Resenha do Capítulo 5: Princípios de Projeto

O Capítulo 5 de "Engenharia de Software Moderna" por Marco Tulio Valente oferece uma análise aprofundada sobre os princípios que guiam o projeto de software, um dos estágios mais críticos no desenvolvimento de sistemas de software. Este capítulo é uma mistura de teoria e prática, explorando como essas diretrizes podem moldar sistemas de software mais eficientes, manuteníveis e escaláveis.

Introdução ao Projeto de Software

Valente inicia o capítulo com uma definição clara do que é o projeto de software, destacando que ele não é apenas uma atividade técnica mas também uma arte que requer equilíbrio entre criatividade e engenharia. Ele menciona que um bom design deve antever não apenas a funcionalidade mas também a manutenção e evolução do sistema ao longo do tempo. O autor utiliza citação de Ousterhout para enfatizar que o projeto é a busca por soluções que atendam a critérios de qualidade, como simplicidade, clareza, e capacidade de manutenção.

Propriedades do Projeto de Software

Integridade Conceitual: Marco explica que para um sistema ser compreensível e manutenível, ele deve possuir uma visão conceitual coerente. Ele dá exemplos de sistemas que sofreram com a falta de integridade, como interfaces de usuário de softwares complexos que apresentam funcionalidades semelhantes de maneiras desnecessariamente diferentes.

- Impacto na Manutenção: Discute como a falta de integridade pode levar a um aumento exponencial no custo de manutenção, à medida que novos desenvolvedores precisam aprender múltiplos conceitos para entender o mesmo sistema.

Ocultamento de Informação: Este conceito é detalhado como uma estratégia para fazer o software mais modular e flexível. Valente usa exemplos de sistemas operacionais e bibliotecas de software, onde o encapsulamento de detalhes implementacionais permite atualizações sem repercussões globais.

- Exemplos de Sistemas Reais: Ele menciona drivers de dispositivos como um caso prático onde o ocultamento de informação é fundamental para a manutenção e atualização sem afetar o funcionamento do sistema operacional.

Coesão: A coesão é abordada como a medida de quanta lógica relacionada está contida em um único módulo. Valente discute diferentes tipos de coesão, como coesão funcional, sequencial, comunicacional, etc., e como cada uma afeta a manutenibilidade do código.

- Métricas e Avaliação: Ele apresenta métodos para medir a coesão, como a análise de dependências internas de classes ou funções, e sugere ferramentas de análise estática para este propósito.

Acoplamento: O acoplamento é detalhado como a dependência entre módulos, com Valente explicando os perigos de um acoplamento excessivo. Ele discute tipos de acoplamento (de dados, de controle, etc.) e estratégias para minimizá-lo.

- Estratégias de Redução: Ele fornece práticas como a utilização de interfaces, injeção de dependências, e padrões de projeto para diminuir a interdependência entre módulos.

Princípios de Projeto

Responsabilidade Única: Valente aprofunda este princípio, explicando como a aplicação do mesmo pode evitar a criação de classes "Deus", que são difíceis de manter e testar, usando exemplos de refatoração de código.

Segregação de Interfaces: Ele detalha como interfaces grandes e monolíticas podem ser segmentadas para atender apenas às necessidades específicas dos clientes, promovendo mais flexibilidade e menos dependências desnecessárias.

Inversão de Dependências: Valente oferece uma explicação detalhada de como este princípio pode ser aplicado para criar sistemas mais testáveis e flexíveis, com exemplos de como frameworks populares implementam esta ideia.

Preferir Composição a Herança: Com uma discussão extensa, ele compara a composição com a herança, destacando os benefícios de composição para flexibilidade e a manutenção, especialmente em sistemas complexos.

Princípio de Demeter: Aqui, Valente explica como limitar o conhecimento entre objetos pode levar a sistemas mais seguros e menos suscetíveis a mudanças em cascata, com exemplos práticos de sua aplicação.

Aberto/Fechado: Ele descreve como sistemas podem ser projetados para serem extensíveis sem a necessidade de alterar o código existente, usando casos reais de frameworks e bibliotecas que seguem este princípio.

Substituição de Liskov: Valente aborda como este princípio garante que subclasses podem ser usadas em qualquer lugar onde a superclasse é esperada, discutindo implicações no design de APIs e contratos de interface.

Métricas de Projeto

Valente não só menciona as métricas mas também discute como elas podem ser utilizadas para avaliar e melhorar o design do software. Ele descreve:

- Métricas de Coesão: Como medir e interpretar a coesão para identificar áreas de melhoria no código.

- Métricas de Acoplamento: Ferramentas e técnicas para detectar e reduzir o acoplamento, incluindo análise de impacto de mudanças.

- Complexidade Cíclica: Como avaliar a complexidade do código para prever dificuldades em manutenção e testes.

Com isso, tenho que Marco Tulio Valente conclui o capítulo reforçando que o projeto de software é uma atividade contínua que deve ser revisada e refinada ao longo da vida do sistema. Ele incentiva a documentação do design, seja através de UML ou outras formas, como uma prática essencial para a comunicação entre desenvolvedores e para a sustentabilidade do software. O Capítulo 5 de "Engenharia de Software Moderna" é um resumo minucioso de princípios de projeto, oferecendo a nós leitores tanto uma base teórica quanto orientações práticas. Valente não apenas ensina os princípios mas também conecta esses conceitos com a realidade dos desafios enfrentados no desenvolvimento de software moderno.