



Hexagonal Architecture (Ports & Adapters)

Hexagonal Architecture Overview

What is Hexagonal Architecture?

- Hexagonal Architecture is an architectural pattern that encourages the separation of concerns. It's also known as Ports and Adapters Architecture.

Key Components

- **Ports:** Interfaces that represent the primary ways of interaction (usually Application and Domain layer interfaces).
 - Primary ports correspond to the use cases the application should fulfill.
 - Secondary ports represent the operations the application needs from external concerns like databases and third-party services.
- **Adapters:** Actual implementations that adapt a port to a specific technology or use-case.
 - Inbound adapters adapt incoming requests to a port (e.g., REST API, GraphQL, CLI).
 - Outbound adapters adapt port operations to external systems (e.g., Database, Message Queue).
- **Core Application:** The business logic and domain model sit at the center of the hexagon. It remains isolated from the outside world, which interacts with it through ports and adapters.

Advantages

- **Decoupling:** Easy to change one part without affecting others.
- **Testability:** Makes it easier to write unit tests as you can easily mock the ports.
- **Flexibility:** Can plug in different adapters without changing the business logic.
- **Maintainability:** Clean separation of concerns leads to easier maintenance.

Disadvantages

- **Complexity:** Can be overkill for simple applications.
- **Learning Curve:** Takes time to understand and implement correctly.
- **Potential for Overengineering:** Easy to get carried away, creating unnecessary ports and adapters.

Comparison with Onion and Clean Architecture

Onion Architecture

- Similar to Hexagonal but focuses more on the domain model being at the center.
- Compare and contrast with Hexagonal: there are similarities in decoupling and testability, differences mainly in layering and focus on the domain model.
- More layers, focuses on the dependency inversion principle at each layer, not just at the edges like Hexagonal.

Clean Architecture

- Proposed by Robert C. Martin, aims for separation of concerns among rules, database, and UI.
- A broader architectural pattern, encompasses more than just the structure of the application, includes considerations for deployment, UI, etc.

Summary and Takeaways

- Hexagonal Architecture promotes decoupling, testability, and flexibility.
- Onion and Clean Architecture offer similar benefits but with slight variations in focus and implementation.