A Arquitetura Hexagonal (Ports and Adapters), proposta por Alistair Cockburn, busca desacoplar o núcleo de uma aplicação — onde reside a lógica de negócios — de seus elementos externos, como interfaces de usuário, bancos de dados e serviços externos. O objetivo é permitir que a aplicação funcione e seja testada independentemente desses componentes, tornando-a mais flexível, testável e resistente a mudanças tecnológicas. A ideia central é que o aplicativo se comunique com o mundo exterior por meio de “portas” (ports), que definem contratos ou interfaces, e “adaptadores” (adapters), que traduzem as interações entre a aplicação e as tecnologias específicas.

O principal problema que essa arquitetura resolve é a mistura de lógica de negócio com código de infraestrutura, como o da interface gráfica ou do acesso a dados. Essa mistura dificulta testes automatizados, impede a troca de componentes e gera alto acoplamento. A solução propõe separar claramente o “dentro” (a lógica da aplicação) do “fora” (interfaces e sistemas externos). Assim, a aplicação comunica-se apenas através de APIs ou protocolos definidos, sem saber detalhes sobre o tipo de interface (gráfica, linha de comando, web, etc.) ou de armazenamento de dados (banco relacional, arquivo, memória, etc.).

A representação em forma de hexágono é apenas uma metáfora visual para destacar que uma aplicação pode possuir múltiplos pontos de entrada e saída (portas), e não apenas duas, como nos tradicionais modelos em camadas. Cada porta pode ter diferentes adaptadores — por exemplo, um adaptador HTTP, um teste automatizado, ou um mock de banco de dados — todos conectados à mesma interface interna. Essa abordagem permite rodar a aplicação de modo “headless” (sem interface), usar bancos simulados durante o desenvolvimento e facilitar a integração entre sistemas sem intervenção humana.

Por fim, a arquitetura hexagonal se relaciona com diversos padrões conhecidos, como Adapter, Model-View-Controller (MVC), Mock Objects e Dependency Injection. Ela complementa esses conceitos ao enfatizar a independência da lógica central em relação a qualquer tecnologia externa. Seu uso tem se mostrado eficaz em contextos que exigem testes automatizados, integração contínua, desenvolvimento distribuído e evolução tecnológica rápida. O resultado é um sistema mais modular, sustentável e adaptável a longo prazo, mantendo a lógica de negócio intacta enquanto os detalhes de interface e infraestrutura podem mudar livremente.