PUC Minas

Resenha do Capítulo 6

Padrões de Projeto

Resenha do capítulo 6 do livro Engenharia de Software Moderna de Marco Tulio Valente

Júlia Medeiros Silva

Belo Horizonte, 2025

O Capítulo 6 do livro "Engenharia de Software Moderna" introduz o conceito de padrões de projeto, sua importância na flexibilidade e extensibilidade de sistemas de software. Inspirados nos estudos de Cristopher Alexander sobre arquitetura, os 23 padrões são divididos em três categorias: Criacionais, Estruturais e Comportamentais. Este capítulo detalha 10 padrões principais, abordando o contexto, o problema e a solução proposta por cada um deles.

6.1 Introdução

Os padrões de projeto são soluções reutilizáveis para problemas comuns no desenvolvimento de software. Eles ajudam a padronizar e facilitar a comunicação entre desenvolvedores.

6.2 Fábrica

O Padrão Fábrica encapsula a criação de objetos em um método específico, garantindo maior flexibilidade e promovendo o Princípio Aberto/Fechado.

6.3 Singleton

Singleton impede a criação de múltiplas instâncias ao tornar o construtor privado e fornecer um método estático para obter a instância única.

6.4 Proxy

O Proxy age como um intermediário, interceptando chamadas para adicionar funcionalidades sem modificar o objeto original.

6.5 Adaptador

O Adaptador cria uma camada intermediária que traduz chamadas de métodos entre as interfaces.

6.6 Fachada

Uma classe Fachada encapsula várias chamadas internas e fornece um único ponto de interação para o cliente.

6.7 Decorador

O Decorador encapsula o objeto original e adiciona funcionalidades por meio de composição.

6.8 Strategy

Define uma interface para algoritmos e permite que diferentes implementações sejam utilizadas dinamicamente.

6.9 Observador

Implementa uma relação um-para-muitos, onde os observadores são notificados sobre mudanças no sujeito.

6.10 Template Method

Implementa um método na classe base que chama métodos abstratos, permitindo que subclasses forneçam suas próprias versões desses métodos.

6.11 Visitor

Implementa um método accept() nas classes base, permitindo que diferentes visitadores adicionem novas funcionalidades.

6.12 Outros Padrões de Projeto

Iterador: Fornece um meio padronizado para percorrer coleções de objetos sem expor sua estrutura interna.
Builder: Facilita a criação de objetos complexos com muitos atributos opcionais.

6.13 Quando Não Usar Padrões de Projeto?

Embora padrões de projeto promovam flexibilidade, seu uso excessivo pode adicionar complexidade desnecessária, um fenômeno conhecido como paternite. Antes de adotar um padrão, é essencial avaliar se a flexibilidade oferecida justifica a introdução de novas classes e abstrações. A recomendação é sempre priorizar soluções simples e recorrer a padrões apenas quando sua necessidade é evidente.

Conclusão

O capítulo apresenta uma abordagem detalhada sobre padrões de projeto e como eles ajudam a estruturar melhor sistemas de software. Cada padrão resolve um problema específico e pode ser aplicado conforme as necessidades do projeto. No entanto, é fundamental utilizar esses padrões com moderação, garantindo que a complexidade adicional seja justificada pela flexibilidade e reutilização do código. Ao dominar esses conceitos, desenvolvedores podem criar sistemas mais escaláveis e manuteníveis.