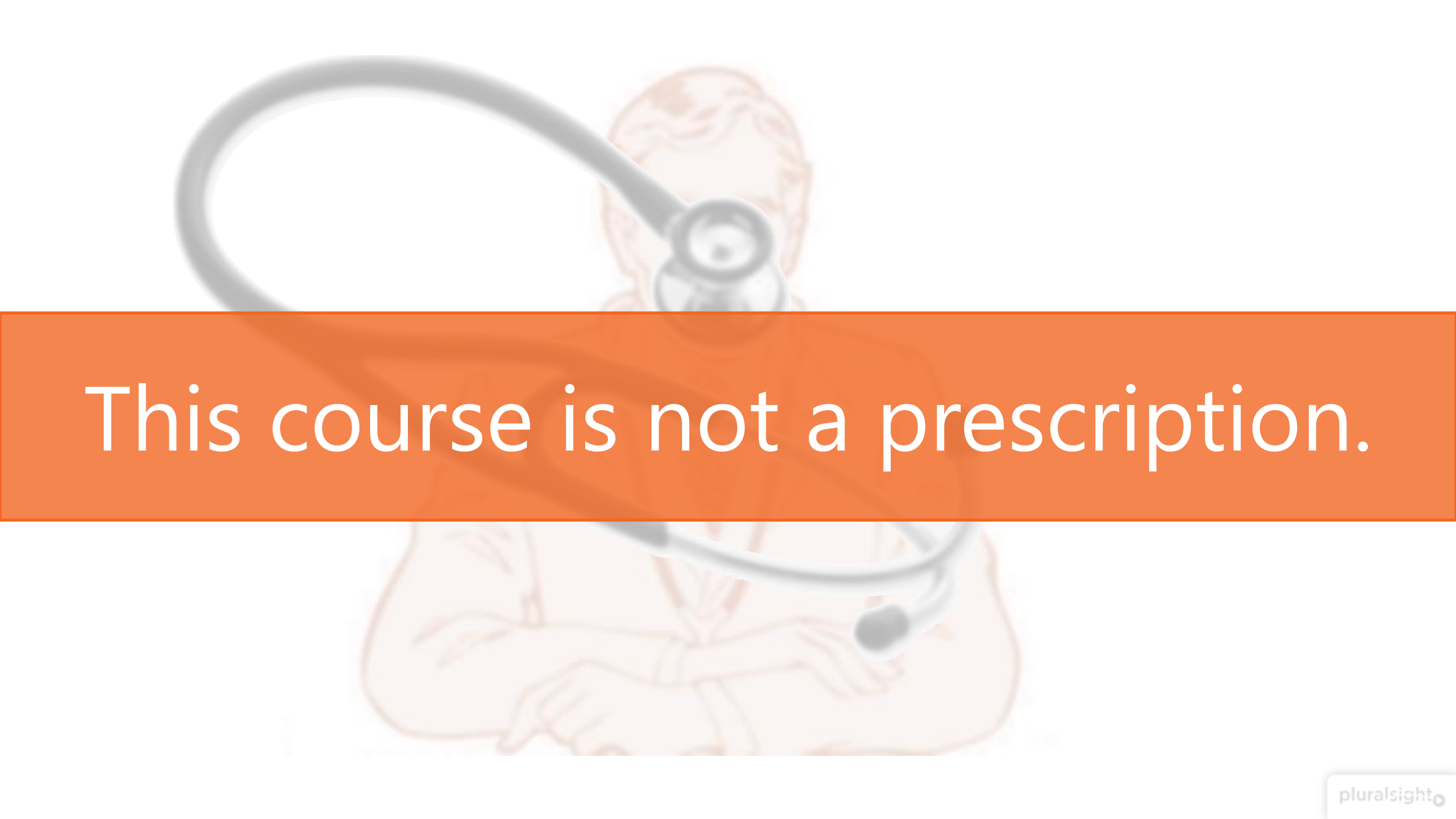
Modern Software Architecture: Domain Models, CQRS, and Event Sourcing

DDD AT A GLANCE

Dino Esposito
AUTHOR

@despos www.software2cents.wordpress.com



This course is not a prescription.

Key Points

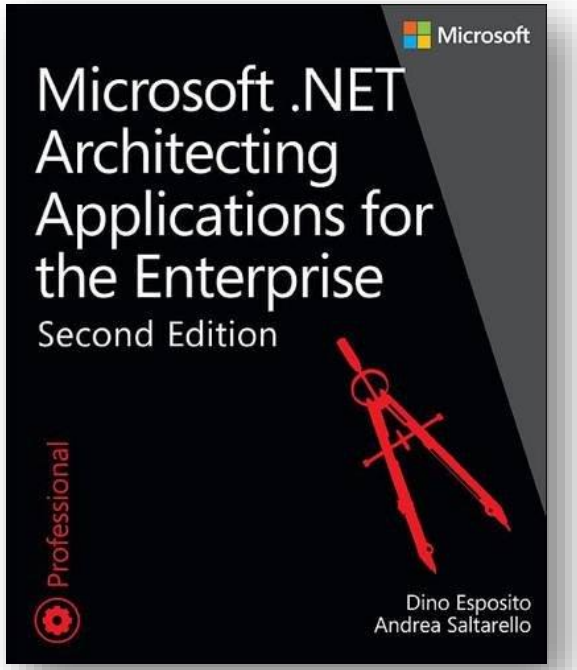
Repositioning
Domain-driven Design

Exploring Supporting
Architectures

UX-first Design
Methodology

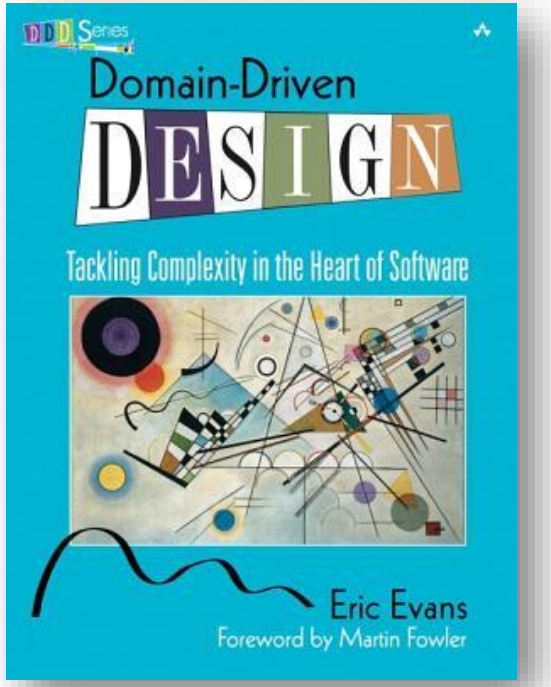
References

http://naa4e.codeplex.com



http://facebook.com/naa4e

Domain-driven Design in History



Introduced 10+ years ago by **Eric Evans**

Primary intent of **tackling complexity** in the heart of software

Innovative guidelines and approach to design

Where Does Complexity Come From?

Make sense of requirements

Build a (relational) data model

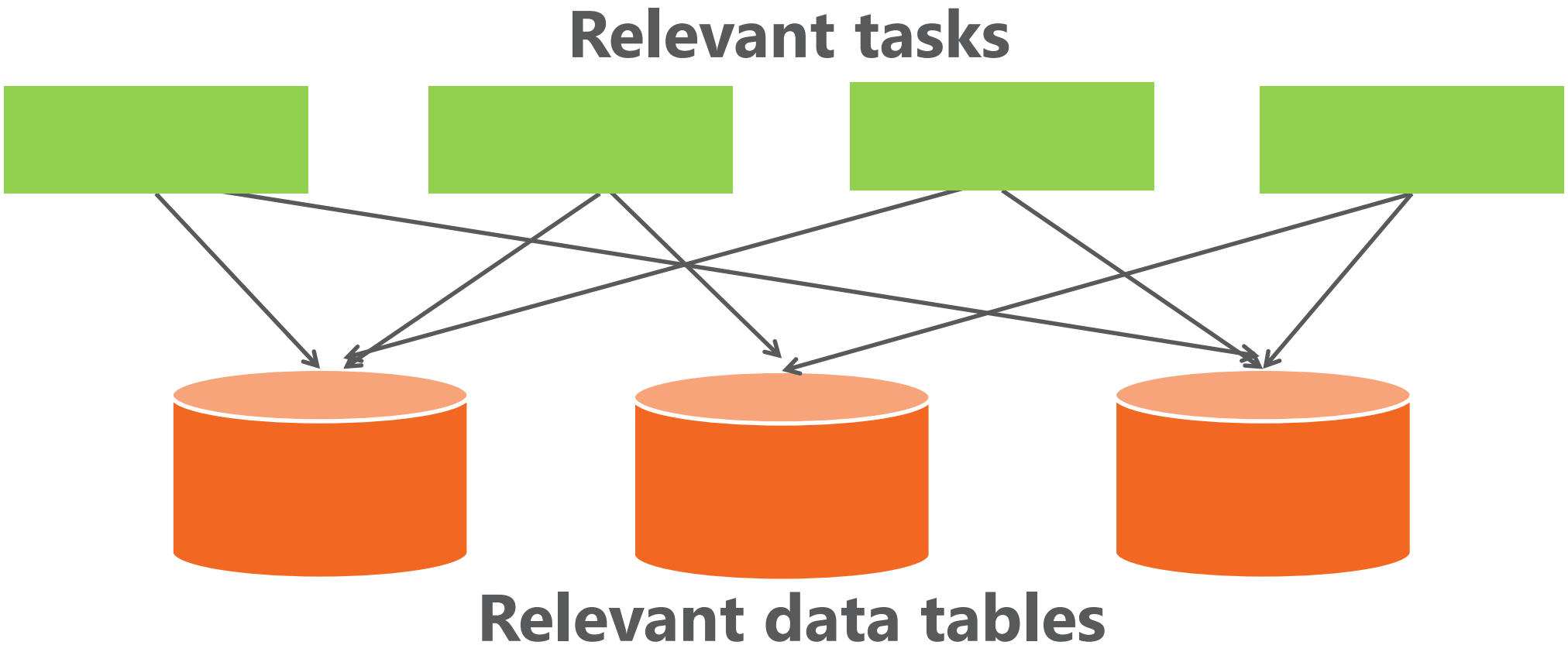
Identify relevant tasks and data tables

Build a user interface

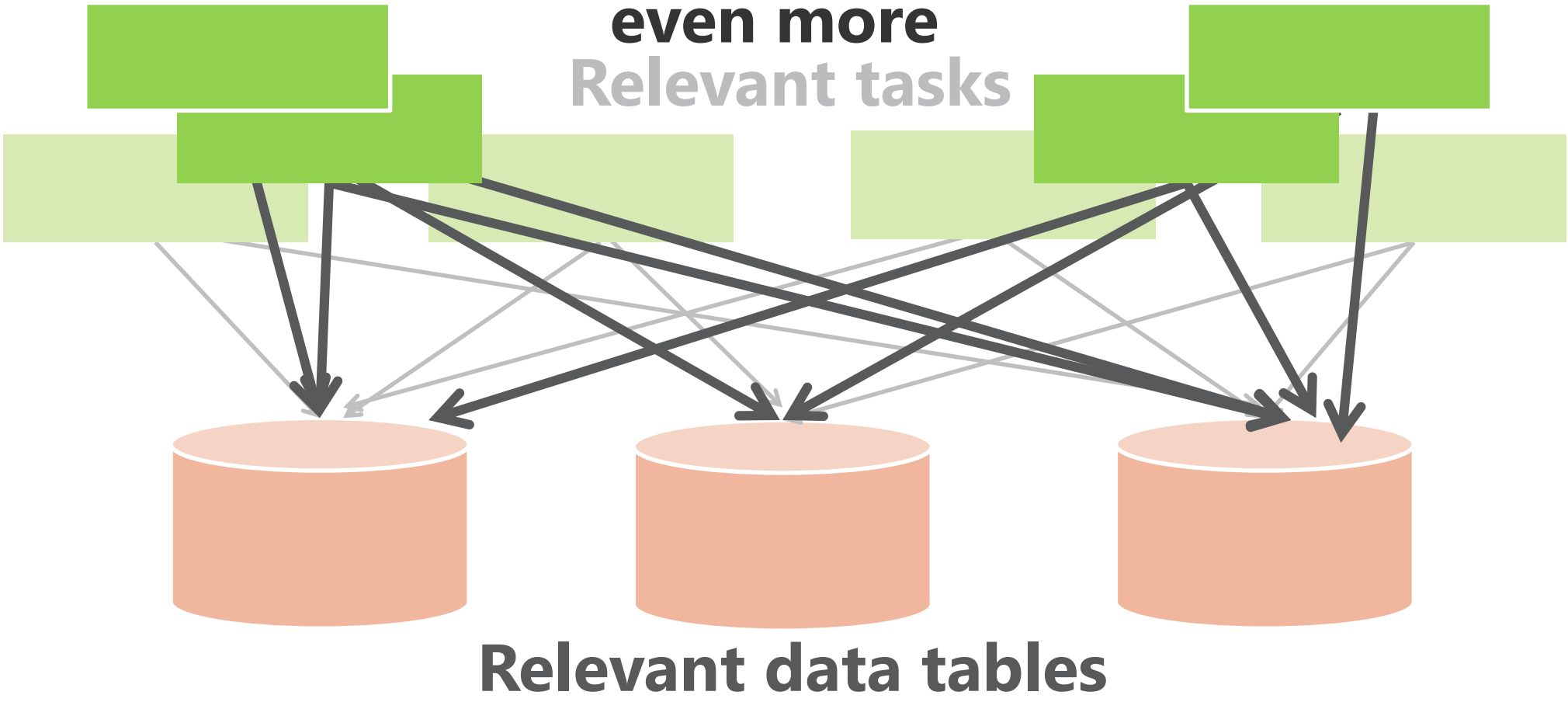
Close to what users wanted but…

Try again over and over

Where Does Complexity Come From?



Where Does Complexity Come From?



Unmanageable

http://www.laputan.org/pub/foote/mud.pdf

Big Ball of Mud (BBM)

A system that's largely unstructured, padded with hidden dependencies between parts, with a lot of data and code duplication and an unclear identification of layers and concerns—a spaghetti code jungle.

Ordinary Stories

Why Is DDD so Intriguing?

Captured known elements of design process

Organized them into a set of principles

Made **domain modeling** the focus of development

Different approach to building business logic

DDD Is **Still** About Business Logic

Crunch knowledge about the domain

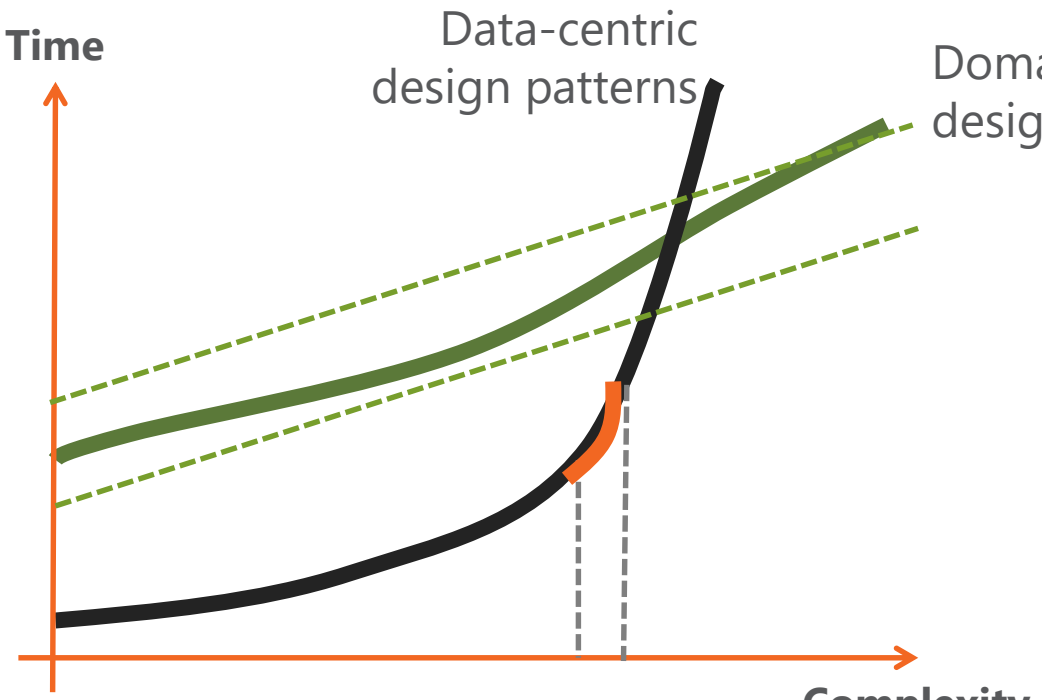
Recognize subdomains

Design a rich domain model

Code by telling objects in the domain model what do to

The Secret Dream of Any Developer

An all-encompassing object model describing the entire domain



NOTE: Adapted from Martin Fowler's PoEAA

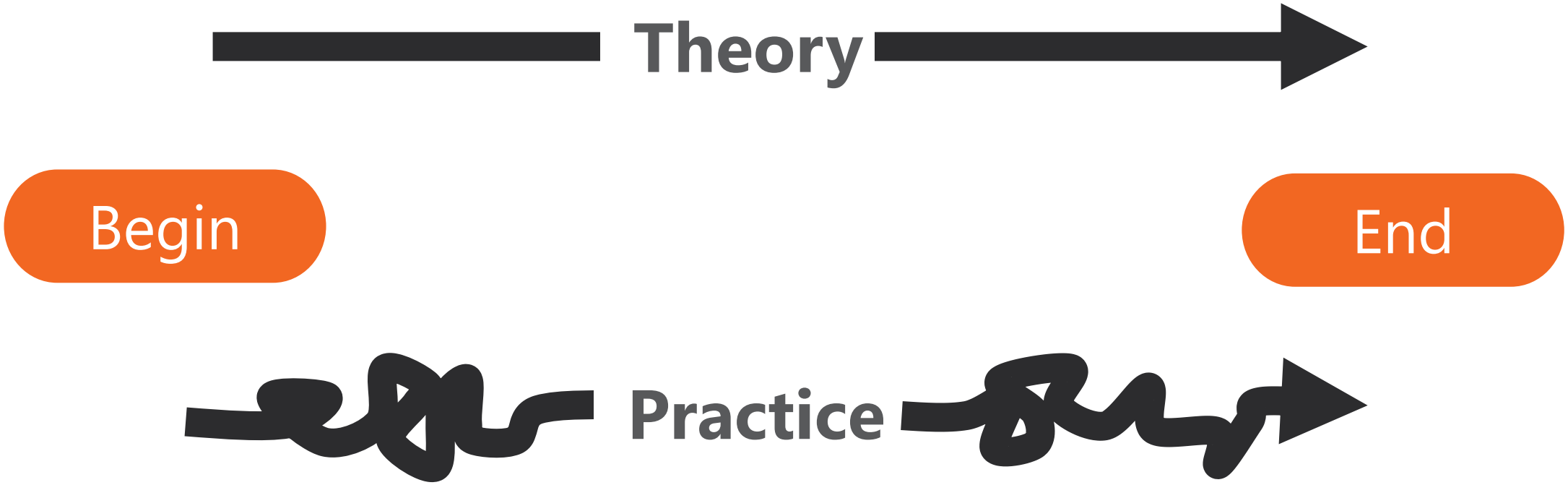
Give me enough time and enough specs and I'll build the world for you.

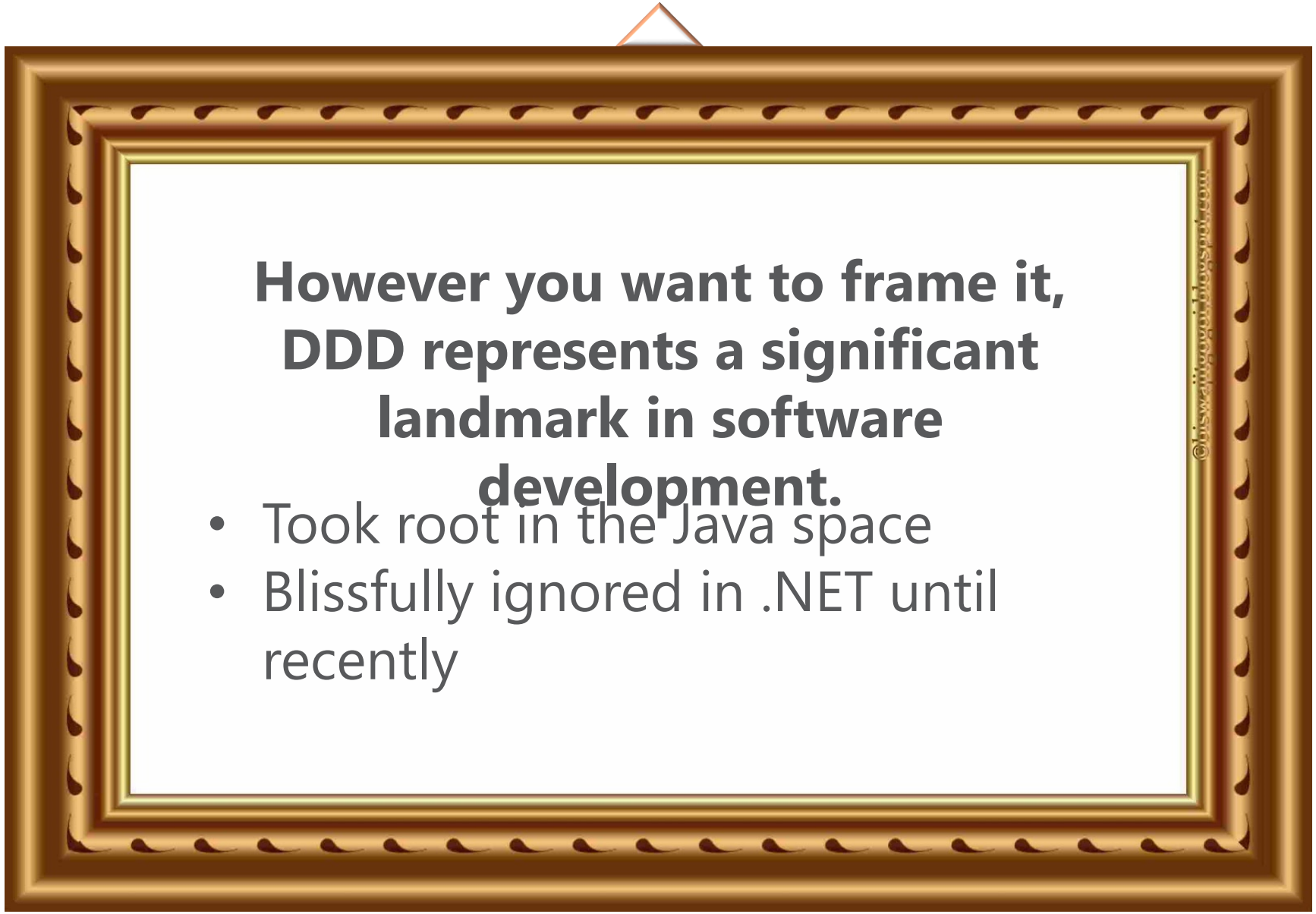
Supreme Goal

Tackling **Complexity** in the Heart of Software

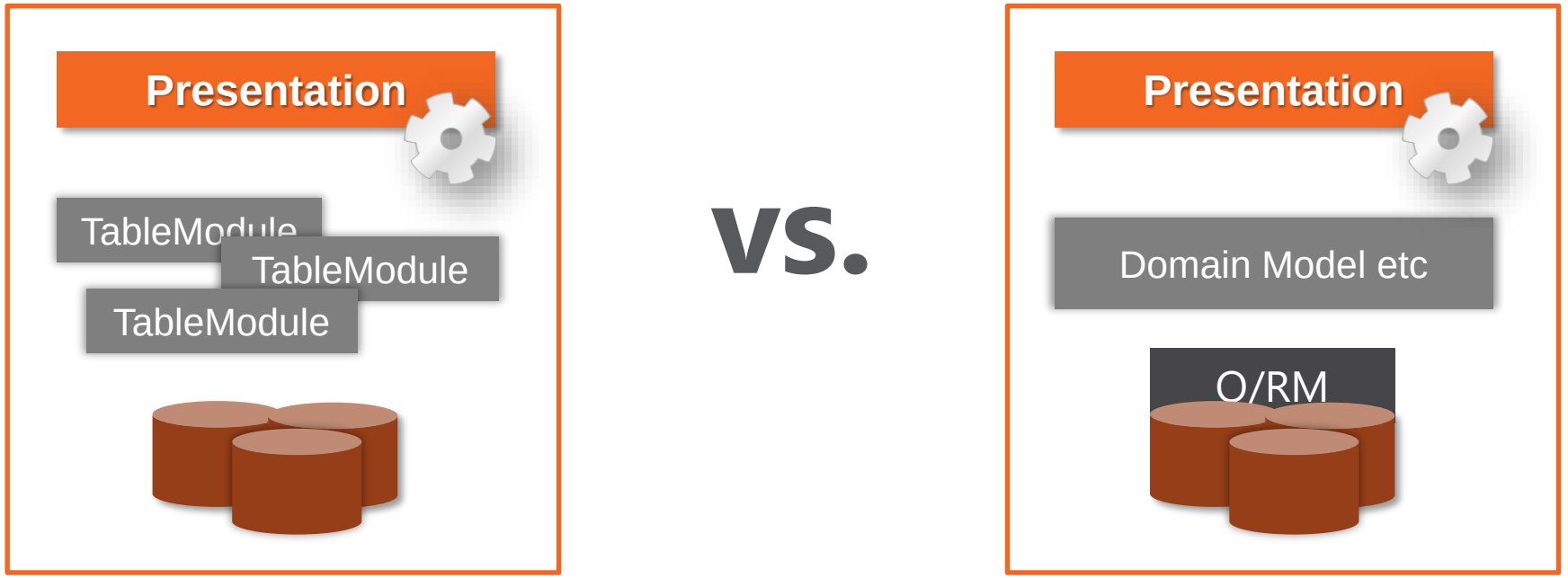
- Wonderful idea
- Not a mere promise
- Not really hard to do right
- But just easier to do wrong

In Other Words ...





DDD is another way to organize business logic



Why should I spend days around the design of a class when I can find a **non-classy** way out far more quickly?

DDD Was Not Cheating

- DDD was the right hammer **bundled** with the wrong set of nails
- DDD is more about analysis than about actual coding strategies

2009: Five Years Later

The main focus of DDD has shifted

- **Discovering** the domain architecture more than **organizing** the business logic
- **Domain Model** remains a valid pattern to organize the business logic but other patterns can be used as well

Object-oriented models | Functional models | CQRS | Classic 3-tier | 2-tier

Eric Evans talk at **QCON 2009**

http://www.youtube.com/watch?v=IE6Hxz4yomA

Common Summary of DDD

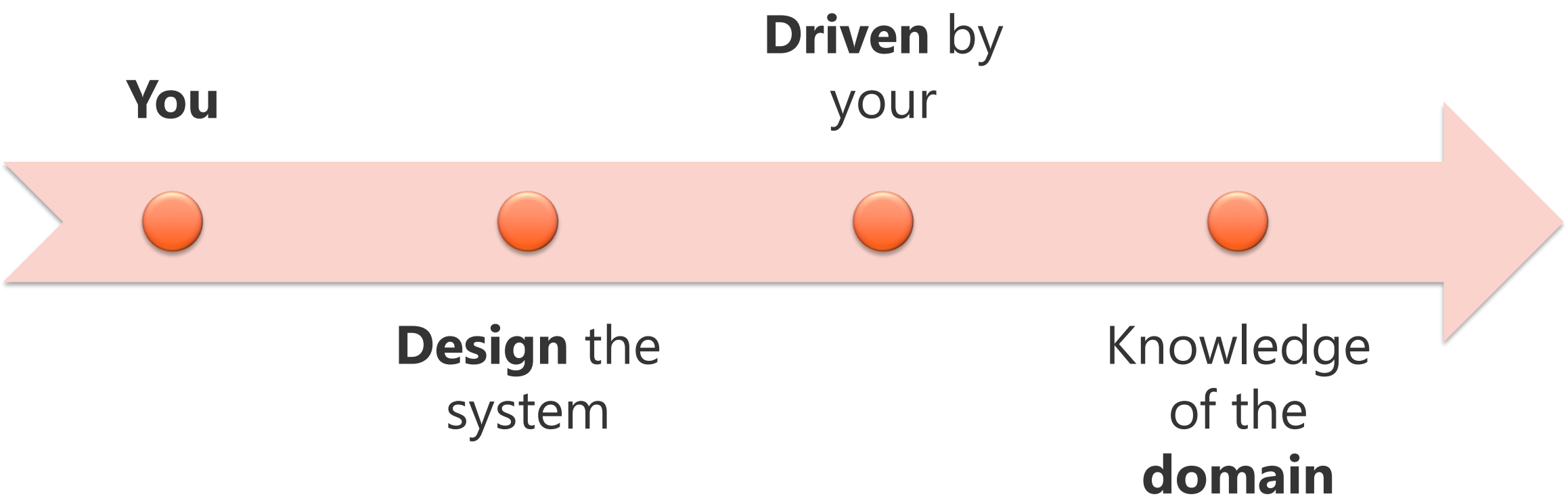
Build an object model for the business domain

- Call it a "domain model"

Consume the model in a layered architecture

- 4 layers, business logic split and renamed
- Application layer and Domain layer

DDD == Domain-driven Design



DDD has **two** distinct parts.

You always **need one** but can happily **ignore** the other.

One of many possible
supporting architectures

Design

driven by

Domain

How Do You Do **Design Driven by the Domain**?