

# Orientação a Objetos

Maurício de Castro

SOLIS/UNIVATES

[mcastro@solis.coop.br](mailto:mcastro@solis.coop.br)

Coordenador de Desenvolvimento de Sistemas em

Software Livre da SOLIS/UNIVATES

# Sumário

- Cenários de Desenvolvimento
- Histórico da OO
- Conceitos de OO
- Análise e Desenvolvimento
- Bancos de Dados OO

# **Cenários de Desenvolvimento**

# Cenário Estrutural

- Foco principal nas funções depois nos dados, informações desagrupadas
- As idéias e necessidades dos usuários normalmente não ficam claras
  - Quais são suas necessidades para o sistema?
  - Preciso de um sistema que controle todas as vendas dos meus produtos

# Cenário Estrutural

- **Validação com o Usuário**
  - É muito difícil para o usuário leigo entender um modelo ER
  - SAGU tem 99 tabelas e centenas de relacionamentos
  - Problemas se descobrem durante a implementação
- **MIOLO – FrameWork OO para Desenvolvimento de Software**

# Cenário Estrutural

- Excesso de documentação ou nenhuma
- Cronograma apertado, prazos estourados, dificuldade de reaproveitamento de código
- Cliente insatisfeito
- Horas infindáveis de manutenção corretiva
- Custo elevado de projeto

# Cenário OO

- Foco principal nos objetos do mundo real, com suas funções e dados agrupados.
- Deve-se fazer o levantamento de requisitos já pensando nos objetos do mundo real.
- Facilita o entendimento por parte do programador das necessidades do usuário
  - O que vamos controlar?
  - Nossos carros

# Cenário OO

- Diminuição do tempo e custo de desenvolvimento
- Atendimento da demanda gerada pela evolução tecnológica
- Reutilização de código, facilidade de manutenção

# **Histórico da Orientação a Objetos**

# Histórico

- 1967: Simula - introduz os primeiros conceitos de OO
- 1972: Smalltalk
- 1980: C++ linguagem híbrida, derivada da linguagem C
- 1983: Ada criada para uso militar nos EUA
- 1984: Eiffel primeiras características formais de OO
- 1986: Object pascal
- 1995: JAVA - Linguagem puramente orientada a objetos
- 1995: Várias linguagens agregando conceitos de OO

# Simula

- A primeira linguagem a incorporar facilidades para definir classes de objetos genéricos na forma de uma hierarquia de classes e sub-classes
- Foi idealizada em 1966, na Noruega, como uma extensão da linguagem ALGOL 60.
- Uma classe em Simula é um módulo englobando a definição da estrutura e do comportamento comuns a todas as suas instâncias (objetos).

# Smalltalk

- Smalltalk foi desenvolvida no Centro de Pesquisas da Xerox durante a década de 70
- Incorporou idéias de Simula
- Criou o princípio de objetos ativos, prontos a "reagir" a "mensagens" que ativam "comportamentos" específicos do objeto

# C++

- Questões no projeto de C++
  - Ser melhor do que C
  - Suportar abstração de dados
  - Suportar programação orientada a objetos
- C++ foi projetada para dar suporte a abstração de dados e programação orientada a objetos
- C++ não impõe um paradigma

# Ada

- Ada é uma linguagem de programação criada através de um concurso realizado pelo U.S. Department of Defense (DoD)
- O principal projetista da equipe foi o francês Jean Ichbiah.
- Esse concurso foi feito para por ordem na situação, o DoD em 1974 usava cerca de 450 linguagens ou dialetos de programação.
- A linguagem foi primeiramente padronizada em 1983 pelo ANSI e em 1985 a Organização Internacional de Padronização (ISO).

# Eiffel

- Eiffel é uma Linguagem de Programação avançada, puramente orientada a objeto que enfatiza o projeto e construção de software reusável e de alta qualidade.
- Eiffel foi criada por Bertrand Meyer que tinha uma extensa experiência com programação orientada a objeto, particularmente com SIMULA.

# Object Pascal

- O Object Pascal é uma linguagem orientada a objetos, isto é, todas as informações são tratadas como objetos
- Todos estes objetos pertencem a uma classe, que são categorias de objetos
- Delphi / Kylix / Lazarus são exemplos de ferramentas que utilizam esta linguagem.

# Java

- O Java é ao mesmo tempo um ambiente e uma linguagem de programação desenvolvida pela Sun Microsystems, Inc.
- Trata-se de mais um representante da nova geração de linguagens orientadas a objetos e foi projetado para resolver os problemas da área de programação cliente/servidor.
- Os aplicativos em Java são compilados em um código de bytes independente de arquitetura.

# Java

- Esse código de bytes pode então ser executado em qualquer plataforma que suporte um interpretador Java.
- O Java requer somente uma fonte e um binário e, mesmo assim, é capaz de funcionar em diversas plataformas, o que faz dele um sonho de todos os que realizam manutenção em programas.

# Evolução da OO

- Não se configura como uma mudança de paradigma abrupta. Evoluiu de idéias já manifestadas há muito tempo.
  - Larry Constantine - (Década de 1960) Foi quem primeiro lançou a idéia de que softwares poderiam ser projetados antes que fossem programados
  - O. J. Dahl e K. Nygaard - (1966) - Foi quem primeiro lançou a idéia de Classes introduzida na linguagem Simula
  - Alan Klay, Adele Goldberg e outros - (1970) Iniciaram o conceito de Mensagem e Herança, usados na linguagem SmallTalk.

# Evolução da OO

- Programação orientada a objetos é uma evolução da programação estruturada;
- Na programação estruturada temos funções (procedures ou rotinas) e dados (normalmente globais) que podem ser acessados por qualquer função;

# Evolução da OO

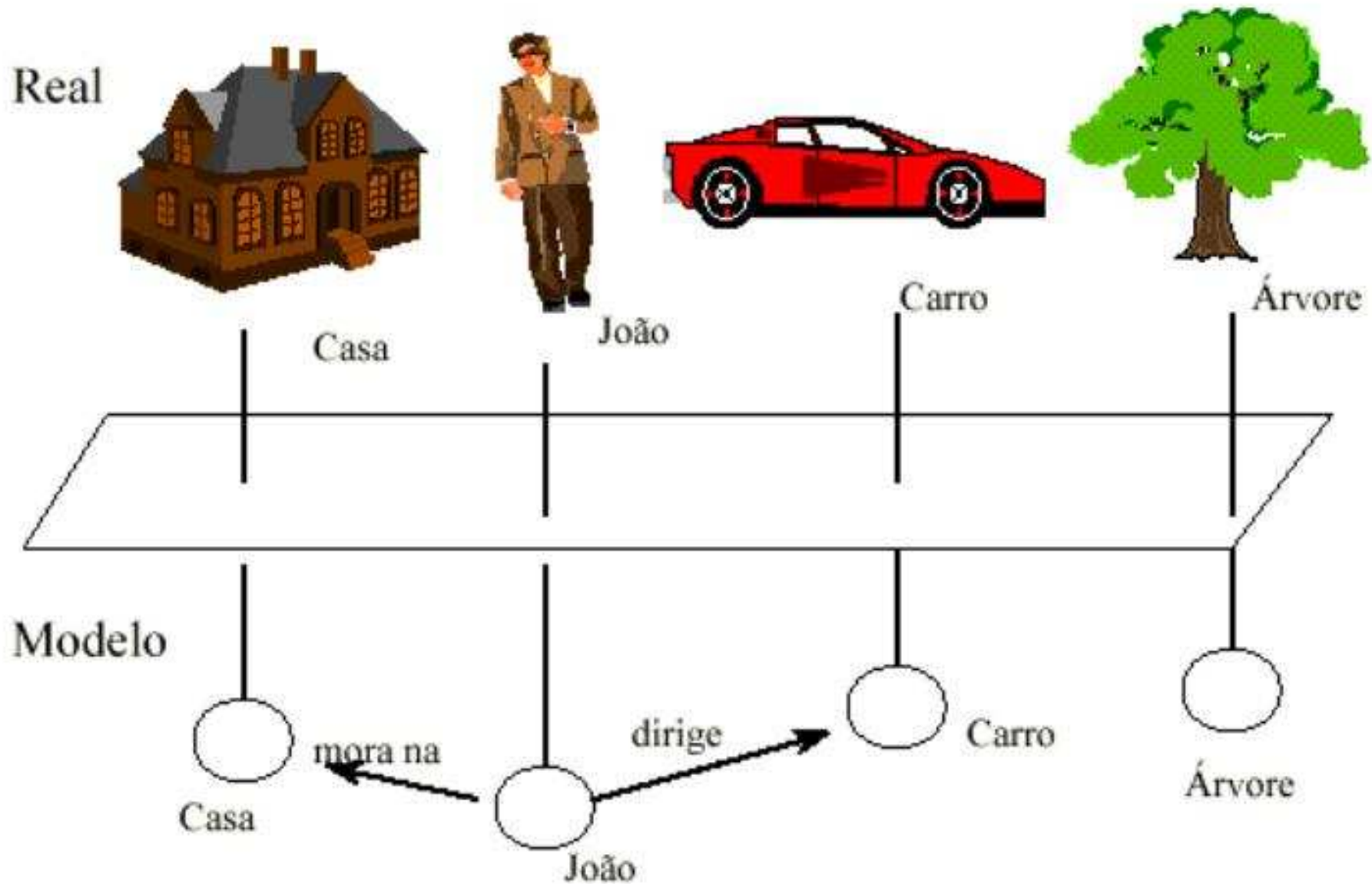
- Na programação orientada a objetos, temos funções agregadas aos dados em uma unidade chamada objeto, ou seja, os dados não estão separados das funções, mas sim unidos as mesmas;
- A tendência para os próximos anos é que a maioria das linguagens de programação sejam baseadas em objeto

# Conceitos de Orientação a Objetos

# Bases da Orientação a Objetos

- Na compreensão do mundo, os seres humanos utilizam-se de três métodos de organização dos pensamentos:
  - Diferenciação
  - Distinção entre todo e parte
  - Classificação
  - Vê a orientação a objetos, como técnica para modelagem de sistemas, utiliza estes métodos para diminuir a diferença semântica entre a realidade e o modelo

# Exemplo



# Conceitos

- Objetos e Instâncias
- Classes
- Atributos
- Métodos
- Visibilidade de atributos e operações
- Mensagens

# Objeto

- Um objeto denota uma entidade, seja ela de natureza física, conceitual ou de software.
  - Entidades físicas: um carro, uma pessoa, uma casa
  - Entidade conceitual: um organograma de uma empresa
  - Entidade de software: um botão em uma GUI

# Objeto

- É uma entidade capaz de reter um estado (informação/atributos/propriedades) e que oferece uma série de operações (comportamentos/métodos) ou para examinar ou para afetar este estado
- Um objeto é um conceito, uma abstração, algo com limites e significados nítidos em relação ao domínio de uma aplicação
- Objetos facilitam a compreensão do mundo real e oferecem uma base real para implementação em computador

# Objeto

- Um objeto é algo que tem:
  - Estado
  - Comportamento
  - Identidade

# Estado de um Objeto

- O estado de um objeto representa uma das possíveis condições em que um objeto pode existir
- O estado é representado pelos valores das propriedades de um objeto em um determinado instante
- O estado do objeto usualmente muda ao longo do tempo:
- Exemplo (*Aluno Maurício*):
  - Nome: Maurício
  - Matrícula: 105018
  - Semestre de Ingresso: 2000A

# Comportamento de um Objeto

- O comportamento determina como um um objeto pode responder a interações mediante à ativação de operações decorrentes de mensagens recebidas de outros objetos
- O Comportamento é determinado pelo conjunto de operações que o objeto pode realizar.
- Controle Academico
  - Maurício
- Solicita matrícula
  - Retorna: 105018

# Identidade de um Objeto

- Cada objeto tem um único identificador, mesmo que seu estado seja idêntico ao de outro objeto
  - Maurício (objeto)
  - Controle Acadêmico (Sistema que está sendo construído)
  - Semestre (estado)
  - Matrícula 105018 (Propriedade de um aluno)
  - Lista de Semestres Cursados (candidato a objeto)
  - Semestre corrente (o mesmo que semestre)

# Classe

- Uma classe é a descrição de um grupo de objetos com propriedades Semelhantes (atributos), mesmo comportamento (operações), mesmos relacionamentos com outros objetos (associações e agregações), e mesma semântica
- Um objeto é uma instância de uma classe

# Classe

- Uma classe é uma abstração que
  - Enfatiza características relevantes
  - Abstrai outras características
  - Abstração: ajuda lidar com a complexidade

# Exemplos de Classes

- Professor
  - Atributos: Nome, Matrícula, Data de Contratação, Titulação
  - Operações: DefineNome(), AlteraNome(), TempoServiço(), DefineTitulação(), AlteraTitulação(), ...
- Turma:
  - Atributos: Cod, Nome, Local, Créditos, Horário, Capacidade
  - Operações: DefineCod(), AlteraCod(), DefineNome(), AlteraNome(), NrCreditos(), AdicionaAluno(), EliminaAluno(), VerificaEstado(), ...

# Classe

- Encontrando Classes
  - Uma classe deveria capturar uma e somente uma abstração chave.
    - Abstração ruim: classe "Aluno" que conhece a informação do aluno e as disciplinas que aquele aluno está matriculado.
    - Boa abstração: separar em uma classe para Aluno e uma classe para Disciplina

# Classe

- Nomeando Classes
  - Uma classe deveria ser um substantivo singular que melhor caracteriza a abstração
  - Dificuldades na nomeação das classes podem indicar abstrações mal definidas
  - Nomes deveriam surgir diretamente do domínio do problema

# Classe

- Estilo para Nomear Classes
  - Um guia de estilo deveria ditar convenções de nomeação para classes
  - Uma proposta simples:
    - Classes são nomeadas com um substantivo no singular
    - O nome de uma classe inicia com a primeira letra maiúscula
    - Não são utilizados símbolos de sublinhado ("\_") - nomes compostos de múltiplas palavras são organizados com todas as palavras juntas, onde a primeira letra de cada uma fica em maiúscula
    - Exemplos: Aluno, Professor, ControleAcadêmico

# Atributo

- O estado de um objeto é dado por valores de atributos e por ligações que tem com outros objetos
- Todos os objetos de uma classe são caracterizados pelos mesmos atributos, ou variáveis de instâncias
- O mesmo atributo pode ter valores diferentes de objeto para objeto

# Atributo

- Atributos são definidos ao nível da classe, enquanto que os valores dos atributos dos objetos são definidos ao nível do objeto
- Exemplos:
  - uma pessoa (classe) tem os atributos nome, data de nascimento e peso
  - João é uma pessoa (objeto da classe pessoa) com nome "João da Silva", data de nascimento "18/03/1973" e peso "70Kg"

# Métodos

- O comportamento invocável de objetos são os métodos
- Um método é algo que se pode pedir para um objeto de uma classe fazer
- Objetos da mesma classe tem os mesmos métodos
- Métodos são definidos ao nível de classe, enquanto que a invocação de uma operação é definida ao nível de objeto

# Construtor

- É um método utilizado para inicializar objetos da classe quando estes são criados
- Este método possui o mesmo nome da Classe e não tem nenhum tipo de retorno, nem mesmo void

# Visibilidade de atributos e métodos

- Atributos e Métodos Públicos
  - São atributos e métodos dos objetos que podem ser visíveis externamente, ou seja, outros objetos poderão acessar os atributos e métodos destes objetos sem restrições
- Atributos e Métodos Privados
  - Atributos e métodos que só podem ser acessados por operações internas ao próprio objeto são ditos privados

# Métodos

- Interface
  - Os métodos públicos de uma classe definem a interface da classe. Os métodos privados não fazem parte interface da classe pois não pode ser acessada externamente
- Assinatura
  - Deve ser definido o nome dos métodos, se tem tipo de retorno, e se recebe parâmetros, a este conjunto de informações sobre o método dá-se o nome de assinatura do método

# Mensagens

- Uma mensagem é uma solicitação feita por um objeto A a um objeto B
- Como resultado desta solicitação, o objeto B irá modificar seu estado ou irá retornar algum valor
- O conceito de mensagem está diretamente associado ao conceito de operação
- A interação entre os objetos é feita através da troca de mensagens

# Herança e Hierarquia

- **Hierarquia**
  - Quando vamos trabalhar com um grande conjunto de classes de objetos, é necessário organizar estas classes de maneira ordenada de modo que tenhamos uma hierarquia
  - Em uma hierarquia de classes teremos as classes mais genéricas no topo, e as mais específicas na base.

# Hierarquia

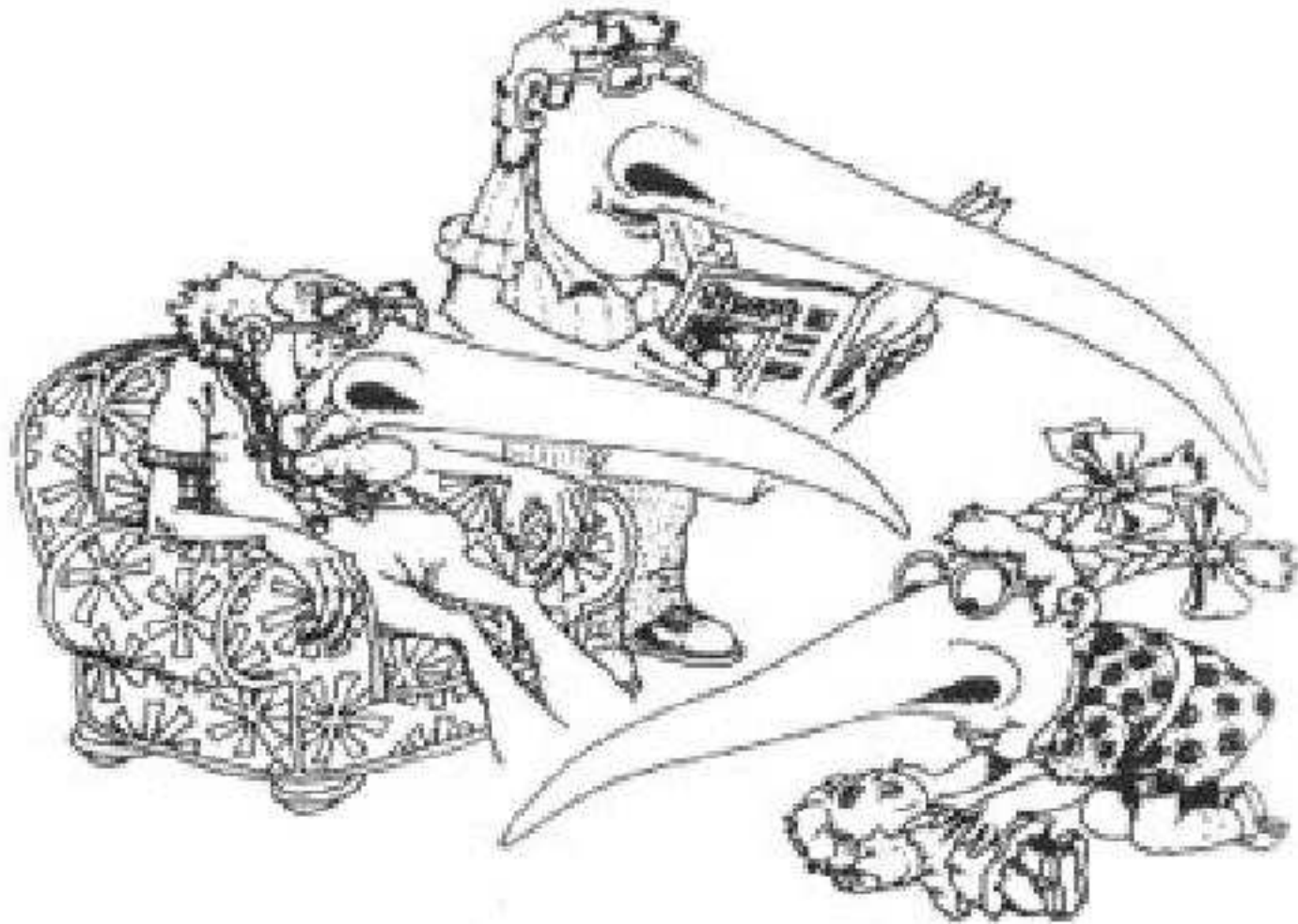
- **Automóveis**
  - automóveis utilitários (camionetes leves)
    - utilitários urbanos
    - utilitários off-road
  - automóveis de passeio
    - passeio família
    - passeio esportivo
  - automóveis de carga
    - carga inflamáveis
    - carga com frígirifico

# Herança e Hierarquia

- **Herança**

- Em uma hierarquia de classes semelhantes podemos dizer que as classes mais específicas herdam as características das mais genéricas, ou seja, todo automóvel de passeio família é um automóvel de passeio
- A classe de nível superior na na associação de herança é chamada de super-classe e a inferior de sub-classe
- Portanto automóvel de passeio família é uma sub-classe de automóvel de passeio

# Herança



# Herança e Hierarquia

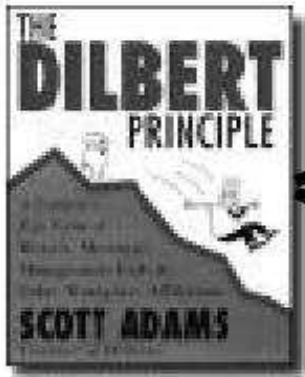
- Agregação
- Composição
- Herança x Agregação x Composição

# Agregação / Composição

- É o princípio que permite o desenvolvedor considerar algo muito grande através do enfoque TODO-PARTE
- Utilizado para definir regras de negócio
  - Se a instância do todo for removida, suas partes também deverão ser removidas (composição)
  - Se a instância do todo for removida, suas partes não necessariamente deverão ser removidas (agregação)

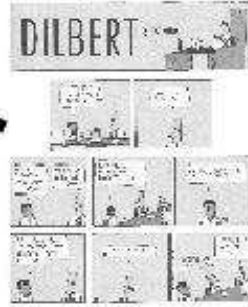
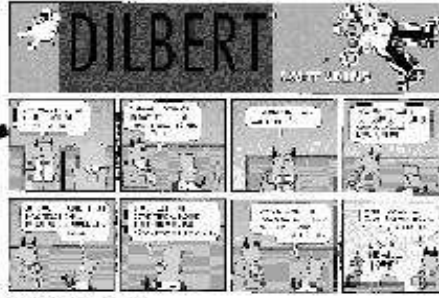
# Agregação

- Define uma "dependência fraca" entre o todo e suas partes
- Se o todo for removido, as suas partes poderão continuar existindo
  - é um mecanismo que permite a construção de uma classe agregada a partir de outras classes componentes. Usa-se dizer que um objeto da classe agregada (Todo) tem objetos das classes componentes (Parte)

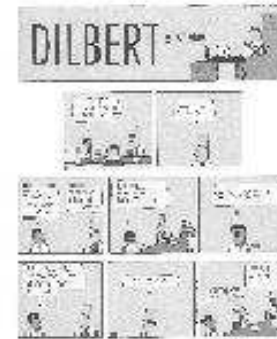
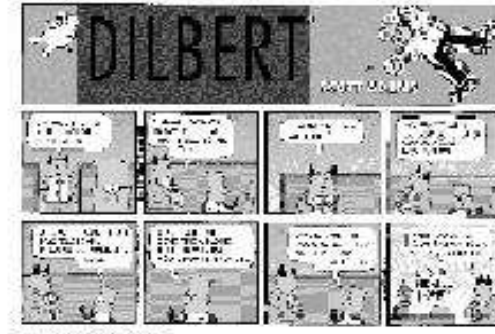


Todo

Agrega



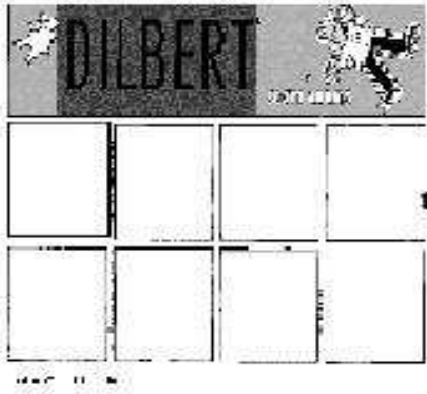
Partes



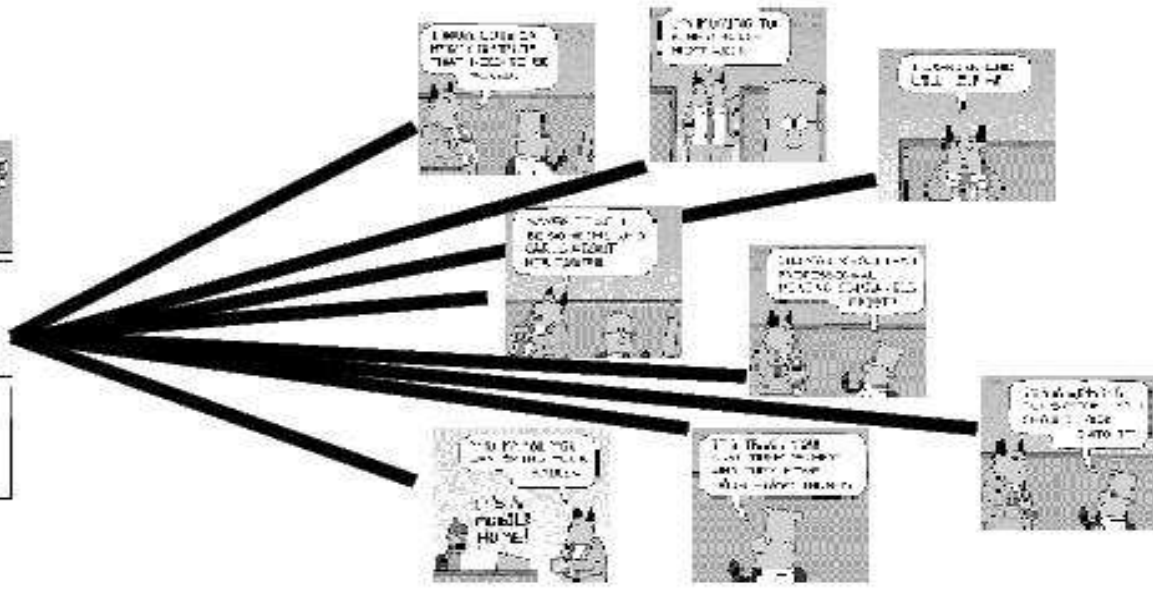
Partes

# Composição

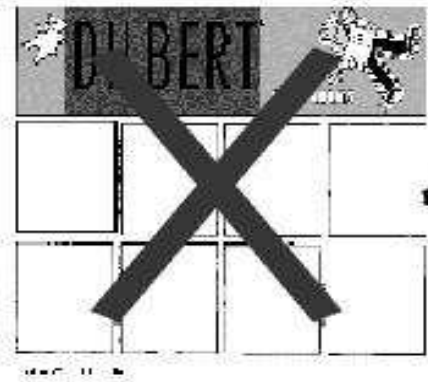
- Define uma "dependência forte" o todo e suas partes
- Se o todo deixar de existir, as suas partes também deixam de existir



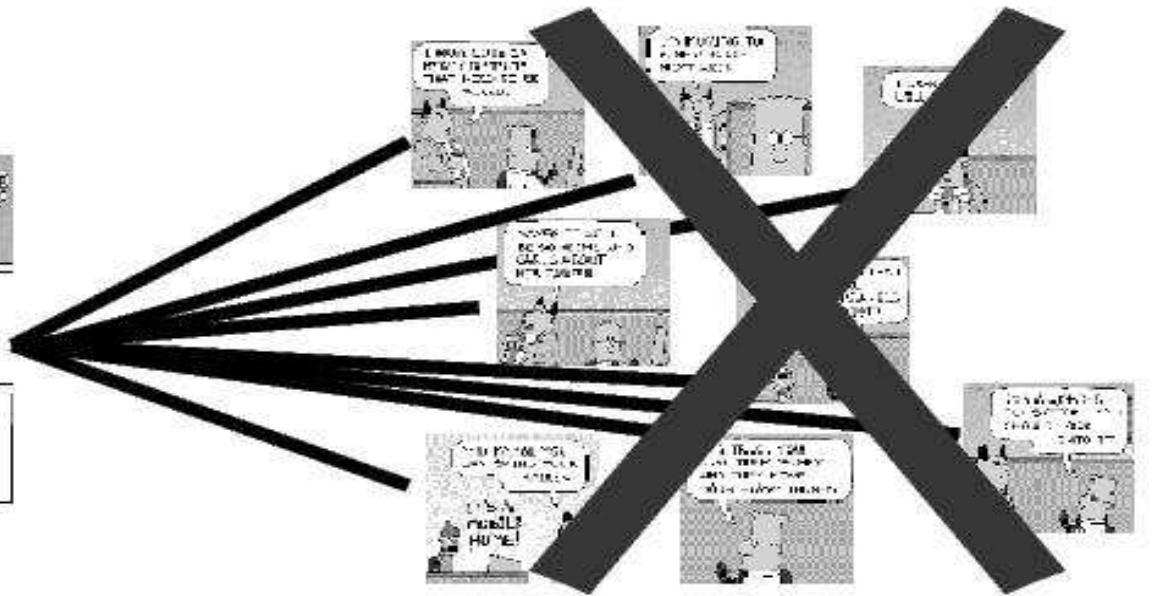
**Todo**



**Partes**



**Todo**



**Partes**

# Herança X Agregação x Composição

- Herança
  - "é um tipo de", as propriedades são herdadas de outra classe
- Associação
  - "usam um(a)", um objeto envia uma mensagem para outro objeto
- Composição
  - "composto por", um objeto é composto por outros objetos

# Abstração

- Abstração é o processo através do qual detalhes são ignorados, para nos concentrarmos nas características essenciais
- A abstração nos leva a representar os objetos de acordo com o ponto de vista e interesse de quem os representa

# Abstração

- Para descrevermos um automóvel (do ponto de vista de um observador externo), identificamos a cor do mesmo, o número de portas, o tipo das rodas e pneus.
- Quando identificamos o automóvel apenas a partir destas características externas estamos fazendo uma abstração pois uma série de detalhes internos não estarão sendo descritos.

# Abstração

- Ex: Ao modelarmos um objeto avião no contexto de um sistema de marcação de passagens aéreas, não vai nos interessar a característica número de turbinas do avião, mas irá nos interessar a característica número de assentos disponíveis
- Ao ignorarmos algumas características não relevantes em um determinado contexto, estamos fazendo uma abstração

# Encapsulamento

- É o processo de ocultação das características internas do objeto
- O encapsulamento cuida para que certas características não possam ser vistas ou modificadas externamente

# Encapsulamento

- Exemplo
  - Podemos dizer que o motor de um automóvel está encapsulado, pois normalmente não podemos ver ou alterar características do motor
  - Podemos então utilizar um automóvel sem conhecer nada das complexidades do motor, que estão encapsuladas

# Encapsulamento

- No contexto de uma linguagem OO um objeto pode ser definido como um conjunto de "funções" unidas aos dados que elas necessitam
- O encapsulamento "protege" os dados que estão "dentro" dos objetos, evitando assim que os mesmos sejam alterados erroneamente
- Os dados só poderão ser alterados pelas "funções" dos próprios objetos
- As "funções" dos objetos são chamadas de operações ou métodos

# Polimorfismo

- É a capacidade de objetos diferentes reagirem segundo a sua função a uma ordem padrão.
- O comando "abre", por exemplo, faz um objeto entrar em ação, seja ele uma janela, uma porta ou uma tampa de garrafa

# Polimorfismo

- Sobrecarga
  - Através do mecanismo de sobrecarga, dois métodos de uma classe podem ter o mesmo nome, desde que suas assinaturas sejam diferentes.
  - Tal situação não gera conflito pois o compilador é capaz de detectar qual método deve ser escolhido a partir da análise dos argumentos do método.

# Persistência

- Refere-se à habilidade de um objeto existir além da execução que o criou, ou seja, ser armazenado em memória secundária (permanente ou persistente).
- Não é o armazenamento apenas dos atributos (dados), mas também dos métodos para realizar os acessos.
- Muito encontrado em sistemas que manipulam banco de dados

# **Análise e Desenvolvimento**

# Por que utilizar Orientação a Objetos?

- Quando bem empregada, a Orientação a Objetos traz diversas vantagens:
  - reutilização,
  - confiabilidade,
  - modelo de sistema mais realístico,
  - facilidade de interoperabilidade e de manutenção,
  - aumento da qualidade,
  - maior produtividade
  - unificação do paradigma (da análise a implementação).
- A Orientação a Objetos é um paradigma que pode ser aplicado ao longo de todo o processo de construção do software.

# UML (Unified Modeling Language)

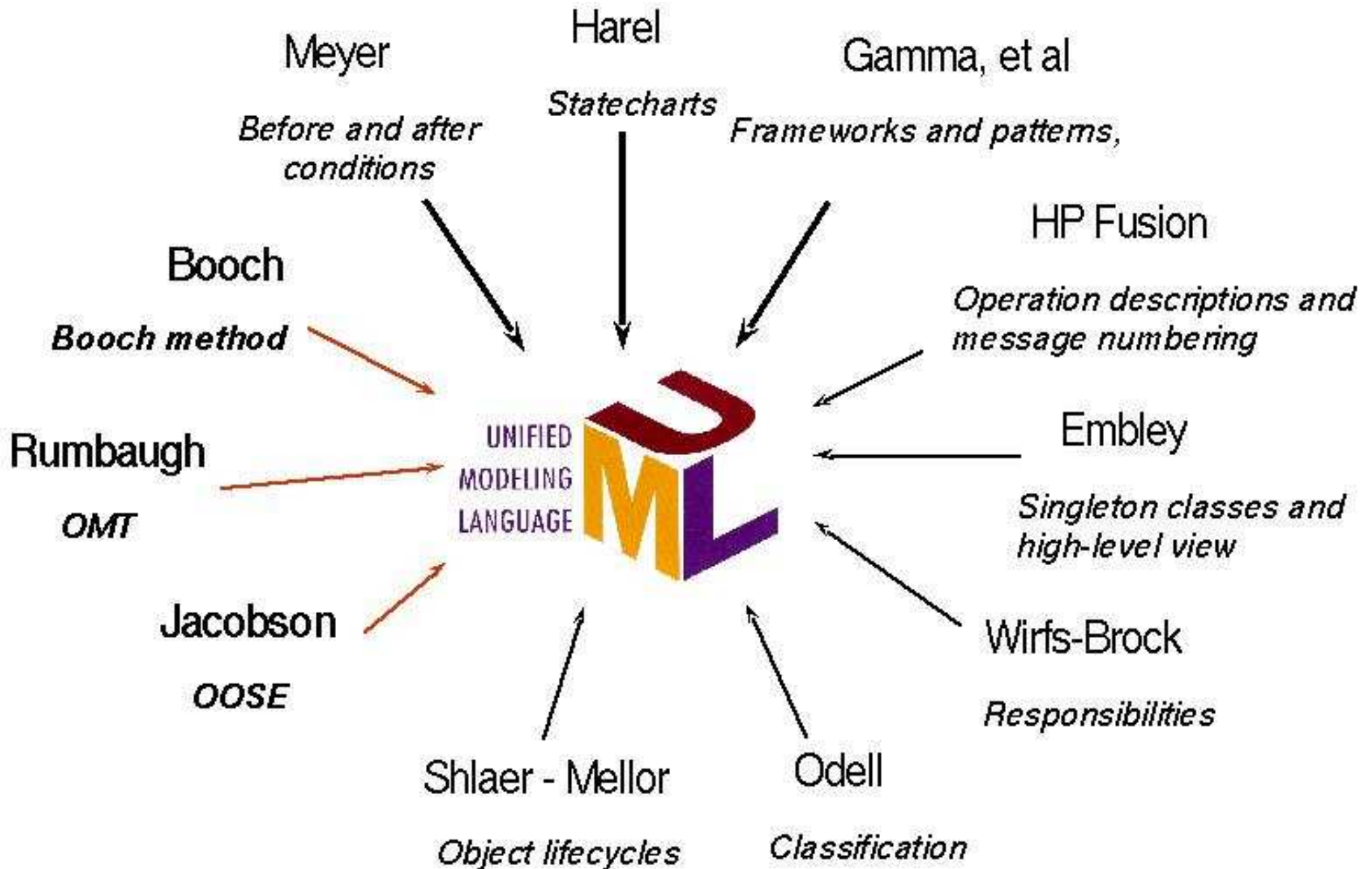
- UML é uma linguagem para **especificação, construção, visualização e documentação** de sistemas de software.
  - UML não é uma metodologia não diz quem deve fazer o quê, quando e como UML pode ser usado segundo diferentes metodologias, tais como RUP (Rational Unified Process), FDD (Feature Driven Development), etc.
  - UML não é uma linguagem de programação

# UML (Unified Modeling Language)

- Modelos e Diagramas
  - Um modelo é uma representação em pequena escala, numa perspectiva particular, de um sistema existente ou a criar
  - Ao longo do ciclo de vida de um sistema são construídos vários modelos, sucessivamente refinados e enriquecidos
  - Um modelo é constituído por um conjunto de diagramas (desenhos) consistentes entre si, acompanhados de descrições textuais dos elementos que aparecem nos vários diagramas

# UML (Unified Modeling Language)

- Um diagrama é uma vista sobre um modelo
- O mesmo elemento (exemplo: classe) pode aparecer em vários diagramas de um modelo
- No UML, há nove diagramas standard
- Diagramas de visão estática: casos de utilização (use case), classes, objectos, componentes, distribuição (deployment)
- Diagramas de visão dinâmica: sequência, colaboração, estados (statechart), actividades



**F I M**

**Maurício de Castro**  
SOLIS/UNIVATES  
[mcastro@solis.coop.br](mailto:mcastro@solis.coop.br)

Coordenador de Desenvolvimento de Sistemas em  
Software Livre da SOLIS/UNIVATES