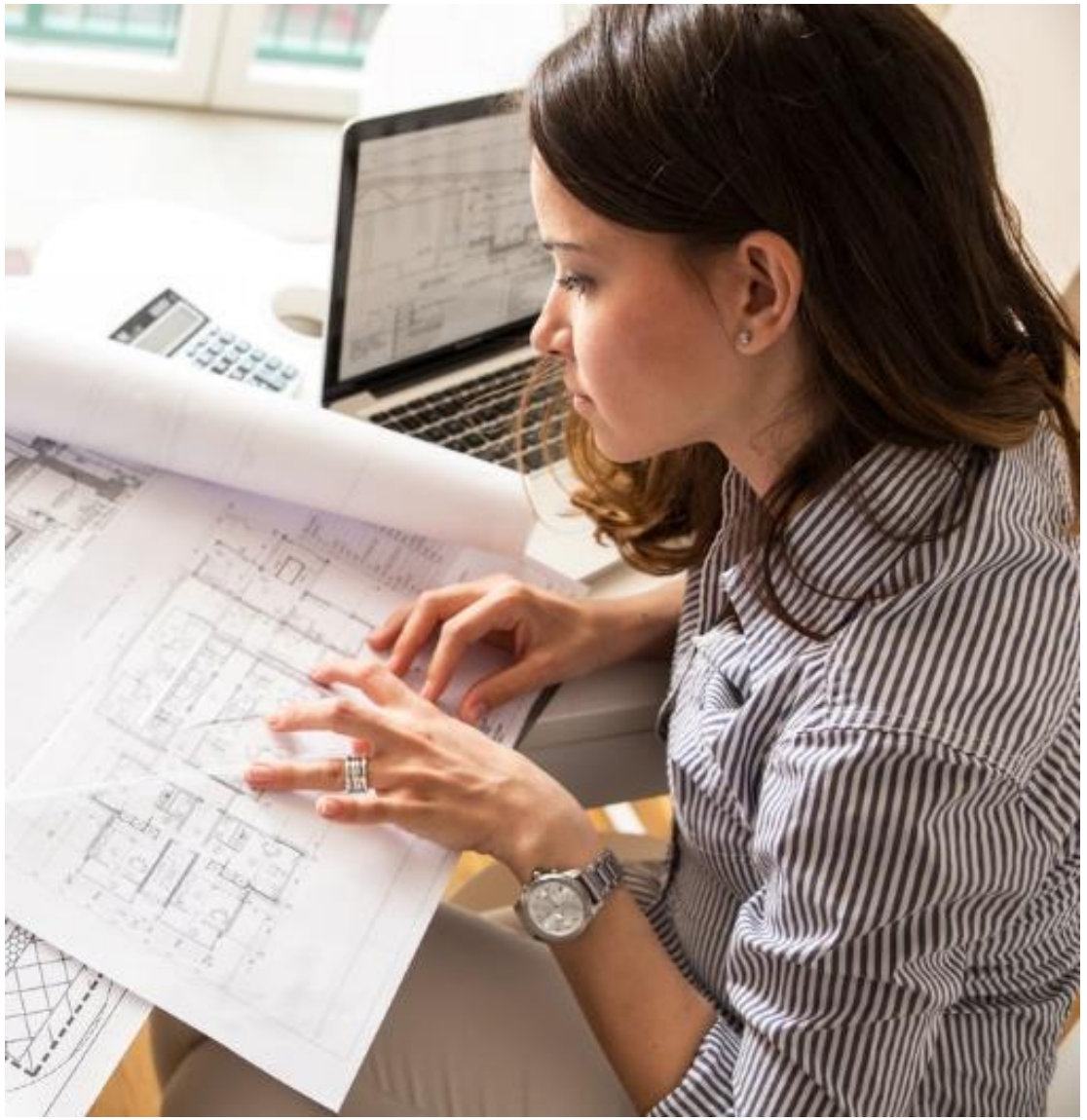
Clean Architecture

Patterns, Practices, and Principles

@matthewrenze

#NebraskaCode



About Me

- Independent software consultant
- Education
  - B.S. in Computer Science (ISU)
  - B.A. in Philosophy (ISU)
- Community
  - Pluralsight Author
  - ASPInsider
  - Public Speaker
  - Open-Source Software

IOWA STATE UNIVERSITY

Overview

- Clean Architecture
- Domain-Centric Architecture
- Application Layer
- Commands and Queries
- Functional Organization
- Microservices

Focus

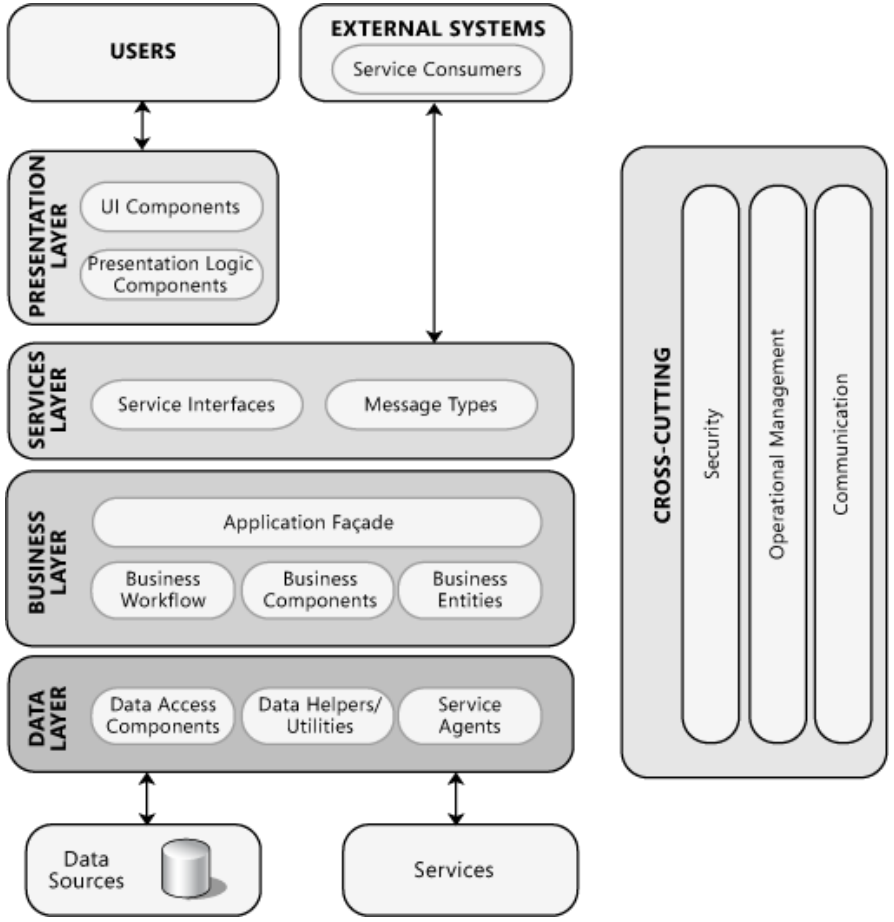
- Enterprise Architecture
- Line-of-Business Applications
- Modern equivalent of 3-Layer

Focus

- Generally applicable
- 6 Key Points
- Q & A

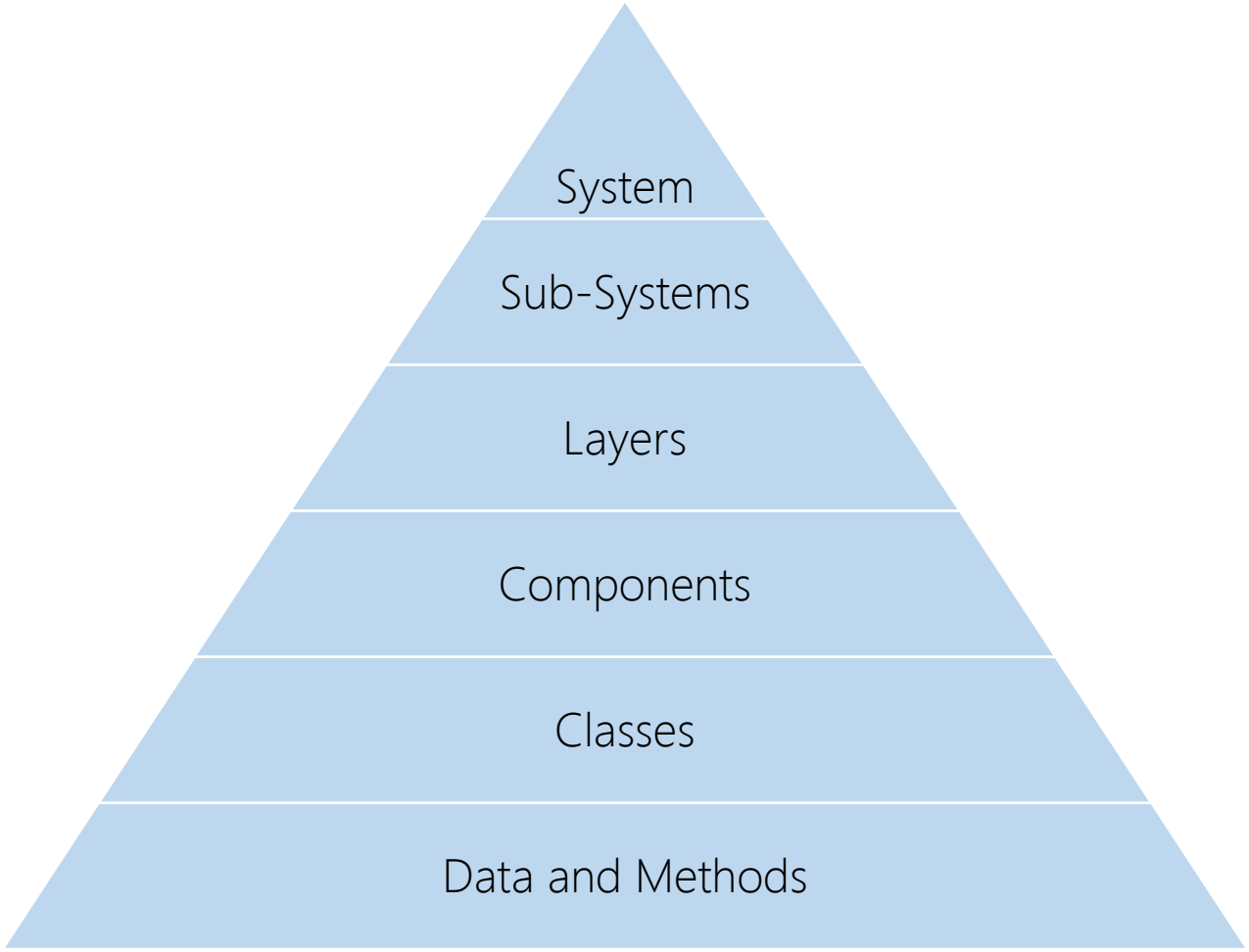
What is Software Architecture?

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

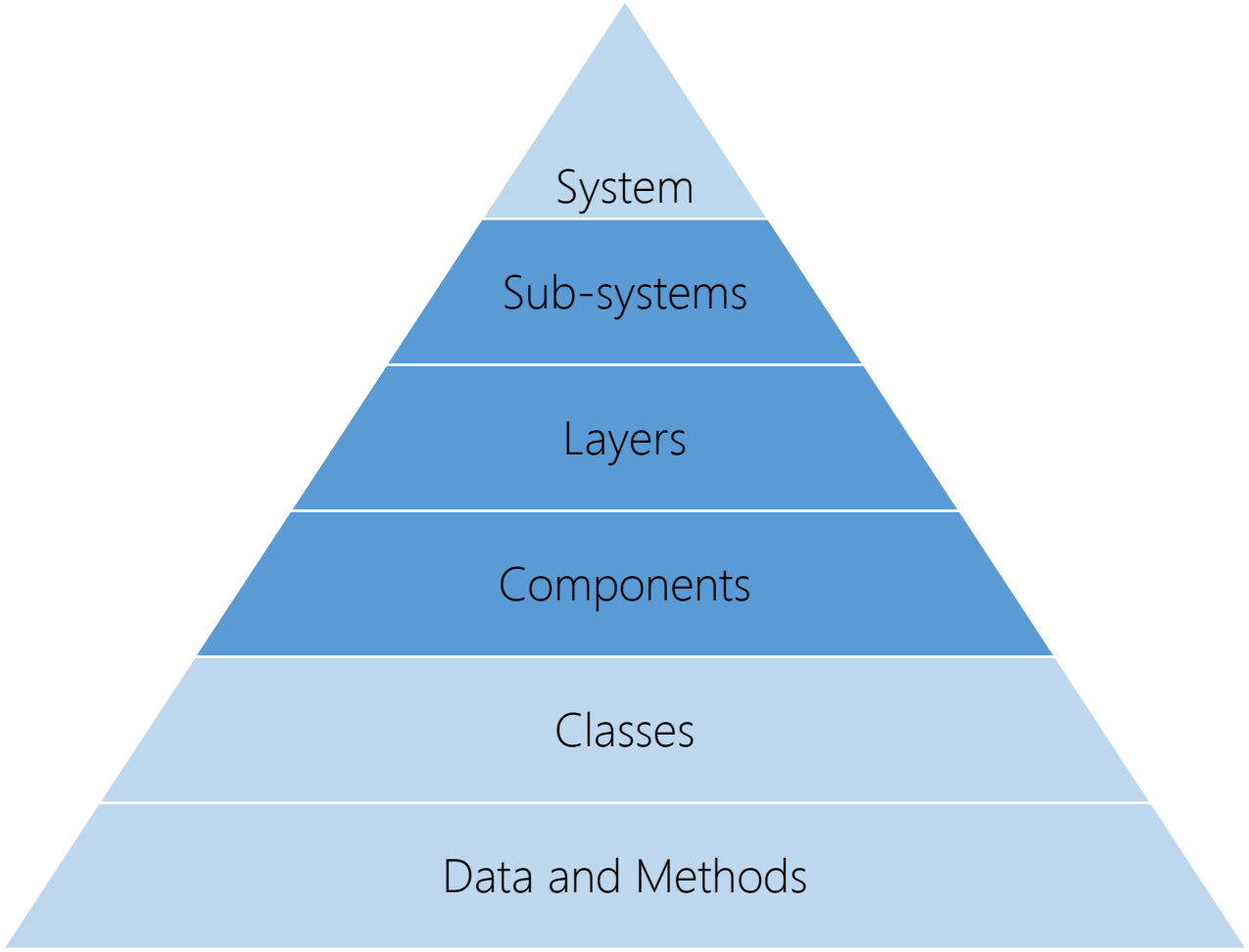


Source: http://msdn.microsoft.com/en-us/library/ff650706.aspx

Levels of Architectural Abstraction



Levels of Architectural Abstraction



Messy vs Clean Architecture

Messy vs Clean Architecture

Messy vs Clean Architecture

What is Bad Architecture?

- Complex
- Inconsistent
- Incoherent
- Ridged
- Brittle
- Untestable
- Unmaintainable

What is Clean Architecture?

- Simple
- Understandable
- Flexible
- Emergent
- Testable
- Maintainable

What is Clean Architecture?

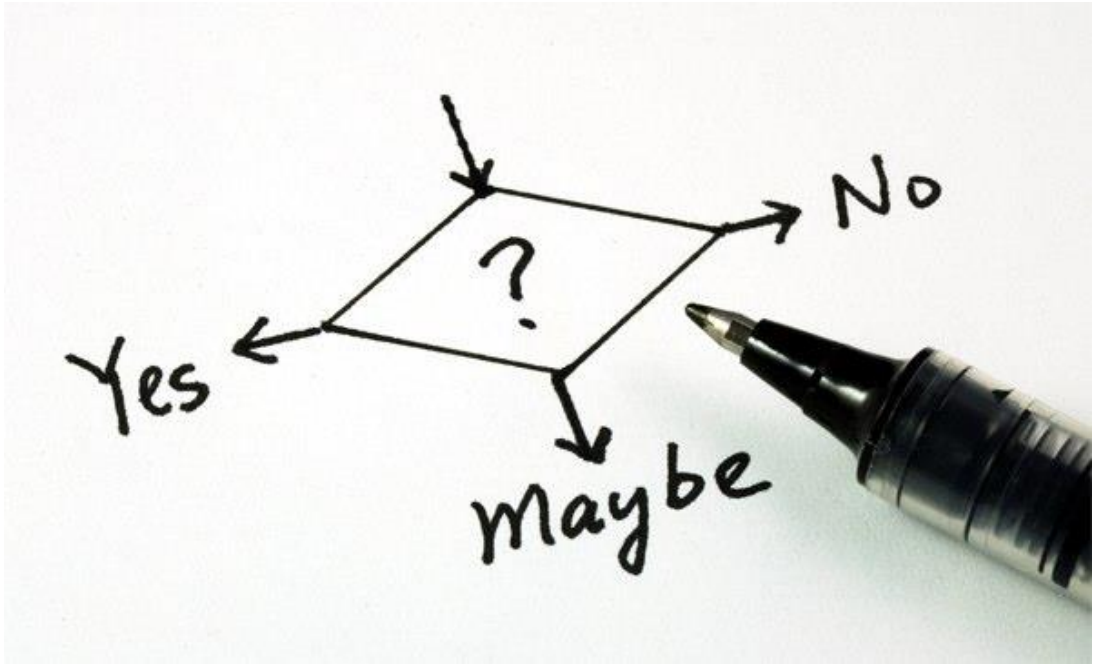
Architecture that is designed for the inhabitants of the architecture…
not for the architect… or the machine

Why is Clean Architecture Important?

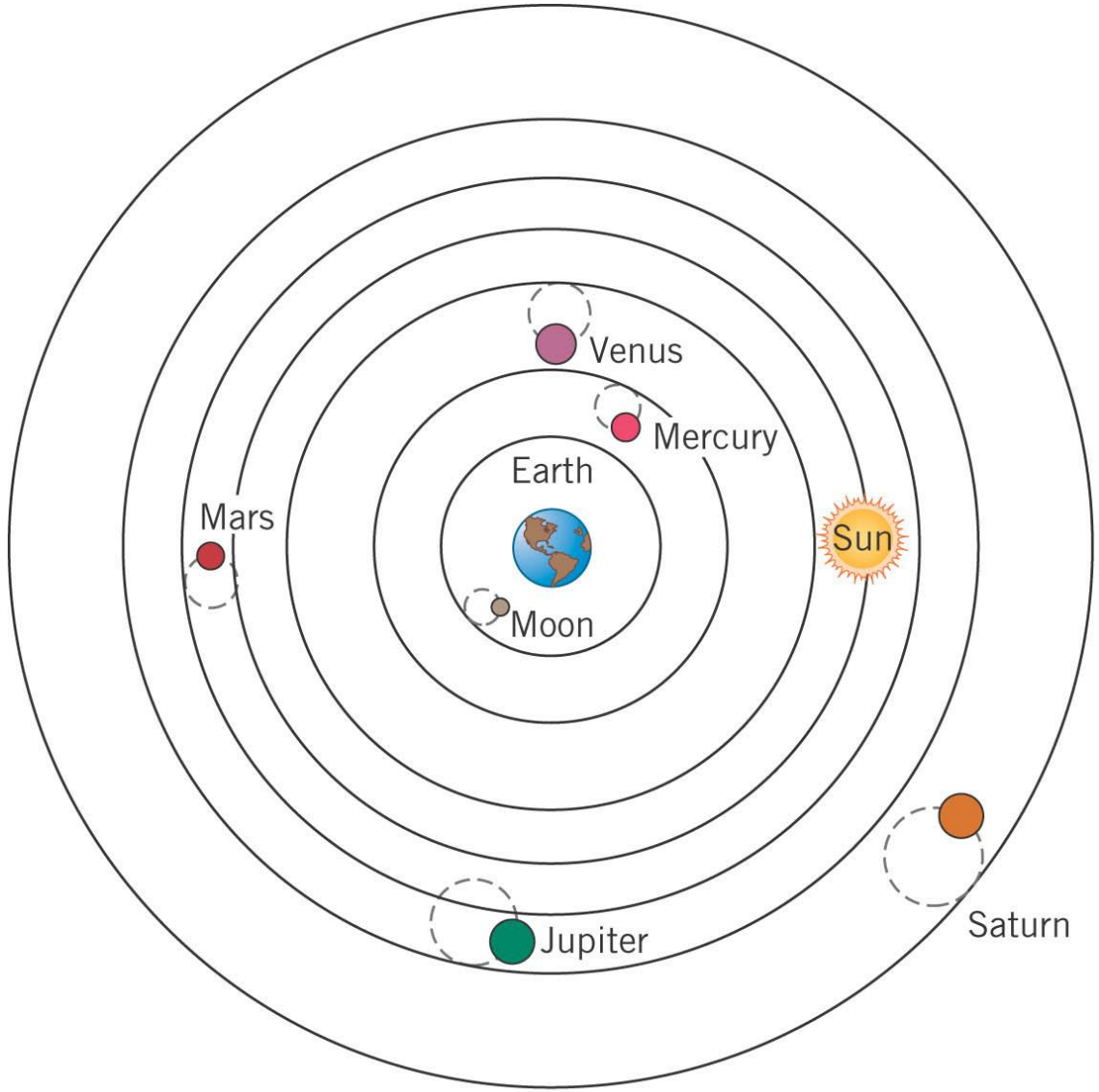
- Cost/benefit
- Minimize cost to maintain
- Maximize business value

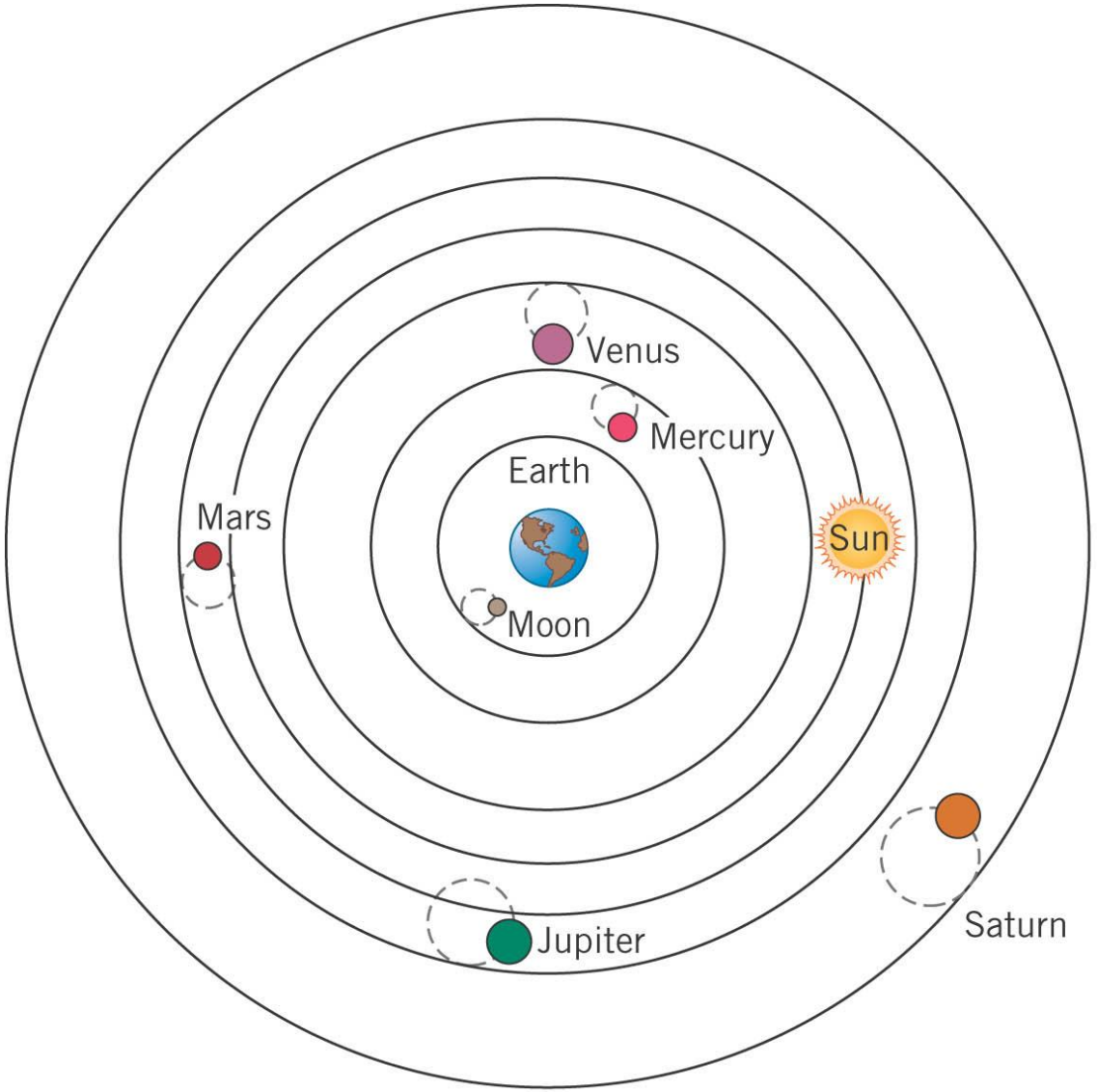
Decisions, Decisions, Decisions…

- Context is king
- All decisions are a tradeoff
- Use your best judgement

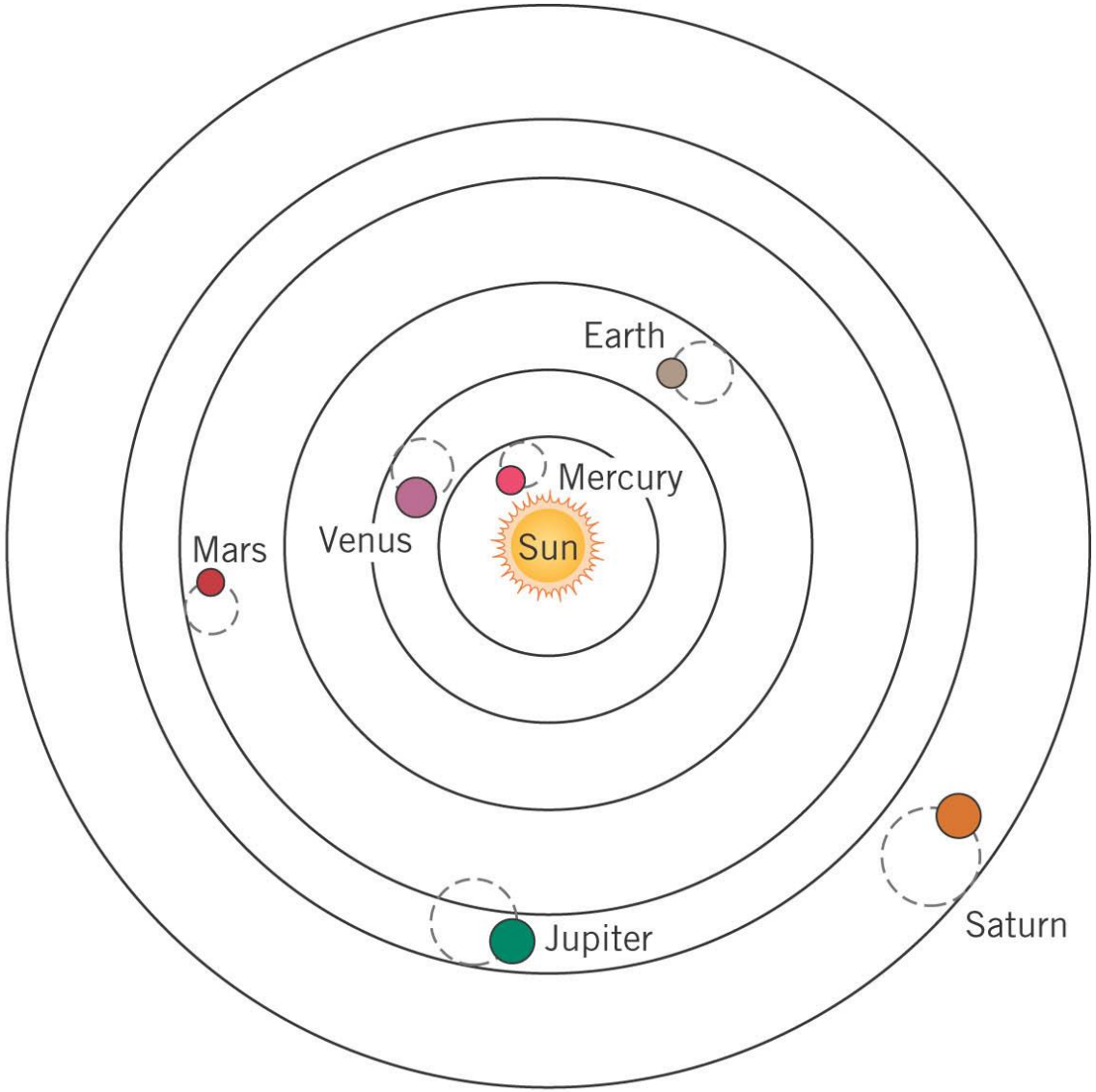


Domain-Centric Architecture


Venus
Mercury
Earth
Mars
Sun
Moon
Jupiter
Saturn

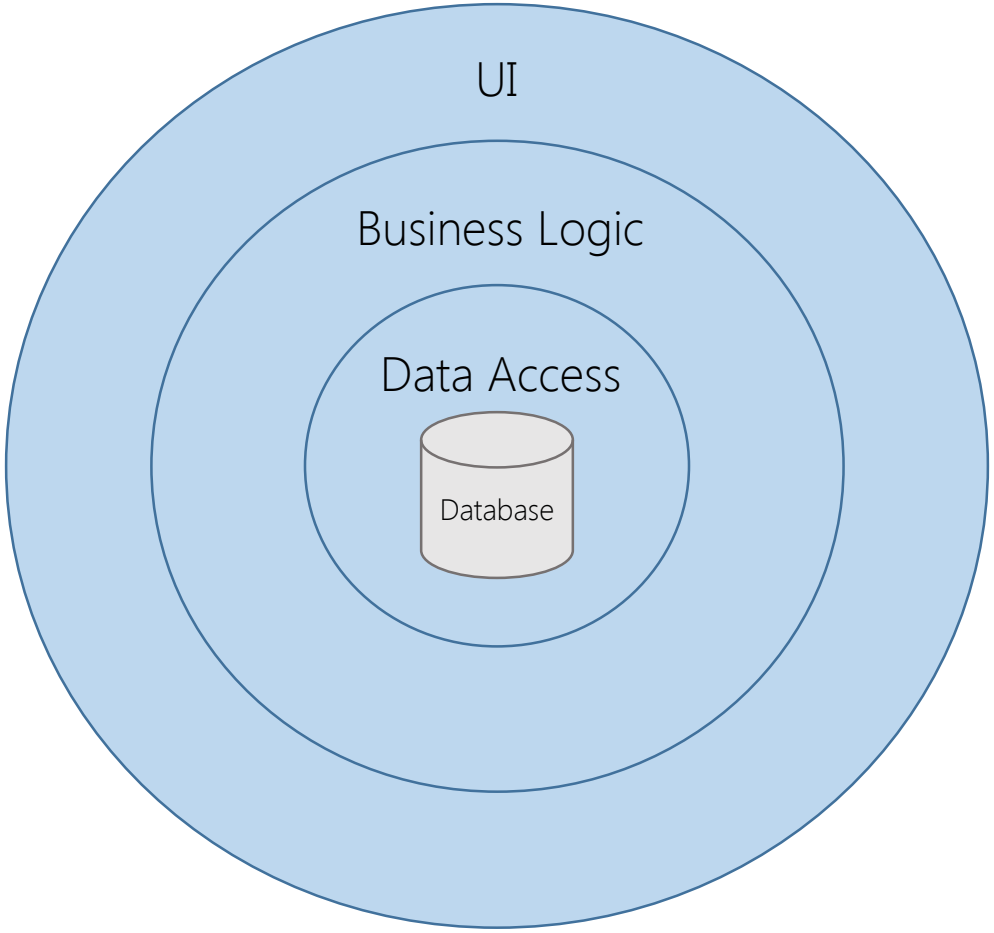


Geocentric Model

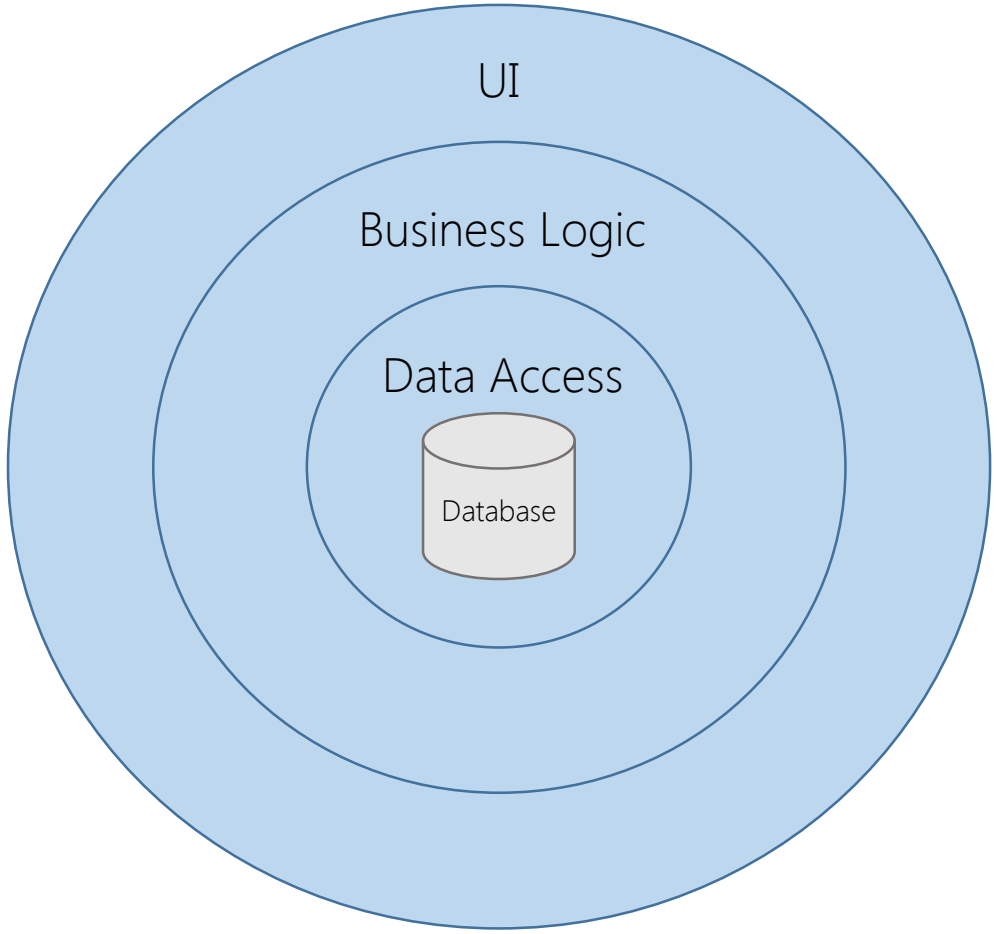


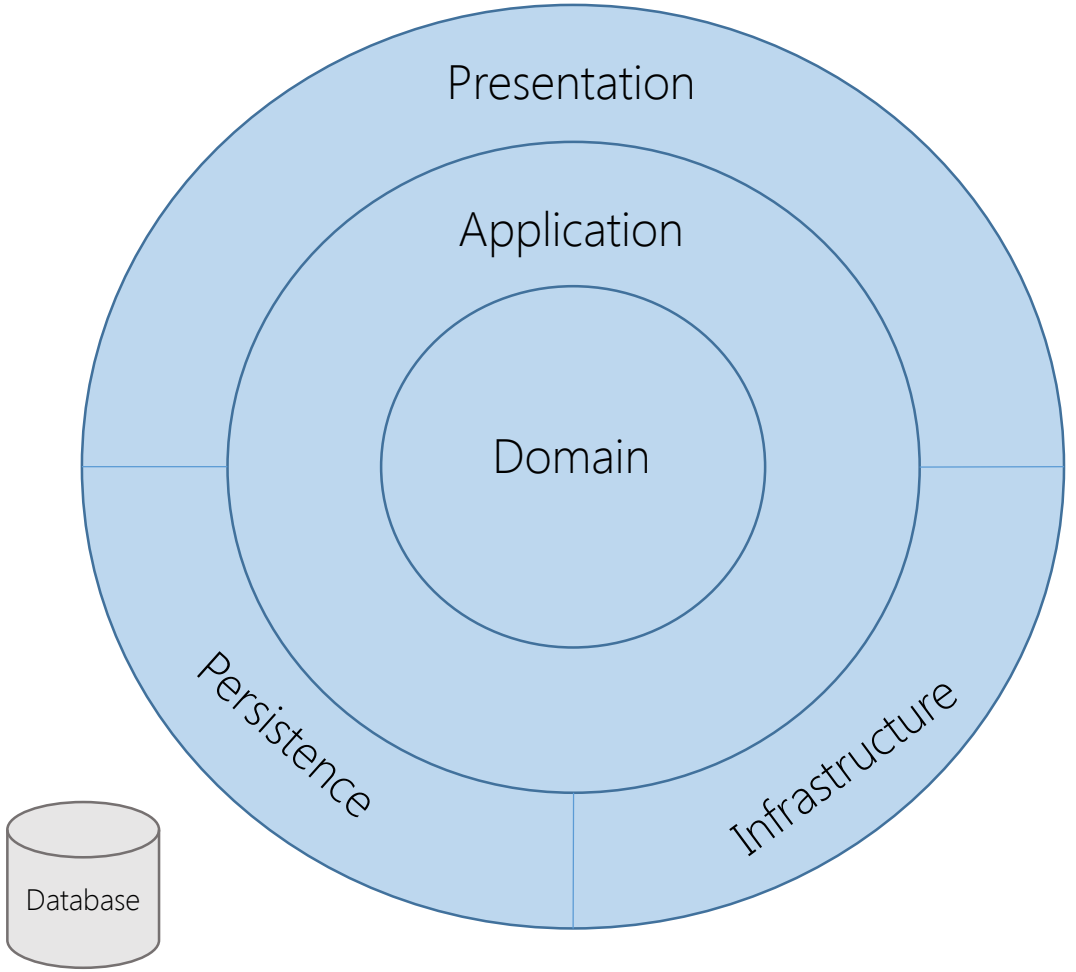
Heliocentric Model

Classic 3-Layer Database-centric Architecture



Database- vs. Domain-centric Architecture





"The first concern of the architect is to make sure that the house is usable, it is not to ensure that the house is made of brick."

– Uncle Bob

Essential vs. Detail

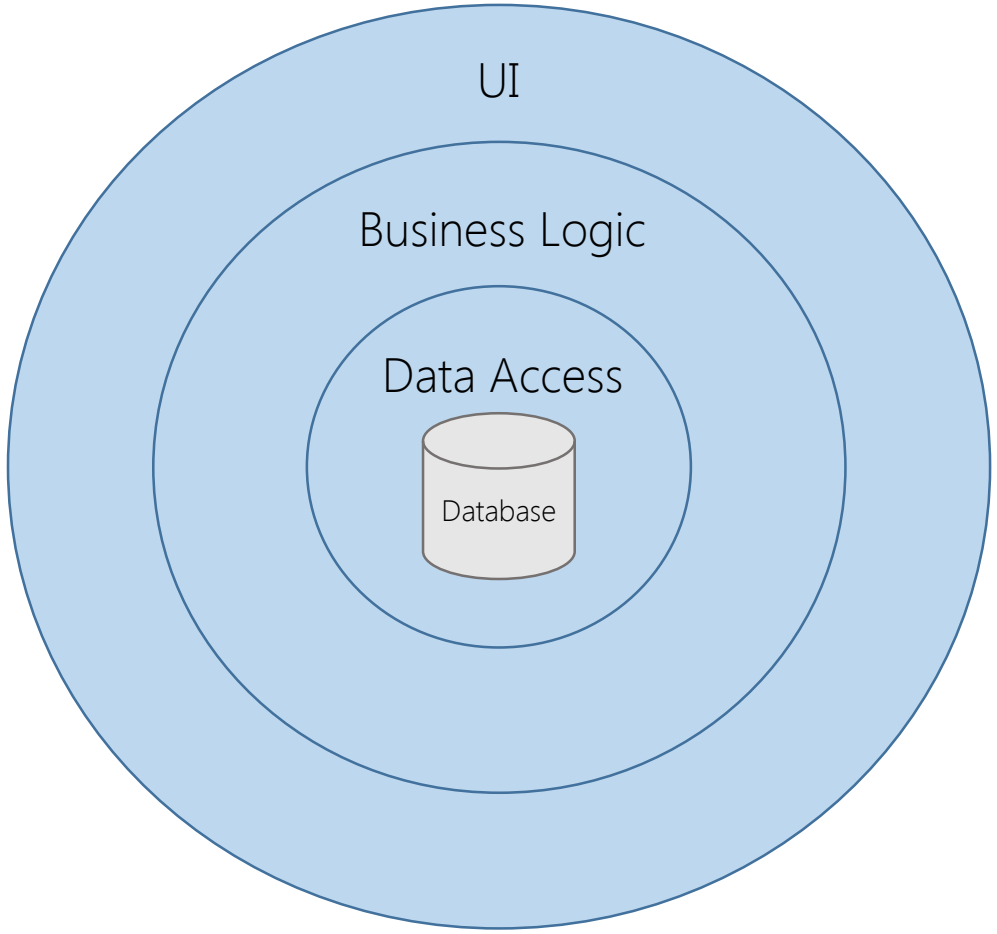
- Space is essential
- Usability is essential
- Building material is a detail
- Ornamentation is a detail

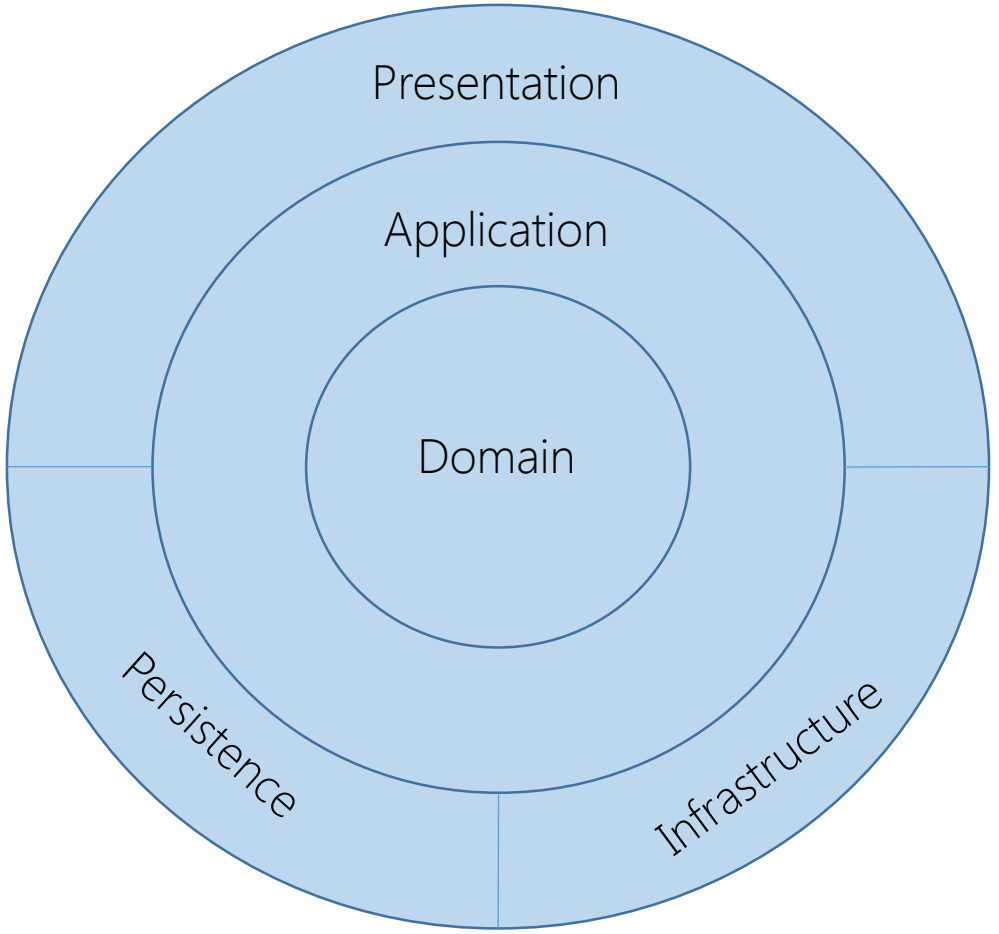
Source: http://www.whitegadget.com/attachments/pc-wallpapers/85254d1320380902-house-house-wallpaper.jpg

Essential vs. Detail

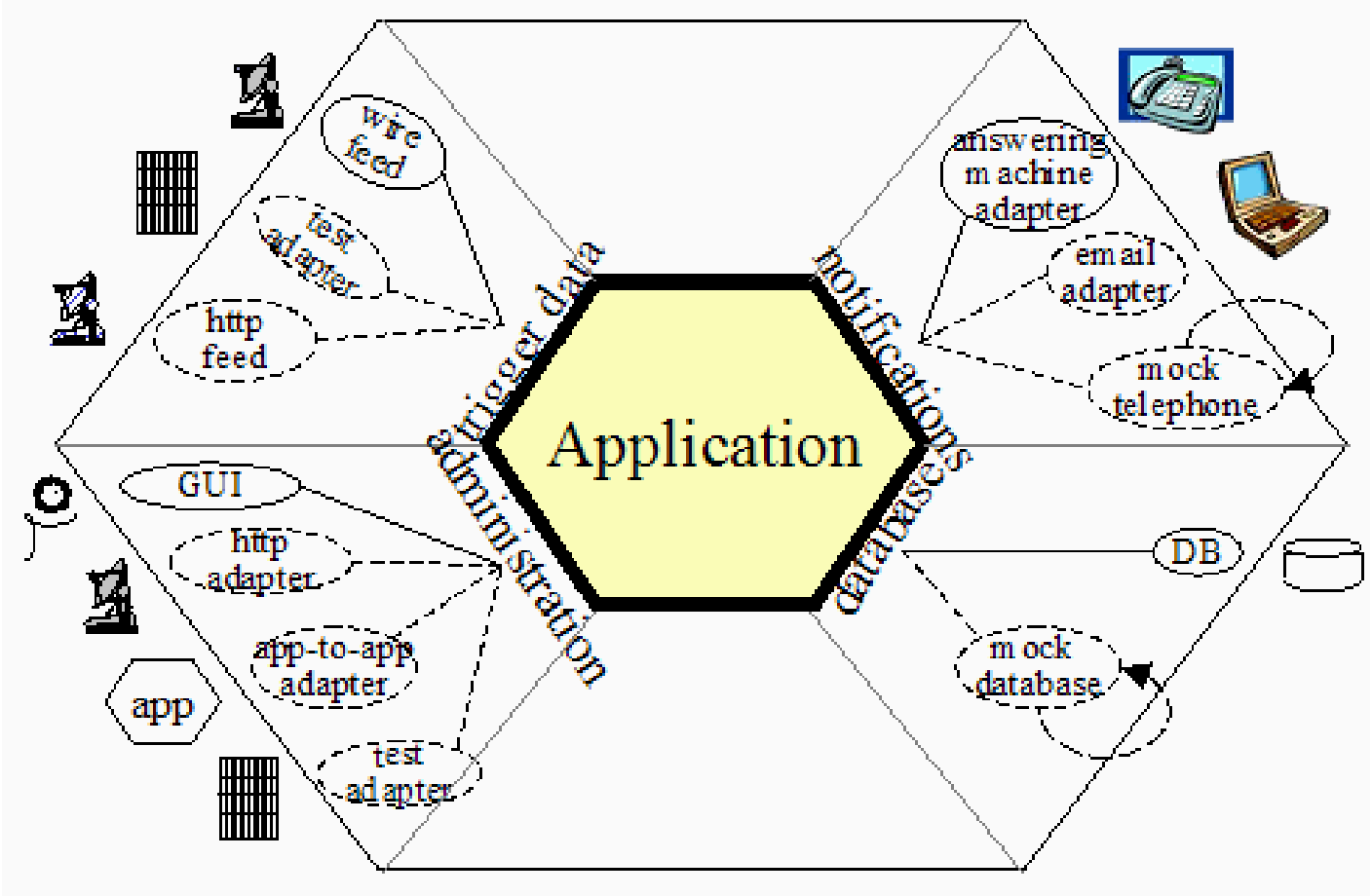
- Domain is essential
- Use cases are essential
- Presentation is a detail
- Persistence is a detail

Database- vs. Domain-centric Architecture

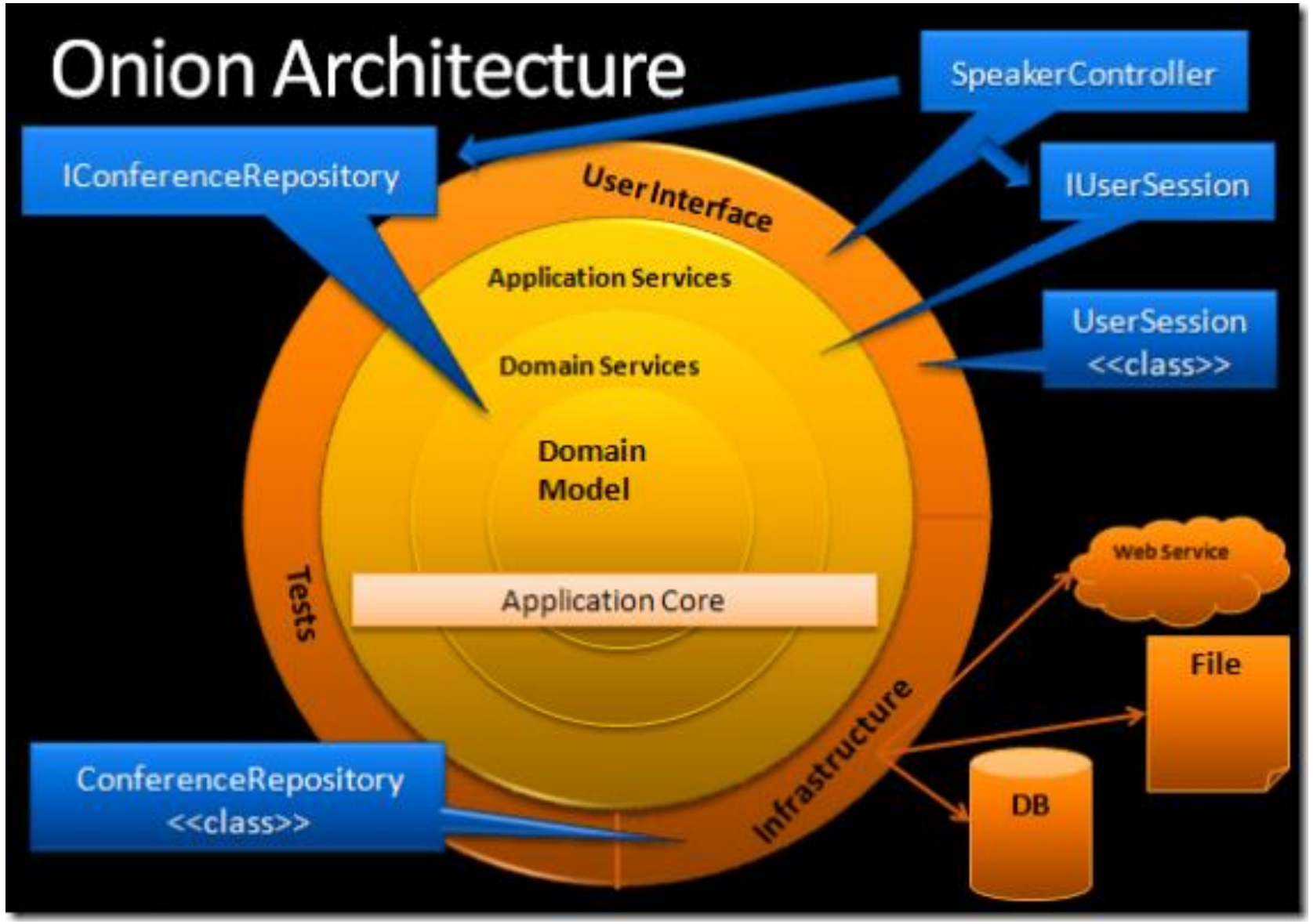




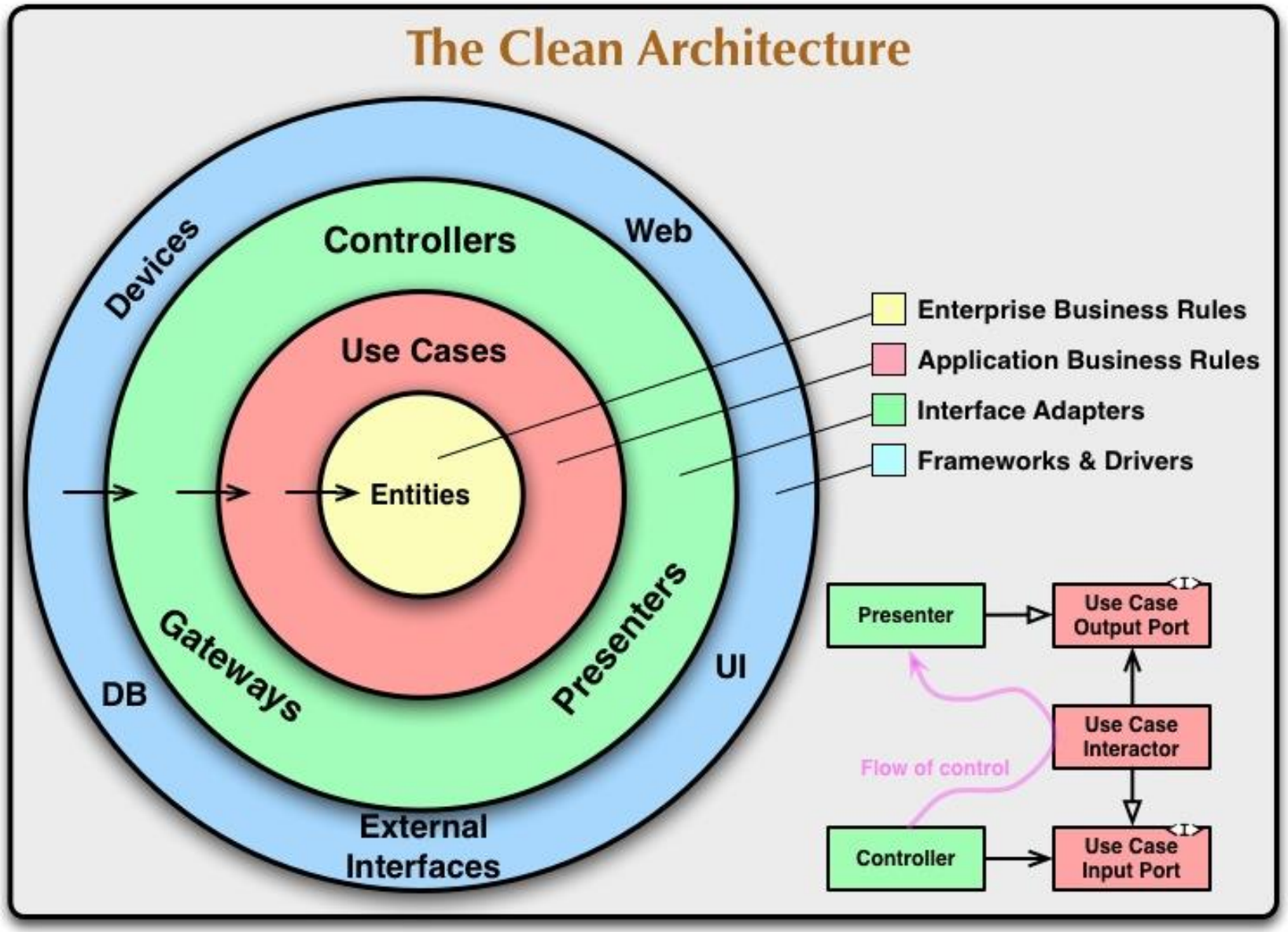
Hexagonal Architecture



Source: http://alistair.cockburn.us/Hexagonal+architecture

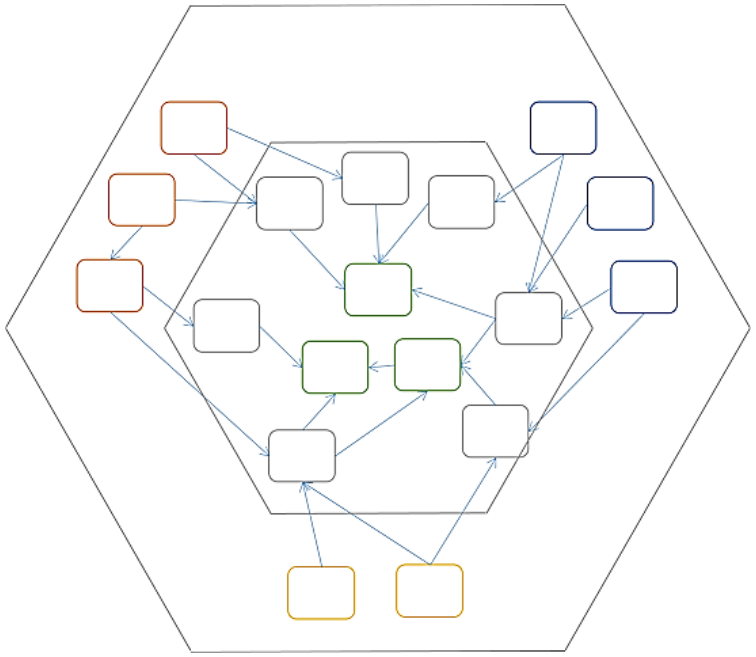


Source: http://jeffreypalermo.com/blog/the-onion-architecture-part-2/

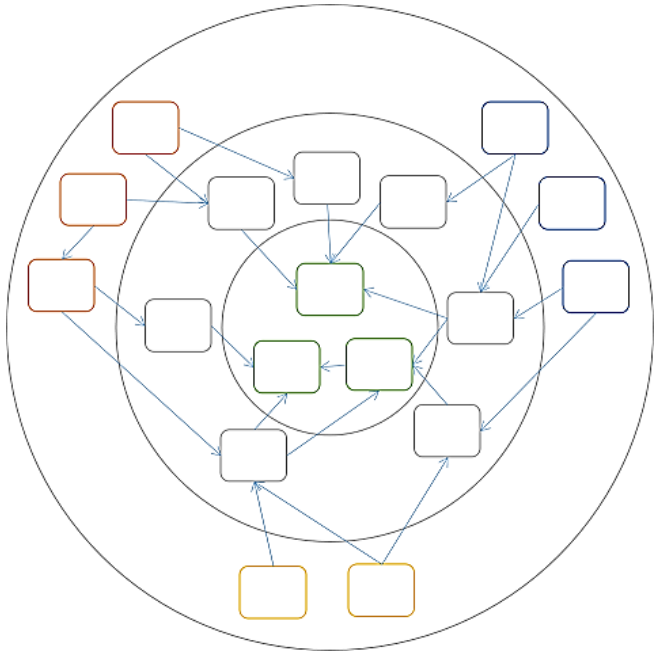


Source: http://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html

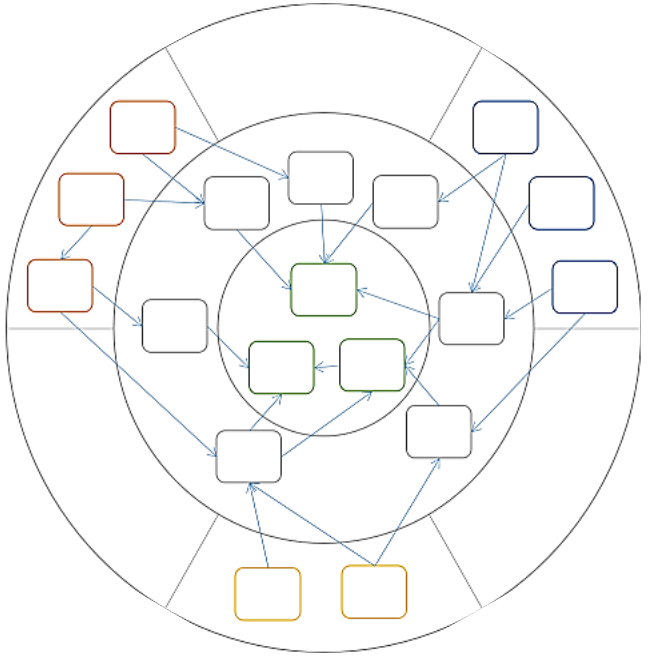
It's All the Same Thing



Hexagonal



Onion



Clean

http://blog.ploeh.dk/2013/12/03/layers-onions-ports-adapters-its-all-the-same/

Why Use Domain-Centric Architecture?

Pros

- Focus on essential
- Less coupling to details
- Necessary for DDD

Why Use Domain-Centric Architecture?

Pros

- Focus on essential
- Less coupling to details
- Necessary for DDD

Cons

- Change is difficult
- Requires extra thought
- Initial higher cost

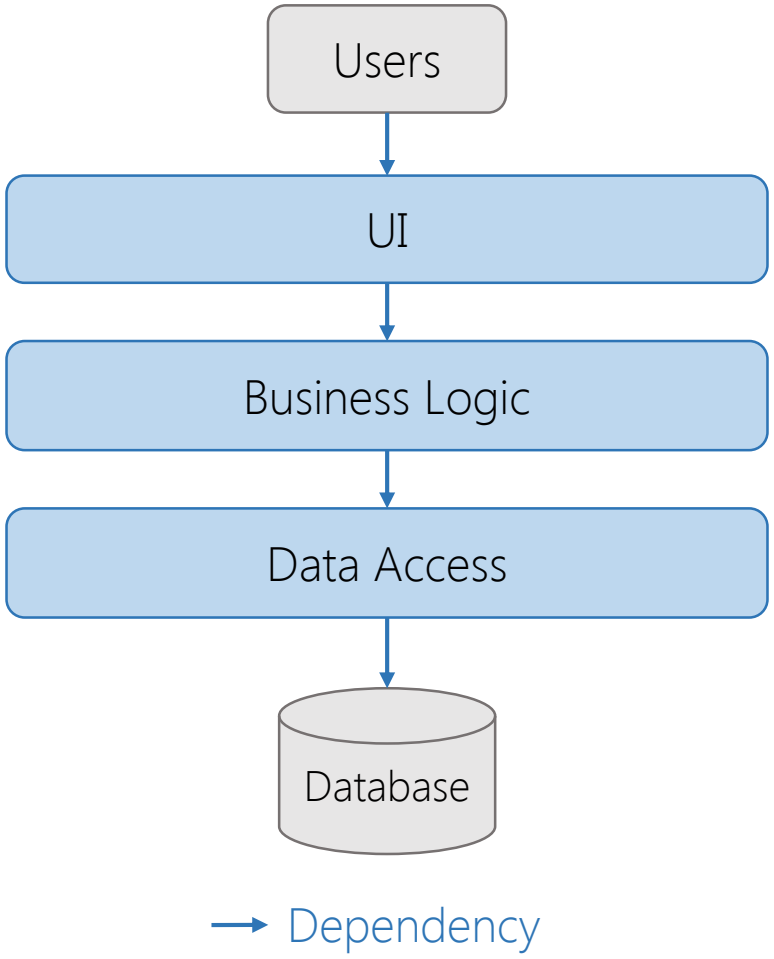
Application Layer

What are Layers?

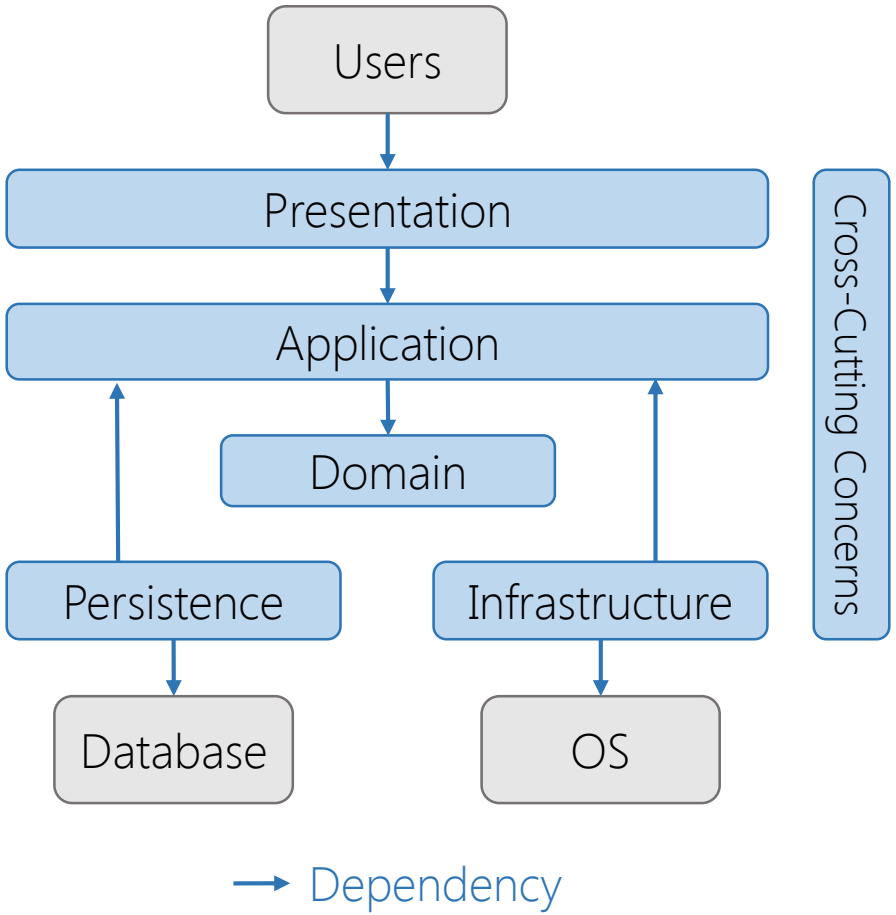
- Levels of abstraction
- Single-Responsibility Principle
- Developer roles / skills
- Multiple implementations
- Varying rates of change

Source: http://www.followmefoodie.com/wp-content/uploads/2012/03/Spumone-Layered-Cake.jpg

Classic 3-Layer Architecture

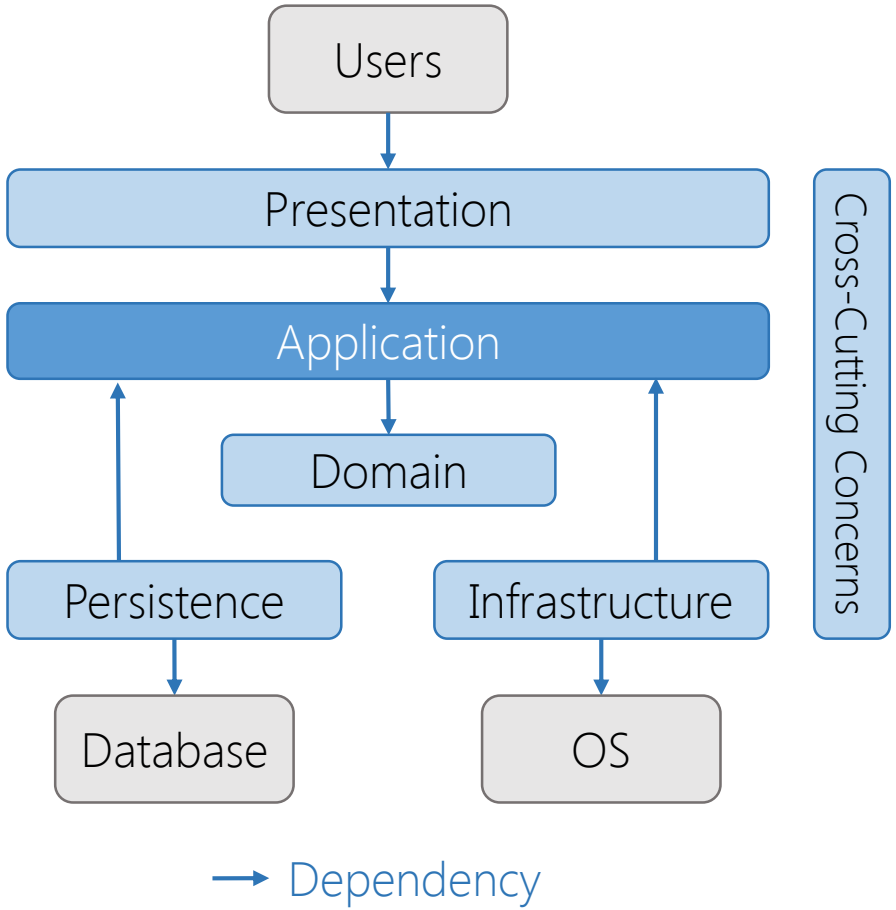


Modern 4-Layer Architecture



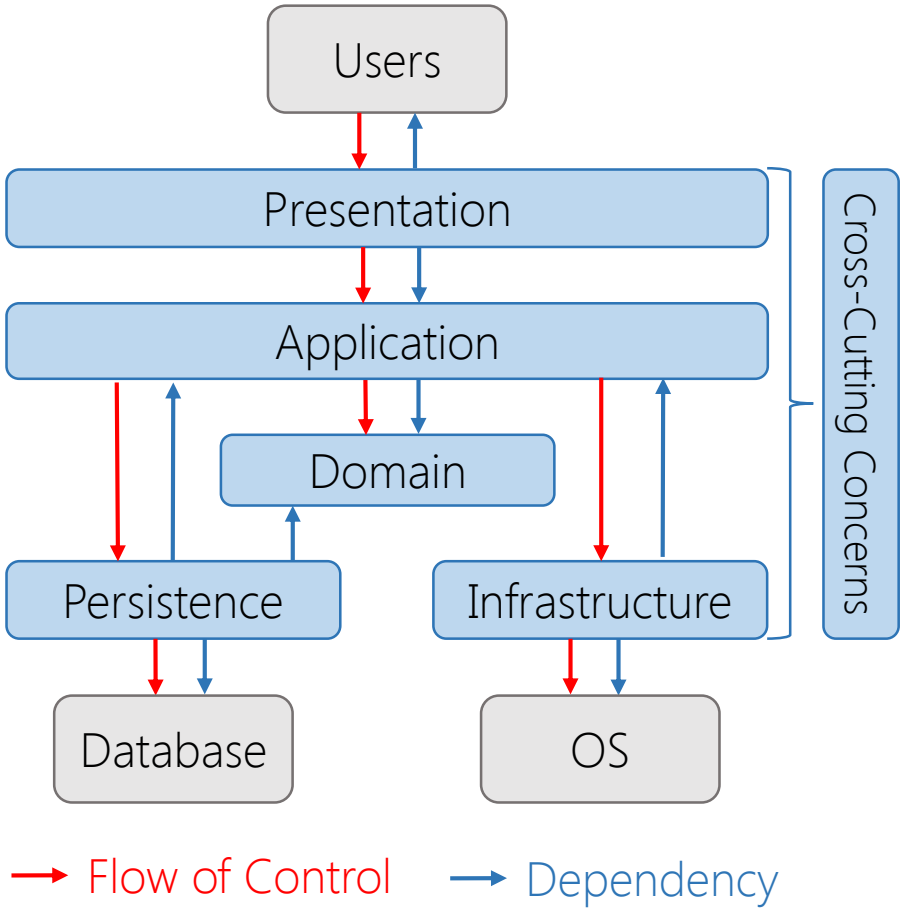
Application Layer

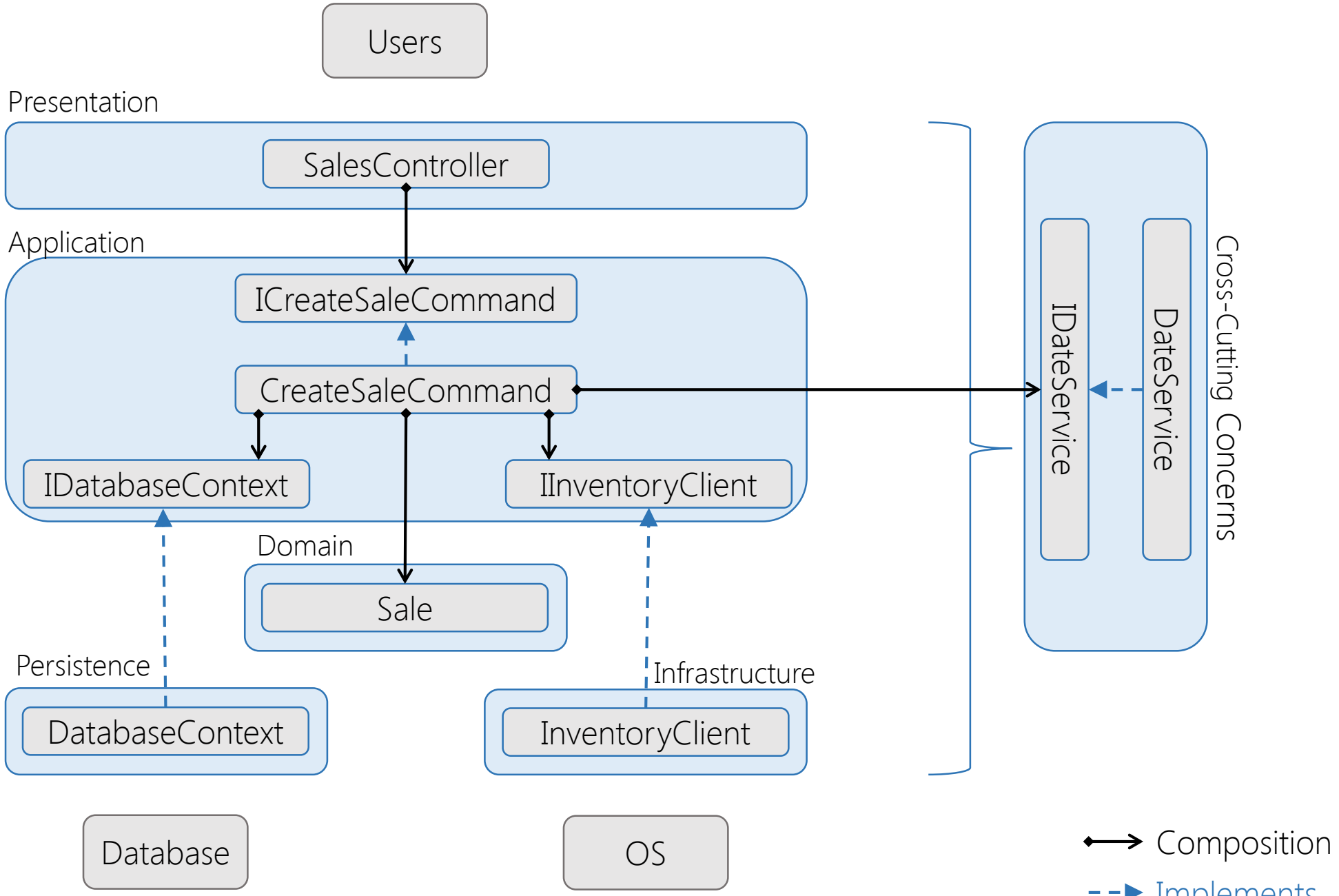
- Implements use cases
- High-level application logic
- Knows about lower layers
- No knowledge of upper layers
- Contains interfaces for details



Layer Dependencies

- Dependency inversion
- Inversion of control
- Independent deployability
- Flexibility and maintainability




Users
Presentation
SalesController
Application
ICreateSaleCommand
CreateSaleCommand
IDatabaseContext
IInventoryClient
Domain
Sale
Persistence
DatabaseContext
Infrastructure
InventoryClient
Database
OS
IDateService
DateService
Cross-Cutting Concerns
Composition
Implements

Why Use an Application Layer?

Pros

- Focus is on use cases
- Easy to understand
- Follows DIP

Why Use an Application Layer?

Pros

- Focus is on use cases
- Easy to understand
- Follows DIP

Cons

- Additional cost
- Requires extra thought
- IoC is counter-intuitive

Commands and Queries

Command-Query Separation

Command

- Does something
- Modifies state
- Should not return a value

Command-Query Separation

Command

- Does something
- Modifies state
- Should not return a value

Query

- Answers a question
- Does not modify state
- Always returns a value

Command-Query Separation

Command

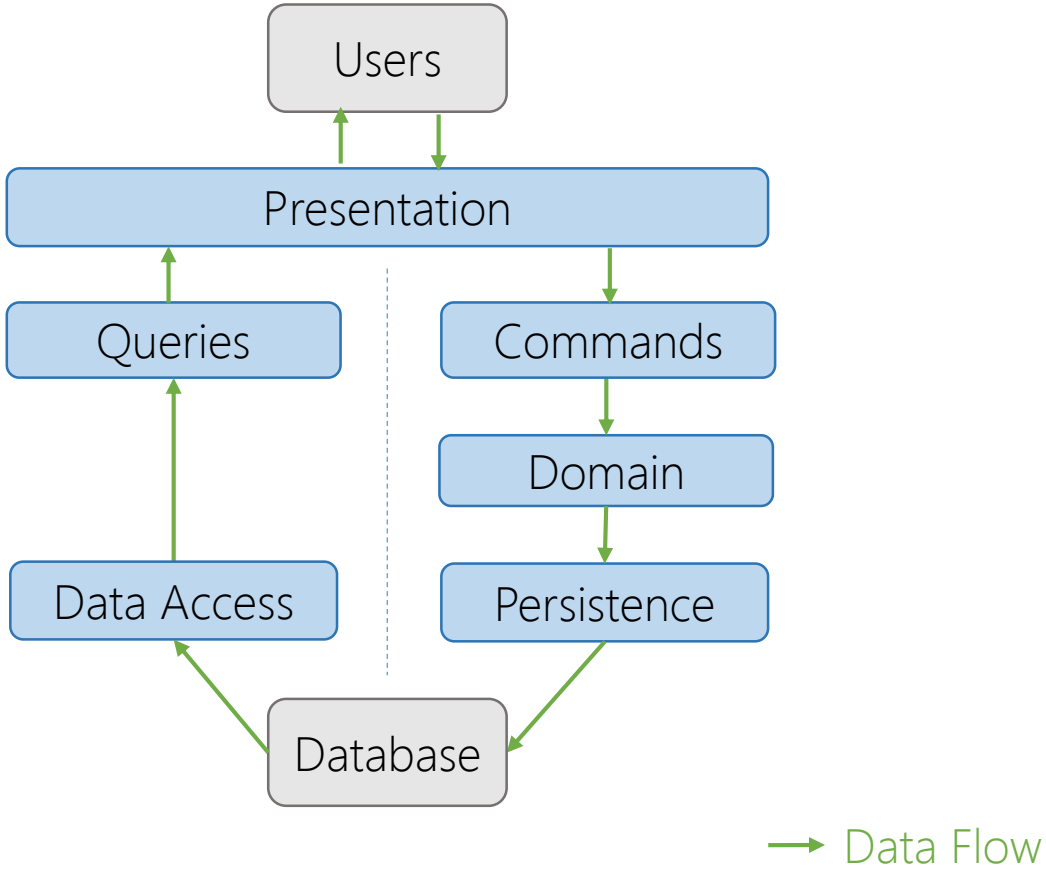
- Does something
- Modifies state
- Should not return a value *(ideally)*

Query

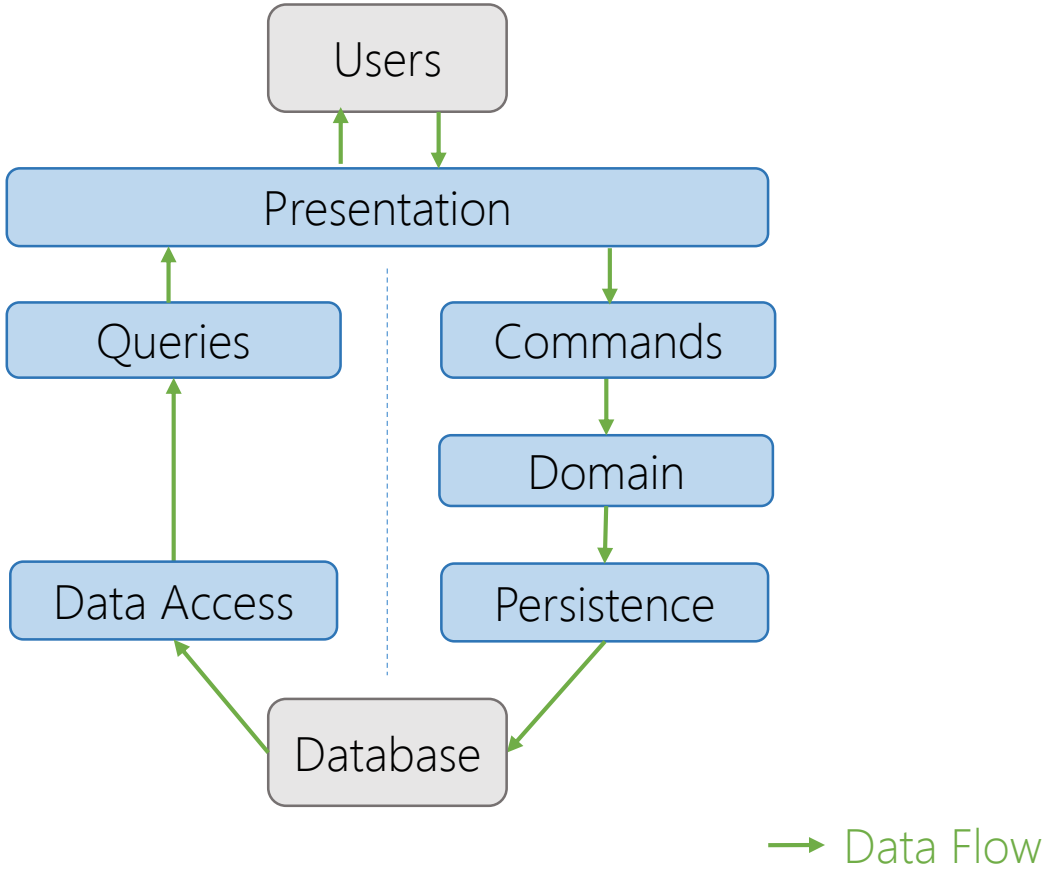
- Answers a question
- Does not modify state
- Always returns a value

Avoid mixing the two!

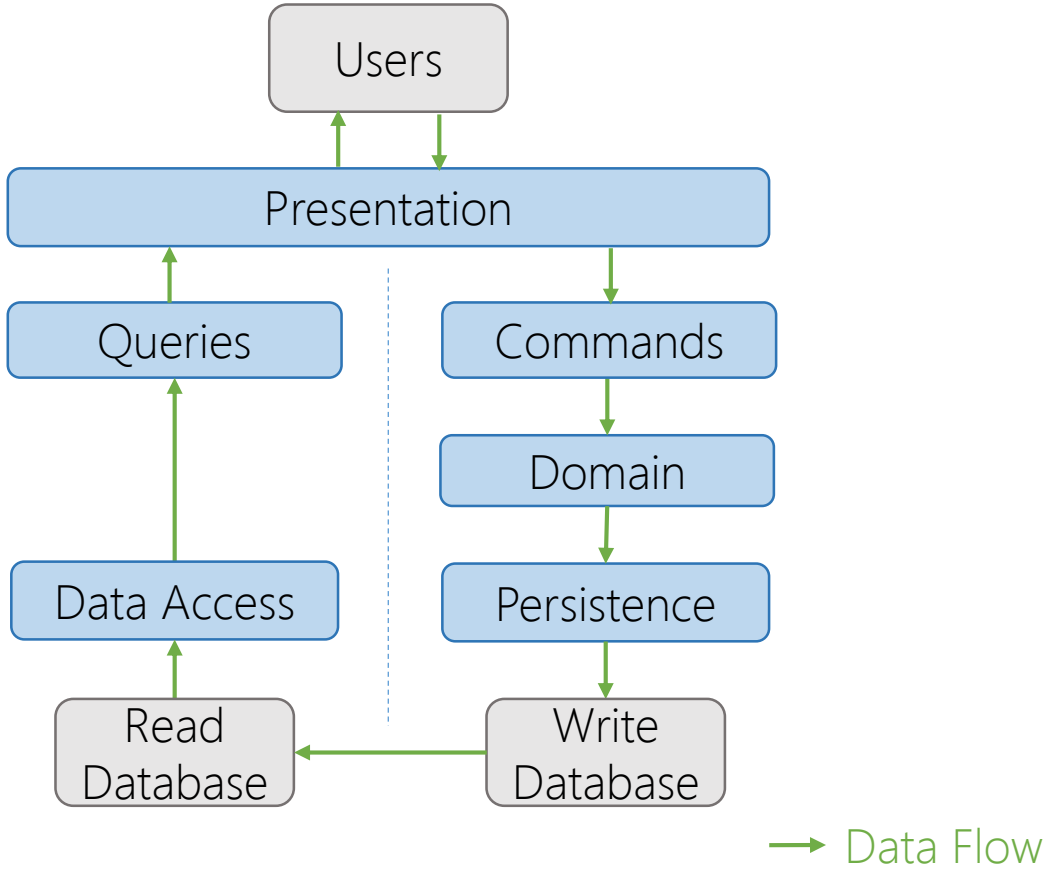
CQRS Architectures



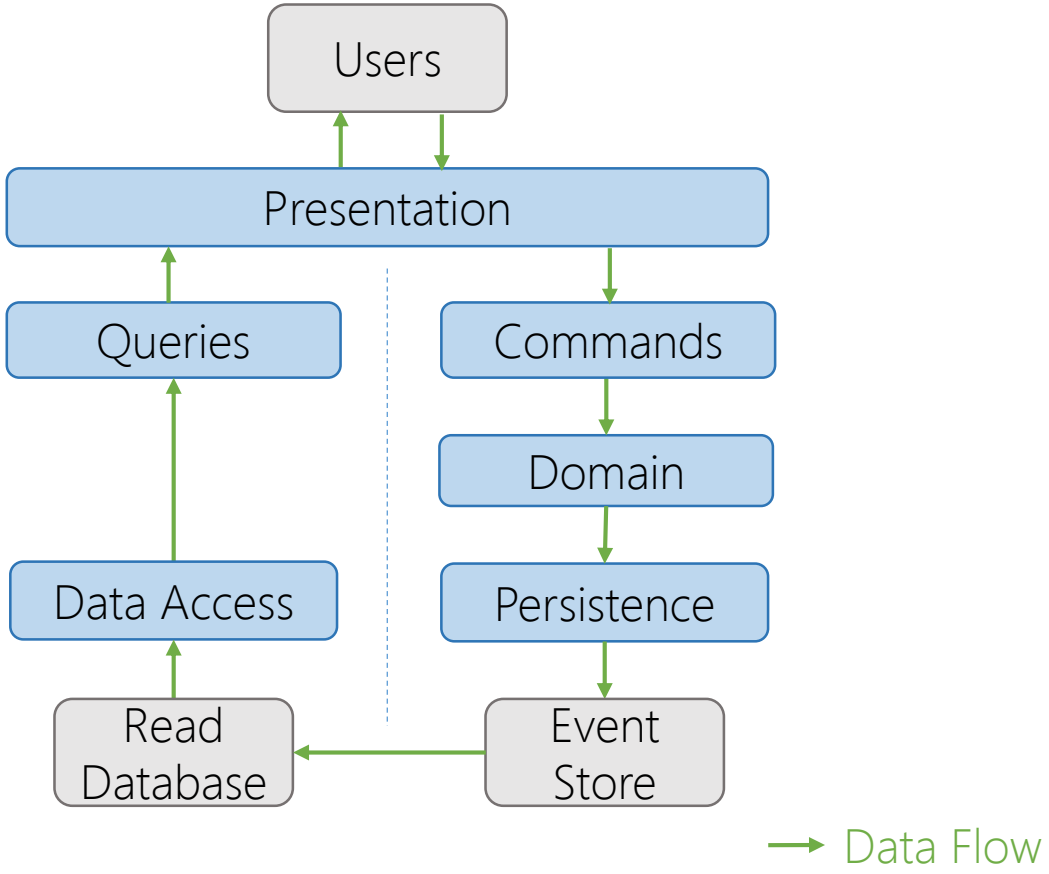
CQRS Type 1 – Single Database



CQRS Type 2 – Read/Write Databases

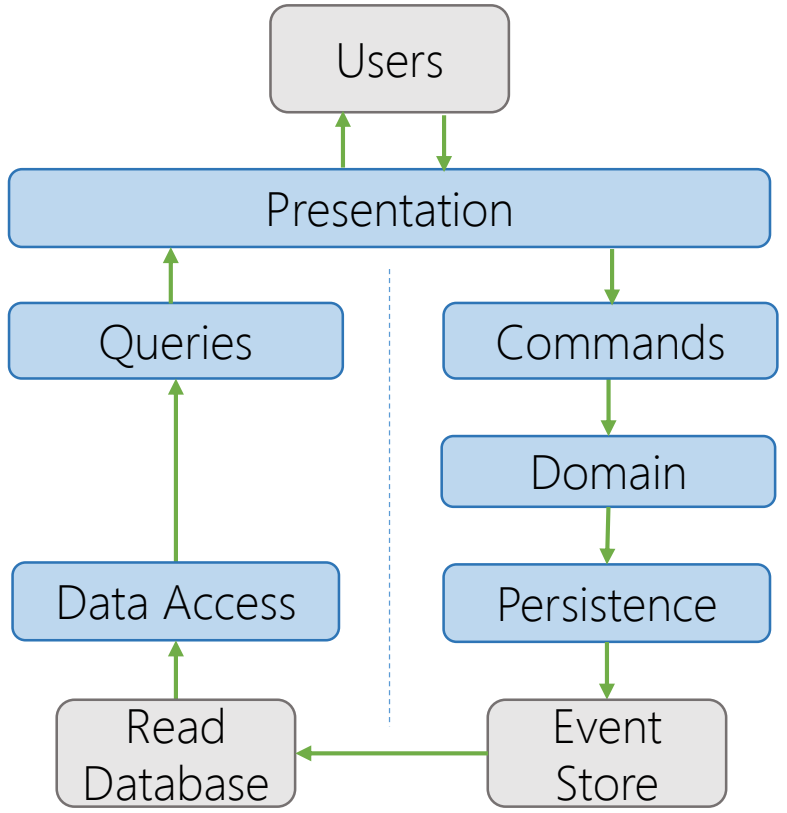


CQRS Type 3 – Event Sourcing



CQRS Type 3 – Event Sourcing

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



Why Use CQRS?

Pros

- More efficient design
- Simpler within each stack
- Optimized performance

Why Use CQRS?

Pros

- More efficient design
- Simpler within each stack
- Optimized performance

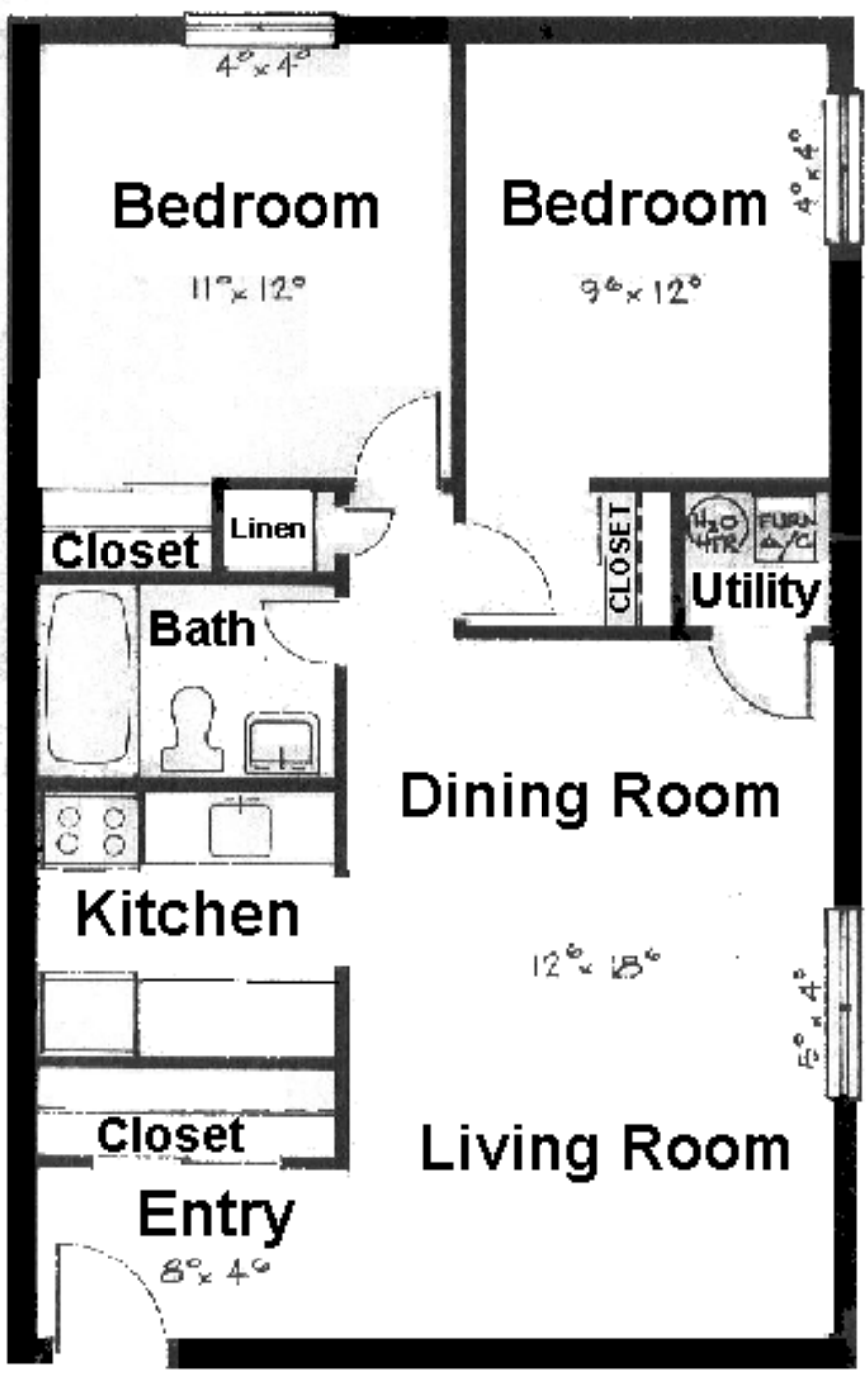
Cons

- Inconsistent across stacks
- Type 2 is more complex
- Type 3 might be overkill

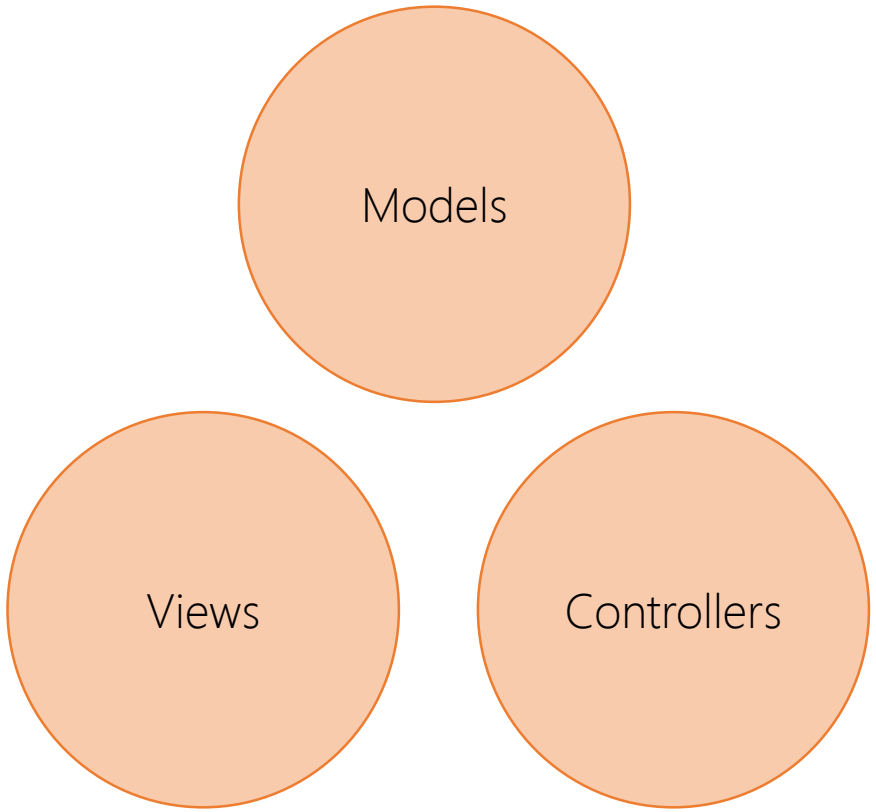
Functional Organization

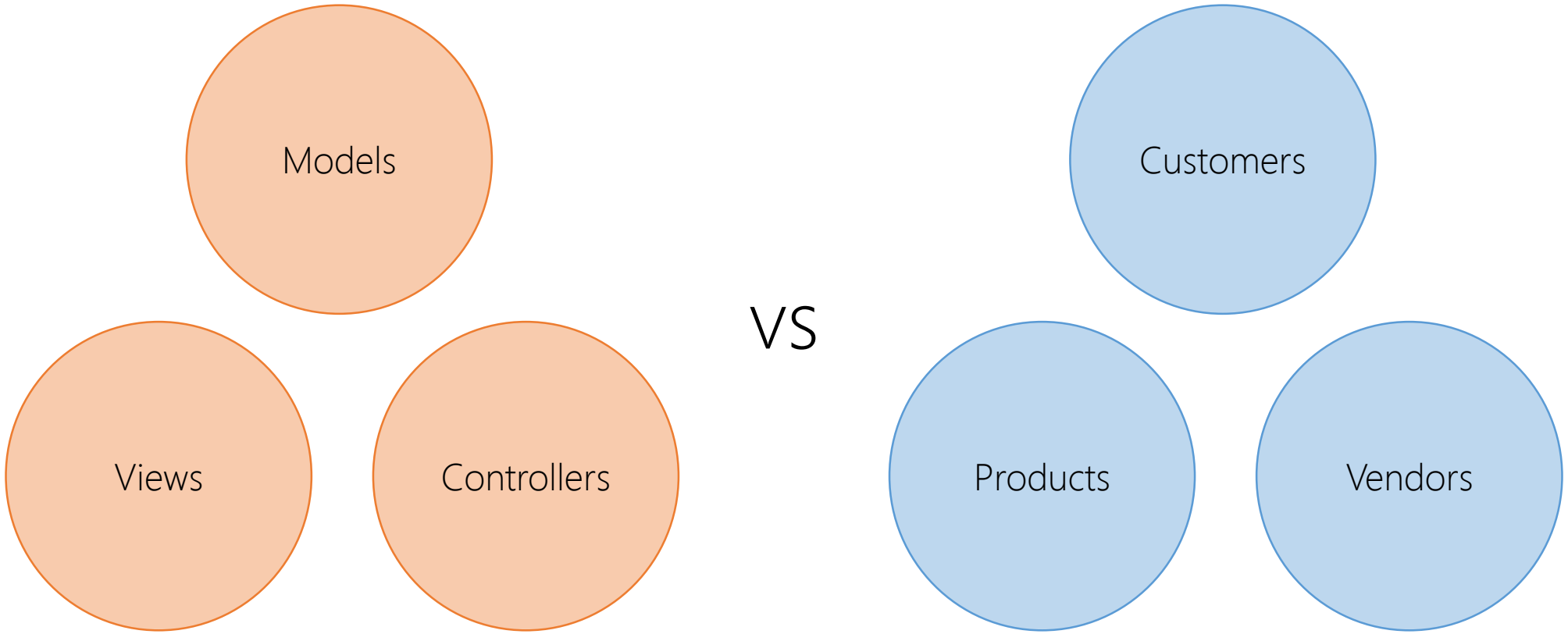
"The architecture should scream
the intent of the system!"

– Uncle Bob


4' x 4'
Bedroom
11' x 12'
Bedroom
9'6 x 12'
4' x 4'
Linen
Closet
Bath
CLOSET
H2O HTR
FURN A/C
Utility
Dining Room
Kitchen
12'6 x 15'
5' x 4'
Closet
Living Room
Entry
8' x 4'

Material	Quantity	Cost
Appliances	5	$5,000
Cabinets	10	$2,500
Doors	15	$750
Fixtures	12	$2,400
Floors	9	$4,000
Walls	20	$10,000
Windows	8	$2,500


Models
Views
Controllers


Models
Views
Controllers
VS
Customers
Products
Vendors

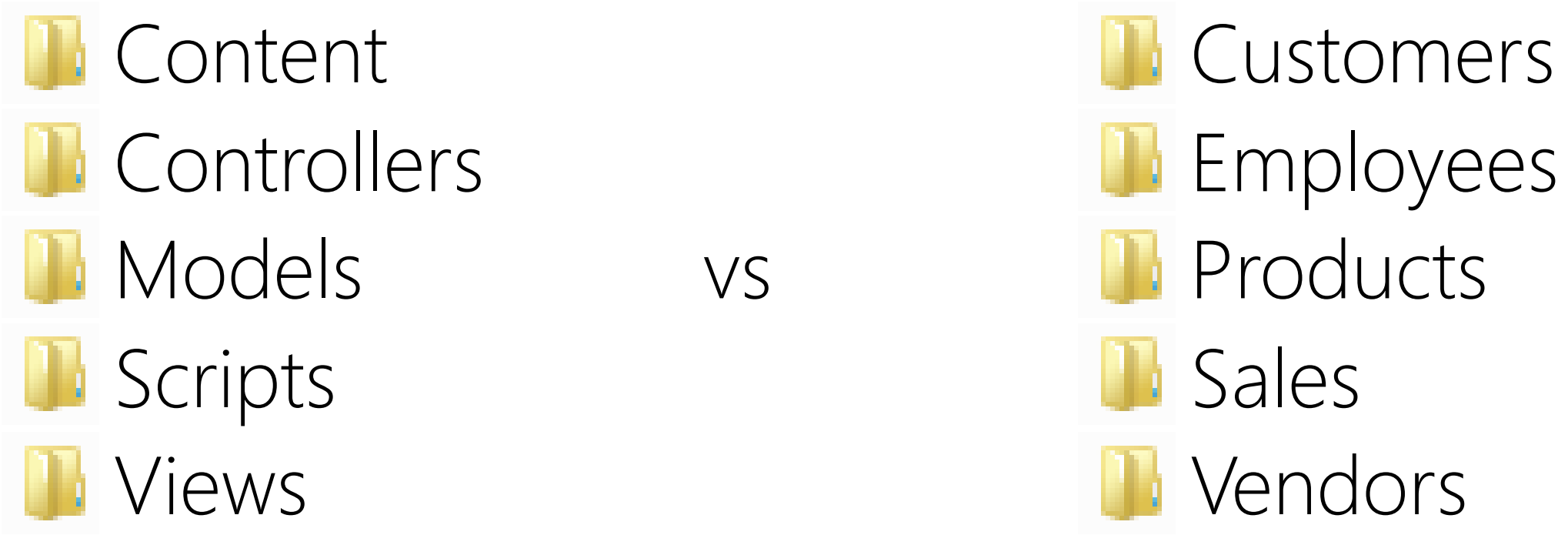
Content

Controllers

Models

Scripts

Views


Content
Controllers
Models
Scripts
Views
vs
Customers
Employees
Products
Sales
Vendors

So what?

VS

Why Use Functional Organization

Pros

- Spatial locality
- Easy to navigate
- Avoid vendor lock-in

Why Use Functional Organization

Pros

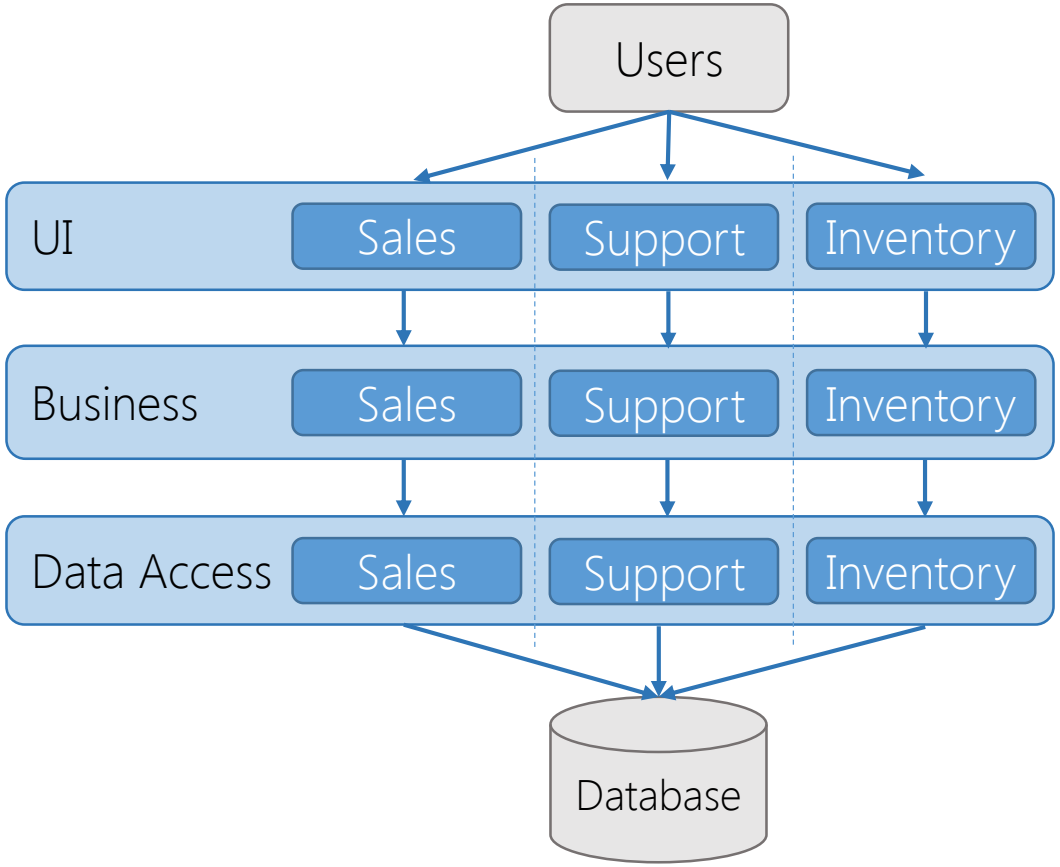
- Spatial locality
- Easy to navigate
- Avoid vendor lock-in

Cons

- Lose framework conventions
- Lose automatic scaffolding
- Categorical is easier at first

Microservices

Components


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

Problem Domain

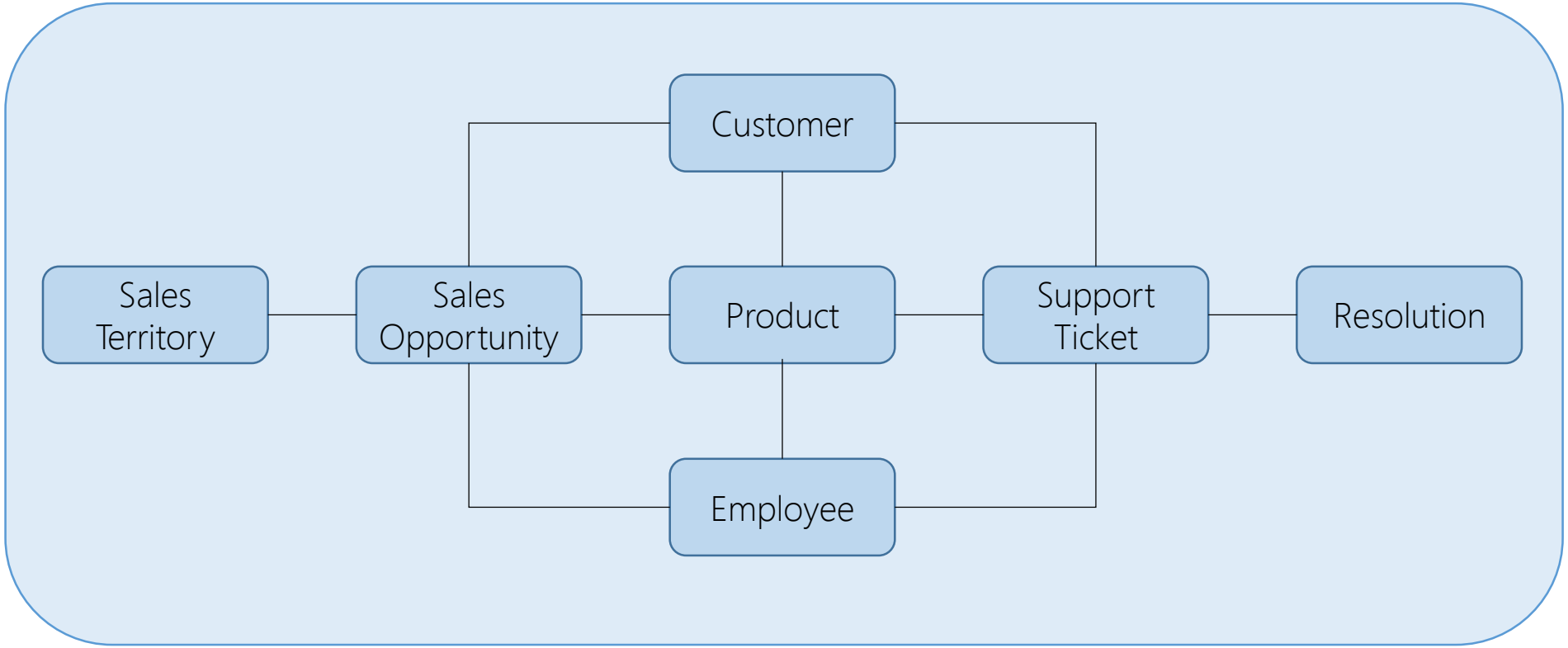
Sales

- Sales Opportunity
- Contact
- Sales Person
- Product
- Sales Territory

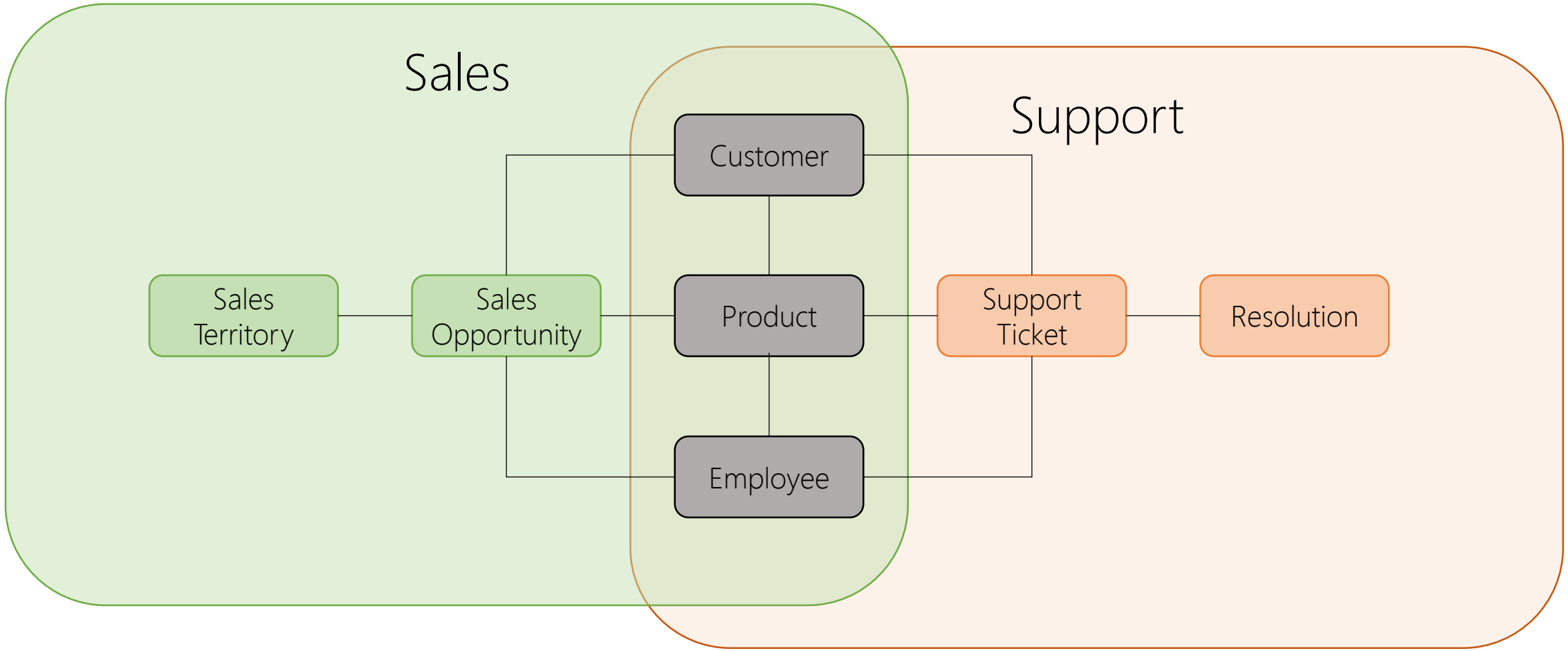
Support

- Support Ticket
- Customer
- Support Person
- Product
- Resolution

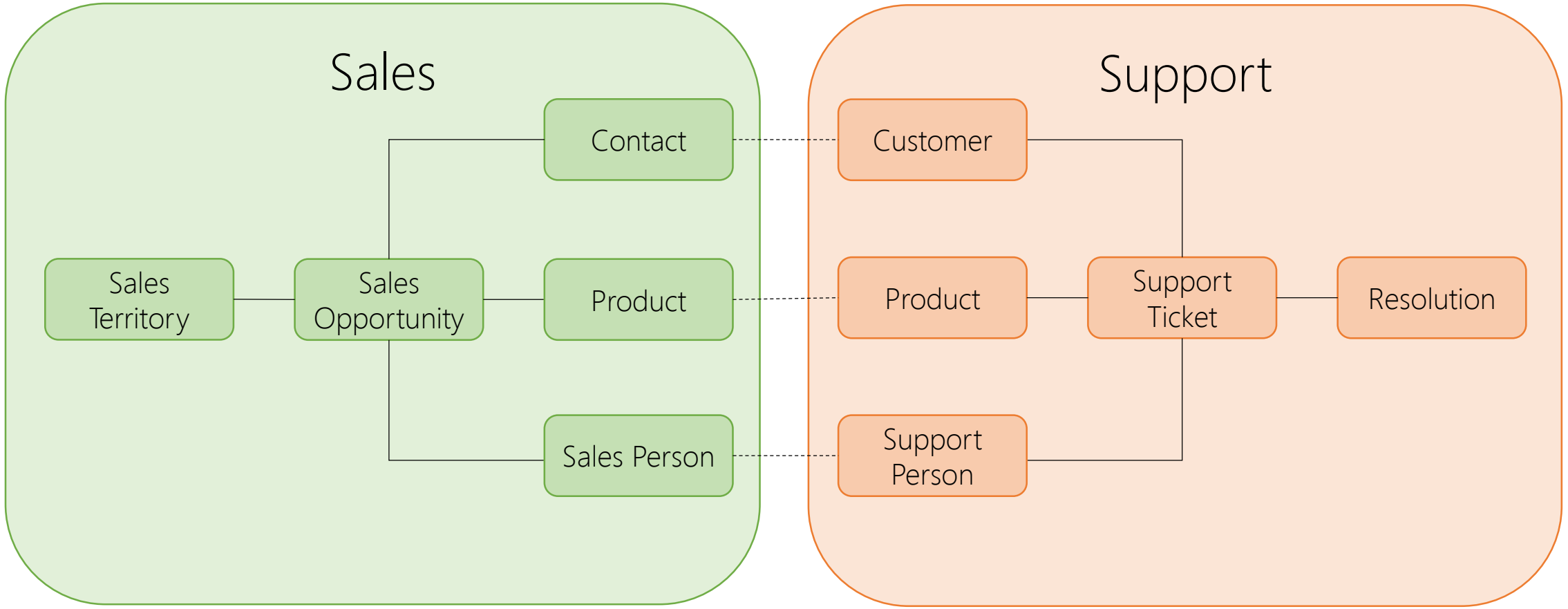
Single Domain Model



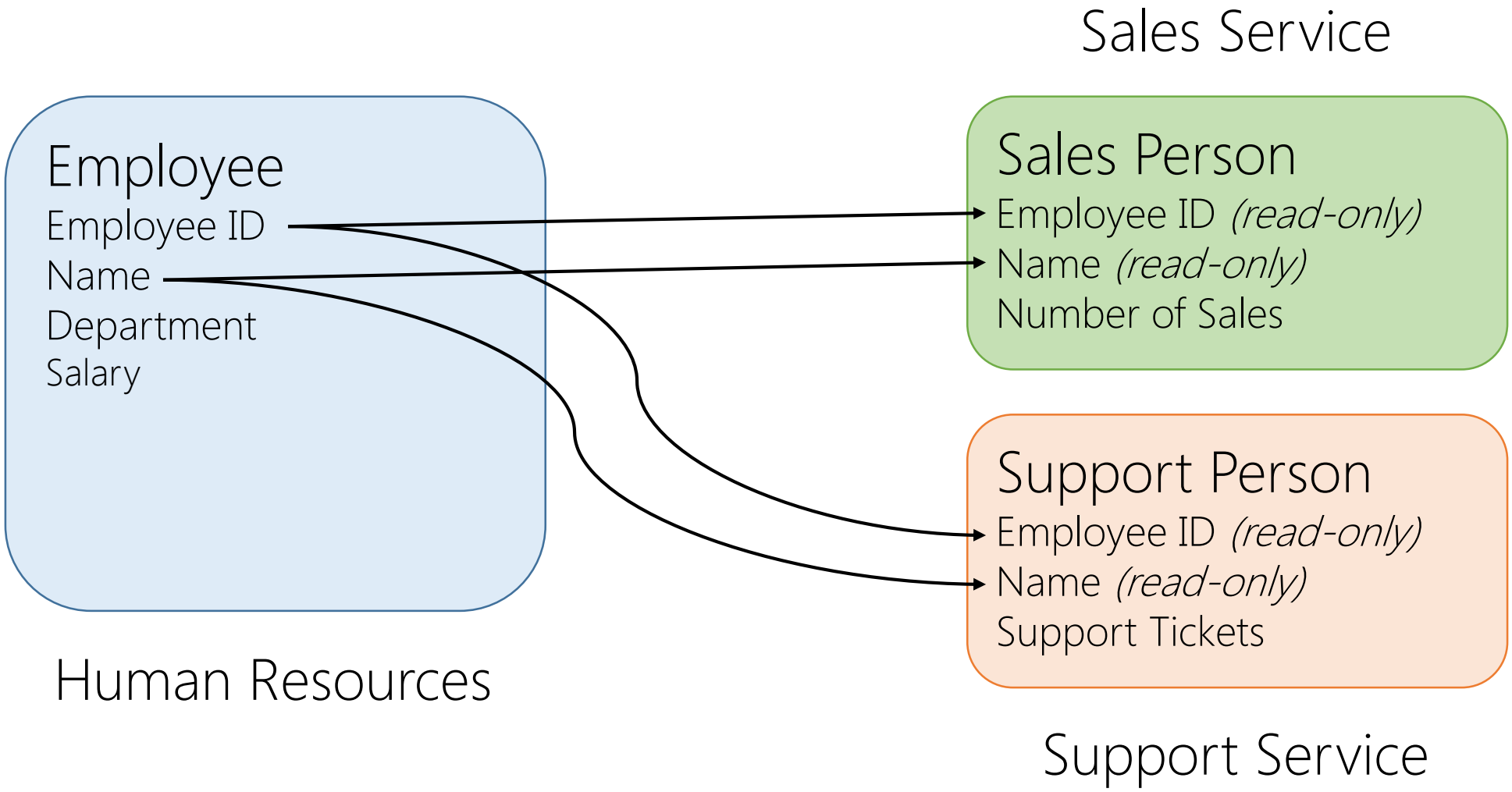
Bounded Contexts



Bounded Contexts

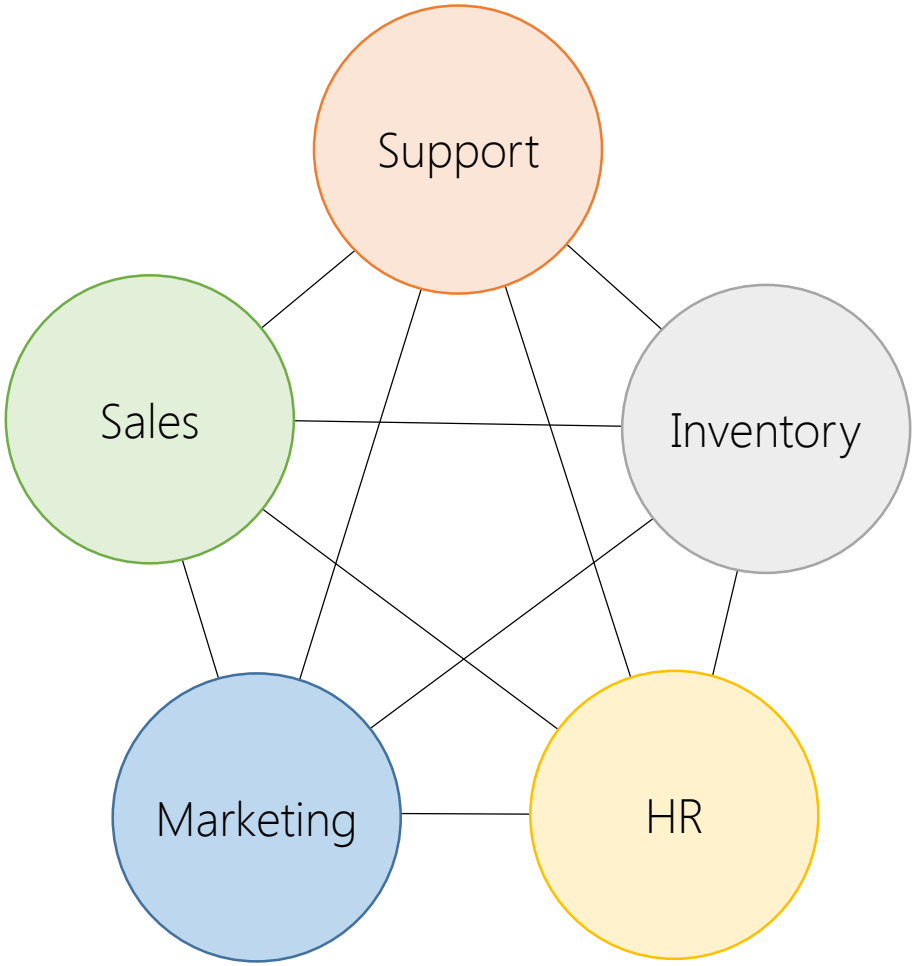


Bounded Contexts



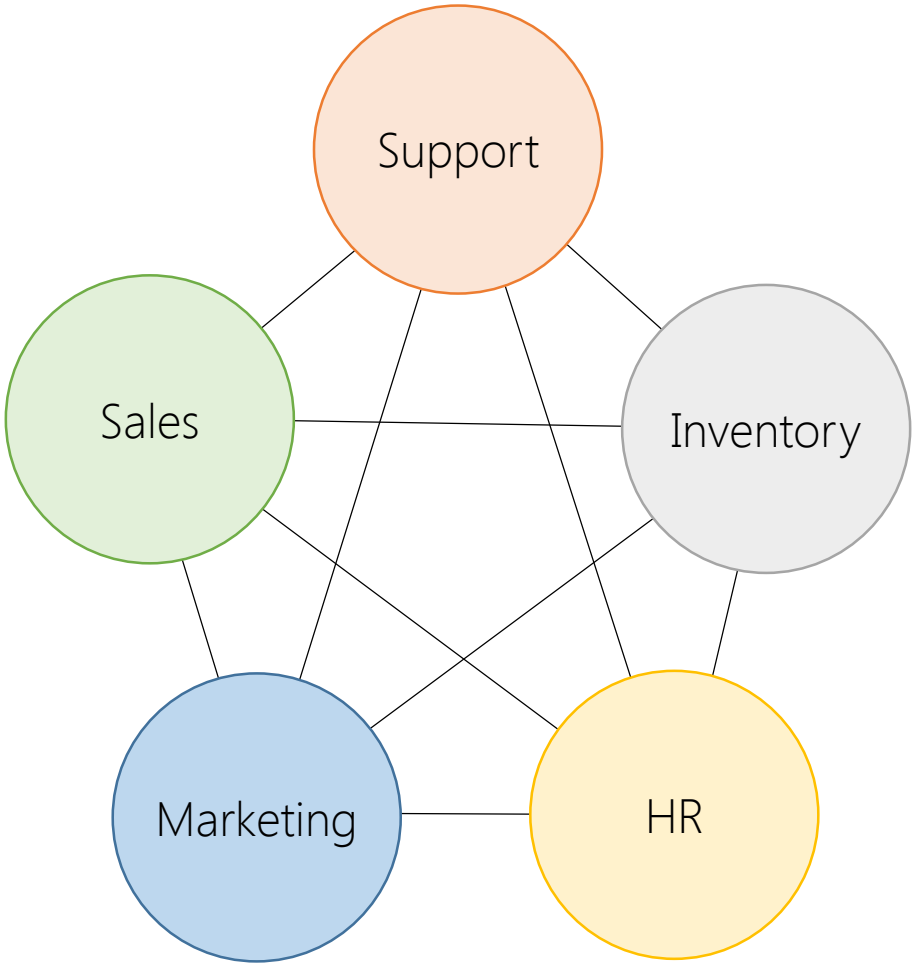
Microservice Architectures

- Subdivide system
- Bounded contexts
- Small teams



Microservice Architectures

- Independent
- Similar to SOA



Why Use Microservices?

Pros

- Less cost for large domains
- Smaller teams
- Independence

Why Use Microservices?

Pros

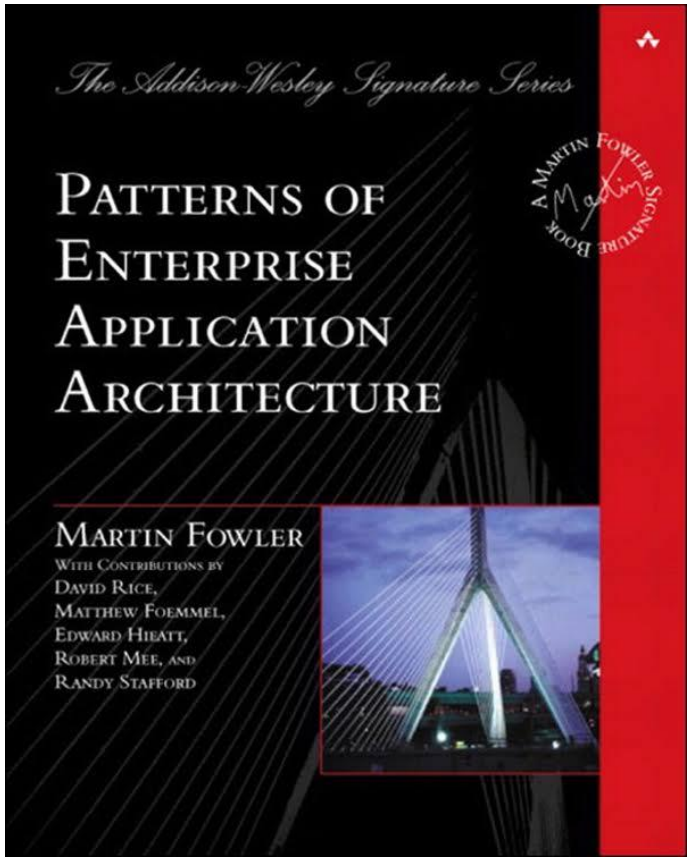
- Less cost for large domains
- Smaller teams
- Independence

Cons

- Only for large domains
- Higher up-front cost
- Distributed system costs

Code Demo

Where to Go Next?



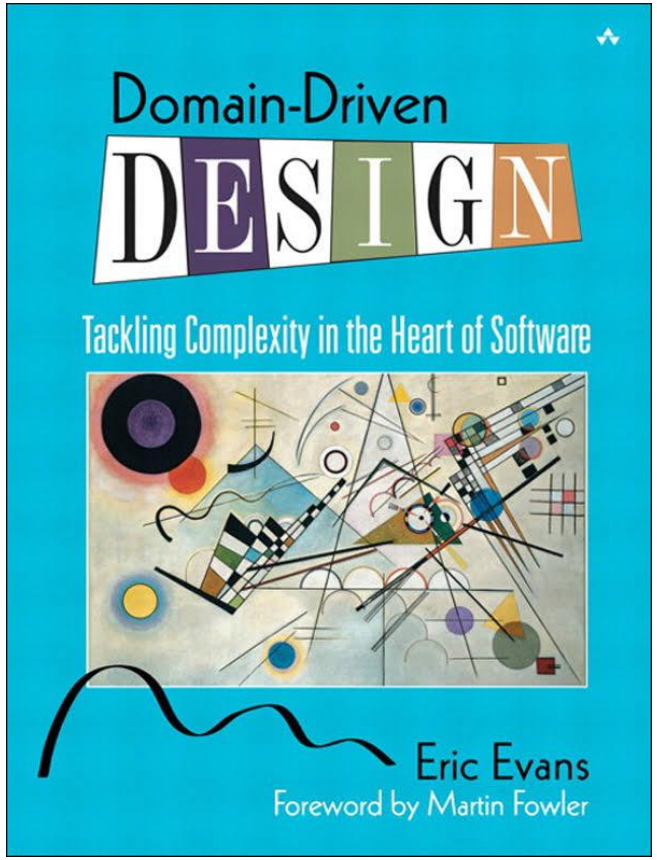
Martin Fowler

Where to Go Next?

http://cleancoders.com/

Robert C. Martin

Where to Go Next?



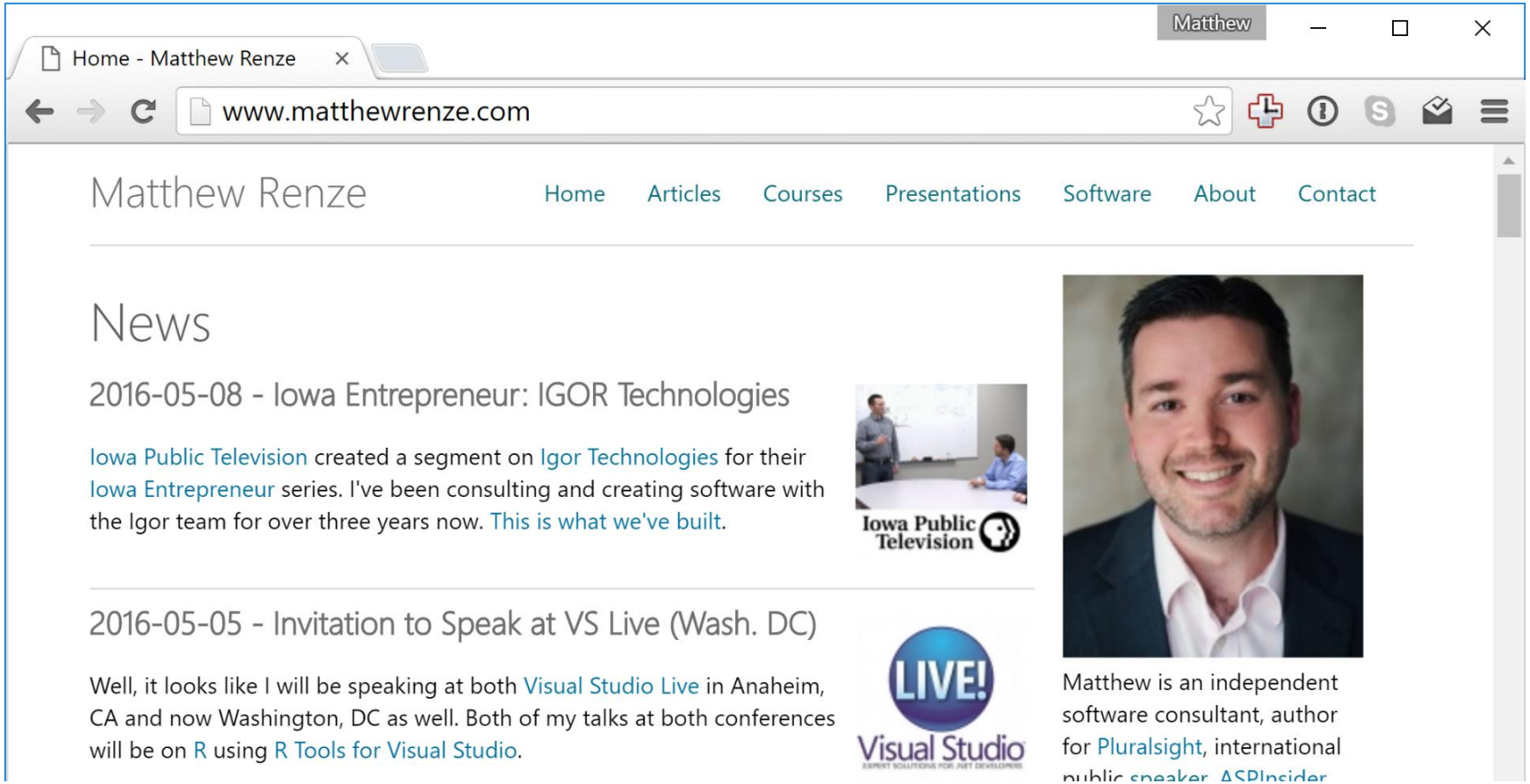
Eric Evans

Where to Go Next?

Greg Young

Udi Dahan

Where to Go Next?



www.matthewrenze.com

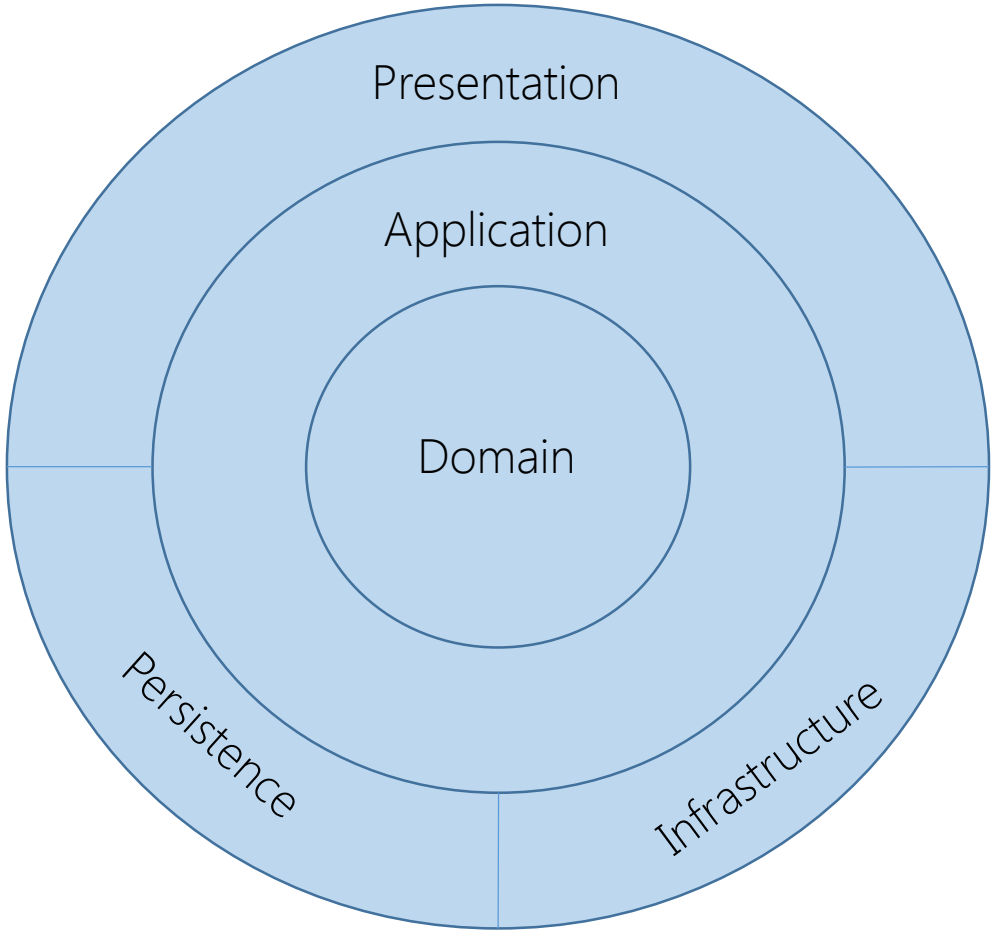
Conclusion

Summary

- Focus on the inhabitants

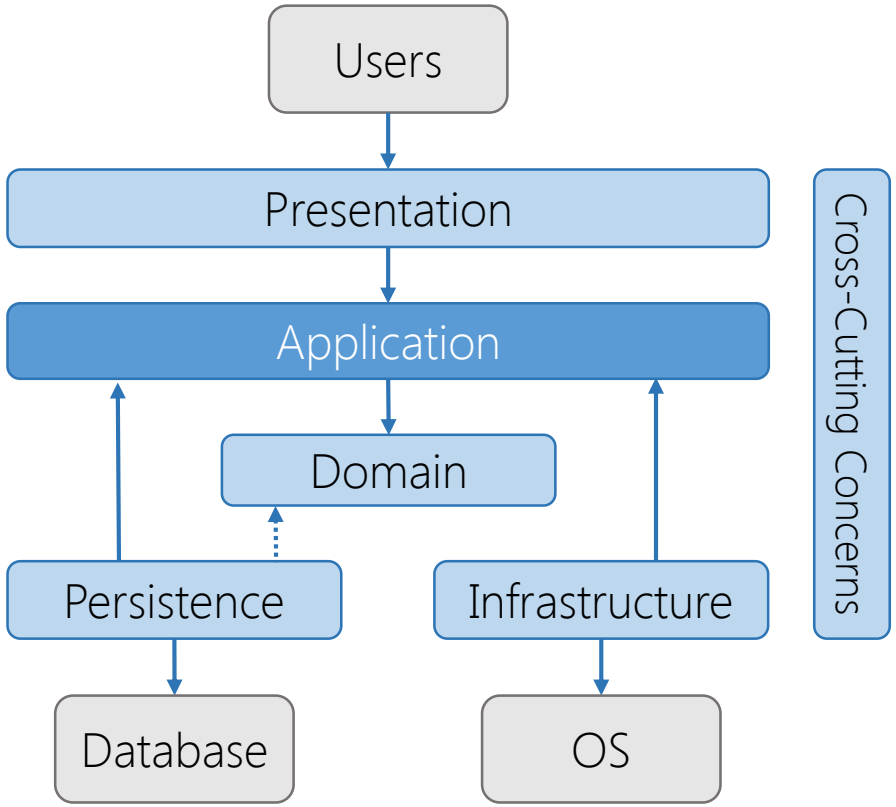
Summary

- Focus on the inhabitants
- Domain-centric Architecture



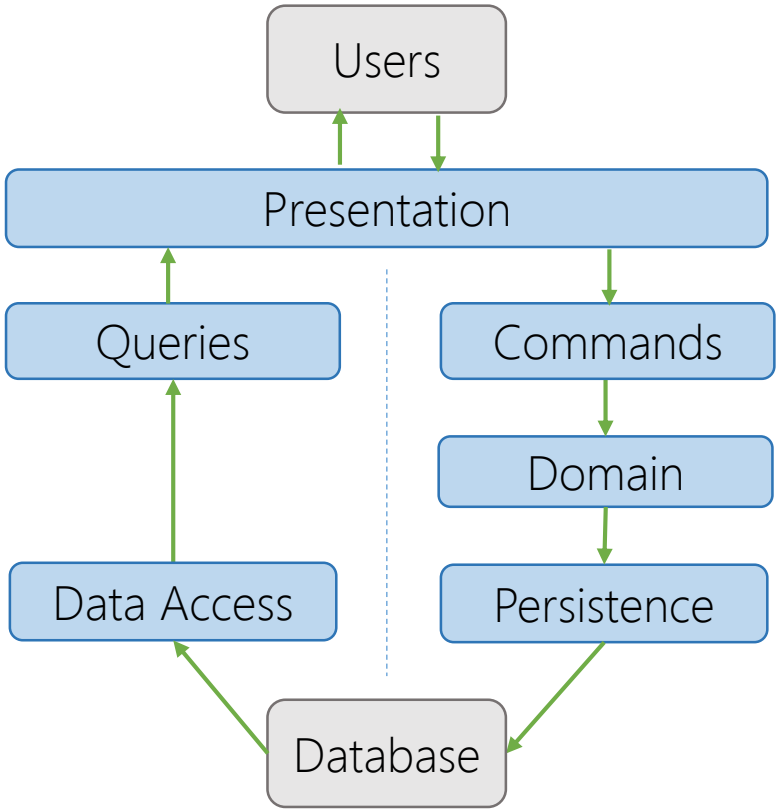
Summary

- Focus on the inhabitants
- Domain-centric Architecture
- Application Layer



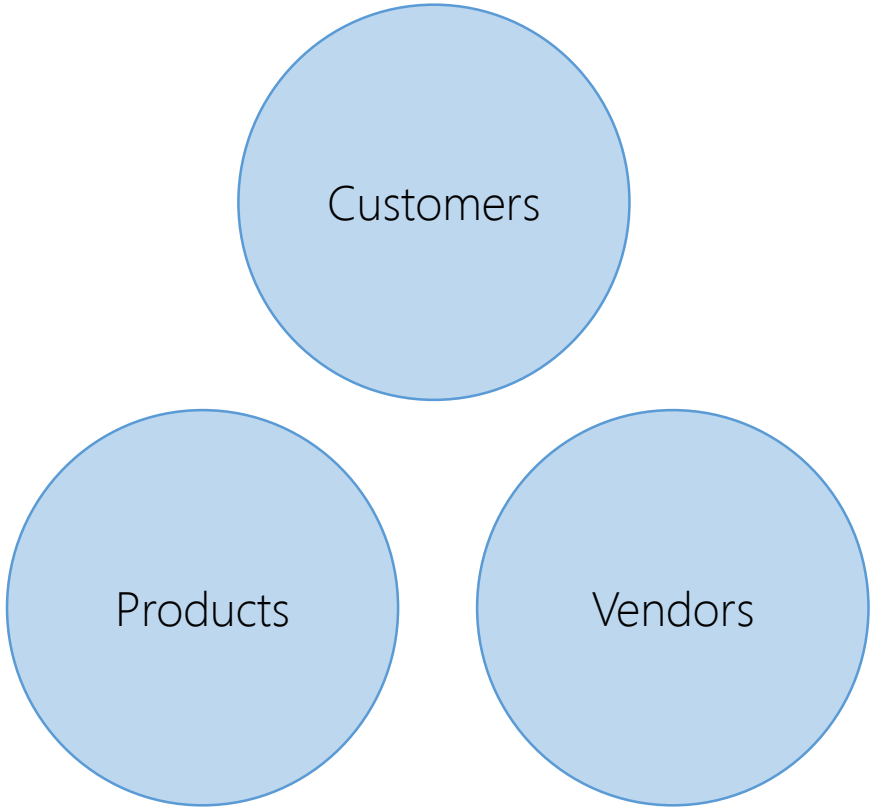
Summary

- Focus on the inhabitants
- Domain-centric Architecture
- Application Layer
- Commands and Queries



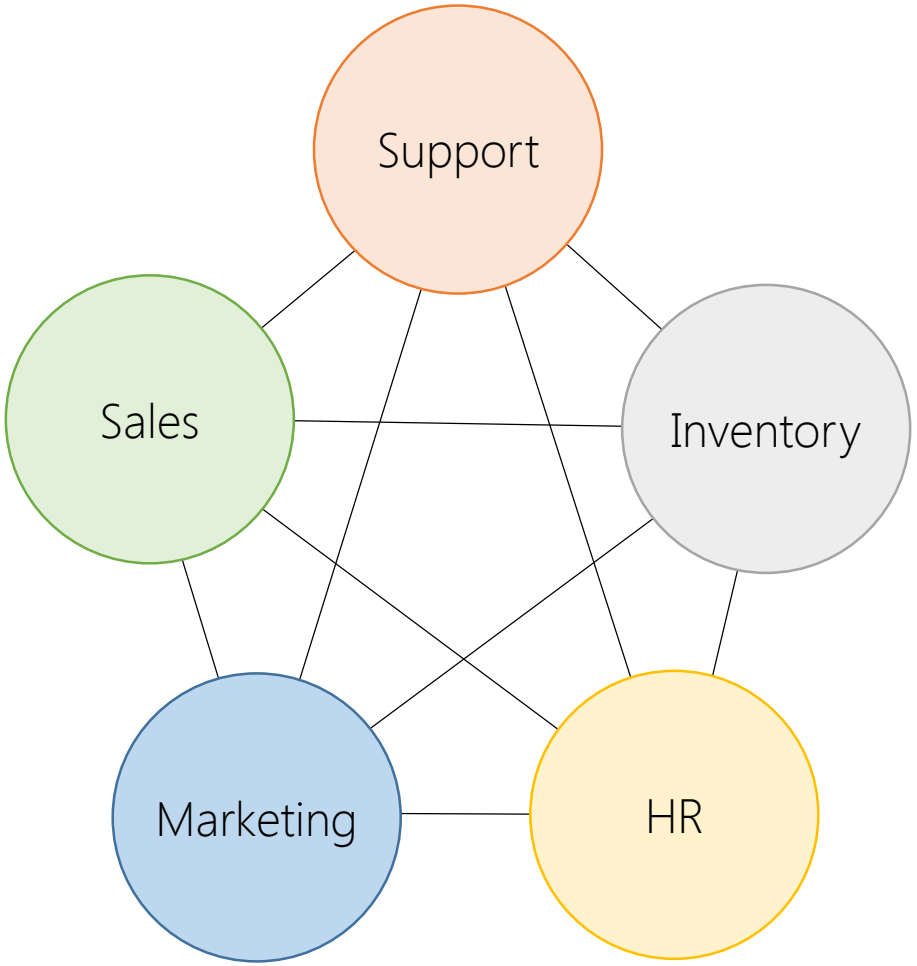
Summary

- Focus on the inhabitants
- Domain-centric Architecture
- Application Layer
- Commands and Queries
- Functional Cohesion



Summary

- Focus on the inhabitants
- Domain-centric Architecture
- Application Layer
- Commands and Queries
- Functional Cohesion
- Bounded Contexts



Feedback

- Feedback is very important to me!
- One thing you liked?
- One thing I could improve?

Contact Info

Matthew Renze

- Twitter: @matthewrenze
- Email: matthew@renzeconsulting.com
- Website: www.matthewrenze.com

Thank You! :)