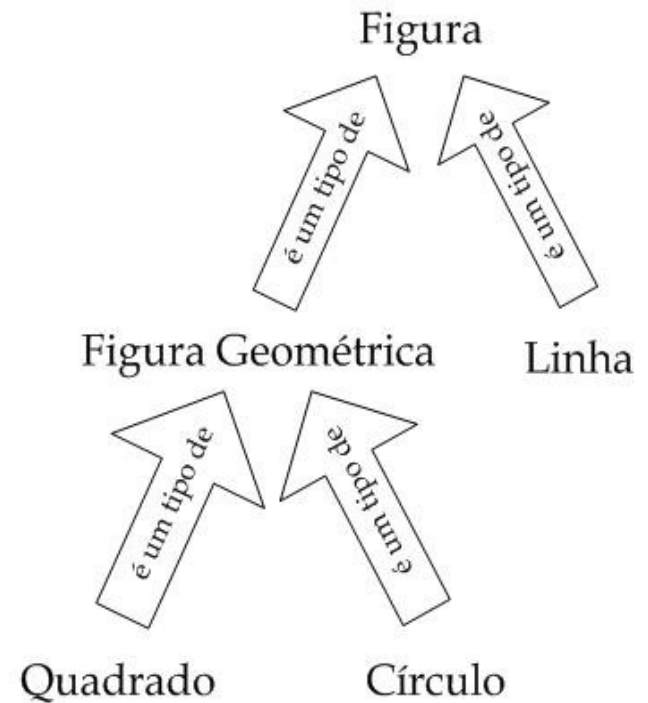
# Herança

- A herança facilita o compartilhamento de comportamento entre classes semelhantes
- As diferenças ou variações de uma classe em particular podem ser organizadas de forma mais clara

Maior  
abstração



Menor  
abstração



# A LINGUAGEM DE MODELAGEM UNIFICADA (UML)



# Histórico

- A UML é uma tentativa de padronizar a modelagem orientada a objetos de uma forma que qualquer sistema, seja qual for o tipo, possa ser modelado corretamente, com
  - Consistência
  - Facilidade de comunicação
  - Simplicidade de atualização
  - Facilidade de compreensão

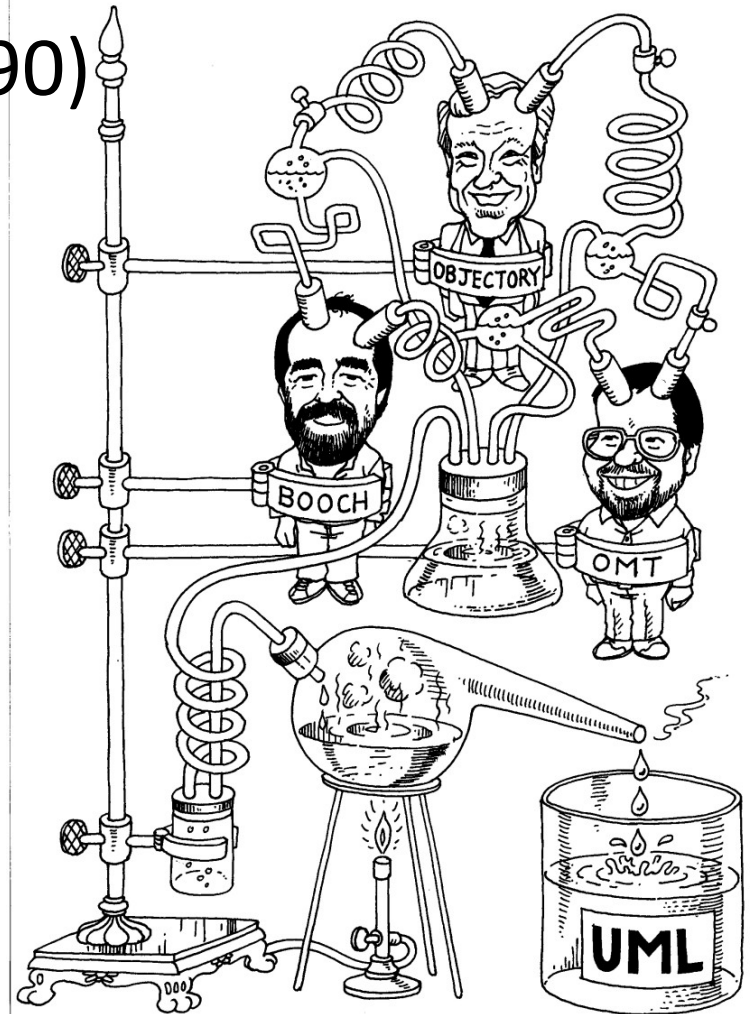
# Histórico

- Segundo a OMG, a UML é uma linguagem visual para especificação, construção e documentação de artefatos de software
- O propósito da modelagem é, principalmente, entender e não documentar o software
- UML deve ser usada dentro de um processo de desenvolvimento de software funcionando como uma notação de apoio à especificação e documentação de artefatos



# Histórico

- Principais notações (anos 90)
  - Booch
    - Autor: Grady Booch
  - OMT
    - Autor: James Rumbaugh
  - OOSE/Objectory
    - Autor: Ivar Jacobson



# Histórico

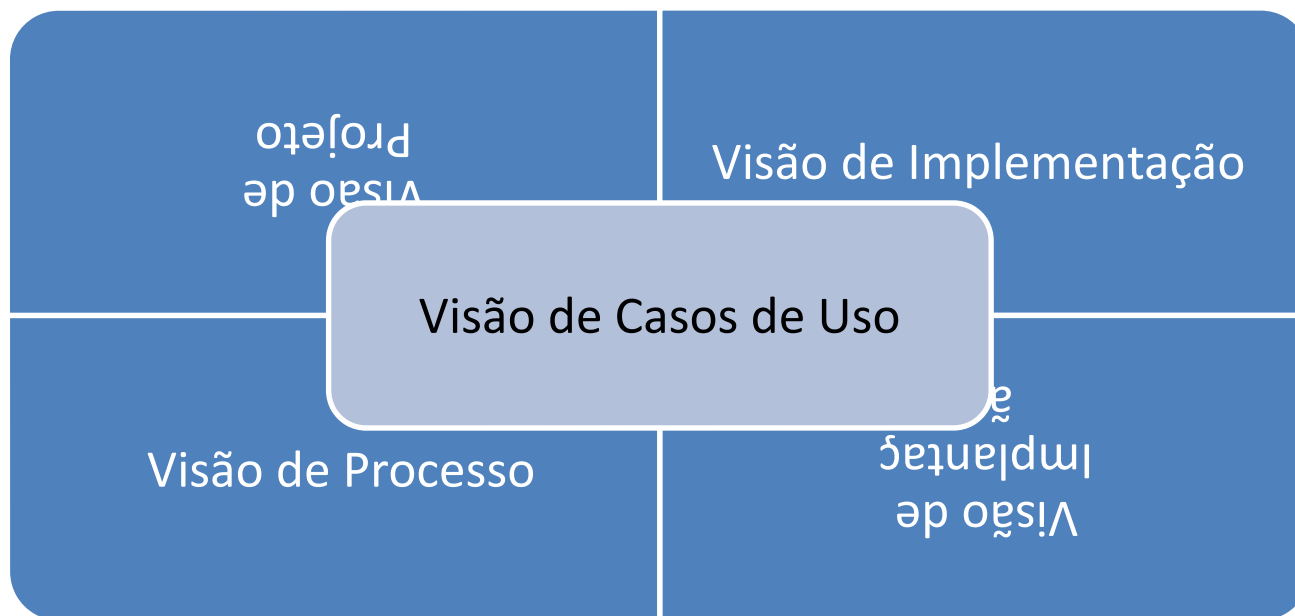
- Versões da UML
  - UML 0.8 - 1995 - OOPSLA'95
  - UML 0.9 - 1996 - Web
  - UML 1.0 - 1997 - Parceiros da UML
  - UML 1.1 - 1997 - OMG
  - UML 1.3 - 1999 - OMG
  - UML 1.4 - 2001 - OMG
  - UML 2.0 - 2004 - OMG

# Características da UML

- UML é...
  - Uma linguagem visual
  - Independente de linguagem de programação
  - Independente de processo de desenvolvimento
- UML **não** é...
  - Uma linguagem programação
  - Uma técnica de modelagem

# Visões de um Sistema

- Os autores da UML sugerem que um sistema pode ser descrito sobre a perspectiva de cinco visões



# Visões de um Sistema

- Visão de Casos de Uso
  - Descreve o sistema do ponto de vista externo como um conjunto de interações entre o sistema e agentes externos ao sistema
- Visão de Projeto
  - Enfatiza as características do sistema que dão suporte tanto estrutural quanto comportamental, às funcionalidades externamente visíveis

# Visões de um Sistema

- Visão de Implementação
  - Abrange o gerenciamento de versões do sistema, construídas pelo agrupamento de módulos (componentes) e subsistemas
- Visão de Implantação
  - Corresponde à distribuição física do sistema em seus subsistemas e à conexão entre essas partes

# Visões de um Sistema

- Visão de Processo
  - Enfatiza as características de concorrência (paralelismo), sincronização e desempenho do sistema

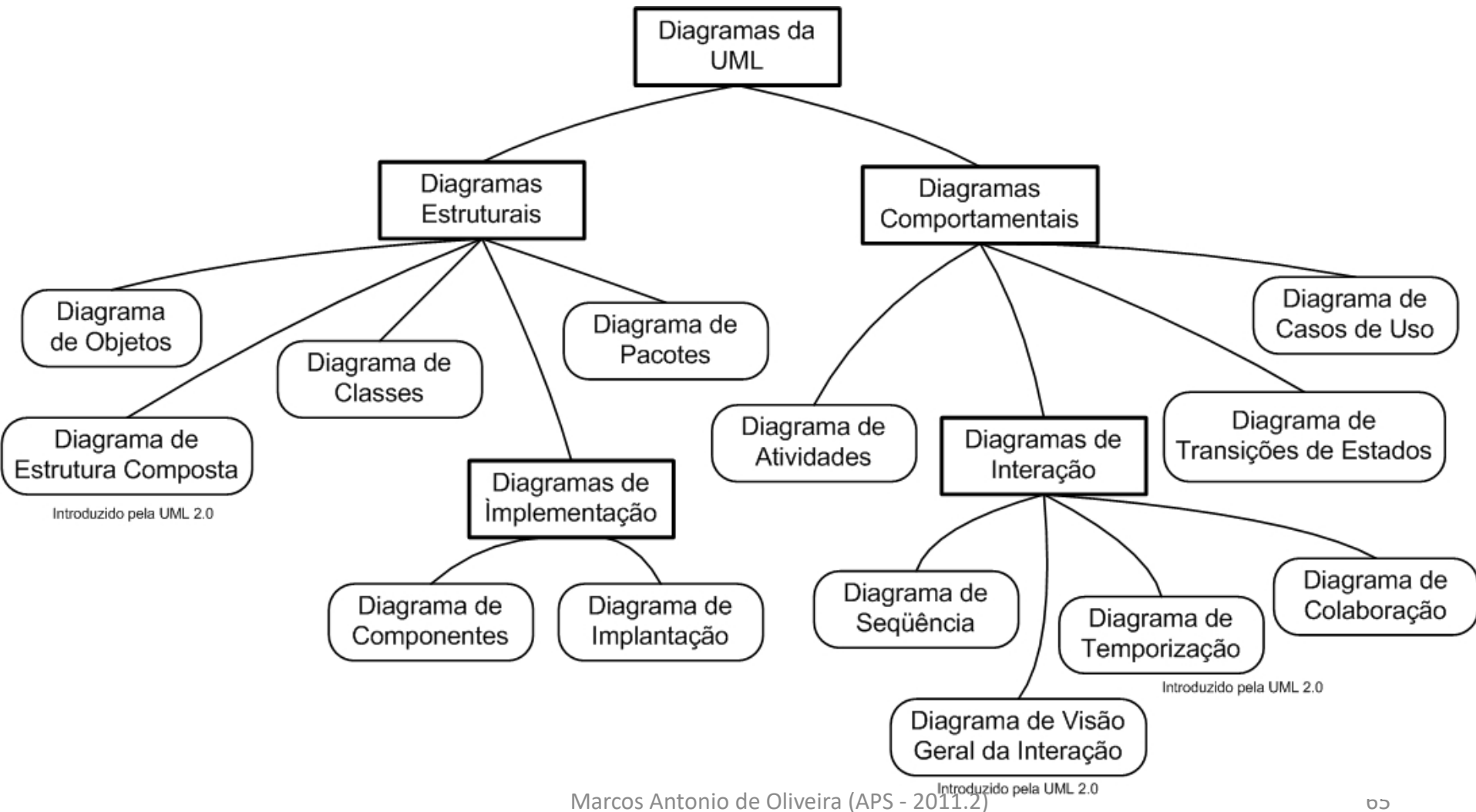
# Diagramas da UML

- Um processo de desenvolvimento que utilize a UML como linguagem de modelagem envolve a criação de diversos documentos
  - Estes documentos podem ser textuais ou gráficos
  - Estes documentos são denominados **artefatos de software**
  - São os artefatos que compõem as visões do sistema

Os artefatos gráficos produzidos durante o desenvolvimento de um sistema de software são definidos através da utilização dos **diagramas da UML**.



# Diagramas da UML



# Referências

- BEZERRA, E. Princípios de Análise e Projeto de Sistemas com UML. 2ª ed. Rio de Janeiro: Elsevier, 2007.
- FOWLER, M. 3. UML Essencial. 3. ed. Porto Alegre: Bookman, 2007.