Aula 04

Site: MoodleWIFI
Curso: Analise de sistemas
Livro: Aula 04
Impresso por: RIANE RUBIO
Data: Friday, 12 Apr 2019, 19:54

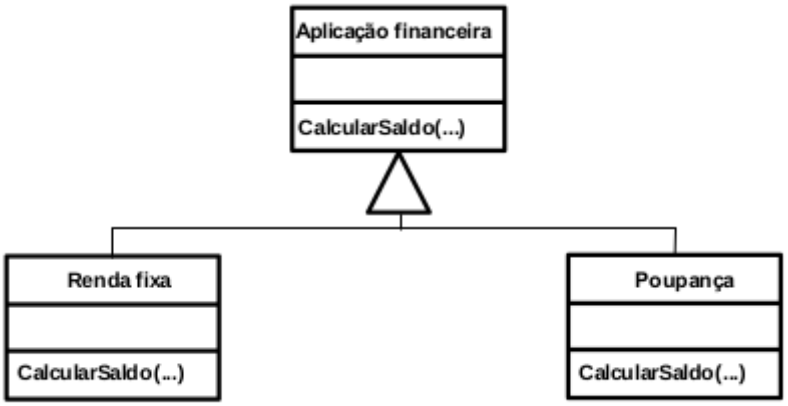
Sumário

1. Introdução

Polimorfismo

Pode-se declarar funções e procedimentos com os mesmos nomes, enquanto suas chamadas possam a ser distintas pelos seus parâmetros de retorno, consistindo do número dos seus argumentos e dos seus tipos de valores de retorno.

No contexto **OO**, Polimorfismo significa que diferentes tipos de Objetos podem responder a uma mesma mensagem de maneiras diferentes.



Pode-se declarar funções e procedimentos com os mesmos nomes, enquanto suas chamadas possam a ser distintas pelos seus parâmetros de retorno, consistindo do número dos seus argumentos e dos seus tipos de valores de retorno.

2. Métodos

Métodos para análise com orientação a objetos

A Análise Orientada a Objeto remonta a década de 80, com modelos que estendiam o Modelo Entidade Relacionamento.

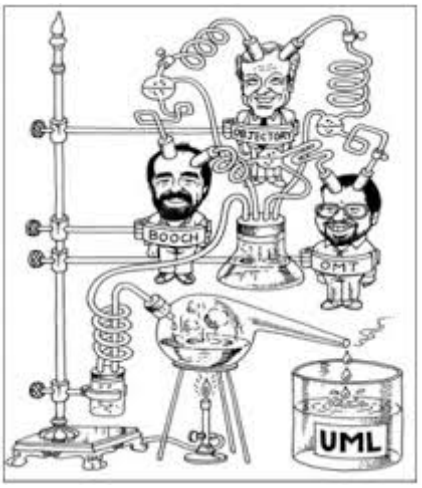
Do crescimento dos métodos de **BOOCH** e **OMT**, independentemente, e o reconhecimento pela classe usuária mundial, nasceu uma tentativa de padronização dos métodos de **AOO**, por **Grady Booch** e **James Rumbaugh** - criação da **Rational Corporation** (*1994*).



Ivar Jacobson juntou-se a equipe em 1995, integrando seu método **OOSE**.

Receberam, então, a incumbência de criar uma linguagem de modelagem unificada que tratasse assuntos de escala inerentes a sistemas complexos e de missão crítica, que se tornasse poderosa o suficiente para modelar qualquer tipo de aplicação, de tempo real, cliente servidor, ou outros tipos de software padrão.

3. O método BOOCH



O método de **BOOCH** consistia no emprego de técnicas de desenho orientado a objeto, apesar de ter sido estendido para contemplar também a **AOO**.

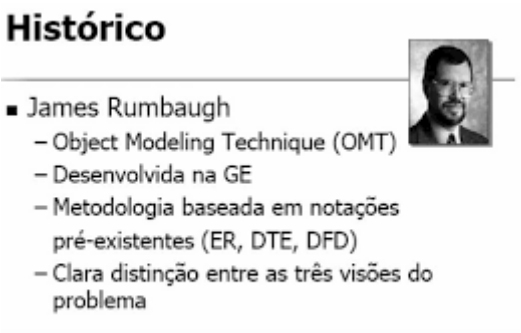
Descreve um objeto como sendo um modelo do mundo real que consiste de dados e habilidades para o tratamento desses dados.

Argumenta que o desenho estruturado funciona bem com linguagem de programação estruturada, e
que o enfoque estruturado direciona a concentração dos esforços na lógica do módulo, dando pouca ênfase no uso dos dados.

O desenho orientado a objeto é organizado via abstrações algorítmicas de forma parecida à programação orientada a objeto.

Estreita relação entre desenho e programação permite tomar decisões de negócio antes da criação do código.

4. O método OMT



O método **OMT** (*Object Modeling Technique*)

Desenvolvido pela **GE** *Corporation*, conhecido como Técnica de Modelagem de Objetos.

Baseado na modelagem semântica dos dados, derivado dos modelos estruturados.

Suporta conceitos como:

- Atributos e Relacionamentos;
- Composição, Agregação e Herança.

O seu ponto forte é a notação utilizada e o enfoque relativamente conservador.

Um problema, em sua estrutura, é a falta de notação para representar a passagem de mensagens entre objetos.

5. O método OOSE

O método **OOSE** (*Object-Oriented Software Engineering*)

Foco em **Casos de Uso** e a categorização de pessoas e equipamentos.

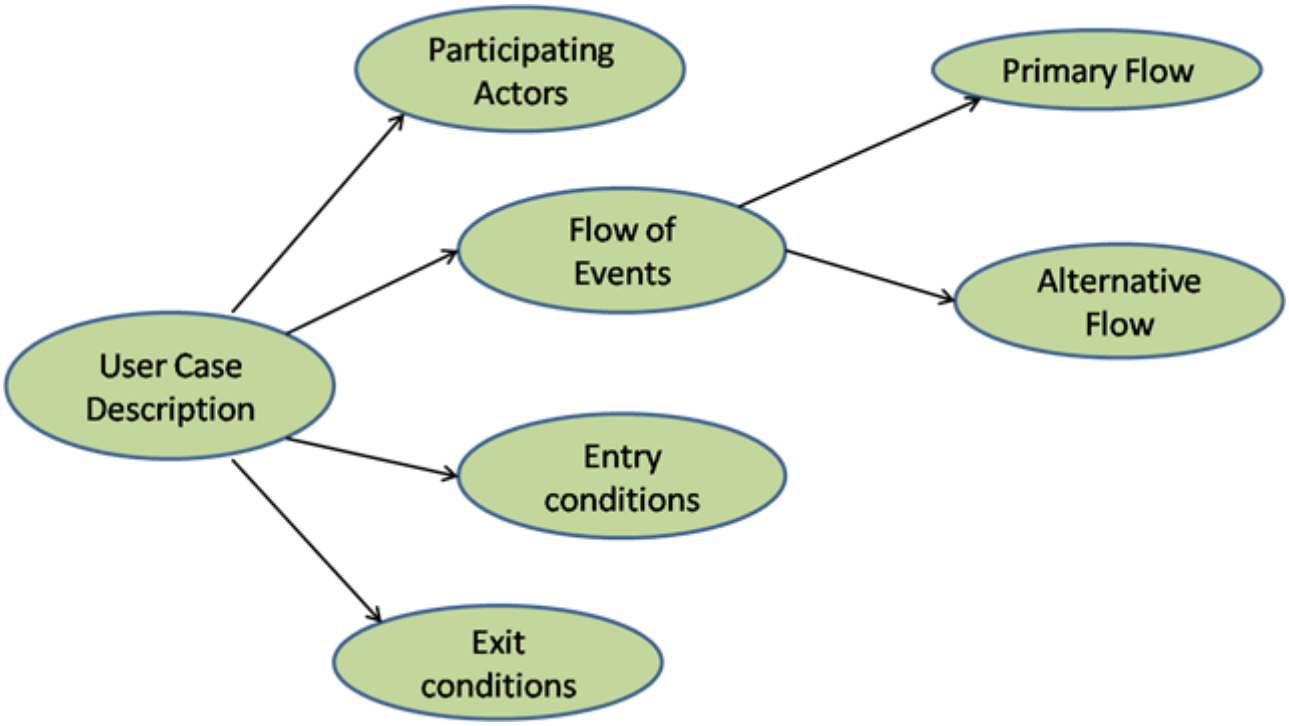
Técnica baseada em modelos de requerimento e análise que consistem em:

Conjunto de Casos de Uso;

Modelo de domínio do problema e Interface do sistema.

O seu ponto forte é o **método Objectory**, que tem sido adaptado para a engenharia de negócio.

Notação simplista usada para objetos de domínio.



6. UML

UML – Unified Modeling Language

A **UML** não é uma metodologia, é uma linguagem de modelagem bem definida, expressiva, poderosa e aberta.

Aprovada pela **OMG** *Object Management Group* em 1997

A **UML** pode ser usada para:

- Mostrar as fronteiras de um sistema e suas principais funções (*Atores e Casos de Uso*);
- Ilustrar a realização de Casos de Uso com Diagramas de Interação;
- Representar a estrutura estática de um sistema utilizando Diagramas de Classe;
- Modelar o comportamento de objetos com Diagramas de Transição de Estado;
- Revelar a arquitetura de implementação física com Diagramas de Componente e de Implantação;
- Estender sua funcionalidade através de estereótipos.

7. Visões em UML

As visões são usadas em **UML** para descrever os diferentes aspectos do sistema sendo modelado.

Uma visão é uma abstração do sistema, formada por um conjunto de diagramas.

A partir de um conjunto de visões se chega a uma descrição completa do sistema a ser construído.

Tipos de visões em UML:

- **Visão de Casos de Uso** - mostra a funcionalidade do sistema do ponto de vista externo. Casos de uso podem ser descritos textualmente através de uma descrição do Fluxo de Eventos e/ou visualmente através dos **Diagramas de Casos de Uso**.
- **Visão Lógica** - A visão lógica tem por objetivo descrever como a funcionalidade do sistema será obtida. A visão de casos de uso descreve o sistema do ponto de vista do mundo externo ao sistema. A visão lógica descreve o sistema do ponto de vista interno. Descreve a organização do sistema, seus módulos principais, como eles se relacionam e suas funcionalidades. A visão lógica envolve a estrutura estática do sistema (*módulos, classes, objetos e relacionamentos*) e também os aspectos dinâmicos (*manifestados através de mensagens entre objetos, eventos externos, etc*). A estrutura estática é descrita através de diagramas de classes. O modelamento dinâmico é feito através de diagramas de sequência , diagramas de colaboração e diagramas de estado.
- **Visão de Componentes** - descreve a arquitetura física do sistema, em termos de componentes de software.

8. Diagramas da UML

Para que as visões possam ser visualizadas graficamente, a UML dispõe de vários tipos de diagramas :

1. Diagramas de casos de uso (*casos + atores + relações*)
2. Diagramas de sequência (*detalhamento - fase projeto*)
3. Diagramas de colaboração (*troca de mensagens entre objetos*)
4. Diagramas de classes (*classes + responsabilidades -- atributos, relacionamentos, métodos, cenários*)
5. Diagrama de atividades (*ordenamento dos casos de uso*)
6. Diagramas de estados (*ciclos de vida, comportamentos*)
7. Diagrama de componentes (*grupo de objetos divididos em nodos físicos*)

9. Casos de uso

Diagrama de casos de uso

A modelagem de um diagrama de casos de uso é uma técnica usada para descrever e definir os requisitos funcionais de um sistema.

Eles são escritos em termos de atores externos, casos de uso e o sistema modelado.

Os atores representam o papel de uma entidade externa ao sistema como um usuário, um hardware, ou outro sistema que interage com o sistema modelado.

Os atores iniciam a comunicação com o sistema através dos casos de uso, onde o caso de uso representa uma seqüência de ações executadas pelo sistema e recebe do ator, que lhe utiliza, dados tangíveis de um tipo ou formato já conhecido, e o valor de resposta da execução de um caso de uso (*conteúdo*) também já é de um tipo conhecido, tudo isso é definido juntamente com o caso de uso através de texto de documentação.

O objetivo final do sistema é oferecer a funcionalidade descrita pelos casos de uso.

Sendo assim, a visão dos casos de uso é importante também na validação do sistema.

Exemplo:

10. Simbologia

Simbologia para diagramas de casos de uso