Software Architecture
Course's Code: CSE 483
Domain Centric Architecture
(Chapter 6)

Chapter 6

Chapter 6. Domain Centric Architecture

6.1 What is domain-centric architecture?

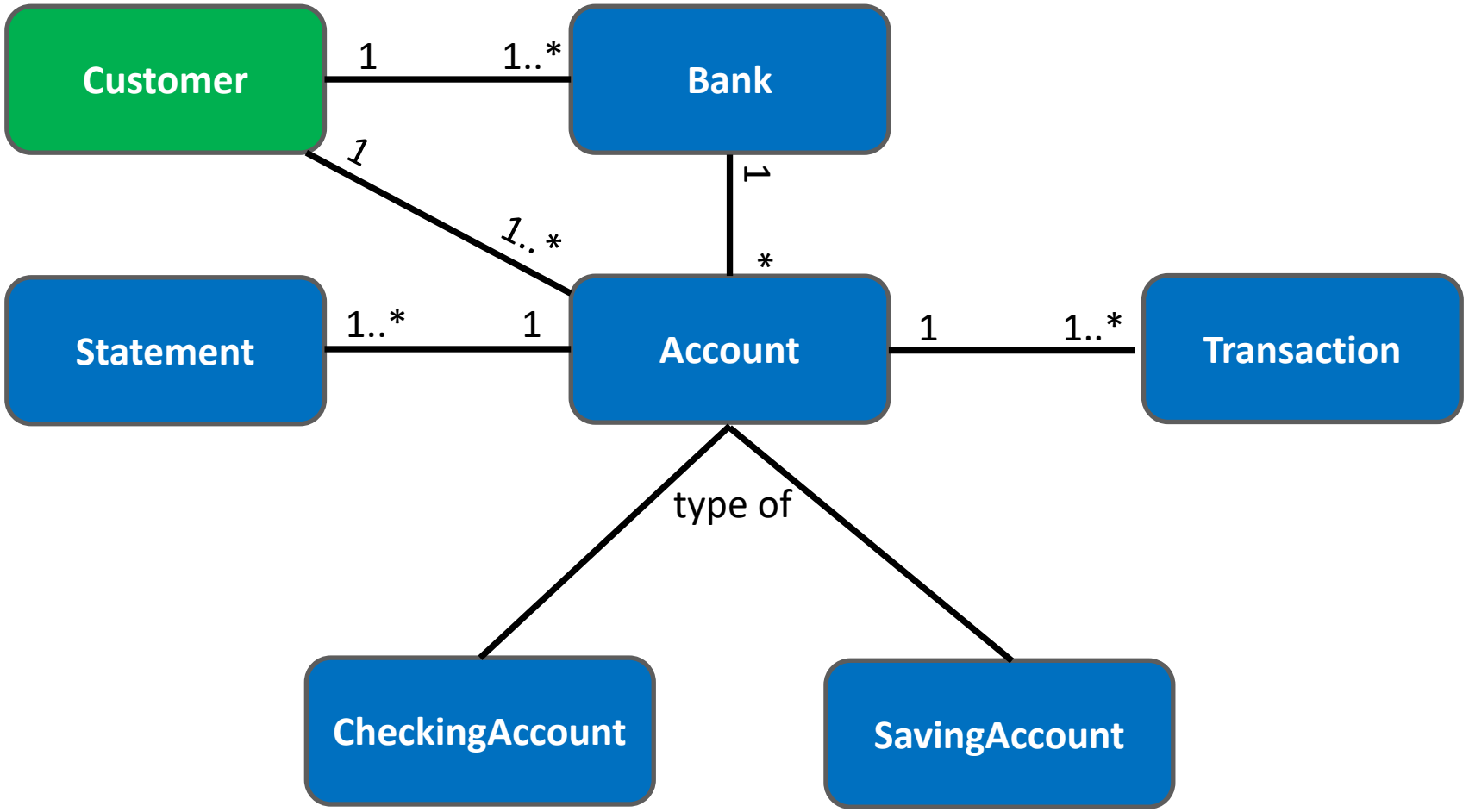
6.2 Modern four-layer architecture?

6.3 Types of domain architecture?

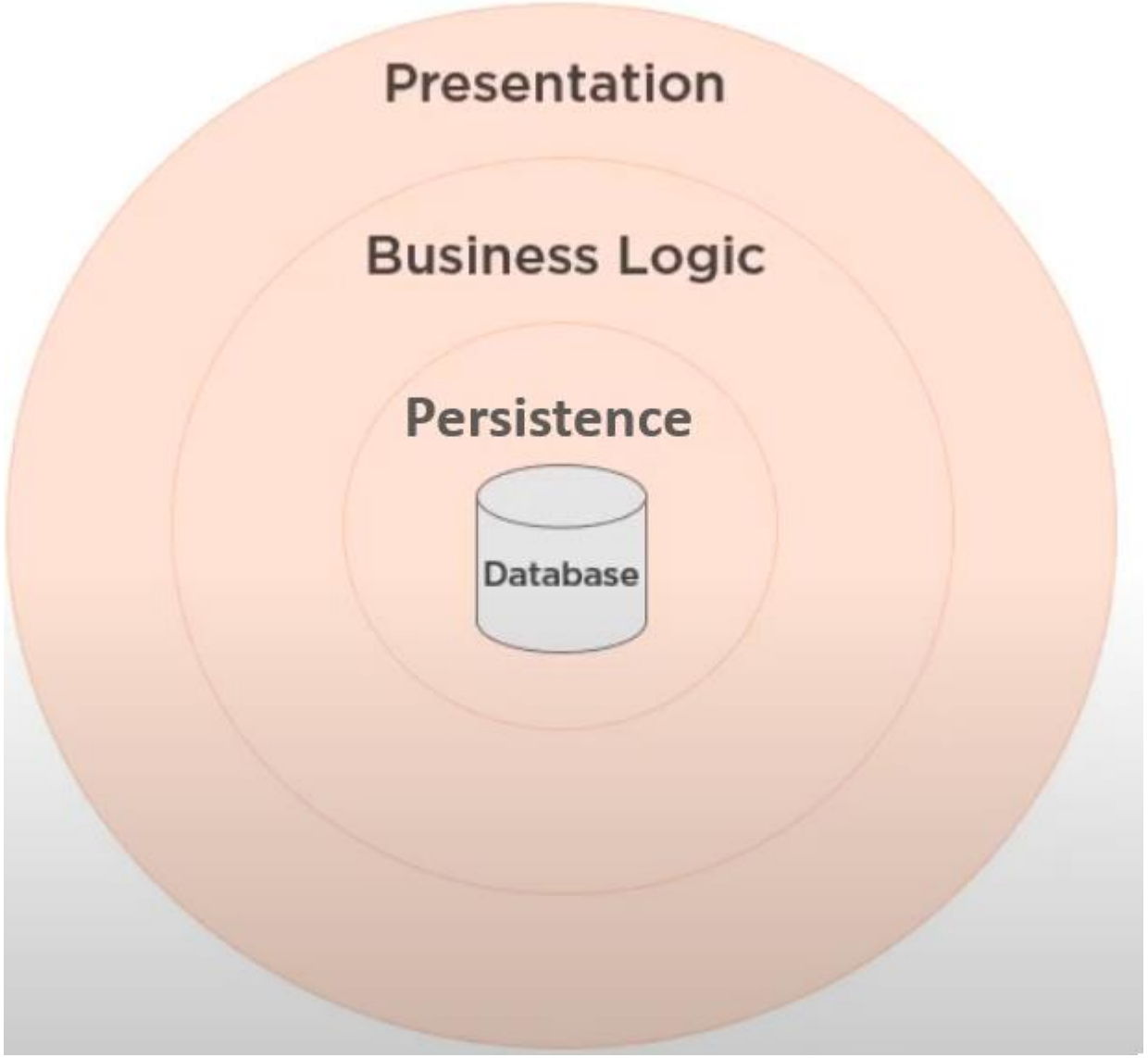
6.4 Advantages and disadvantages

What is Domain?

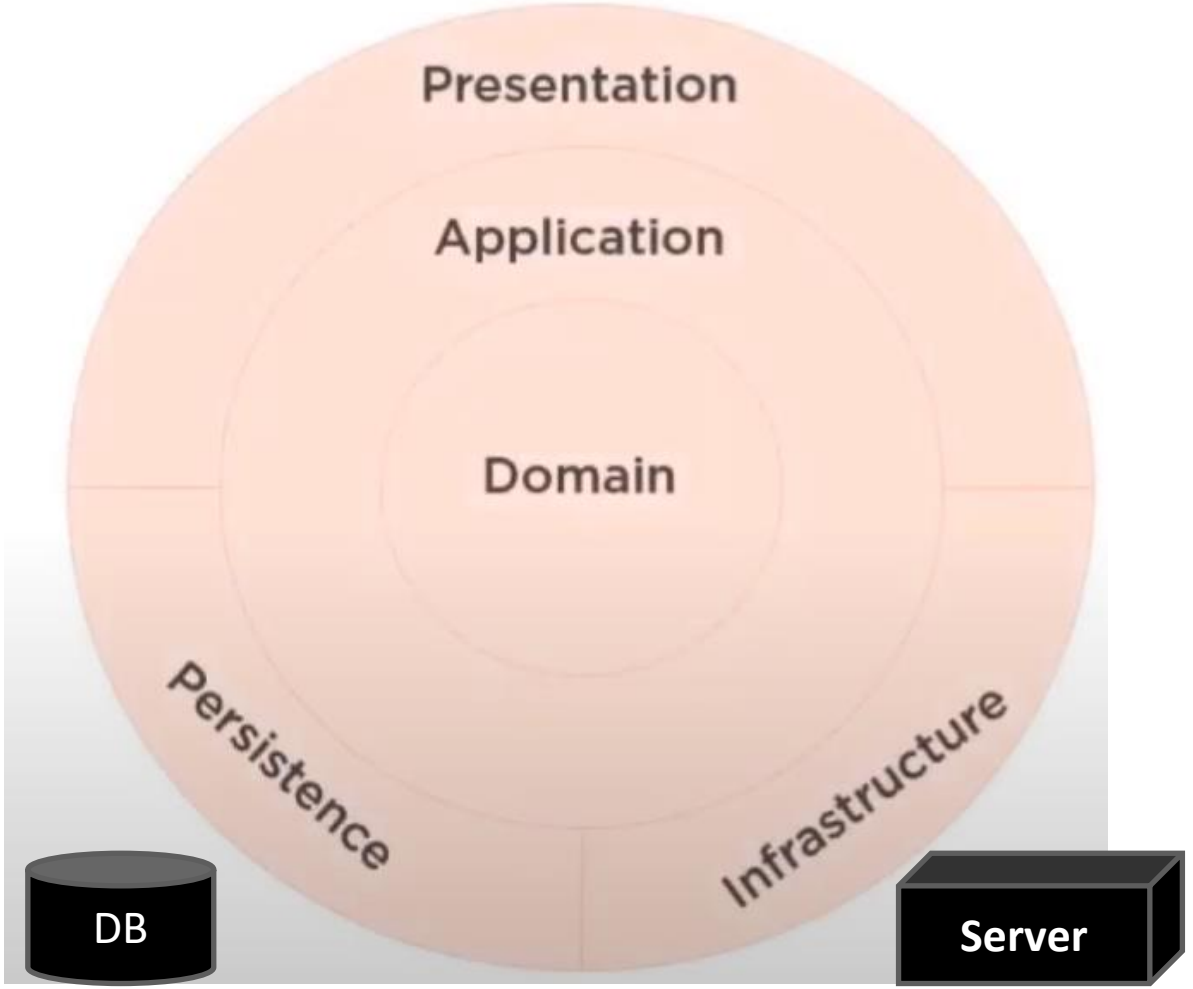
Retail Banking – Domain Model



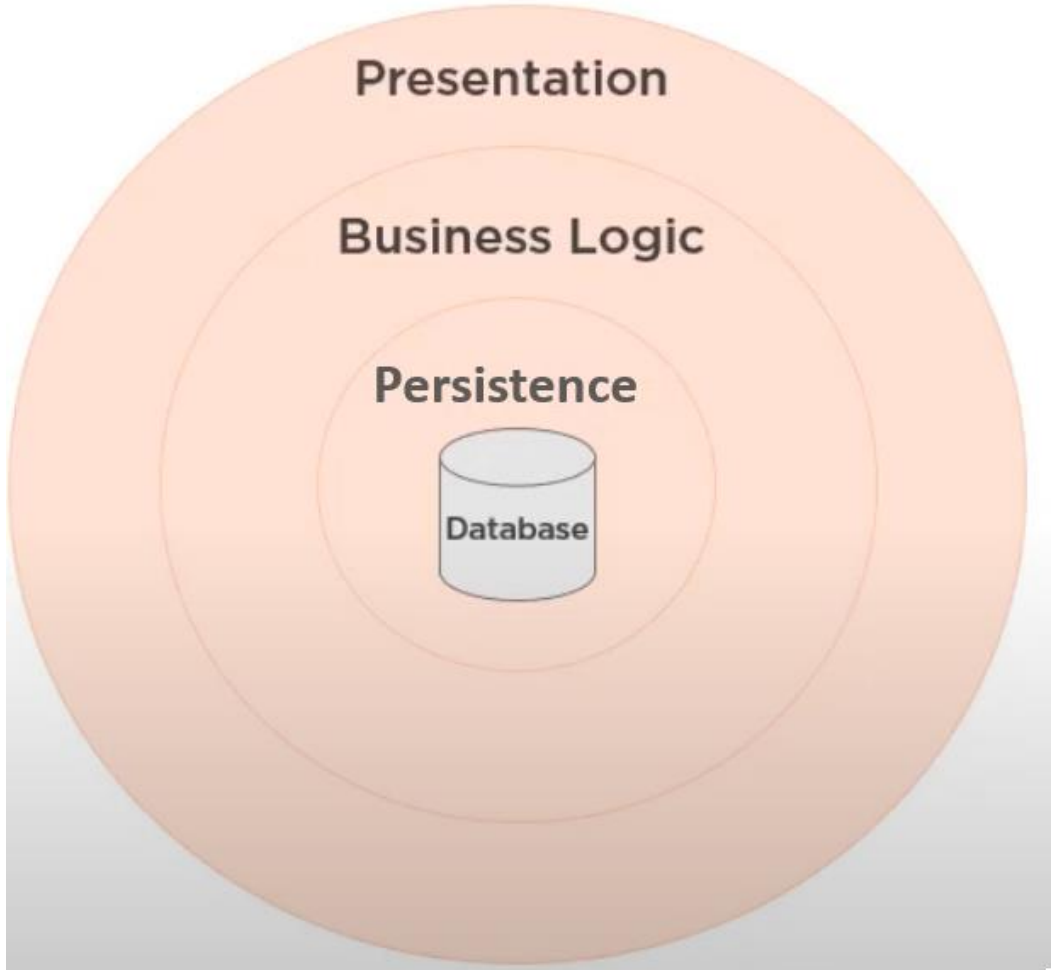
Classic Three-Tier Architecture

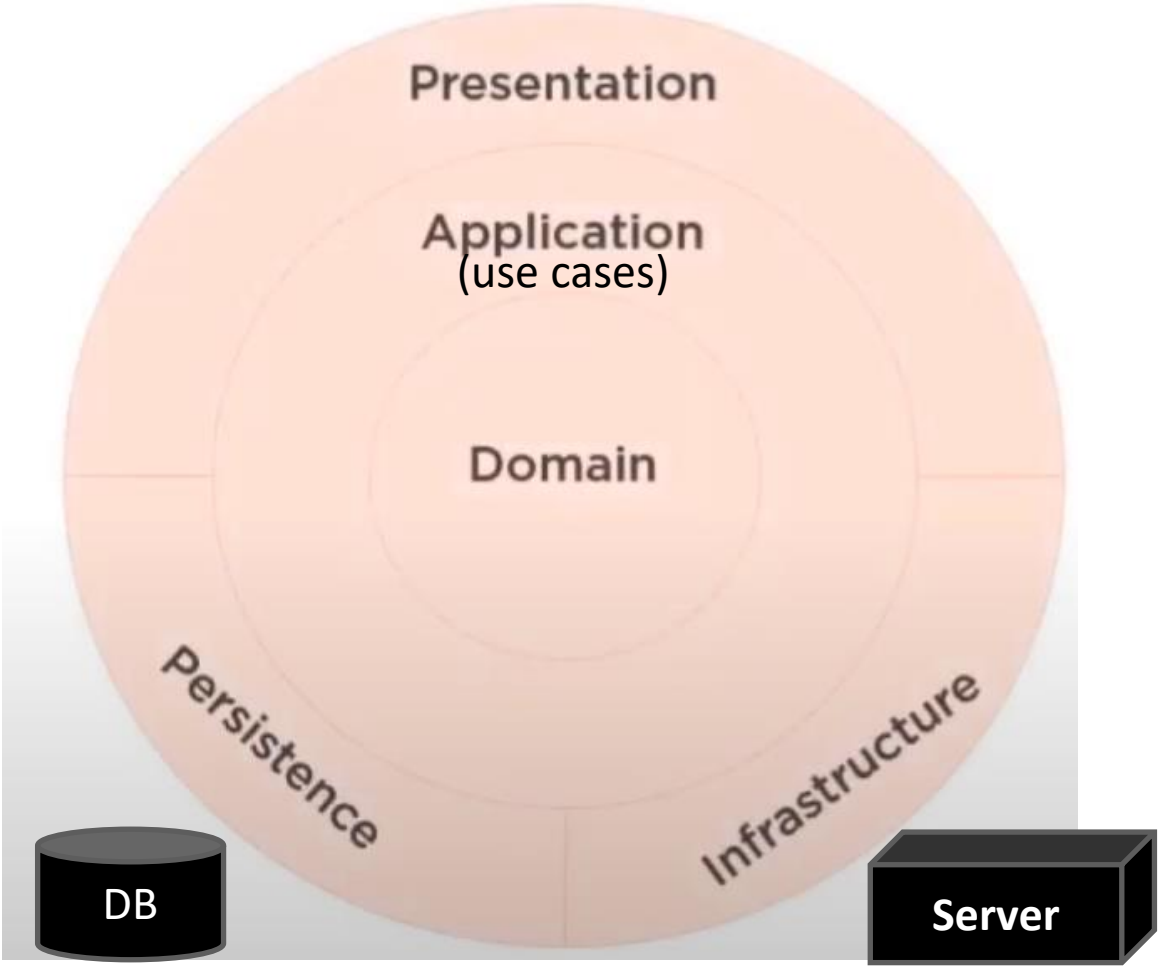


What is Domain Centric Architecture?



Database-Centric vs. Domain-Centric Architecture





Essential vs.Detail

Space is essential

Usability is essential

Building material is a detail

Ornamentation is a detail

Essential vs.Detail

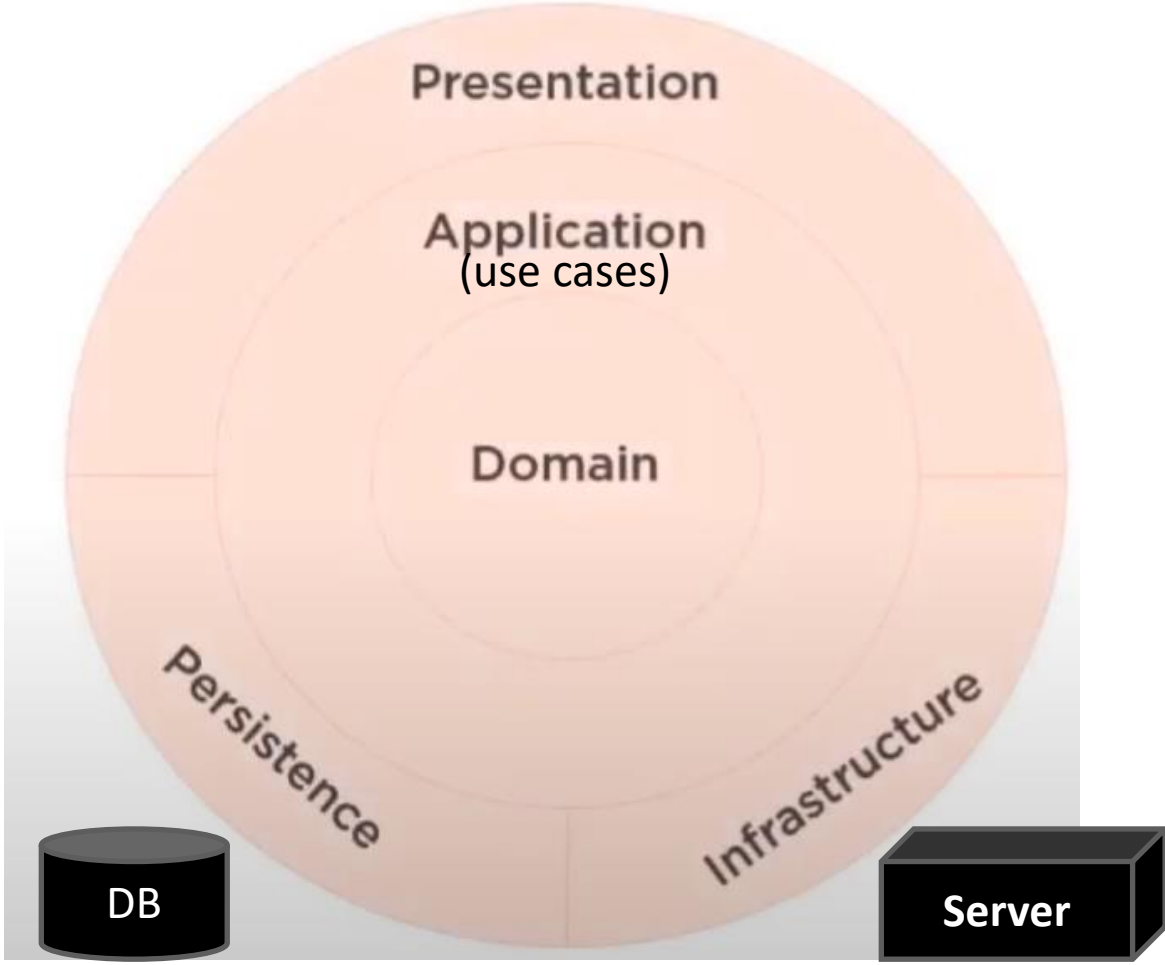
Domain is essential

Use cases (Business) is essential

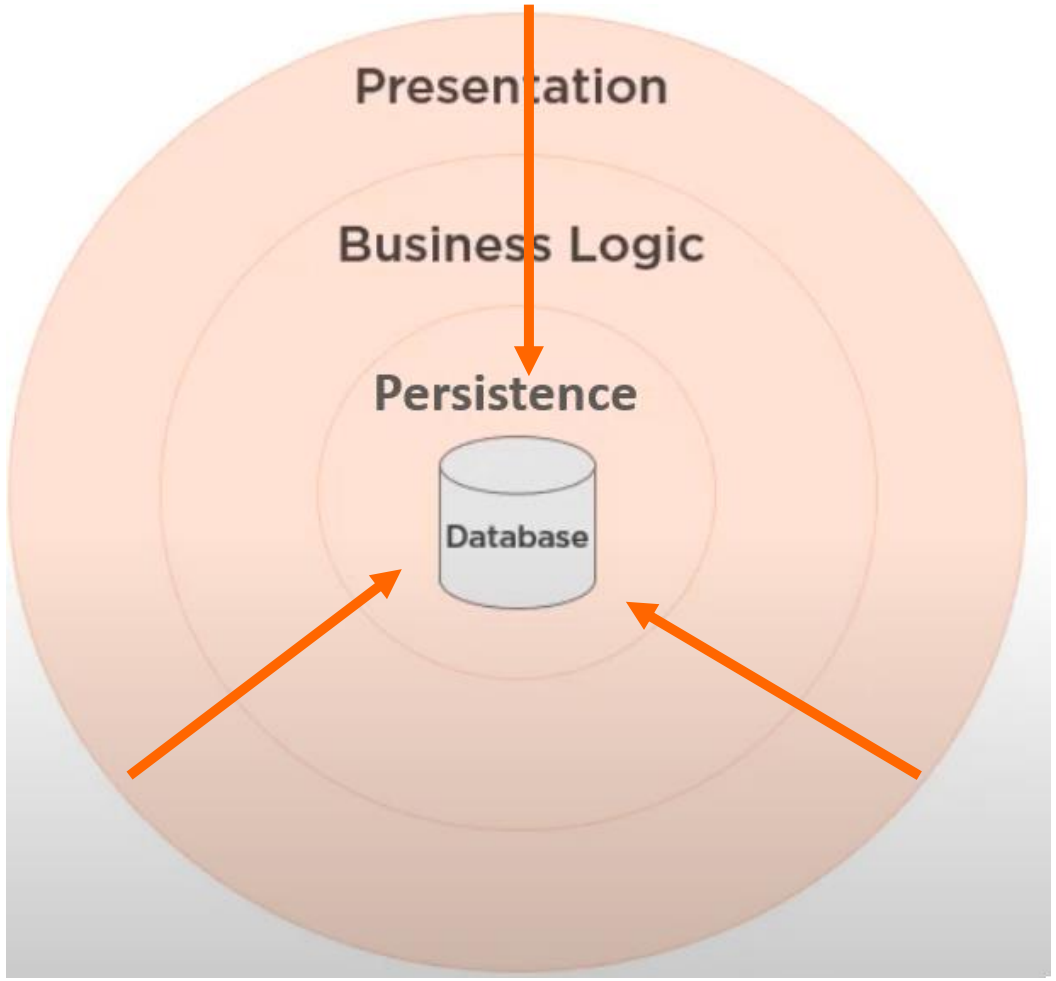
Presentation is a detail

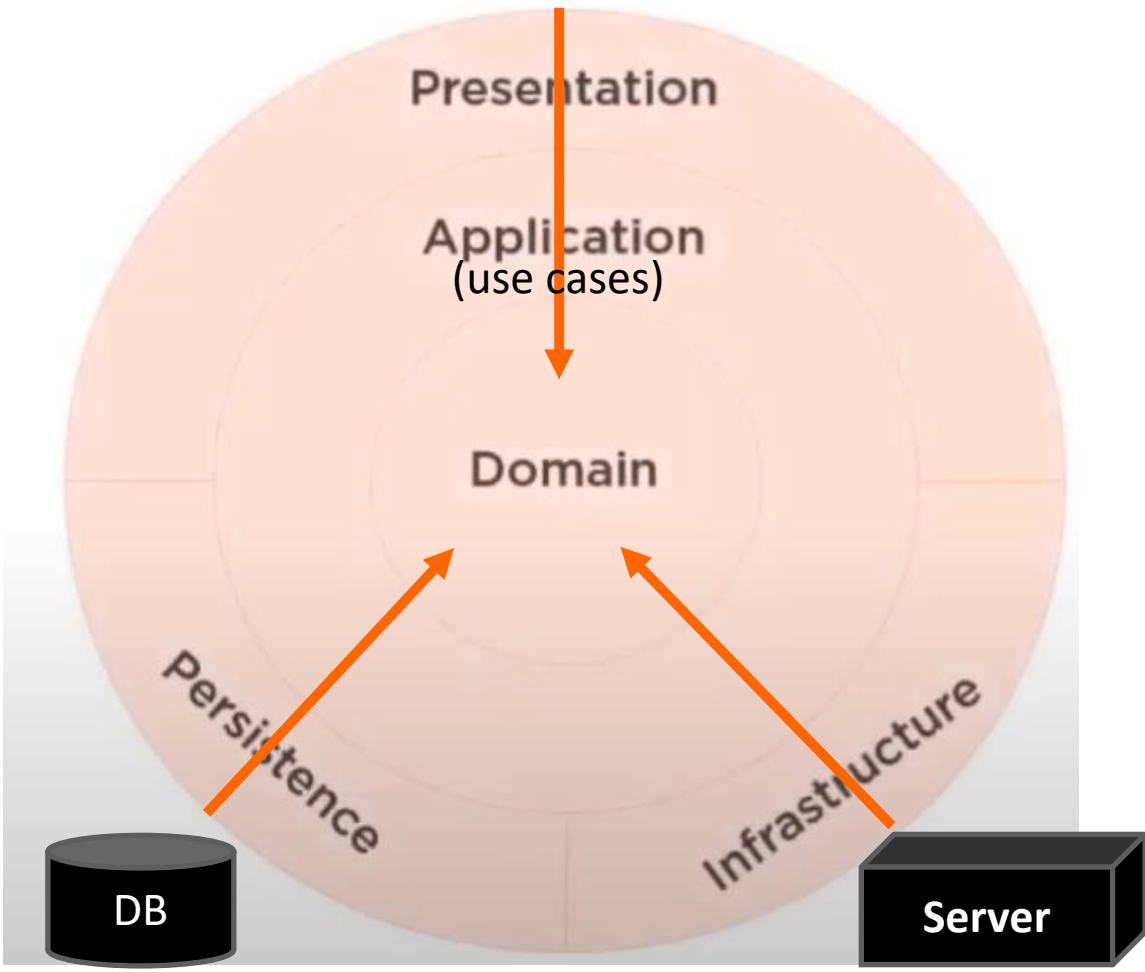
Persistence is a detail

Infrastructure is a detail



Database-Centric vs. Domain-Centric Architecture

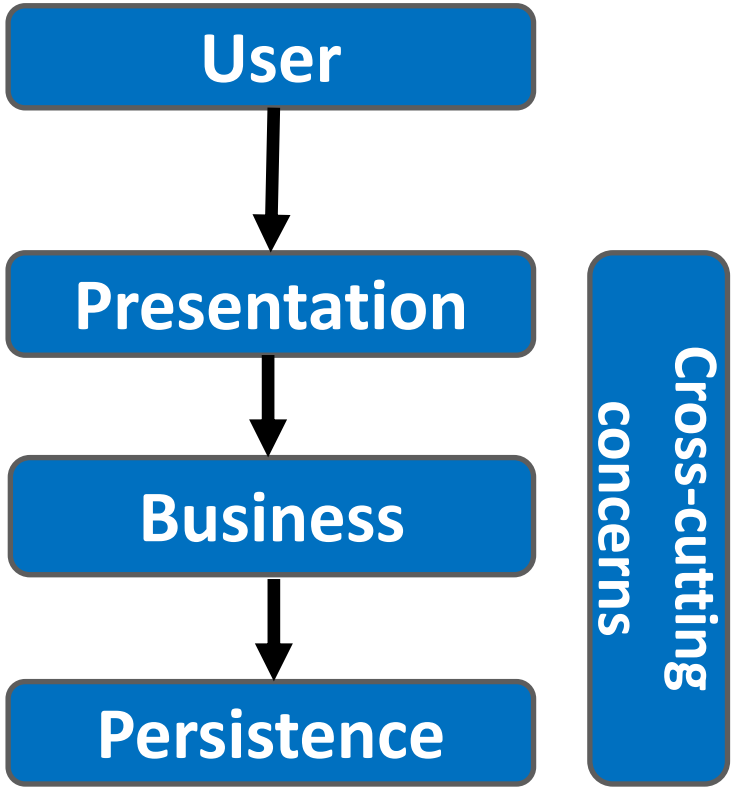


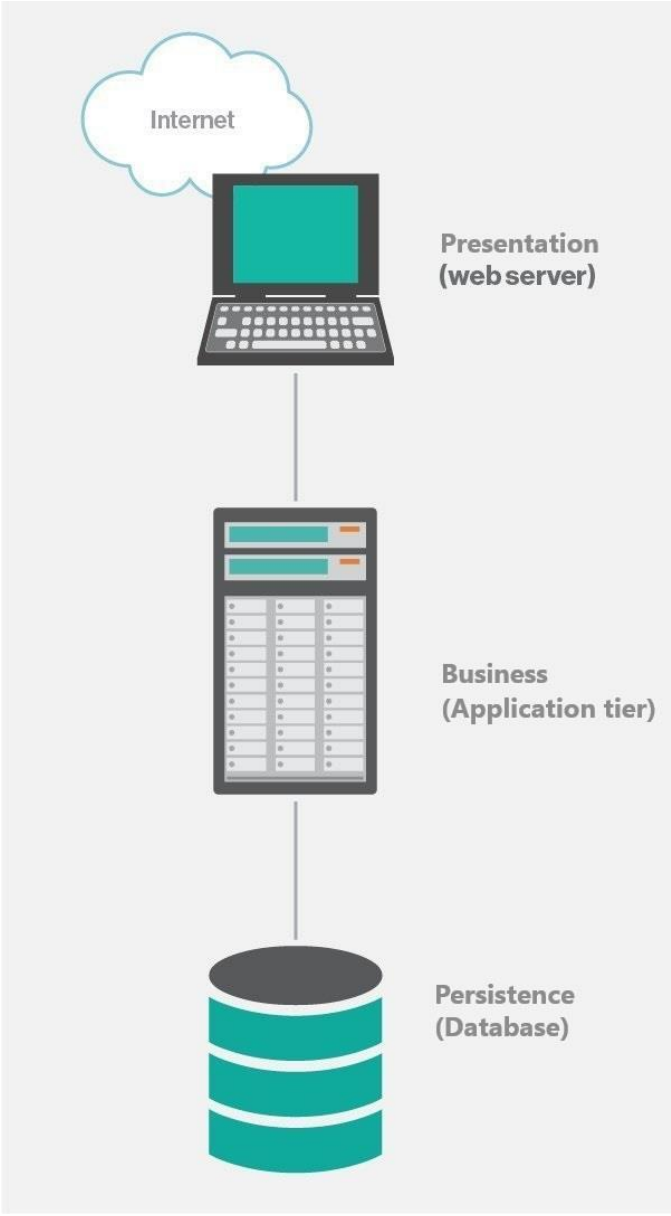


Chapter 6. Domain Centric Architecture

Modern Four-Layered Architecture

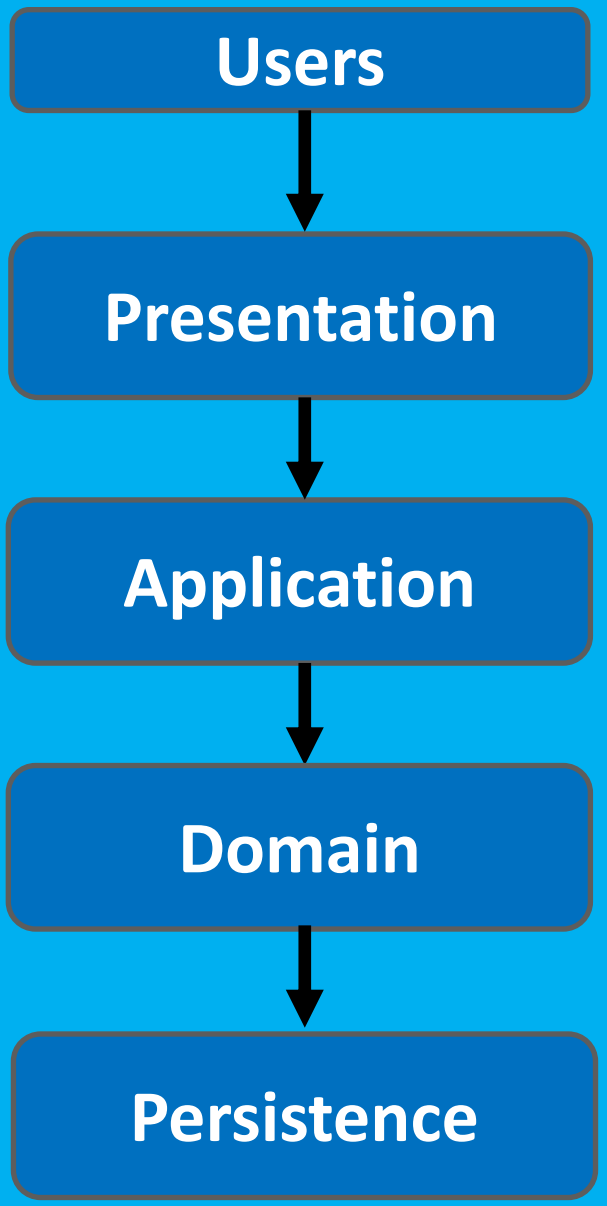
- Disadvantages of three-tier architecture
- Where should abstraction be located?





Modern Four-Layered Architecture

- Similar to three-tier architecture
- Application encapsulates use cases
- Application layer dependent on domain
- Domain layer contains business rules
- Persistence layer gateway to database



Modern Four-Layered Architecture

Advantages

- Separation of concerns
- Straightforward
- Tooling
- Consistency

Modern Four-Layered Architecture

Advantages

Maps to other architecture

Facilitates dependency injection

Modern Four-Layered Architecture

Disadvantages

Hidden purpose

It’s not very scalable

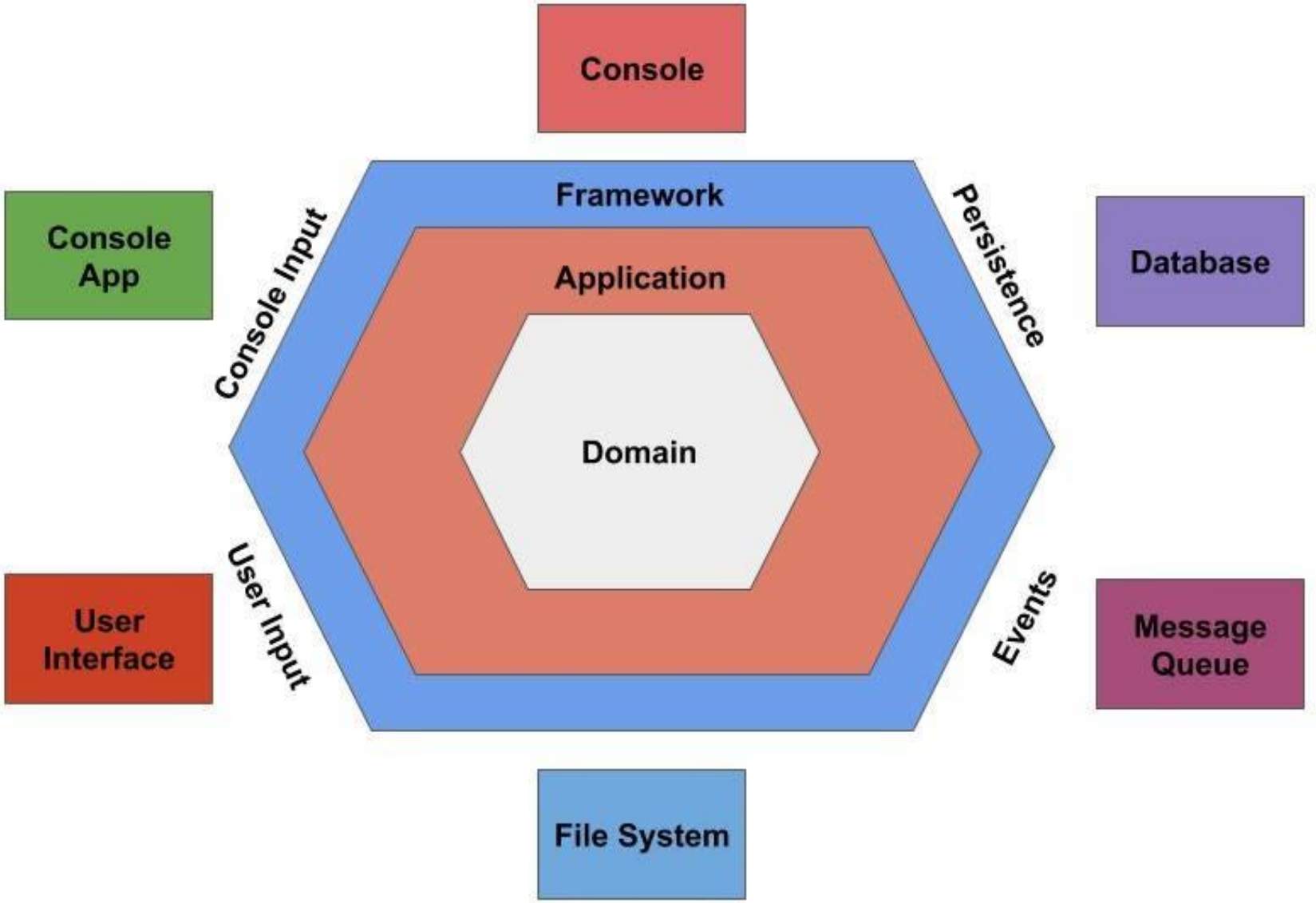
Chapter 6. Domain Centric Architecture

Types of Domain Centric Architecture

Three Domain-Centric variants

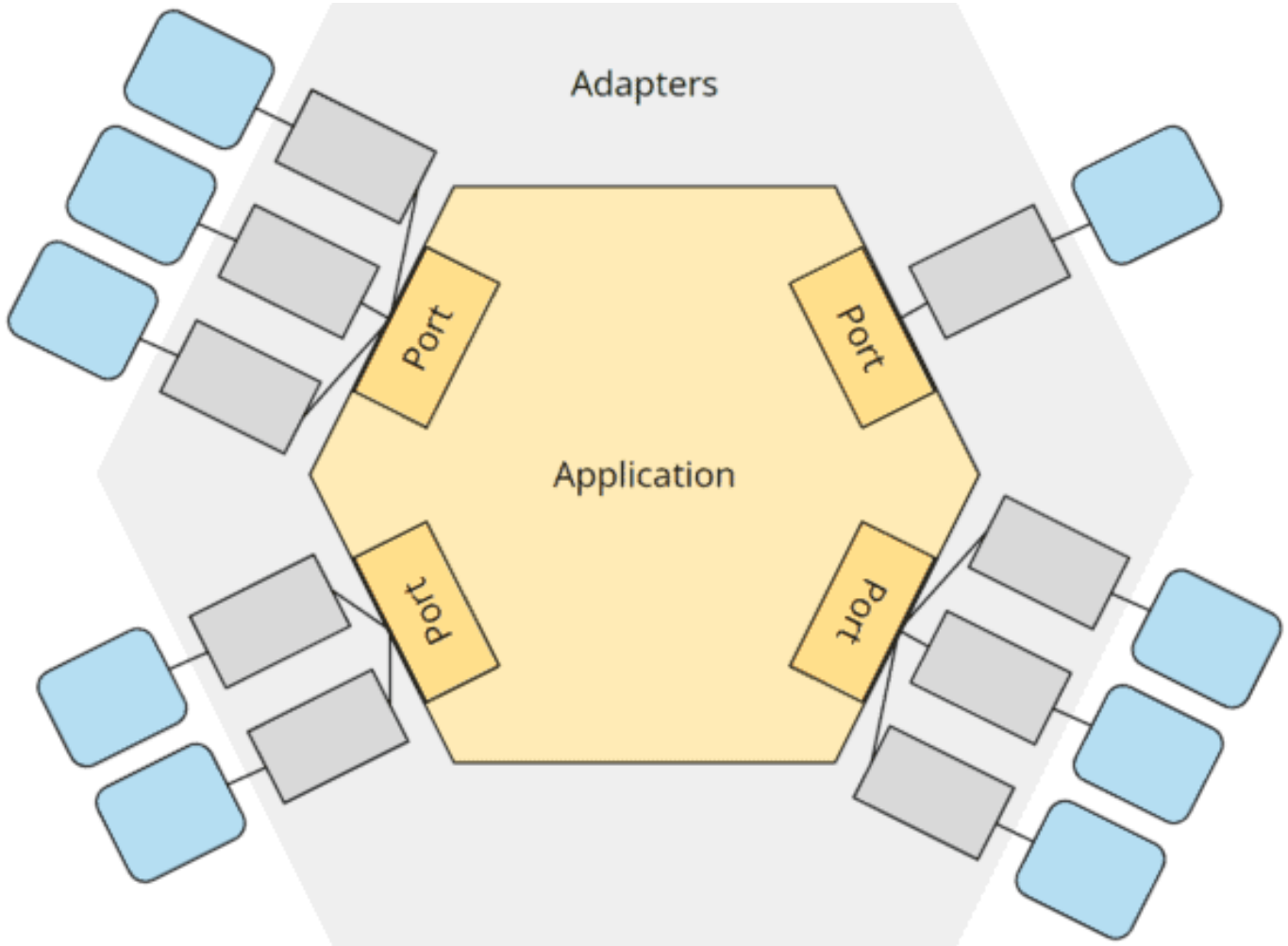
1. Hexagonal architecture
2. Onion Architecture
3. Clean Architecture

Hexagonal Architecture



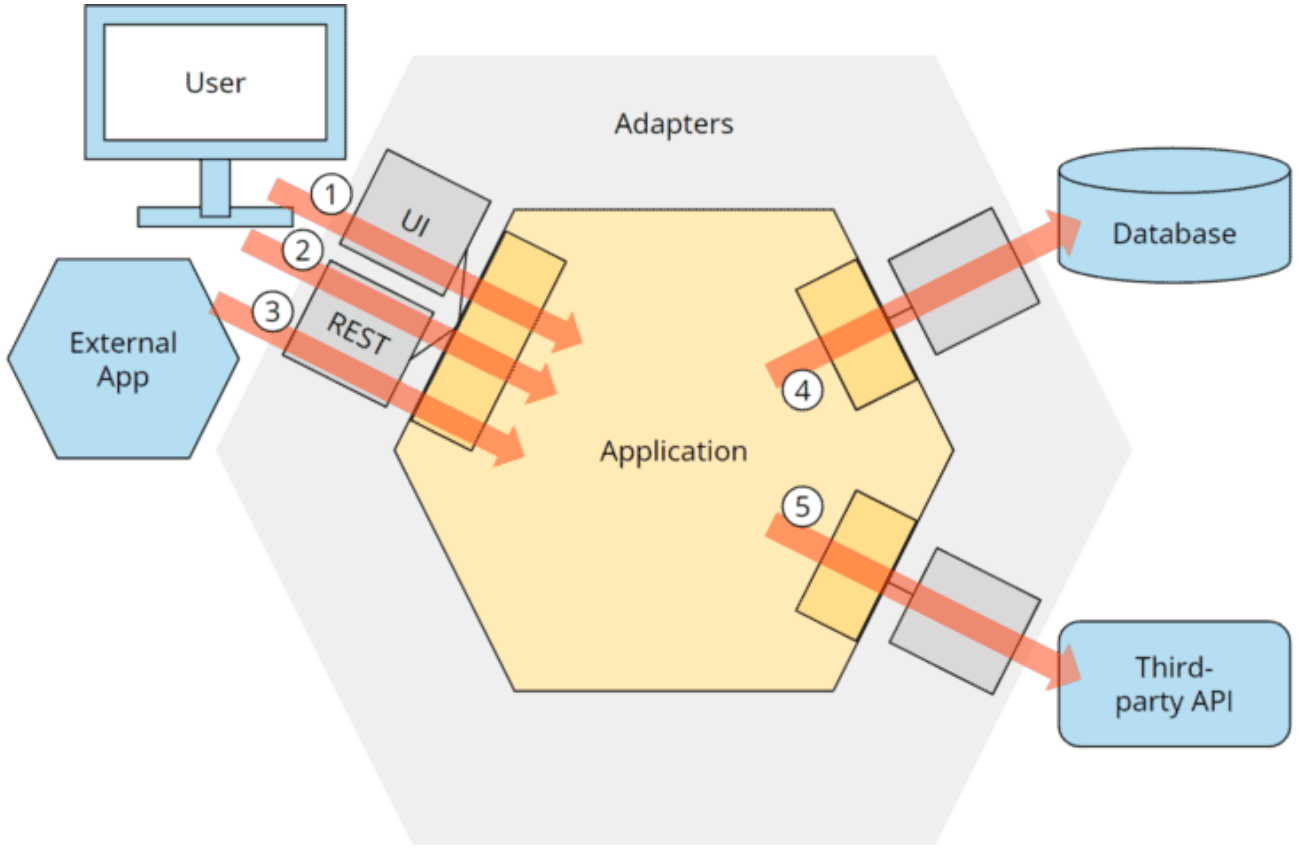
Hexagonal Architecture

- Called **Port-Adapter** Architecture
- **Port**: Interface
- **Adapter**: Implementation



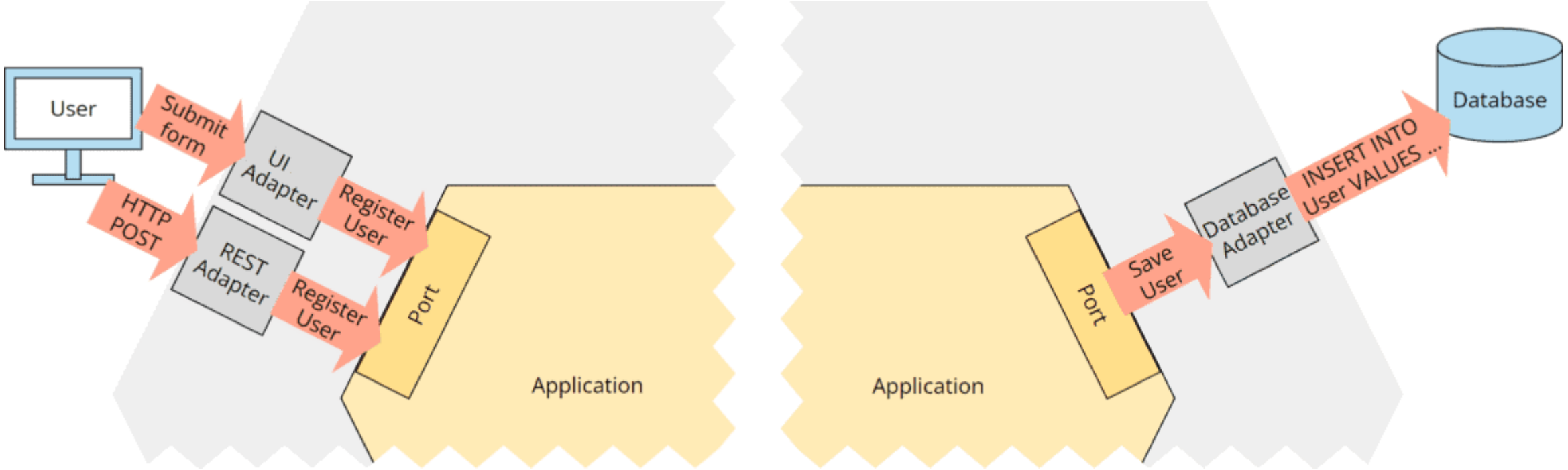
Hexagonal Architecture

- Called **Port-Adapter** Architecture
- **Port**: Interface
- **Adapter**: Implementation



Hexagonal Architecture

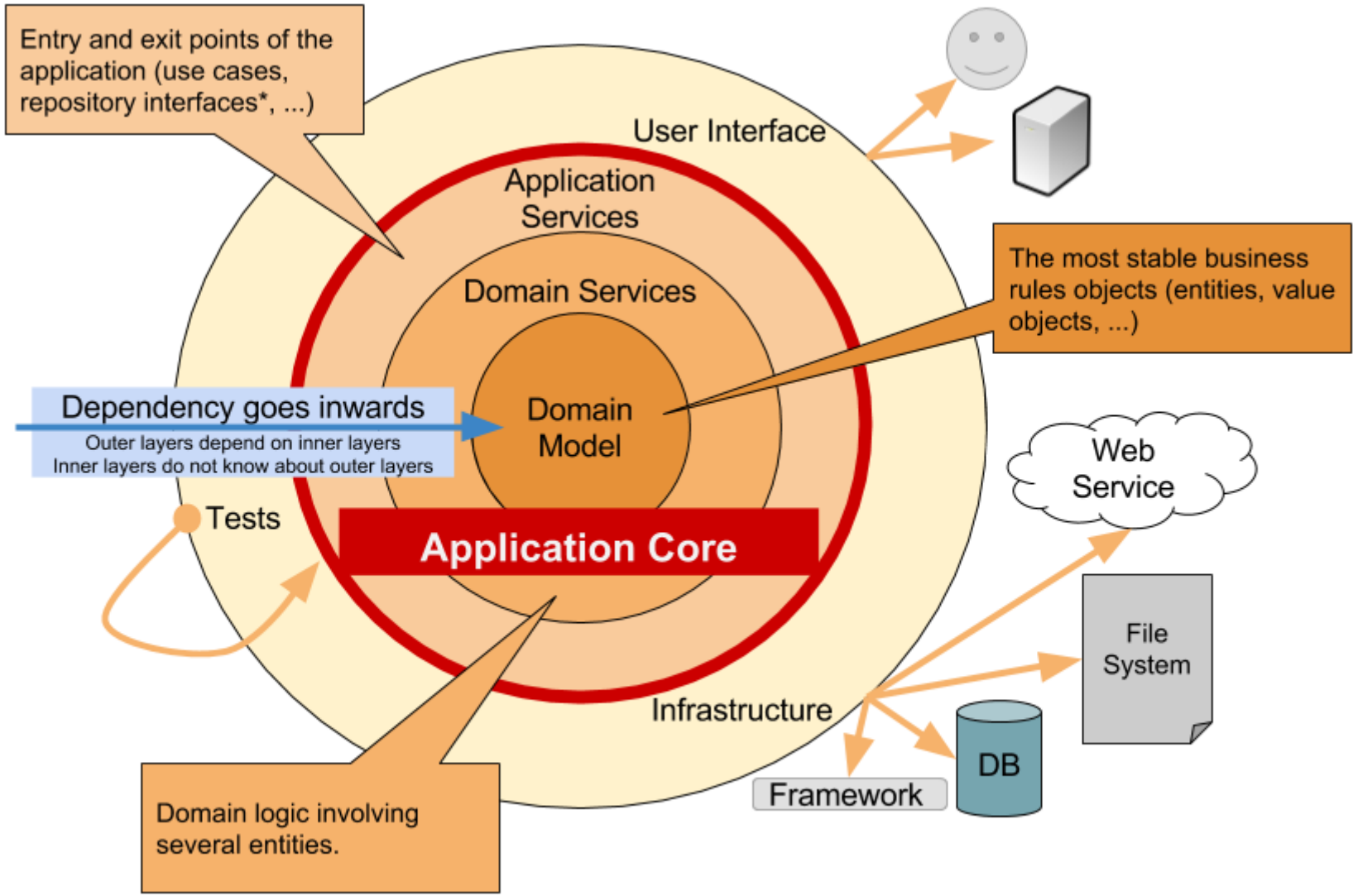
- Called **Port-Adapter** Architecture
- **Port**: Interface
- **Adapter**: Implementation



Benefits of Hexagonal Architecture

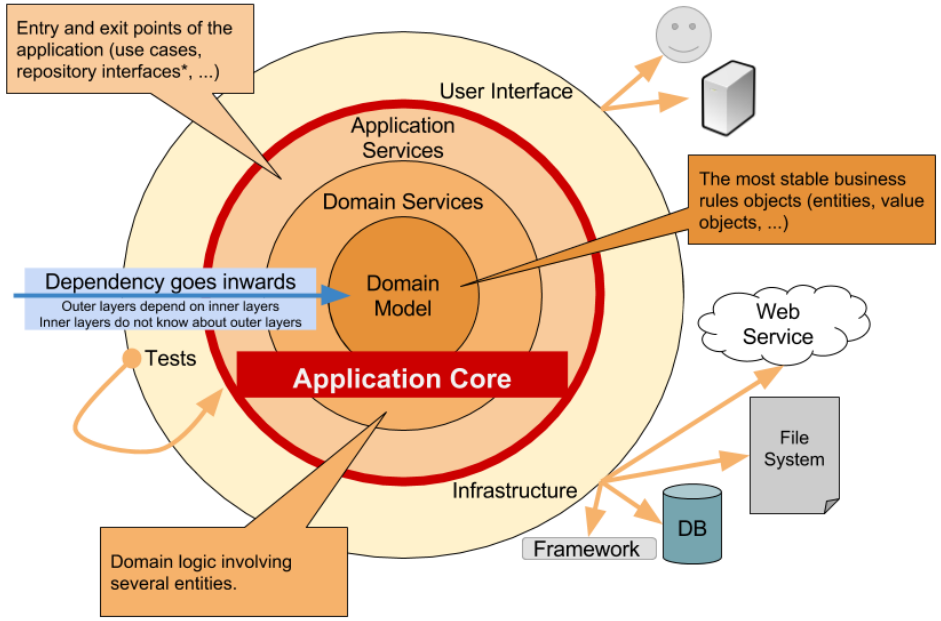
- Isolating the application and business logic from external factors so that they can all be tested easily and separately.
- Designing the user interfaces by their purpose rather than technology ensures that your application's technology stack can freely grow over time.
- Helps implement the Domain-Driven Design by making sure that the domain logic does not leak out of the core.
- The ports and adapters are just as replaceable as all the external entities, further contributing to the scalability of the entire application.
- The advanced separation of concerns also makes the app easier to maintain, as changing the code in one place or adding new dependencies/ways to interact with the app, do not require significant code changes.

Onion Architecture



Key Principles of Onion Architecture

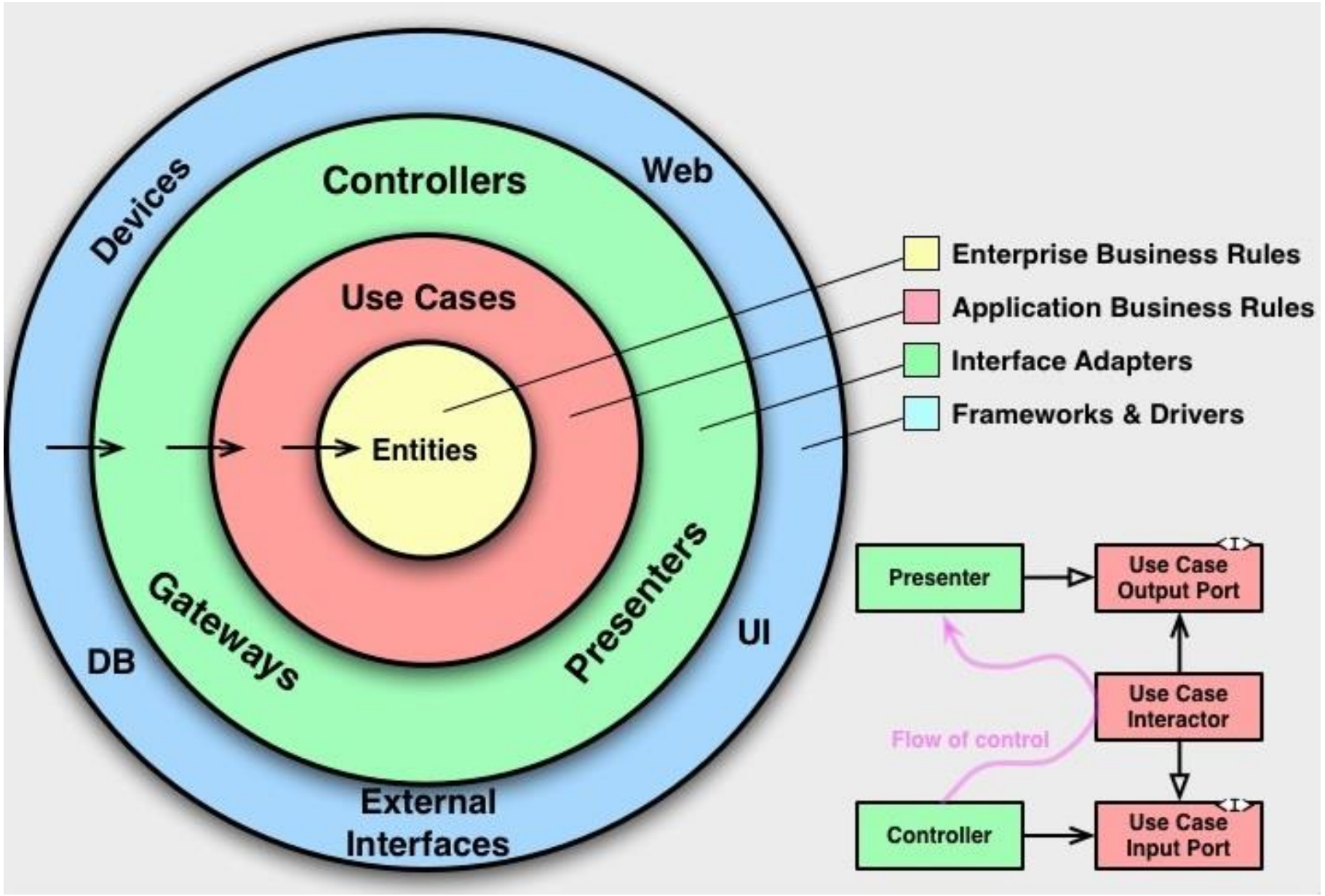
- **The application is built around an independent object model Inner layers define interfaces.**
- **Outer layers implement interfaces**
- **Direction of coupling is toward the center**
- **All application core code can be compiled and run separate from infrastructure**



Benefits of Onion Architecture

- **Testability**: Clear separation of concerns makes individual components easier to test in isolation.
- **Flexibility**: Changes in one layer typically don't affect other layers, promoting easier maintenance and updates.
- **Scalability**: As the application grows, the structure accommodates new features without drastic changes.

Clean Architecture



Types of Domain Centric Architecture

All three models follow the same idea

- Domain at the center
- Application and external infrastructure

Chapter 6. Domain Centric Architecture

Advantages and Disadvantages

Advantages

Focus on what's essential

More maintainable

Presentation, persistence easily changeable

Code more testable

Allows domain-driven design

Advantages and Disadvantages

Disadvantages

Requires a change in thinking

Higher development cost