Resenha do livro Engenharia de Software Moderna

Aluna: Ana Flávia de Carvalho Santos

Capítulo 6

O capítulo inicia com uma introdução aos padrões de projeto, soluções genéricas para problemas comuns em projeto de software. São discutidos dez padrões: Fábrica, Singleton, Proxy, Adaptador, Fachada, Decorador, Strategy, Observador, Template Method e Visitor.

O padrão Fábrica é usado para parametrizar a instanciação de objetos, sendo útil quando se deseja trabalhar com diferentes protocolos de comunicação. O padrão Singleton garante que apenas uma instância de uma determinada classe seja criada, sendo útil quando se deseja evitar a proliferação de objetos. No entanto, este padrão é controverso, pois pode ser usado para camuflar a criação de variáveis e estruturas de dados globais.

O padrão Proxy insere um objeto intermediário entre um objeto base e seus clientes, sendo útil para adicionar funcionalidades a um objeto sem que ele tome conhecimento disso. Já o padrão Adaptador converte a interface de uma classe para outra interface esperada pelos seus clientes, sendo útil quando se tem que usar classes prontas que não implementam a interface desejada.

O padrão Fachada oferece uma interface mais simples para um sistema, encapsulando as classes internas por trás dessa fachada. Este padrão é útil quando se deseja simplificar a interação com um sistema complexo. O padrão Decorador adiciona novas funcionalidades a uma classe base através de composição, em vez de herança, sendo útil quando se deseja adicionar funcionalidades de forma dinâmica e sem criar uma explosão combinatória de subclasses.

O padrão Strategy permite parametrizar os algoritmos usados por uma classe, sendo útil quando se deseja adicionar novas funcionalidades a uma classe base sem alterar seu código fonte. O padrão Observador define como implementar uma relação do tipo um-para-muitos entre objetos sujeito e observadores, sendo útil quando se deseja manter vários componentes sincronizados com o estado de um objeto.

O padrão Template Method especifica como implementar o esqueleto de um algoritmo em uma classe abstrata, deixando alguns passos pendentes para serem implementados nas subclasses. Este padrão é útil quando se deseja padronizar um modelo para um algoritmo que pode ser herdado pelas subclasses. Por fim, o padrão Visitor define como adicionar uma operação a uma família de objetos sem modificar as

classes desses objetos, sendo útil quando se deseja realizar uma operação em todos os objetos de uma estrutura de dados polimórfica.

O capítulo também discute outros padrões de projeto, como o Iterador e o Builder. O padrão Iterador padroniza uma interface para percorrer uma estrutura de dados, enquanto o padrão Builder facilita a instanciação de objetos que têm muitos atributos, alguns dos quais são opcionais.

Por fim, o capítulo discute quando não usar padrões de projeto. Embora os padrões de projeto possam tornar um sistema mais flexível, eles também têm um custo, como a necessidade de implementar classes adicionais. Portanto, a adoção de padrões de projeto deve ser cuidadosamente analisada. Além disso, o uso excessivo de padrões de projeto pode levar à "paternite", uma inflamação associada ao uso precipitado de padrões de projeto.

Capítulo 7

O capítulo inicia com uma introdução ao conceito de arquitetura de software, destacando sua importância como decisões de projeto de alto nível que influenciam o desenvolvimento de um sistema. O capítulo discute diversos padrões arquiteturais, começando pela Arquitetura em Camadas, um dos padrões mais usados que organiza as classes em módulos de maior tamanho, dispostos de forma hierárquica. Uma variação desse padrão é a Arquitetura em Três Camadas, comum na construção de sistemas de informação corporativos, dividindo o sistema em Interface com o Usuário, Lógica de Negócio e Banco de Dados.

O capítulo também aborda a Arquitetura MVC (Model-View-Controller), proposta para a construção de interfaces gráficas e que define a organização das classes de um /sistema em três grupos: Visão, Controladoras e Modelo. O conceito de Single Page Applications (SPAs) é apresentado como um tipo de sistema Web mais parecido com aplicações desktop, onde todo o código é carregado para o navegador, tornando a aplicação mais interativa e responsiva.

Microsserviços surgem como uma solução para o gargalo arquitetural enfrentado ao lançar novas releases de um produto de forma frequente. Em uma arquitetura de microsserviços, os módulos são executados em processos independentes, reduzindo as chances de que mudanças em um módulo causem problemas em outros módulos e permitindo a evolução mais rápida e independente de um sistema.

Arquiteturas Orientadas a Mensagens são apresentadas como uma solução onde a comunicação entre clientes e servidores é mediada por um serviço que provê uma fila de mensagens. Os clientes atuam como produtores de informações, inserindo mensagens na fila, enquanto os servidores atuam como consumidores de mensagens. Essa arquitetura oferece comunicação assíncrona, desacoplamento no espaço e no tempo.

O capítulo também discute Arquiteturas Publish/Subscribe, onde as mensagens são chamadas de eventos e os componentes da arquitetura são chamados de publicadores e assinantes de eventos. Publicadores produzem eventos e os publicam no serviço de publish/subscribe, que notifica os assinantes. Essa arquitetura oferece desacoplamento no espaço e no tempo, comunicação em grupo e notificação assíncrona.

Outros padrões arquiteturais são brevemente discutidos, incluindo Pipes e Filtros, Cliente/Servidor e Peer-to-peer. Por fim, o capítulo apresenta o anti-padrão arquitetural conhecido como "big ball of mud", que descreve sistemas nos quais qualquer módulo se comunica com praticamente qualquer outro módulo, resultando em uma explosão no número de dependências e tornando a manutenção do sistema muito difícil e arriscada.

Conclusão

Os capítulos 6 e 7 do livro Engenharia de Software Moderna oferecem uma visão abrangente sobre padrões de projeto e arquitetura de software, destacando sua importância para o desenvolvimento de sistemas robustos, flexíveis e de fácil manutenção.

Os padrões de projeto apresentados no capítulo 6, como Fábrica, Singleton, Proxy, Adaptador, Fachada, Decorador, Strategy, Observador, Template Method, Visitor e Iterator, são ferramentas valiosas para lidar com problemas recorrentes no desenvolvimento de software. Esses padrões ajudam a organizar o código, promover o reúso e melhorar a manutenibilidade do sistema. No entanto, o livro também alerta para os riscos da aplicação excessiva de padrões, que pode levar a uma complexidade desnecessária.

O capítulo 7 aprofunda a discussão em nível de arquitetura, apresentando padrões como Arquitetura em Camadas, Arquitetura MVC, Microsserviços, Arquiteturas Orientadas a Mensagens e Arquiteturas Publish/Subscribe. Esses padrões arquiteturais são criados para lidar com diferentes necessidades e desafios de

sistemas, como escalabilidade, desacoplamento, comunicação assíncrona e evolução independente de componentes. A análise das vantagens e desvantagens de cada padrão, bem como a discussão sobre o anti-padrão "Big Ball of Mud", gera uma reflexão para a escolha da arquitetura mais adequada para cada projeto.