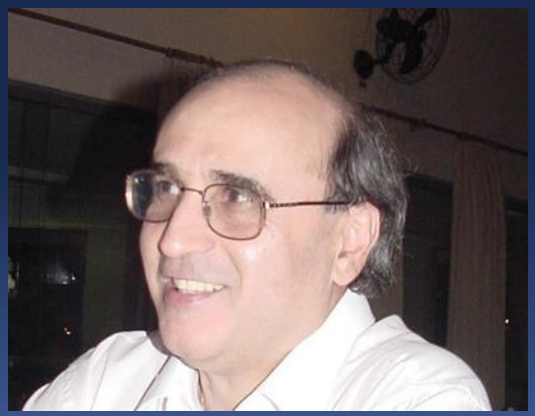
Gestão de Qualidade de Software

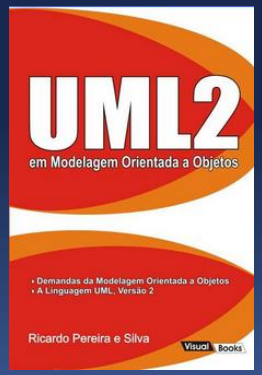
Unidade 6 – Princípios de Modelagem de Requisitos Revisão UML

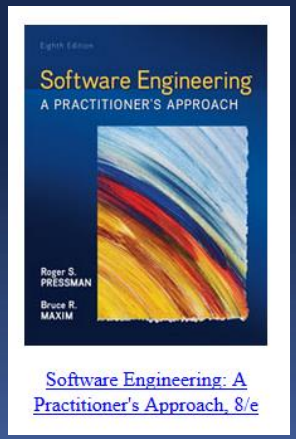


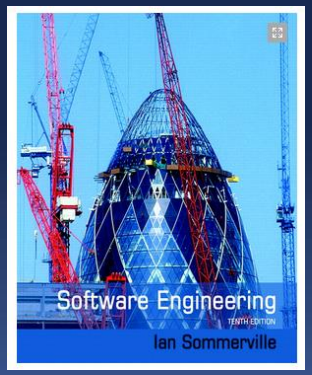
Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP
aparecidovfreitas@gmail.com
CTFL – CTFL AT - CPRE

Bibliografia

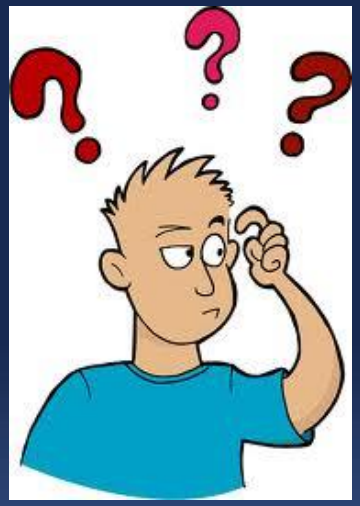
- **Software Engineering – A Practitioner's Approach – Roger S. Pressman – Eight Edition – 2014**
- **Software Engineering – Ian Sommerville – 10th edition - 2015**
- Engenharia de Software – Uma abordagem profissional – Roger Pressman - McGraw Hill, Sétima Edição - 2011
- Engenharia de Software – Ian 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





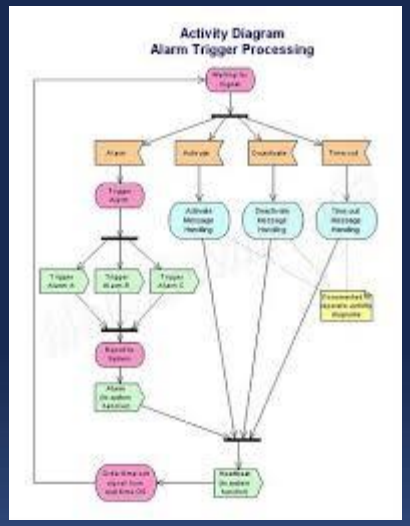


O que é Modelagem do Requisitos ?



Modelagem de Requisitos

- 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);
        }

}
```

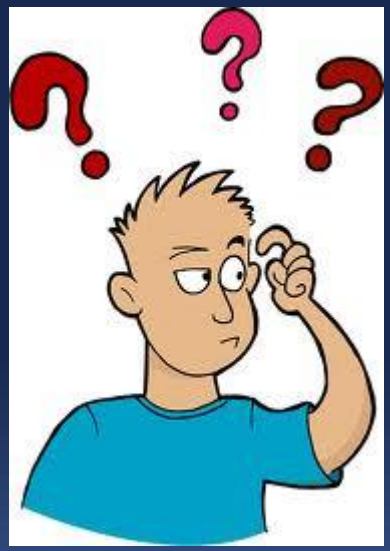
Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

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**)

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

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

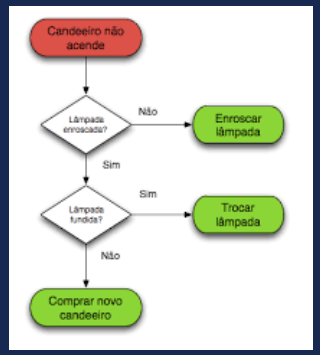
A Modelagem se aplica à qualquer situação ?

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 ?

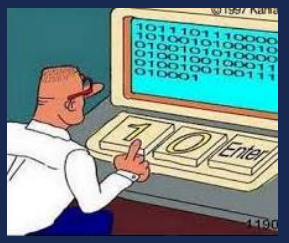
O termo Modelo

- Um modelo é uma **imagem** que abstrai a 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.



```
<?hh
class MyClass {
  public function alpha(): int {
    return 1;
  }

  public function beta(): string {
    return 'hi test';
  }
}

function f(MyClass $my_inst): string {
  // Fix me!
  return $my_inst->alpha();
}
```

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**.
- ✓ 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.
- Recomenda-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.

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

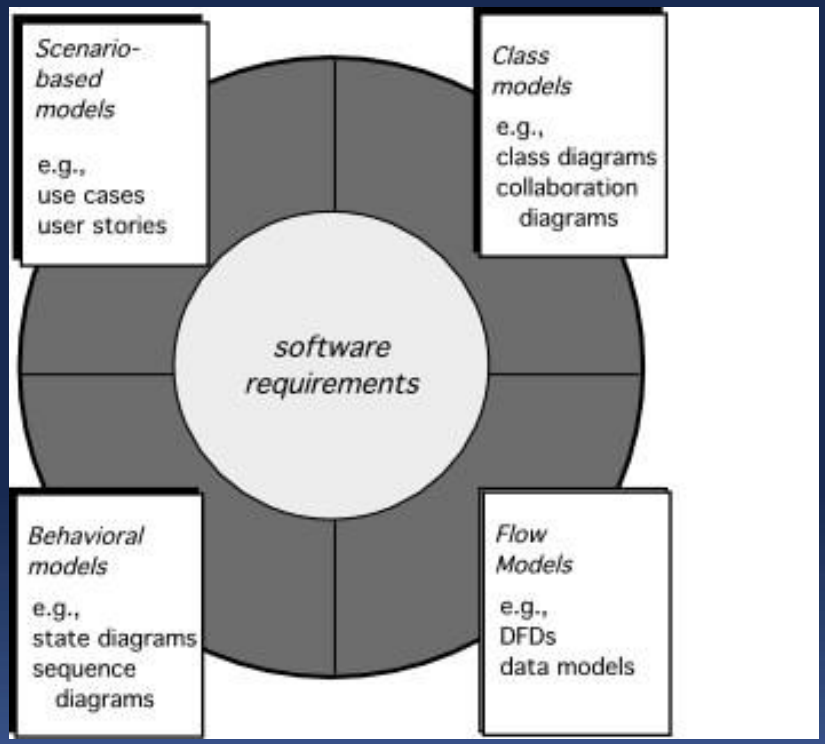
Princípios da Modelagem de Projeto

- Funcionalidades do software devem ser **coesas** (focarem uma, e somente uma, função – **ALTA COESÃO**).
- Relacionamento entre componentes deve ser mantido tão baixo quanto possível (**BAIXO ACOPLAMENTO**).
- **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

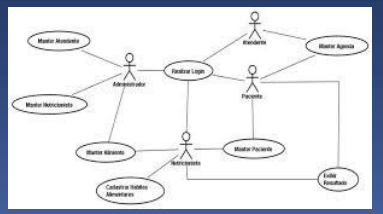
- Elementos baseados em Cenários.
- Elementos baseados em Classes.
- Elementos Comportamentais.
- Elementos Orientados a Fluxos.



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

Modelagem de Requisitos baseada em Cenários

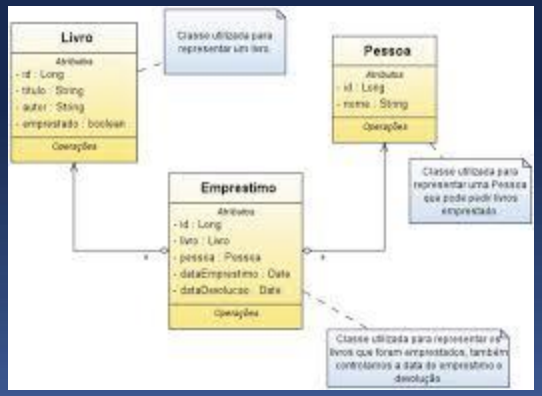
- O sistema é descrito sob o **ponto de vista do usuário**;
- Cria-se um conjunto de **cenários** que identifique um **roteiro** de uso para o sistema a ser construído. **Cenários** normalmente são chamados CASOS de USO (descrevem como o sistema será utilizado).
- Um **caso de uso** conta uma história 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 Classes

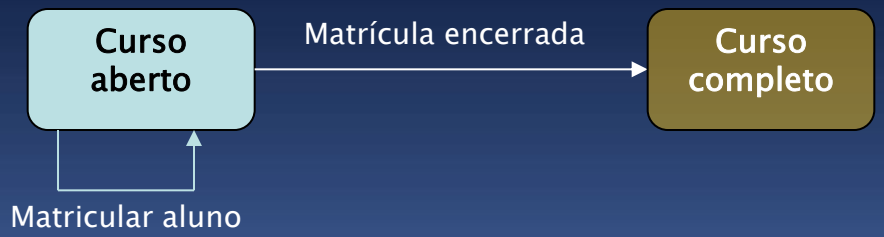
- Cada **caso de uso** implica em um conjunto de **objetos** manipulados à medida em que um **ator** (papel) interage com o software.
- Esses objetos são categorizados em **classes** – conjuntos de objetos que possuem atributos similares e comportamentos comuns.
- Um diagrama de classes **UML** pode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

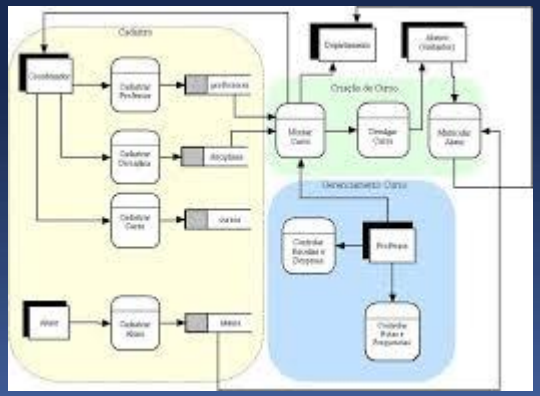
- Eventos podem modificar o **estado** do software em execução.
- A modelagem comportamental indica como o software irá responder a **estímulos** ou **eventos** externos.
- O **diagrama de estados** é um método para representar o comportamento de um software através da representação de seus estados e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

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

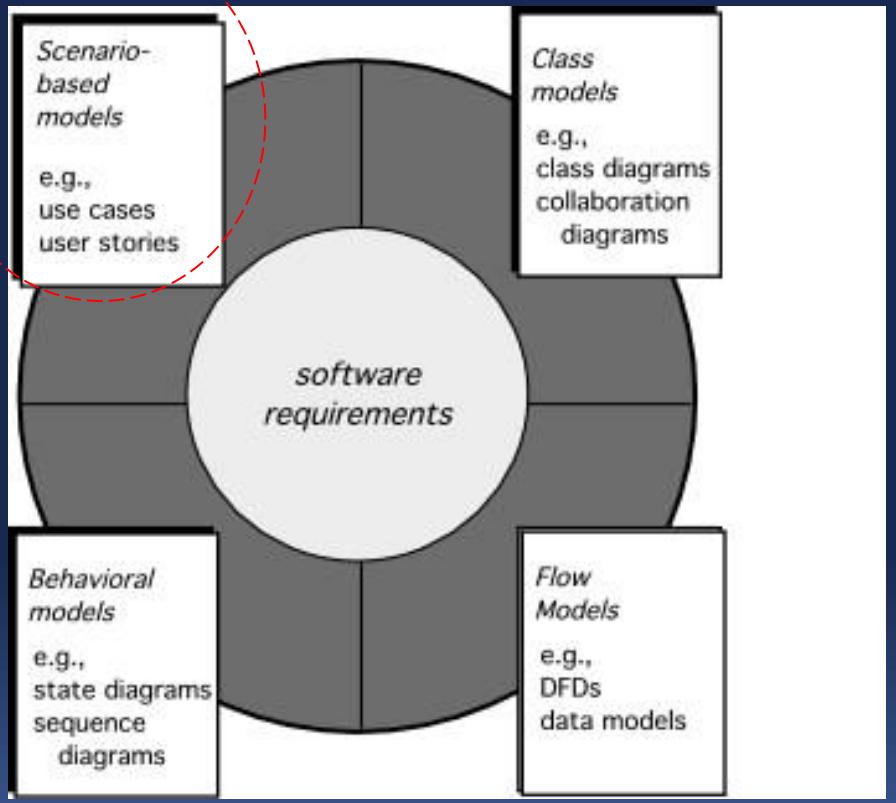
UML – Organização dos Diagramas

- Diagramas Estruturais
- Diagramas de Comportamento

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

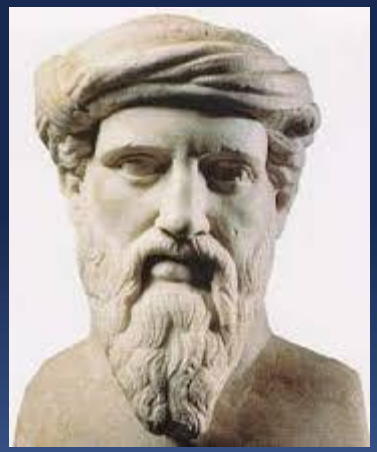
Elementos da Modelagem de Requisitos

- **Modelagem baseados em Cenários.**
- Modelagem baseada em Classes.
- Modelagem Comportamental.
- Modelagem Orientada a Fluxo.



Modelagem baseada em Cenários

- Modela os requisitos **funcionais** de um sistema.
- Na UML, para a modelagem de Cenários emprega-se o **Diagrama de Casos de Uso**.



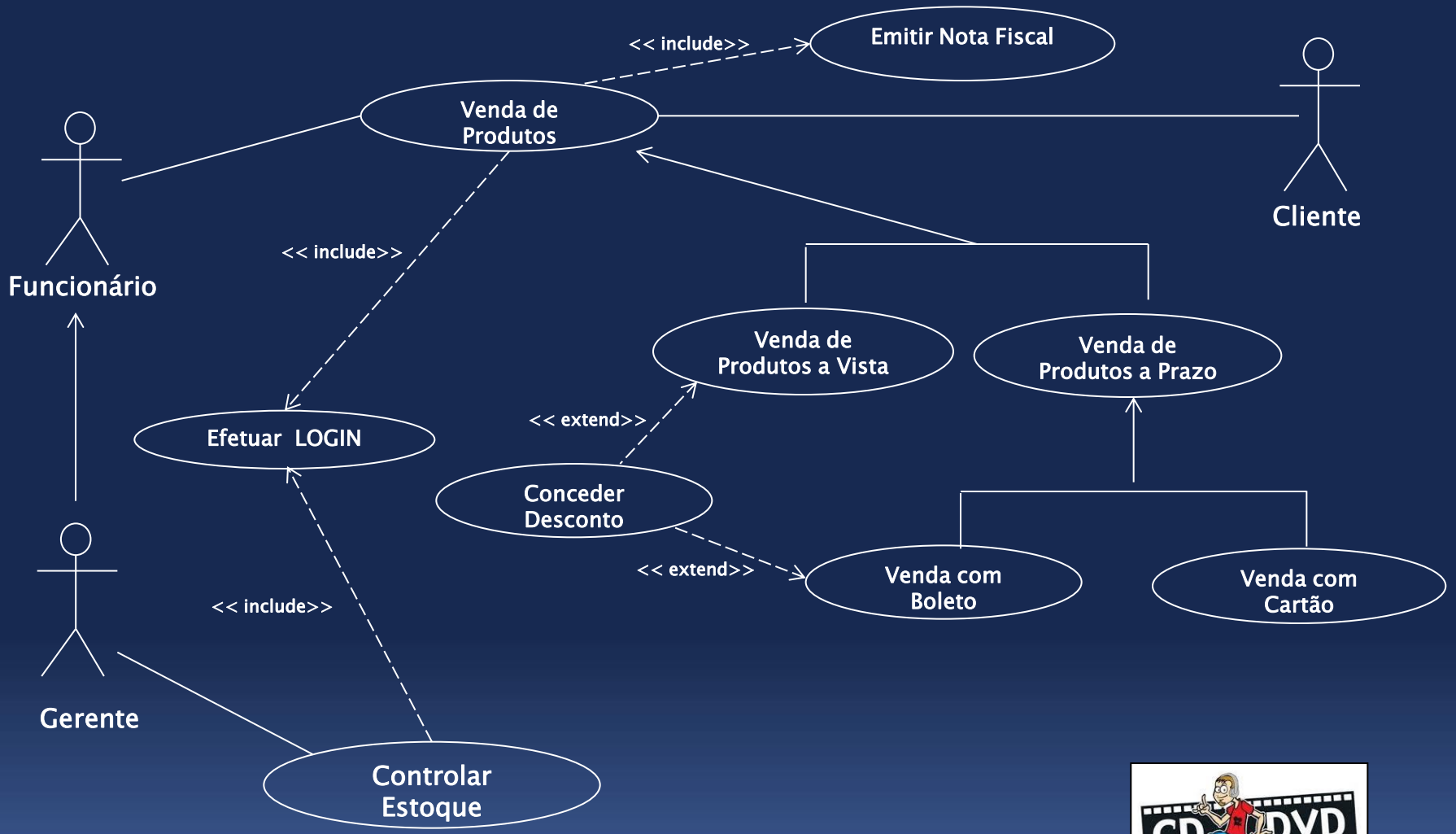
Não diga pouco em muitas palavras,
mas sim, muito em poucas.

Pitágoras

Diagramas de Caso de Uso

- Descreve de forma **esquemática** um **cenário** que exibe as **funcionalidades** do sistema sob ponto de vista do usuário.
- Apresentam o **relacionamento** das funções de um sistema.
- Apresentam também o **relacionamento** das funções de um sistema com seu **ambiente**.

Extensão de Casos de Uso



Documento de Caso de Uso

- UML não define um formato específico.
- Assim, o formato é bastante flexível.
- Pode-se usar pseudo-código, embora esse procedimento fuja bastante do objetivo do Diagrama de Casos de Uso, que é usar uma linguagem simples, de forma que até mesmo leigos possam entendê-la.
- Fornece a base para o **plano de testes** do sistema.

Fonte: UML uma abordagem prática. Gilleanes T. A. Guedes, 2004 – Novatec

Modelagem de Requisitos com Casos de Uso - Observações

- Um Caso de Uso captura um **contrato** que descreve o comportamento do sistema sob várias condições à medida que o sistema responde a uma solicitação de seus interessados. **[Cockburn,2001]**
- O contrato define o **procedimento** pela qual o ator usa o sistema.
- Essencialmente, um caso de uso conta uma história estilizada sobre como os procedimentos que um usuário final (desempenhando uma série de papéis possíveis) faz na **interação** com o sistema.
- Um caso de uso representa o software ou o sistema sob **ponto** de **vista** do **usuário**.
- O caso de uso pode ser complementado por meio de uma representação gráfica do fluxo de interação em um cenário específico. Na **UML**, pode-se usar o **Diagrama de Atividades**.

Fonte: UML uma abordagem prática. Gilleanes T. A. Guedes, 2004 – Novatec

Diagrama de Atividades

- ✓ Exibe o comportamento dinâmico de um sistema ou parte dele.
- ✓ Pode ser usado para **REFINAMENTO** dos Casos de Uso.
- ✓ Podem ser usados para descrever algoritmos de método de Classe.
- ✓ É o diagrama com maior ênfase ao nível de algoritmo.
- ✓ Apresenta muitas semelhanças com os antigos Fluxogramas.

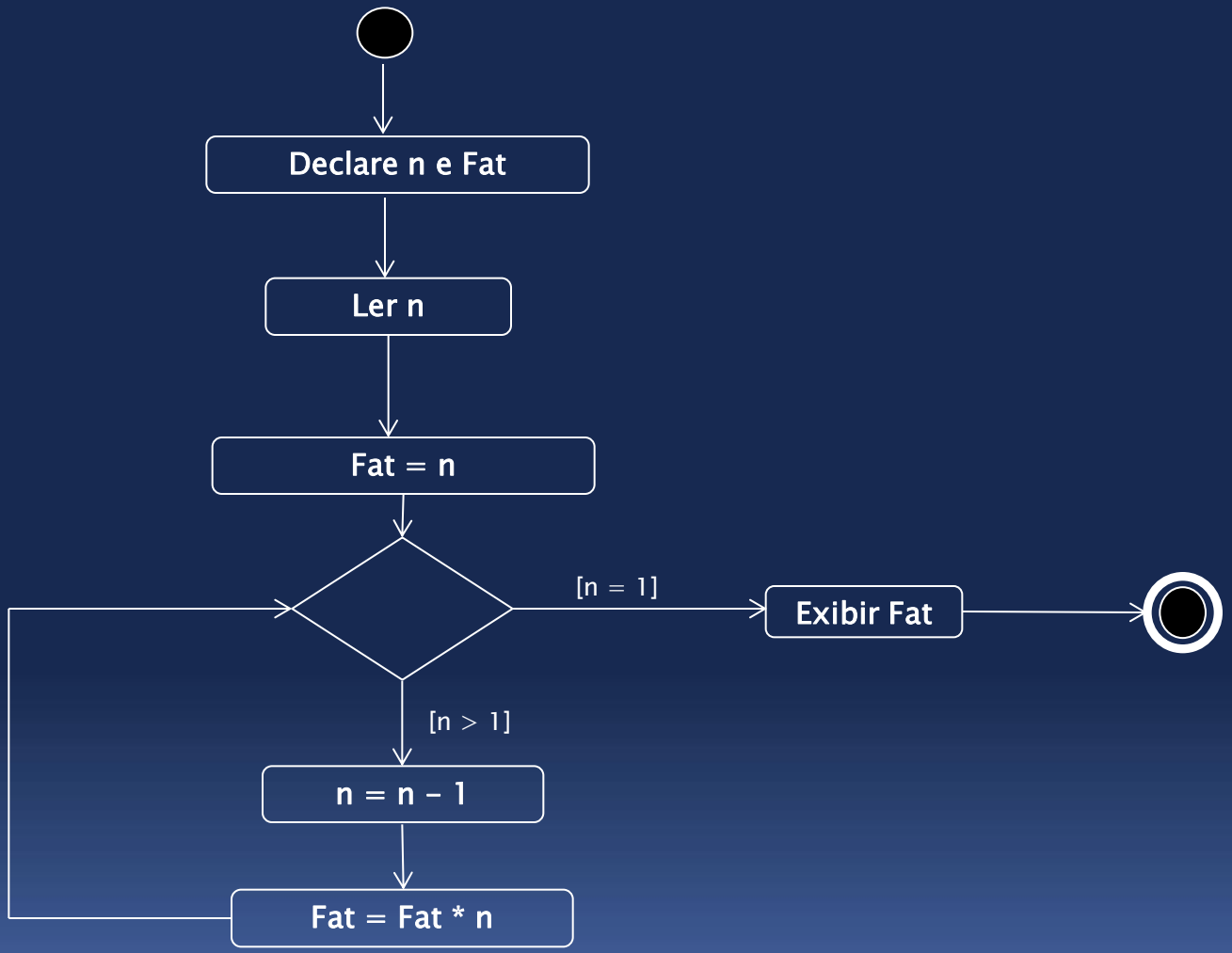
Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

Diagrama de Atividades – Principais Elementos

- Ação
- Fluxo de Controle
- Nodos Inicial e Final
- Pontos de Decisão
- Sincronização

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

Exemplo: Diagrama de Atividades



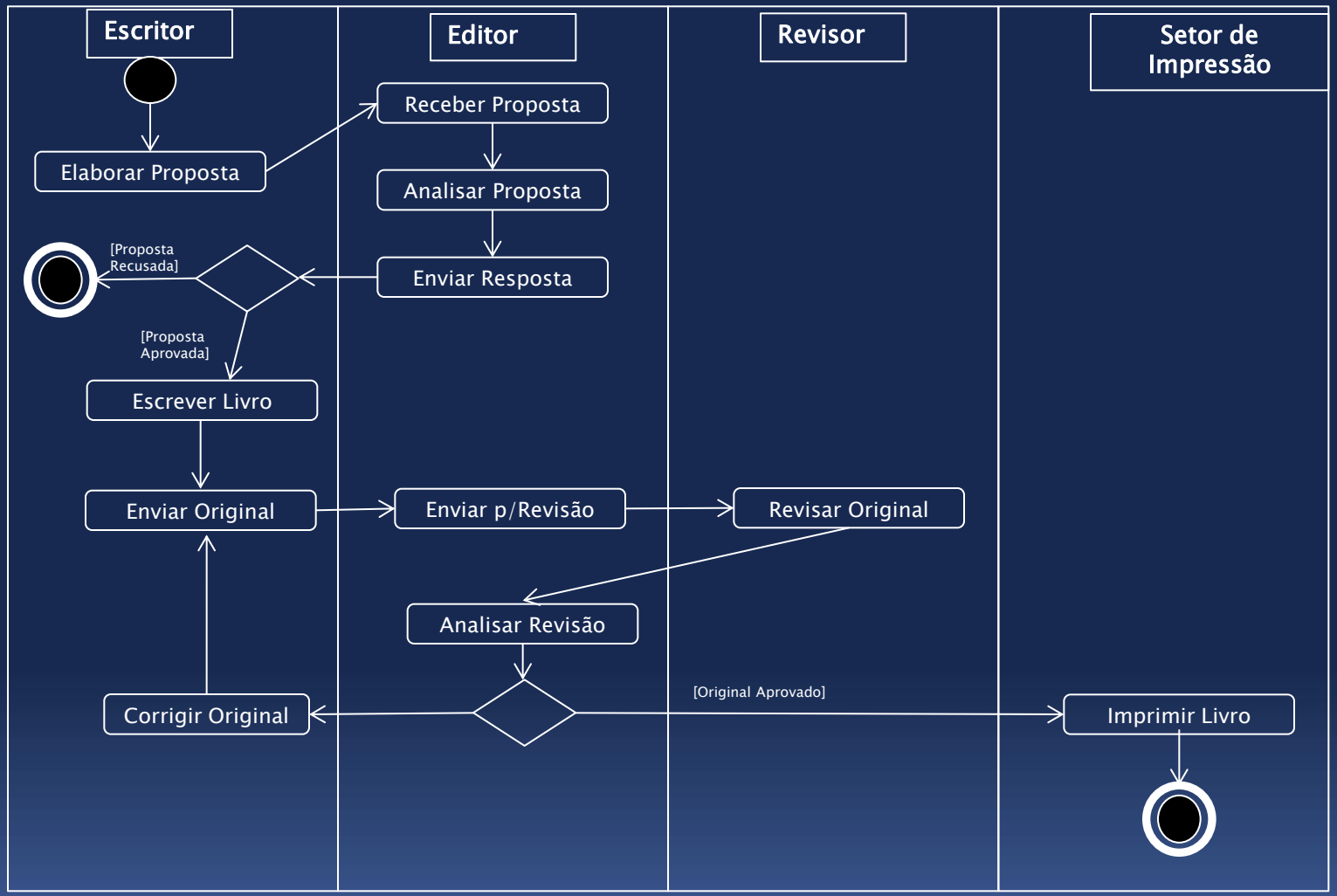
Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

Partição de Atividades

- Permite a divisão das atividades e ações no Diagrama em grupos com determinadas características em comum.
- Qualquer critério pode ser adotado para a definição dos Grupos, como por exemplo: responsável pela execução ou localização física da execução.

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

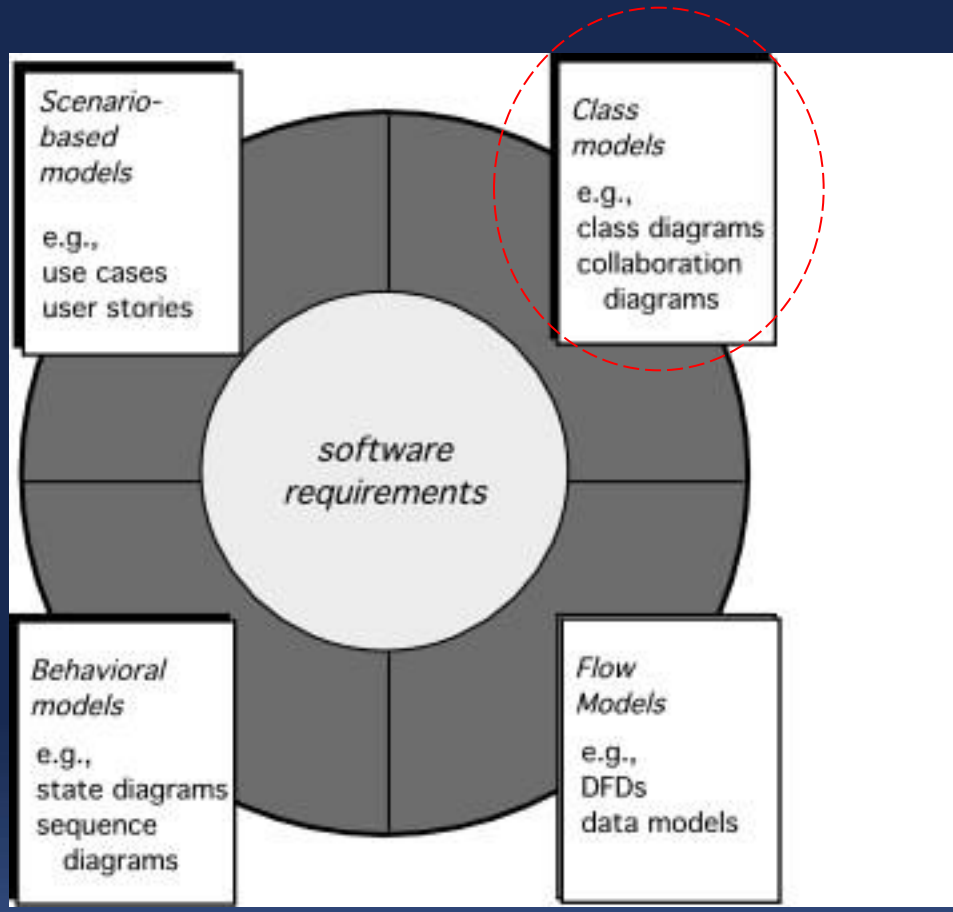
Diagrama de Raias



Fonte: UML uma abordagem prática. Gilleanes T. A. Guedes – Novatec 2004

Elementos da Modelagem de Requisitos

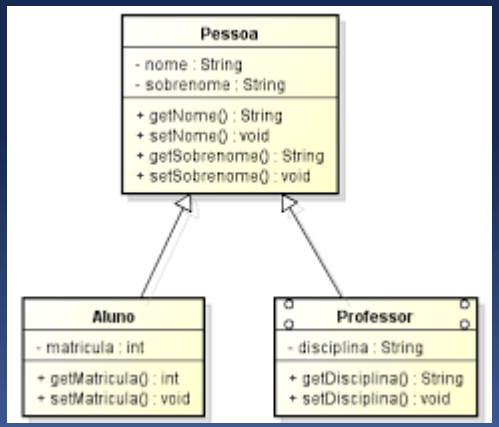
- Modelagem baseados em Cenários.
- **Modelagem baseada em Classes.**
- Modelagem Comportamental.
- Modelagem Orientada a Fluxo.



Modelagem baseada em Classes

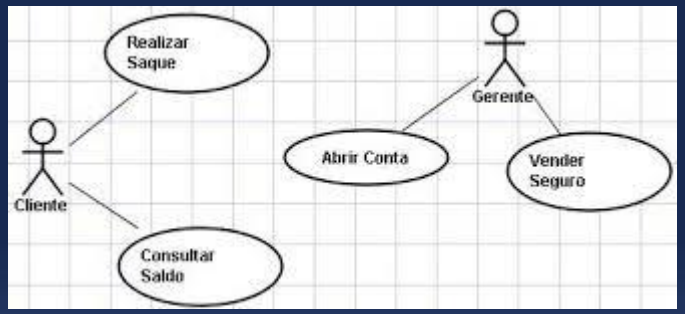
Representam:

- Os **objetos** que o software irá manipular;
- As **operações** (também chamadas de métodos ou serviços) que serão executadas pelos objetos;
- **Relacionamento** entre objetos;



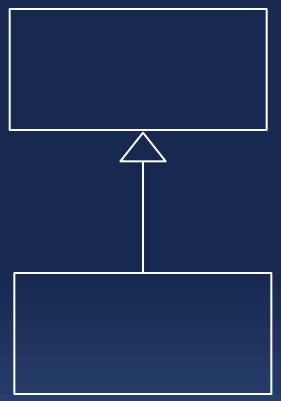
Classes de Análise

- ✓ Classes obtidas por meio dos **cenários de uso** desenvolvidos como parte da modelagem de requisitos (Casos de Uso);
- ✓ Classes podem ser obtidas por meio de "**análise sintática**" dos casos de uso do sistema a ser construído;
- ✓ Sublinham-se os substantivos (nomes) e verbos são colocados em itálico;
- ✓ Listam-se as classes potenciais para incluí-las no modelo de análise.



Casos de Uso

Cenários

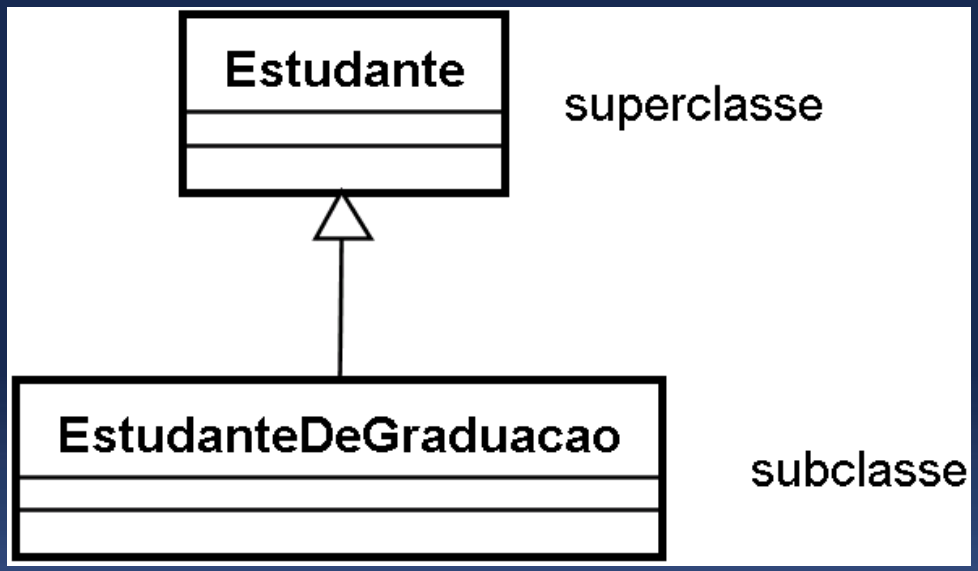


Classes

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

Frase específica da Herança

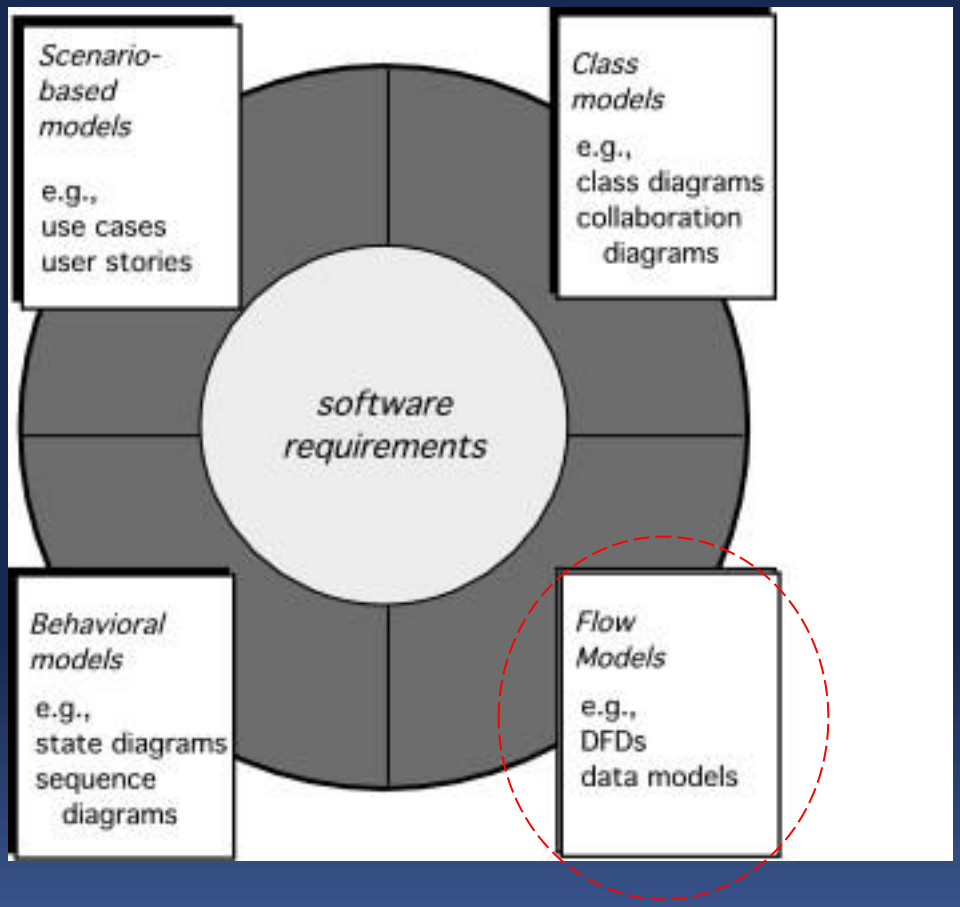
- ✓ <subclasse> **é uma espécie de** <superclasse> ;
- ✓ EstudanteDeGraduacao **é uma espécie de** Estudante;
- ✓ Se a frase não faz sentido, o relacionamento de herança não se aplica;
- ✓ Por exemplo: Leão é uma espécie de Elefante.



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

Elementos da Modelagem de Requisitos

- Modelagem baseados em Cenários.
- Modelagem baseada em Classes.
- **Modelagem Orientada a Fluxo.**
- Modelagem Comportamental.



Modelagem orientada a Fluxo de Dados

- ✓ Atividade fundamental na Análise Estruturada;
- ✓ Representa como os dados são transformados no software;
- ✓ O Diagrama de Fluxo de Dados (DFD) é a forma diagramática para representar as transformações dos dados;

DFD – Diagrama de Fluxo de Dados

- Modela as funcionalidades do sistema por meio de processos (funções), repositórios de dados, fontes (sources) e destinos (sinks) no ambiente do software, bem como **fluxos de dados**.
- Concebido pela **Análise Estruturada** (DeMarco, 1978).
- Modelos de fluxos de dados permitem a modelagem do sistema em diferentes **níveis** de **abstração**.

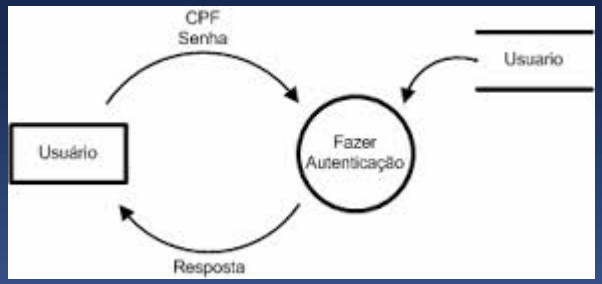
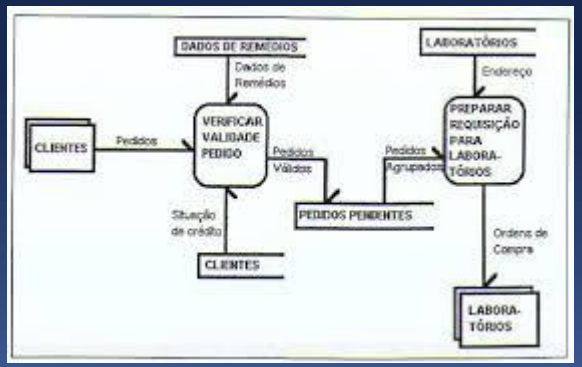
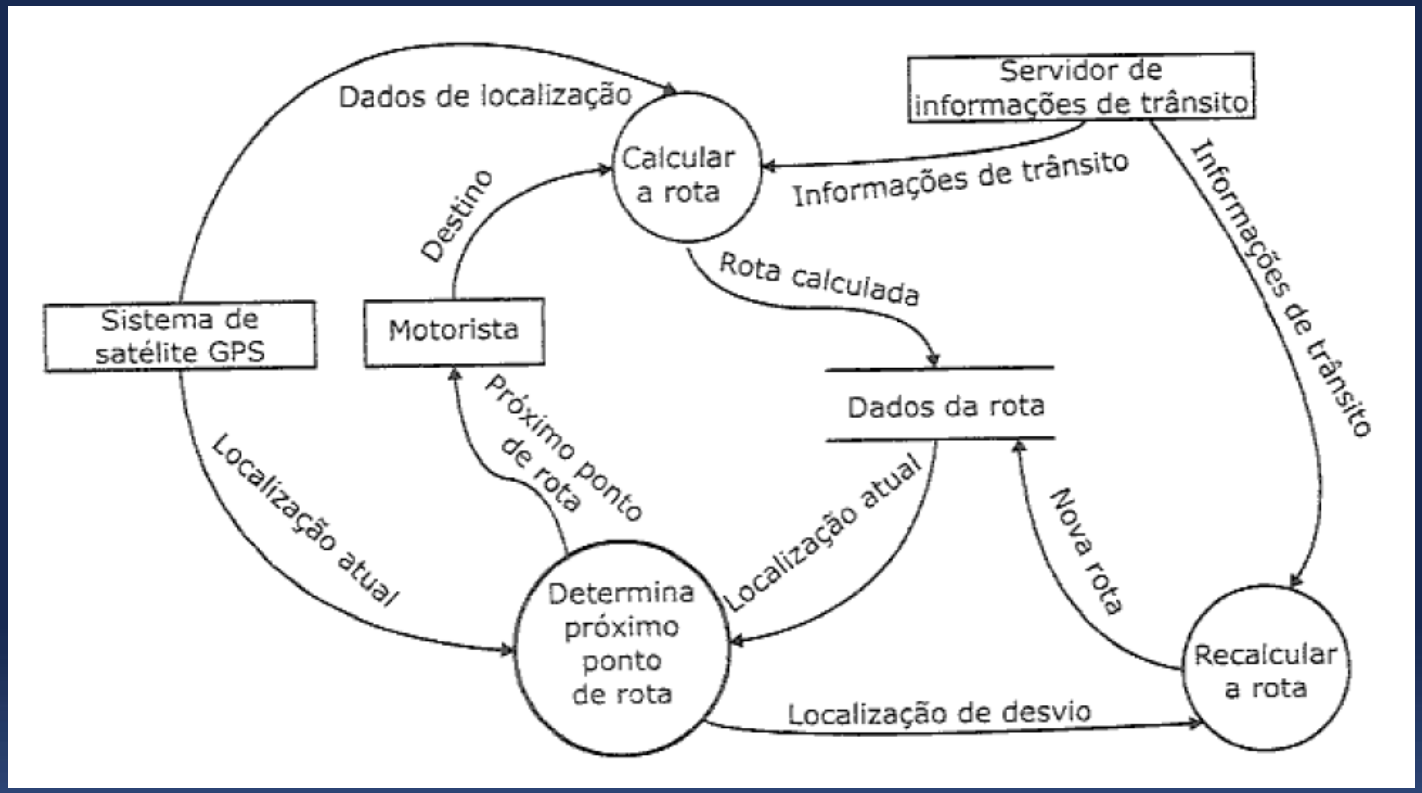


Diagrama de Fluxo de Dados

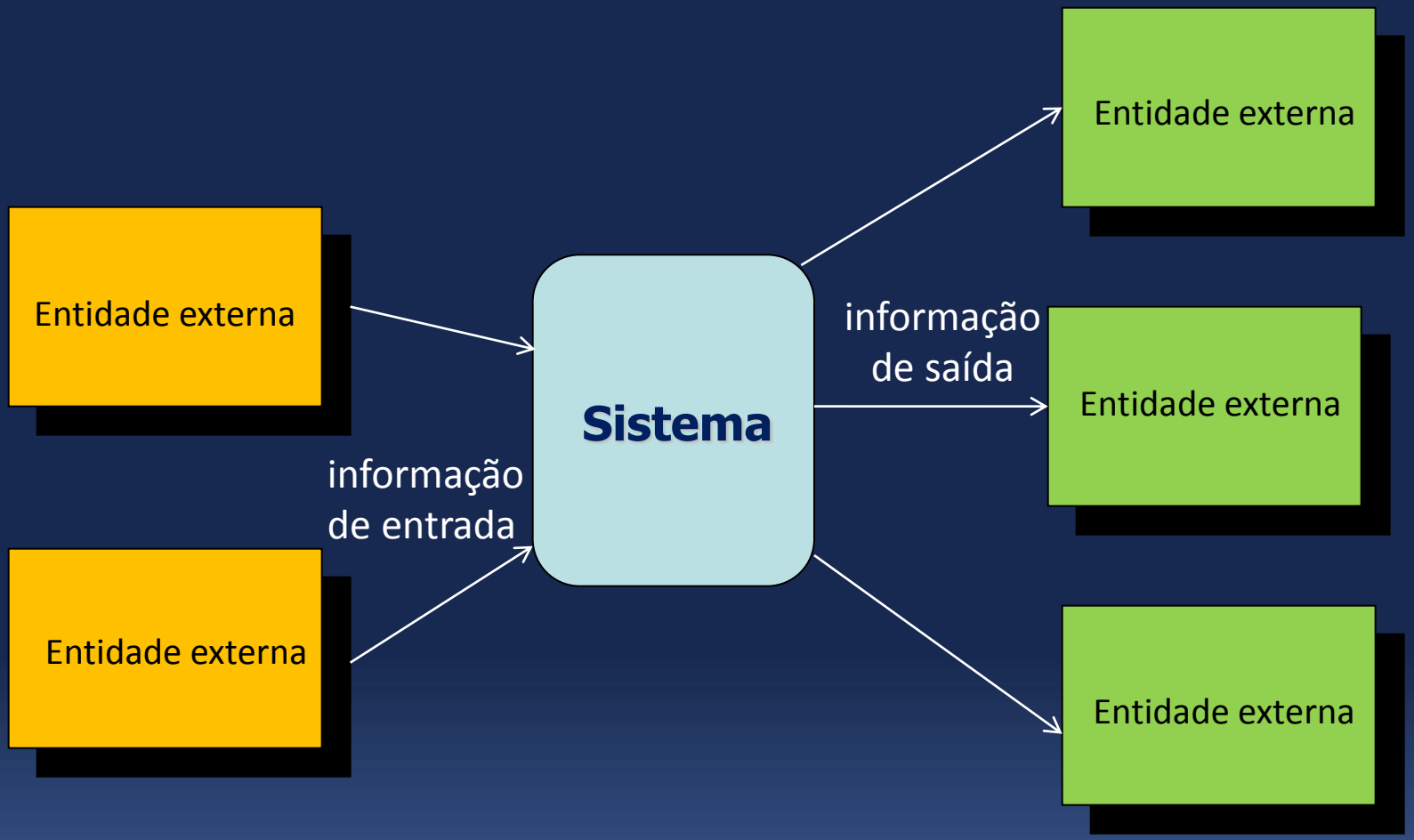
- Embora não façam parte formal da **UML**, são utilizados para **complementar** os diagramas **UML** e darem uma visão adicional sobre o fluxo e requisitos do software;
- O diagrama é representado de forma **hierárquica** (**Estruturada**);
- O primeiro modelo de fluxo de dados (denominado **DFD nível 0** ou **diagrama de contexto**) representa o sistema como um todo;
- Diagramas de Fluxo de Dados subsequentes **refinam** o diagrama de contexto, fornecendo detalhamento progressivo em cada nível subsequente.



DFD – Exemplo

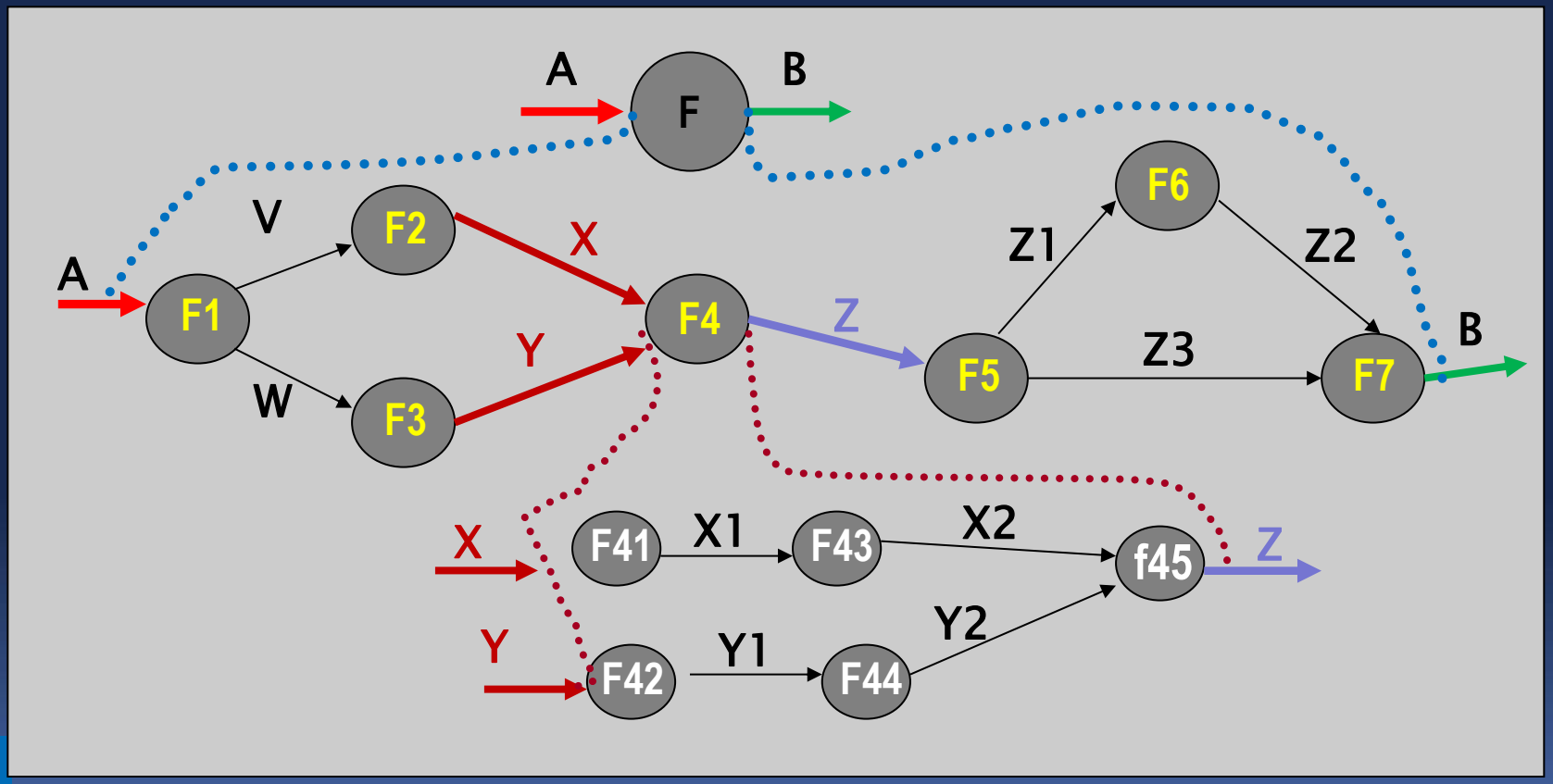

Dados de localização
Servidor de informações de trânsito
Calcular a rota
Informações de trânsito
Informações de trânsito
Destino
Rota calculada
Sistema de satélite GPS
Motorista
Dados da rota
Próximo ponto de rota
Localização atual
Localização atual
Nova rota
Determina próximo ponto de rota
Recalcular a rota
Localização de desvio

DFD de nível 0
Modelo de contexto



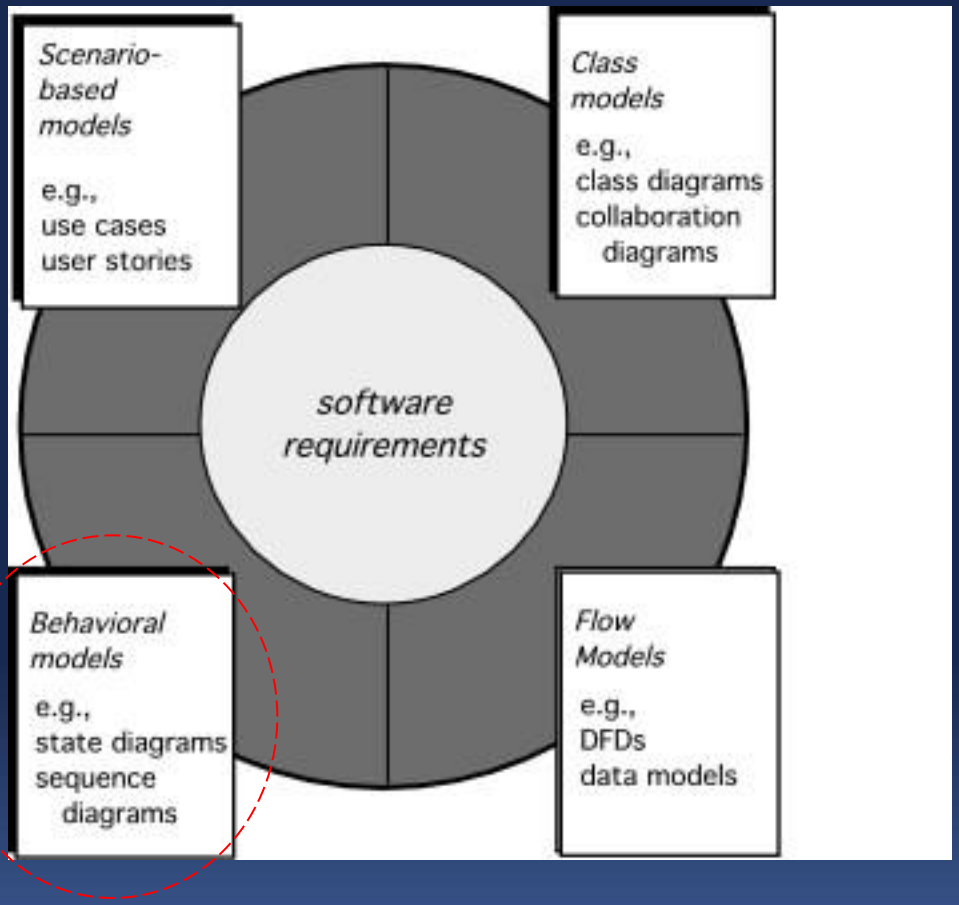
Refinamento do Fluxo de Informação

- ✓ O DFD de nível 0 é dividido em partições para revelar mais detalhes;
- ✓ A continuidade do fluxo de informação deve ser mantida (BALANCEAMENTO).



Elementos da Modelagem de Requisitos

- Modelagem baseados em Cenários.
- Modelagem baseada em Classes.
- Modelagem Orientada a Fluxo.
- **Modelagem Comportamental.**



Interação de Objetos em UML

- Para se descrever a execução de um programa orientado a objetos, necessita-se modelar o seu comportamento (tempo de execução).
- Objetos podem enviar invocações de métodos a outros objetos .
- Assim, a interação de objetos corresponde a um dos requisitos da modelagem dinâmica de sistemas.

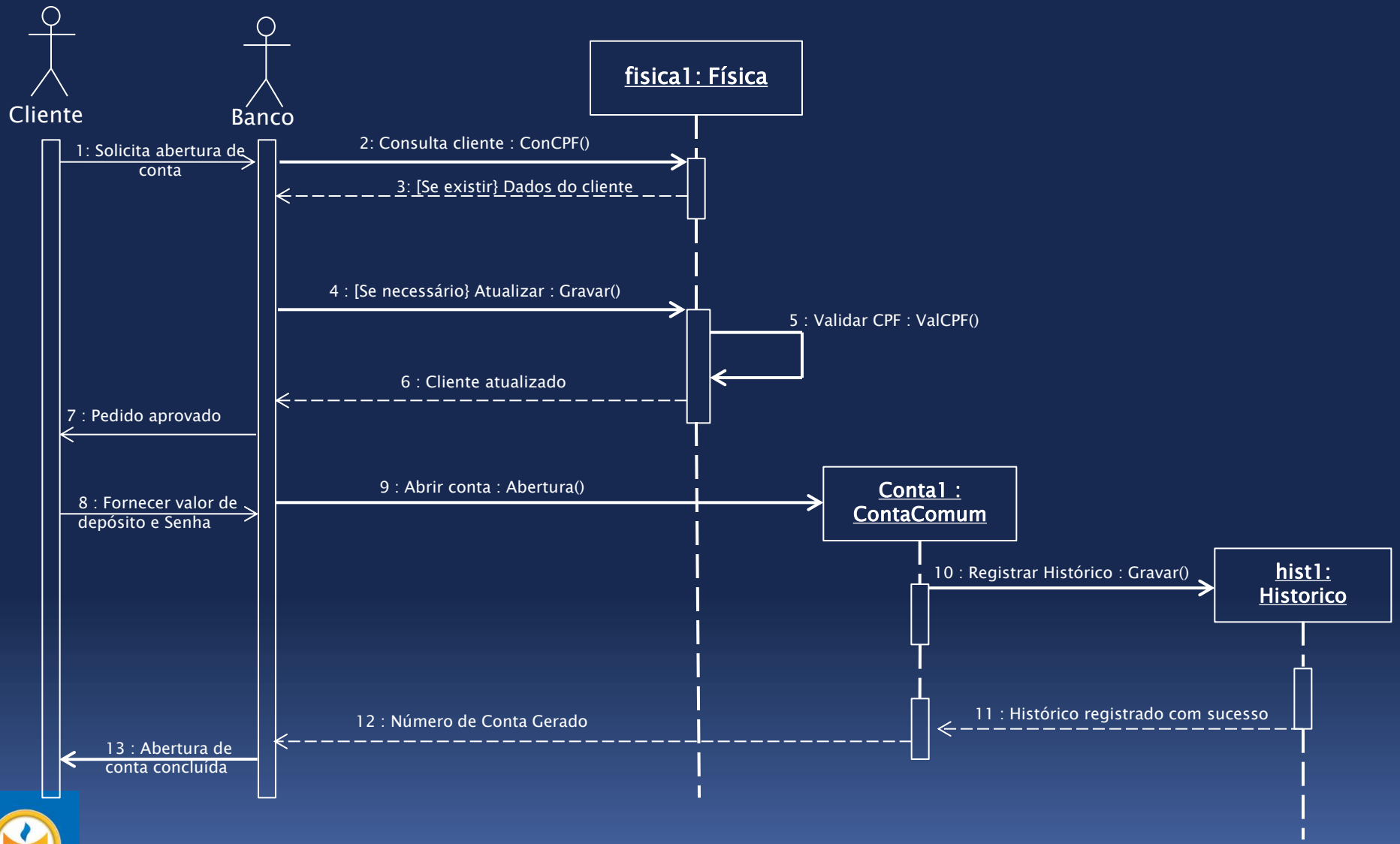
Fonte: UML 2 em Modelagem Orientada a Objetos – Prof. Ricardo Pereira e Silva – UFSC, Visual Books, 2007

Modelagem de Interação de Objetos

- Modelagem de **sequências** de processamento.
- Não descreve completamente o processamento que ocorre no programa.
- Foca-se apenas nas **comunicações** entre as **instâncias** (**objetos**) e avaliação de condições entre essas comunicações.
- Diagramas de modelagem de interação modelam apenas invocações de métodos, sem menção a outras ações, por exemplo: atribuições. (**Não** se descreve, portanto, todo o **algoritmo** de um método...)

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

Exemplo – Diagrama de Sequência



Fonte: UML uma abordagem prática. Gilleanes T. A. Guedes – Novatec 2004

Diagrama de Estados

- O comportamento de um objeto em um determinado instante frequentemente depende do **estado** do objeto (valores de seus **atributos** naquele instante);
- Um Diagrama de Estados modela os estados de um objeto, as ações executadas e as transições entre os estados do objeto;
- Por meio do Diagrama de Estados, garante-se que todos os **eventos** possíveis para os estados tenham sido levados em conta;
- Especifica a **sequência** de **estados** que um objeto pode ter.
- Considera eventos, condições e ações que levam o objeto a alcançar tais estados.

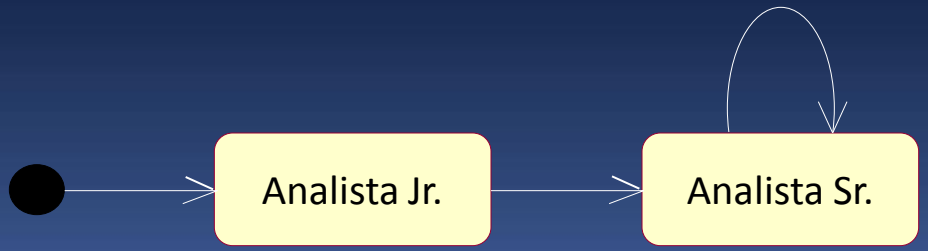


Diagrama de Estados