Pedro Henrique Pires Rodrigues

O livro Engenharia de Software Moderna, por Marco Tulio Valente, em específico os capítulos 6 e 7, tratam-se de padrões de projeto e arquitetura de software respectivamente, sendo esses assuntos extremamente importantes para a atualidade no ramo de desenvolvimento. O capítulo 6, busca discorrer sobre os mais utilizados padrões de projetos dos mais diversos tipos, como uma espécie de guia, apresentando aos seus leitores alguns padrões a serem e, quando serem utilizados. Paralelamente, o capítulo sete trata sobre os padrões arquiteturais que vem sendo adotados ultimamente pelas empresas esclarecendo seu funcionamento e destacando suas principais características.

A começar pelo sexto capítulo do livro, este trata sobre diversos dos padrões mais usados, estando entre eles Fábrica, Singleton, Proxy, Adaptador, Fachada, Decorador, Strategy, Observador, Template Method e Visitor. Ele apresenta cada um deles individualmente, mas os subdivide da mesma forma em três partes sendo elas: contexto, problema e solução. A princípio ele apresenta uma situação em que esse padrão será futuramente útil, em cima do contexto apresentado ele evidencia um problema, e a partir do padrão de projeto em questão, explica como este pode ser solucionado.

Ao decorrer da leitura do material, é evidente a grande preocupação que o autor tem com os princípios SOLID, logo no primeiro item dos padrões de projeto (Fábrica), ainda no contextualização da situação, mostra que o código descrito no contexto possui um problema no qual ele não seguiria o princípio *“Open/Closed Principle”*, aberto para extensões, fechado para modificações. E ainda no solucionador do problema destaca o fato da solução ser aderente ao princípio *“Dependency Inversion Principle”,* inversão de dependência. Esse item do SOLID, de forma resumida, se trata de sempre dar preferências a depender de abstrações a classes concretas de fato, sendo elas muita das vezes representadas por interfaces.

Os exemplos dados ao longo do texto são simples e de fácil entendimento, o texto é consideravelmente curto, sendo muito objetivo, mas abordando todo o conteúdo necessário para uma fácil compreensão do padrão utilizado. A todo momento o autor evidencia as fontes das quais reafirmam o que está dizendo, trazendo mais credibilidade a si, além de também explicitar qual o princípio SOLID que é ou violado ou aderido por aquele código.

Ainda nesse capítulo o autor deixa uma seção exclusivamente para tratar sobre os momentos em que não devemos usar esses padrões, pois ainda que aumentem a abstração do código, também exigem uma maior complexidade de desenvolvimento. Nesse sentido, o principal ponto que ele reforça, é a questão se esse padrão será de fato útil ao código. Por exemplo, antes de se adotar o padrão Strategy por exemplo, é válido se perguntar antes se de fato os usuários vão precisar fazer uso de algoritmos alternativos. Pois caso contrário, os métodos poderiam ser simplesmente implementados na própria classe original, sem a necessidade de poluir as pastas com mais arquivos que na realidade, não possuem uma utilidade de fato.

Além disso, quando ao capítulo sete dessa vez, ele começa introduzindo o real conceito de arquitetura de software antes de apresentar os que serão de fato abordados no texto, esses que são Arquitetura em Camadas, Arquitetura MVC, Microsserviços, Filas de Mensagens e Publish/Subscribe.

O texto traz duas definições para o que seria essa tal arquitetura, sendo a primeira definição sugere que a arquitetura se concentra em componentes de maior escala, como pacotes, módulos e serviços, ao invés de se deter em classes individuais. Essa visão enfatiza a relevância dos componentes arquiteturais, que devem estar alinhados com os objetivos do sistema. Por exemplo, em um sistema de informações, um módulo de persistência é essencial, enquanto em um sistema de inteligência artificial, esse mesmo módulo pode ser considerado menos relevante, destacando a importância de compreender o contexto em que cada componente opera.

A segunda definição ampliada, atribuída a Ralph Johnson, coloca a arquitetura como um conjunto de decisões de design críticas que moldam o sistema. Essas decisões, uma vez tomadas, são desafiadoras de reverter e incluem escolhas de linguagem de programação e bancos de dados, que podem impactar severamente a evolução do sistema ao longo do tempo. O texto menciona que a arquitetura é mais do que a disposição de módulos; trata-se de escolhas que influenciam a durabilidade e a eficácia do sistema.

O debate Tanenbaum-Torvalds, foi um conflito de ideias sobre o sistema operacional Linux em que um dos lados afirmava que o monólito era o ideal enquanto o outro dizia que o software deveria ser subdividido em microkernels. Por mais que possa parecer uma discussão muito específica e talvez não diretamente relacionada ao trabalho do artigo, essa é uma boa forma para pensar na construção de aplicações em um geral, especialmente quando pensamos em micro serviços.

Quando a arquitetura em camadas, o livro faz diversas explicações citando protocolos de redes os usando para exemplificação, e finaliza dando mais detalhes a arquitetura de três camadas especificamente. Até esse momento estava uma explicação bem alto nível, um tanto difícil de interpretar ao que ele estava se referindo exatamente, até aparecer o diagrama. Nesse momento foi possível perceber que não falava de um código único, mas da comunicação entre cliente servidor e banco de dados, sendo basicamente essa as três camadas da arquitetura.

Ao que se refere a parte arquitetural como um todo, apesar de ser um excelente conteúdo, observei que muito do que era dito, eram padrões arquiteturais adeptos a década de 90 ou mesmo anterior. Sendo mais atual somente a parte dos micro serviços. Embora que divisões como cliente, servidor e banco de dados seja extremamente comum, é também uma forma muito simples de abordar isso na atualidade, talvez uma atenção maior em pontos de maior complexidade como serviços de cloud poderia ter soado mais interessante. Ainda sim, é sem dúvidas um excelente resumo de todas as arquiteturas nele citadas.

Ainda que haja alguns pontos a se discutir, o livro apresenta uma linguagem simples e direta, sendo um resumo sucinto de informações essenciais a todo o engenheiro de software. Em todo o momento a obra evidencia de onde as ideias vem, e sempre as justifica com base em princípios amplamente divulgados e bem aceitos pela comunidade. Acredito ser uma excelente obra para qualquer um que queira começar a deixar de ser um simples “pedreiro de código” e começar a virar um verdadeiro engenheiro de software.