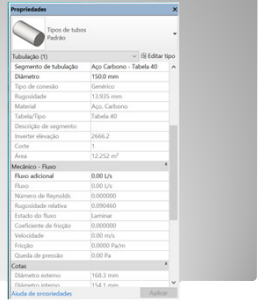




# Modelagem de Informações da Construção

### Propriedades

- Atributos de:
  - Desempenho
  - Materiais
  - Acabamento
  - Composição
  - Posição
  - Geometria (parâmetros)
  - ...



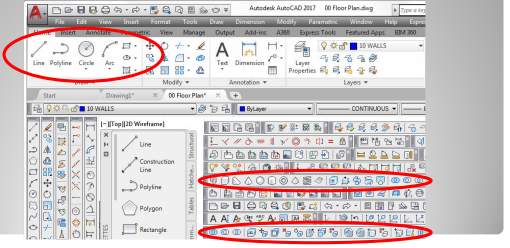
### Diferença CAD 3D x BIM



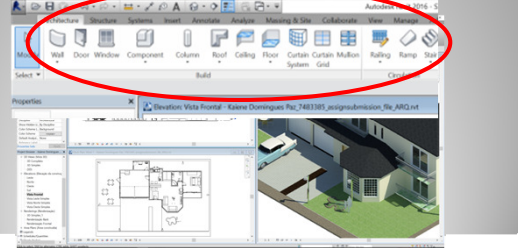
Objetos BIM têm **SEMÂNTICA!**



### Exemplo de Aplicativo CAD



### Exemplo de Aplicativo BIM




### Modelagem orientada a objetos

- Semântica (significado)
  - Tipos de **objetos** (parede, tubo, porta, laje...)
  - E de **propriedades!** (pressão, coef. de transferência térmica, cor, resistência, fabricante, material,...)
- Comportamentos
  - Funcionalidade, inteligência...
- Classes e Instâncias
  - Classe: gabarito para criação de um tipo de objeto
  - Instância: cada exemplar criado a partir de uma classe

### Classes

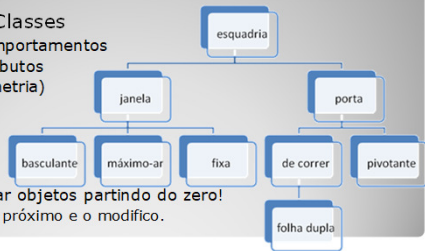
- São organizadas segundo uma hierarquia
- Classes abstratas não podem ser instanciadas



# Modelagem de Informações da Construção

## Modelagem orientada a objetos (cont.)

- Hierarquia de Classes
  - Herança de comportamentos
  - Herança de atributos (inclusive geometria)



- Não preciso criar objetos partindo do zero!
  - Escolho o mais próximo e o modifico.

## Modelagem orientada a objetos (cont.)

- Todas as instâncias de uma mesma classe têm:
  - Os mesmos comportamentos
  - Os mesmos atributos
- Atributos
  - De **tipo**: têm o mesmo valor em todas as instâncias (p. ex: forma)
  - De **instância**: podem ter valores diferentes (p. ex: cor)



## Orientação a Objetos

Conceitos de Modelagem de Dados

- Abstração de dados;
- Objeto;
- Classe;
- Encapsulamento;
- Reusabilidade;
- Herança;
- Tipo abstrato;
- Interface;
- Polimorfismo
- Sobreposição (*overriding*)
- Sobrecarga (*overloading*).

## Conceitos essenciais da OO

- Crise do software (1970's).
  - Hw evoluiu muito mais rápido que o Sw;
  - Custo de manutenção tão importante quanto o de desenvolvimento inicial do Sw;
  - Busca de novos paradigmas de programação...
- Linguagem SIMULA (1964):
  - Introduziu o conceito de Objeto (= dados + métodos)
  - Introduziu Herança e Polimorfismo;
  - Sucedida por Smalltalk, ADA, Objective C, C++, C# Java...
- OO hoje: principal paradigma de programação

## Introdução

- Extensão do conceito de *tipo* (de dados).
  - REAL, INTEGER, CHAR, BOOLEAN...
  - Definição de *tipos de dados abstratos*;
- Definição de um novo *tipo abstrato*:
  - Estrutura dos dados
    - Variáveis internas que o compõem;
  - Operadores (métodos)
    - Criação, manipulação, deleção da estrutura
- Se aproxima mais da maneira das pessoas pensarem os problemas reais.

## Abstração de dados

# Modelagem de Informações da Construção

- Estrutura de dados + métodos = entidade ou *objeto*.
  - Variáveis (dados) = **estado** do objeto;
  - Métodos (procedimentos) = **comportamento**;
  - Mensagens = chamadas de funções.
- Ex. de estado - objeto elevador:
  - Dimensões da cabine;
  - Andar onde está;
  - Direção;
  - Velocidade;
  - Número de pessoas viajando;
  - Massa;
  - Posição da porta (aberta/fechada);
- Universo de Discurso do objeto
  - conjunto de estados possíveis (produto cartesiano de suas variáveis);
  - Idealmente, contém todos e somente os estados possíveis;
  - o comportamento do objeto pode afetar seu U. de Disc.;

**Objeto**


- Classe
  - Modelo/Gabarito ("*template*") a partir do qual são criados ("instanciados") objetos de um certo tipo ("classe").
  - Objetos instanciados contém todas as variáveis e métodos da classe mãe.
  - Quando uma classe é alterada, todas os objetos criados a partir dela (instâncias) também refletem a alteração.

**Classe**

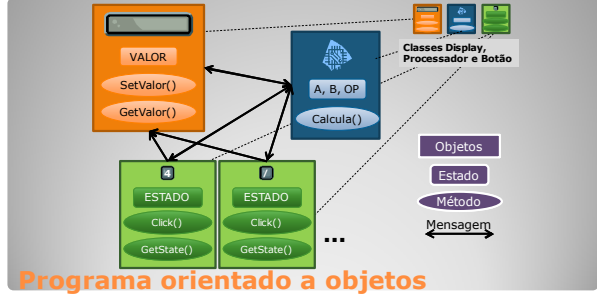
- Programação procedimental
  - "Receita de bolo" (seqüência de comandos)
  - FORTRAN, PASCAL, BASIC, C
  - Subtipo: estruturada ("sem goto").

```

ENQUANTO (VERDADE) {
  val_a = LeValor();
  op = LeOperacao();
  FAÇA {
    val_b = LeValor();
    val_a = Calcula(val_a, val_b, op);
    Mostra(val_a);
    op = LeOperacao();
  } ENQUANTO (op != '=')
}
    
```



**Paradigmas de programação**

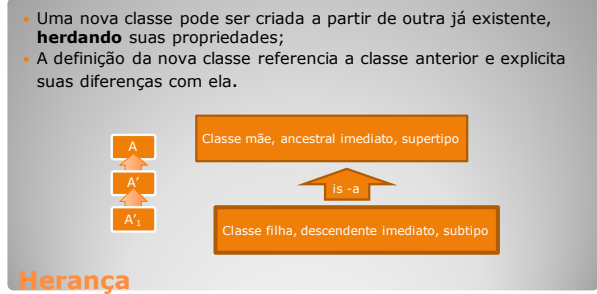


**Programa orientado a objetos**

- Empacotar dados (estado) e as funções (métodos) que os processam:
  - estas funções **não** processam outros dados;
  - os dados **não** são processados por funções externas.
- Viabiliza o reuso;
  - Reusabilidade: facilidade para reaproveitar código já desenvolvido e testado;
  - Bibliotecas de objetos.

**Encapsulamento / Reusabilidade**


- Uma nova classe pode ser criada a partir de outra já existente, **herdando** suas propriedades;
- A definição da nova classe referencia a classe anterior e explicita suas diferenças com ela.



**Herança**

# Modelagem de Informações da Construção

- Herança simples
  - somente uma mãe;



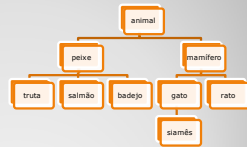
```

graph TD
    animal[animal] --> peixe[peixe]
    animal --> mamifero[mamifero]
    peixe --> truta[truta]
    peixe --> salmão[salmão]
    peixe --> badejo[badejo]
    mamifero --> gato[gato]
    mamifero --> rato[rato]
    gato --> siamês[siamês]
  
```

- Herança múltipla
  - atributos de mais de uma classe podem ser herdados por uma classe filha;
  - Possível problema com duplicações (métodos e/ou variáveis).

**Herança**

- Tipos **abstratos** servem para organizar uma hierarquia de classes;
- Não podem ser instanciados.
  - Não se poderia criar um objeto do tipo "animal", por exemplo, se sua forma não estiver definida na classe animal. Ela seria uma **classe abstrata**.



```

graph TD
    animal[animal] --> peixe[peixe]
    animal --> mamifero[mamifero]
    peixe --> truta[truta]
    peixe --> salmão[salmão]
    peixe --> badejo[badejo]
    mamifero --> gato[gato]
    mamifero --> rato[rato]
    gato --> siamês[siamês]
  
```

**Tipo (classe) abstrato**


- Uma interface específica como uma classe é para o mundo externo.
  - Quais métodos implementa;
  - Que variáveis públicas tem;
- Diferentes classes podem implementar a mesma interface.
  - Exemplo:
    - A interface "alimentação oral" é implementada por mamíferos e por peixes, mas não por plantas e insetos.

**Interface**

- Um objeto herda as propriedades (atributos e operadores) de todos os seus ancestrais (mãe, avó, bisavó...);
  - Um objeto pode pertencer a **múltiplas classes** (a dele e as de todos os seus ancestrais):
    - Um siamês é um gato e é um mamífero. Mãe como todos os gatos e mama como todos os mamíferos;
    - Um siamês pode substituir um gato ou um mamífero em qualquer situação.
  - Há diferentes formas de compatibilidade entre classes herdadas, de acordo com a linguagem de programação adotada;

**Polimorfismo**

- Ao estender-se uma classe, seus métodos (procedimentos) podem ser **sobrepostos** (*overridden*) por outro da classe filha, mantendo-se o mesmo nome mas alterando seu comportamento;




```

graph TD
    poligono[Polígono] --> poligono_hachurado[Polígono hachurado]
  
```

**Sobreposição**

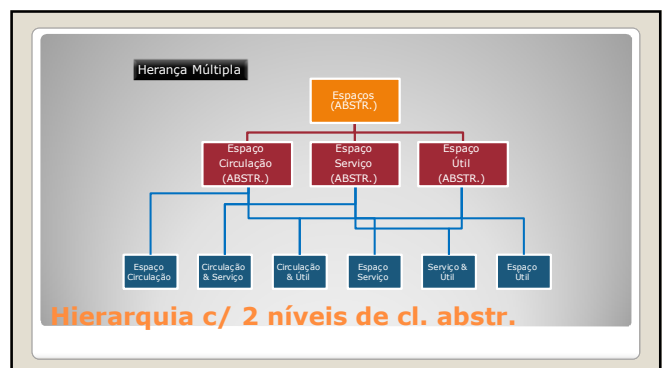
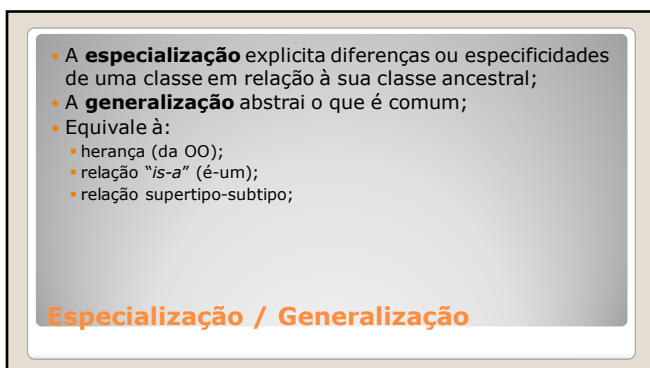
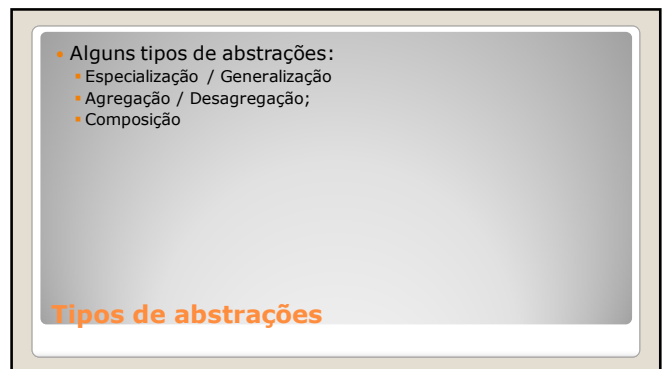
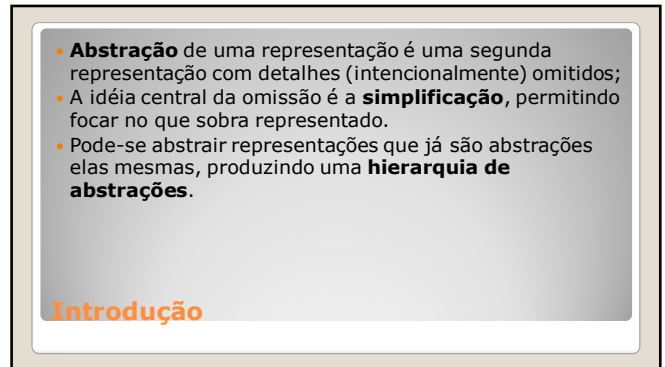
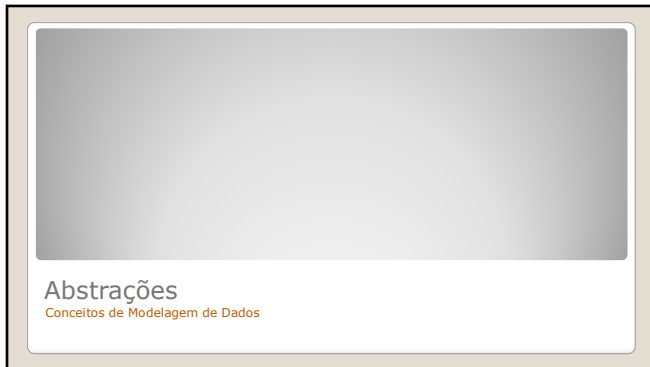
- Operadores ou métodos podem ser sobrecarregados (*overloaded*), fazendo com que seu comportamento mude de acordo com seus parâmetros (tipo e quantidade), porém sem mudar de nome.
  - $x^2 = \text{quadrado}(x)$ ;



```

graph TD
    quadrado_int[quadrado(int)]
    quadrado_real[quadrado(real)]
    quadrado_complex[quadrado(complex)]
  
```

**Sobrecarga**



# Modelagem de Informações da Construção

- ANY
  - Uma classe pode ser qualquer combinação de suas superclasses abstratas;
- ONEOF
  - Uma classe só pode herdar de uma superclasse.

## Restrições de Herança

- Agregação é o tipo mais básico de abstração;
- Envolve agrupar dados/entidades e dar a elas um nome (outra entidade) que as descreve genericamente;

### Exs:

- "animal de *pêlo* curto, de *porte* pequeno a médio, que *late*" = *cachorro*;
- "veículo de *passageiros* com duas *fileiras de bancos*, com *espaço* considerável no banco de trás para três adultos, com três *volumes*, sendo o *compartimento traseiro* para bagagens" = *sedan*.



## Agregação / desagregação

- Os dados agregados são normalmente chamados de **atributos** da entidade:
  - Ex: Parede: cor, material acabamento;
- Atributos são entidades usadas para descrever propriedades de outras entidades.
- A entidade descrita é a **referência** do atributo;
- Atributos não têm um valor significativo sem um referente:
  - são entidades de "2ª classe";
  - os referentes são entidades de "1ª classe";
- Atributos podem ser medidas ou propriedades do referente.
- Conjuntamente, os atributos definem o estado do referente.

## Atributos

- Em CAD, a **descrição geométrica** não é uma entidade de 1ª classe; é um **atributo** de outra entidade;
  - Deve ser possível definir um componente de um edifício e posicioná-lo sem descrever sua forma;
- Pode-se também definir os atributos de um objeto antes de decidir o que o objeto é:
  - Criar uma forma como parte de um edifício sem saber o que irá dentro dela (sua função);
  - Estabelecer um conjunto de requisitos de desempenho para um edifício, sem saber qual é;
  - (Nestes casos, é necessário referir os atributos a uma entidade genérica que será, mais tarde, especializada).

## Atributos

- Dada uma agregação, todos os seus elementos (atributos) se referem à **mesma** entidade particular:
  - Pode-se definir regras de agregação.
  - Ex:
    - todas as propriedades do material devem ser possíveis num único material;
    - todas as entidades geométricas associadas a uma entidade 3D (planta, elevação) devem ser consistentes com o modelo.
    - área e volume devem ser consistentes com a geometria de uma entidade
- Agregação é implementada nas classes (estado)
  - Herança é usada para abstrair agregações, formando hierarquias.

## Regras de agregação

- Se referem à qualidade dos dados.
- Há **consistência** de dados, se:
  - Dados duplicados carregam valores equivalentes;
  - Dados que podem ser derivados de outros dados são também equivalentes aos que seriam derivados do dado original.
- **Integridade** de dados:
  - Os dados são corretos de acordo com quaisquer regras e relações que sejam aplicados a eles;
  - É sempre relativa a algum conjunto definível de regras semânticas.

## Consistência e Integridade

# Modelagem de Informações da Construção

- **Composição** agrupa entidades em objetos compostos (p. ex. montagens);
- Relação **part\_of**;
  - Classe do objeto composto;
  - Definido pelos seus componentes.
- Classes dos objetos componentes.
  - que podem, por si mesmos, também serem compostos (composição hierárquica).
- **Composição x agregação**
  - Os atributos das partes podem ser *independentes* dos atributos do objeto composto;
  - Ex: atributos de uma janela e da parede que a contém.
  - Os atributos agregados são *herdados* do referente.

## Composição

- **Componentes:**
  - Não são atributos;
  - São entidades de 1ª classe.
  - São independentes da montagem que compõem.
  - Podem fazer parte de mais de uma composição
  - Ex: uma bomba pode fazer parte da montagem elétrica e da montagem hidráulica.

## Componentes

- A composição tem uma organização e **regras** que definem quais organizações são possíveis e quais não são;
- Ex:
  - Regras mínimas que definem a correção de uma tubulação:
    - Objetos devem estar conectados;
    - Condições de conexão respeitadas (diâmetro, rosca/cola);
    - Direção de fluxos consistentes (bombas, medidores de pressão, etc.);
    - Etc...
  - Regras codificam conhecimento:
    - Algumas regras são intuitivas:
      - tubos devem estar conectados;
      - ...mas nunca para o computador.
    - Outras são mais sutis:
      - Vazão e direção do fluxo numa tubulação.

## Regras de composição

- No exemplo abaixo, pode-se considerar espaço uma abstração de **especialização** ou de **composição**.
- Como especialização: os **atributos** de espaço (genérico) são herdados pelos espaços específicos;
- Como composição: um espaço é composto por espaços úteis e de serviço. Pode haver **regras** específicas para essa composição (adjacência, sobreposição, etc.)



## Especialização x Composição

## Relações

Conceitos de Modelagem de Dados

- É uma associação de uma entidade a outra(s).
- Relações de abstração **genéricas**: *especialização, agregação, composição*.
- Outras relações, **específicas** da construção:
  - "colocado dentro de" (móveis em um espaço);
  - "conexão elétrica";
  - "conexão estrutural";
  - "precede" (relação entre duas tarefas);
  - "recurso necessário" (entre material/equipamento e tarefa);
  - "gera" (entre um evento e um resultado construído);
- Portanto, há a necessidade de definir-se relações **específicas**.

## Relações



# Modelagem de Informações da Construção

- **Cardinalidade** ou aridade
  - A relação é entre 2 entidades?; ou entre uma e muitas?; ou um número fixo...;
  - 1-para-1; 1-para-2; 1-para-M; ou N-para-M;
  - Importante em Banco de Dados.
- Em BD, todas as relações são bi-direcionais (se  $A \rightarrow B$ , então  $B \rightarrow A$ );
- Em OO, as relações são uni-direcionais.
  - A bidirecionalidade tem que ser explicitada com duas relações distintas:  $A \rightarrow B$  e  $B \rightarrow A$ ;
  - As relações entre A e B necessariamente são inversas.

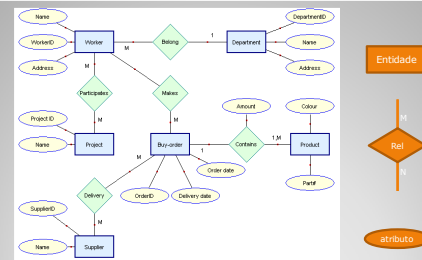
## Propriedades de relações

- EASTMAN, C. M. **Building Product Models: Computer Environments Supporting Design and Construction**. Boca Raton: CRC Press, 1999. 411p., **Chapter 4**.

## Bibliografia

## Modelagem conceitual e de dados

Conceitos de Modelagem de Dados



## Modelo Entidade-Relacionamento (E-R)