Image from https://domainlanguage.com

- Entities
- Value objects
- Aggregates
- Repositories
- Bounded contexts
- Domain events

Domain-Driven Design in Practice

Introduction

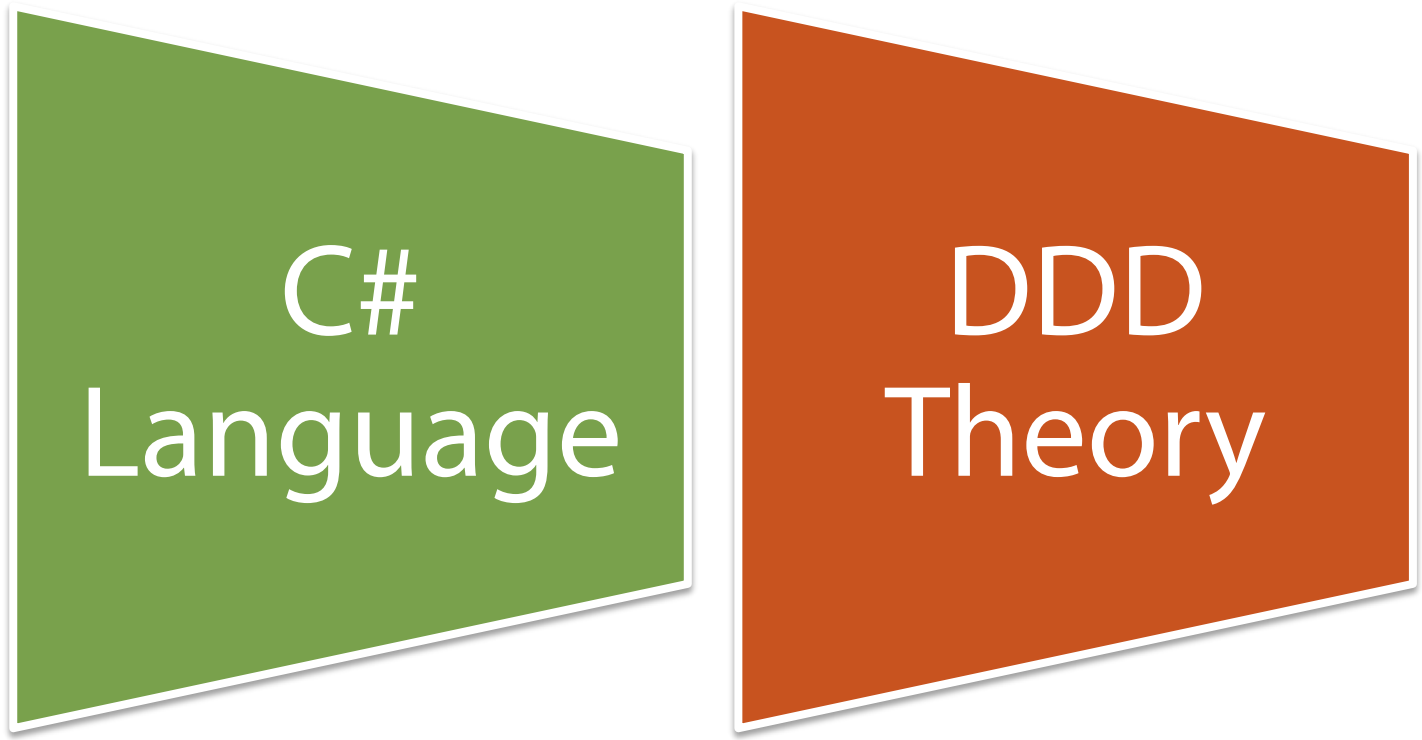
Vladimir Khorikov

@vkhorikov | www.enterprisecraftsmanship.com

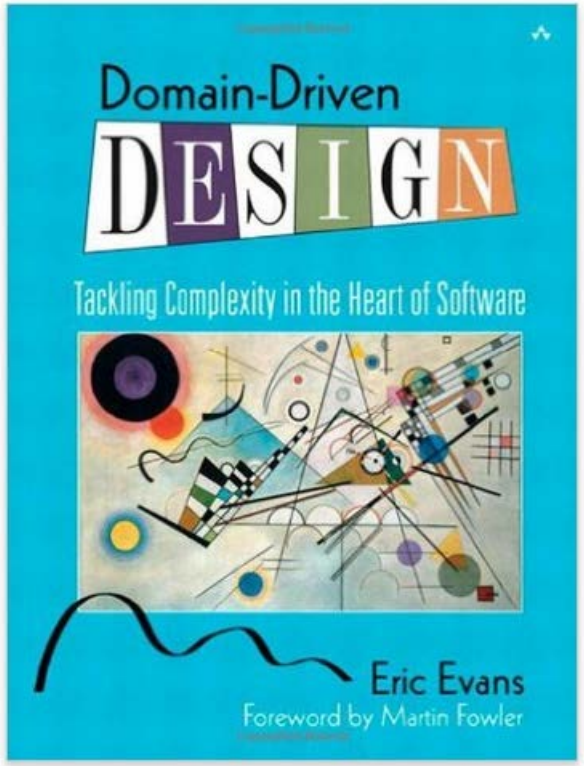
Course Outline

- Introduction
- Starting with the First Bounded Context
- Introducing UI and Persistence Layers
- Extending the Bounded Context with Aggregates
- Introducing Repositories
- Introducing the Second Bounded Context
- Working with Domain Events
- Looking Forward to Further Enhancements

Prerequisites



Prerequisites



Domain-Driven Design: Tackling Complexity in the Heart of Software

By Eric Evans

Domain-Driven Design Fundamentals

by Steve Smith and Julie Lerman

This course teaches the fundamentals of Domain-Driven Design (DDD) through a demonstration of customer interactions and a complex demo application, along with advice from Eric Evans.

▶ Resume Course

Table of contents | Description | Transcript | Exercise files | Discussion | Learning Check | Recommended

Expand all

Area of Application for Domain-Driven Design

Software Project

- Amount of data
- Performance
- Business logic complexity
- Technical complexity

Area of Application for Domain-Driven Design

Software Project

- Amount of data
- Performance
- Business logic complexity
- Technical complexity

Area of Application for Domain-Driven Design

Twitter

DDD

Business logic complexity

Amount of data

Performance

Area of Application for Domain-Driven Design

DDD

Performance

Amount of data

Technical complexity

Area of Application for Domain-Driven Design

DDD

Business logic complexity

Why Domain-Driven Design?

YAGNI

- You are not gonna need it
- Shortening development time

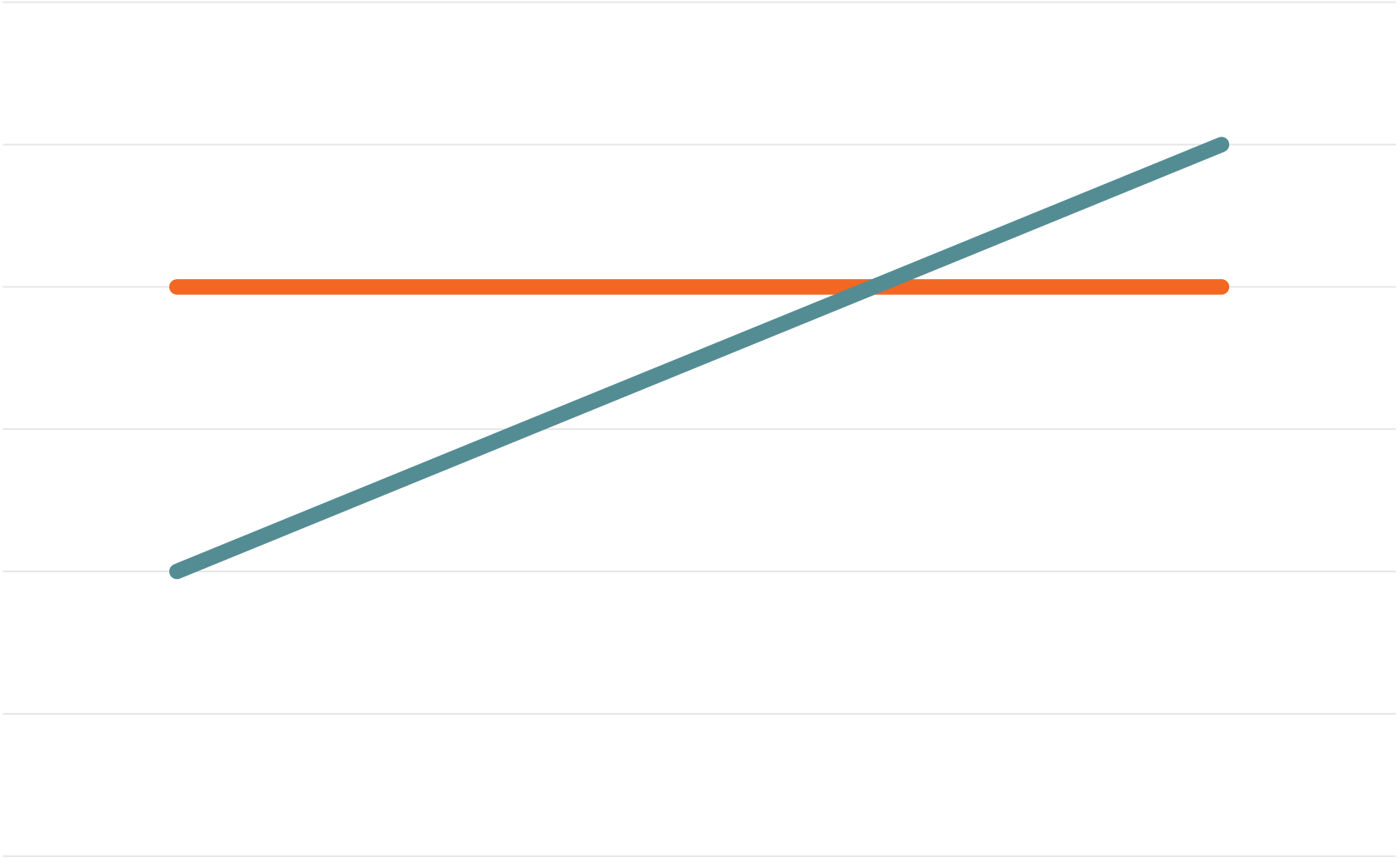
KISS

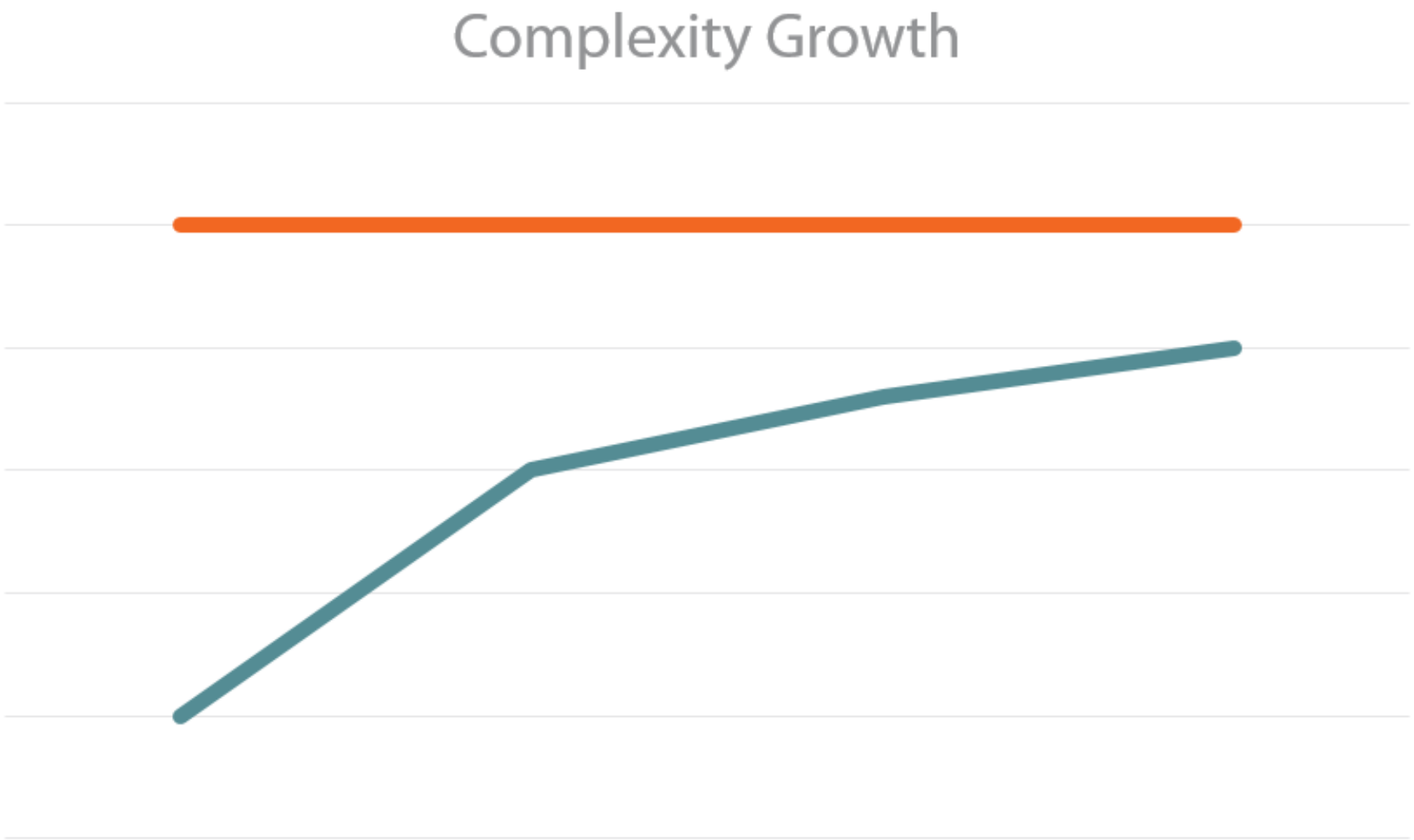
- Keep it short and simple
- Maintainable code

DDD

- Focus on essential parts
- Simplifying the problem

Complexity Growth

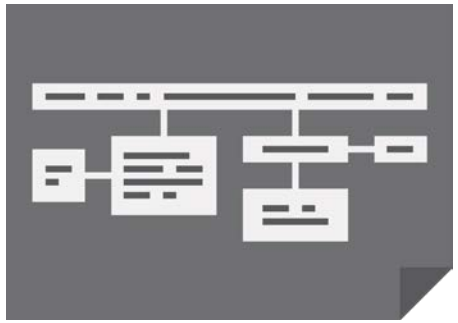




Main Concepts of Domain-driven Design

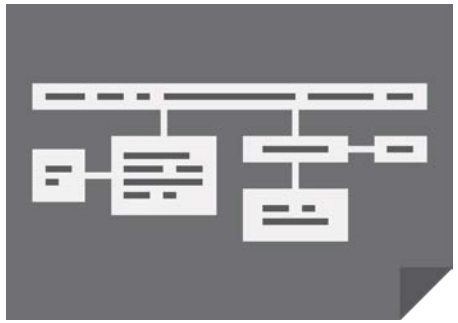
Ubiquitous language

- Bridges the gap between developers and experts



Product

Product Package



Product

Product Package

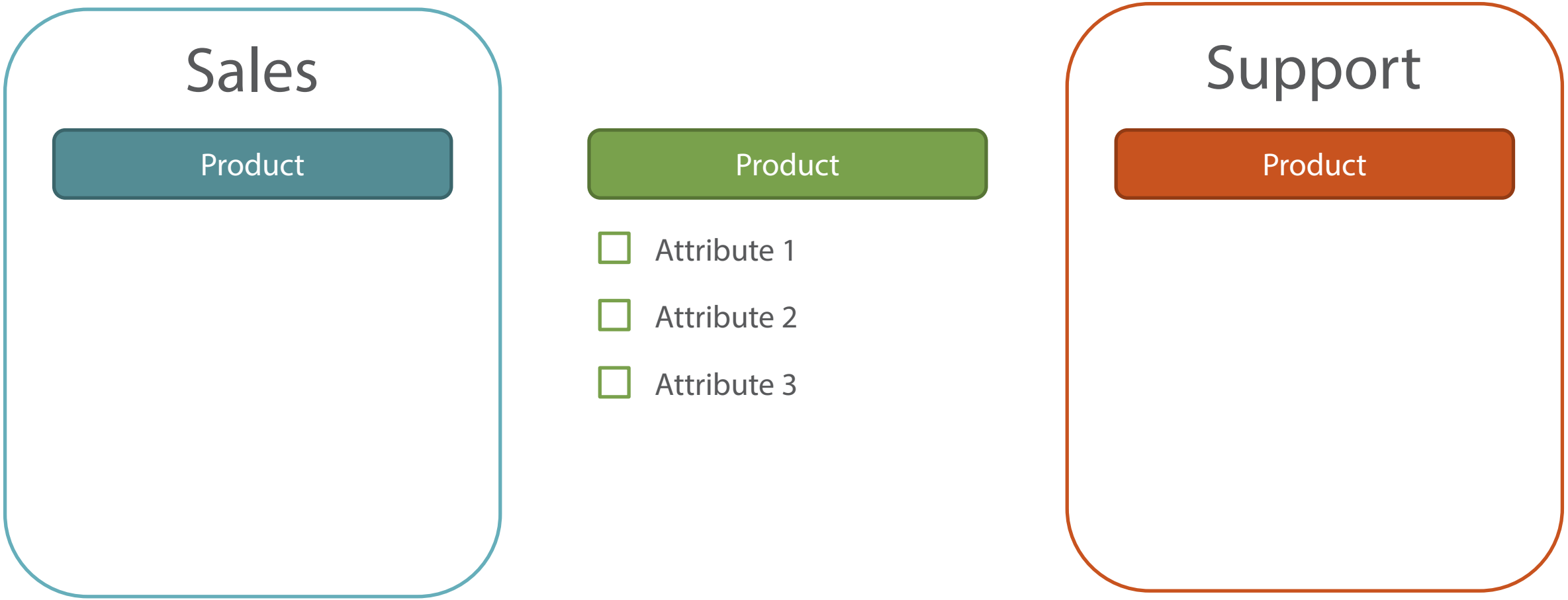
Main Concepts of Domain-driven Design

Ubiquitous language

- Bridges the gap between developers and experts

Bounded context

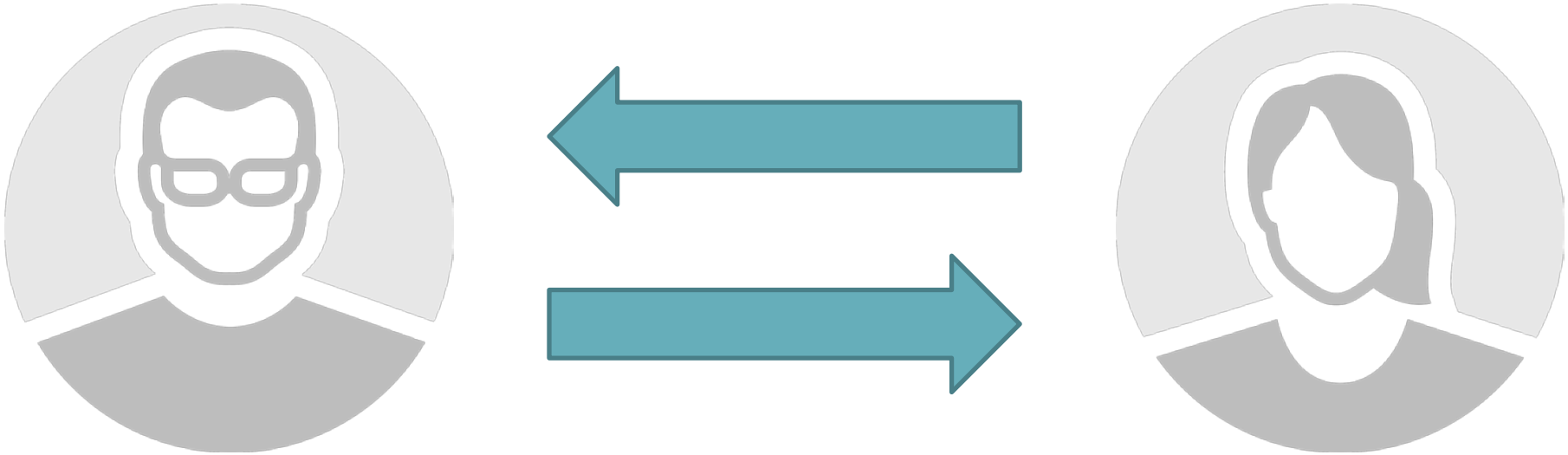
- Clear boundaries between different parts of the system



Main Concepts of Domain-driven Design

Ubiquitous language	• Bridges the gap between developers and experts
Bounded context	• Clear boundaries between different parts of the system
Core domain	• Focus on the most important part of the system

DDD Is Not Only About Writing Code



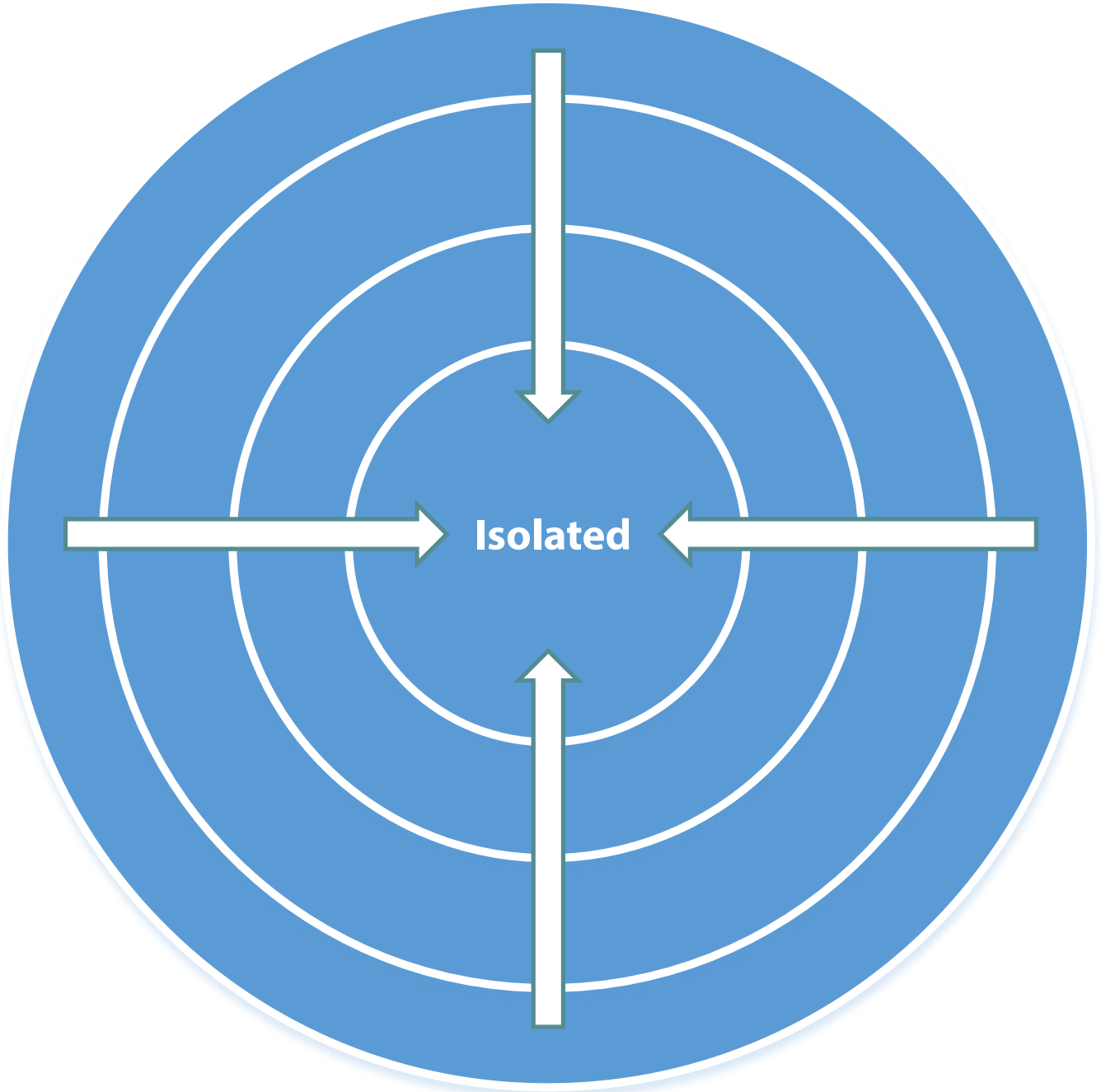
DDD Is Not Only About Writing Code

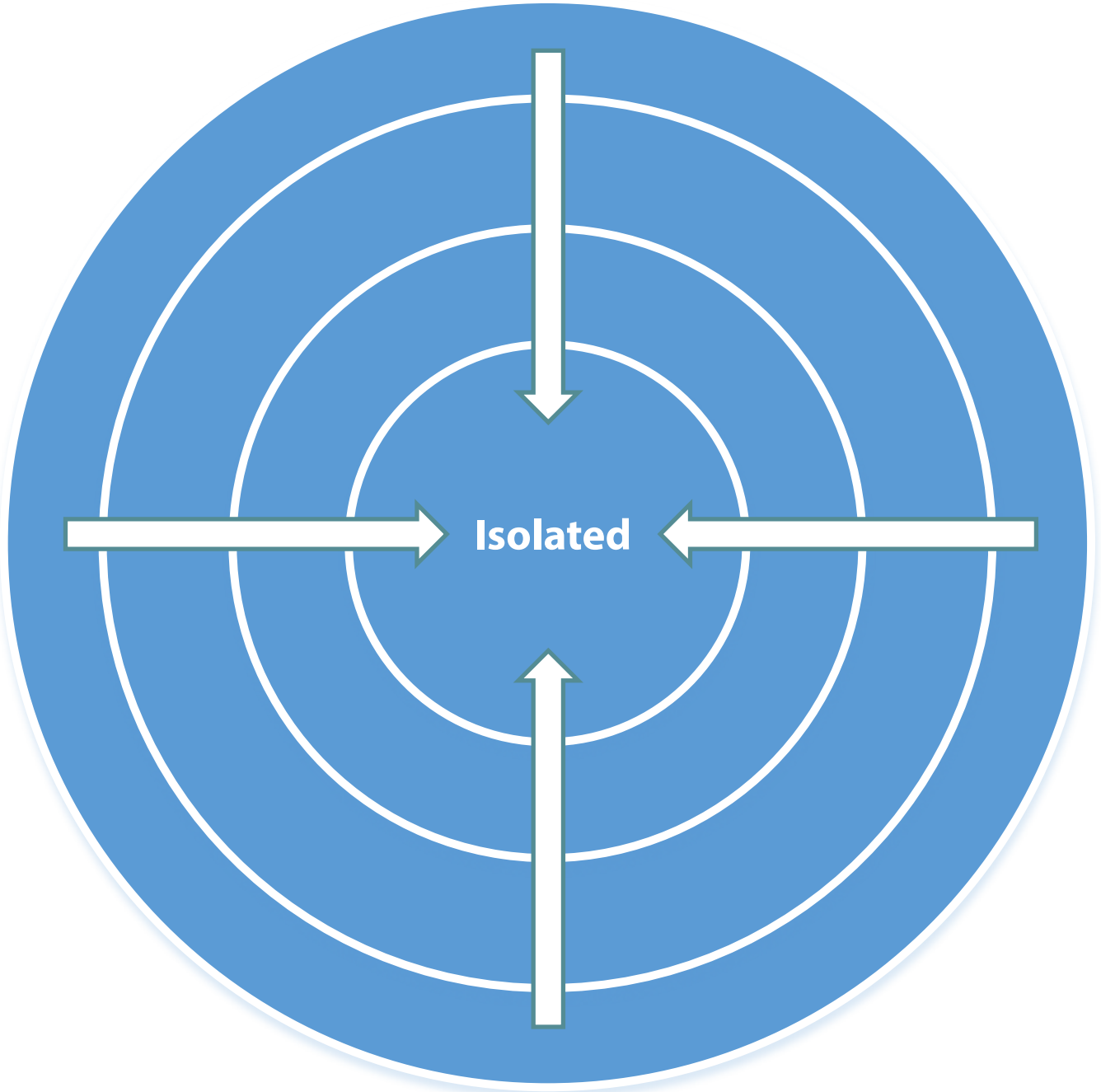
Developer

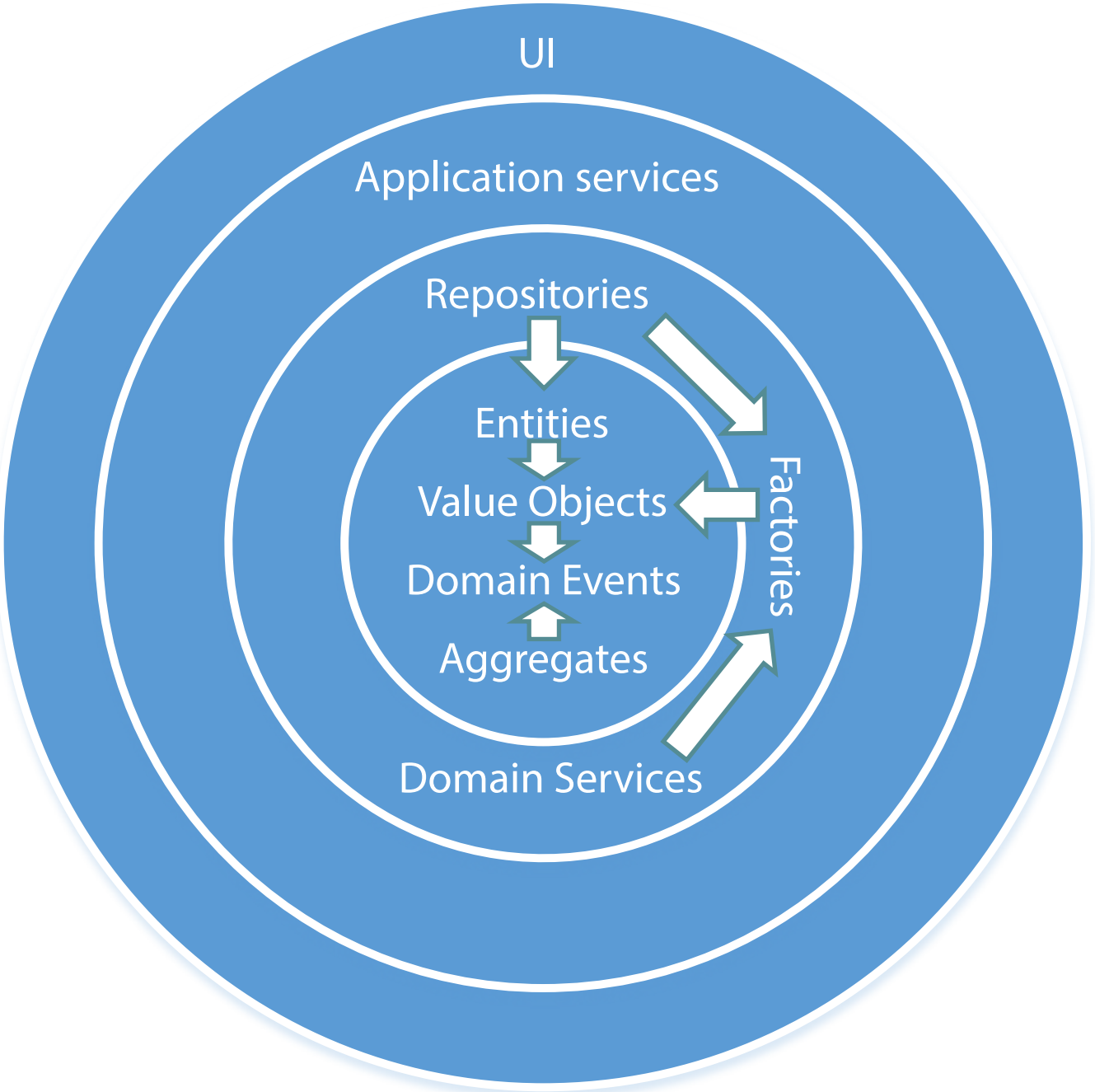
- [ ] Technical challenge
- [ ] Technical knowledge reuse

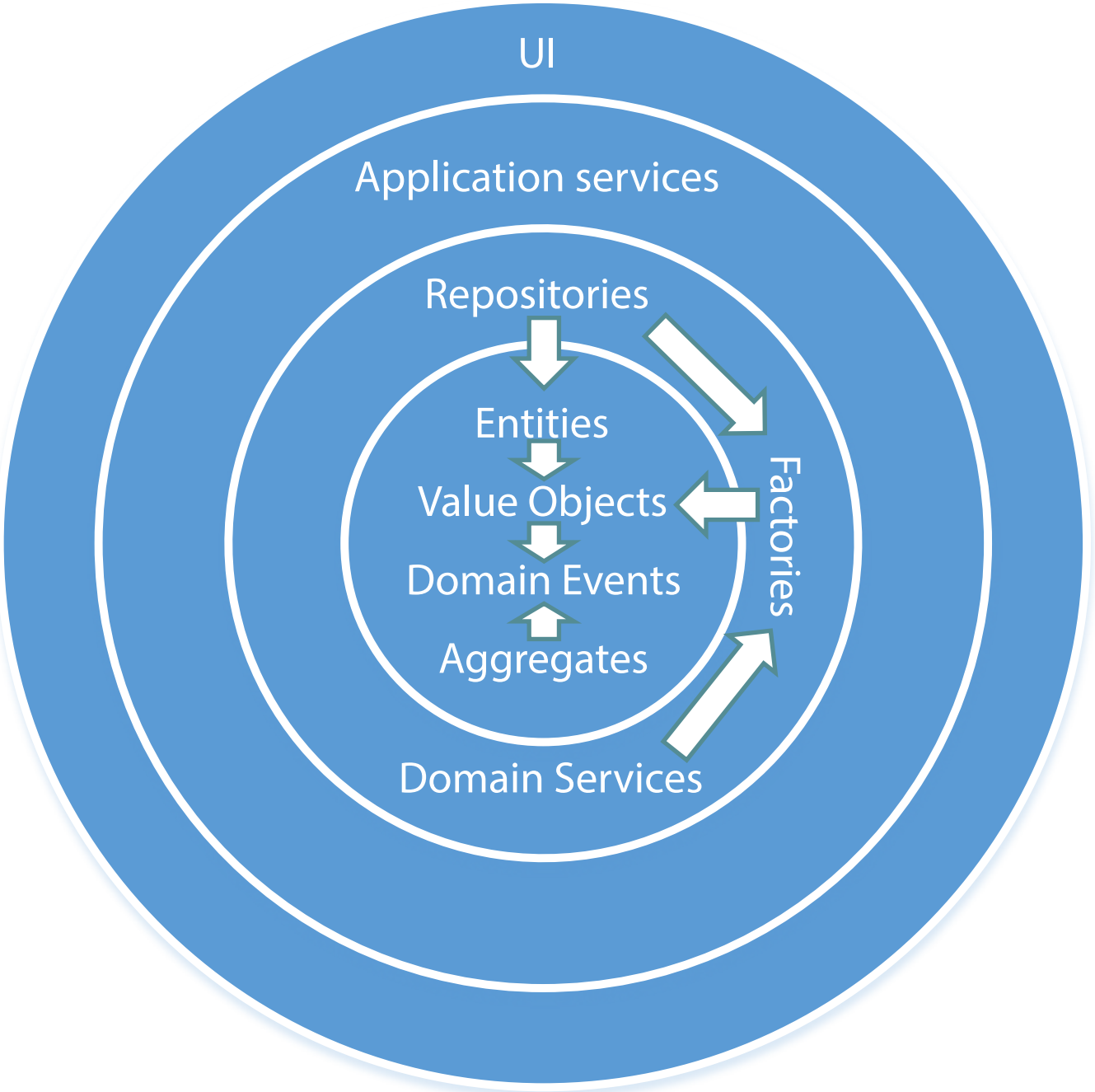
Domain Expert

- [ ] Domain experts' view point
- [ ] The skill of systematizing problem domains









Isolation

Entity | Domain Event | Value Object | Aggregate

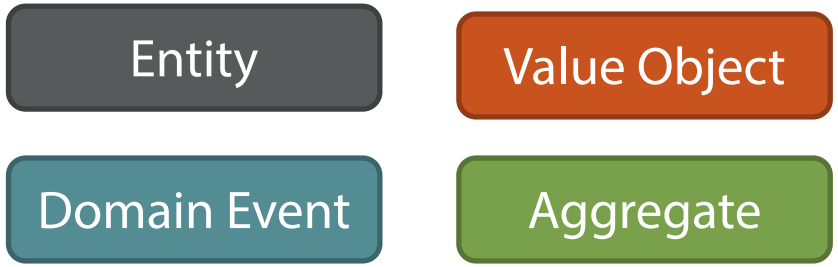
- Domain knowledge 
- Persistence 
- Construction 
- Mapping to the database 

```
public class Product
{
    public string Name { get; }

    protected Product()
    {
    }

    public Product(string name)
    {
        Name = name;
    }
}
```

Isolation

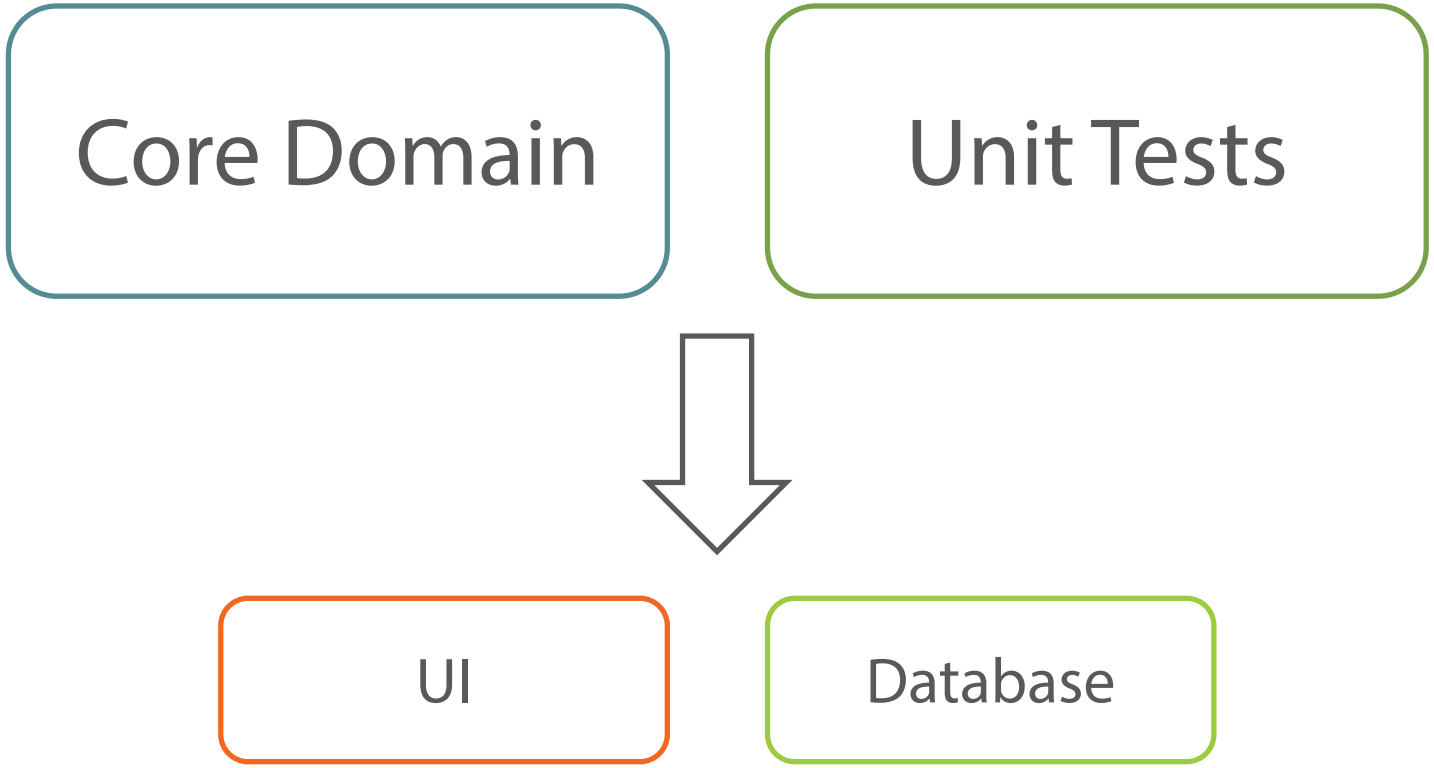


- [ ] Clean domain model
- [ ] Proper separation of concerns
- [ ] Dealing with ORM side effects

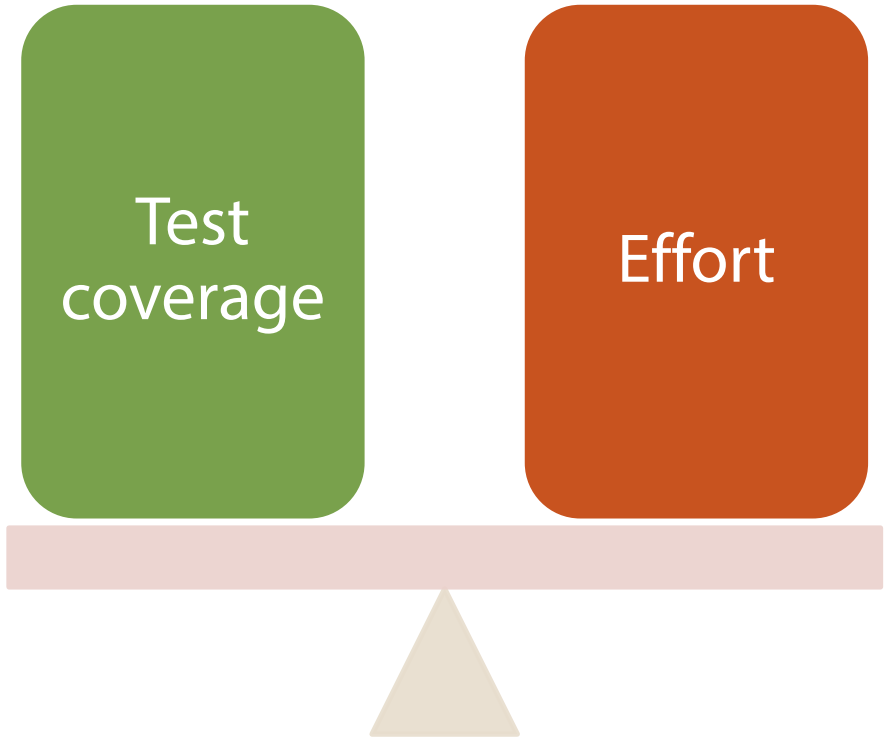
Modeling Best Practices

Focus on the Core Domain

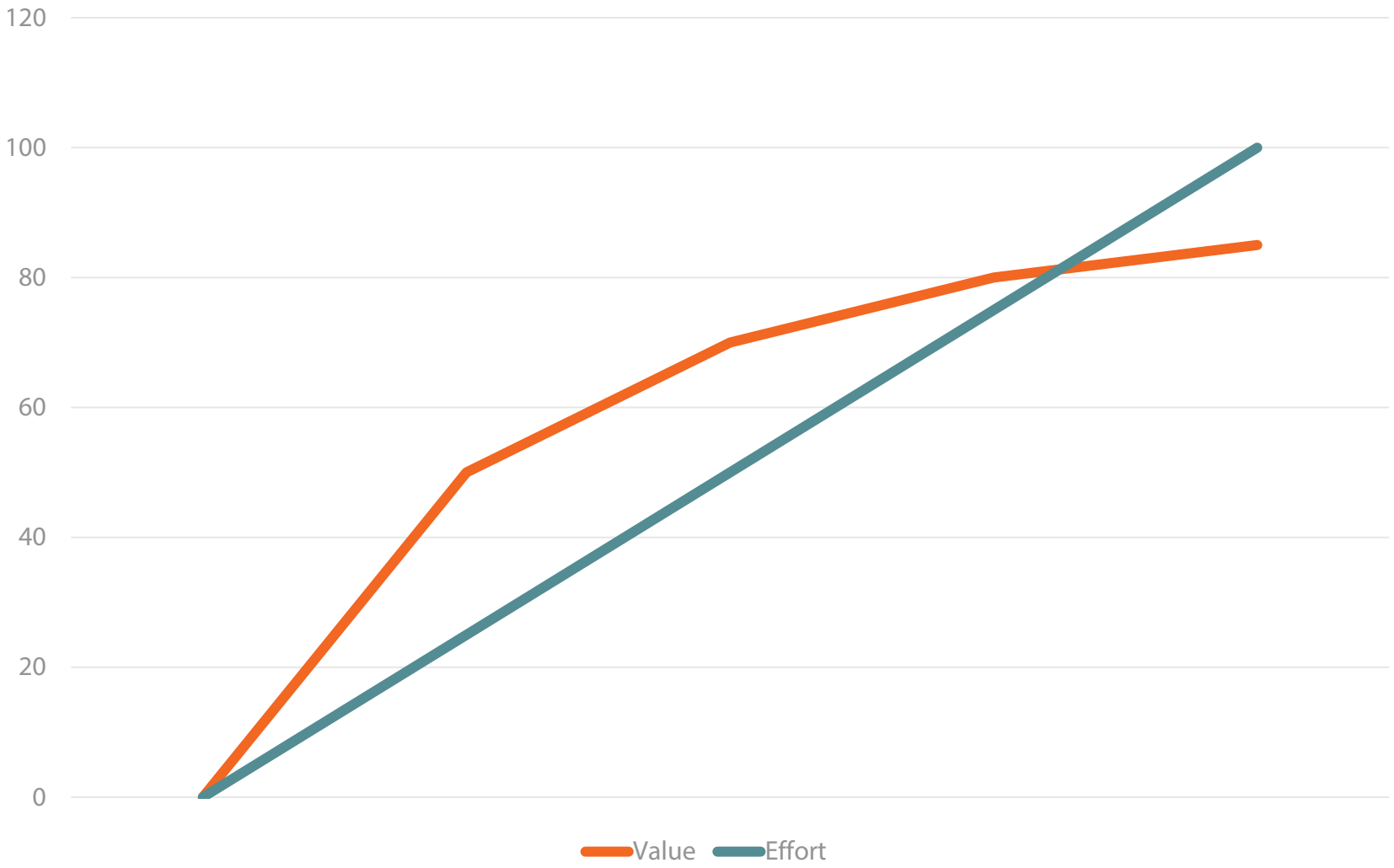
Modeling Best Practices

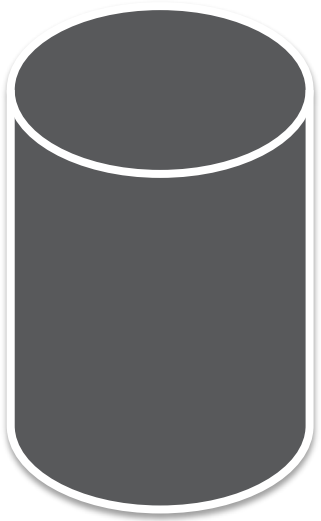


Domain-Driven Design and Unit Testing

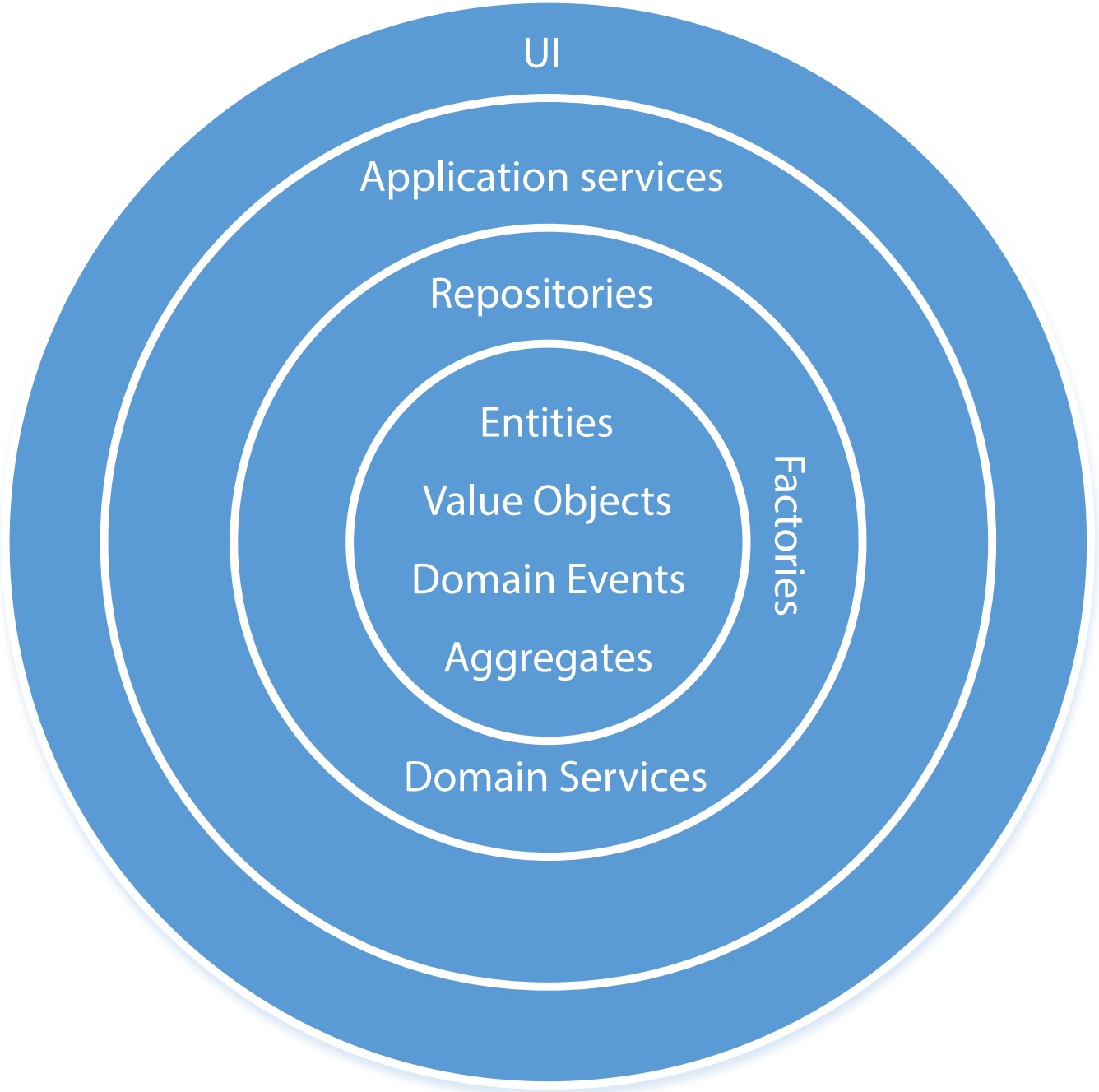


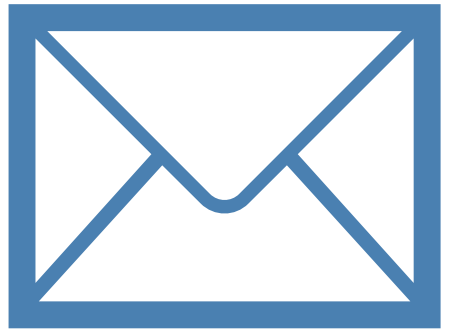
Test Coverage vs. Value Distribution





Database





Email service

Domain-Driven Design and Unit Testing

Learn more at
http://bit.ly/1hT842g

The Problem Domain Introduction

The Problem Domain Introduction

Low-level implementation details

Business logic

Summary

- DDD area of application
- Core software design principles: YAGNI and KISS
- Main DDD concepts: ubiquitous language, bounded context and core domain
- DDD is not only about writing code
- Onion architecture and domain model isolation
- DDD and unit testing

Summary

Learn by doing