Livro Engenharia de Software Moderna

Capítulo 5

O projeto de software envolve decompor problemas complexos em partes menores e independentes. Esse processo combate a complexidade por meio de abstrações, facilitando a implementação e manutenção do sistema. Integridade conceitual garante coerência entre funcionalidades. O ocultamento de informação recomenda que detalhes internos de uma classe sejam escondidos, expondo apenas o essencial. A coesão indica que cada classe ou método deve ter uma única responsabilidade, melhorando o entendimento, a manutenção e o reuso do código. O acoplamento mede a dependência entre classes e deve ser mínimo, pois acoplamento é ruim para o software.

Os princípios SOLID e outros princípios auxiliam no desenvolvimento de um software bem estruturado. O princípio da responsabilidade única estabelece que uma classe deve ter apenas um motivo para mudar. A segregação de interfaces recomenda que elas sejam pequenas e adaptadas aos clientes. A inversão de dependências sugere que as classes dependam de abstrações. O princípio de prefira composição a herança alerta que a herança pode levar a um acoplamento forte e dificuldades de manutenção. O princípio de Demeter determina que uma classe deve se comunicar apenas com objetos diretamente relacionados a ela. O princípio aberto/fechado propõe que sistemas sejam abertos para extensões, mas fechados para modificações diretas. O último, a substituição de Liskov indica que subclasses devem manter o comportamento da classe base.

Existem métricas de qualidade. Linhas de código medem o tamanho do código. LCOM, ou falta de coesão entre métodos, mede se os métodos de uma classe. O CBO verifica quantas classes uma classe referencia. A complexidade ciclomática mede o número de caminhos lógicos no código. Esses princípios e métricas ajudam a criar softwares de fácil manutenção e expansão.