O artigo Hexagonal Architecture (Ports & Adapters) apresenta uma estrutura arquitetural destinada a facilitar o desenvolvimento, teste e evolução de aplicações de software, separando de forma clara o “núcleo” da aplicação ou seja, sua lógica de negócio e das interações com dispositivos externos, interfaces de usuário, bancos de dados e outros serviços.

A motivação central surge da constatação de que, historicamente, sistemas sofrem com a contaminação da lógica de negócio por detalhes de interface ou de persistência: quando a lógica está embutida no UI ou no banco de dados, torna-se difícil testar, reutilizar e evoluir a aplicação.

Para endereçar esse desafio, Cockburn propõe que a aplicação seja desenhada de dentro para fora: o núcleo da aplicação define “portos” (interfaces ou pontos de comunicação) através dos quais interage com o mundo externo. “Adaptadores” então conectam tecnologia específica a esses portos: por exemplo, um adaptador para interface gráfica, outro para testes automatizados, outro para chamadas de serviço remoto, outro para conexão com banco de dados, todos cumprem o papel de converter entre o protocolo tecnológico (GUI, HTTP, banco SQL) e a API do porto definida pela aplicação.

Essa arquitetura é frequentemente ilustrada como um hexágono para enfatizar que há vários portos ao redor da aplicação e que não se trata simplesmente de uma arquitetura em camadas verticais (UI->Lógica->Banco). Ao usar esse esquema, a aplicação torna-se igualmente dirigível por usuários, por scripts ou outros sistemas; e a persistência ou serviços externos tornam-se substituíveis e isoláveis, o que facilita testes, manutenção e evolução.

O artigo distingue ainda dois tipos de portos/adaptadores: os “primários” (driving adapters) que representam atores que acionam a aplicação (por exemplo, usuário, outro sistema) , e os “secundários” (driven adapters), que representam atores que são acionados ou consultados pela aplicação (por exemplo: banco de dados, serviços externos). Essa diferenciação ajuda a clarificar o fluxo de controle e a estruturar o design de forma que a lógica permaneça central e independente de detalhes de tecnologia.

Cockburn apresenta também exemplos de implementação, mostrando como se pode começar testando a aplicação isoladamente (com adaptadores de teste e mocks para banco de dados), adicionar posteriormente a interface GUI e por fim a persistência real, garantindo que cada etapa seja testável e modular. Esse caminho de evolução incremental destaca a aplicabilidade da arquitetura em ambientes ágeis ou onde os requisitos ou tecnologias podem mudar.

Em termos de consequências e benefícios, essa abordagem oferece maior testabilidade, flexibilidade tecnológica, menor acoplamento entre lógica e infraestrutura, e facilita a substituição ou evolução de componentes externos sem alterar a lógica de negócio central.

Por outro lado, exige disciplina no design das portas/adaptadores, atenção para a definição clara de interfaces e pode demandar um esforço inicial maior de abstração.

Em suma, o artigo de Alistair Cockburn propõe uma arquitetura que coloca a lógica de negócio em posição central, com a infraestrutura periférica como elementos plugáveis promovendo separação de preocupações, testabilidade e adaptabilidade tecnológica. Essa estrutura se mostra relevante para equipes que buscam desenvolver sistemas evolutivos, robustos e com boa capacidade de manutenção.