Programação Orientada por Objetos

Kecia Aline Marques Ferreira

2007

Programação Orientada por Objetos

Princípios, objetivos e filosofia

Princípios, objetivos e filosofia

- Fatores de qualidade de software
- Mudança de Paradigma
- A idéia da Orientação por Objetos

Fatores de Qualidade de Software

Fatores de Qualidade de Software

- Obter software de qualidade é um dos principais objetivos da Engenharia de Software.
- Glenford Myers [1975] escreveu que talvez o maior desafio daquela época na produção de software fosse criar e dar manutenção em sistemas de programação de grande porte.

E hoje?

Fatores de Qualidade de Software

- A qualidade de um software pode ser observada por fatores internos e fatores externos.

Bertrand Meyer, 1995.

- **Fatores externos**: ponto de vista do usuário. *Correção, robustez, extensibilidade, reusabilidade, eficiência, compatibilidade, facilidade de uso, portabilidade, integridade e verificabilidade.*
- **Fatores internos**: ponto de vista estrutural do software. Permitem atingir os fatores externos. *Modularidade, legibilidade, manutenibilidade.*

Fatores de Externos de Qualidade de Software

- **Correção**: característica do software que realiza as tarefas como foram definidas em sua especificação de requisitos.
- **Robustez**: um software é robusto se realiza as suas tarefas de forma correta mesmo quando submetido a condições anormais.
- **Extensibilidade**: característica de um software poder ser facilmente adaptado a inclusões e alterações de requisitos.
- **Reusabilidade**: característica de um software que pode ser reutilizado ao todo ou em parte por outros softwares.
- **Compatibilidade**: facilidade de se combinar o software com outros softwares. Essa característica é importante porque raramente um software é construído sem interação com outros softwares.

Fatores de Externos de Qualidade de Software

- **Eficiência**: refere-se ao bom uso que o software faz dos recursos de hardware, tais como memória e processadores.
- **Portabilidade**: é a facilidade de se utilizar o software em diferentes ambientes de hardware e software.
- **Verificabilidade**: é a facilidade de se preparar rotinas para se verificar a conformidade do software com os seus requisitos.
- **Integridade**: é uma característica relacionada à segurança de dados, programas e documentos. Integridade é a habilidade de proteger tais componentes contra acessos não autorizados.
- **Facilidade de uso**: também denominada usabilidade, é a facilidade com que o software pode ser aprendido e utilizado.

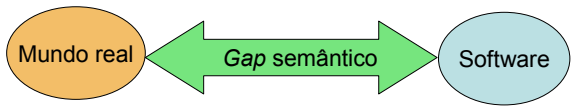
Fatores de Internos de Qualidade de Software

- **Modularidade**: característica de um software que é constituído por unidades denominadas *módulos.*
- **Legibilidade**
- **Manutenibilidade**: facilidade de realizar manutenção em um software.
- A forma com um software é construído permite atingir os fatores internos de qualidade.
- Os fatores internos de qualidade permitem atingir os fatores externos de qualidade.
- **A Orientação por Objetos é um paradigma cujas características permitem a obtenção de software *modular*, *legível* e de *fácil manutenção*.**

Paradigma Estruturado

Gap Semântico

- *Gap* semântico: é a distância entre as entidades do mundo real e o modelo computacional que o representa.
- A diminuição do *gap* semântico permite a melhor compreensão do software.



Paradigma Estruturado

- O paradigma estruturado é um método que se baseia na **função** do sistema.

  O que o sistema faz?
- O método baseia-se nas abstrações de ***comandos*** e ***expressões***.
- A construção *top-down* de um software neste paradigma constitui-se resumidamente dos seguintes passos:
  1. Identifica-se a função que o software deve desempenhar.
  2. A função genérica do software passa por refinamentos sucessivos, isto é, a cada passo detalham-se as funções do passo anterior em funções mais específicas.
  3. O refinamento é finalizado quando obtém-se um nível de detalhe apropriado para a implementação.

Paradigma Estruturado

- O método estruturado:
  - Promove a organização do pensamento sobre o problema a ser solucionado.
  - É uma forma de vencer a barreira da complexidade inicial do problema a ser solucionado.
  - É util para pequenos programas.

Paradigma Estruturado

- Desvantagens do método estruturado:
  - O método introduz dificuldade na evolução do sistema.
  - Não favorece o reúso.
  - Considera que é possível descrever todo software por uma única função. Porém, um software é melhor descritos pelos serviços que oferece.
  - O foco na função do software negligencia as estruturas de dados do problema.

Paradigma Estruturado

- Problema:

  Necessidade de métodos e técnicas que promovam fatores como modularidade, extensibilidade e reusabilidade.

- Solução:

  Orientação por Objetos

A Idéia da Orientação por Objetos

"Vivemos num mundo de objetos"
Pressman, 2002.

Orientação por Objetos

- A idéia básica da OO é: o software deve ser constituído por objetos que representem os objetos que constituem 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 por objetos está mais próximo do mundo real do que aquele utilizado no paradigma estruturado → redução do *gap* semântico.

Orientação por Objetos

- Questões chave na construção de software orientado por objetos:
  1. Como encontrar os objetos?
     - O software deve refletir os objetos do mundo real, ou seja, os objetos estão no mundo real, basta identificá-los.
  2. Como descrever os objetos?
     - A descrição dos objetos deve representar suas características e o comportamentos.
  3. Como descrever as relações entre os objetos?
     - O tipo de relacionamento existente entre dois objetos (por exemplo: é um, possui, compõe, etc.) deve ser representado diretamente no software.

Orientação por Objetos

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

Bibliografia

- Barnes, David e Kölling, M. *Programação Orientada a Objetos com Java*. São Paulo: Pearson Prentice Hall, 2004.
- Bigonha, R. S. e Bigonha, M. A. S. *Programação Modular.* Apostila. Belo Horizonte: DCC-UFMG, 2001.
- Ferreira, Kecia A. M. *Avaliação de Conectividade em Sistemas Orientados por Objetos.* Dissertação de Mestrado. Belo Horizonte: DCC-UFMG, 2006.
- MEYER, Bertrand. *Object-oriented software construction*. 2. Ed. Estados Unidos: Prentice Hall International Series in Computer Science, 1997. 1254 p.
- MYERS, Glenford J. *Reliable software through composite design*. Nova York: Petrocelli/Charter, 1975. 159 p.
- PRESSMAN, Roger S. *Engenharia de Software*. Rio de Janeiro: MacGraw Hill, 2002. 843 p.