Introdução a Orientação a Objetos

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Fevereiro de 2011

Qualidade de *Software*

- Obter *software* de qualidade é um dos principais objetivos da Engenharia de *Software*;
- Desafio desde os primórdios da computação;
- Glend Myers(1975), escreveu que: obter qualidade em *software* de grande porte, era um dos maiores desafios para época.

Fatores de Qualidade

- **Fatores Externos**: Visão do usuário:
  - Robustez – um *software* é robusto, quando realiza de forma correta suas tarefas, mesmo quando submetido a condições extremas;
  - Confiabilidade – um *software* deve transmitir confiança ao seu usuário, realizando de maneira simples e correta, as tarefas para qual foi desenvolvido;
  - Eficiência – refere-se ao bom uso que um *software* faz dos recursos de *hardware*. Ex.: processador e memória.

Fatores de Qualidade

- **Fatores Externos**: Visão do usuário:
  - Usabilidade - termo usado para definir a facilidade com que as pessoas podem empregar uma ferramenta ou objeto, a fim de realizar uma tarefa específica;
  - Integridade – refere-se à segurança empregada aos dados e documentos utilizados pelo sistema;
  - Portabilidade – possibilidade de utilização do *software* em diversos ambientes de *software* e/ou *hardware*, e
  - etc.

Fatores de Qualidade

- **Fatores Internos**: Visão Estrutural do *Software*. Permitem que o *software* atinja os objetivos dos fatores externos. Isto se dá através da:
  - Modularidade – característica de um *software* que foi construído em módulos;
  - Legibilidade - qualidade que determina a facilidade de leitura de alguma coisa, no caso de *software*. Ex.: Lógica e simples e fácil de ser entendida;
  - Manutenibilidade – facilidade na hora de realizar a manutenção no *software* e
  - etc.

Paradigmas da Programação

- Ao longo dos anos, a Engenharia de *Software* disponibilizou vários métodos, técnicas e ferramentas para auxiliar os desenvolvedores de *software* a produzirem com maior qualidade e menos tempo;
- Assim foram criados os diversos paradigmas da programação:
  - Programação “desestruturada”,
  - Programação Estruturada,
  - Programação Orientada a Objetos e
  - Programação Orientada a Aspecto.

Programação “Desestruturada”

- Lógica de programação difícil de ser entendida;
- Uso indiscriminado de comandos de desvio de fluxo (*GO TO*);
- Difícil de criar código reutilizável;
- Uma verdadeira “macarronada”, como denominam alguns autores.

Programação Estruturada

- Também conhecida como Programação Imperativa;
- Grande avanço na programação da época;
- Funcionalidades do *software* agrupadas em funções ou procedimentos. Que eram chamados a partir de um programa principal;
- Baniu-se o uso da instrução de desvio de fluxo (*GO TO*);
- Possibilidade de reutilização de código de maneira simples. Desde que devidamente projetados.

Programação Orientada a Objetos

- Com o advento da Programação Orientada a Objetos (POO), houve um salto na organização de sistemas. O mundo passou a ser **modelado** na forma de **classes** e objetos;
- O domínio do problema a ser resolvido caracteriza-se por classes de objetos, que possuem **atributos** e **comportamentos** e que se relacionam entre si;
- As classes podem ser reutilizadas com maior facilidade;
- A área de componentização, conceitos de pacotes e reusabilidade, deixaram a teoria e passaram a frequentar a realidade;
- Apesar destes avanços, o aumento da complexidade (nos sistemas) continua crescendo e tornando o desenvolvimento de sistemas uma atividade onerosa em termos de tempo, complexidade e pessoal capacitado a desenvolvê-los.

Programação Orientada a Aspectos

- A Programação Orientada a Aspecto (POA), tem o objetivo de tentar separar os níveis de preocupação durante o desenvolvimento de *software*;
- Assim sendo, com POA é possível pensar separadamente nos problemas referentes à persistência de dados, modelagem de negócios, segurança, distribuição, auditoria, *logs* e etc.
- A proposta da POA é desenvolver partes do sistema sem se preocupar com as demais partes;
- Cada parte do sistema, que se deseja trabalhar isoladamente pode ser chamada de “aspecto”;
- Por enquanto, seu uso está quase que totalmente restrito a área acadêmica.

Fundamentos das Linguagens Orientadas a Objetos

- A ideia básica da Orientação a Objetos (OO) é:
  - o *software* deve ser constituído por objetos que representem os objetos que compõem o mundo real;
  - O domínio do problema a ser resolvido caracteriza-se por classes de objetos, que possuem atributos e comportamentos e que se relacionam entre si;
  - O modelo utilizado no *software* orientado a objetos, está mais próximo do mundo real do que aquele utilizado no paradigma estruturado.

Fundamentos das Linguagens Orientadas a Objetos

- Para se construir um *software*:
  - Usando Programação Estruturada, devemos fazer a seguinte pergunta: “O que o sistema faz?”;
  - Usando Programação Orientada a Objetos, perguntaremos da seguinte forma: “Quais são as classes / objetos que compõem o sistema?”.

Questões-chave na construção de um *software* orientado a objetos

1. Identificar objetos:
   - O *software* deve refletir os objetos do mundo real, ou seja, os objetos estão no mundo real, basta identificá-los;
2. Descrever os objetos:
   - A descrição dos objetos deve representar suas características e comportamentos. O elemento que descreve os objetos são as classes.

Questões-chave na construção de um *software* orientado a objetos

3. Descrever as relações entre as classes de objetos:
   - O tipo de relacionamento existente entre dois objetos (por exemplo: é um, possui, compõe, etc.) deve ser representado diretamente no *software*.

Orientação a Objetos

- Os **objetos** encontrados são categorizados em **classes;**
- As classes constituem o núcleo de um programa orientado a objetos;
- Conclusão:
  - Um *software* OO, é constituído por classes que descrevem o comportamento e as características de objetos, que interagem entre si;
  - O programa principal tem a função de criar os objetos principais e iniciar a execução.

Bibliografia

- DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**; tradução Edson Furmankiewicz; revisão técnica Fábio Lucchini. 6. ed., São Paulo: Pearson, 2005.
- FERREIRA, Kecia Aline Marques. *Slides* da disciplina de Programação de Computadores II. CEFET-MG, 2009.
- RESENDE, A. M. P. de; SILVA C. C. **Programação Orientada a Aspectos em Java**. Rio de Janeiro: Brasport, 2005.