Resenha Engenharia de Software Moderna

Capítulo 7

Arquitetura

Quando se fala em arquitetura em engenharia de software, geralmente se trata do projeto em mais alto nível, que ao invés de realizar a modelagem de classes com seus atributos e métodos ela vai tratar de unidades de maior tamanho, sejam elas pacotes, componentes, módulos, subsistemas, camadas ou serviços. Porém, ter um maior tamanho não é suficiente, eles também devem ser pertinentes ao sistema, de um modo que atenda todos os seus objetivos. Além dessa definição para arquitetura de software, há outra igualmente popular, sendo cunhada por Ralph Johnson, que defende que a arquitetura de software trata das decisões mais importantes durante a fase de planejamento de um software, pois, uma vez que essas decisões forem implementadas em um projeto, elas dificilmente poderão ser retrabalhadas devido aos altos custos de refatoração de código. E dentre essas decisões incluem-se a definição dos módulos do sistema e outros aspectos técnicos que entram como requisitos não funcionais, tais como a definição do framework, banco de dados, linguagem de programação adotada, dentre outras.

Para essa organização de mais alto nível, foram propostas uma série de padrões arquiteturais, que irão cuidar dos módulos e de como eles serão organizados para solucionar problemas específicos de arquitetura. E dentre esses estilos, podemos citar o padrão Model-View-Controller, ou MVC, que contempla sistemas que possuem interface gráfica separando seus módulos em três diferentes tipos de pacotes, os responsáveis por servir de modelo (classes estruturais), os responsáveis pela visualização (interface gráfica [HTML, Swing, etc]) e por fim os responsáveis por realizar o controle do sistema.

Para sistemas que não necessariamente necessitam de uma interface gráfica integrada, como por exemplo, sistemas operacionais, podem se citar dois, o microkernel, proposto e defendido por Tanenbaum, que propõe que funcionalidades de sistemas operacionais sejam divididas de modo que cada uma seja responsável por uma parte da operação do sistema, diferente da arquitetura defendida por Linus Torvalds, o criador do sistema operacional Linux, que por sua vez defendia que todas as funcionalidades deveriam estar em apenas um único bloco. Essas diferentes visões sobre como um sistema deveria ser arquitetado gerou um acalorado debate em 1992, com sua sequência acontecendo em 2009, com Linus admitindo que “Não somos mais o kernel simples, pequeno e hiper eficiente que imaginei há 15 anos. Em vez disso, nosso kernel está ficando grande e inchado. E sempre que adicionamos novas funcionalidades, o cenário piora.” Se referindo ao estado bloated e de difícil manutenção que o kernel do Linux se encontrava na época, pois certas funcionalidades se encontravam nos amontoados de linhas de código que compunham a estrutura monolítica do sistema operacional do finlandês.

A arquitetura em camadas por sua vez é um dos padrões estruturais mais utilizados desde a sua concepção nos anos 70, e é responsável por ligar computadores em rede por meio de serviços que estão presentes em cada uma de suas camadas. Ele defende que sistemas devem ser projetados em camadas de modo que respeitem uma hierarquia entre elas, de modo que uma camada só consiga utilizar serviços de uma classe inferior. Pode se simplificar essa estrutura de modo que fique apenas com 3 camadas, sendo elas as camadas responsáveis, respectivamente, por ser a interface com o usuário, que por sua vez utiliza funcionalidades da camada de lógica, que acessa dados da camada responsável pelo banco de dados.

A arquitetura de micro serviços vem ganhando popularidade nos últimos anos, junto da popularização de métodos ágeis, pois, em sistemas monolíticos, realizar simples adições de funcionalidades pode se tornar um desafio pois, será similar ao tentar encontrar uma ponta em um novelo de lã, pois eventualmente ela será encontrada, porém, acabará gerando um custo de tempo maior no começo de cada sprint de desenvolvimento. Micro serviços por outro lado, separam o software em módulos e pacotes, de modo que fiquem bem encapsulados e que modificações sejam fáceis de serem realizadas sem causar problemas em outras partes do software. Além de ser um modelo mais robusto a falhas, pois, se uma parte do sistema cair, as outras continuam em operação.

Portanto, pode se concluir que a arquitetura de software no contexto contemporâneo da engenharia de software é parte integral do desenvolvimento, pois todo o processo de implementação utiliza algum padrão como base. Uma vez que os custos de tempo de planejamento eventualmente são superados em tempo de manuntenção, que é a parte mais cara e demorada de um projeto de software.