Princípios de Análise e Projeto de Sistemas com UML

2ª edição

Eduardo Bezerra

Editora Campus/Elsevier

Capítulo 1
Visão Geral

“Coisas simples devem ser simples e coisas complexas devem ser possíveis.”. -Alan Kay

Sistemas de Informações

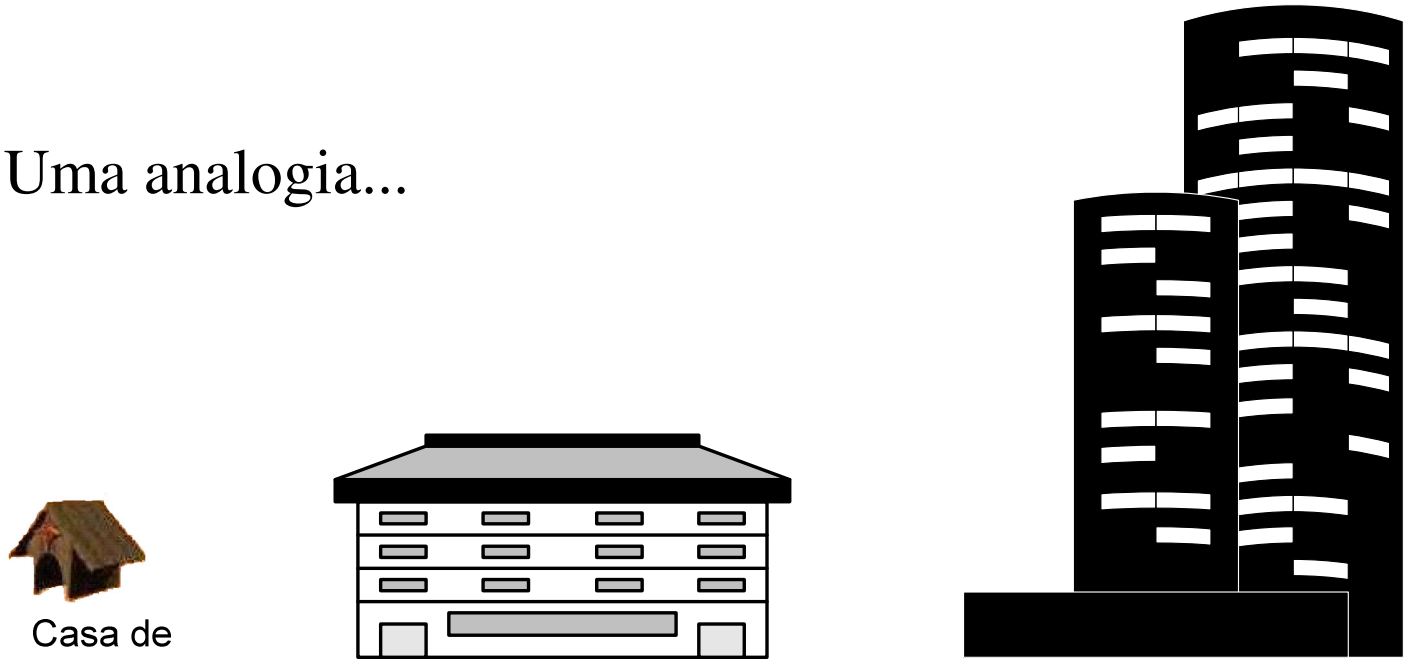
- A necessidade é a mãe das invenções
  - Em conseqüência do crescimento da importância da informação, surgiu a necessidade de gerenciar informações de uma forma adequada e eficiente e, desta necessidade, surgiram os denominados ***sistemas de informações***.
- Um SI é uma combinação de pessoas, dados, 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 à organização*.

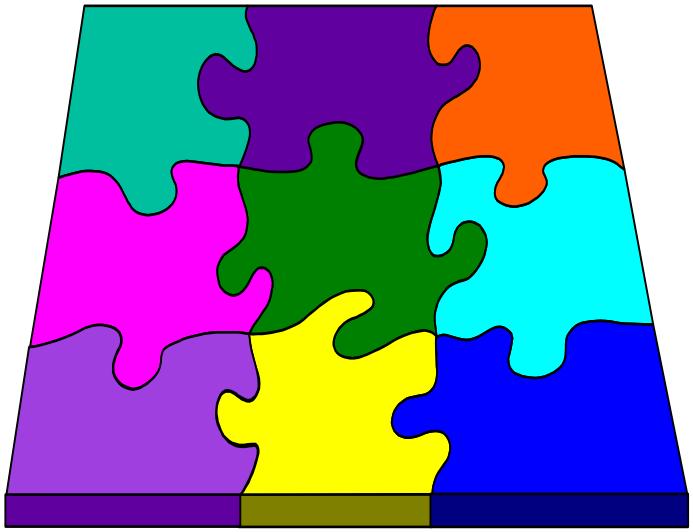
Sistemas de Software

- Um dos componentes de um 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...

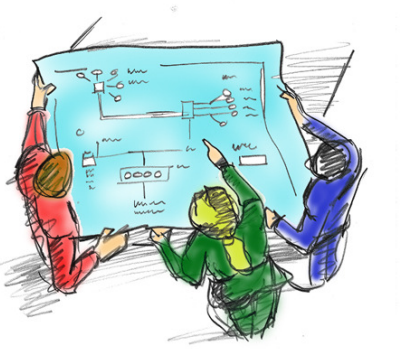


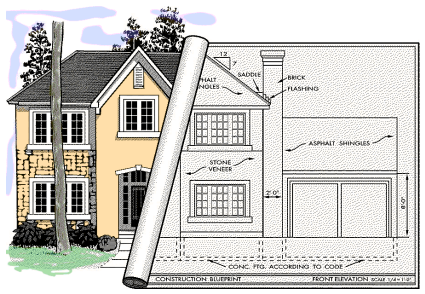


1.1 Modelagem de sistemas de software

Modelos de Software

- 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.
- Um modelo pode ser visto como uma representação idealizada de um sistema que se planeja construir.
- Maquetes de edifícios e de aviões e plantas de circuitos eletrônicos são apenas alguns exemplos de modelos.





Razões para construção de modelos

- A princípio, podemos ver a construção de modelos como uma atividade que atrasa o desenvolvimento do software propriamente dito.
- Mas essa atividade propicia...
  - O **gerenciamento da complexidade** inerente ao desenvolvimento de software.
  - A **comunicação** entre as pessoas envolvidas.
  - A **redução dos custos** no desenvolvimento.
  - A **predição do comportamento** futuro do sistema.
- Entretanto, note o fator **complexidade** como condicionante dessas vantagens.

Diagramas e Documentação

- No contexto de desenvolvimento de software, correspondem a desenhos gráficos que seguem algum padrão lógico.
- Podemos também dizer que um diagrama é uma apresentação de uma coleção de elementos gráficos que possuem um significado predefinido.
- Diagramas normalmente são construídos de acordo com regras de notação bem definidas.
  - Ou seja, cada forma gráfica utilizada em um diagrama de modelagem tem um significado específico.

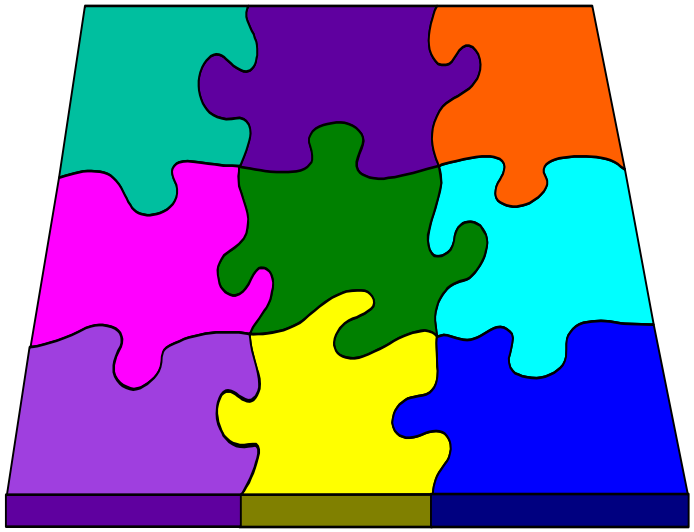
Diagramas e Documentação

- Diagramas permitem a construção de uma representação concisa de um sistema a ser construído.
  - "**uma figura vale por mil palavras**"

- No entanto, 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



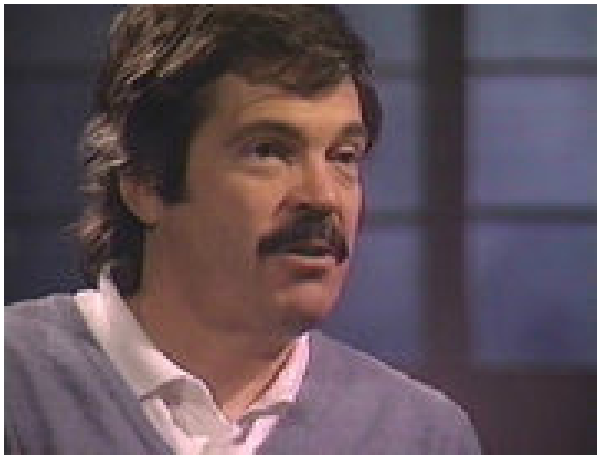
1.2 O paradigma da orientação a objetos

Paradigma?

- *Um paradigma é uma forma de abordar um problema*.
- No contexto da modelagem de um sistema de software, um paradigma tem a ver com a forma pela qual esse sistema é entendido e construído.
- A primeira abordagem usada para modelagem de sistemas de software foi o ***paradigma estruturado***.
  - Uso da técnica de *decomposição funcional*
  - "divida sucessivamente um problema complexto em subproblemas"
- Hoje em dia, praticamente suplantou o paradigma anterior, o ***paradigma da orientação a objetos***...

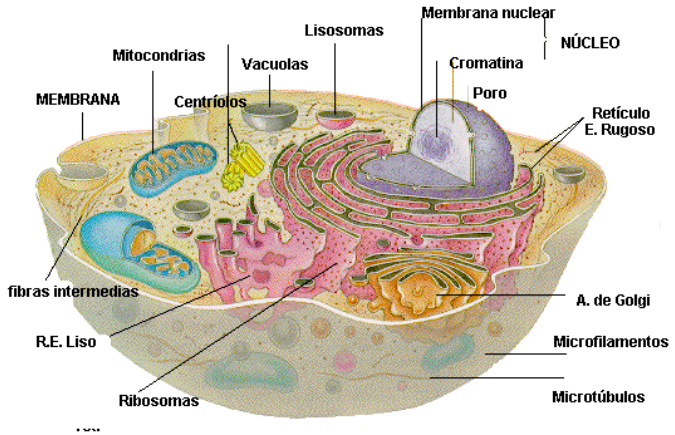
O Paradigma da Orientação a Objetos

- O paradigma da OO surgiu no fim dos anos 60.
- Alan Kay, um dos pais desse paradigma, 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.
- De uma forma mais geral, Kay pensou em como construir um sistema de software a partir de agentes autônomos que interagem entre si.

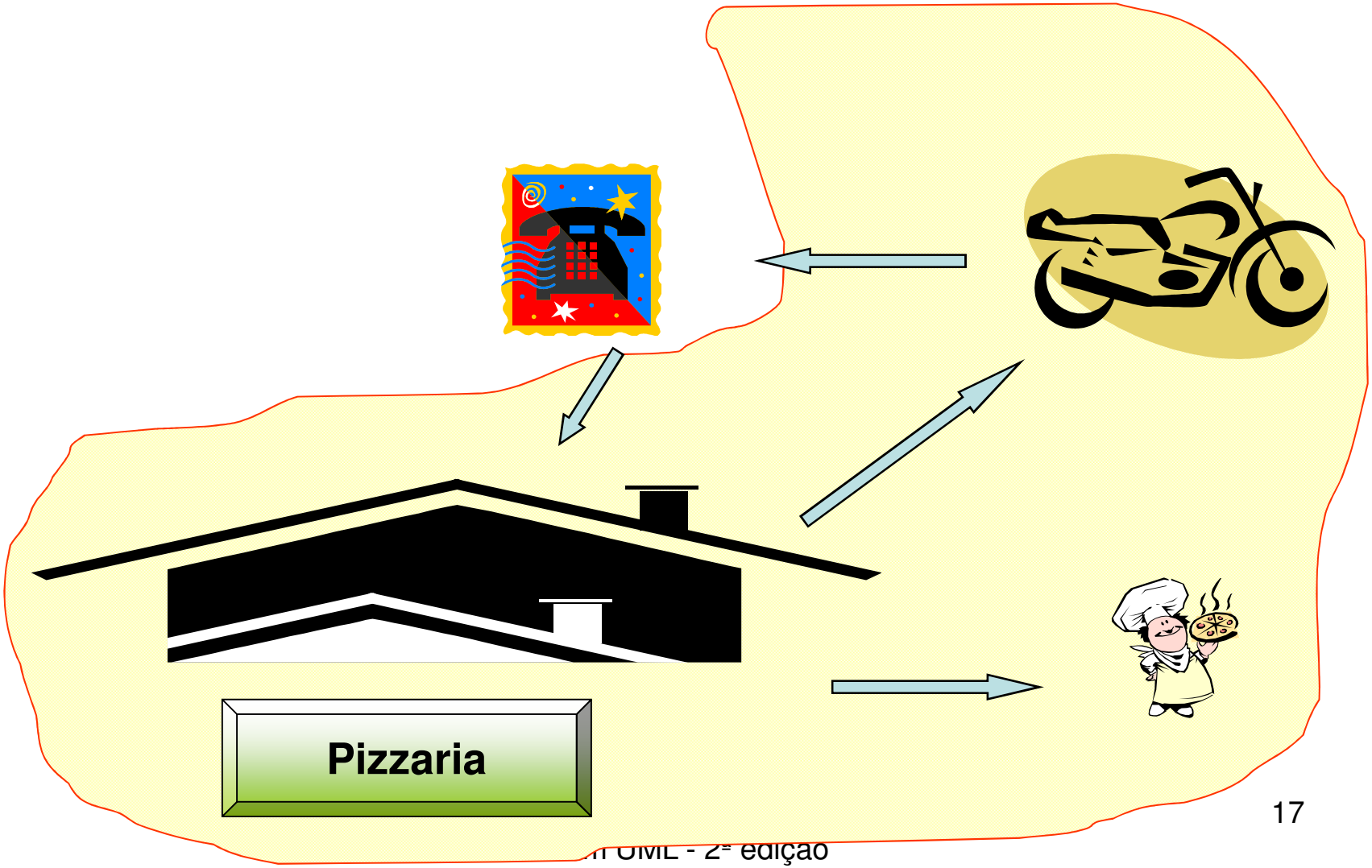


Fundamentos da Orientação a Objetos

- Através de sua analogia biológica, Alan Kay definiu os fundamentos da orientação a objetos.

1. Qualquer coisa é um objeto.
2. Objetos realizam tarefas através 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.

SSOO: uma analogia



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.

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 e Princípios da OO

- Conceitos
  - Classe
  - Objeto
  - Mensagem
- Princípios
  - Encapsulamento
  - Polimorfismo
  - Generalização (Herança)
  - Composição

Classes, objetos e mensagens

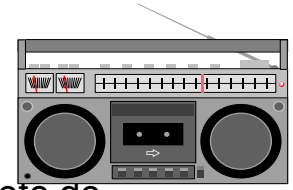
- O mundo real é formado de coisas.
- Na terminologia de orientação a objetos, estas coisas do mundo real são denominadas *objetos*.
- Seres humanos costumam agrupar os objetos para entendê-los.
- A descrição de um grupo de objetos é denominada ***classe de objetos***, ou simplesmente de ***classe***.

O que é uma classe?

- Uma classe é um molde para objetos. Diz-se que um objeto é uma instância de uma classe.
- Uma classe é uma ***abstração*** das características ***relevantes*** de um grupo de coisas do mundo real.
  - Na maioria das vezes, um grupo de objetos do mundo real é muito complexo para que *todas* as suas características e comportamento sejam representados em uma classe.

Representante

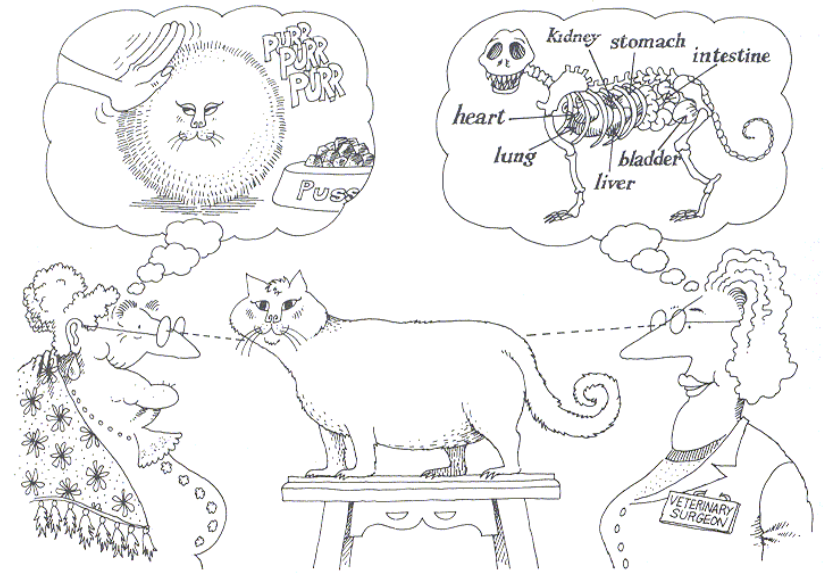
Cliente



Produto

Abstração

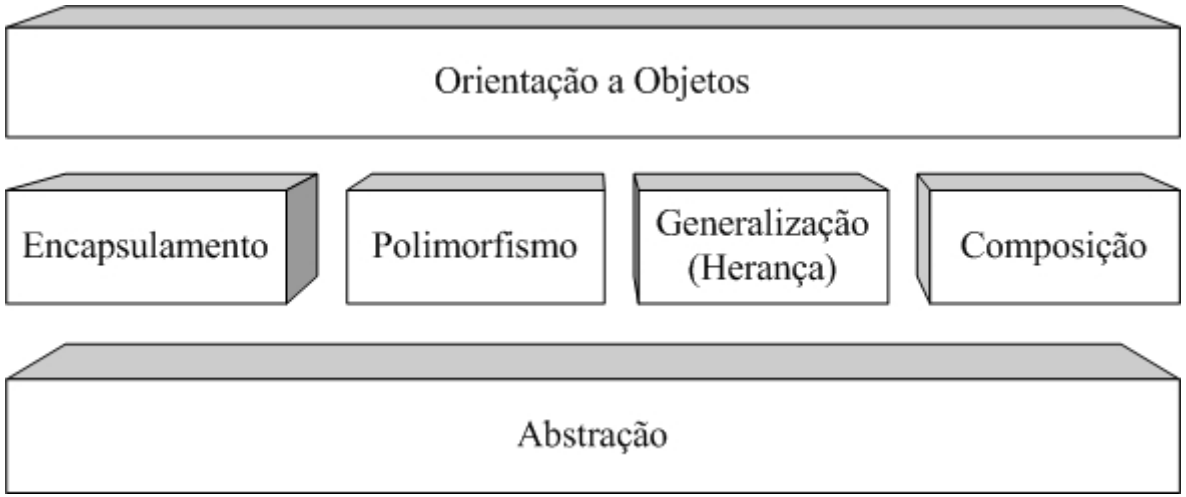
- Uma abstração é qualquer modelo que inclui os aspectos relevantes de alguma coisa, ao mesmo tempo em que ignora os menos importantes. *Abstração depende do observador.*



Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

Abstração na orientação a objetos

- A orientação a objetos faz uso intenso de abstrações.
  - Os princípios da OO podem ser vistos como aplicações da abstração.
- **Princípios da OO**: encapsulamento, polimorfismo, herança e composição.



Objetos como abstrações

- Uma abstração é uma representação das características e do comportamento relevantes de um conceito do mundo real para um determinado problema.
- Dependendo do contexto, um mesmo conceito do mundo real pode ser representado por diferentes abstrações.
  - 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)

Classe X Objeto

- Objetos são abstrações de entidades que existem no mundo real.
- Classes são definições estáticas, que possibilitam o entendimento de um grupo de objetos.

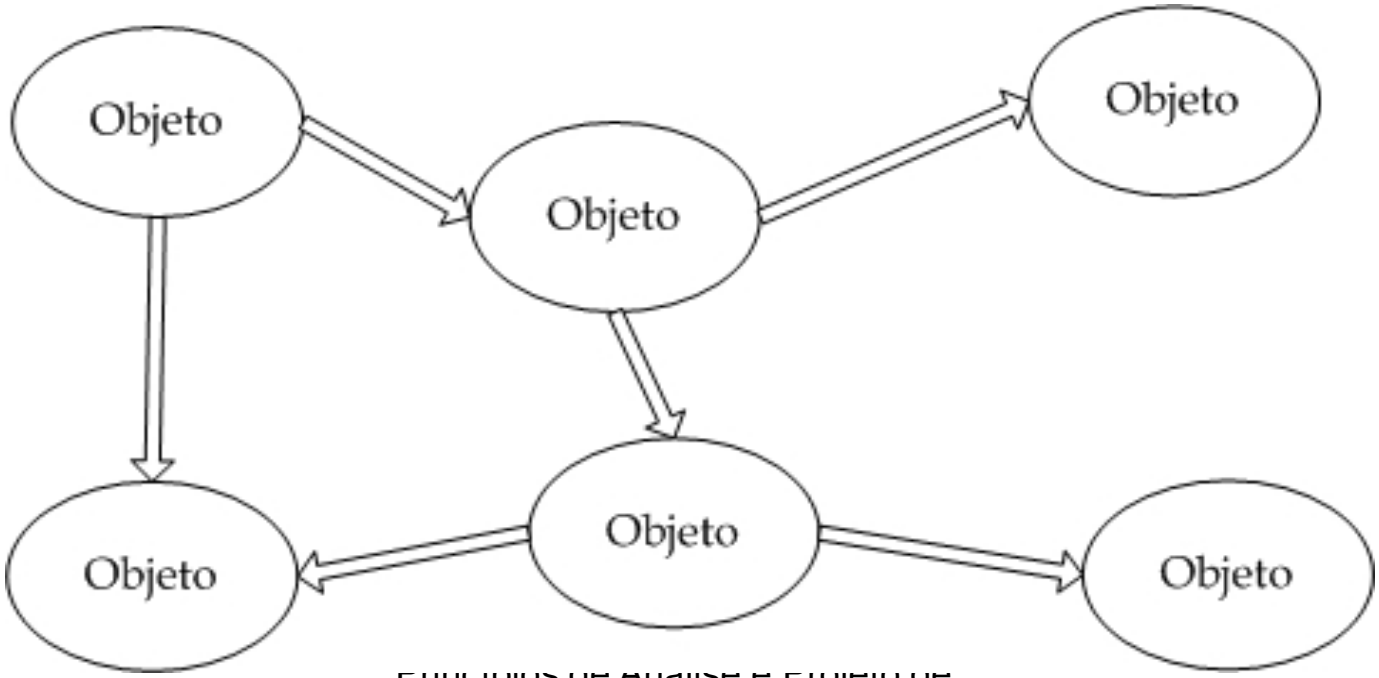
- **CUIDADO**: estes dois termos muitas vezes são usados indistintamente em textos sobre orientação a objetos.

Mensagens

- Para que um objeto realize alguma tarefa, deve haver um estímulo enviado a este objeto.
- 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.
- 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.

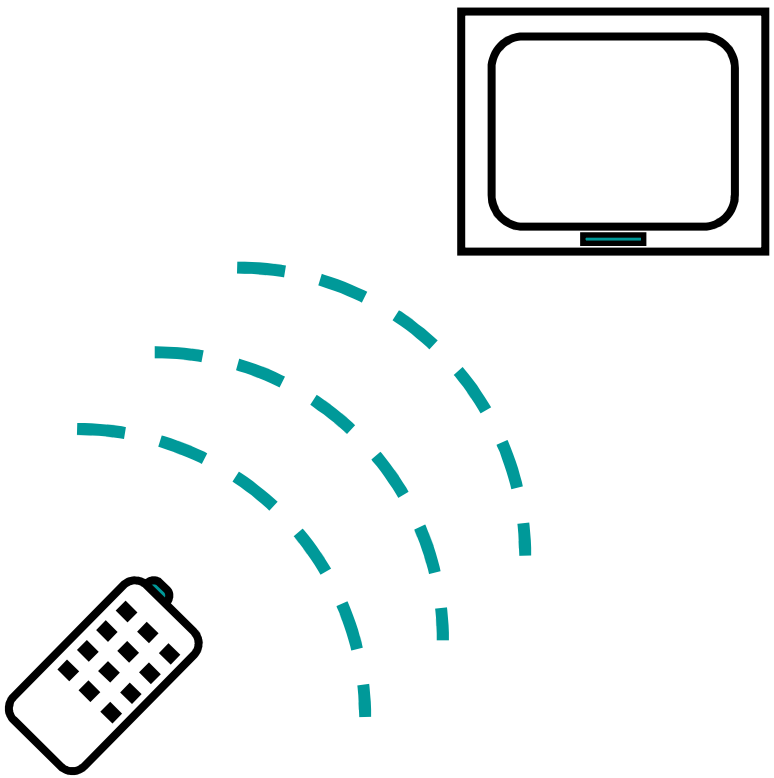
Mensagens

- *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.



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.
- De acordo com o encapsulamento, objetos devem "esconder" a sua complexidade...
- Esse princípio aumenta qualidade do SSOO, em termos de:
  - Legibilidade
  - Clareza
  - Reuso

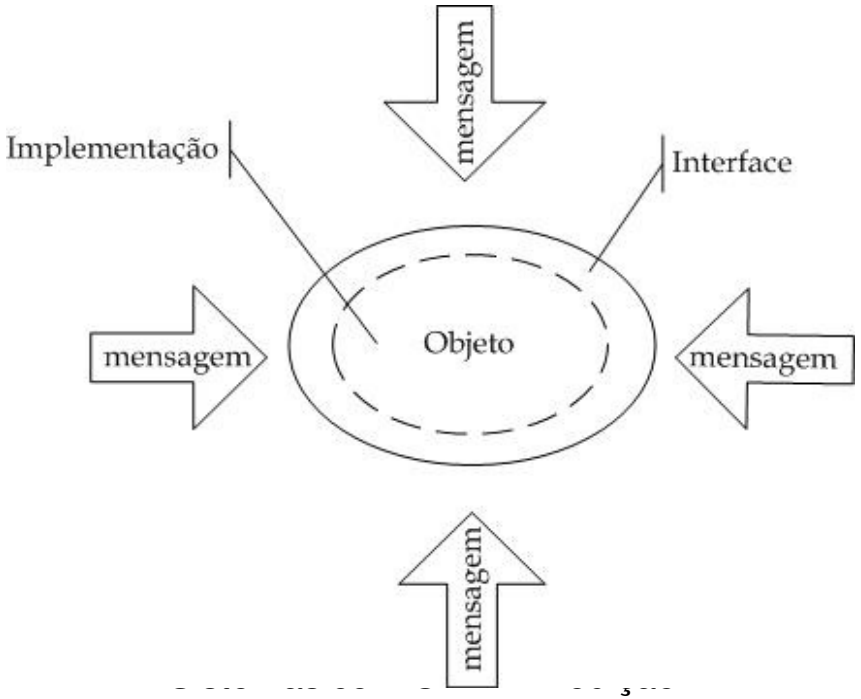


Encapsulamento

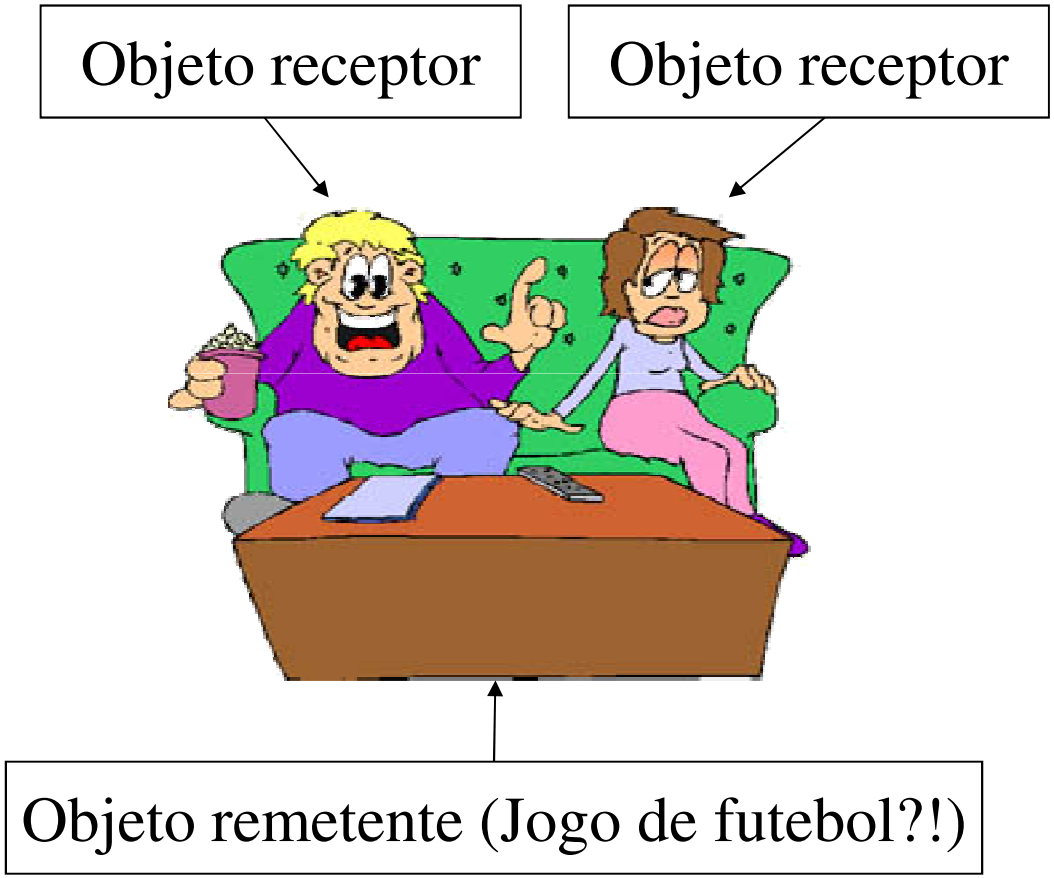
- O *encapsulamento* é uma forma de restringir o acesso ao comportamento interno de um objeto.
  - Um objeto que precise da colaboração de outro para realizar alguma tarefa simplesmente envia uma mensagem a este último.
  - O método (maneira de fazer) que o objeto requisitado usa para realizar a tarefa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



Polimorfismo



Polimorfismo

- É a habilidade de objetos de classes diferentes responderem a mesma mensagem de diferentes maneiras.
- Em uma linguagem orientada a objetos:

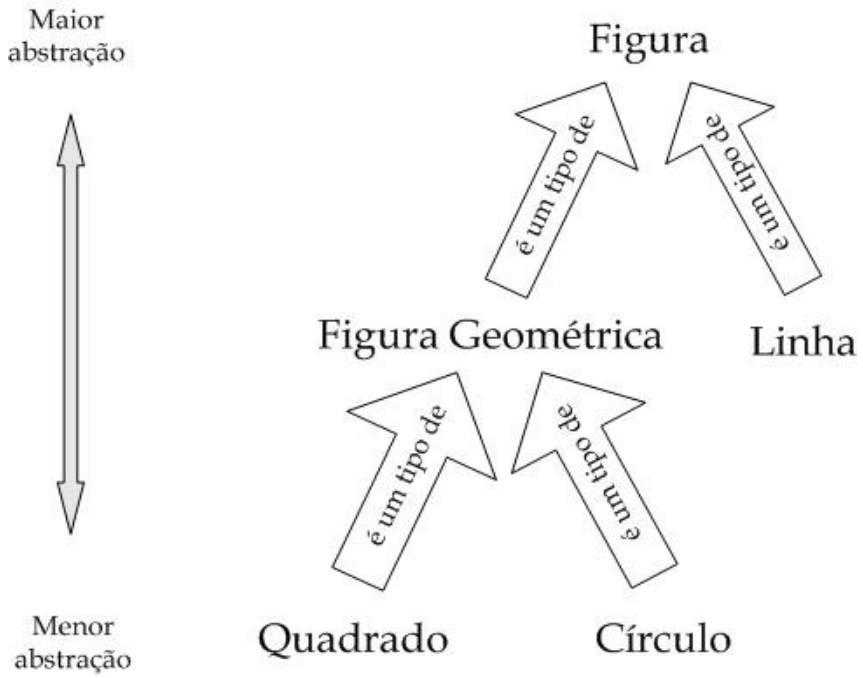
```
for(i = 0; i < poligonos.tamanho(); i++)
      poligonos[i].desenhar();
```

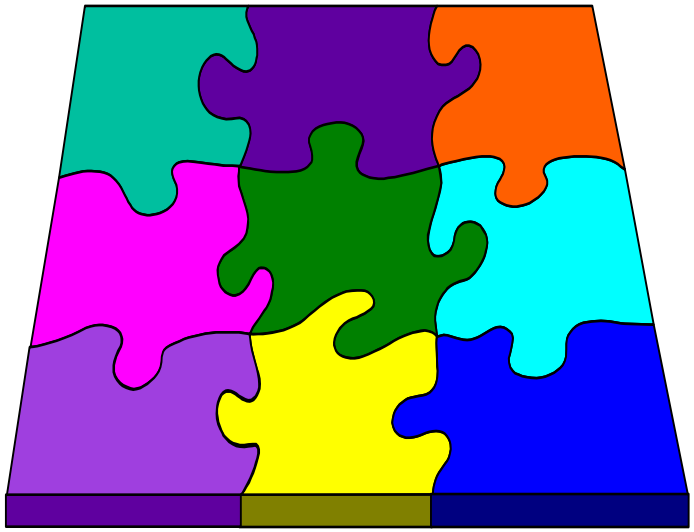
Generalização (Herança)

- 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.

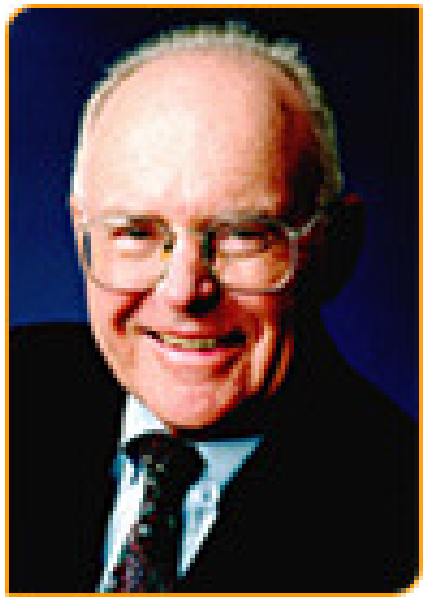




1.3 Evolução histórica da modelagem de sistemas
1.4 A Linguagem de modelagem unificada

Evolução do Hardware

- A chamada ***Lei de Moore*** é bastante conhecida da comunidade de computação.
- Essa lei foi declarada em 1965 pelo engenheiro Gordon Moore, co-fundador da Intel.
- ***Lei de Moore:*** "*A densidade de um transistor dobra em um período entre 18 e 24 meses*".



Evolução do Software

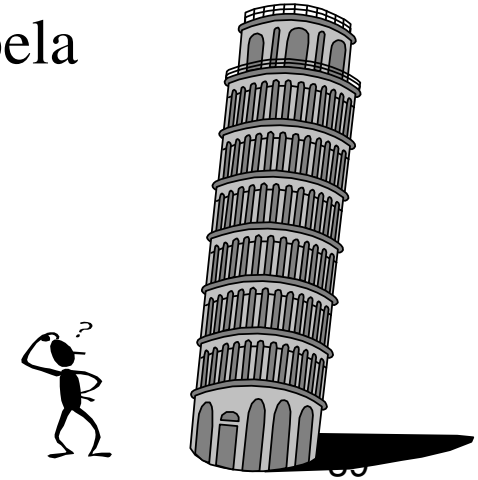
- O rápido crescimento da capacidade computacional das máquinas resultou na demanda por sistemas de software cada vez mais complexos.
- O surgimento de sistemas de software mais complexos resultou na necessidade de reavaliação da forma de se desenvolver sistemas.
- Conseqüentemente as técnicas utilizadas 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.

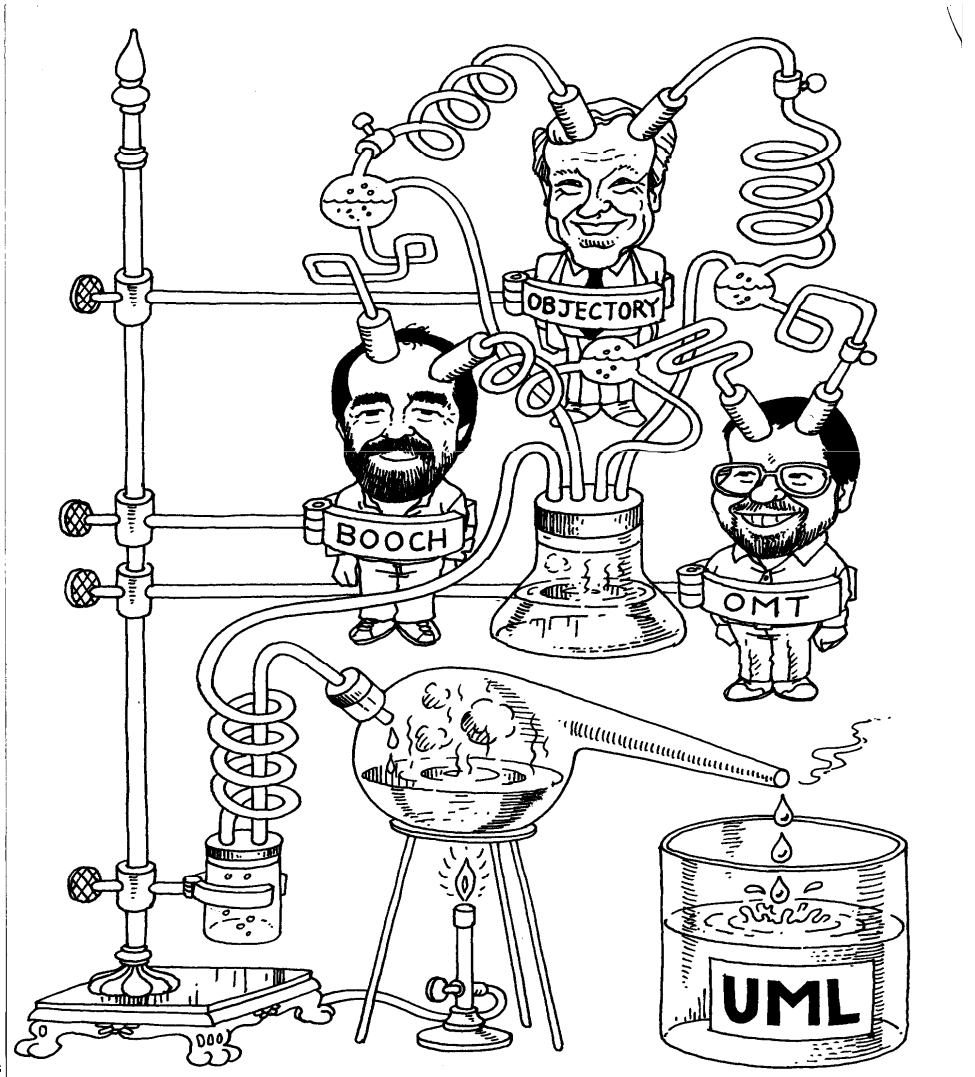
Necessidade de um Padrão

- Percebeu-se a necessidade de um padrão para a modelagem de sistemas, que fosse aceito e utilizado amplamente.
- Alguns esforços nesse sentido de padronização, o principal liderado pelo “três amigos”.
- Surge a UML (Unified Modeling Language) em 1996 como a melhor candidata para ser linguagem “unificadora”.
- Em 1997, a UML é aprovada como padrão pelo OMG.
- Desde então, a UML tem tido grande aceitação pela comunidade de desenvolvedores de sistemas.
- É uma linguagem ainda em desenvolvimento.
- Atualmente na versão 2.0.



UML (Linguagem de Modelagem Unificada)

- "A UML é a linguagem padrão para visualizar, especificar, construir e documentar os artefatos de software de um sistema."
- Unificação de diversas notações anteriores.
- Mentores: Booch, Rumbaugh e Jacobson
  - "Três Amigos"
  - IBM Rational (www.rational.com)



UML (Linguagem de Modelagem Unificada)

- UML é...
  - uma linguagem visual.
  - independente de linguagem de programação.
  - independente de processo de desenvolvimento.
- UML **não** é...
  - uma linguagem programação (mas possui versões!).
  - uma técnica de modelagem.
- Um processo de desenvolvimento que utilize a UML como linguagem de modelagem envolve a criação de diversos documentos.
  - Estes documentos, denominados **artefatos de software**, podem ser textuais ou gráficos.
- Os artefatos gráficos produzidos de um sistema OO são definidos através dos **diagramas da UML**.

Diagramas da UML

- Um diagrama na UML é uma apresentação de uma coleção de ***elementos gráficos*** que possuem um significado predefinido.
  - No contexto de desenvolvimento de software, correspondem a desenhos gráficos que seguem algum padrão lógico.
- Um processo de desenvolvimento que utilize a UML como linguagem de modelagem envolve a criação de diversos documentos.
  - Estes documentos, denominados **artefatos de software**, podem ser textuais ou gráficos.
- Os artefatos gráficos produzidos no desenvolvimento de um SSOO são definidos através dos **diagramas da UML**.

Diagramas da UML 2.0