Introdução a Engenharia de Software

O que é software?

- São programas de computadores, em suas diversas formas, e a documentação associada.
- Um programa é um conjunto de soluções algorítmicas, codificadas numa linguagem de programação, executado numa máquina real.
- Os produtos de software podem ser desenvolvidos para um cliente em particular ou para o mercado geral.
  - Genérico (COTS – Commercial Off-The Shelf)
  - Personalizado – sob encomenda
- Software é um produto conceitual e lógico.

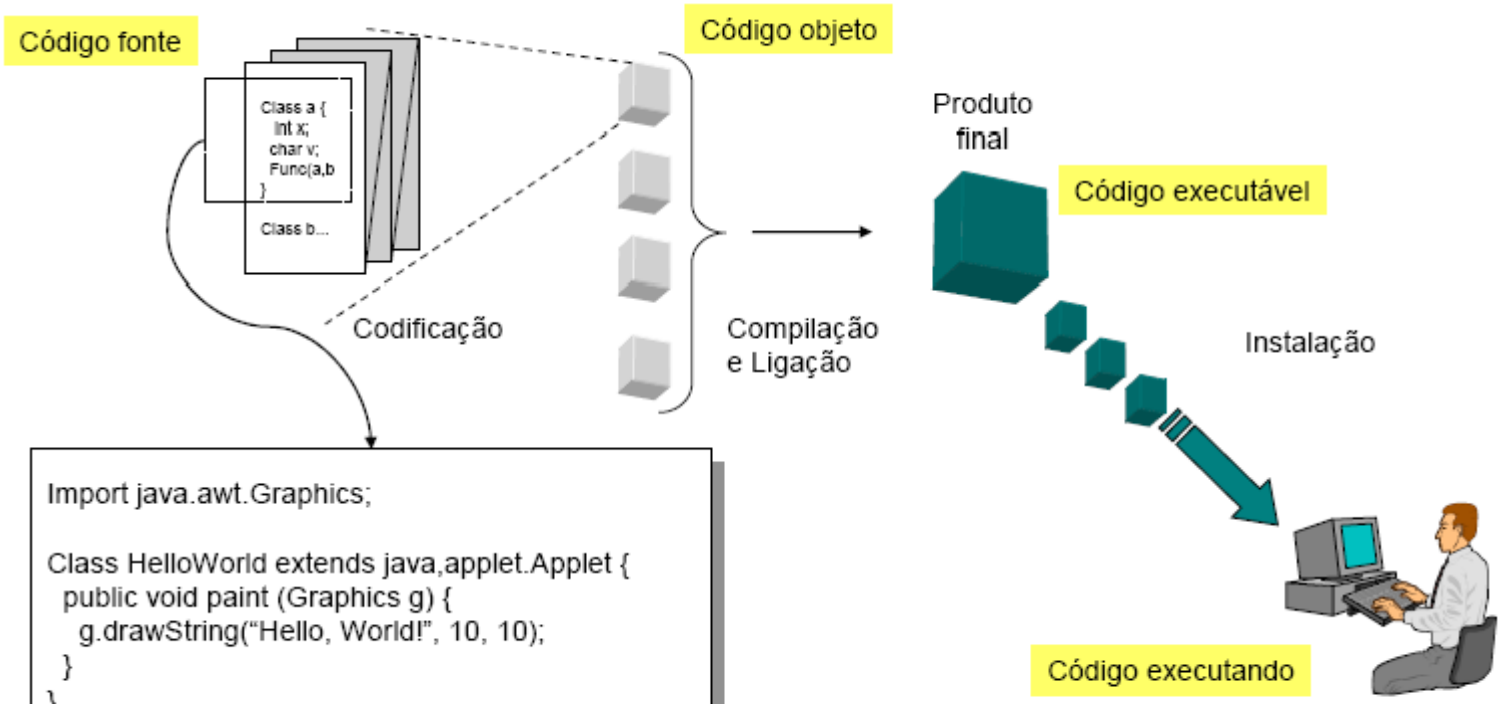
Características

- Invisibilidade
  - Software é invisível e invisualizável
- Complexidade
  - Software é mais complexo do que qualquer outro produto construídos por seres humanos
- Mutabilidade
  - Existe sempre uma pressão para se fazer mudanças em um software

Características

- Conformidade
  - O software deve ser desenvolvido conforme o ambiente. Não é o ambiente que deve se adaptar ao software.
  - Se o software esta conforme os requisitos (o ambiente) todo o suporte operacional deve se adaptar ao software.

Formas do Software



Mitos do Software

- O estabelecimento de objetivos gerais é suficiente para se começar a escrever programas.
- Dê a uma pessoa técnica um bom livro de programação e você terá um programador.
- Mudanças no software podem ser feitas facilmente porque ele é "flexível".
- Até que o programa esteja "rodando" não é possível verificarmos a sua qualidade.
- Uma vez que o programa esteja escrito e funcionando, nosso trabalho está feito.
- Um projeto é bem sucedido se conseguirmos um programa funcionando corretamente.

Histórico

- Os primeiros anos (1950 a início dos 60)
  - Aplicações científicas e de engenharia
- A segunda era (1960 a meados de 80)
  - Aplicações comerciais em grande-porte (sistemas de informação BD)
- A terceira era (meados de 70 e década de 80)
  - Aplicativos pessoais em microcomputadores
- A quarta era (meados de 80 a meados de 90)
  - Aplicativos com Interfaces Gráficas
  - Redes e Arquitetura Cliente-Servidor

Histórico

- A quinta era (de meados de 90 a ???)
  - Software Distribuídos, Internet, Groupwares e Intranets
- Sexta era??
  - Computação Pervasiva, Móvel e Ubíqua

Categorias de Tamanho de Softwares

Categoria	Equipe	Duração	Linhas cod.
Trivial	1	1-4 semanas	500
Pequeno	1	1-6 meses	1000 a 2000
Médio	2-5	1-2 anos	5000 a 50 mil
Grande	5-20	2-3 anos	50 mil a 100 mil
Muito grande	100-200	4-5 anos	1 milhão
Extremamente grande	2000-5000	5-10 anos	1 a 10 milhões

- Win 95: teve 11 milhões de linhas e 200 programadores
- Nestscape: teve 3 milhões de linhas e 120 programadores

Contextualização da Engenharia de Software

O que é a Engenharia de Software?

- É uma disciplina da engenharia dedicada a todos os aspectos da produção de software.
- Engenheiros de software devem adotar uma abordagem sistemática e organizada para o seu trabalho e usar técnicas e ferramentas apropriadas, de acordo com o problema a ser resolvido, e com as restrições e recursos disponíveis.

Engenharia

- Desenvolvimento de um produto;
- Processo de desenvolvimento envolvendo análise, design, implementação e avaliação;
- Baseado em teoria, princípios, modelos, métodos, técnicas e ferramentas;
- Equipe de especialistas;
- Planejamento e gerenciamento de recursos, custos e prazos.

Objetivos da Engenharia de Software

- Aplicação de teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para o desenvolvimento sistemático de software.
- Aplicação de métodos, técnicas e ferramentas para o gerenciamento do processo de desenvolvimento.

- Produção da documentação formal destinada a comunicação entre os membros da equipe de desenvolvimento bem como aos usuários.

Definições de Engenharia de Software

- O estabelecimento e uso de princípios de engenharia para a produção economicamente viável de software de qualidade que funcione em máquinas reais;
- A engenharia de software é a disciplina envolvida com a produção e manutenção sistemática de software que são desenvolvidos com custos e prazos estimados;
- Disciplina que aborda a construção de software complexo com muitas partes interconectadas e diferentes versões por uma equipe de analistas, projetistas, programadores, gerentes, "testadores", etc.

Aspectos históricos

- 1968 Conferência da OTAN, Garmisch
- Objetivo: resolver a “Crise do Software”
- Software é entregue:
  - Atrasado
  - Com orçamento estourado
  - Com falhas residuais
- Custo do hardware decrescente e custo do software em ascensão.

Qual a diferença entre engenharia de software e engenharia de sistemas?

- A engenharia de sistemas está interessada em todos os aspectos de um sistema baseado em computador, incluindo hardware, software, fatores humanos, informação e o processo. A engenharia de software é parte dela.

Princípios da Engenharia de Software

- Todo engenheiro de software deve desenvolver com:
  - Rigor e Formalidade
  - Separação de interesses
  - Modularidade
  - Abstração
  - Antecipação de mudanças
  - Generalidade
  - Possibilidades de evolução

Processos de Software

Como transformar necessidades em software?

- Principais Atividades Envolvidas:
  - Entender as necessidades do cliente;
  - Planejar uma solução;
  - Implementar e testar a solução;
  - Entregar a solução.
- Como essas atividades são executadas?
  - De forma desordenada e informal;
  - Apenas por uma pessoa.

Processo de Desenvolvimento

- O conjunto de atividades de desenvolvimento, sua ordem temporal e a atribuição de responsabilidades (papéis de desenvolvedores) definem um processo de desenvolvimento de software;
- Um processo de software é a especificação do processo de transformar necessidades em software;
- Ciclo de Vida de um Processo:
  - Determina as fases do processo;
  - Define atividades importantes e opcionais para cada fase.

Modelagem

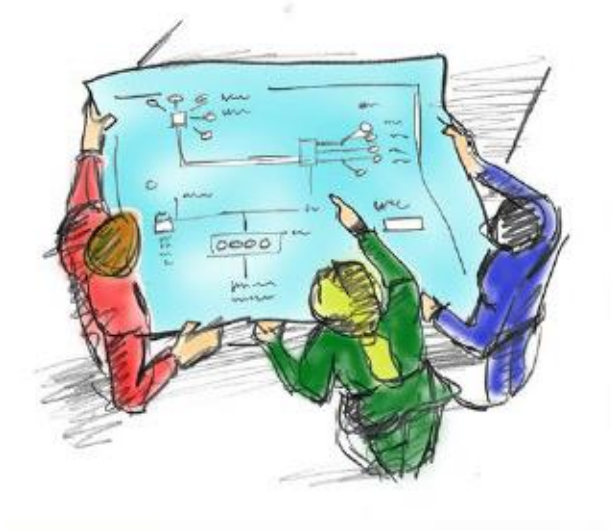
O que são modelos?

- Modelos descrevem um determinado sistema, muitas vezes de forma simplificada;
- Modelo de um processo de desenvolvimento:
  - É a especificação (documentada) de um processo de desenvolvimento de software que servirá de parâmetro para uso/especificação de um processo para uma equipe/projeto.

Modelos de Software

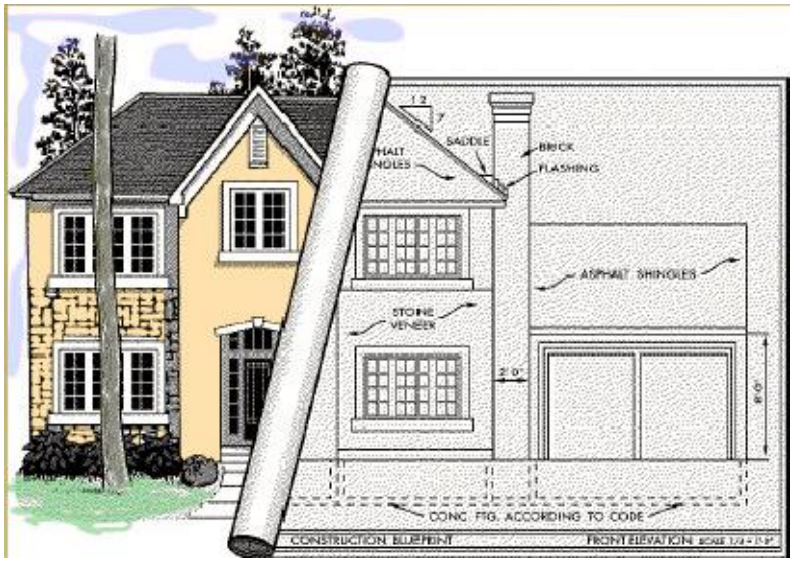
- Na construção de sistemas de software, assim como na construção de sistemas habitacionais, também há uma gradação de complexidade:

– A construção desses sistemas necessita de um planejamento inicial



Modelos de Software

- 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.



Razão para a Construção de Modelos

- Em 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

Diagramas e Documentação

- 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.

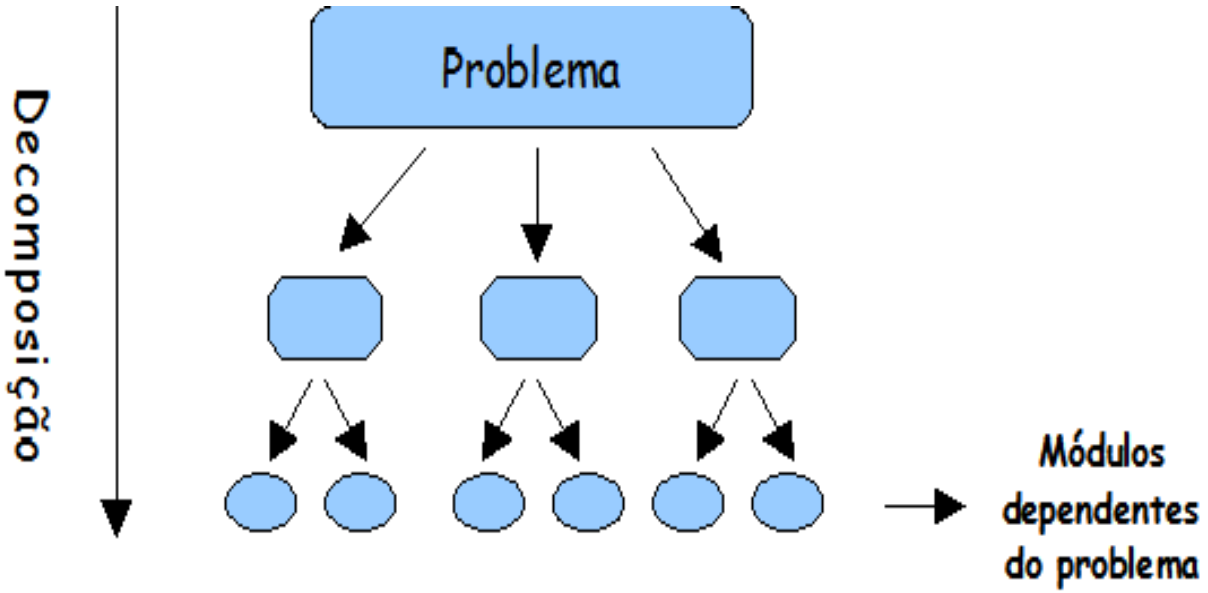
Paradigmas

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, projetado e construído.

Paradigma?

- A primeira abordagem usada para modelagem de sistemas de software foi o *paradigma estruturado*.

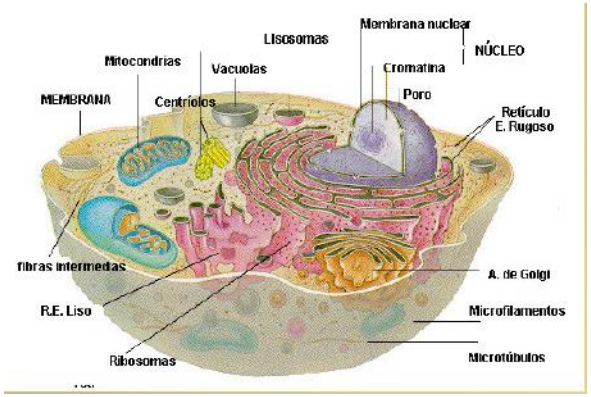


Paradigma da Orientação a Objetos

- Hoje em dia, praticamente suplantou o paradigma anterior, 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

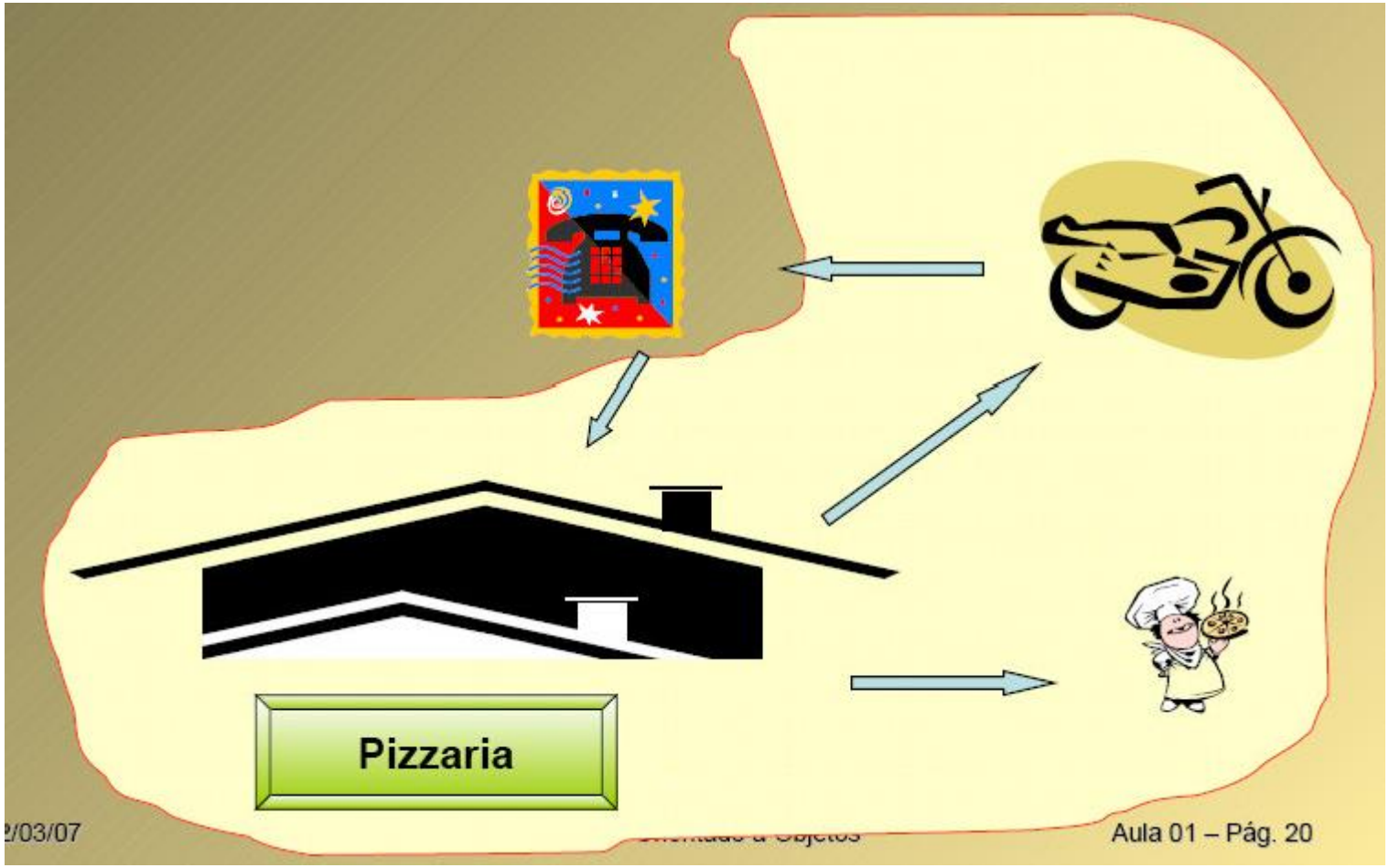
- Uma “célula” interage com outras células enviando mensagens para realizar um objetivo comum;
- Adicionalmente, cada célula se comportaria como uma unidade autônoma;
- De uma forma mais geral, Kay pensou em como construir um sistema de software a partir de agentes autônomos que interagem entre si.



Fundamentos da OO

- Através de sua analogia biológica, Alan Kay definiu os fundamentos 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.

Sistema de Software OO: uma analogia



Resumindo

- 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 complexa 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 de OO

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

Classes, Objetos e Mensagens

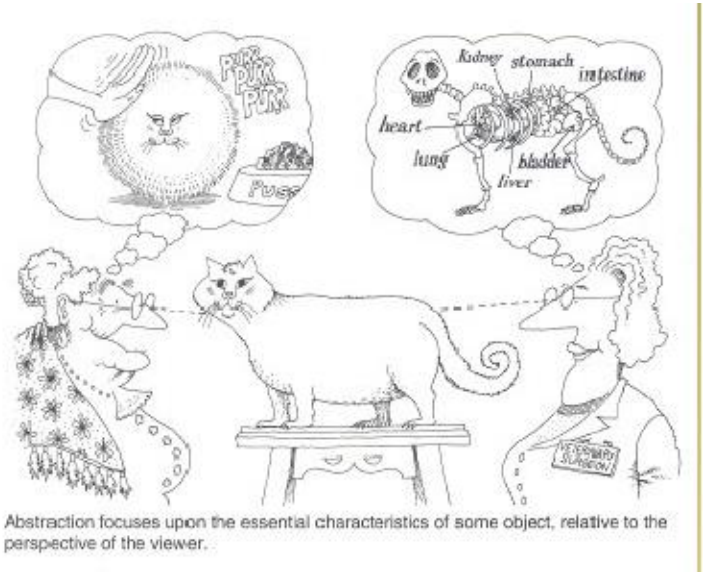
- O mundo real é formado de coisas;
- Na terminologia de orientação a objetos, estas coisas do mundo real são denominadas *objetos;*
- Seres humanos costumam agrupar os objetos para entendê-los;
- A descrição de um grupo de objetos é denominada *classe de objetos*, ou simplesmente de *classe.*

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

Como detectar propriedades relevantes?

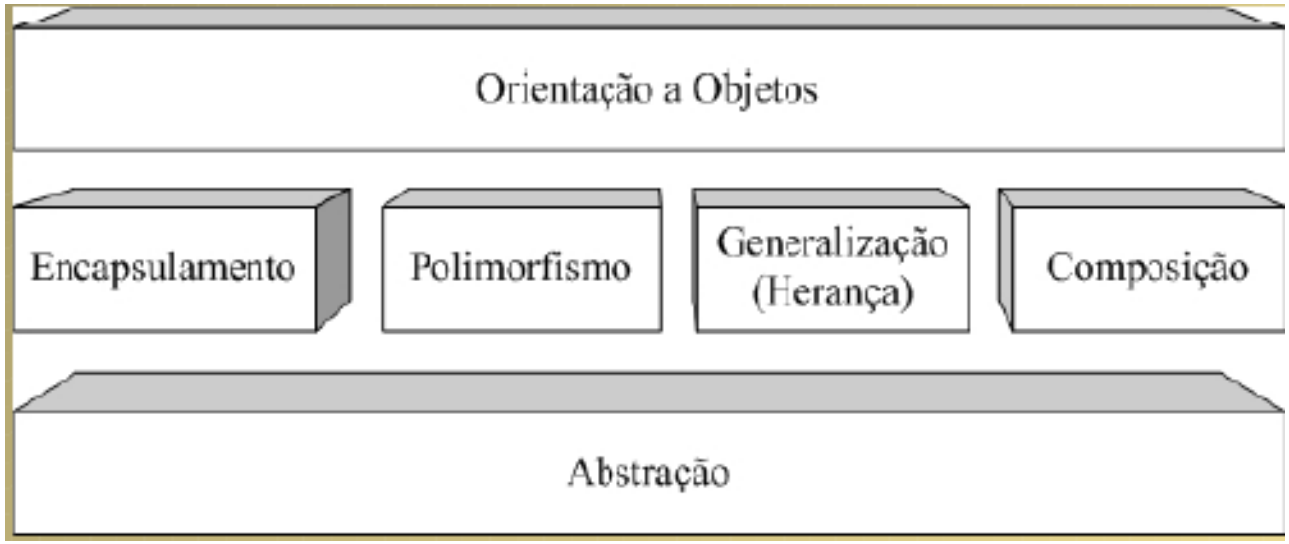
- Uma abstração é qualquer modelo que inclui os aspectos relevantes de alguma coisa, ao mesmo tempo em que ignora os menos importantes.
- *Abstração depende do observador.*



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

Abstração em OO

- 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)

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.

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

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

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

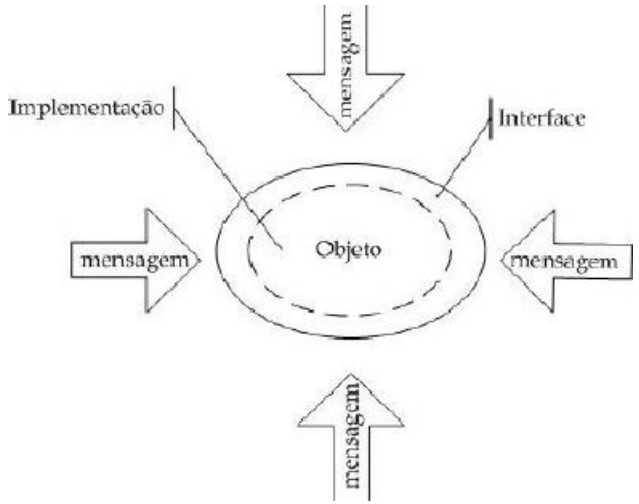
Encapsulamento

- 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

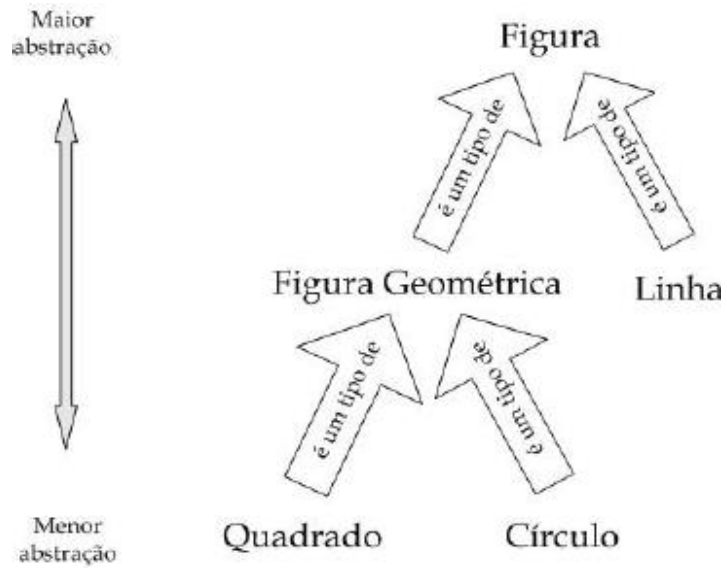
É a habilidade de objetos de classes diferentes responderem a mesma mensagem de diferentes maneiras.

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;



Entendido o Paradigma, como criar modelos?

- O rápido crescimento da capacidade computacional das máquinas (lei de moore) 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 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.

Entendido o Paradigma, como criar modelos?

- 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.

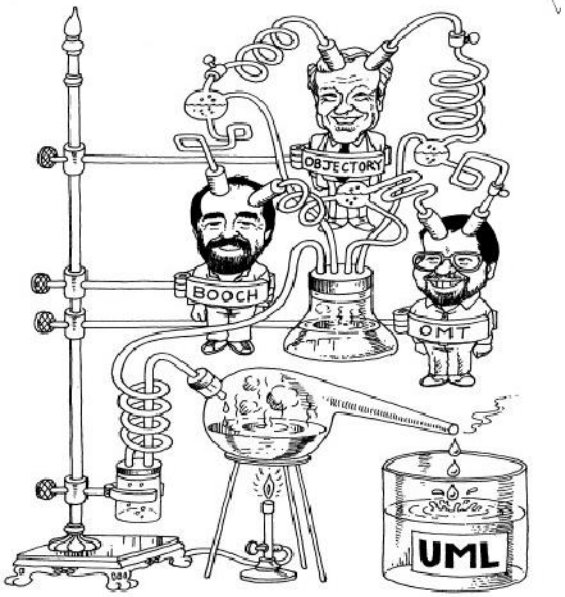
Entendido o Paradigma, como criar modelos?

- 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”.

UML

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

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

Diagramas da UML

- 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.

Referências

- JACOBSON, I.; BOOCH, G. and RUMBAUGH, J. *The Unified Software Development Process*. Readding, MA.: Addison-Wesley, 1999,
- LARMAN, C. *Utilizando UML e Padrões: uma introdução á análise e ao projeto orientados a objetos e ao Processo Unificado*. 2a edição - Porto Alegre: Bookman, 2004.
- ALLEIXO, F. *Notas de aula da disciplina de Análise e Projeto Orientado a Objeto,* CEFET/RN, 2007.
- LEITE, J.C. *Notas de aula da disciplina de Engenharia de Software, UFRN, 2006.*