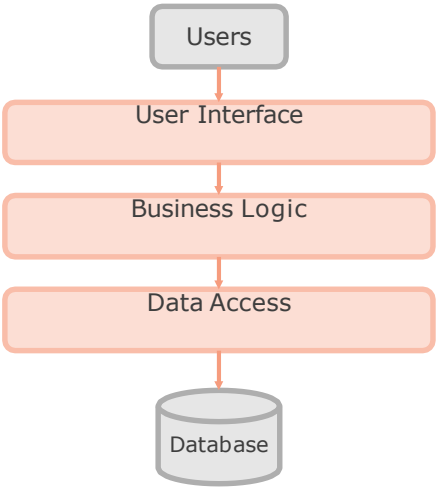


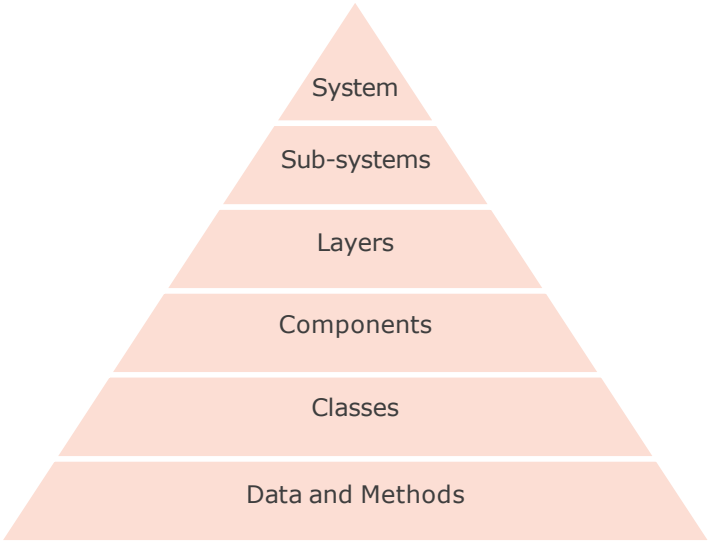

Clean Architecture

What is Software Architecture?

- High-level
- Structure
- Layers
- Components
- Relationships



Levels of Architectural Abstraction



What Is Bad Architecture?

Complex

Incoherent

Brittle

Untestable

Unmaintainable

What Is Good Architecture?

Simple

Understandable

Flexible

Emergent

Testable

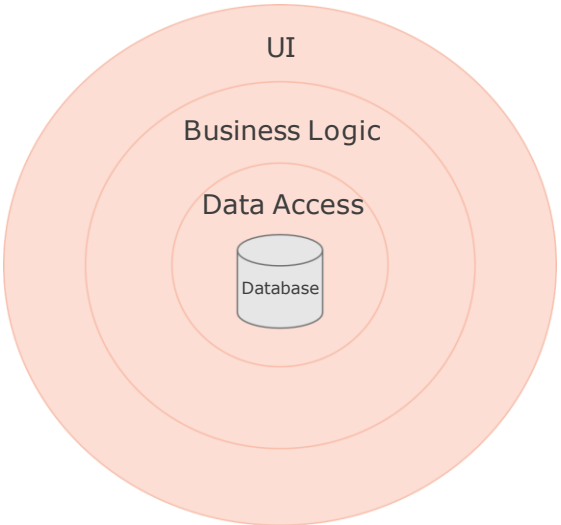
Maintainable

What Is Clean Architecture?

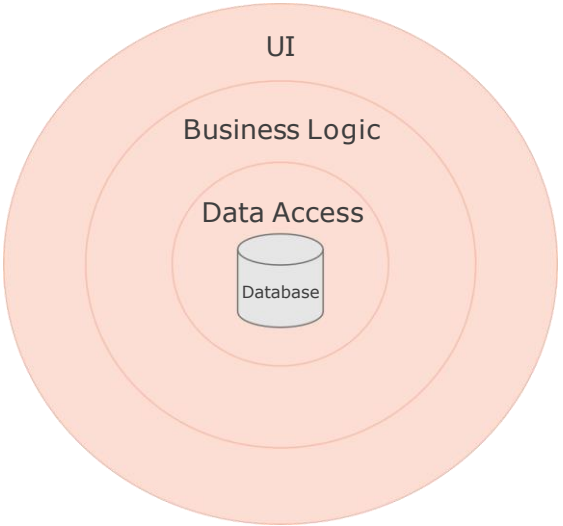
Architecture that is designed for
the inhabitants of the architecture…
not for the architect… or the machine

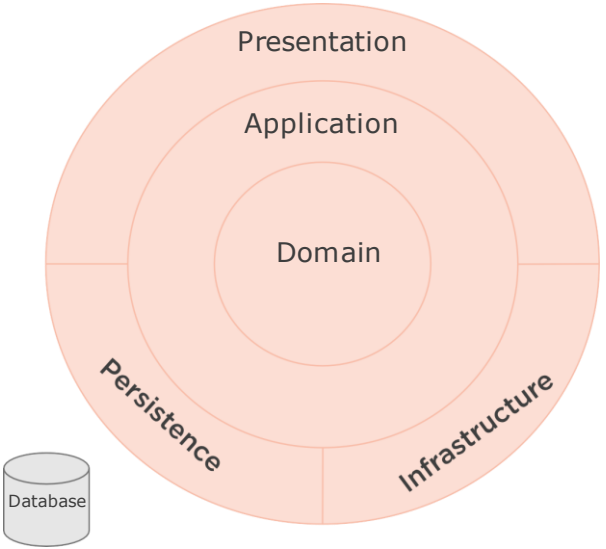

Domain-centric Architecture

Classic Three-layer Database-centric Architecture



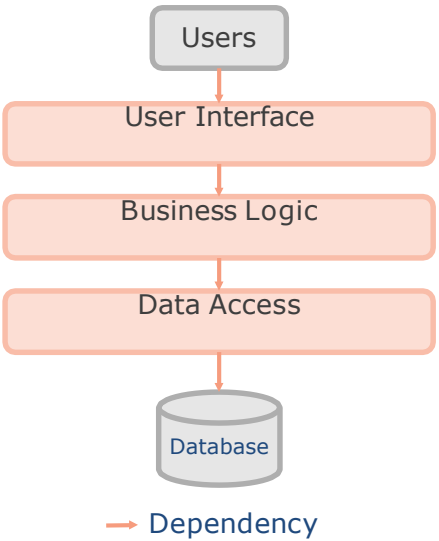
Database-centric vs. Domain-centric Architecture



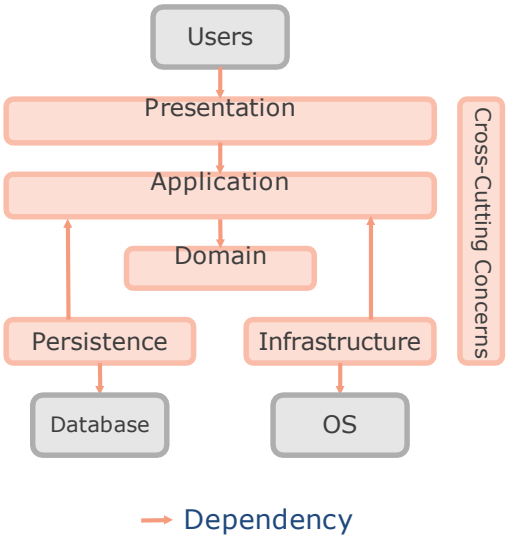


Application Layer

Classic Three-layer Architecture

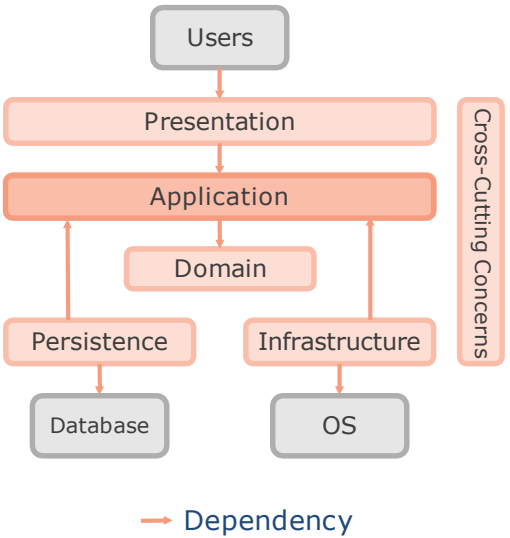


Modern Four-layer Architecture



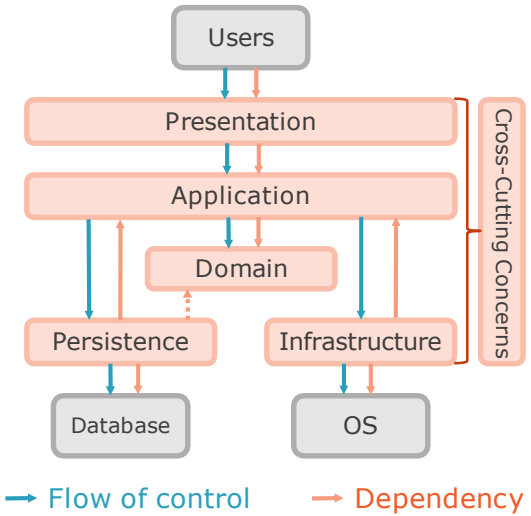
Application Layer

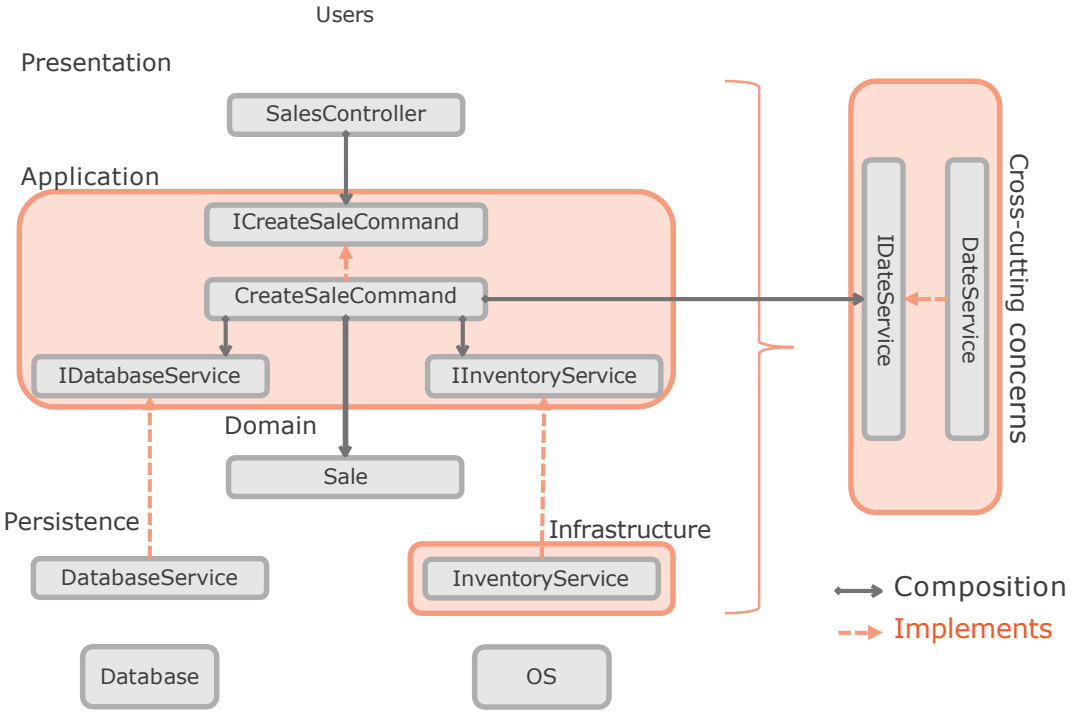
- Implements use cases
- High-level application logic
- Knows about the domain
- No knowledge of other layers
- Contains interfaces



Layer Dependencies

- Dependency inversion
- Inversion of control
- Independent deployability
- Flexible and maintainable




Users
Presentation
SalesController
Application
ICreateSaleCommand
CreateSaleCommand
IDatabaseService
IInventoryService
Domain
Sale
Persistence
DatabaseService
Infrastructure
InventoryService
Database
OS
IDateService
DateService
Cross-cutting concerns
Composition
Implements


Commands and Queries

Command-query Separation

Command

Does something

Should modify state

Should not return a value

Query

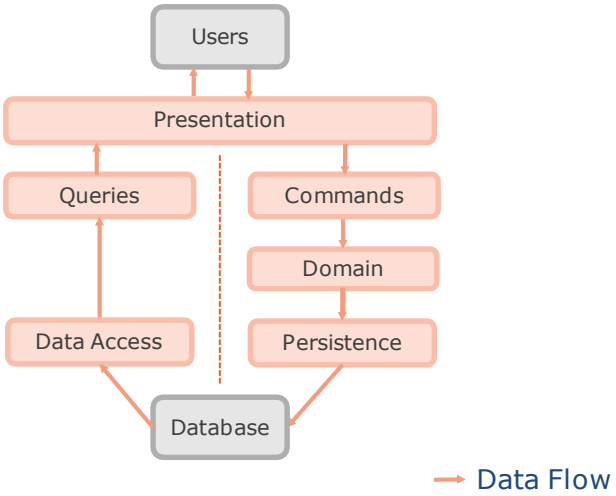
Answers a question

Should not modify state

Should return a value

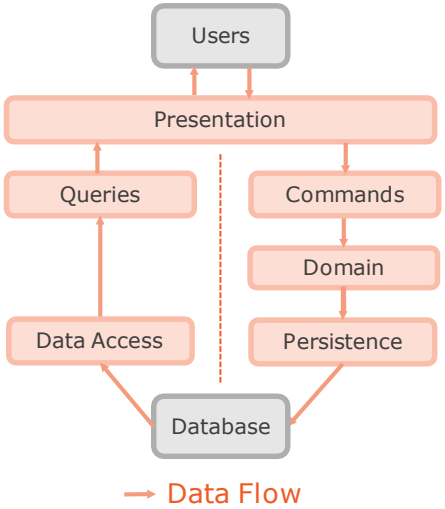
CQRS – Commands and Queries Responsibility Segregation

CQRS Architectures



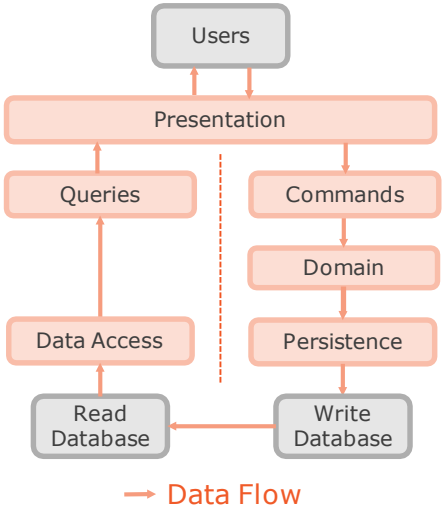
Single-database CQRS

- Single database
- Commands use domain
- Queries use database
- Simplest of the three



Two-database CQRS

- Read and write databases
- Commands use write DB
- Queries use read DB
- Eventual consistency
- Orders of magnitude faster



Event Sourcing CQRS

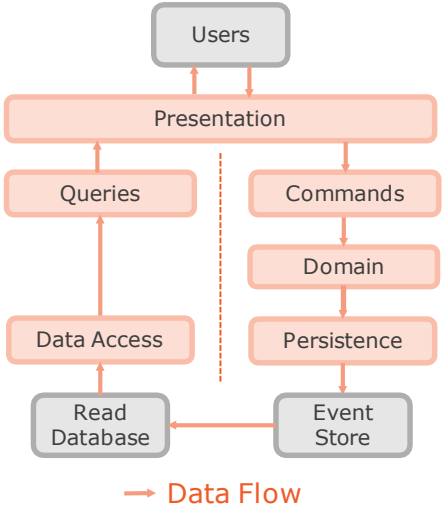
Store events

Replay events

Modify entity

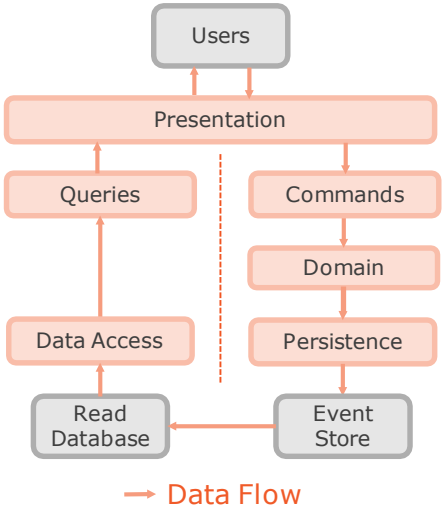
Store new event

Update read database

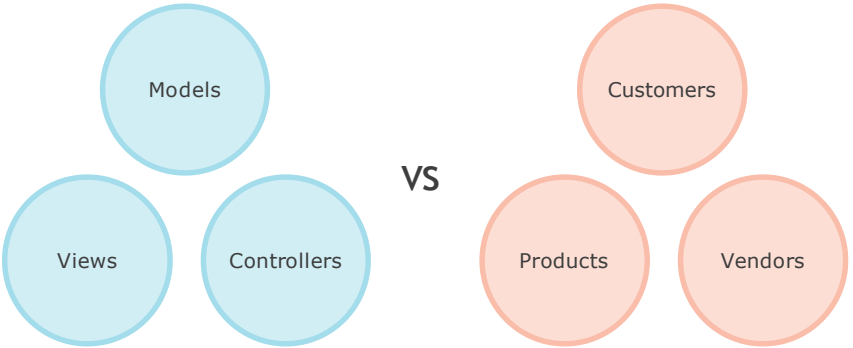


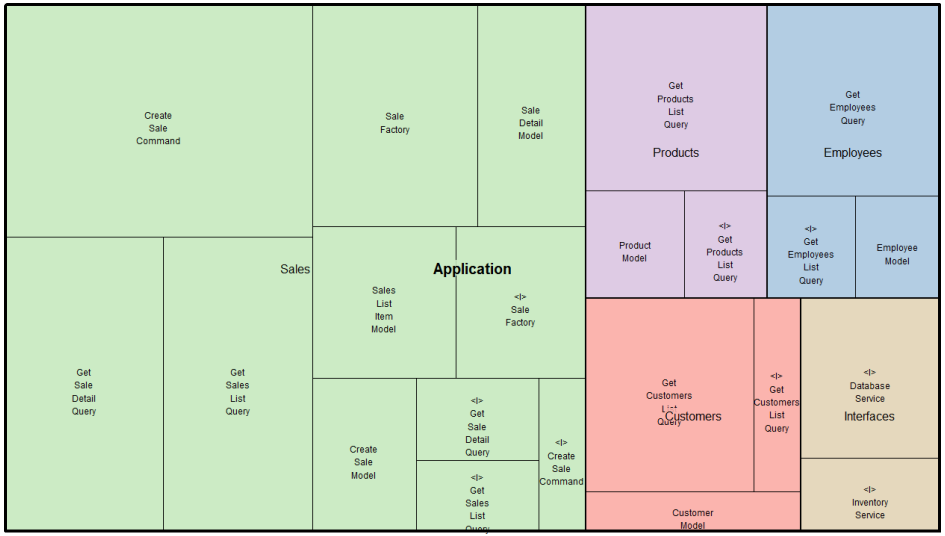
Event Sourcing CQRS

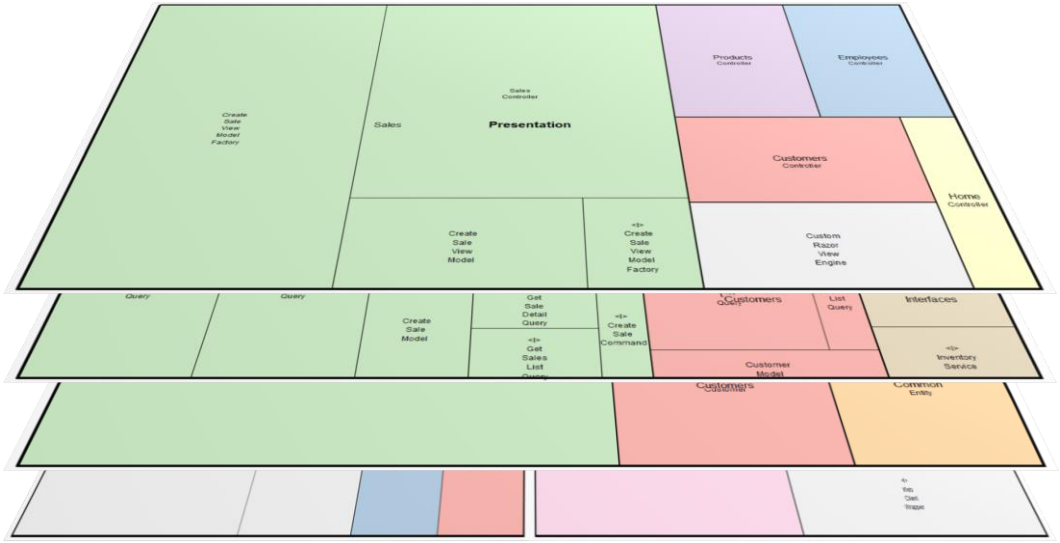
- Complete audit trail
- Point-in-time reconstruction
- Replay events
- Multiple read database
- Rebuild production database



Functional Organization

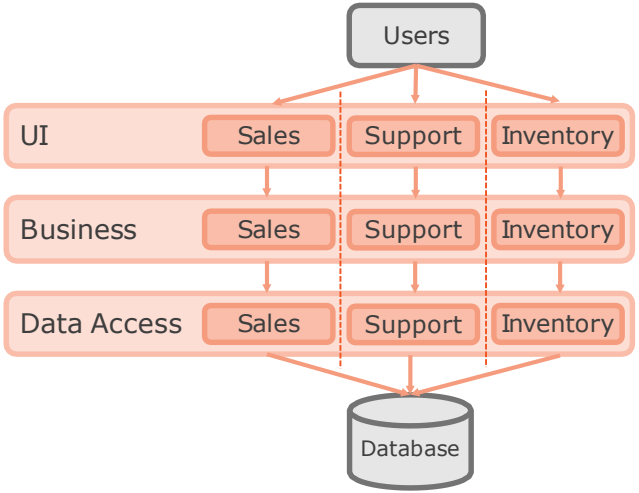

Models
Views
Controllers
VS
Customers
Products
Vendors


Create Sale Command
Sale Factory
Sale Detail Model
Get Products List Query
Products
Get Employees Query
Employees
Sales
Application
Product Model
<I> Get Products List Query
<I> Get Employees List Query
Employee Model
Get Sale Detail Query
Get Sales List Query
Sales List Item Model
<I> Sale Factory
Create Sale Model
<I> Get Sale Detail Query
<I> Get Sales List Query
<I> Create Sale Command
Get Customers List Query
Customers
<I> Get Customers List Query
<I> Database Service
Interfaces
Customer Model
<I> Inventory Service


Create Sale View Model Factory
Sales
Sales Controller
Presentation
Products Controller
Employees Controller
Customers Controller
Home Controller
Create Sale View Model
<i> Create Sale View Model Factory
Custom Razor View Engine
Query
Query
Create Sale Model
Get Sale Detail Query
<i> Get Sales List Query
<i> Create Sale Command
Customers Query
List Query
Interfaces
<i> Inventory Service
Customer Model
Customers Controller
Common Entity


Microservices

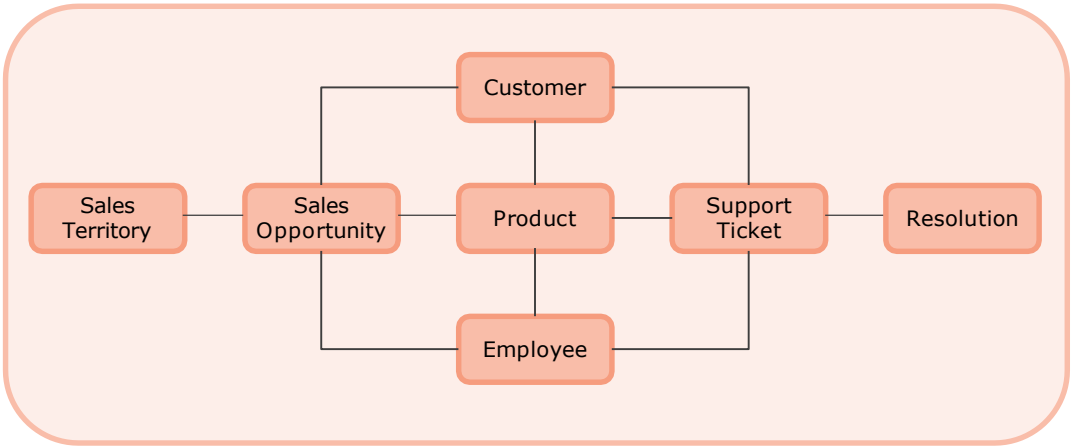
Components


Users
UI
Sales
Support
Inventory
Business
Sales
Support
Inventory
Data Access
Sales
Support
Inventory
Database

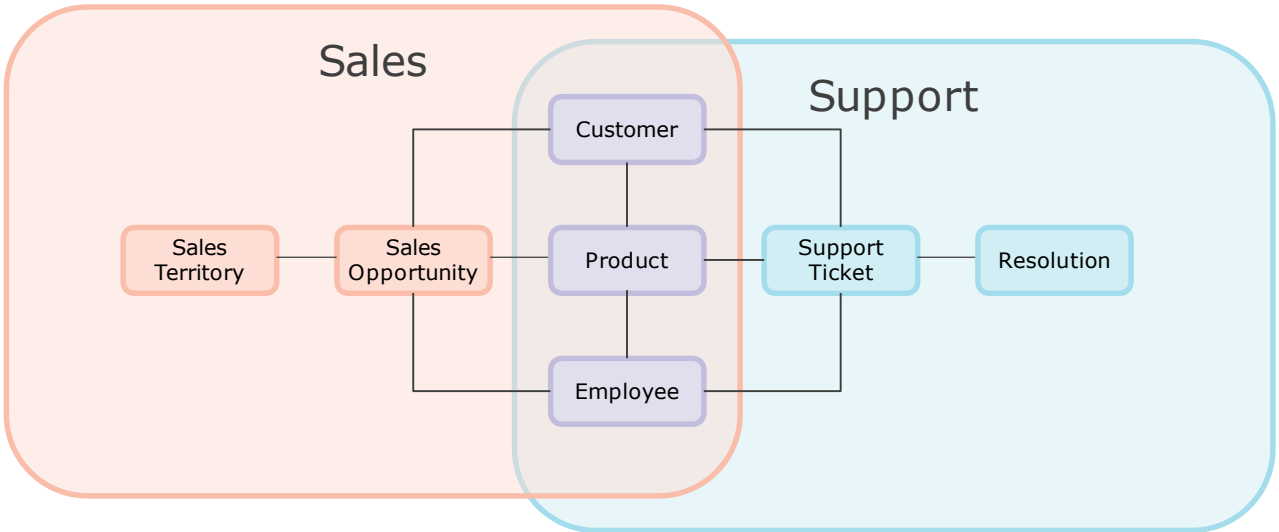
Problem Domain

Sales	Support
Sales opportunity	Support ticket
Contact	Customer
Sales person	Support person
Product	Product
Sales territory	Resolution

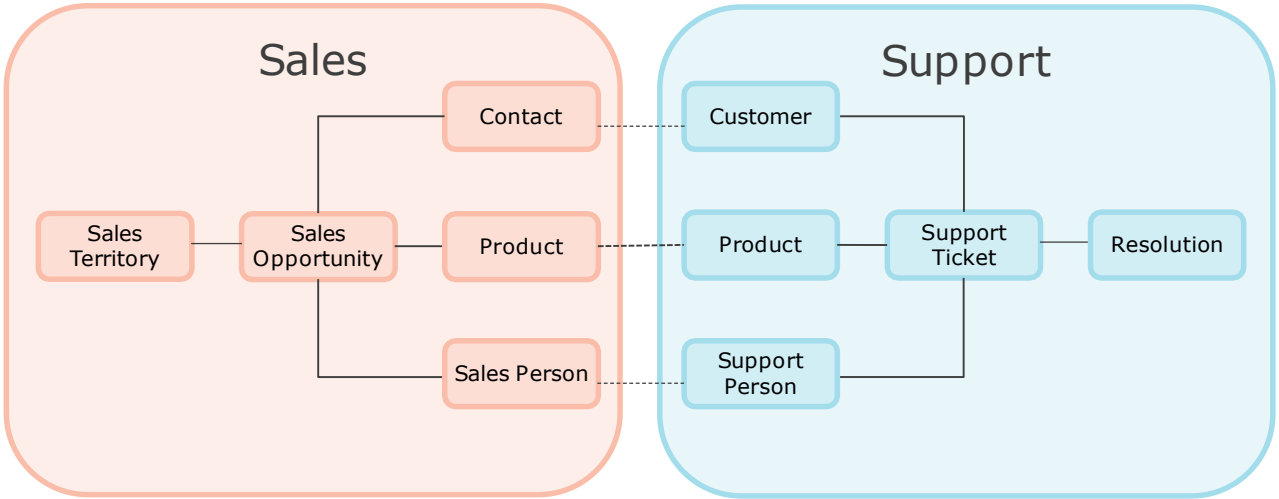
Single Domain Model



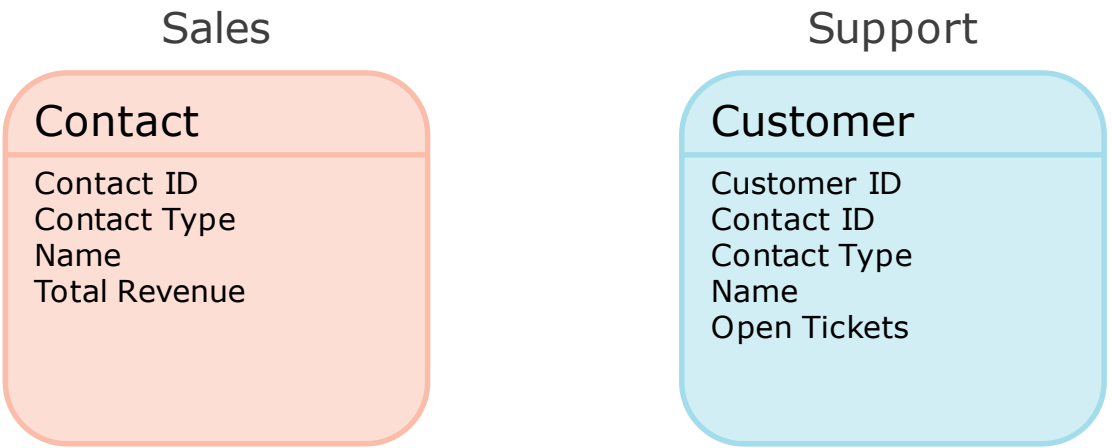
Overlapping Contexts



Bounded Contexts

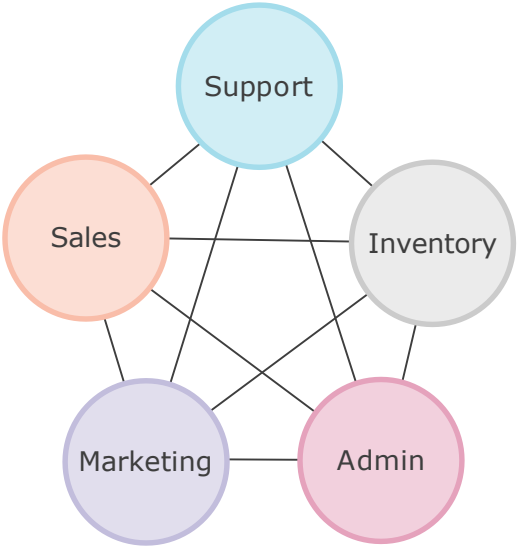


Cross-boundary Entities



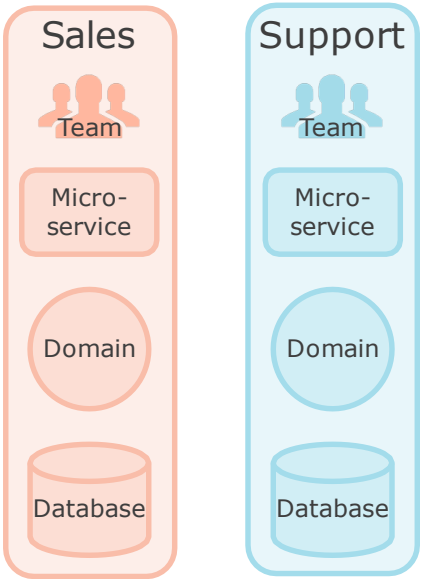
Microservices

- Subdivide monoliths
- Clearly-defined interfaces
- Small teams
- Independent



Microservices

- Bounded context
- Cohesion/coupling
- Single domain of knowledge
- Consistent data model
- Independence



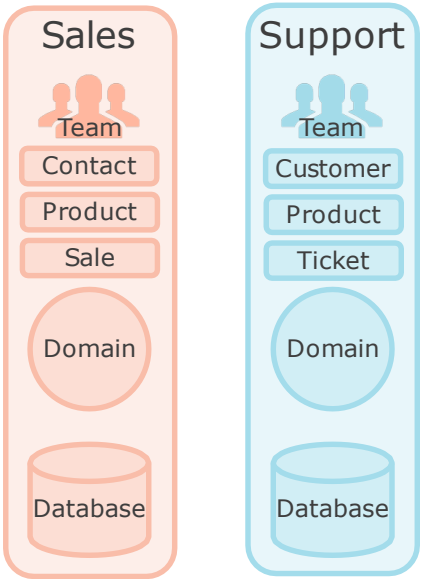
Microservices

How small?

Where to divide?

Microservice per aggregate root

Database per bounded context



Literature, Demo

Robert C. Martin: Clean Architecture. Craftsman's Guide to Software Structure and Design

`https://github.com/matthewrenze/clean-architecture-demo`

Thank You!