1-Orientação a Objetos

ANÁLISE ESTRUTURADA X ANÁLISE O.O.

- Enfoque Tradicional: Conjunto de programas que executam processos sobre dados.
- Enfoque Baseado em Objetos: Conjunto de "coisas" que tem características e comportamentos próprios.

Novas tecnologias sempre trazem novas soluções e novos problemas (nada é perfeito)

Com o rápido crescimento da utilização da Orientação a Objeto surgiram vários métodos

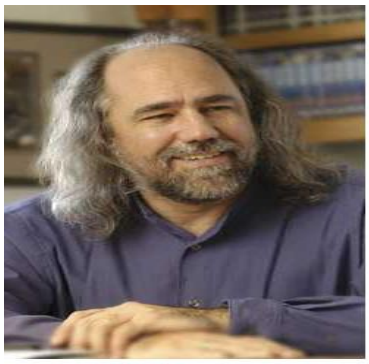
Os primeiros sugiram nos anos 70-80 mas no período de 1989-1994 passaram de 10 para 50

Entre os mais significativos podemos destacar:

- BOOCH: Desenho e análise bem próximos permitem que os projetistas tomem decisões de negócio antes de codificar o sistema
- RUMBAUGH: Baseado na semântica de dados, tornou-se um método testado e maduro
- OOSE (Jacobson): Foco em um conjunto de casos de uso, de um modelo de domínio de problema e de uma descrição da interface do sistema
- SHLAER/MELLOR: Método orientado a objeto que utiliza ferramentas tradicionais (não O.O.)
- COAD/YOURDON: Divide a análise orientada a objeto como sendo classes (coleção de objetos com atributos e serviços), objetos(capacidade do sistema reter dados)

UML (The Unifiled Modeling Language)

Como os método Booch e OMT estavam sendo largamente utilizados, Grady Booch e James Rumbaugh juntaram as forças para fazer um método unificado, posteriormente Ivar Jacobson juntou-se a equipe. Atualmente a UML conta com o apoio de vários autores e de várias empresas como: Microsoft, HP, Oracle, IBM, etc . Por notação entende-se especificar, visualizar e documentar sistemas em OO.


grady booch


ivar jacobson


james rumbaugh

http://www.fabiobmed.com.br/booch-rumbaugh-jacobson-e-a-uml-unified-modeling-language/

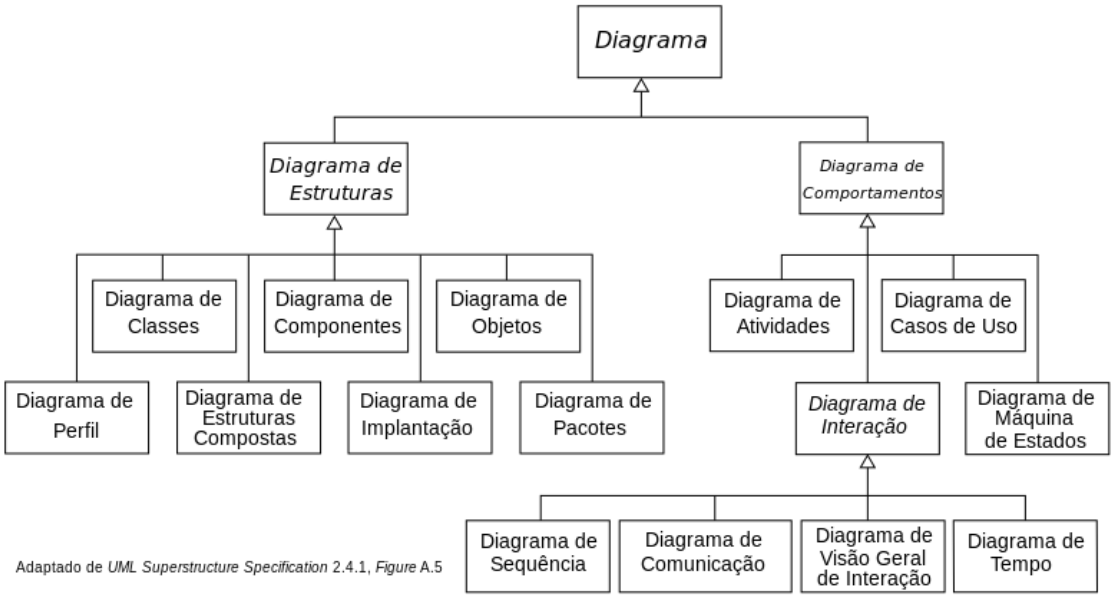


Imagem Wikipidia

1.1 O que é um objeto

"Um objeto é qualquer coisa, real ou abstrata, na qual nós armazenamos dados e as operações que manipulam dados." [Martin]

"Uma abstração de alguma coisa no domínio do problema, refletindo as capacidades do sistema de manter informação sobre ele, interagir com ele, ou ambos; um encapsulamento de valores de Atributos e seus Serviços exclusivos. (sinônimo: uma Instância)." [Yourdon]

Portanto , qualquer pessoa, lugar, evento, coisa, evento, tela, relatório ou conceito aplicáveis ao projeto de um sistema .

É uma extensão do conceito de objeto do mundo real, em que se podem ter coisas tangíveis ,um incidente (evento ou ocorrência) ou uma interação (transação ou contrato).

Tangíveis => pessoas,livro,automóvel

Incidente => competição, projeto, conserto

Interação => transação,saque, venda

1.2 Como visualizar um objeto ?

Pode-se visualizar um objeto como algo que guarda dentro de si os dados ou informações sobre sua estrutura (atributos) e que possui comportamento definido pelas suas operações. Os dados ficam protegidos pela interface, que se comunica com os demais objetos do sistema. Nessa interface, todo tipo de alteração nos dados do objeto (atributos) somente poderá ser feito por meio de operações, que recebem as solicitações externas , fazem as alterações nos dados (se permitidas) e retornam outras informações para o meio externo.

1.3 Classes

"É o agrupamento de objetos com a mesma estrutura de dados (atributos) e comportamento (operações)" [Rumbaugh]

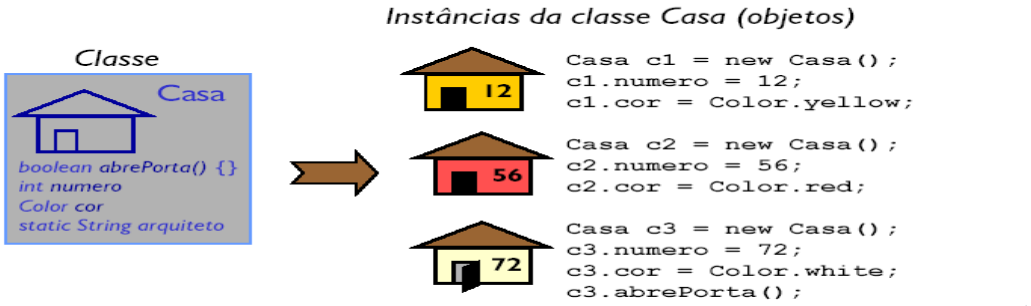
"Uma coleção de um ou mais Objetos com um conjunto uniforme de Atributos e Serviços, incluindo uma descrição de como criar novos Objetos e Classe." [Yourdon]

Uma classe é uma coleção de objetos que podem ser descritos por um conjunto básico de atributos e possuem operações semelhantes. Falamos em um conjunto básico de atributos e operações semelhantes, pois nem todos os objetos da mesma classe precisam ter exatamente o mesmo conjunto de atributos e operações.

Quando um objeto é identificado com atributos e operações semelhantes em nosso sistema, diz-se que pode ser agrupado em uma classe. Esse processo é chamado de generalização. Por outro lado, pode ocorrer que um objeto, ao ser identificado, constitua-se, na verdade, de uma classe de objetos, visto que dele podem se derivar outros objetos. Esse processo é chamado de generalização.

Exemplo : Casa

- *Classes são uma especificação para objetos*
- *Uma classe representa um tipo de dados complexo*
- *Classes descrevem*
  - *Tipos dos dados que compõem o objeto (o que podem armazenar)*
  - *Procedimentos que o objeto pode executar (o que podem fazer)*



1.4 Instâncias de Objetos

Quando se fala em classes de objetos, está sendo considerado que se podem incluir objetos em cada uma delas.

Classe	Instância
Veículos	marca : Opel
	modelo : Fire
	ano : 2002
	potencia : 195cv
	eixos : 2
	carga : 1500 kg

Para cada novo veículo adquirido seria criada uma nova instância de objeto de uma determinada classe.

1.5 Variáveis de Classe :

public (+)– Pode ser acessada por qualquer outra classe , sendo apenas necessário que a classe primitiva que a contém seja pública também e esteja presente no mesmo diretório.

private(-) – Só pode ser utilizada pela classe que ela foi criada .

protected(#) – Só pode ser usada por classes que pertençam ao mesmo pacote que contém a classe que ela foi criada.

1.6 Herança

"É o compartilhamento de atributos e operações entre as classes baseado em uma relação hierárquica." [Rumbaugh]

"Qualquer mecanismo que permite um objeto incorporar toda ou parte da definição de outro objeto como parte de sua própria definição." [Yourdon]

- Conexão semântica de elementos na qual a subclasse herda as propriedades da super-classe, ou seja um objeto pertencente a sub-classe tem além de seus atributos e métodos também possui todos os atributos e métodos da super-classe.

- Relação "é um" / "é um tipo de"

- Ligada ao conceito de Generalização e Especialização:

1.6.1 Generalização:

- Observamos objetos e classes com grau mais alto de especificidade e a partir destas criamos classes mais genéricas de dizemos que estas são superclasses e as mais específicas sub-classes.

- Classes especificas herdam atributos e métodos das classes genéricas .

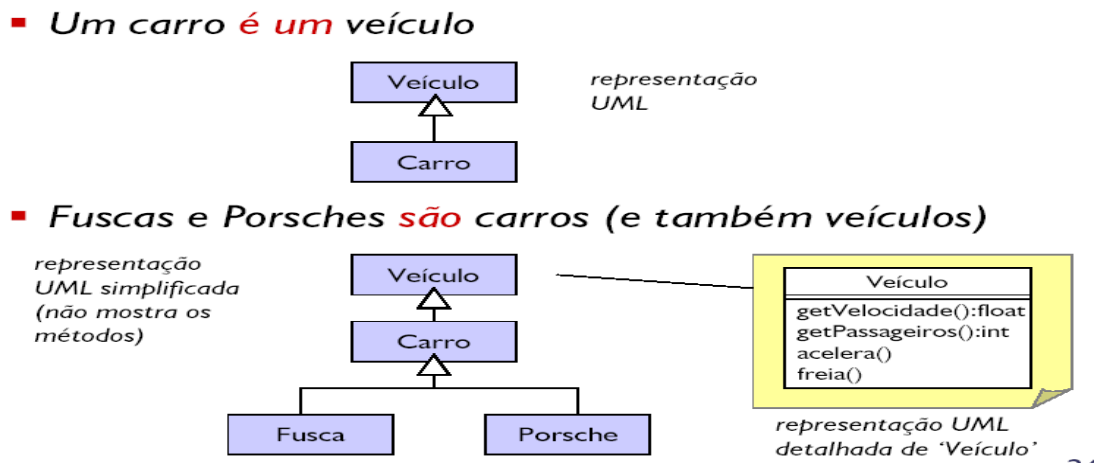
- Ex: Temos as classes Leão, Cachorro, Gato, Macaco, etc, e a partir destas criamos uma classe Animal que contém todos os atributos e métodos comuns.

1.6.2 Especialização:

- Temos uma classe genérica e a partir desta criamos as classes mais especializadas.

- Classes mais especializadas devem conter além dos atributos e métodos da classe mais genérica, o seus próprios.

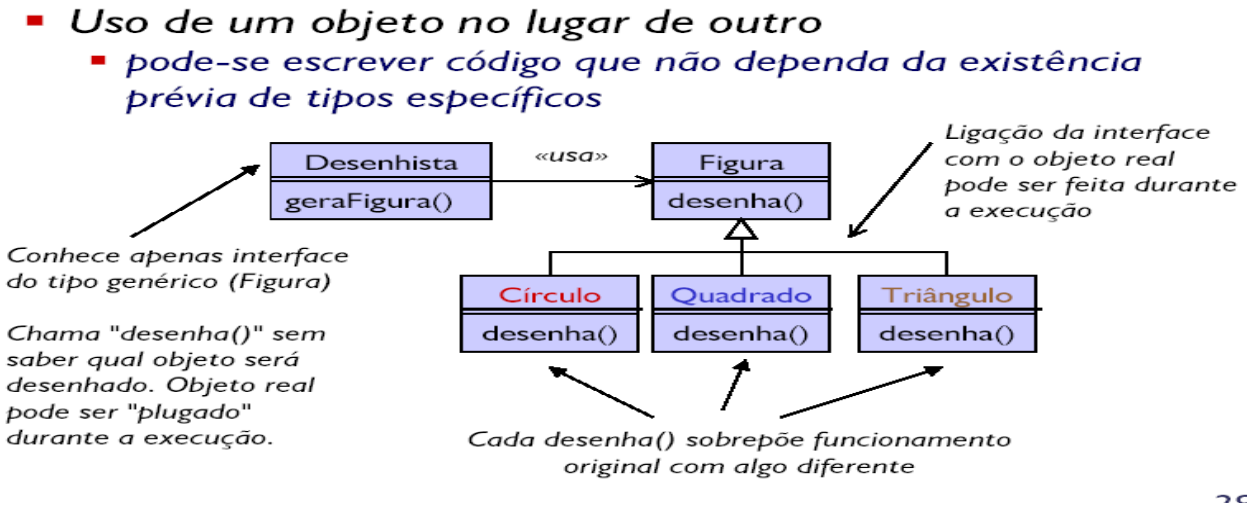
Uma classe é constituída de objetos com atributos e operações semelhantes. Esse princípio orienta a implementação da herança. A herança nada mais é do que a implementação da generalização; é o compartilhamento de atributos e operações entre classes com base em um relacionamento hierárquico.Quando se cria uma nova instância de um objeto, dizemos em orientação a objetos, que esse novo objeto herda os atributos e operações de sua classe.



Superclasse – Se a classe B herda da classe A, então dizemos que A é uma superclasse de B e B é subclasse de A .

1.7 Polimorfismo

Associado ao conceito de herança , o polimorfismo permite que um objeto assuma um comportamento diferente daquele definido em sua classe. Polimorfismo significa a capacidade de assumir muitas formas e que os outros objetos podem interagir com ele sem se preocuparem com a sua forma específica num dado momento.



1.8 Métodos

Um método é simplesmente uma função que pertence a uma classe.

Métodos construtores :

Ao criar uma instância de objeto, seu programa atribui normalmente valores iniciais aos membros de dados do objeto. Para simplificar o processo de inicialização dos membros do objeto, Java oferece suporte a uma função especial chamada construtor, que executa automaticamente quando é criada a instância. A função construtor é um método publico que utiliza o mesmo nome da classe.

- *Construtores são procedimentos realizados na construção de objetos*
  - *Parecem métodos, mas não têm tipo de retorno e têm nome idêntico ao nome da classe*
  - *Não fazem parte da definição do tipo do objeto (interface)*
  - *Nem sempre aparecem explícitos em uma classe: podem ser omitidos (o sistema oferece uma implementação default)*
- *Para cada objeto, o construtor é chamado exatamente uma vez: na sua criação*
  - *Exemplo:*

```
> Objeto obj = new Objeto();
```

Chamada de construtor

  - *Alguns podem requerer parâmetros*

```
> Objeto obj = new Objeto(35, "Nome");
```

1.9 Métodos Get e Set

As classes frequentemente fornecem métodos public para permitir aos clientes da classe configurar (Set (atribuir valores)) ou obter ((Get(obter os valores de)) variáveis de instância private.

1.10 Diagrama de Classes de Análise

Representa termos do domínio do negócio : idéias, coisas, e conceitos no mundo real.

•Objetivo: descrever o problema representado pelo sistema a ser desenvolvido, sem considerar características da solução a ser utilizada.

•É um dicionário "visual" de conceitos e informações relevantes ao sistema sendo desenvolvido.

Modelo de Classes de Análise

•Duas etapas: modelo conceitua l(modelo de domínio) e modelo da aplicação.

•Elementos de notação do diagrama de classes normalmente usados na construção do modelo de análise: classes e atributos; associações, composições e agregações (com seus adornos); classes de associação; generalizações (herança)

O modelo de análise não representa detalhes da solução do problema. Embora este sirva de ponto de partida para uma posterior definição das classes de software (especificação).

Uma classe descreve esses objetos através de atributos e operações.

• Atributos correspondem às informações que um objeto armazena.

• Operações correspondem às ações que um objeto sabe realizar.

• Notação na UML: "caixa" com no máximo três .

•Notação na UML: "caixa" com no máximo três compartimentos exibidos.

No diagrama de classes :

- Os atributos e métodos seguem a nomenclatura da UML, ou seja, iniciam com letra minúscula, tendo a primeira letra de cada palavra (a partir da segunda) em maiúscula;

- Todos os atributos possuem seus tipos identificados. Quando o atributo for um objeto, o tipo é citado como *Classe XYZ,* onde XYZ é o nome da classe. Quando o atributo for uma coleção (uma lista), o tipo é citado como *Coteção de XYZ,* onde XYZ é o nome da classe correspondente a cada item da lista;

- Todos os métodos possuem, se existir, sua lista de parâmetros (acompanhados do tipo) e o tipo de retorno;

- Os métodos descritos foram deduzidos a partir do cenário. Entretanto, a lista completa dos métodos só é possível obter a partir de um diagrama de sequências, que tem por objetivo identificar a troca de mensagens existente entre objetos, em cada caso de uso. Por este motivo, relaciono apenas os métodos mais relevantes;

- Todo atributo cujo tipo seja uma classe *enumeration,* não é definido como um atributo derivado, visto a classe *enumeration* atuar como um tipo de dado e não como um relacionamento;

- Atributos e métodos de classe são representados sublinhados, conforme notação da UML;

2.Exemplos

a)CENÁRIO: As informações a seguir se referem á planilha Excel de Gabriel,que controla os gastos mensais com sua conta de luz. Para cada conta de luz cadastra-se: data em que a leitura do relógio de luz foi realizada, número da leitura, quantidade de Kw gasto no mês, valor a pagar pela conta, data do pagamento e média de consumo.
Mensalmente, são realizadas as seguintes pesquisas:
- verificação do mês de menor consumo;
- verificação do mês de maior consumo.

IMAGEM DA PLANILHA:

LISTA DE ACOMPANHAMENTO DE GASTO DE LUZ					
data leitura	nº leitura	kw gasto	valor a pagar	data pagto	média consumo
04/07/2005	4166	460	206,43	15/07/2005	15,33
02/08/2005	4201	350	157,07	15/08/2005	12,06

Menor Consumo	350	ago/05
Maior Consumo	460	jul/05

Identifique as classes, atributos e os métodos.

RESOLUÇÃO:

Classe	Atributos	Métodos
ContaLuz	dataLeitura : date númeroLeitura : integer qtdKwGasto : integer valorPagar : real dataPagamento : date mediaConsumo : real	cadastrarConta verificaMesMenorConsumo : string verificaMesMaiorConsumo : string

b) CENÁRIO: Para fixação do conceito de classes em sala de aula, Prof. Cristina criou com seus alunos a classe TextoSaida.
O objetivo do exercício é criar uma classe que permita configurar um texto por meio de atributos (tamanho da letra, cor da fonte e cor do fundo), escolhendo em que tipo de componente ele deve ser exibido (entre as opções: *labet, edit* e memo).
Para não haver vínculo com linguagens de programação, essa classe não foi criada como herança de uma classe visual.
As cores podem ser escolhidas entre os tons: preto, branco, azul, amarelo ou cinza.

Identifique as classes, atributos e métodos.

RESOLUÇÃO:

Classe	Atributos	Métodos
TextoSaida	texto : string tipoComponente : EnumTipoComponente tamanhoLetra : integer corFonte: EnumCor corFundo : EnumCor	cadastrar exibirTexto
«enumeration» EnumTipoComponente	label edit memo	
«enumeration» EnumCor	preto branco azul amarelo cinza	

Associações

Um associação representa um relacionamento entre classes, e fornece a semântica comum e a estrutura para muitos tipos de "conexões" entre objetos.

Associações são o mecanismo que permite objetos comunicarem-se entre si. Elas descrevem a conexão entre diferentes classes (a conexão entre os objetos atuais é chamada conexão do objeto, ou *link*.

Associações podem ter um regra que especifica o propósito da associação e pode ser uni ou bidirecional (indicando se os dois objetos participantes do relacionamento podem mandar mensagens para o outro, ou se apenas um deles sabe sobre o outro). Cada ponta da associação também possui uma valor de multiplicidade, que dita como muitos objetos neste lado da associação pode relacionar-se com o outro lado.

Em UML, associações são representadas como linhas conectando as classes participantes do relacionamento, e podem também mostrar a regra e a multiplicidade de cada um dos participantes. A multiplicidade é exibida como um intervalo [min...máx] de valores não negativos, com uma estrela (*) no lado máximo representando infinito.

Representação visual de uma Associação em UML

Agregação

Agregações são um tipo especial de associação no qual as duas classes participantes não possuem em nível igual, mas fazem um relacionamento “todo-parte”. Uma Agregação descreve como a classe que possui a regra do todo, é composta (tem) de outras classes, que possuem a regra das partes. Para Agregações, a classe que age como o todo sempre tem uma multiplicidade de um.

- “Parte Todo”, quando uma classe “é parte da outra classe” então ela deve agregar a outra classe

- Um objeto pode ser compostos de outros objetos e espessamos isso através da agregação

- “é parte de” Agregação / “é um tipo de” Herança

- Um carro tem Porta, Roda, Motor, etc (Agregação) / um carro esporte e um carro de luxo é um carro (Herança)

Em UML, Agregações são representadas por uma associação que mostra um romboide no lado do todo.