PUC Minas

Resenha do Capítulo 5

Princípios de Projeto

Resenha do capítulo 5 do livro Engenharia de Software Moderna de Marco Tulio Valente

Júlia Medeiros Silva

Belo Horizonte, 2025

O capítulo 5 do livro “Engenharia de Software Moderna” trata dos princípios fundamentais do projeto de software. Destaca a importância da decomposição dos problemas em partes menores e independentes. Ele aborda conceitos como integridade conceitual, ocultamento de informação, coesão e acoplamento, além de apresentar alguns princípios de projeto, como o SOLID.

5.1 Introdução

O autor inicia citando John Ousterhout, que define o projeto como uma tentativa de resolver um problema através da decomposição em pequenas partes e que são mais gerenciáveis. No contexto da engenharia de software, essa prática de decomposição é essencial para lidar com a complexidade dos sistemas modernos. De forma que cada parte possa ser desenvolvida independentemente, facilitando a manutenção e escalabilidade do software.

5.2 Integridade Conceitual

O conceito de integridade conceitual, proposto por Frederick Brooks, defende que um sistema deve ser coeso e consistente, evitando a acumulação de funcionalidades descoordenadas. Sistemas sem integridade conceitual apresentam interfaces inconsistentes e funcionalidades redundantes, o que dificulta a experiência do usuário e aumenta a complexidade do desenvolvimento.

5.3 Ocultamento de Informação

O ocultamento de informação é um princípio essencial para a modularização, garantindo que detalhes internos de implementação de uma classe ou módulo permaneçam ocultos de outros componentes do sistema. Esse conceito permite o desenvolvimento paralelo, aumenta a flexibilidade para mudanças e melhora o entendimento para novos desenvolvedores.

Os métodos getters e setters são frequentemente utilizados para controlar o acesso a atributos privados de uma classe. Getters permitem a leitura de valores privados, enquanto setters permitem sua modificação de forma controlada. No entanto, Eles não garantem o ocultamento de informação.

5.4 Coesão

Uma classe com alta coesão tem métodos e atributos bem definidos e voltados para um único propósito, facilitando a manutenção e reutilização do código. Já uma classe com baixa coesão tende a ter responsabilidades dispersas, tornando sua compreensão e modificação mais difíceis.

5.5 Acoplamento

O acoplamento define o grau de dependência entre classes. Um acoplamento aceitável ocorre quando uma classe usa apenas métodos públicos de outra classe estável. Já um acoplamento ruim ocorre quando mudanças em uma classe impactam diretamente outras classes.

5.6 SOLID e Outros Princípios de Projeto

Os princípios SOLID oferecem diretrizes para a manutenção e evolução do software. Os principais são:

Responsabilidade Única: Cada classe deve ter uma única razão para ser modificada.

Segregação de Interfaces: Interfaces devem ser pequenas e especializadas.

Inversão de Dependências: Classes devem depender de abstrações e não de implementações concretas.

Prefira Composição a Herança: Composição proporciona maior flexibilidade do que herança.

Princípio de Demeter: Classes devem interagir apenas com objetos diretamente relacionados.

Princípio Aberto/Fechado: Classes devem ser abertas para extensões, mas fechadas para modificações diretas.

Substituição de Liskov: Subclasses devem manter o comportamento esperado da classe base.

5.7 Métricas de Código Fonte

Métricas quantitativas são usadas para avaliar a qualidade de um projeto de software. Algumas das principais incluem:

Linhas de Código (LOC): Mede o tamanho do código-fonte, mas não deve ser usada isoladamente como indicativo de produtividade.
Falta de Coesão Entre Métodos (LCOM): Mede a falta de coesão de uma classe, indicando se os métodos trabalham sobre os mesmos atributos.
Acoplamento Entre Objetos (CBO): Mede o número de dependências estruturais entre classes.
Complexidade Ciclomática (CC): Mede a complexidade do código com base no número de estruturas de decisão, como if e loops.

Conclusão

O capítulo 5 apresenta princípios essenciais para o projeto de software, abordando desde a decomposição de problemas complexos até princípios modernos de design como SOLID. O entendimento e aplicação desses conceitos ajudam a construir sistemas escaláveis, flexíveis e de fácil manutenção, promovendo qualidade e sustentabilidade a longo prazo.