LINKING ARCHITECTURE PATTERNS AND DDD

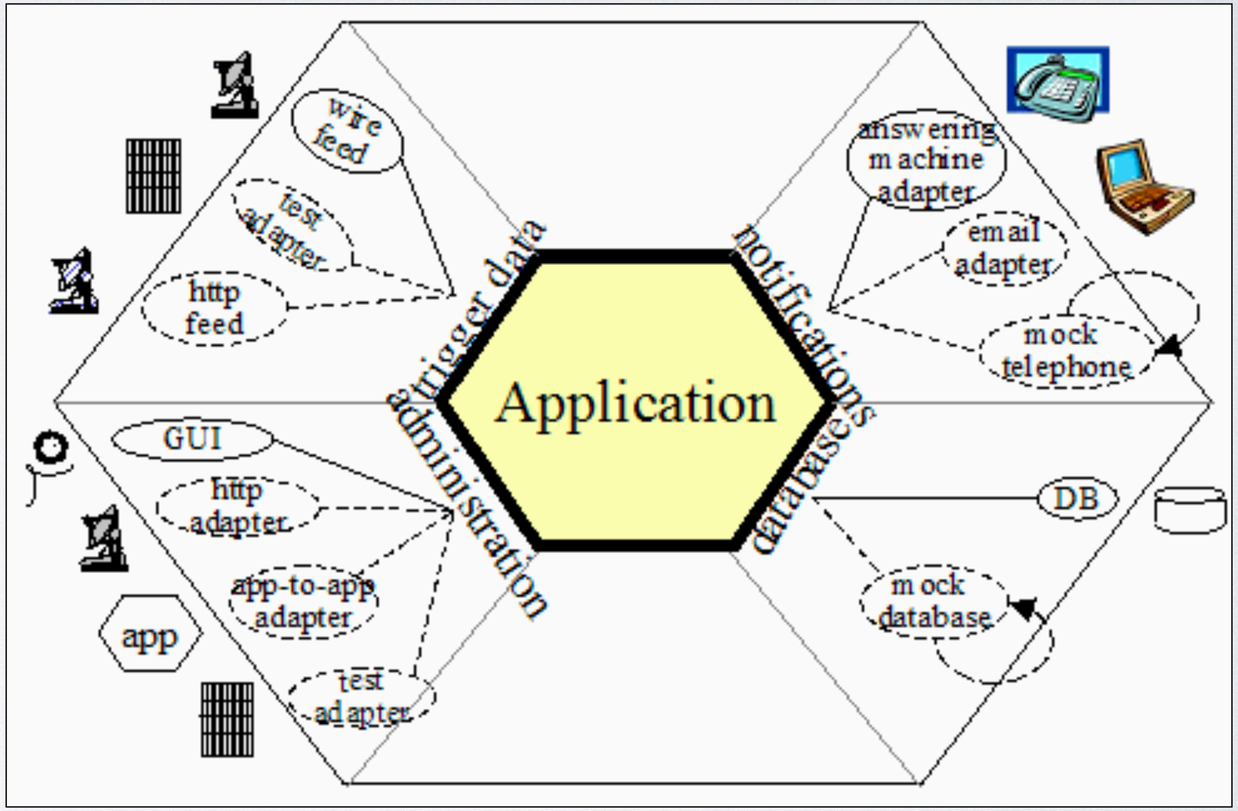
How Clean Architecture, Onion Architecture, Hexagonal Architecture and Domain-Driven Design play together

AGENDA

- Hexagonal Architecture
- Onion Architecture
- Clean Architecture
- Domain-Driven Design
- Domain-Driven Design Concepts
- Combining DDD and Architecture Patterns
- Final Thoughts

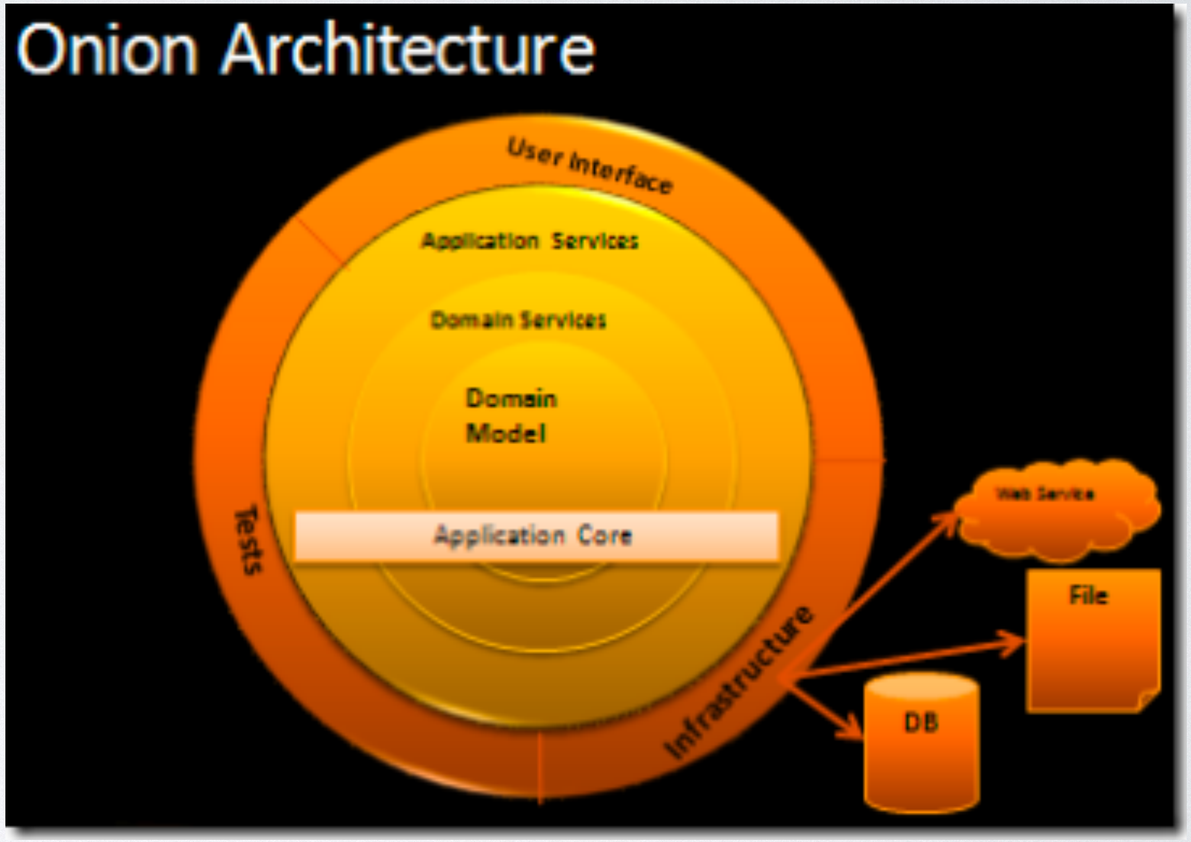
HEXAGONAL ARCHITECTURE

- By Alistair Cockburn in 2005
- Aka Ports and Adapters
- Dependencies flow inwards



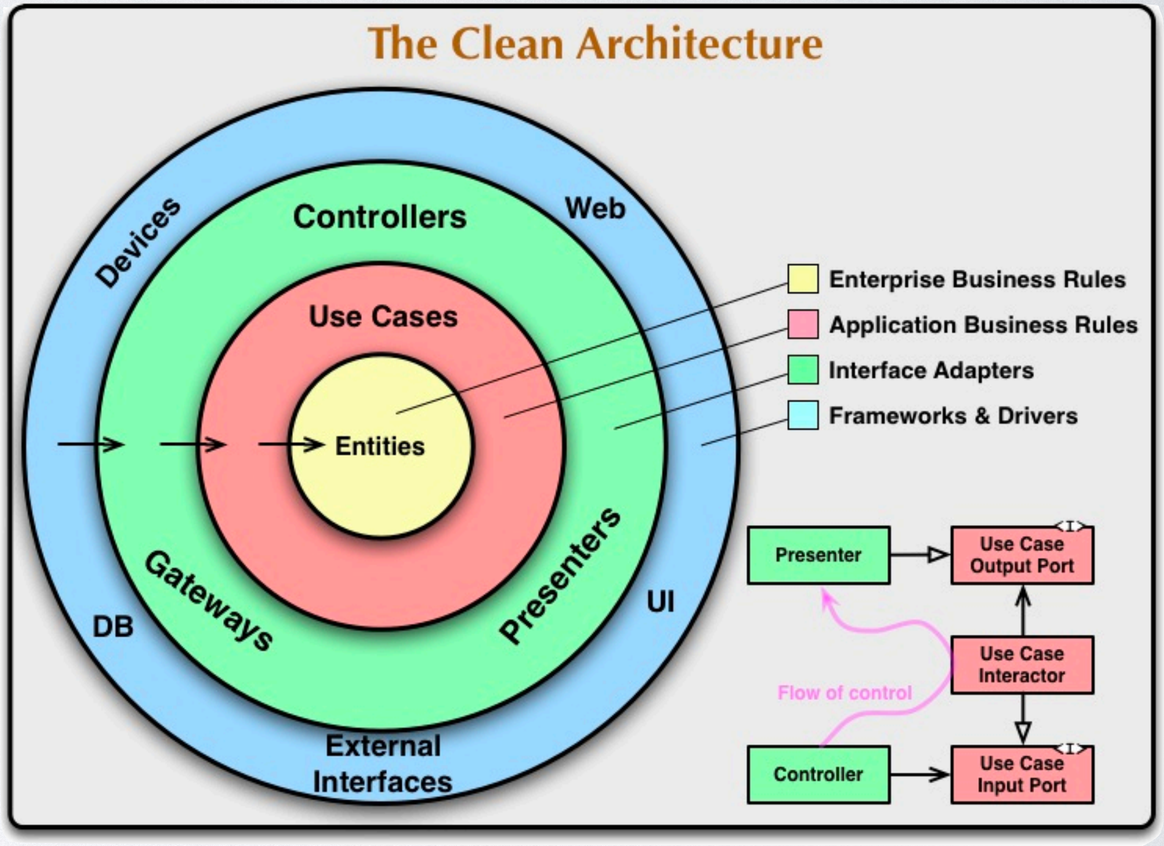
ONION ARCHITECTURE

- By Jeffrey Palermo in 2008
- Concentric Layers
- Relies on Interfaces
- Dependencies flow inwards



CLEAN ARCHITECTURE

- By Robert C. Martin in 2012
- Interface Adapters
- Dependencies only point inwards



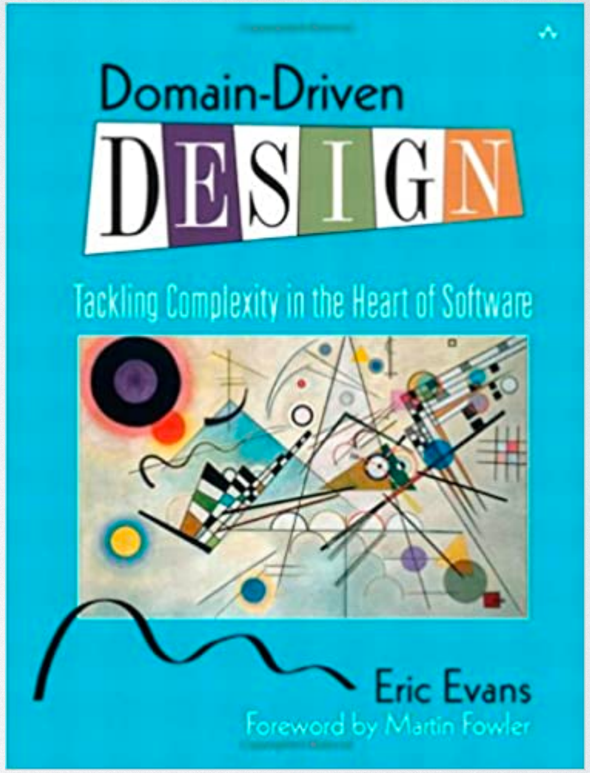
SIMILARITIES & DIFFERENCES

- Dependency Rule
- Separation of Concerns
- Testability
- Independence from Frameworks, Databases, UI
- Modularity

- Mainly in Terminology
- Slightly different layer organization
- Slightly different focus

DOMAIN-DRIVEN DESIGN

- By Eric Evans in 2003
- Modeling software to match a domain according to input from that domain's experts
- Concepts: Ubiquitous Language, Bounded Contexts and many more



DDD-CONCEPTS

Entities:
Objects defined by there identity, rather than there attributes
(Customer, Order, Product)

Value Objects:
Immutable objects that contains attributes but has no conceptual identity
(Money, Date, Address)

Aggregates:
Cohesive clusters of domain objects that can be treated as a single unit.
(Order-Aggregate containing Order-Entity, OrderLines, Payments and Shipping Address)

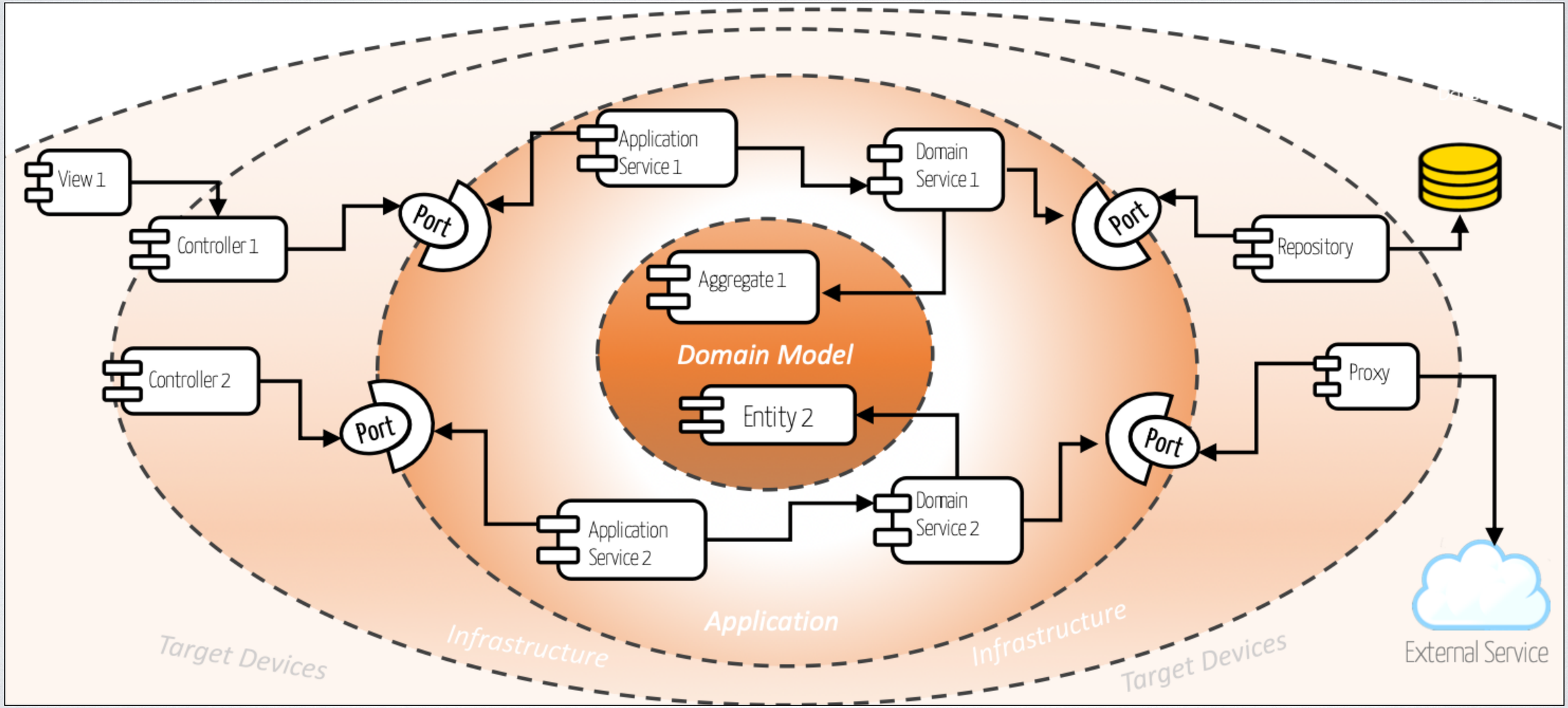
Aggregate Roots:
Entry point for updating or retrieving the aggregate's data.
(Order-Entity)

DDD-CONCEPTS

Domain Services:
Contain domain logic that doesn't naturally fit within an entity or value object
(TaxCalculator)

Repositories:
Encapsulate logic required to access domain objects from a data source.
Provide Collection-Like Interface
(OrderRepository)

ARCHITECTURE PATTERN AND DDD



FINAL THOUGHTS

- Separating the domain code from application and infrastructure code
- Architecture patterns aim directly for an isolation of the „core“ logic
- With DDD the domain separation it is more of a side effect
- Complementary approaches

THANK YOU FOR YOUR ATTENTION!

QUESTIONS?

- Sources:
  - https://alistair.cockburn.us/hexagonal-architecture/
  - https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/
  - https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
  - https://martinfowler.com/tags/domain%20driven%20design.html
  - ALCOR Foundation Training Slides

patrick.ronecker@css.ch