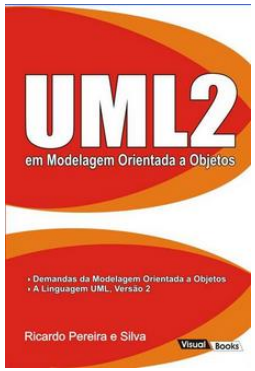
Engenharia de Software

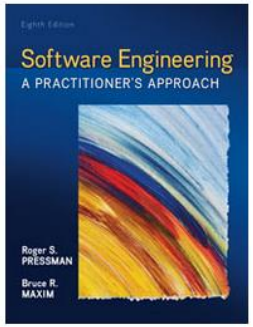
Unidade 6 - Modelagem de Software

Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP

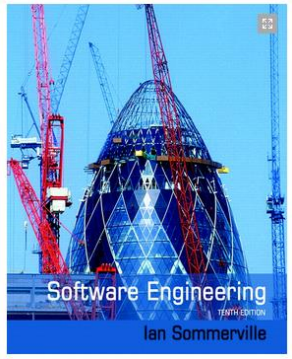
Bibliografia

- **Software Engineering – A Practitioner's Approach – Roger S. Pressman – Eight Edition – 2014**
- **Software Engineering – Ian Sommerville – 10th edition - 2015**
- Engenharia de Software – Uma abordagem profissional – Roger Pressman - McGraw Hill, Sétima Edição - 2011
- Engenharia de Software – Ian Sommerville – Nona Edição – Addison Wesley, 2007
- UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007
- Como modelar com UML 2 – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2009

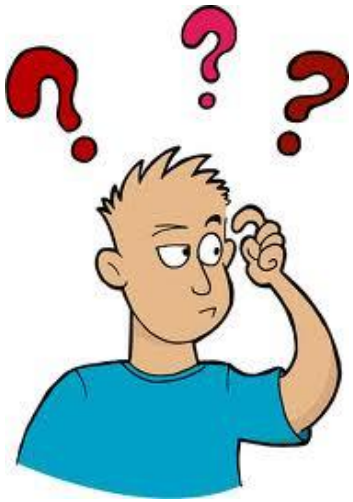




Software Engineering: A Practitioner's Approach, 8/e

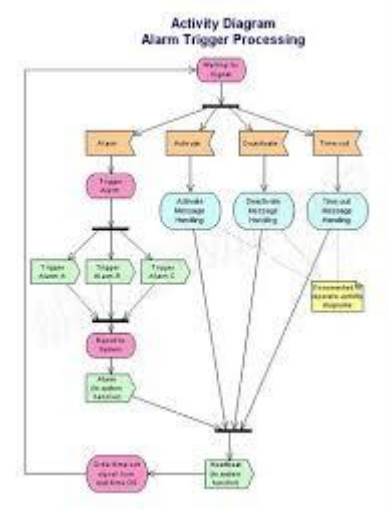


O que é Modelagem do Software ?



Modelagem de Software

- Descrição diagramática (por meio de diagramas) de um software a ser implementado em uma Linguagem de Programação.
- A modelagem representa o mesmo conteúdo do código, porém em outro formato.



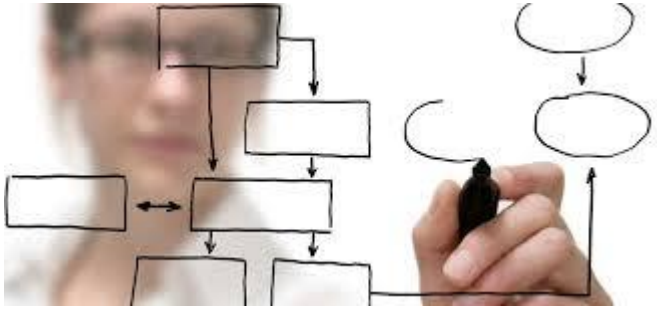
≡

```
class Teste{

        private static long fibo(long num){
                if (num == 1 || num == 2) return (1);
                else return (fibo(num-1) + fibo(num-2));
        }

        public static void main(String args[]){
                int num = Integer.parseInt(args[0]);
                long result = fibo(num);
                System.out.printf("Resultado: %d \n", result);
        }

}
```

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

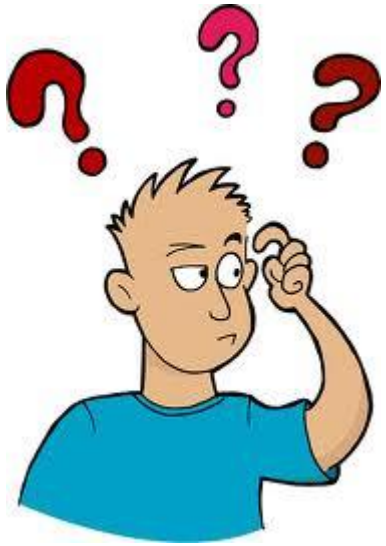
Tipos de Modelagem

- Modelagem **Prescritiva**: Antes do Código (Desenvolvimento)
- Modelagem **Descritiva**: Após o Código (Manutenção)



Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

Vé ! Mas não há esforço duplicado ?



Primeiro, modela-se !
Em seguida, codifica-se !

Não seria fazer a mesma coisa, duas vezes ?

Esforço Dobrado ?

- Não, software é produto altamente manutenível !
- Não, fazer a manutenção diretamente no código é muito complexo ! ! !
- Não, software em geral é desenvolvido em equipes ! ! !
- Não, modelagem auxilia na concepção da solução (**Não** é apenas **documentação**)

Exemplo das Engenharias

- Toda grande obra de Engenharia exige um grande esforço de planejamento (**Projeto**) antes da Construção.
- Exemplo: Ninguém constrói uma ponte sem antes efetuar um projeto.
- Ninguém constrói um edifício, iniciando diretamente no assentamento de tijolos (sem um planejamento prévio).

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

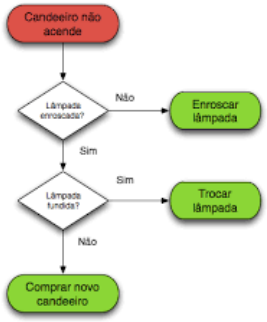
A Modelagem se aplica à qualquer situação ?

Modelagem de Software

- A Engenharia nos ajuda a responder essa questão.
- A construção de uma grande **barragem**, certamente exige um **projeto**.
- A construção de uma simples casinha de cachorro: pode dispensar o projeto, pois a implementação é muito simples (mãos-à-obra).

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

Complexidade de Software



- Associada ao tamanho do software a ser desenvolvido.
- Programas muito simples podem dispensar a modelagem (poucas linhas de código).
- **Alta Complexidade:**
  - ✓ Planejamento -> Modelagem Orientada a Objetos
  - ✓ Construção -> Codificação
- **Baixa Complexidade:** Possível codificação direta em algum ambiente de desenvolvimento.

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

Processo de Desenvolvimento de Software

- Inicia-se com a elaboração dos Requisitos do Software.
- O resultado da Engenharia de Requisitos deve ser um documento formal, contendo as Especificações dos Requisitos do Software.
- Para uma melhor compreensão do que será realmente construído, criam-se **modelos de software**.

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

O que é Modelo ?

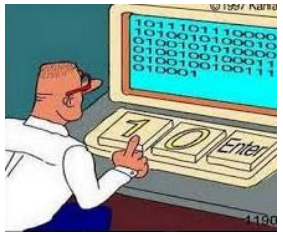
O termo Modelo

- Um modelo é uma imagem que abstrai da realidade ou que funciona como uma representação abstrata da realidade a ser criada.
- A modelagem pode ser aplicada a objetos materiais ou imateriais de uma realidade **existente** ou de uma realidade **a ser desenvolvida**.
- Um modelo é uma **representação abstrata** de uma realidade existente ou de uma realidade a ser criada. [Stachowiak, 1973]

Quais as vantagens da Modelagem ?

Modelagem - Vantagens

- A Modelagem não é materializada apenas na escrita do código, mas na fase de **concepção**.
- Com isso, obtém-se um software melhor estruturado (alta **manutenibilidade** e alta **reusabilidade**).
- Mais próxima da forma como as pessoas pensam.
- Não é natural "pensar-se" em código de Linguagem de Programação.



```
1  <?hh
2  class MyClass {
3    public function alpha(): int {
4      return 1;
5    }
6
7    public function beta(): string {
8      return 'hi test';
9    }
10 }
11
12 function f(MyClass $my_inst): string {
13   // Fix me!
14   return $my_inst->alpha();
15 }
```

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

Modelagem de Software – Vantagens

- Permitem uma **melhor compreensão** do que será realmente construído. [Pressman]
- Proporciona diferentes **visões** do software (visão de partes / visão do todo).
- Descreve os elementos **estruturais** do software. (modelagem **estrutural**)
- Descreve o software em **execução**. (modelagem **dinâmica**)
- Modelagem permite que se veja o software em vários níveis de abstração.
- **Codificação**: apresenta apenas visão em baixo nível de abstração.

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

Boas práticas de Modelagem

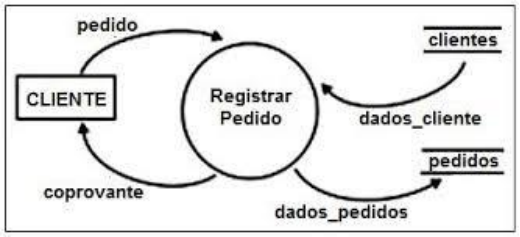
- ✓ O **objetivo** principal da equipe de software é **construir software** e não modelos;
- ✓ Não crie mais modelos do que se **necessita**;
- ✓ Esforce-se ao máximo para produzir modelos o mais **simples** possível;
- ✓ Construa modelos que facilitem **alterações**;
- ✓ Seja capaz de estabelecer um **propósito** claro para cada modelo;
- ✓ **Adapte** o modelo ao sistema a ser desenvolvido;
- ✓ Crie modelos **iterativos**.
- ✓ Sintaxe do modelo é secundária. O mais importante é transferir **conteúdo**;
- ✓ Obtenha o **feedback** o quanto antes (membros da equipe devem revisar o modelo).

Fonte: Engenharia de Software: Uma abordagem Profissional – Pressman R. S. – McGraw Hill, Sétima Edição,

Classes de Modelagem

- **Modelagem de Requisitos**. Representam os requisitos dos clientes, descrevendo o software em três domínios: Domínio da Informação, Domínio Funcional e Domínio Comportamental. (Também conhecido por Modelo de Análise).

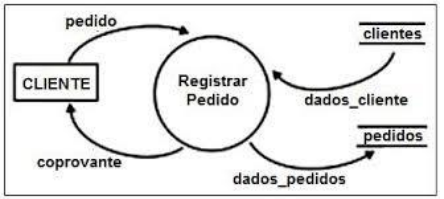
- **Modelagem de Projeto**: Representam características do software que auxiliam os desenvolvedores a construí-lo efetivamente: a arquitetura, a interface para o usuário e os detalhes quanto a componentes.



Fonte: Engenharia de Software: Uma abordagem Profissional – Pressman R. S. – McGraw Hill, Sétima Edição

Princípios da Modelagem de Requisitos

- Dados que **fluem** do sistema e dados **persistentes** devem ser compreendidos;
- As **funções** que o software desempenha devem ser definidas;
- O **comportamento** do software, como consequência de eventos externos, deve ser representado. Ex. alimentações fornecidas pelos usuários finais.
- Deve-se usar a estratégia "**Divisão-e-Conquista**", dividindo-se um problema grande e complexo em subproblemas até que cada um seja suficientemente compreendido.
- A análise (modelagem) de requisitos se inicia pela descrição do problema sob a perspectiva do usuário final. A "**essência**" **do problema** é descrita sem se levar em conta como a solução será implementada.



Fonte: Engenharia de Software: Uma abordagem Profissional – Pressman R. S. – McGraw Hill, Sétima Edição

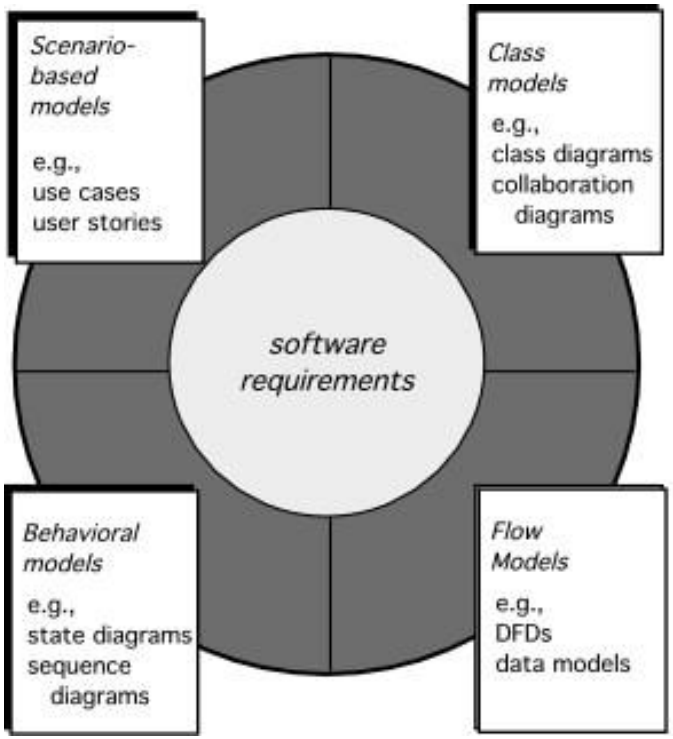
Princípios da Modelagem de Projeto

- A modelagem de projeto traduz a informação obtida na modelagem de requisitos em uma **solução computacional** (arquitetura, conjunto de subsistemas e componentes);
- O projeto deve iniciar com as **considerações arquitetônicas** do software a ser construído (espinha dorsal do sistema);
- O projeto dos **dados** é tão importante quanto o projeto das funções.
- **Interfaces** bem elaboradas (**usabilidade**) estão associadas à eficiência de processamento, simplicidade de projeto e auxiliam nos testes de validação.
- Funcionalidades do software devem ser **coesas** (focarem uma, e somente uma, função).
- Relacionamento entre componentes deve ser mantido tão baixo quanto possível (**acoplamento** fraco).
- Modelos devem ser de **fácil compreensão** aos desenvolvedores.
- Modelagem de projeto deve ser iterativa, visando obtenção do maior grau de simplicidade.

Fonte: Engenharia de Software: Uma abordagem Profissional – Pressman R. S. – McGraw Hill, Sétima Edição

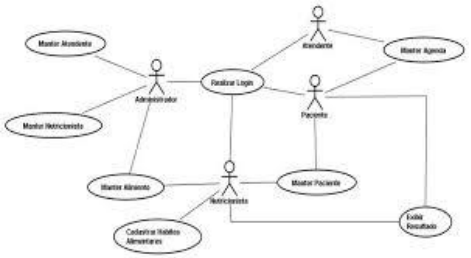
Elementos da Modelagem de Requisitos

- Elementos baseados em Cenários.
- Elementos baseados em Classes.
- Elementos Comportamentais.
- Elementos Orientados a Fluxos.



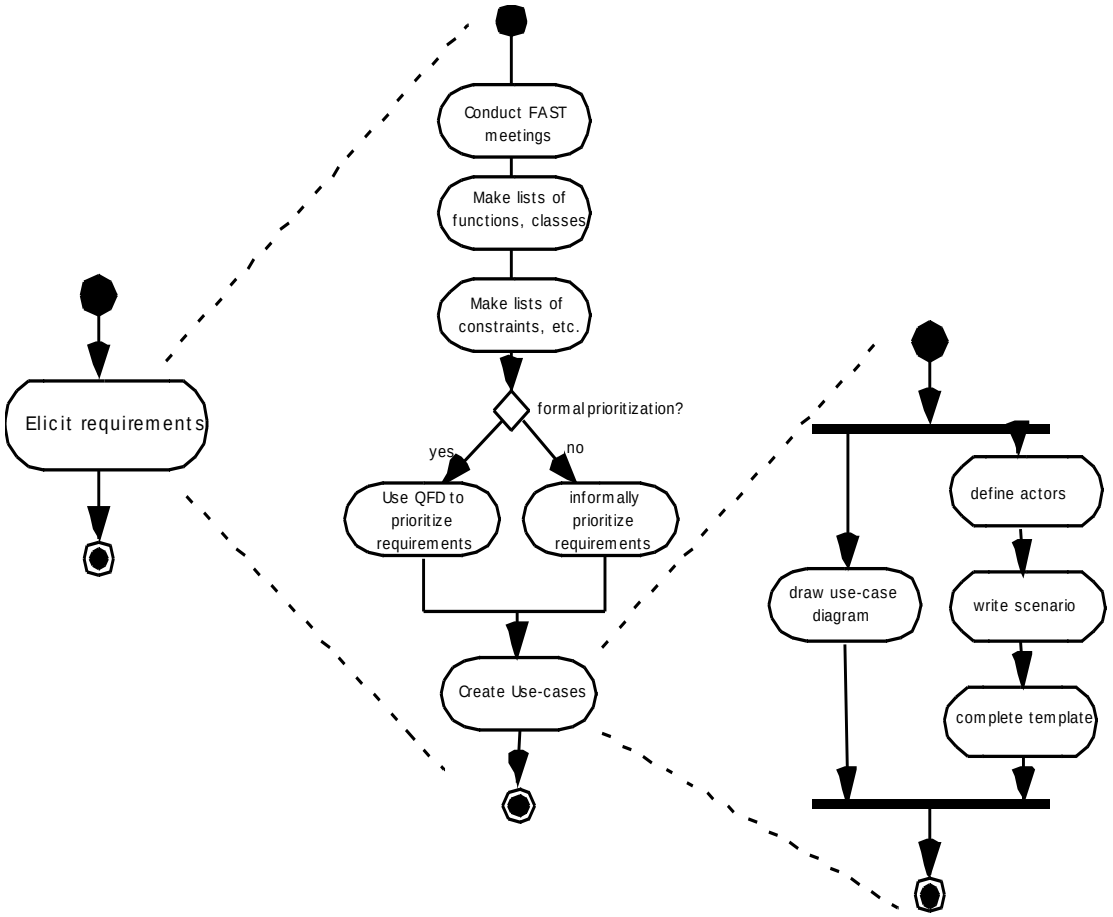
Modelagem de Requisitos baseada em Cenários

- O sistema é descrito sob o **ponto de vista do usuário**;
- Cria-se um conjunto de **cenários** que identifique um **roteiro** de uso para o sistema a ser construído. **Cenários** normalmente são chamados **CASOS de USO** (descrevem como o sistema será utilizado).
- Um caso de uso conta uma história sobre **como** um **usuário** (desempenhando uma série de papéis) **interage** com o **software**. Pode ser um texto narrativo ou uma representação esquemática. Independentemente do formato, um caso de uso **representa** o software o do ponto de vista do usuário.
- Em geral, é o **primeiro** modelo a ser desenvolvido. Serve como **base** para outros modelos da modelagem de requisitos.



Fonte: Engenharia de Software: Uma abordagem Profissional – Pressman R. S. – McGraw Hill, Sétima Edição

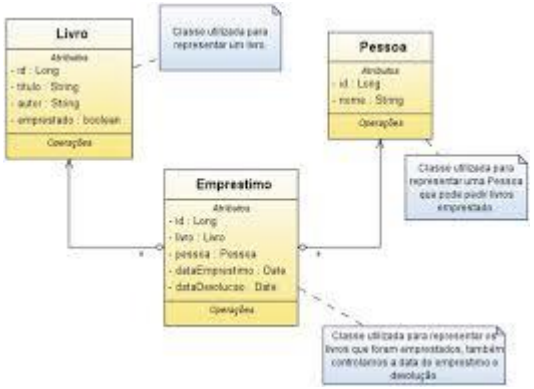
Modelagem de Requisitos baseada em Cenários



Fonte: Engenharia de Software: Uma abordagem Profissional – Pressman R. S. – McGraw Hill, Sétima Edição

Modelagem de Requisitos baseada em Classes

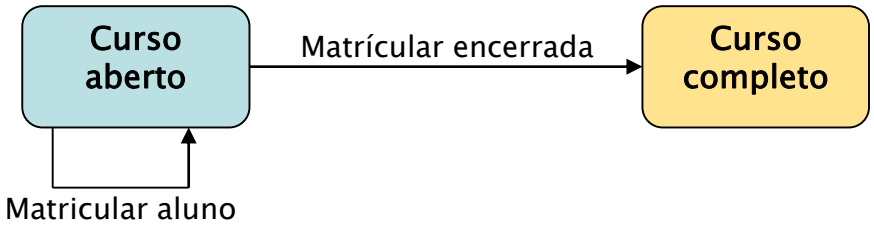
- Cada caso de uso implica em um conjunto de **objetos** manipulados à medida em que um ator (papel) interage com o software.
- Esses objetos são categorizados em **classes** – conjuntos de objetos que possuem atributos similares e comportamentos comuns.
- Um diagrama de classes **UML** pode ser utilizado para representar as classes e seus relacionamentos.



Fonte: Engenharia de Software: Uma abordagem Profissional – Pressman R. S. – McGraw Hill, Sétima Edição

Modelagem Comportamental de Requisitos

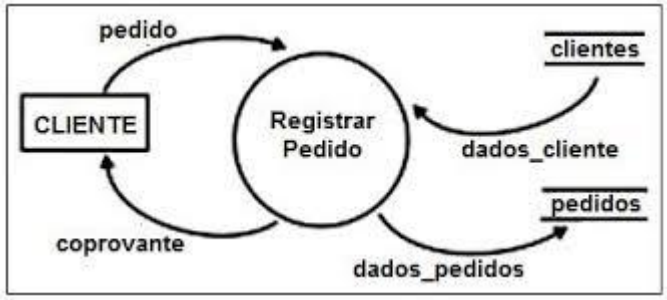
- Eventos podem modificar o **estado** do software em execução.
- A modelagem comportamental indica como o software irá responder a **estímulos** ou **eventos** externos.
- O diagrama de estados é um método para representar o comportamento de um software através da representação de seus estados e eventos que causem mudanças de estado.
- Estado é qualquer modo de comportamento externamente observável.

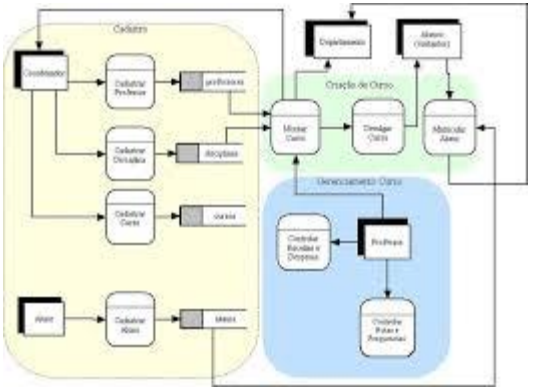


Fonte: Engenharia de Software: Uma abordagem Profissional – Pressman R. S. – McGraw Hill, Sétima Edição

Modelagem Orientada a Fluxos

- Informações são **transformadas** à medida que **fluem** através de um software;
- Software aceita **entrada** em uma variedade de formas, aplica **funções** para transformá-las e gera **saída** também em uma variedade de formas;
- Pode-se criar modelos de **fluxos** de dados para qualquer software baseado em computadores, indiferentemente de seu tamanho e complexidade.





Fonte: Engenharia de Software: Uma abordagem Profissional – Pressman R. S. – McGraw Hill, Sétima Edição

Linguagens de Modelagem

- Linguagens específicas para construção de modelos conceituais.
- São definidas por sua sintaxe e semântica.
- **UML** (Unified Modeling Language) é frequentemente utilizada para construir modelos de requisitos.
- **UML** tornou-se o padrão para a construção de sistemas de software baseada em modelos.

Como se capacitar para Modelar OO ?

Modelagem OO - Capacitação

- Noções de Engenharia de Software
- Paradigma Orientado a Objetos
- Conhecer uma Linguagem de Modelagem (UML)
- Saber os passos a serem seguidos (Quais diagramas utilizar ?)
- Avaliar o que foi produzido (Modelagem Iterativa)

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

Modelagem x Código

- Modelar é um meio e não um fim.
- O objetivo da modelagem é ao final gerar um programa que compile e execute sem erros, cumprindo os requisitos estabelecidos.
- A geração do código é uma das etapas do processo.
- Código gerado subsidia a melhoria da Modelagem.

```
Python script
1  # The script MUST include the following function,
2  # which is the entry point for this module:
3  # Param<dataframe1>: a pandas.DataFrame
4  # Param<dataframe2>: a pandas.DataFrame
5  def azureml_main():
6      import pandas as pd
7      import Hello
8      Hello.print_hello("World")
9      return pd.DataFrame(["Output"]),
10
11
```

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

Modelagem Dinâmica x Estrutural

- Softwares são sistemas físicos com características dinâmicas.
- Quando um software é executado, observam-se eventos que ocorrem ao longo do tempo.
- Portanto, software demanda Modelagem Estrutural e Modelagem Dinâmica.

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

UML

- 1995 – Método Unificado (Versão 0.8)
- 1996 – Unified Modelling Language (Versão 0.9)
- 1996 – Unified Modelling Language (Versão 0.91)
- 1998 – Versão 1.2
- 1999 – Versão 1.3
- 2002 – Versão 1.4
- 2003 – Versão 1.5
- 2005 – Versão 2

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

UML – Organização dos Diagramas

- Diagramas Estruturais
- Diagramas de Comportamento

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

UML – Diagramas Estruturais

- Diagrama de Classes
- Diagrama de Pacotes
- Diagrama de Componentes
- Diagrama de Objetos
- Diagrama de Estrutura Composta
- Diagrama de Utilização

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

UML – Diagramas de Comportamento

- Diagrama de Casos de Uso
- Diagrama de Estados
- Diagrama de Atividades
- Diagramas de Interação
  - Diagrama de Sequência
  - Diagrama de Visão Geral de Interação
  - Diagrama de Comunicação
  - Diagrama de Temporização

Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007