Aluno: Thomás Ramos Oliveira

The Hexagonal (Ports & Adapters) Architecture

A Arquitetura Hexagonal, também conhecida como Ports and Adapters, é um padrão de design que busca separar a lógica de negócio de uma aplicação de suas interações externas, como interfaces de usuário, bancos de dados e outros sistemas. Esse padrão surgiu como resposta a problemas recorrentes em aplicações, onde a lógica de negócio frequentemente se mistura com a interface ou com o acesso a dados, dificultando a manutenção, os testes automatizados e a integração com outros sistemas. A proposta central é que a aplicação seja desenvolvida e testada de forma isolada, permitindo que funcione sem a presença de interfaces ou bancos de dados reais, possibilitando testes automatizados e execução “headless”. Isso garante que a aplicação possa ser acionada por usuários, scripts de teste, batch scripts ou outros programas sem depender de elementos externos, aumentando a flexibilidade e a confiabilidade do sistema.

Um dos principais problemas enfrentados por sistemas tradicionais é a infiltração da lógica de negócio na interface de usuário, o que compromete a testabilidade, a adaptabilidade e a integração com outros sistemas. Quando a lógica está acoplada a elementos visuais instáveis, torna-se difícil implementar testes automatizados, migrar a aplicação para execução automatizada ou permitir que outros sistemas utilizem suas funcionalidades sem intervenção humana. A solução comum, de criar camadas adicionais na arquitetura, muitas vezes falha, pois não há mecanismos para impedir que a lógica de negócio seja incorporada às novas camadas, reproduzindo o problema original. A Arquitetura Hexagonal resolve isso ao garantir que todas as funcionalidades estejam acessíveis por meio de APIs definidas, permitindo que testes automatizados detectem rapidamente qualquer alteração que quebre funções previamente corretas. Além disso, possibilita que a aplicação funcione de maneira independente, isolando o núcleo da lógica de negócio de bancos de dados e interfaces instáveis, e facilitando a integração com outros sistemas, incluindo serviços web e aplicações empresariais.

A estrutura do padrão é baseada na separação entre o núcleo da aplicação e os elementos externos, utilizando **ports** e **adapters**. Os ports definem os pontos de comunicação e as APIs da aplicação, enquanto os adapters traduzem essas APIs para as tecnologias externas específicas, como GUIs, scripts de teste, serviços web ou bancos de dados. O formato hexagonal não se refere ao número de lados, mas sim à visualização da aplicação com múltiplos ports, permitindo que cada port tenha vários adapters substituíveis. Essa separação promove a testabilidade, pois é possível substituir componentes externos por mocks ou objetos simulados, garantindo que a lógica de negócio seja validada de forma isolada. Ports primários representam atores

que acionam a aplicação, enquanto ports secundários representam atores ou sistemas que a aplicação consulta ou notifica, reforçando a simetria e a modularidade da arquitetura.

Cockburn ilustra o padrão com um exemplo simples de aplicação para cálculo de desconto, mostrando como o sistema pode ser desenvolvido em etapas: inicialmente com testes automatizados e uma base de dados simulada, depois integrando a interface de usuário, e finalmente adicionando um banco de dados real ou outro tipo de repositório substituível. Essa implementação demonstra claramente o desacoplamento entre o núcleo da aplicação e os elementos externos, permitindo adicionar ou modificar adapters sem impactar a lógica central. A utilização de mocks e testes automatizados garante que a aplicação funcione corretamente mesmo quando elementos externos estão ausentes ou instáveis, e permite integrar sistemas diferentes de forma eficiente.

Os benefícios da Arquitetura Hexagonal incluem maior isolamento e testabilidade, flexibilidade na manutenção, facilidade de integração com outros sistemas e prevenção de vazamentos de lógica de negócio para camadas externas. Além disso, o padrão incentiva que casos de uso sejam escritos considerando apenas o núcleo da aplicação, sem detalhes tecnológicos externos, tornando-os mais claros, curtos e estáveis ao longo do tempo. O número de ports deve ser definido com equilíbrio, geralmente variando entre dois e quatro, representando diferentes tipos de interações, como usuários, bancos de dados, administradores ou sistemas externos. A distinção entre ports primários e secundários auxilia na organização de testes e na definição de adapters apropriados para cada tipo de interação.

Aplicações reais do padrão incluem sistemas de monitoramento climático que processam feeds de dados, notificam usuários e administradores, e interfaces de gerenciamento de assinantes, bem como aplicações web modernas, como Google Maps e Flickr, que expõem APIs para interação programática. Nesses casos, o padrão permitiu a substituição ou adição de adapters sem impactar o núcleo da aplicação, facilitando a manutenção e expansão do sistema. Projetos de desenvolvimento distribuído e de grandes equipes também se beneficiam, pois permitem que diferentes partes da aplicação sejam testadas de forma independente com mocks e FIT, integrando-se posteriormente com bancos de dados reais e interfaces definitivas. A Arquitetura Hexagonal também se relaciona com outros padrões, como Adapter, Model-View-Controller, Mock Objects, Loopback e Dependency Injection, reforçando conceitos de desacoplamento, modularidade e flexibilidade.

Em síntese, a Arquitetura Hexagonal oferece uma abordagem moderna e eficiente para desenvolvimento de software, especialmente relevante para aplicações complexas, distribuídas ou de longo prazo, onde a manutenção, testabilidade e integração são

preocupações centrais. Para estudantes de Engenharia de Software, compreender esse padrão é fundamental, pois permite projetar sistemas mais robustos, modulares e preparados para mudanças tecnológicas, evitando problemas históricos de acoplamento e dificuldade de testes, além de facilitar a evolução das aplicações sem comprometer sua estabilidade ou qualidade.