Pontifícia Universidade Católica de Minas Gerais

Engenharia de Software

Resenha dos Capítulos 6 e 7

Engenharia de Software Moderna

Autor: Marco Tulio Valente

Aluno: Luiz Nery
Disciplina: Projeto de Software
Professor: João Paulo Aramuni

Belo Horizonte, 13 de outubro de 2024

Sumário

1 Resumo

Os capítulos 6 e 7 abordam conceitos fundamentais para o desenvolvimento de software, focando em padrões de projeto e arquitetura de software. O capítulo 6 discute diversos padrões que ajudam a resolver problemas recorrentes no design de sistemas, enfatizando a importância da reutilização de soluções. Já o capítulo 7 apresenta padrões arquiteturais que orientam a estruturação de sistemas complexos, destacando a necessidade de uma arquitetura bem definida para garantir a escalabilidade, manutenção e evolução das aplicações.

2 Capítulo 6: Padrões de Projeto

O capítulo 6 inicia com uma introdução aos padrões de projeto, destacando a importância de considerar as mudanças no design para evitar grandes reestruturações no futuro. Este conceito é fundamental, pois os sistemas de software estão sempre sujeitos a modificações e adaptações.

2.1 Seções 6.2 a 6.11: Padrões Específicos

Em seguida, o autor apresenta dez padrões de projeto, cada um dividido em três partes essenciais:

1. **Contexto**: Descreve o ambiente ou sistema onde o padrão pode ser aplicado. 2. **Problema**: Apresenta um desafio específico que pode surgir no projeto desse sistema. 3. **Solução**: Oferece uma solução para o problema utilizando o padrão em questão.

Os padrões discutidos incluem:

- **Fábrica**: Facilita a criação de objetos sem especificar a classe exata. - **Singleton**: Garante que uma classe tenha apenas uma instância e fornece um ponto de acesso global a ela. - **Proxy**: Fornece um substituto ou representante de um objeto para controlar o acesso a ele. - **Adaptador**: Permite a interação entre interfaces incompatíveis. - **Fachada**: Oferece uma interface simplificada para um conjunto de interfaces em um subsistema. - **Decorador**: Permite adicionar comportamento a objetos de forma dinâmica. - **Strategy**: Define uma família de algoritmos, encapsulando cada um e tornando-os intercambiáveis. - **Observador**: Permite que um objeto notifique outros objetos sobre mudanças em seu estado. - **Template Method**: Define o esqueleto de um algoritmo em uma operação, permitindo que subclasses preencham os detalhes. - **Visitor**: Permite adicionar novas operações a objetos sem alterar suas classes.

2.2 Seções 6.12 a 6.13: Considerações Finais

Na seção 6.12, o autor menciona brevemente outros padrões, reforçando a ideia de que existem muitas abordagens que podem ser utilizadas no design de software. Na seção final, 6.13, é destacado que, apesar dos benefícios dos padrões de projeto, seu uso não deve ser visto como uma solução mágica. O autor alerta para as situações em que a aplicação de padrões pode ser desnecessária ou contraproducente, enfatizando a necessidade de uma avaliação cuidadosa antes de sua adoção.

3 Capítulo 7: Arquitetura

O capítulo 7 começa com uma introdução ao conceito de arquitetura de software, definindo-a como a estrutura fundamental de um sistema e as relações entre seus componentes. A arquitetura é vista como um aspecto crítico no desenvolvimento de software, pois determina a qualidade, a escalabilidade e a manutenção do sistema.

3.1 Seções 7.2 a 7.4: Padrões Arquiteturais Principais

O autor discute vários padrões arquiteturais importantes, incluindo:

- **Arquitetura em Camadas**: Organiza o sistema em camadas que se comunicam entre si, facilitando a separação de preocupações. - **Arquitetura em Três Camadas**: Uma forma específica da arquitetura em camadas que inclui a camada de apresentação, a camada de negócios e a camada de dados. - **Arquitetura MVC (Model-View-Controller)**: Um padrão que separa a lógica de negócios da interface do usuário, promovendo a escalabilidade e a manutenção do código. - **Arquitetura baseada em Microsserviços**: Aborda a decomposição de um sistema em serviços independentes, cada um com sua própria funcionalidade. O autor explora o contexto histórico que levou ao surgimento desse padrão, bem como os benefícios, como a escalabilidade e a resiliência, e os desafios, como a complexidade de comunicação entre serviços.

3.2 Seções 7.5 a 7.8: Padrões para Escalabilidade e Anti-padrões

O autor também aborda dois padrões arquiteturais voltados para garantir escalabilidade e desacoplamento em sistemas distribuídos:

- **Filas de Mensagens**: Facilitam a comunicação assíncrona entre diferentes componentes do sistema. - **Publish/Subscribe**: Um padrão que permite a comunicação entre emissores e assinantes, promovendo uma maior flexibilidade e desacoplamento.

Na seção 7.7, são discutidos outros padrões arquiteturais relevantes, e na 7.8, é apresentado um exemplo de anti-padrão arquitetural, alertando os desenvolvedores sobre práticas que devem ser evitadas para garantir a integridade e a eficiência da arquitetura do software.

4 Conclusão

Os capítulos 6 e 7 fornecem uma base sólida sobre padrões de projeto e arquitetura de software, destacando a importância de ambos para o desenvolvimento de sistemas eficazes e sustentáveis. Enquanto o capítulo 6 oferece uma visão detalhada sobre a utilização de padrões para resolver problemas recorrentes no design de software, o capítulo 7 se concentra na estruturação dos sistemas, enfatizando a relevância de uma arquitetura bem pensada. Juntos, esses capítulos não apenas equipam os desenvolvedores com conhecimento técnico, mas também os encorajam a adotar uma abordagem crítica e reflexiva em suas práticas de desenvolvimento.