Resenha capítulo 6 Engenharia de Software Moderna

Padrões de projeto

Padrões de projeto em software, como um conceito, tiveram sua ideia inicial vinda de um arquiteto chamado Christopher Alexander, que pensava que “Cada padrão descreve um problema que sempre ocorre em nosso contexto e uma solução para ele, de forma que possamos usá-la um milhão de vezes.”. No contexto de engenharia de software, pode se pensar em um padrão de projeto como “padrões de objetos que devem se inter relacionar para resolver um problema genérico em um contexto particular”, sendo essa uma definição cunhada por Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides, em seu livro Padrões de Projetos: Soluções Reutilizáveis de Software Orientados a Objetos, e defendiam que, era necessário, em etapas: Entender o problema que o padrão pretende resolver, o contexto em que o problema ocorre e a solução proposta. E para isso, neste mesmo livro foram propostos 23 padrões de projetos a serem adotados em soluções baseadas em software, e divididas nas categorias de Criacionais, Estruturais e Comportamentais.

Sobre os padrões criacionais, pode se dizer que são padrões relacionados com a criação de objetos de modo que fiquem flexíveis. O primeiro deles é o factory method, que por meio de uma interface define um padrão para a criação de um objeto, e que permite que suas subclasses alterem o tipo de objeto que vai ser criado, de maneira similar a uma fábrica que segue um determinado molde durante a fabricação do produto. Em resumo, pode se dizer que, o factory method delega a criação de objetos para uma subclasse, realiza uma abstração no período de instância e permite as suas subclasses criar uma implementação específica do objeto. Como por exemplo, em um jogo, onde uma classe "inimigo” serve como molde para a classe esqueleto e a classe zumbi. Sobre o padrão Singleton, ele propõe que determinadas classes deverão ter apenas uma instância, para que essa instância possa ser usada e reutilizada durante o código sem a necessidade de duplicatas, como por exemplo, não há necessidade de haver mais de uma classe que realiza ligações com o banco de dados da aplicação.

Outro tipo de padrão proposto na obra, é o padrão estrutural, que por sua vez cuida das soluções necessárias para a composição de classes e objetos. Dentre eles, aplicado por meio de uma interface, é o modelo adaptador, que faz com que uma determinada a ação possa funcionar com uma classe similar, porém distinta a aquela que o projeto comporta. Além do modelo adaptador, também há o padrão proxy, que por sua vez propõe que, para evitar de deixar uma classe *bloated* com muitas funções atreladas, se utiliza uma classe para fazer a intermediação, um proxy, como por exemplo, uma classe buscadora que busca dados de outra classe passada por parâmetro em um banco de dados. Também há a classe de fachada, que defende que é necessário haver uma classe de fachada para esconder do usuário classes internas do sistema. A classe decorator é responsável por adicionar funções por meio de composição a uma classe já existente, como um adorno responsável por realizar uma determinada ação na classe em que ele é colocado. Há o modo strategy,

que prega pelo encapsulamento de uma família de algoritmos para torná-los intercambiáveis e parte do princípio SOLID de aberto para expansão e fechado para mudanças. O método observador parte do ponto que a solução deve ser implementada de uma maneira um-para-muitos, de forma que várias classes possam “olhar” para uma classe e ser notificadas de quaisquer mudanças que ocorreram nela. O método template indica que deve se ter uma classe que sirva de esqueleto para classes herdeiras, e que contenha somente o estritamente necessário para servir de base as outras classes com quem irá compartilhar atributos e métodos.

Padrões de projeto tem como fim tornar projetos de software mais previsíveis, abertos a expansões e menos suscetíveis a refatorações, a fim de criar produtos mais robustos, estáveis e com menor custo de manutenção.