Domain Driven Design

Patrícia Oliveira

FARFETCH

AGENDA

1. About me
2. Domain Driven Design
   a. What is?
   b. Advantages/Disadvantages
   c. Building Blocks
   d. In Practice
   e. Patterns
      i. Design
      ii. Architecture
   f. Bounded Context
   g. Anti-Corruption Layer
3. How to begin using DDD?
4. Documentation
5. Q&A

About me

- Degree of Mathematics and Computer Sciences, University of Minho (2007);
- Professional Experience:
  - 13 years in software development
    - 5 years at Farfetch
      - 2 years as a Senior .Net Engineer;
      - last 3 years as a Teams Lead;

- Enthusiastic by
  - DDD and CQRS
  - SCRUM methodology
  - People coaching and performance

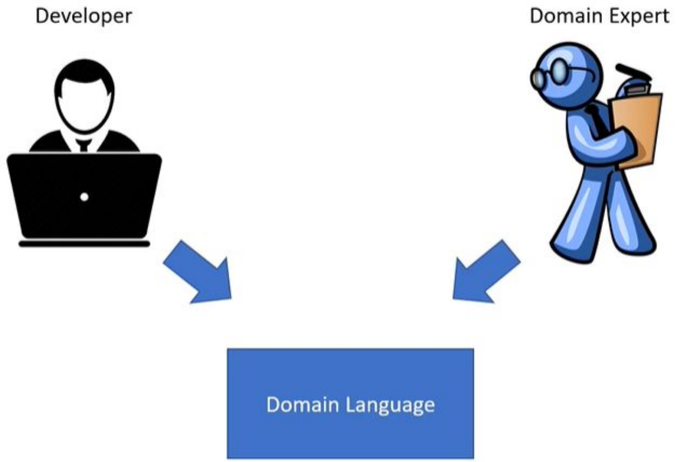
- Photography

LinkedIn

DOMAIN DRIVEN DESIGN

What is DDD?

- Development philosophy
- Defined by Eric Evans
- Used in large and complex systems
- Developers work closely to domain experts
- Uses a ubiquitous language



Any fool can write code that a computer can understand.
Good programmers write code that humans can understand.
in Refactoring: Improving the Design of Existing Code, 1999

Advantages

- Communication
- Flexibility
- Maintainability

Disadvantages

- Requires Domain Expertise
- Costly

Suitable for projects that are

- Long term
- High domain complexity
- Have clear benefits of the communication

DDD Building Blocks

- **Domain:** the subject area of the program;
- **Domain Model:** a conceptual object model representing different parts of the domain;
- **Bounded Context:** the context to which a model can be applied;
- **Ubiquitous Language:** the common domain language used by the team and in the code;

DDD Building Blocks

- **Entity:** an object that is defined by a thread of continuity and its identity rather than by its attributes;

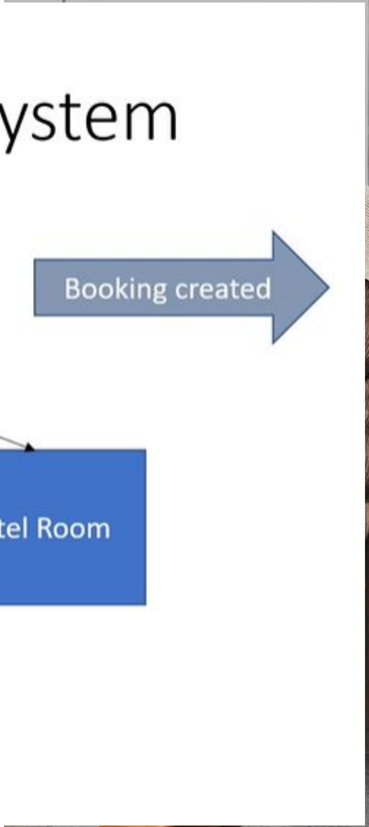
- **Value:** An object that has attributes but no identity (it is immutable);

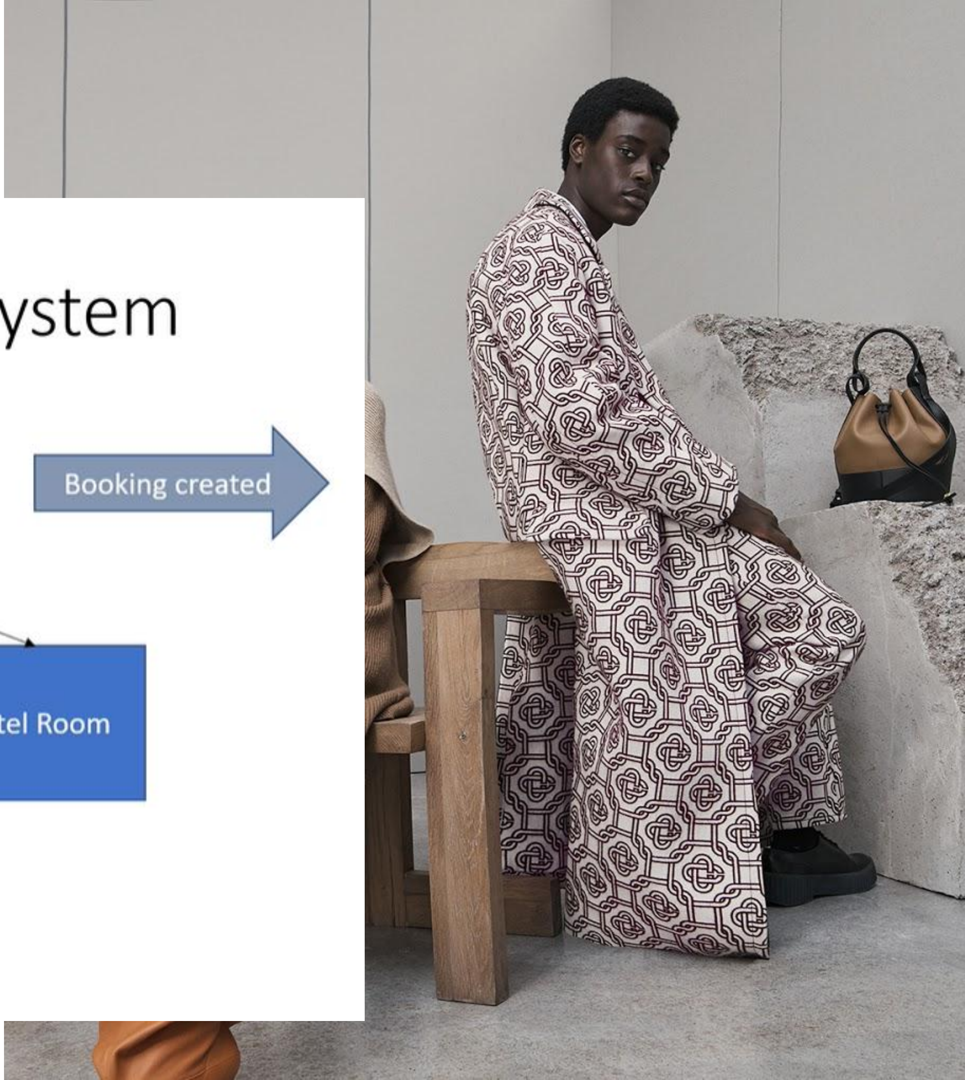
- **Aggregate:** A collection of values and entities which are bound together by a root entity, known as an aggregate root;

- **Domain Event:** an event directly related to the domain;

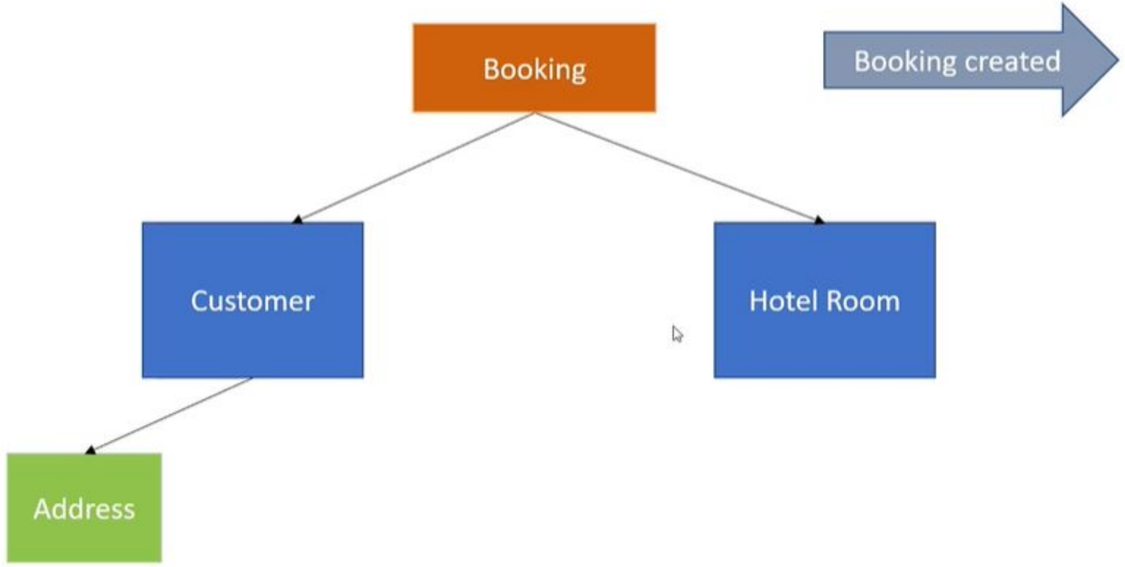
DDD - Use Case

DDD in Practice



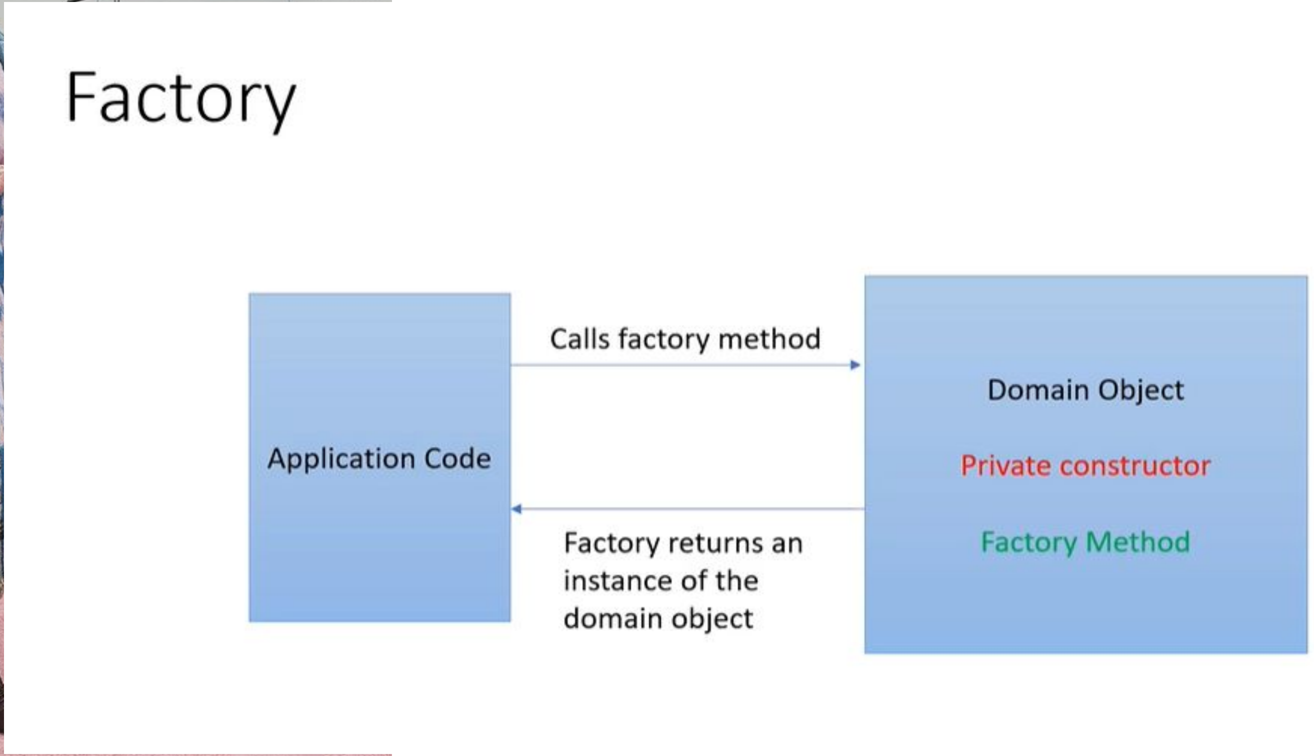


An example: Hotel booking system



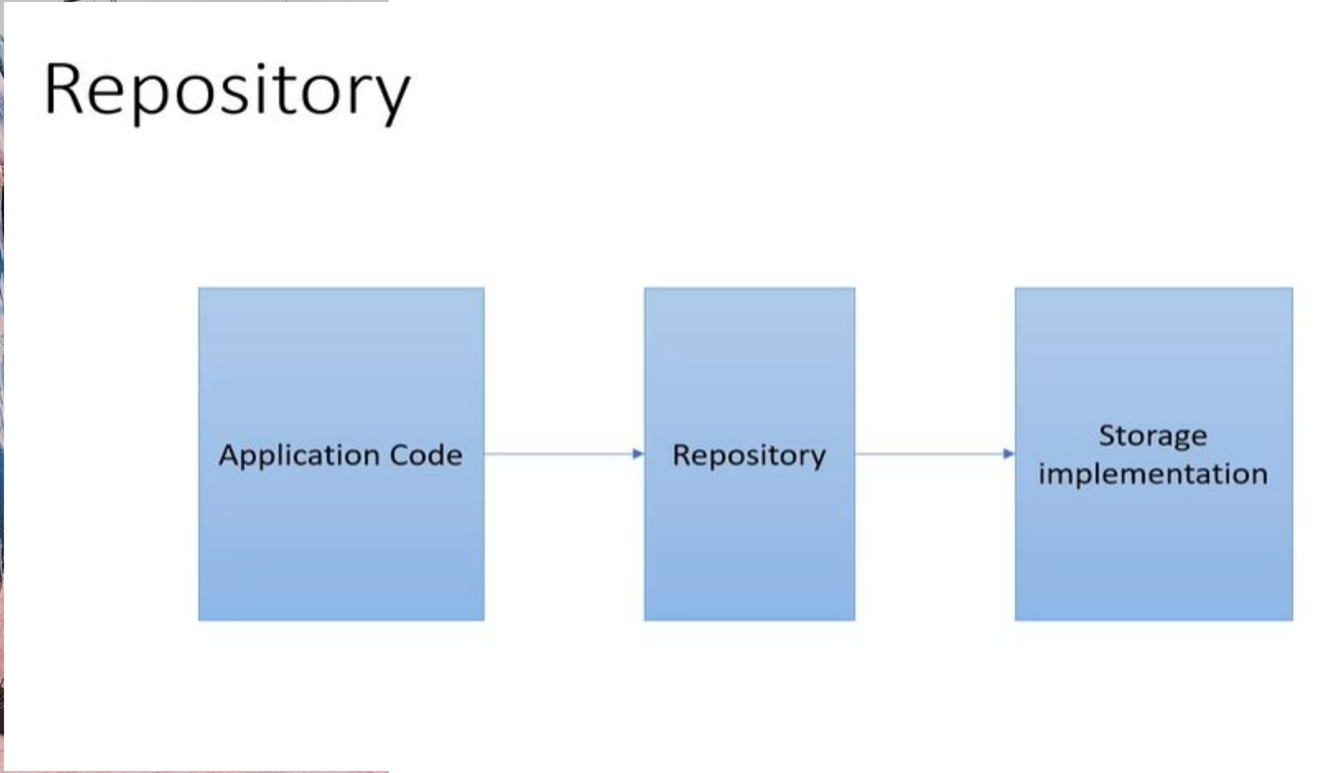
DDD - Design/Architectural Patterns

Design Patterns

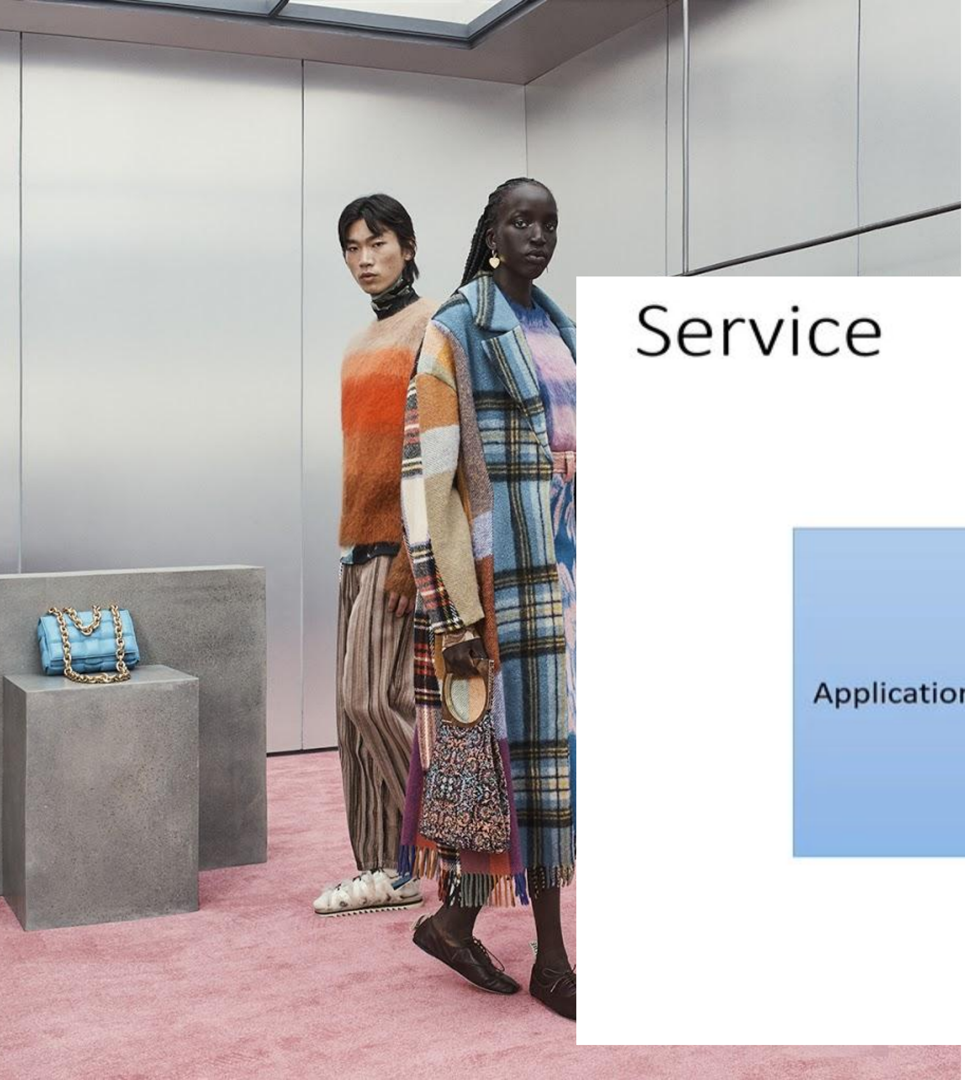


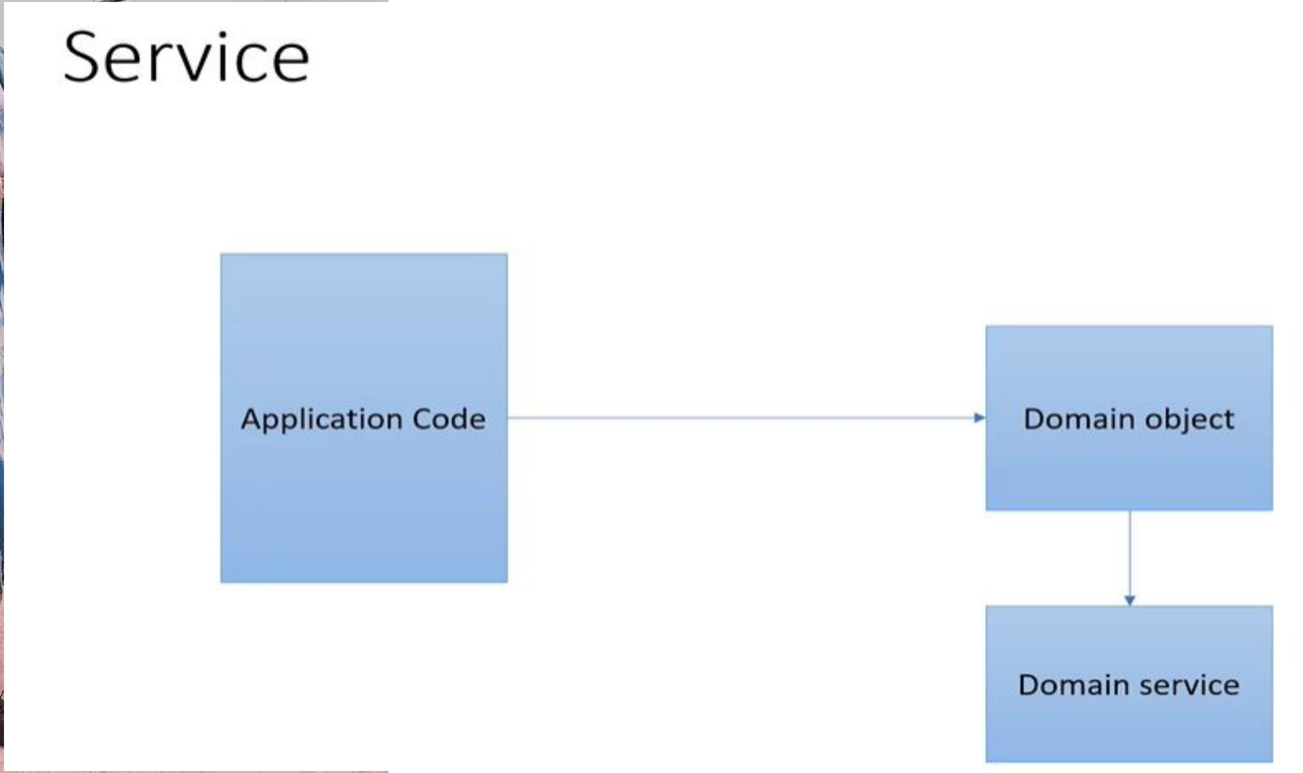
Factory

Design Patterns


Repository

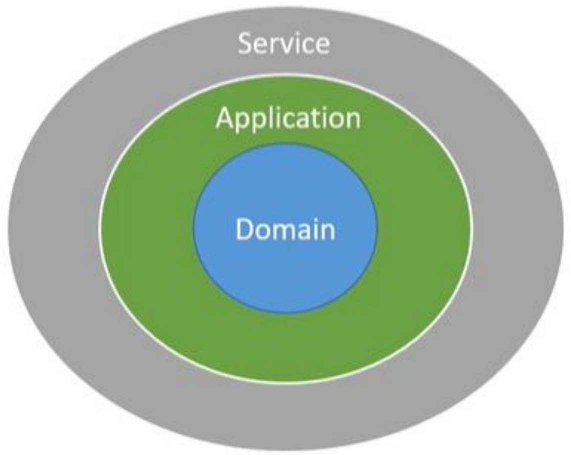
Design Patterns





Architectural Pattern

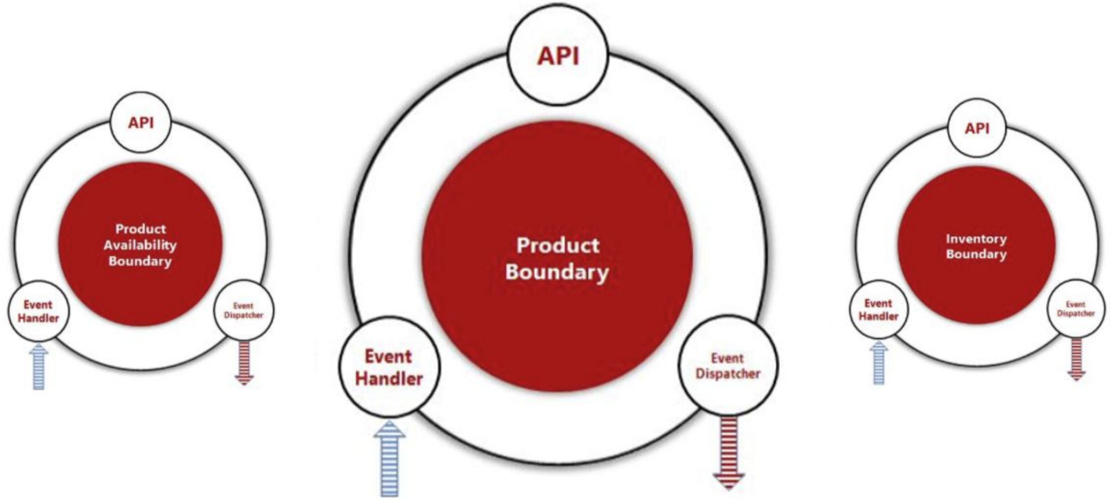
Onion Architecture



DDD - Bounded Context and Anti-Corruption Layer

Bounded Context

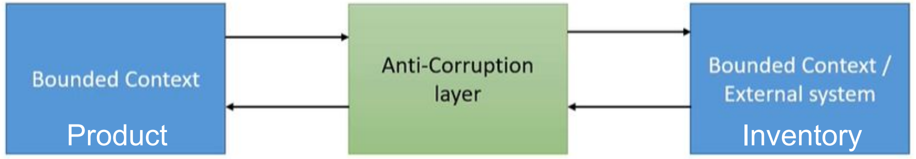
- Internal services
- Boundary services:
  - API
  - Handler
  - Dispatcher



- **Main advantages:**
  - Isolated and limited components:
    - develop and deploy with confidence
    - easier to scale up
    - single responsibility
  - Restrict access to internal services

Anti-Corruption Layer

- Different Bounded Contexts can share concepts;
- How to avoid problems in the communication between them?
  - Using Anti-corruption layers

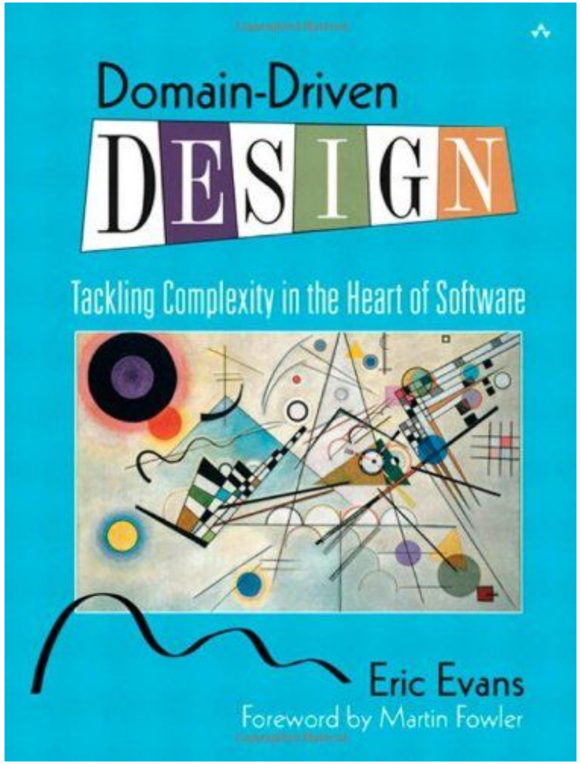


Using DDD

How to begin using DDD?

- Try on a small subset of your domain;
- Break down larger domains into smaller;
- Microservices, CQRS and Event Sourcing;
- Learn more about DDD;

Documentation


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

Q&A