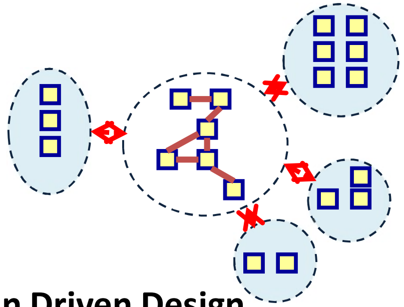
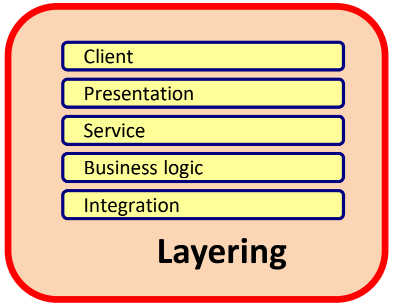
Architecture styles



Domain Driven Design



Client

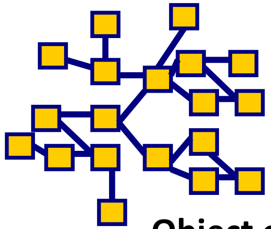
Presentation

Service

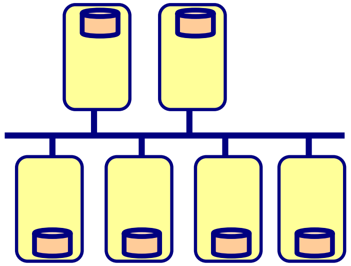
Business logic

Integration

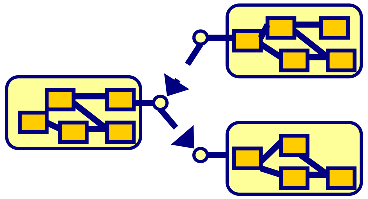
Layering



Object oriented



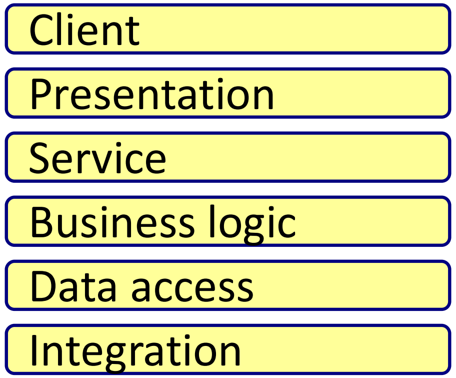
Microservices



Component based

Layering

- Separation of concern
- Layers are independent
- Layers can be distributed
- Layers use different techniques



Client

Presentation

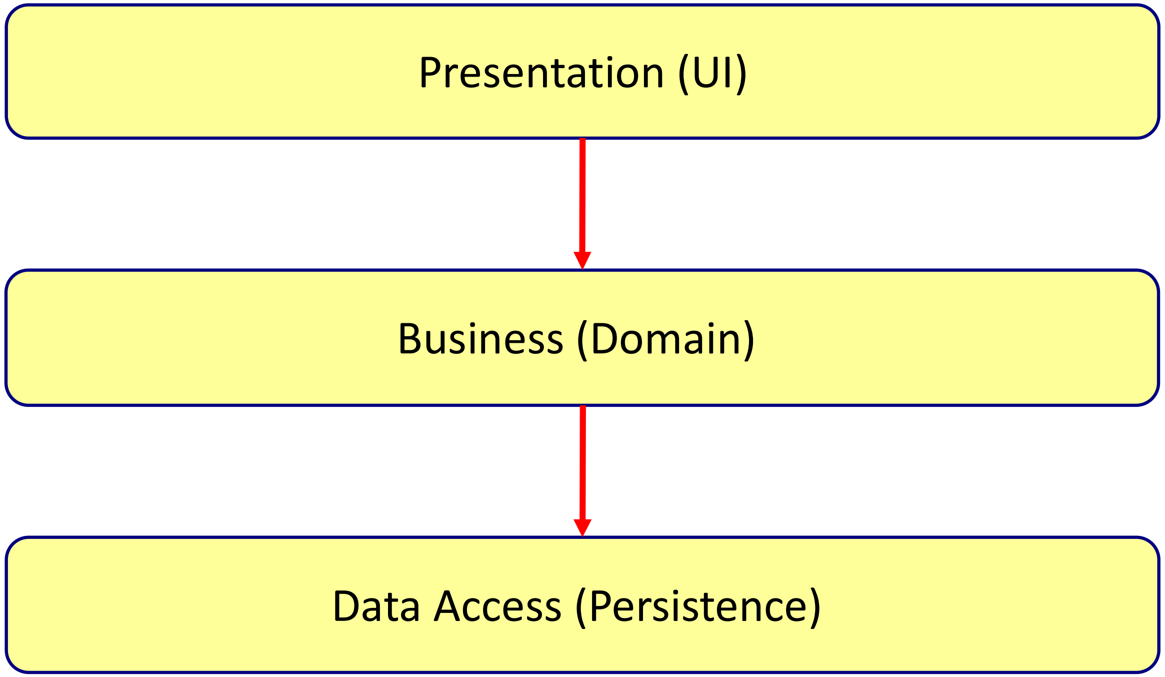
Service

Business logic

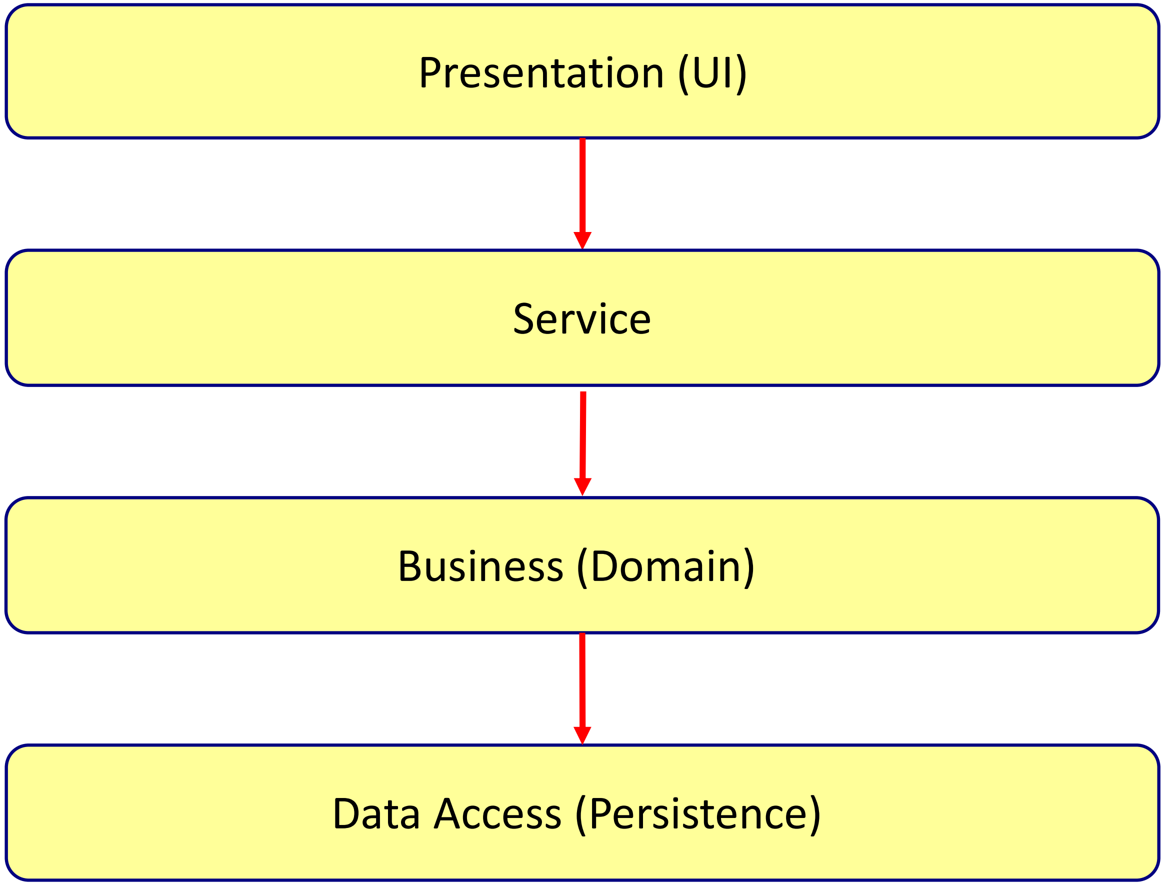
Data access

Integration

3 layered architecture



4 layered architecture



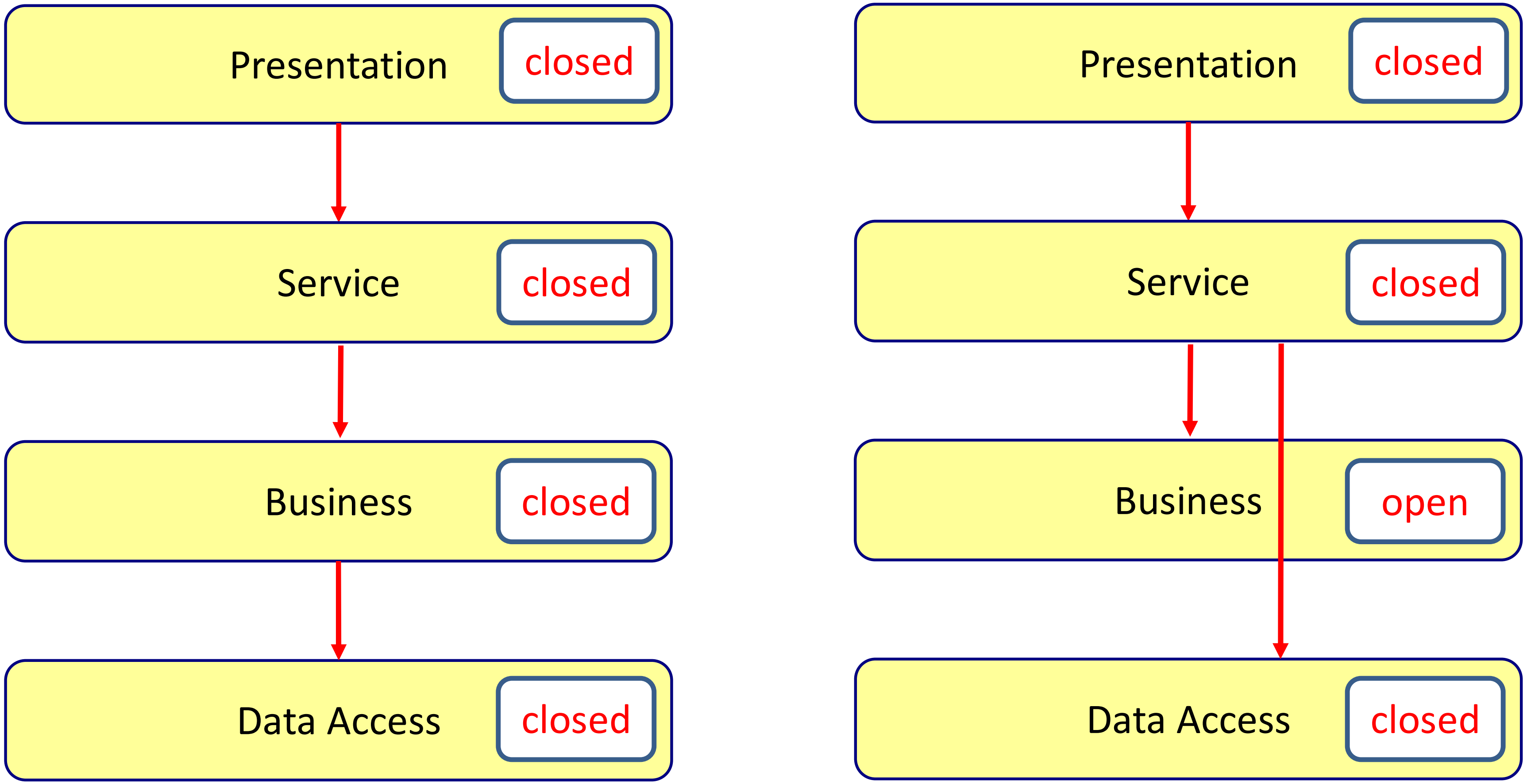
Presentation (UI)

Service

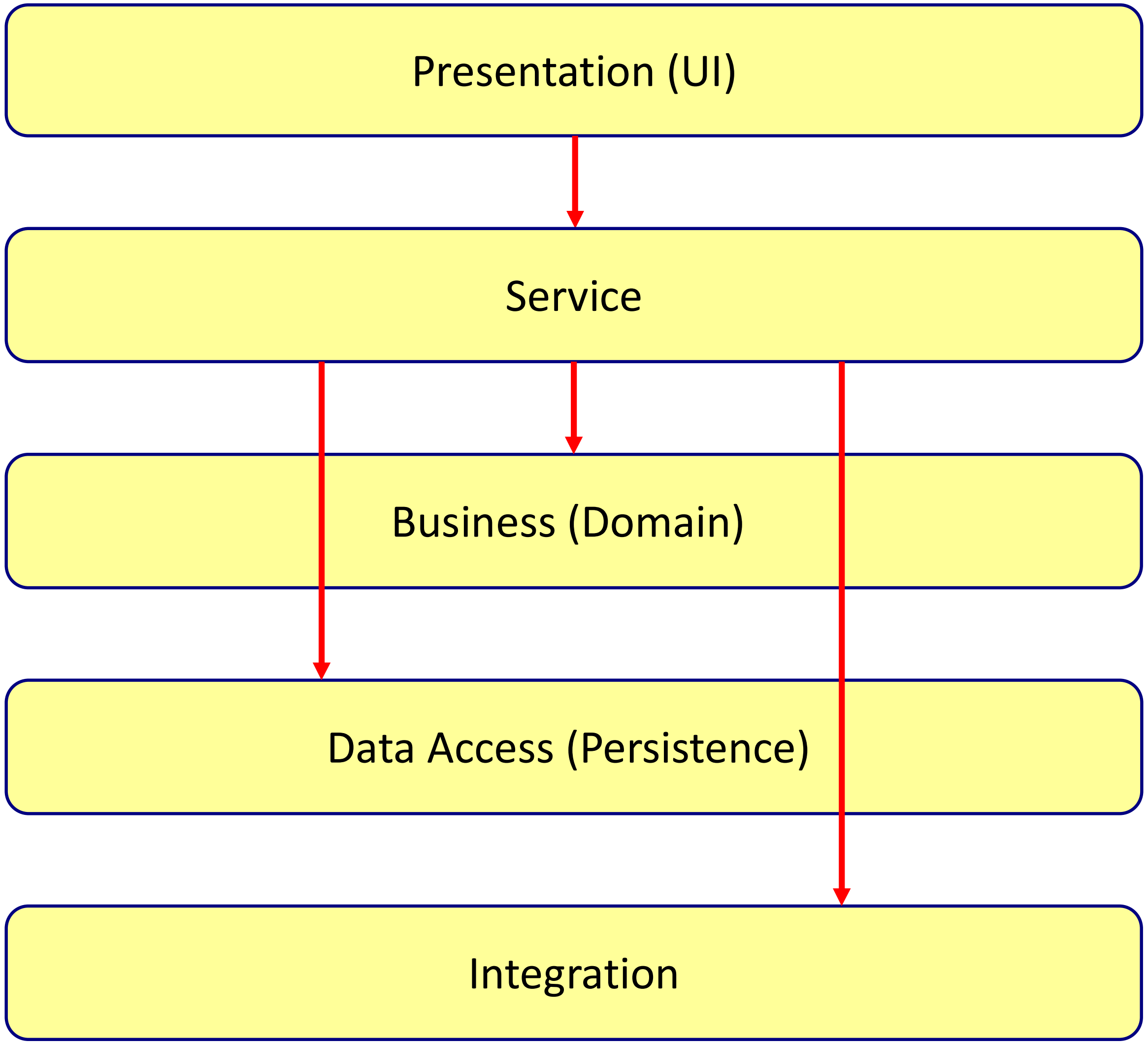
Business (Domain)

Data Access (Persistence)

Open and closed layers



5 layered architecture



Presentation (UI)

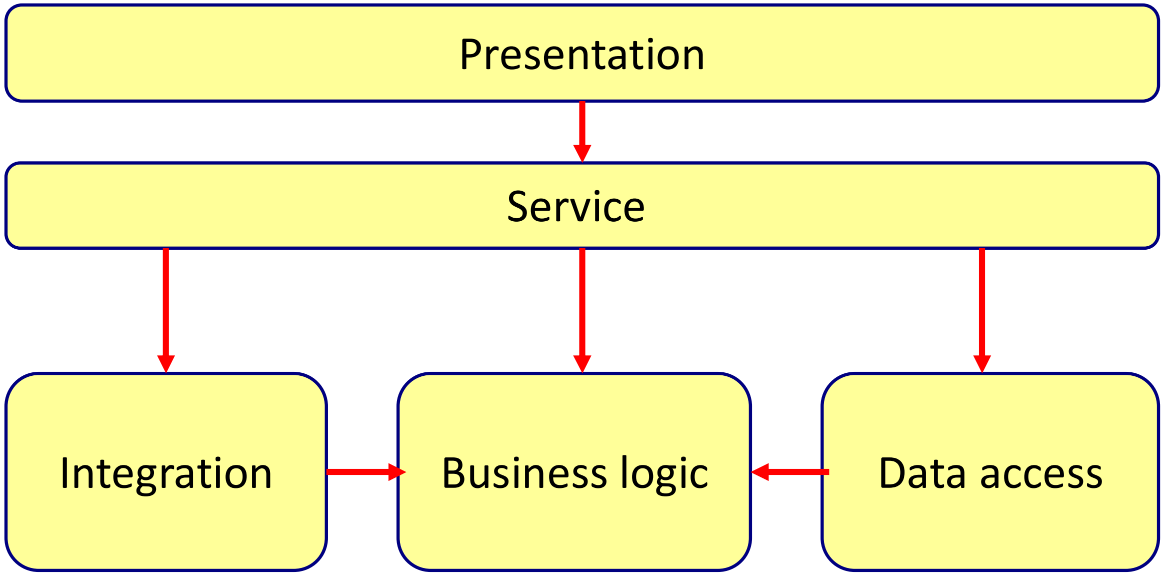
Service

Business (Domain)

Data Access (Persistence)

Integration

Layered architecture


Presentation

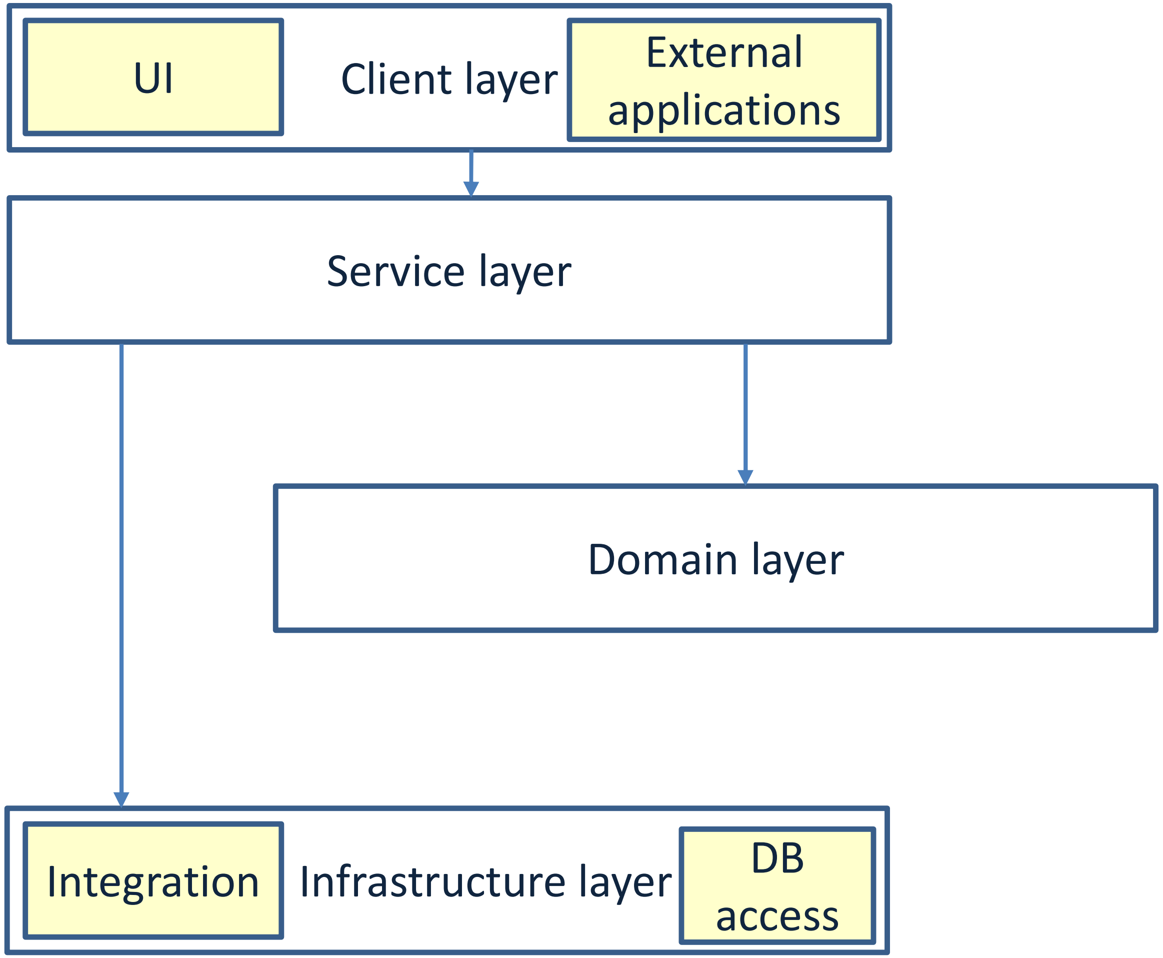
Service

Integration

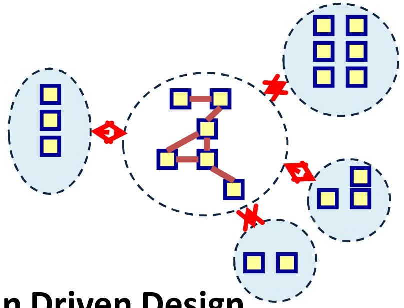
Business logic

Data access

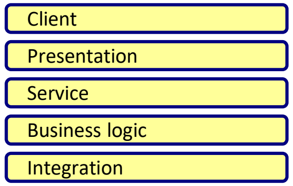
Layered architecture



Architecture styles



Domain Driven Design



Client

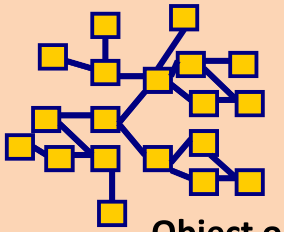
Presentation

Service

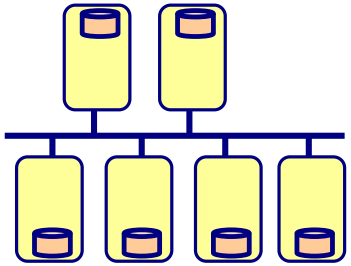
Business logic

Integration

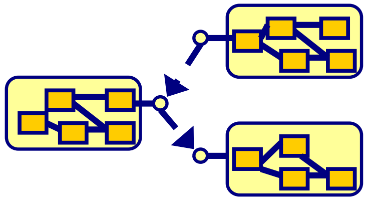
Layering



Object oriented



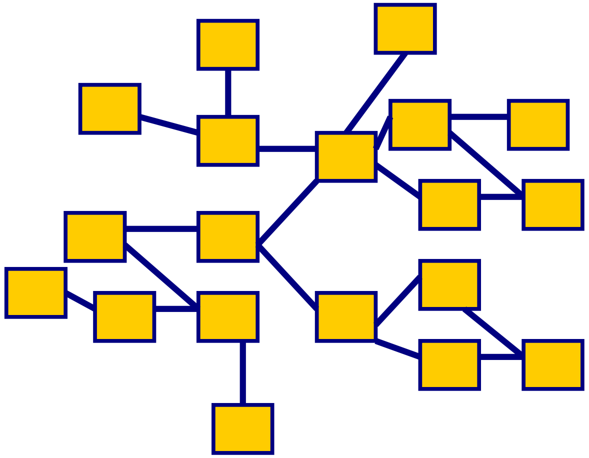
Microservices



Component based

Object Oriented

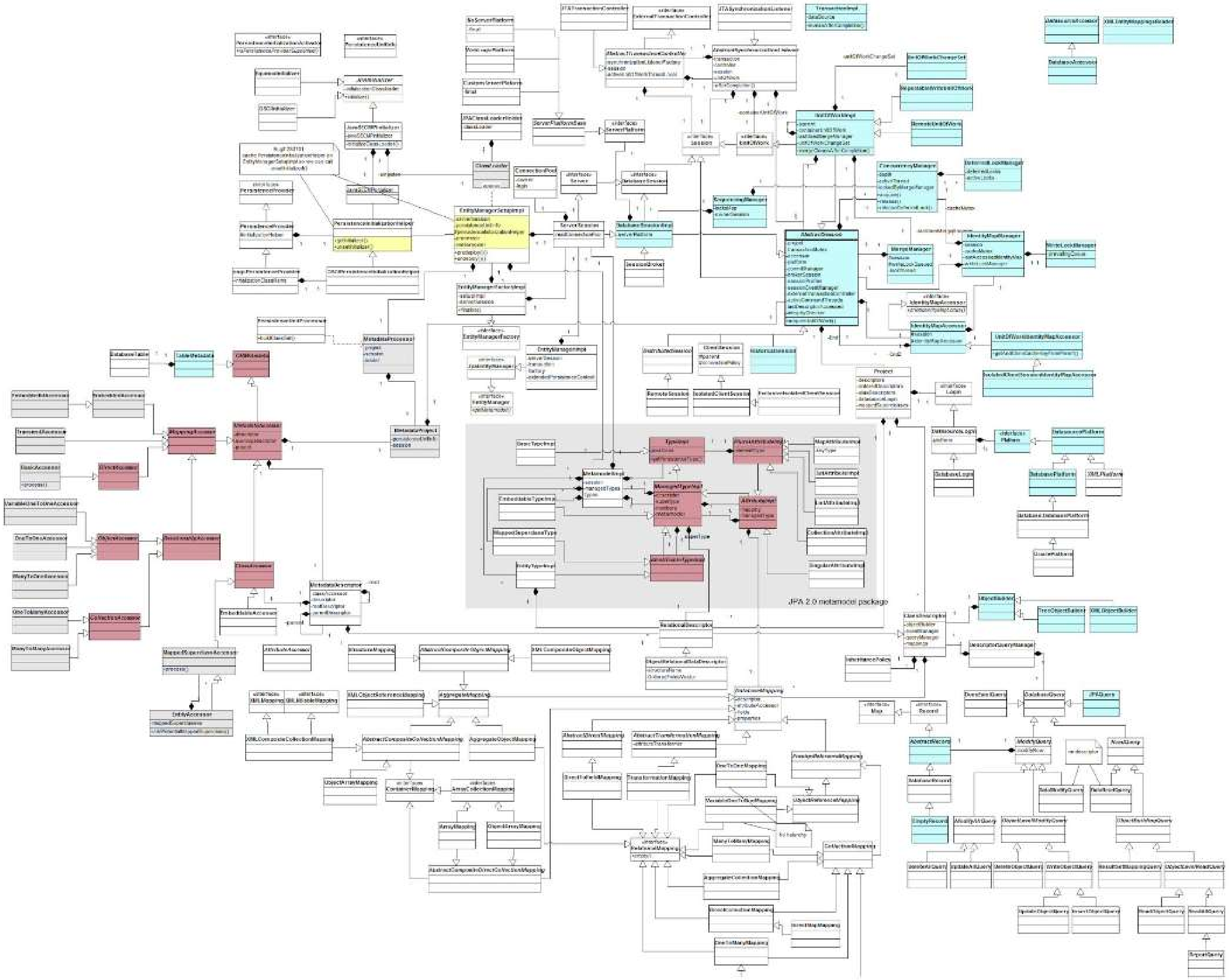
- Decompose the domain in real world objects
- Data and functionality together



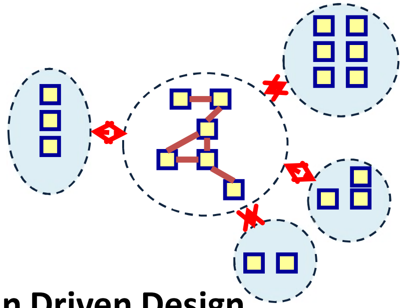
Advantages of object orientation

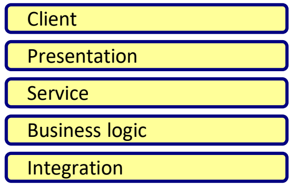
- High cohesion, low coupling
- Encapsulation
  - Data hiding
- Flexibility
  - Polymorphism
- Reuse ?
  - Inheritance

Big ball of mud

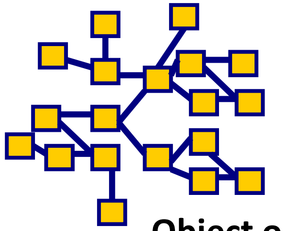


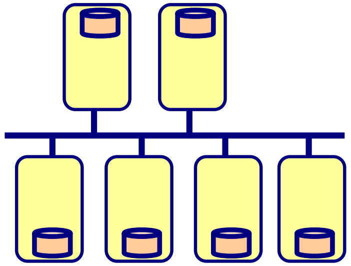
Architecture styles

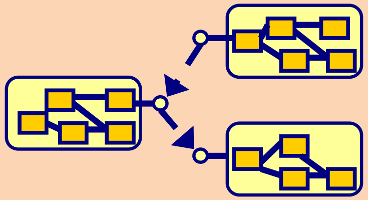

Domain Driven Design



Layering

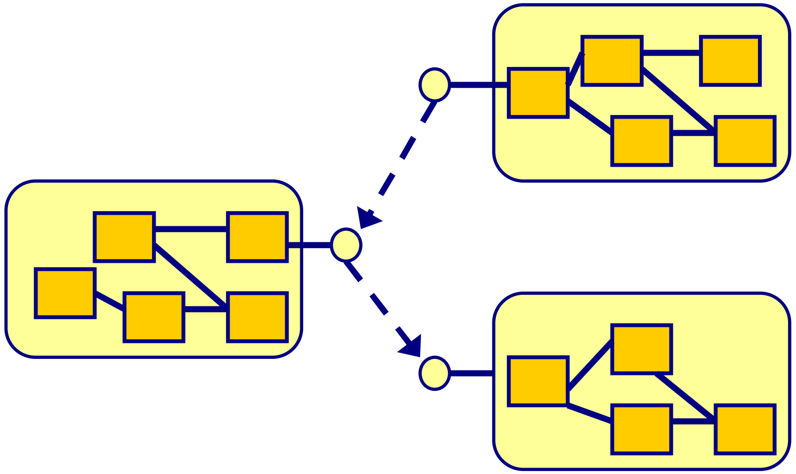

Object oriented


Microservices


Component based

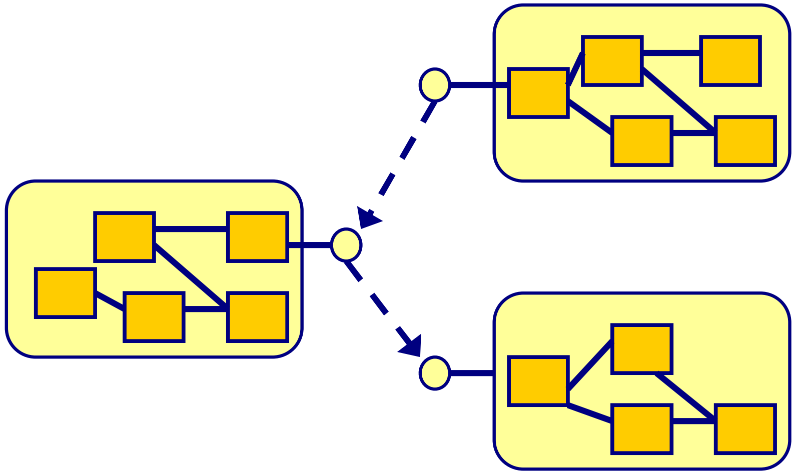
Component Based Development

- Decompose the domain in functional components



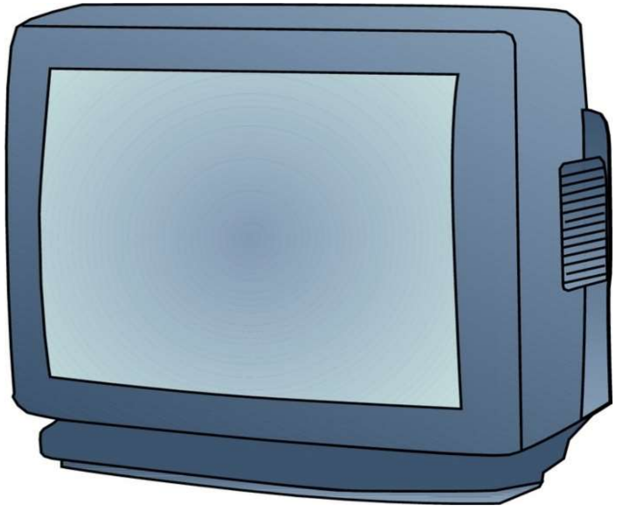
What is a component?

- There is no definition
- What we agree upon:
  1. A component has an interface
  2. A component is encapsulated
- Plug-and-play
- A component can be a single unit of deployment



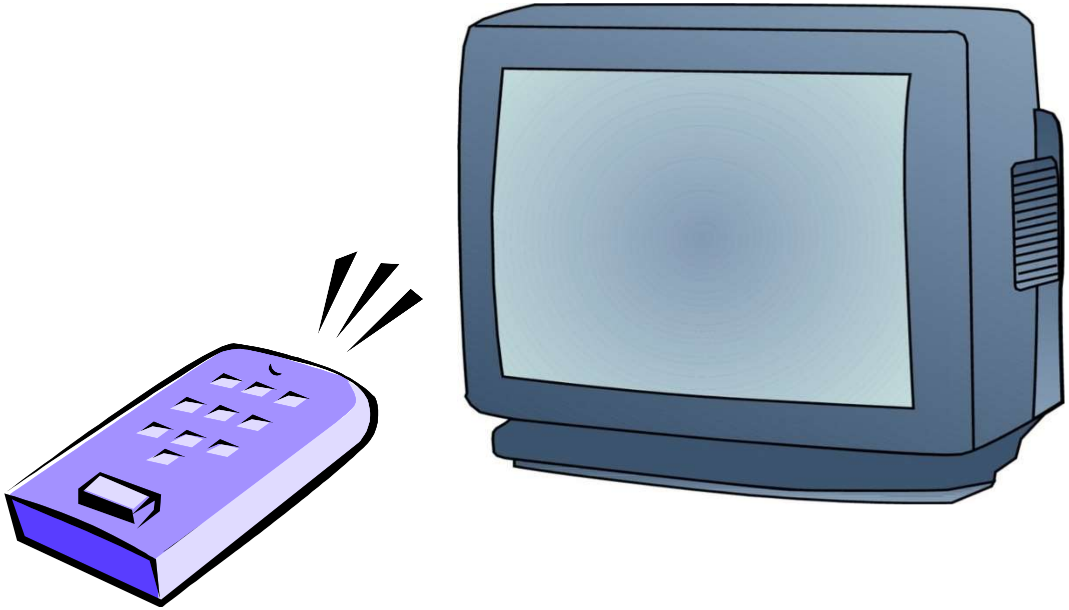
Encapsulation

- The implementation details are hidden



Interface

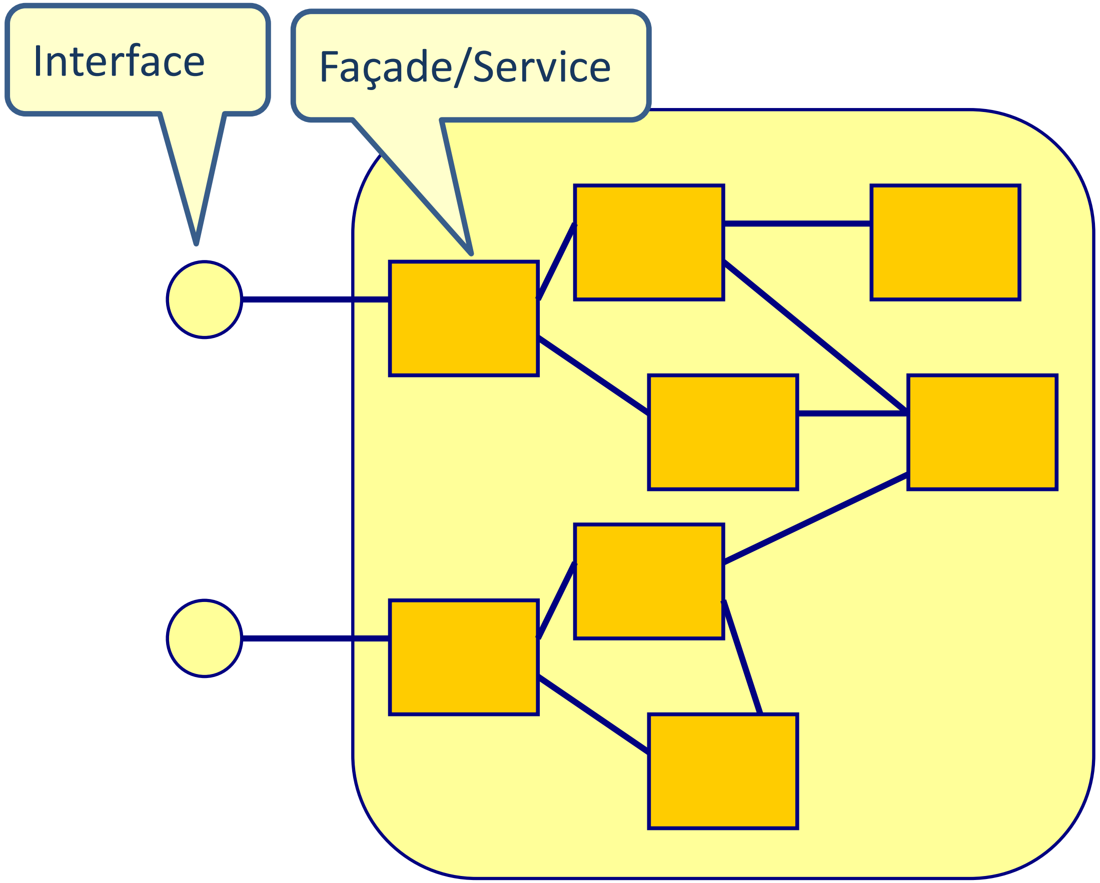
- The interface tells what you can do (but not how)



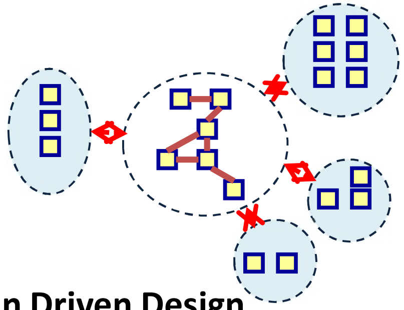
Advantages of components

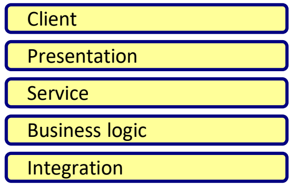
- High cohesion, low coupling
- Flexibility
- Reuse ?

Internals of a component



Architecture styles


Domain Driven Design


Client

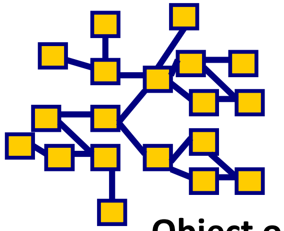
Presentation

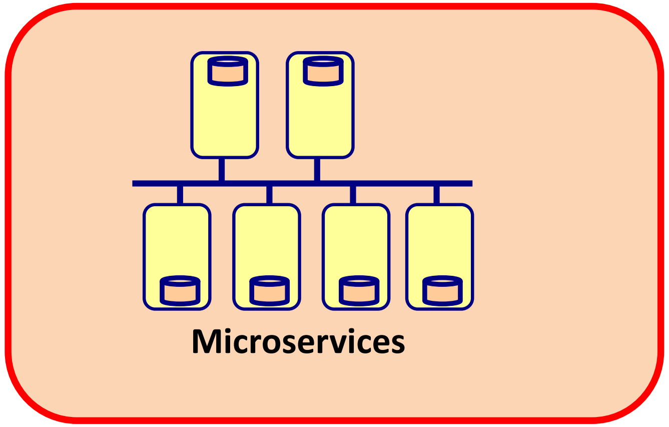
Service

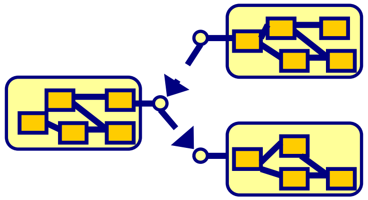
Business logic

Integration

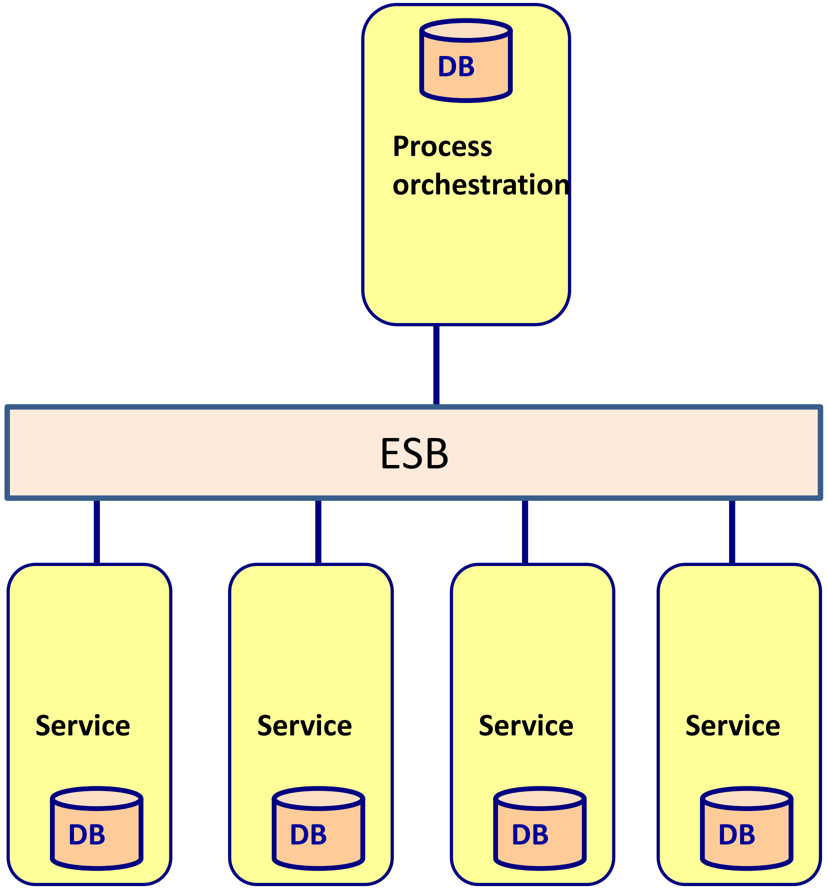
Layering


Object oriented


Microservices

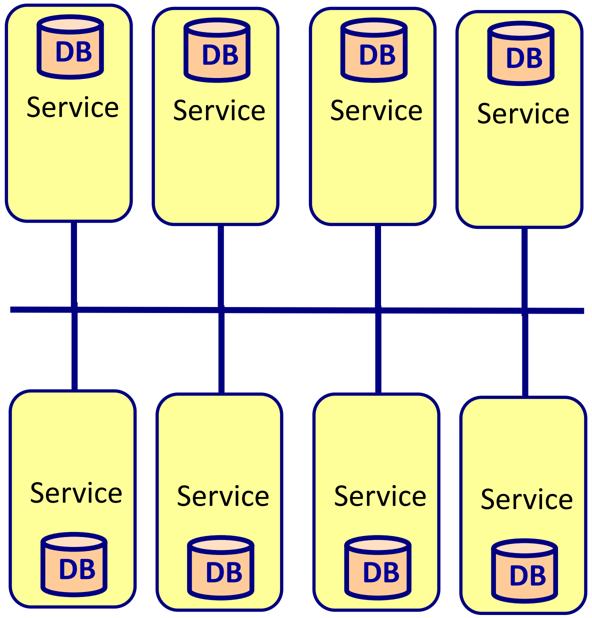

Component based

Service Oriented Architecture



Microservices

- Small independent services
  - Simple and lightweight
  - Runs in an independent process
  - Technology agnostic
  - Decoupled



Orchestration vs. choreography

- Orchestration
  - One central brain

- Choreography
  - No central brain