Resenha Capítulo 6 - Padrões de Projeto

Neste Capítulo do livro Engenharia de Software Moderna, de Marco Túlio Valente, nos apresenta como os padrões de projeto devem ser utilizados como soluções recorrentes no desenvolvimento de software.

Através destes padrões, conseguimos acompanhar com mais facilidade o desenvolvimento do projeto. Conseguimos prever erros com mais facilidade, aumentar a escalabilidade do projeto sem dificuldades, tornamos o projeto mais flexível, melhoramos a comunicação entre os desenvolvedores devido a um vocabulário comum para soluções de design, etc.

Estas incríveis propostas são tragas em diversos padrões de projeto. No capítulo fomos apresentados a três categorias principais, Padrões Criacionais, Padrões Estruturais e Padrões Comportamentais. Dentre esses padrões, nos foram apresentados 10 padrões específicos.

Os Padrões Criacionais são focados na criação de objetos de maneira flexível, evitando a dependência direta de classes. Padrões que compõem esta categoria:

- **Fábrica (Factory)**: Permite a criação de objetos sem especificar suas classes concretas.

- **Singleton**: Garante que uma classe tenha apenas uma instância e fornece um ponto de acesso global a ela.

Os Padrões Estruturais tratam da composição de classes e objetos para formar estruturas maiores. Padrões que compõem esta categoria:

- **Proxy**: Atua como intermediário no acesso a outro objeto, controlando seu uso.
- **Adaptador (Adapter)**: Permite a interação entre interfaces incompatíveis.
- **Fachada (Facade)**: Fornece uma interface simplificada para um conjunto de interfaces de um subsistema.
- **Decorador (Decorator)**: Adiciona funcionalidades dinamicamente a um objeto sem modificar sua estrutura original.

Os Padrões Comportamentais lidam com a interação entre classes e objetos. Padrões que compõem esta categoria:

- **Strategy**: Permite que um algoritmo seja escolhido em tempo de execução.

- **Observador (Observer)**: Define uma dependência entre objetos, de forma que a mudança em um notifica automaticamente os outros.
- **Template Method**: Define o esqueleto de um algoritmo na superclasse, permitindo que subclasses personalizem etapas específicas.
- **Visitor**: Separa algoritmos da estrutura de objetos sobre os quais operam, permitindo adicionar novas operações sem modificar as classes originais.

Cada padrão é apresentado com um contexto prático, descrevendo o problema que resolve e como pode ser implementado. O capítulo também alerta contra o uso excessivo de padrões, um fenômeno conhecido como "paternite", que pode levar a soluções desnecessariamente complexas. Apesar deste alerta os padrões podem ser implementados em basicamente todos os casos de desenvolvimento de softwares presentes no cotidiano.

Em suma, o capítulo fornece uma visão abrangente e aplicada sobre os padrões de projeto, destacando sua importância para a qualidade do software e a eficiência no desenvolvimento.