PCS3413

Engenharia de Software e Banco de Dados

Aula 1

Escola Politécnica da Universidade de São Paulo

Bibliografia

[1] BEZERRA, E.; Princípios de Análise e Projeto de Sistemas com UML. 3ª Edição. Editora Campus, Rio de Janeiro, 2015 .

[2] Wazlawick, R.S.; Engenharia de Software – conceitos e práticas. Ed. Elsevier e Campus. 2013.

[3] PRESSMAN, R. S. Engenharia de software: Uma Abordagem Profissional. 7ª Edição. Porto Alegre: AMGH, 2011.

[4] HEUSER, C. A.; Projeto de Banco de Dados; 6ª edição, Editora Boookman, 2009.

[5] SILBERSCHATZ, A.; KORTH, H.F.; SUDARSHAN, S.; Sistema de Banco de Dados. 6ª edição. Editora Campus, 2012

[6] SOMMERVILLE, I. Engenharia de Software. 9ª Edição. Pearson Prentice-Hall. 2011.

Avaliação

- Provas: P1 e P2 – semana de provas

- Média final = (2 * MP + ME) / 3
  - ✓ MP =(P1 + P2)/2
  - ✓ ME (média dos exercícios)

- Slides de aula carregados no STOA: https://edisciplinas.usp.br/course/
- exercícios em grupo: 4 integrantes
  - ✓ um por grupo.
- Exercícios
  - ✓ Os exercícios feitos e entregues na aula.
  - ✓ membro ausente – nota zero no exercício

Aula	Data	Conteúdo
1	27/02	Apresentação da disciplina. Engenharia de Software e Banco de Dados: definições, projeto de software. Conceitos de OO e sistemas OO. UML e Visões.
2	01/02	Requisitos: funcionais e não funcionais. Modelos de Casos de Uso. Exemplo de Modelos de Caso de Uso.
3	6/03	Exercício 1 – identificação de casos de uso a partir de uma especificação textual e laboração de modelos de casos de uso (diagrama e descrição)
4	08/03	Conceito de Classes. Representação de classe com UML. Exemplos de modelos de classes. Exemplo de identificação de classes a partir da descrição de CU.
5	13/03	Correção e comentários sobre o exercício 1. Exercício 2 – elaboração do Diagrama de Classes para os CU descritos no Exercício .
6	15/03	Correção e Comentários sobre os exercícios.
7	20/03	Categorias de Objeto (Fronteira, Controle e Entidade) e Divisão de esponsabilidades. Diagrama de Sequência e representação UML. Exemplo de Diagrama de sequência e consistência com CU e Diagrama de Classes
8	22/03	Exercício 3 – Elaboração de Diagrama de Sequência para cenários de CU do exercício e 2.
9	03/04	Diagrama de Transição de Estado (DTE) e representação UML. Exemplo de DTE
10	05/04	Conceitos de Sistema de Gerenciamento de Banco de Dados (SGBD). Modelos de Dados Modelo Entidade-Relacionamento (MER)
11	10/04	MER – continuação Exemplo de um Modelo Entidade-Relacionamento. Diagrama Entidade Relacionamento (DER) X Diagrama de Classes.
12	12/04	Exercício 4 – elaboração do DER
13	17 e 19/04	Semana de Prova – P1
	24/04	Correção e comentários sobre o exercício 4. Comentário sobre a P1. Características do Modelo Relacional. Transformação do MER para Modelo Relacional.
14	26/04	Exercício 5 – elaboração do modelo lógico para o diagrama do exercício 4.

15	03/05	Correção e comentários sobre o exercício 5. SQL – manipulação de dados Álgebra Relacional – principais operadores.
16	08/05	Exercício 6 – uso de um SGBD com uma base pré-criada para especificação de uma expressão SQL para um conjunto de consultas sobre a base de dados.
18	08/05	continuação
19	15/05	SQL – definição de dados. Aspectos de um SGBD – transação, concorrência.
20	17/05	Procedimentos e Views em SGBDs
21	22/05	Exercício 7 – SQL views, transação, concorrência
22	24/05	Conceitos de Arquitetura: camadas. Estilos de Arquitetura Pattern e Requisitos não funcionais. Persistência. Diagrama de Implantação (UML)
23	29/05	Testes de Software: objetivo, testes de caixa branca e testes de caixa preta. Tipos de testes: unidade, testes de integração, teste de stress.
24	05/06	Modelos de Processos: Cascata, Espiral, Prototipação, Unificado
	07/06	Jogo Brasil
25	12/06	Modelo de Processos: XP, Scrum
26	14/06	Aspectos Engenharia de Requisitos. Desenvolvimento orientado a serviço (webservers) Aspectos de Banco de dados NoSQL
28	19 e 21/06	Semana de Prova – P2
29	26 e 28/06	Sub

Engenharia de Software

Engenharia de Software????

Software???

Sistema???

Qual a diferença?

```
public class Turma  {
      private  Set<OfertaDisciplina> ofertasDisciplina = new
      HashSet ();
      public  Turma ( ) {
      }
      public  void adicionarOfertaDisciplina(OfertaDisciplina
      oferta) {
            this.ofertasDisciplina.add(oferta);
      }
      public  boolean remove.OfertaDisciplina(OfertaDisciplina
      oferta)  {
            return this.ofertasDisciplina.remove(oferta);
      }
      private  Set  get OfertasDisciplina ( )    {
            return  Collections.unmodifiableSet
            (this.ofertasDisciplina);
      }
}
```

Engenharia de Software????

Software???

Sistema???

Controle Acadêmico

Engenharia de Software????

Pessoas

constroem,
definem,
disponibilizam
e acessam

Documentação

Software???

Sistema???

Hardware

Serviços

Serviços

Software

Rede

Definição de Engenharia de Software

Termo	Definição
Sistema	1. Conjunto de partes que formam um todo complexo ou unitário. 2. Grupo de corpos celestes associados e agindo segundo determinadas leis naturais. 3. Conjunto formado por um ou mais computadores, seus periféricos e programas utilizados. ... (Michaelis)
Software	Programas de Computador e documentação associada. Os produtos de software podem ser desenvolvidos para um cliente específico ou para um mercado geral
Engenharia de Software	É uma disciplina de engenharia relacionada a todos os aspectos de produção de software.

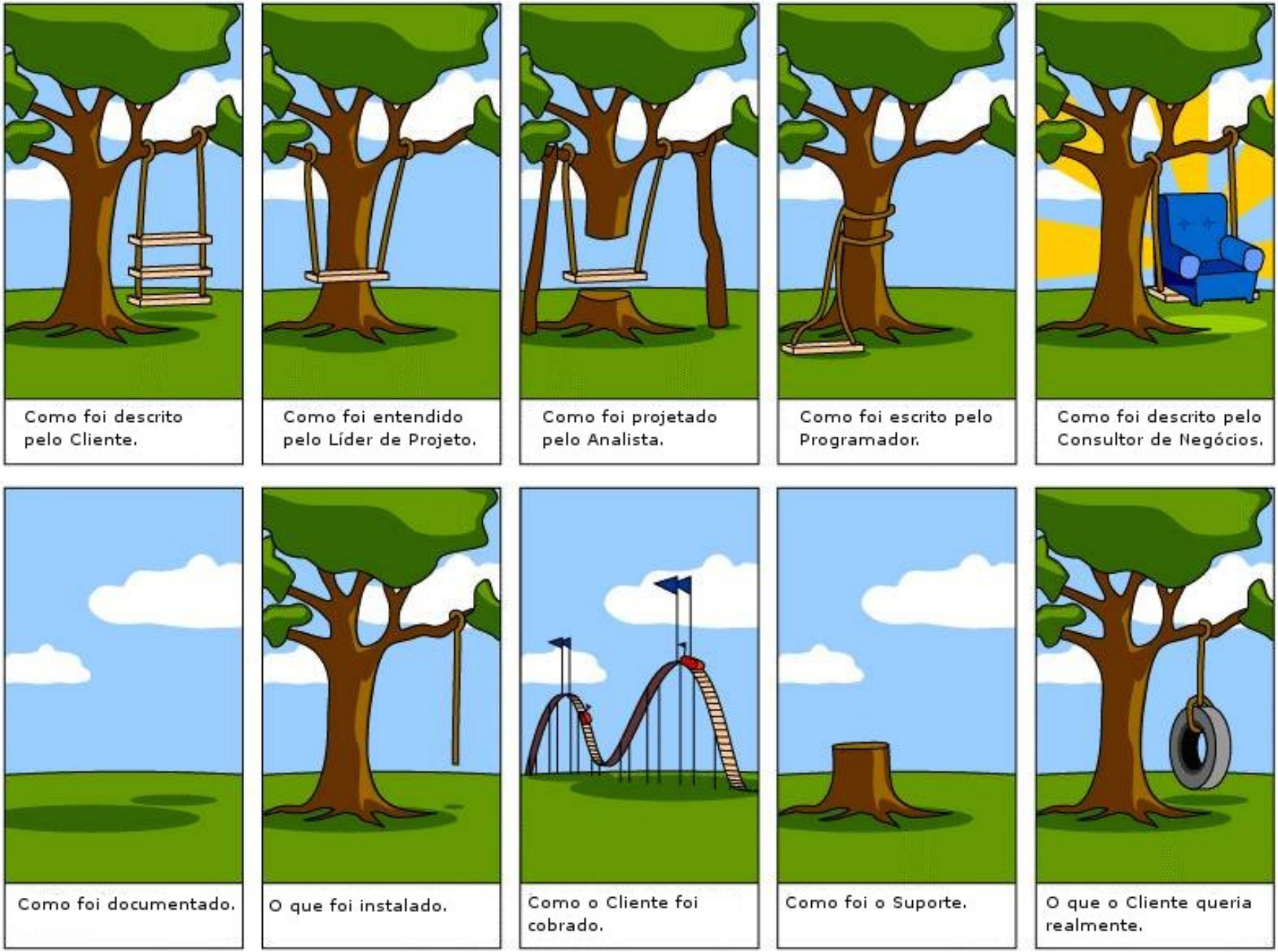
Engenharia de Software

“ O estabelecimento e uso de sólidos princípios de engenharia para que se possa **obter** economicamente um **software** que seja confiável e que funcione eficientemente em máquinas reais”

Fritz Bauer

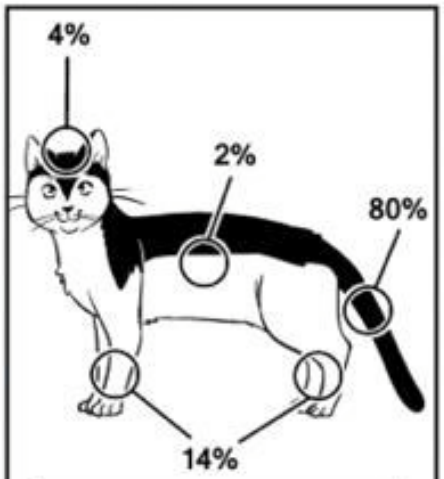
Para se construir algo é preciso conhecer o que se deseja

REQUISITOS

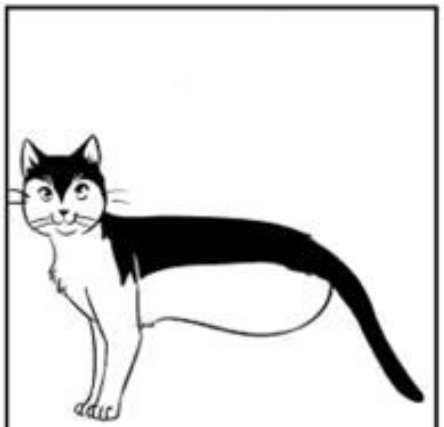

Como foi descrito pelo Cliente.
Como foi entendido pelo Líder de Projeto.
Como foi projetado pelo Analista.
Como foi escrito pelo Programador.
Como foi descrito pelo Consultor de Negócios.
Como foi documentado.
O que foi instalado.
Como o Cliente foi cobrado.
Como foi o Suporte.
O que o Cliente queria realmente.

Richard's guide to software development

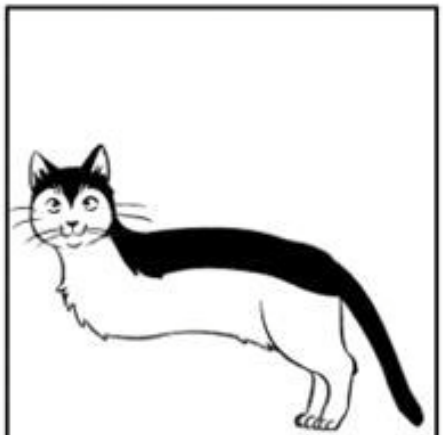
HOW THE SOFTWARE IS DESIGNED.



HOW MUCH TIME HAS TO BE SPENT ON EACH PART.



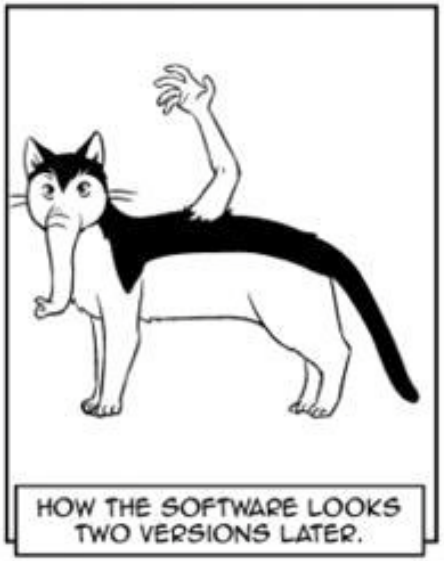
HOW THE SOFTWARE LOOKS BEFORE THE BETA TEST.



HOW THE SOFTWARE LOOKS AFTER THE BETA TEST.

HOW THE SOFTWARE IS ADVERTISED.

WHAT THE CUSTOMER REALLY WANTED.



HOW THE SOFTWARE LOOKS TWO VERSIONS LATER.

Sandra and Woo by Oliver Knörzer (writer) and Powree (artist) – www.sandraandwoo.com

Engenharia de Software

Métodos, ferramentas, procedimentos e

processos

como fazer

- sequência de aplicação das atividades dos métodos
- produtos a entregar
- controles para assegurar qualidade e coordenar mudanças

- planejamento e estimativa de projeto
- análise de requisitos
- projeto
- codificação
- Teste e manutenção

EVOLUÇÃO E CRISE DO SOFTWARE

- Produtos de software amplamente distribuídos

  → acrescentaram milhares de linhas de código (requisitos de usuário)

- Estimativas de prazo e de custo imprecisas

  → Tempo para coletar dados sobre o processo de desenvolvimento de software

    → falta de indicação de medidas de produtividade para avaliar novas ferramentas, métodos

- Qualidade de software em geral é menos que adequada
  - Insatisfação do cliente com o produto entregue
    - ✓ comunicação entre cliente e desenvolvedor é fraca;
    - ✓ alta taxa de erros:
  - qualidade do software suspeita
    - ✓ testes de software sistemáticos e completos
- Produtividade das pessoas da área de software não tem acompanhado a demanda por serviço

- É preciso entender os requisitos
  - ➢ que o cliente realmente deseja, o que deve ser construído
  - ➢ é preciso entender o cliente e seus reais desejos …

- É preciso planejar adequadamente
  - quanto tempo gastar em cada tarefa?
    - ✓ como foi feito em projetos passados?
    - ✓ que aconteceu em projetos passados?
    - ✓ quanto tempo se gastou em cada atividade?
  - que método usar?
    - ✓ ferramentas
    - ✓ linguagens

- É preciso entender o cliente, mas…
  - **Mas a comunicação nem sempre é fácil!!**

DILBERT®

BY SCOTT ADAMS

© Scott Adams, Inc./Dist. by UFS, Inc.

Paradigma de Orientação a Objetos

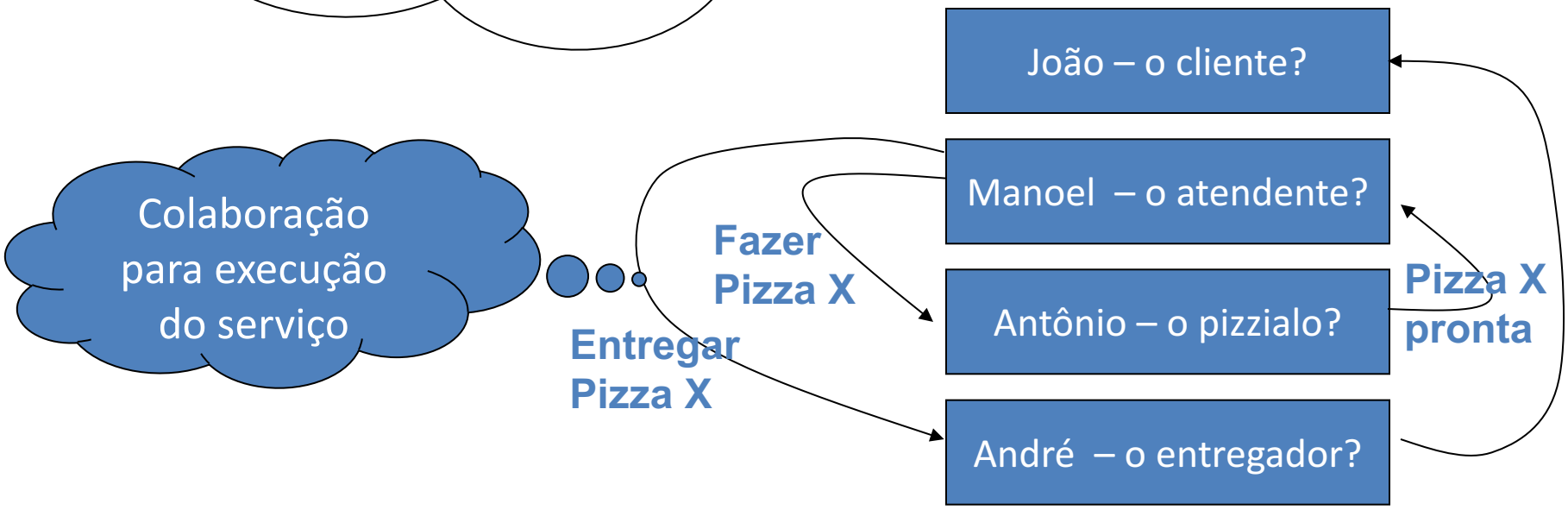
- Um paradigma é uma forma de abordar um problema.
- Então: usa-se o conceito de objetos para desenvolver sistemas de software orientado a objetos.

❑ Objetos:
- Qualquer coisa é um objeto.
- Objetos realizam tarefas por meio de requisições 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.

Objetos realizam tarefas por meio de requisições de serviços a outros objetos.

COLABORAÇÃO ENTRE OBJETOS PERMITE QUE A TAREFA COMPUTACIONAL SEJA REALIZADA

O que acontece quando alguém pede uma pizza por telefone?
Que personagens estão envolvidos e quais atividades executam desde o recebimento do pedido, a preparação da pizza e sua efetiva entrega no endereço solicitado?



- O paradigma de orientação a objetos visualiza um sistema de software como uma coleção de agentes interconectados chamados objetos.

- Um sistema de software orientado a objetos consiste em objetos em colaboração com o objetivo de realizar as funcionalidades desse sistema. Cada objeto é responsável por tarefas específicas. É graças a cooperação entre objetos que a computação do sistema se desenvolve.

Eduardo Bezerra – Princípios da Análise e Projeto de Sistemas com UML

Objetos

- Qualquer coisa do mundo real – concreta ou abstrata

Antônio

André

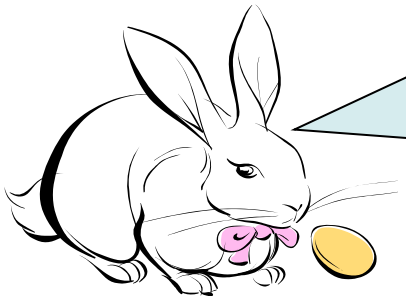
Manoel

C/C

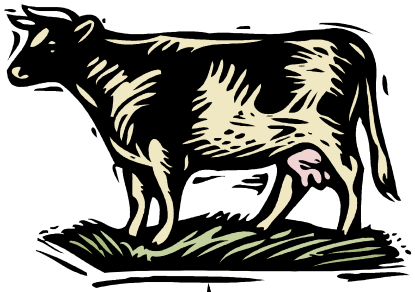
- Cada objeto pertence a uma determinada classe. Uma classe agrupa objetos similares.
- A classe é um repositório para comportamento associado ao objeto.

Classes

É a idéia. Reune características e comportamentos.



Coellho: Quatro patas, orelhas pontudas, pelo espesso, etc.



Pessoa: sexo, altura, nome, idade, etc.

Professor: área de conhecimento, anos de docência, etc. Ensina

Vaca: quatro patas, mamífero, tem rabo, etc. Produz leite

Estrutura – atributos
propriedades dos objetos que compõem a classe – identificam o estado de um objetos.

Atributos

Nome:

Cargo:

Data de Admissão:

Salário:

...

Comportamento -
Modificam o estado ou
- Prestam serviços a outros objetos

Classe Pizzaiolo

Operações

Fazer Pizza

lista Compra de Ingredientes

...

Classe Pizzaiolo

cada objeto pertence a uma classe

Objeto é uma ocorrência (instância) de uma classe.

Objeto Antônio

Numero Funcional: 00962
Nome: Antônio Marins
Data Nascimento: 01/03/1970
Salário: 10.000

uma classe é uma abstração de coisas do mundo real
uma classe representa uma abstração de características relevantes do mundo real!

Operação

- ação que o objeto sabe realizar
- objetos não executam suas operações aleatoriamente
- um objeto realiza uma de suas operações a partir de um estímulo enviado a ele.

Objetos realizam tarefas por meio de requisições de serviços a outros objetos.

Mensagem

- estímulo para que o objeto realize uma de suas operações
- um objeto envia uma mensagem a outro objeto solicitando que determinada operação seja executadas

Estado de um objeto

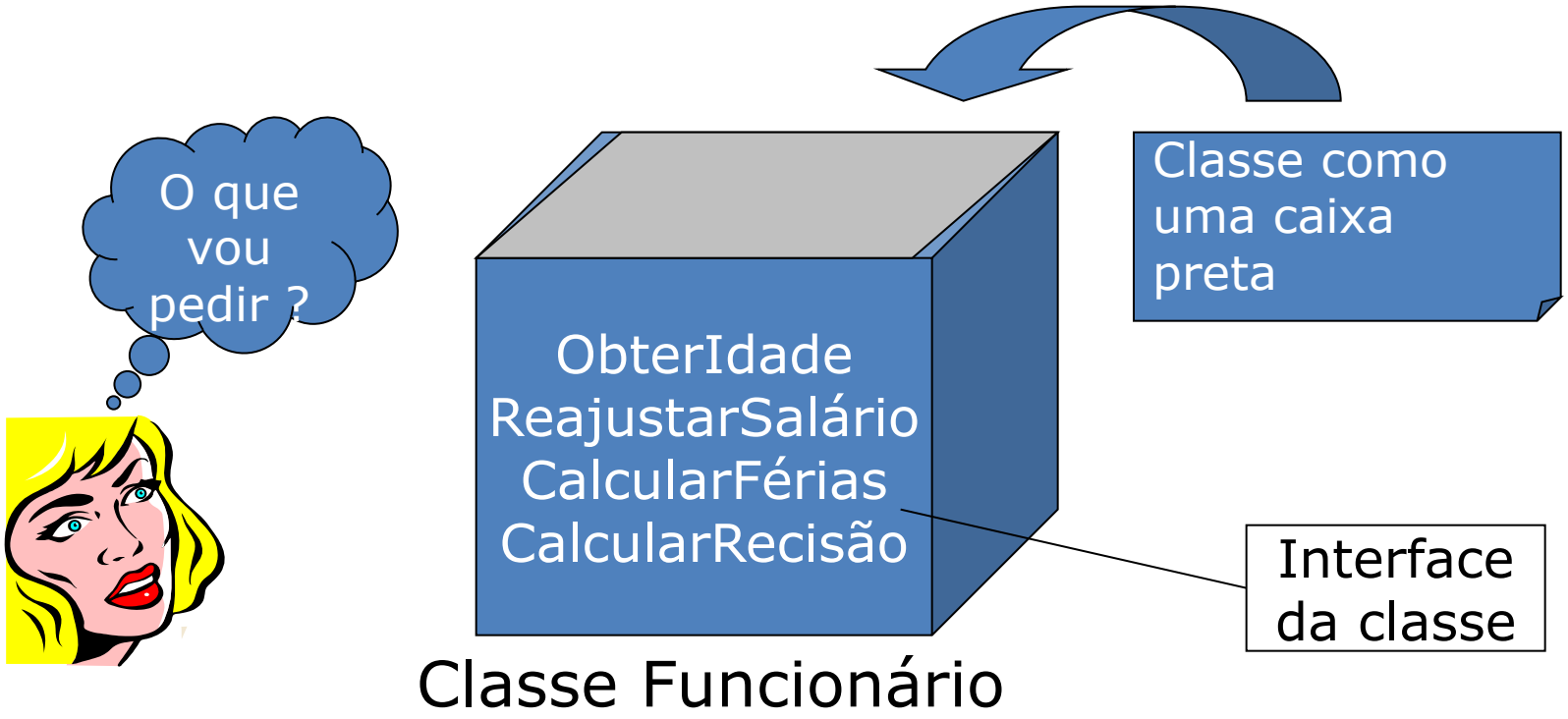
- conjunto de valores de seus atributos em determinado momento

Abstração

- processo mental pelo qual destacamos as características principais de alguma coisa ou conceito. Nesse processo aspectos menos importantes ou irrelevantes são ignorados.
- os aspectos importantes variam com o contexto em que a “coisa” é observada.
- vantagem da abstração: gerenciar a complexidade com que se precisa lidar.

Encapsulamento

- A utilização de um sistema não depende de sua implementação e sim de sua interface



Encapsulamento

- Objetos encapsulam sua estrutura e seu comportamento:
  - Objetos oferecem serviços a outros objetos através de sua interface.
  - Uma interface declara para outros objetos quais operações o objeto pode executar e quais informações são necessárias para a execução e o que se obtém como resultado da operação.
  - A implementação (MÉTODO) da operação e a estrutura interna não são acessíveis aos outros objetos.
  - Permite que a implementação ou a estrutura interna do objeto seja alterada sem que seja necessário a alteração de outros objetos – o serviço oferecido pelo objeto (OPERAÇÃO e seus parâmetros) não foram alterados.

Polimorfismo, Generalização e Composição

- Polimorfismo
  - capacidade de abstrair diferentes implementações em uma única interface.
- Generalização
  - forma de abstração que permite que classes sejam organizadas em hierarquias que reuniem características comuns a classes num determindado nível da hierarquia. Classes em níveis mais baixos herdam as características das classes em nível mais alto.
- Composição
  - abstração que mostra que **objetos** que **são compostos por outros objetos**.
  - também se definem herarquias de composição
- Papel dos profissionais é identificar a correta representação quando precisar agrupar classes em hierarquias.

Desenvolvimento de software

- Desenvolver software não envolve só uma linguagem de programação
  - Métodos, Processos, Ferramentas, Modelos, etc.

- Um aspecto importante é a *modelagem de sistema de software*
  - modelo é representação idealizada de um sistema de software a ser construído
  - diferentes modelos permitem idealizar o sistema sob diferentes perspectivas. Cada modelo permite focar num aspecto importante

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 várias perspectivas diferentes e complementares.

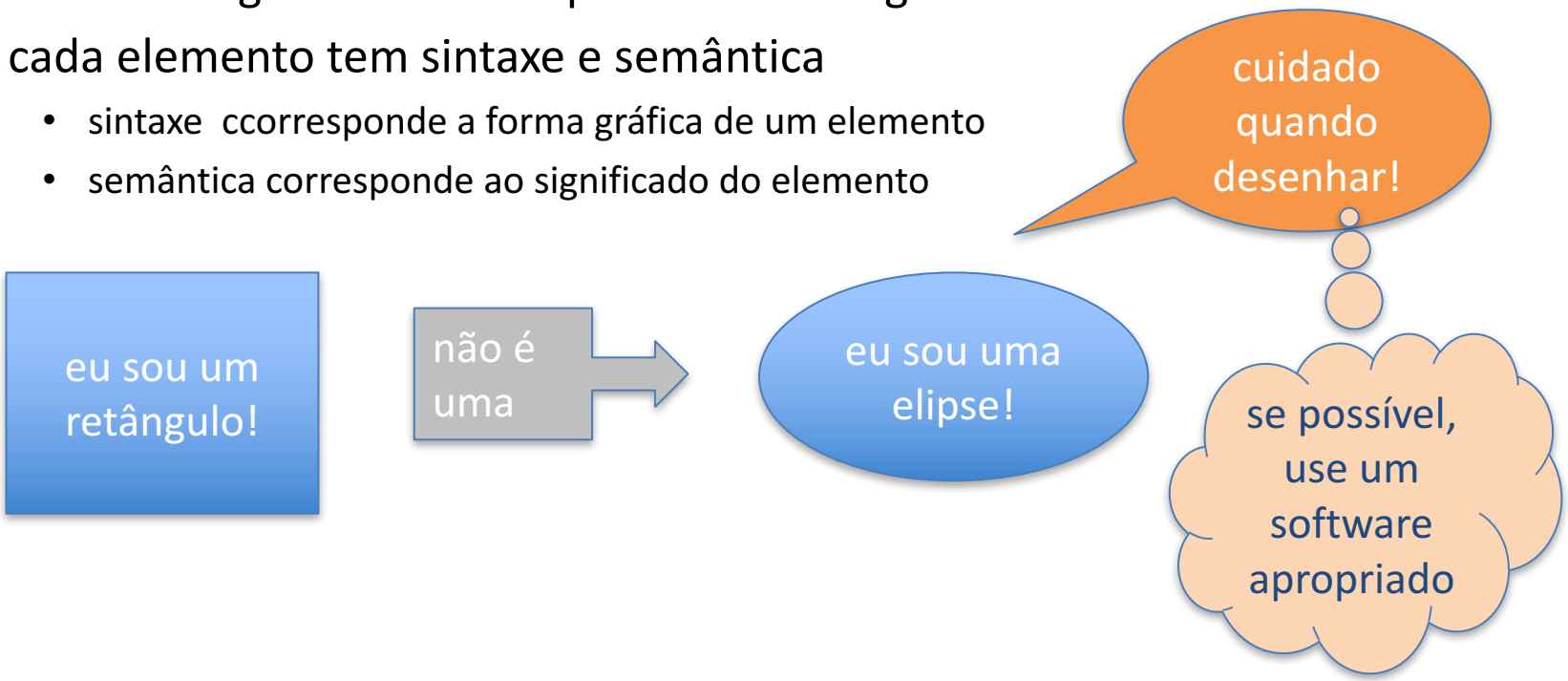
Bezerra, E.

Razões para construção de modelos

- Gerenciamento de complexidade
- comunicação entre pessoas envolvidas
- Redução de custos no desenvolvimento
- Previsão de comportamento futuro do sistema

UML – Unified Modeling Language

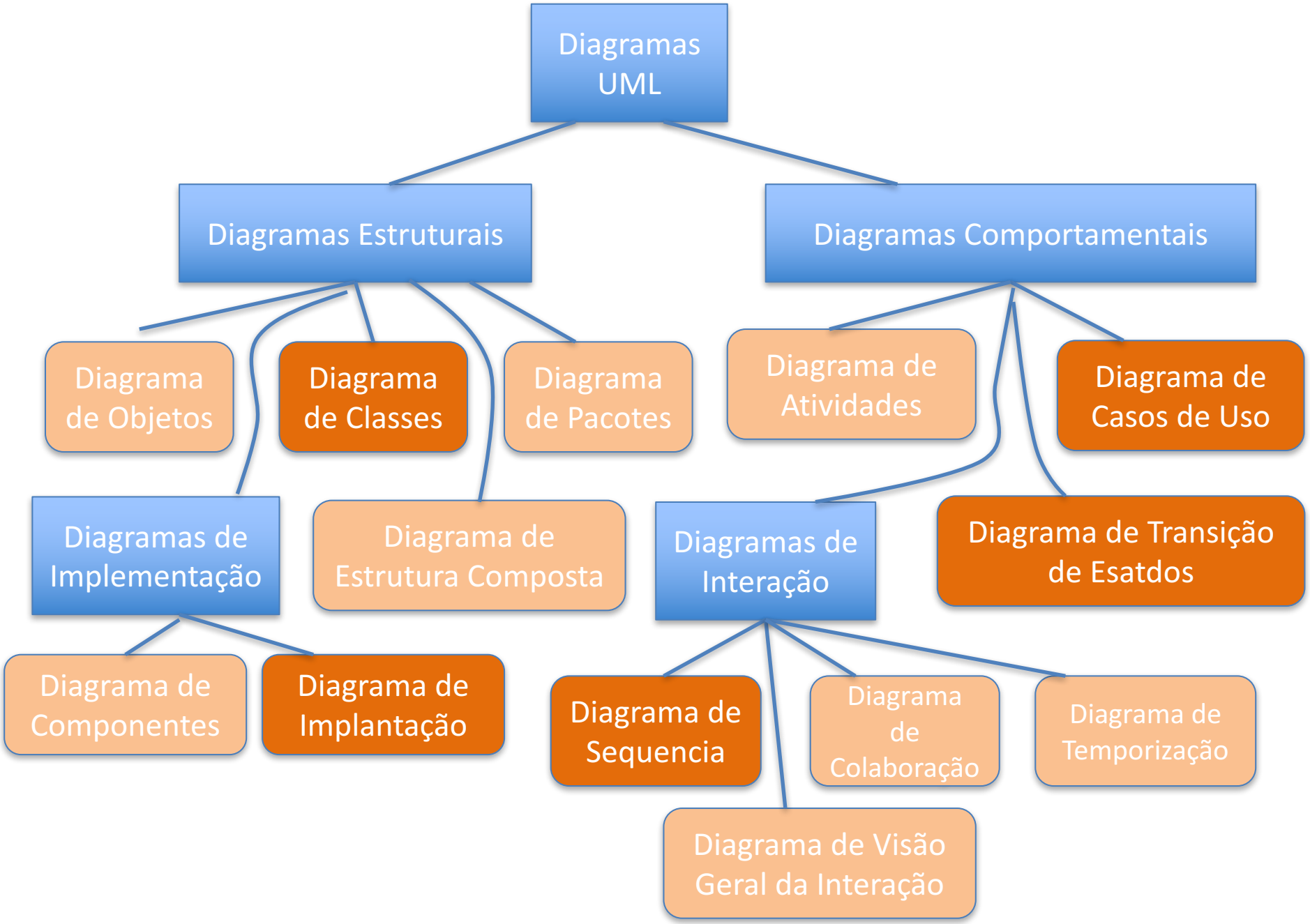
- "os três amigos"
  - Grady **Booch**; James **Rumbaugh** e Ivar **Jacobson**
- linguagem visual para modelar sistemas orientados a objetos
  - elementos gráficos usados para definir Diagramas
  - cada elemento tem sintaxe e semântica
    - sintaxe ccorresponde a forma gráfica de um elemento
    - semântica corresponde ao significado do elemento



- UML é uma especificação da OMG e atualmente está na versão 2.5
- especificação completa: http:// www.uml.org

Diagramas da UML

- composto por elementos gráficos
- documentos podem ser textuais ou gráficos
- terminologia UML: documentos são **artefatos de software** ou **artefatos**
- **A UML é indenpendente de linguagem de programação e de processo**
  - diagramas podem ser utilizados para gerar artefatos utilizados para implementar em qualquer linguagem e em qualquer processo


Diagramas UML
Diagramas Estruturais
Diagramas Comportamentais
Diagrama de Objetos
Diagrama de Classes
Diagrama de Pacotes
Diagrama de Atividades
Diagrama de Casos de Uso
Diagramas de Implementação
Diagrama de Estrutura Composta
Diagramas de Interação
Diagrama de Transição de Esatdos
Diagrama de Componentes
Diagrama de Implantação
Diagrama de Sequencia
Diagrama de Colaboração
Diagrama de Temporização
Diagrama de Visão Geral da Interação

Visões de um Sistema de Software

- o desenvolvimento de um sistema complexo exige que seus desenvolvedores possam examiná-lo e estudá-lo sob diferentes perspectivas.
- UML: 5 visões independentes
  - Visão de CU: descreve o sistema sob a perspectiva externa de interações entre o sistema e agentes externos ao sistema
  - Visão de Projeto: características estruturais e comportamentais do sistema que viabilizam as funcionalidades visíveis externamente.
  - Visão de Processo: características de concorrência (paralelismo), sincronização e desempenho do sistema
  - Visão de Implementação: gerenciamento de versões construídas pelo agrupamento de módulos (componentes) e subsistemas.
  - Visão de Implantação: distribuição física dos subsistemas e a conexão entre eles.

características estruturais e comportamentais

versões e seus componentes

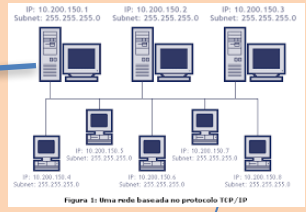
Visão de Projeto

Visão de Implementação

Visão de Casos de Uso

Visão de Processo

Visão de Implantação

interação entre agentes e o sistema

características não funcionais