RESENHA CAPITULO 5 E 6  
  
Eu li os capítulos do livro Engenharia de Software Moderna e pude compreender conceitos essenciais para o projeto de software. O livro destaca a importância da decomposição, independência e abstração como pilares fundamentais para lidar com a complexidade de sistemas modernos. Um exemplo que me chamou a atenção foi a implementação de um compilador, que é dividido em módulos — analisador léxico, sintático, semântico e gerador de código —, ilustrando o princípio da modularização. Esse exemplo deixa claro como a divisão em partes independentes facilita o desenvolvimento e a manutenção do sistema.

Outro ponto relevante foi o estudo das propriedades fundamentais do projeto de software, como integridade conceitual, ocultamento de informação, coesão e acoplamento. Aprendi que, segundo Frederick Brooks, a integridade conceitual é essencial para garantir a consistência e a uniformidade do sistema, sendo necessário que uma liderança centralizada tome as principais decisões de projeto. Já o ocultamento de informação, um conceito de David Parnas, mostrou-se fundamental para a manutenção e o desenvolvimento paralelo, pois permite que as mudanças internas em um módulo não afetem o restante do sistema.

Os princípios SOLID também tiveram grande destaque, e entendi sua importância para manter o software mais flexível e de fácil manutenção. Esses princípios incluem responsabilidade única, aberto/fechado, substituição de Liskov, segregação de interfaces e inversão de dependência. Com eles, percebi como é possível reduzir o acoplamento entre os componentes e aumentar a coesão do sistema.

Além dos princípios de projeto, estudei métricas de código como linhas de código (LOC), falta de coesão entre métodos (LCOM), acoplamento entre objetos (CBO) e complexidade ciclomática (CC). Com essas métricas, aprendi a identificar e avaliar possíveis problemas de qualidade no código, como métodos muito longos ou com baixa coesão.

Por fim, explorei os padrões de projeto apresentados pelos autores da Gang of Four (GoF), que oferecem soluções reutilizáveis para problemas comuns no desenvolvimento de software. Achei muito interessante como cada padrão resolve uma situação específica. Por exemplo:

O padrão Factory facilita a criação de objetos sem que o código cliente precise conhecer sua implementação;

O Singleton garante que uma classe tenha uma única instância, útil para gerenciadores de logs;

O Proxy insere uma camada intermediária, auxiliando no controle de acesso e no cache de resultados;

O Adapter resolve o problema de incompatibilidade entre interfaces, facilitando a reutilização de classes existentes;

A Fachada oferece uma interface simplificada para sistemas complexos, reduzindo a dependência dos clientes em relação aos componentes internos;

O Decorator permite estender funcionalidades de objetos de forma dinâmica, evitando a criação de várias subclasses;

O Strategy possibilita a troca de algoritmos durante a execução, tornando o sistema mais flexível.

Concluo que esse estudo foi essencial para ampliar meus conhecimentos sobre boas práticas no desenvolvimento de software. Os conceitos e padrões abordados no livro não apenas aprimoraram minha visão técnica, mas também me ajudaram a compreender como construir sistemas mais organizados, modulares e fáceis de manter.