

Domain-Driven Design Fundamentals

Introducing Domain-Driven Design

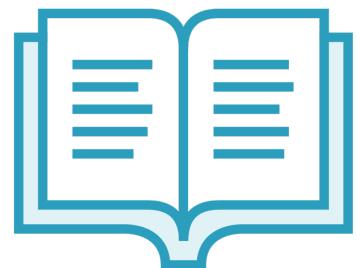


Steve Smith
Force Multiplier for
Dev Teams
[@ardalis](https://twitter.com/ardalis) ardalis.com



Julie Lerman
Software coach,
DDD Champion
[@julielerman](https://twitter.com/julielerman) thedatafarm.com

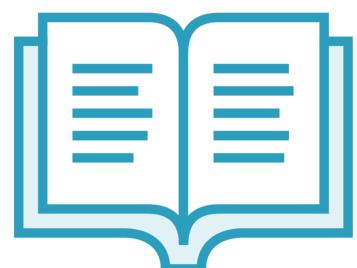
Some Recommended Books on DDD



Domain-Driven Design,
Eric Evans (2003)



**Applying Domain-Driven
Design and Patterns,**
Jimmy Nillson (2006)



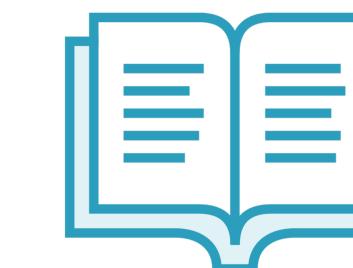
**Implementing Domain-
Driven Design,**
Vaughn Vernon (2013)



**Professional Domain-
Driven Design Patterns,**
**Scott Millett, Nick Tune
(2014)**



**Domain-Driven Design
Distilled,**
Vaughn Vernon



**Domain Modeling Made
Functional,**
Scott Wlaschin

Conferences

**Domain Driven
Design Europe
(Amsterdam)**

**KanDDDinsky
(Berlin)**

**Explore DDD
Conference
(Colorado)**

**DDD Exchange
from Skills Matter
(London)**

**Virtual DDD
Meetup
(on the web)**

Course Overview



Why should you care about DDD?

What does a DDD solution look like?

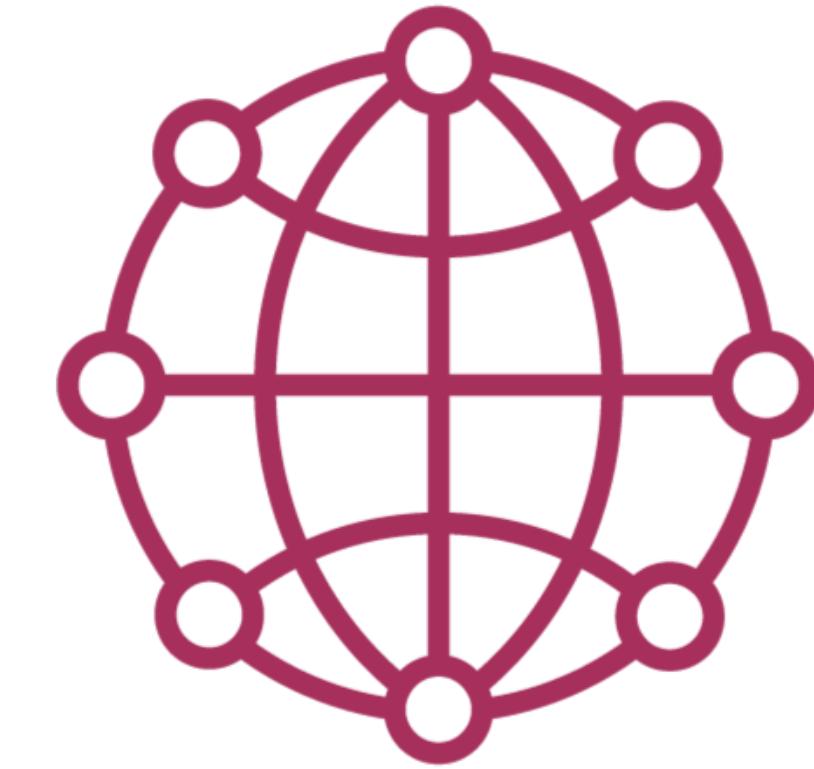
Modeling problems in software

Technical components of DDD

Managing complexity using DDD

Extending our solution as the domain evolves

Goals of This Course



“The more you know, the more you realize you know nothing.”

Socrates

Module Overview



What is Domain-Driven Design?

Goals of DDD

Benefits and potential drawbacks

View reference application

Understanding the Value of Domain-Driven Design

“Domain-Driven Design is an approach to software development that centers the development on programming a domain model that has a rich understanding of the processes and rules of a domain.”

Martin Fowler

Why DDD?





Why DDD?

Value Proposition of DDD

**Principles and patterns to
solve difficult problems**

**History of success
with complex projects**

**Aligns with practices
from our own experience**

**Clear, readable, testable code
that represents the domain**

Solve Problems

Write code

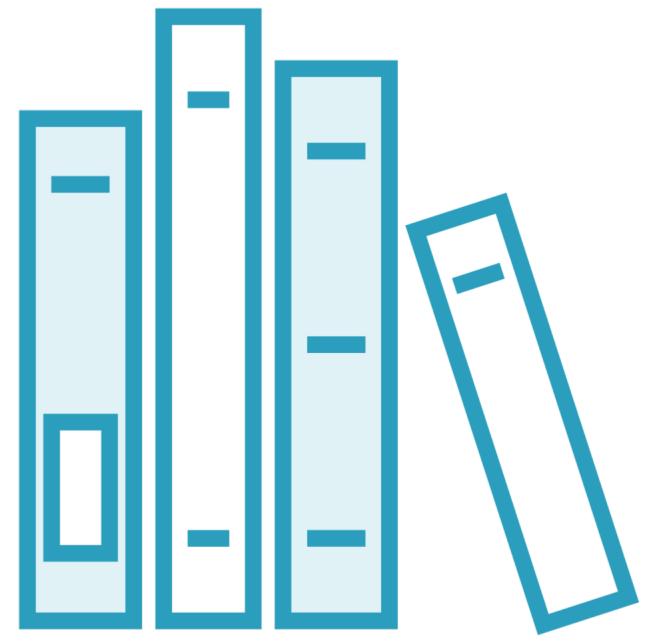
Design system

Understand client needs

Full partnership



Gaining a High-Level Understanding of DDD



“Tackling Complexity in the Heart of Software”

A photograph of a person standing on top of a large, craggy rock formation. The person is wearing a red jacket and blue jeans, and is looking out over a landscape of green fields and more rocky outcrops under a sky filled with white and grey clouds.

Interaction with domain experts



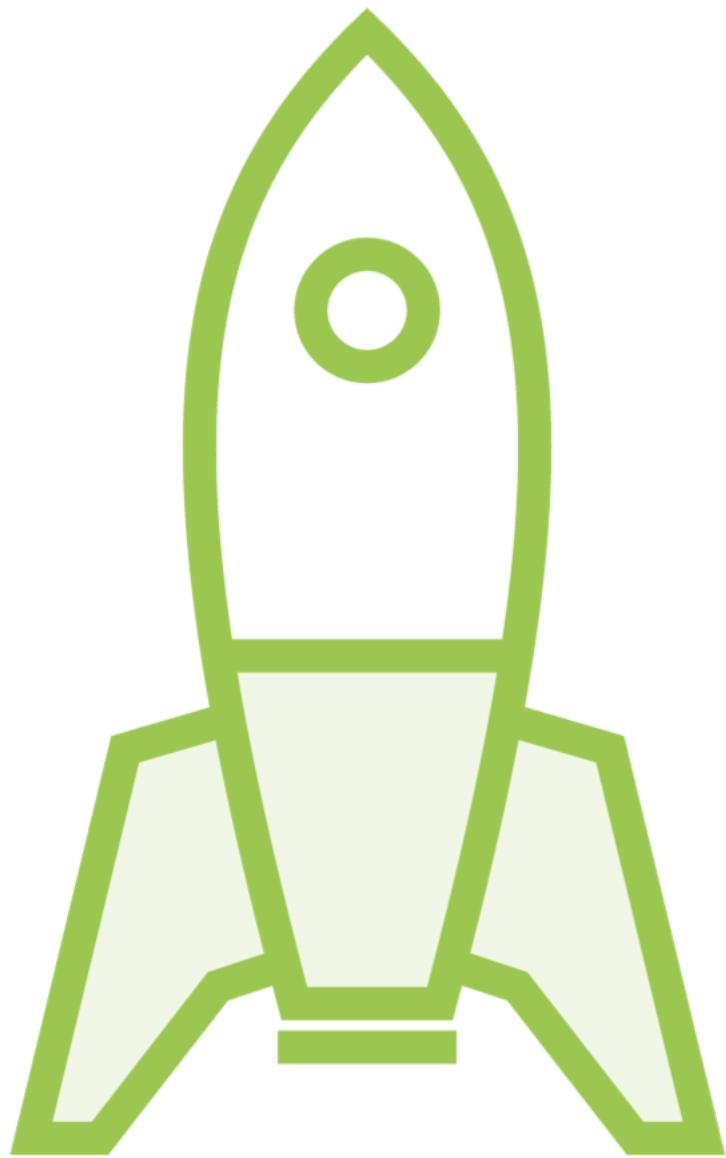
“You really need to cultivate your ability to communicate with business people to free up people's creative modeling.”

Eric Evans

A photograph of a person standing on top of a large, craggy rock formation. The person is wearing a red jacket and blue jeans, and has a backpack. The sky is filled with white and grey clouds. The rock formation is in the foreground, and the background shows more of the same rocky terrain and a green hillside.

Interaction with **domain experts**

Model a **single subdomain** at a time



Purchase materials

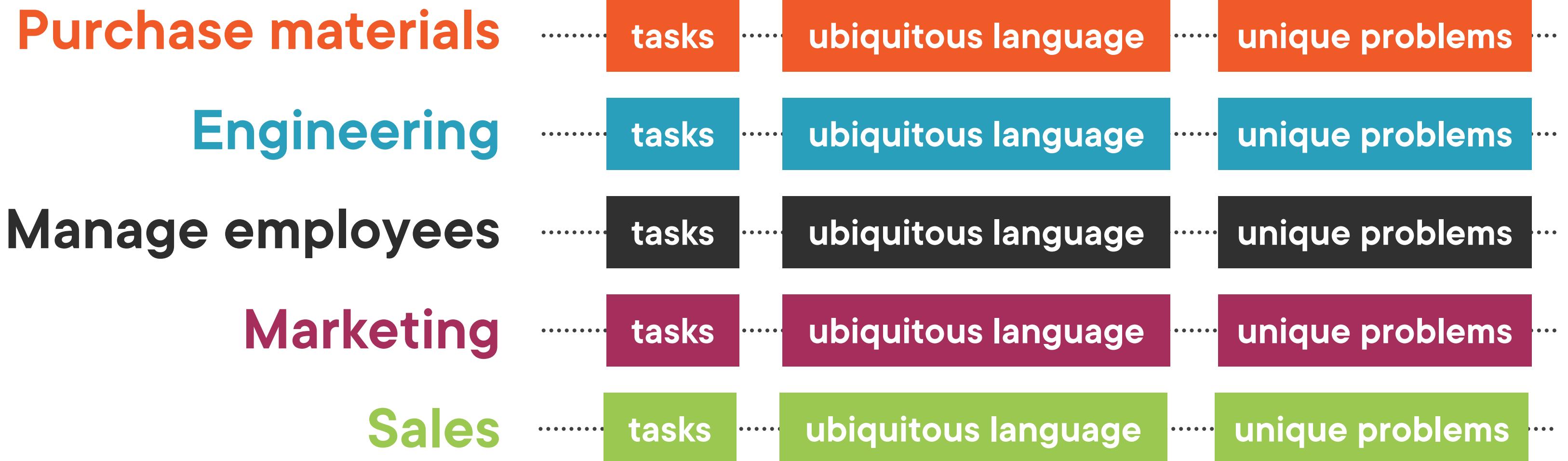
Engineering

Manage employees

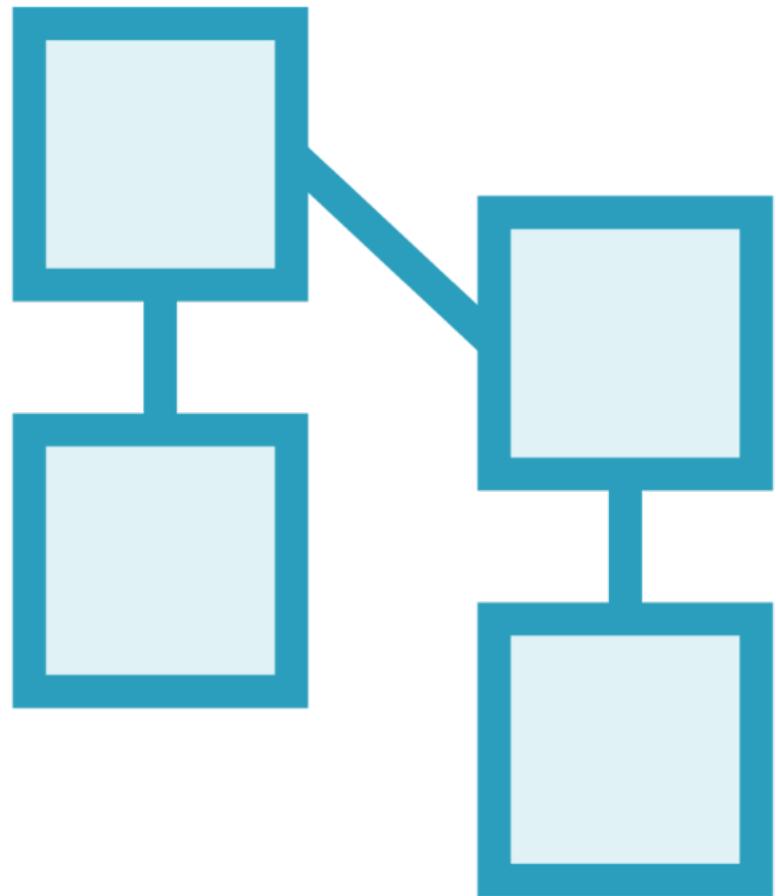
Marketing

Sales

Separate Subdomains For Each Problem

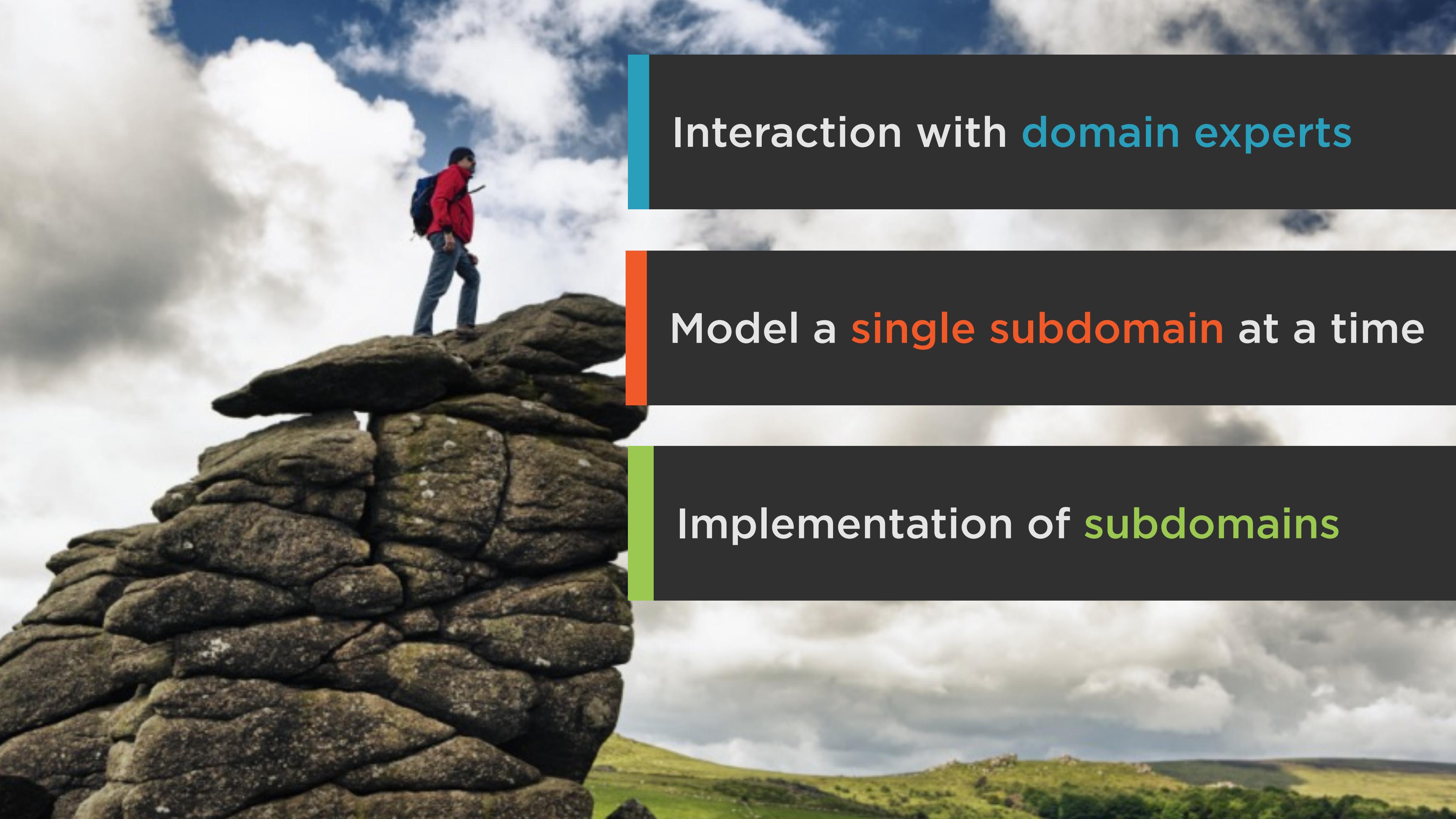


Divide
and
Conquer



Modeling

**How you decipher and
design each subdomain**



Interaction with **domain experts**

Model a **single subdomain** at a time

Implementation of **subdomains**

Separation of Concerns

plays an important role in
implementing subdomains

Exploring the Benefits and Potential Drawbacks of DDD

Benefits of Domain-Driven Design

Flexible

**Customer's
vision/perspective
of the problem**

**Path through a very
complex problem**

**Well-organized and
easily tested code**

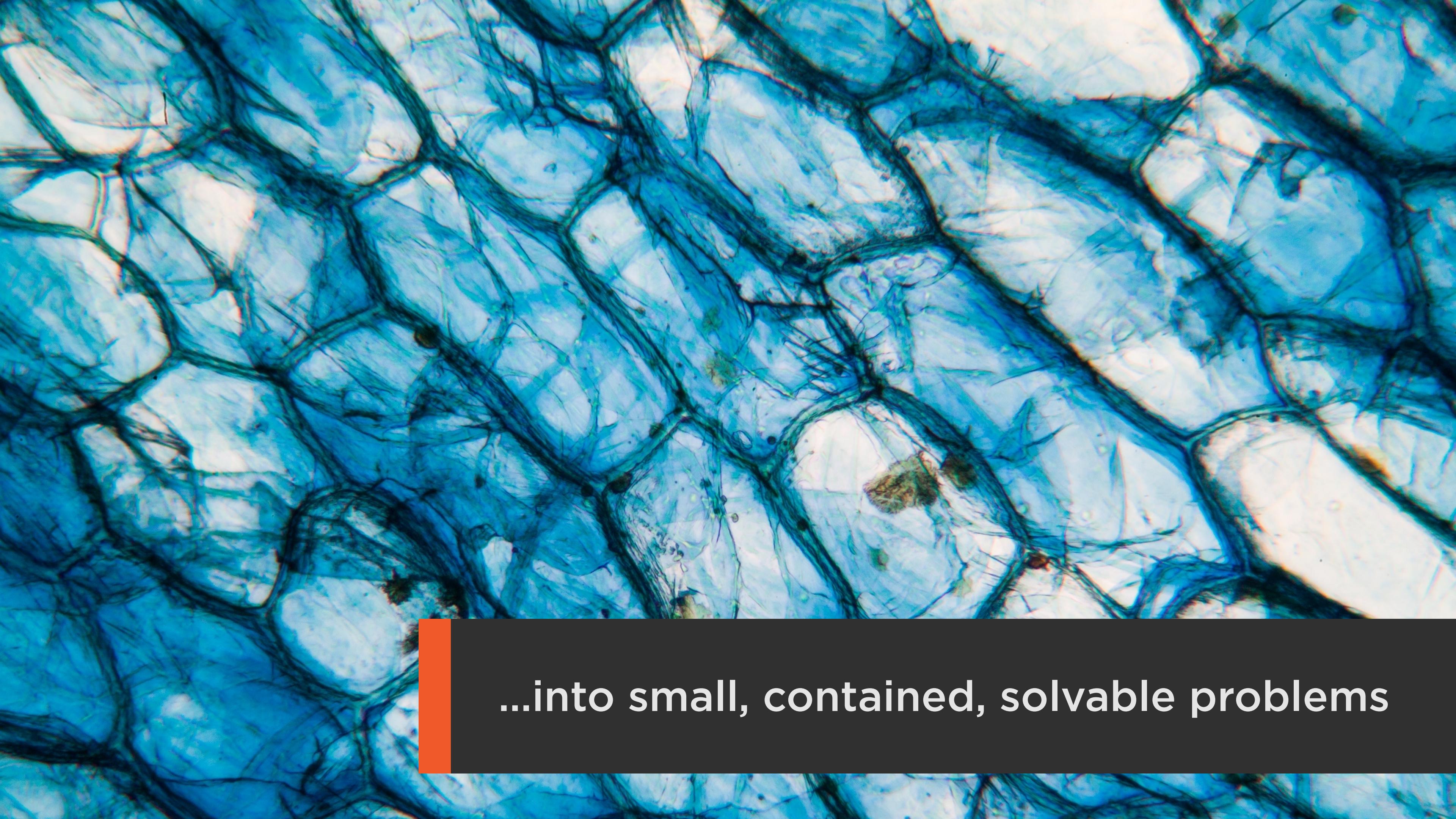
**Business logic lives
in one place**

**Many great patterns
to leverage**

DDD aims to
tackle business complexity,
not technical complexity



Transform big, messy problems...

A microscopic image showing a dense network of plant cells, likely from a stem or root, stained with iodine. The cells appear blue, and the iodine-stained structures, such as the endodermis and xylem, appear darker greenish-blue. A solid red vertical bar is positioned on the left side of the slide.

...into small, contained, solvable problems

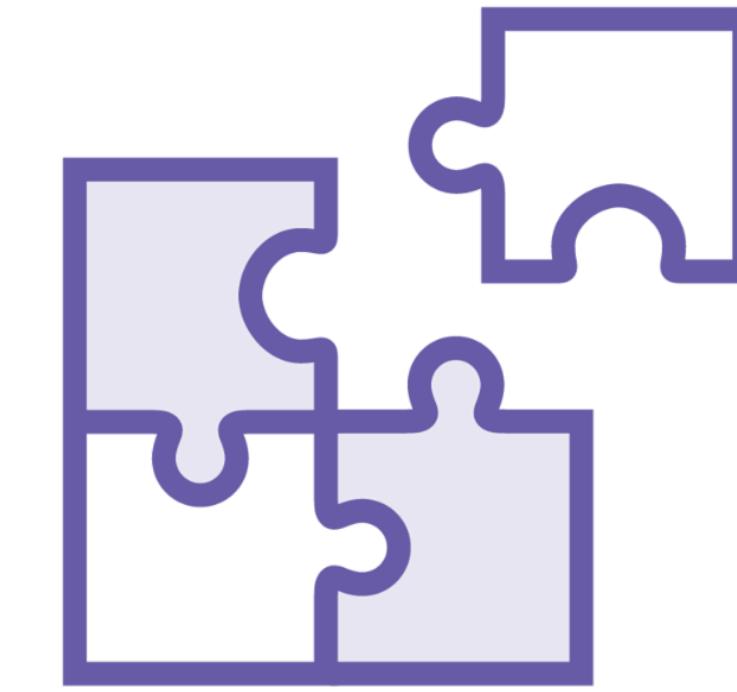
“While Domain-Driven Design provides many technical benefits, such as maintainability, it should be applied **only to complex domains** where the model and the linguistic processes provide clear benefits in the **communication of complex information**, and in the formulation of a **common understanding of the domain**.”

Eric Evans, Domain-Driven Design

Be Prepared for Time and Effort

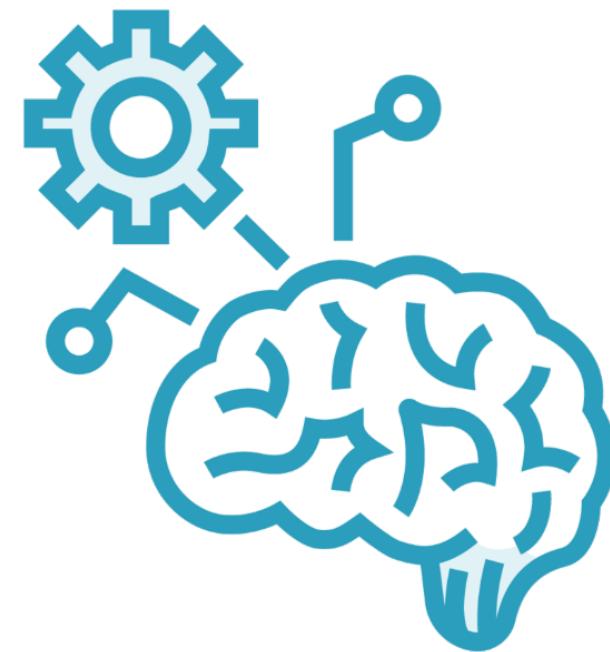


**Discuss and model the problem
with domain experts**

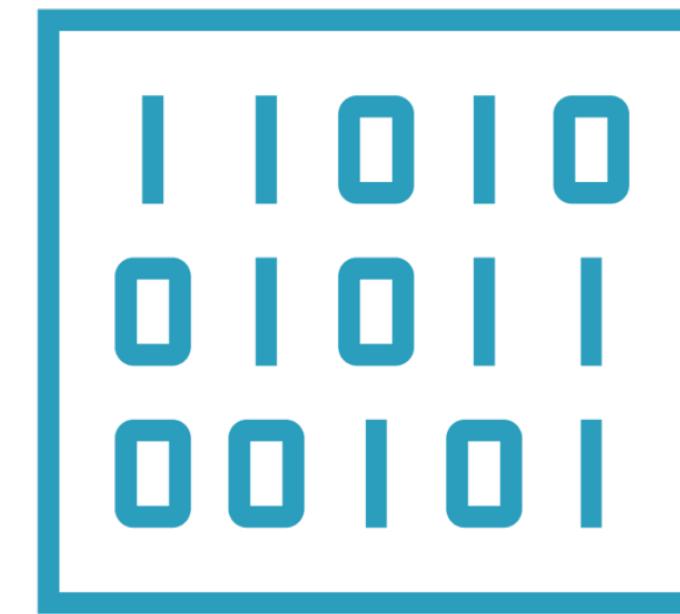


**Isolate domain logic from
other parts of application**

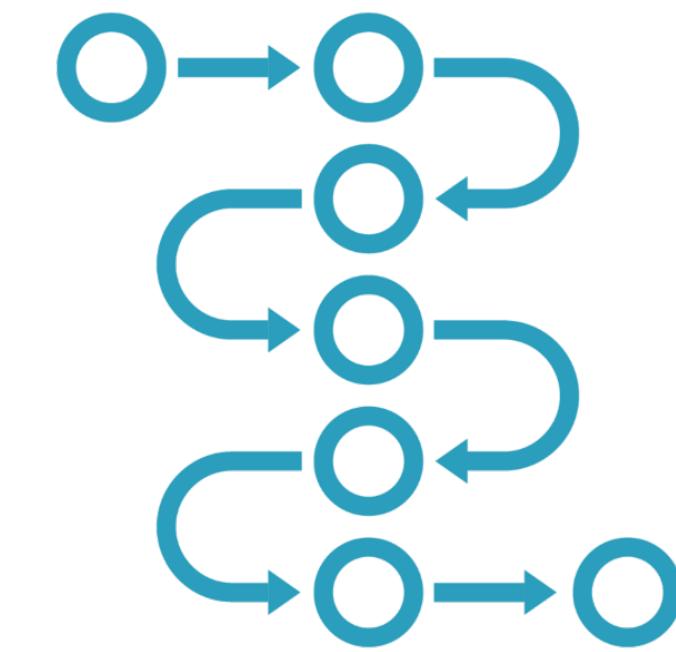
Be Prepared for a Learning Curve



New principles



New patterns

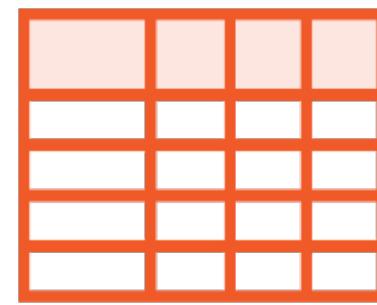


New processes

Be Thoughtful About Possible Overuse



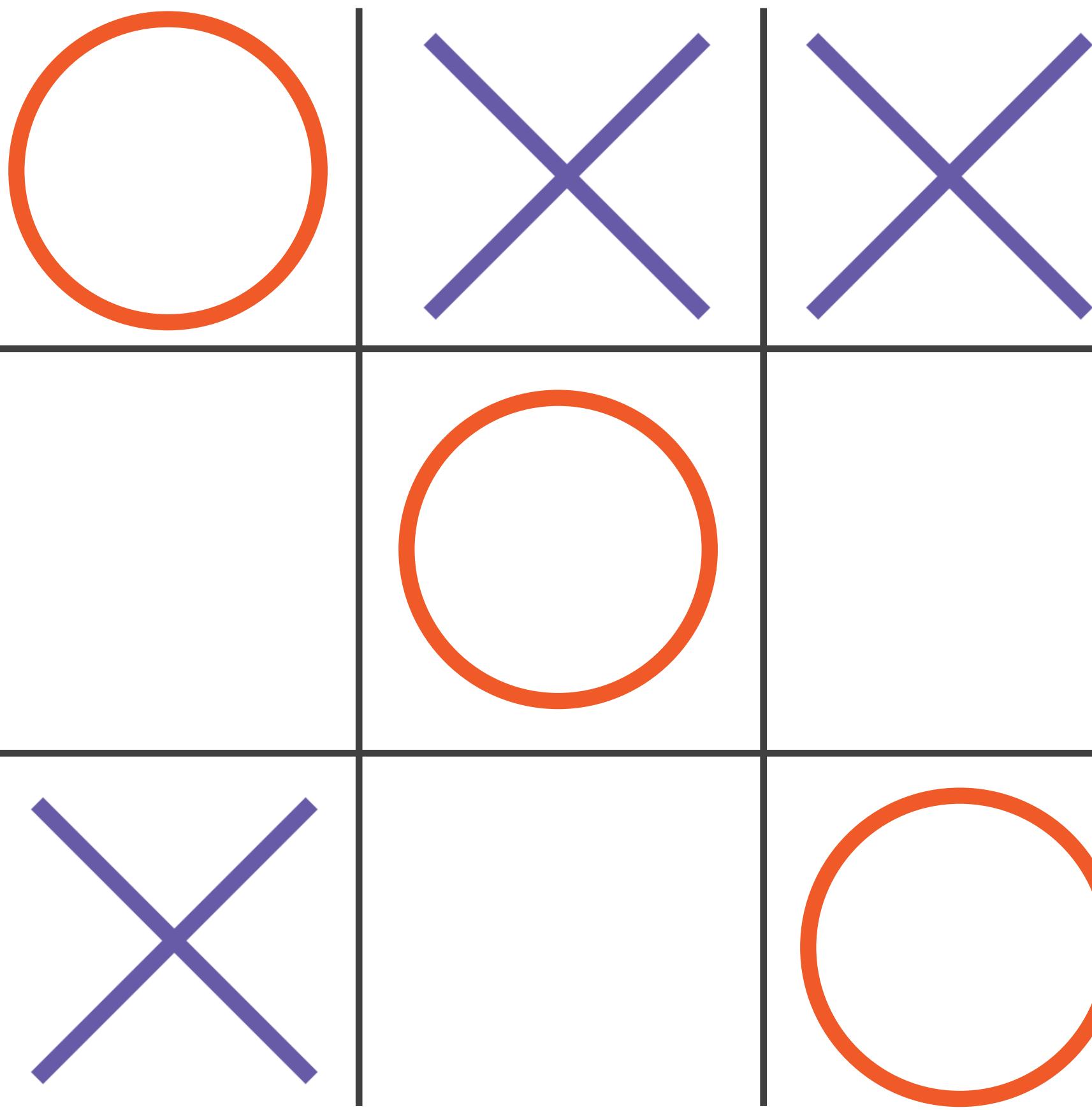
DDD is for handling complexity in business problems



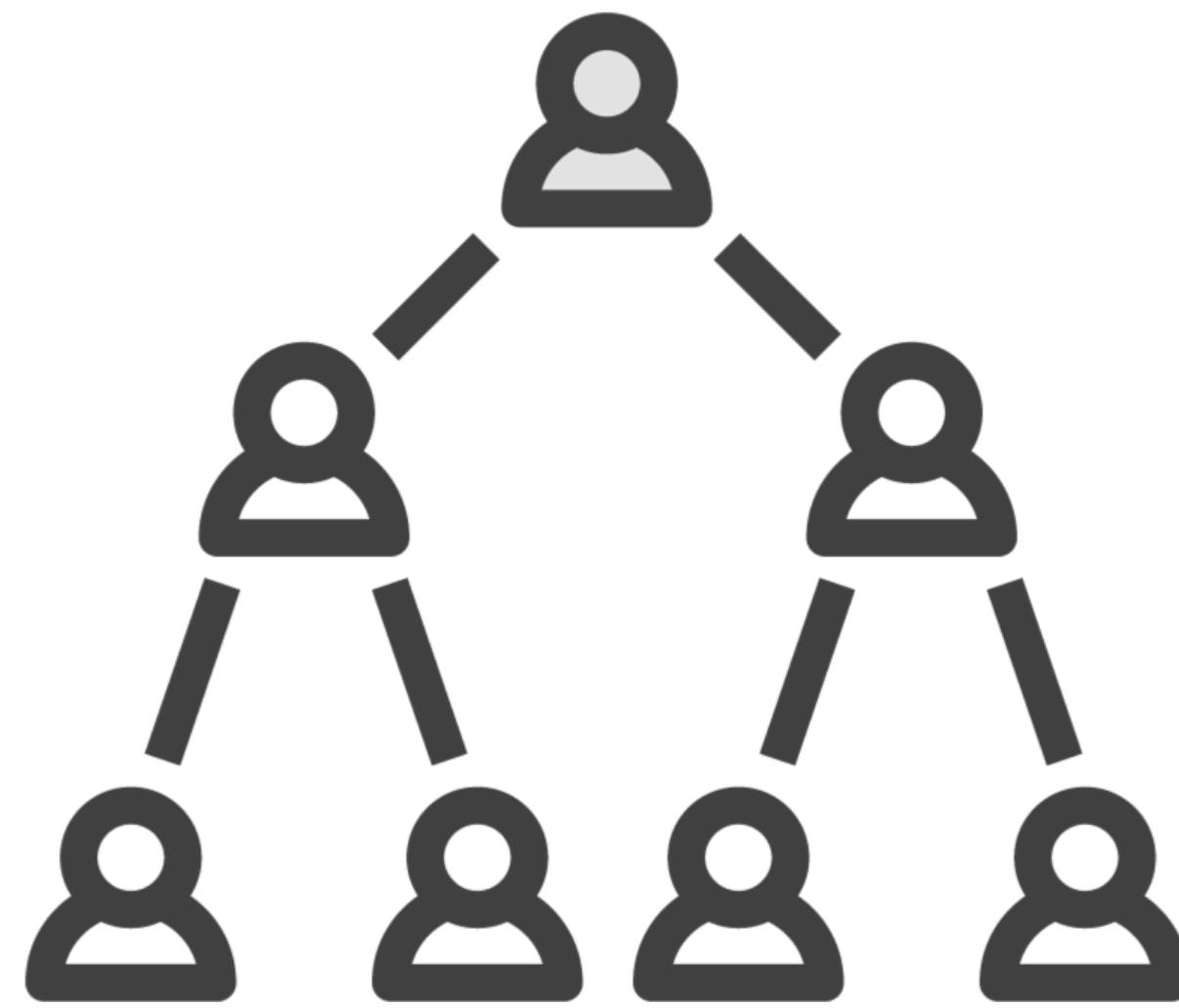
**Not just CRUD or
data-driven applications**



**Not just technical complexity
without business domain complexity**



Requires Team and Business Buy-In



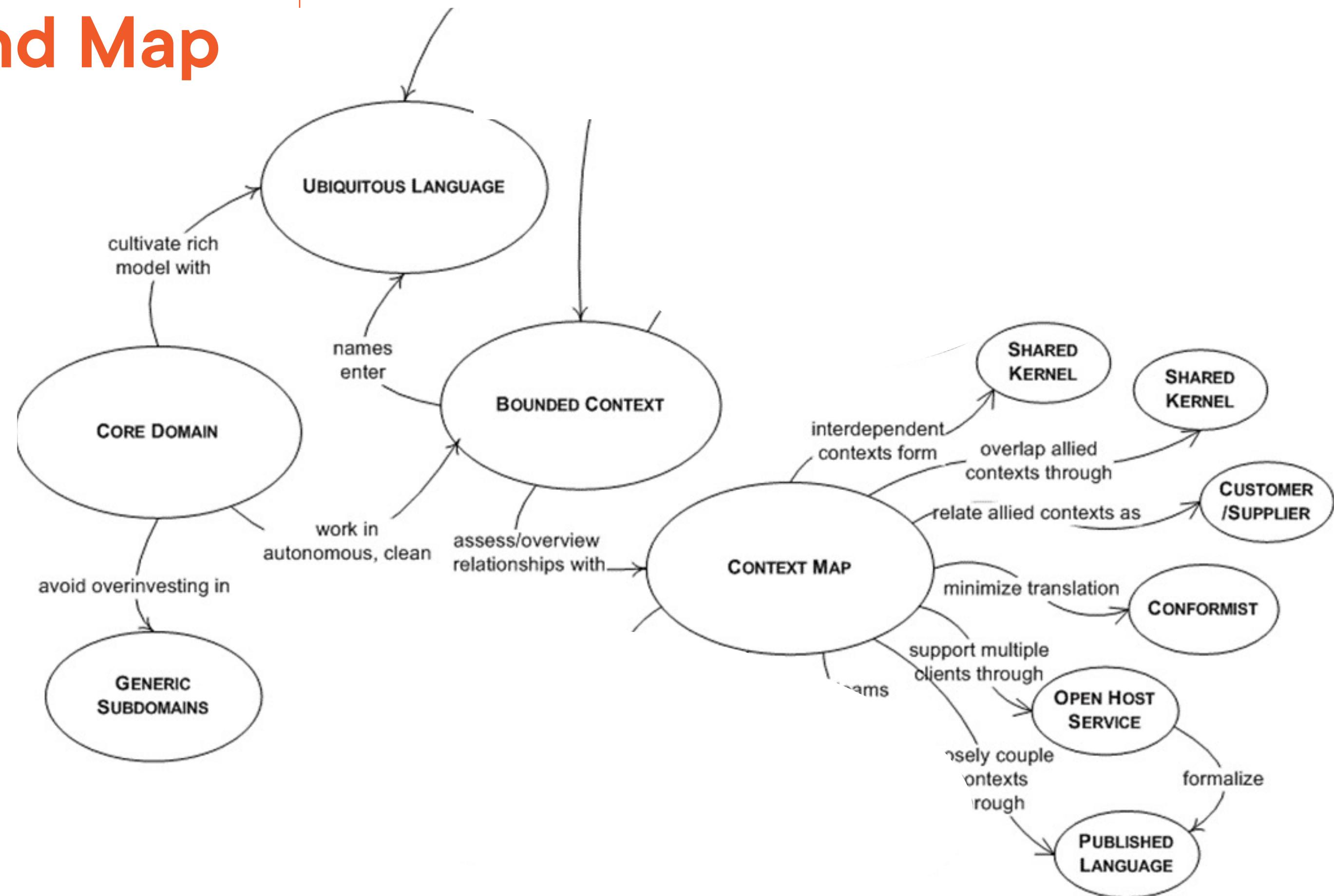
Inspecting a Mind Map of Domain-Driven Design

DDD Mind Map



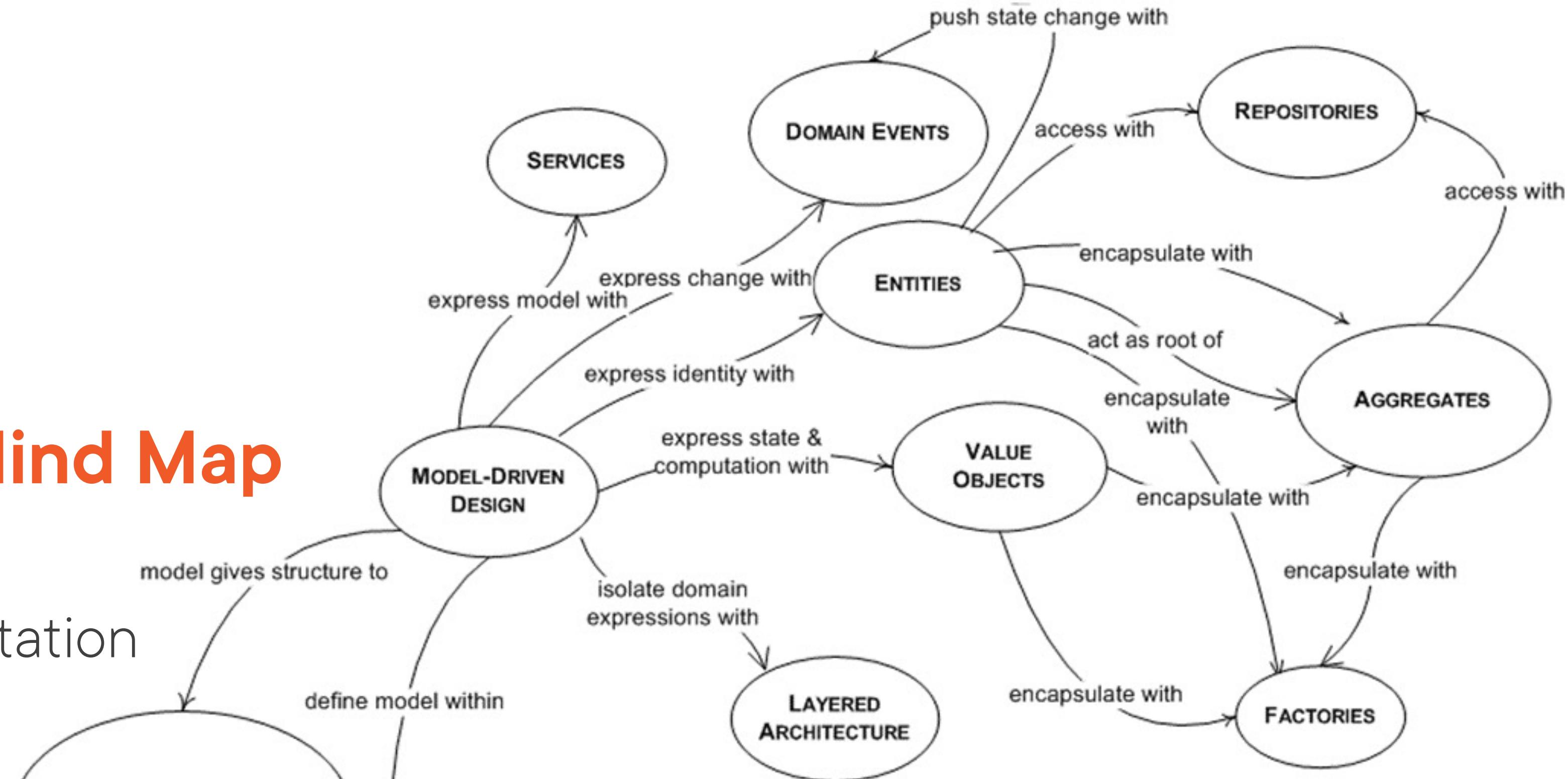
DDD Mind Map

Modeling



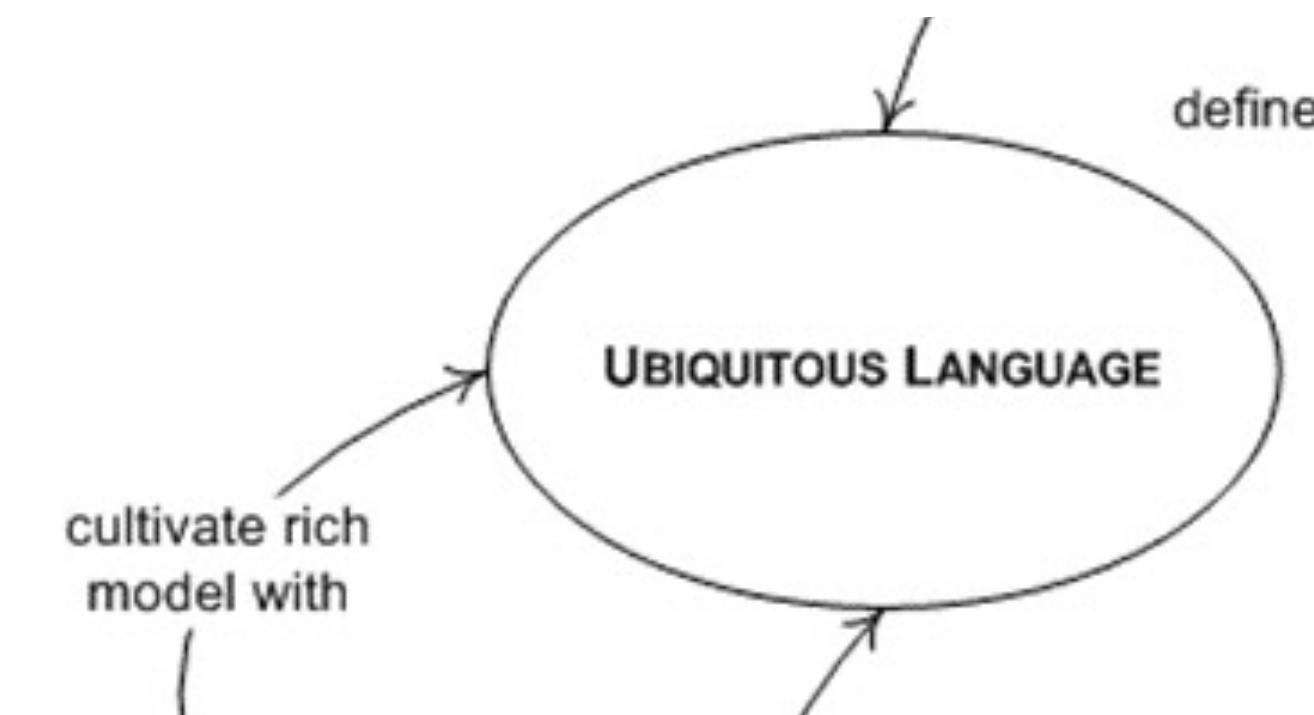
DDD Mind Map

Software
Implementation



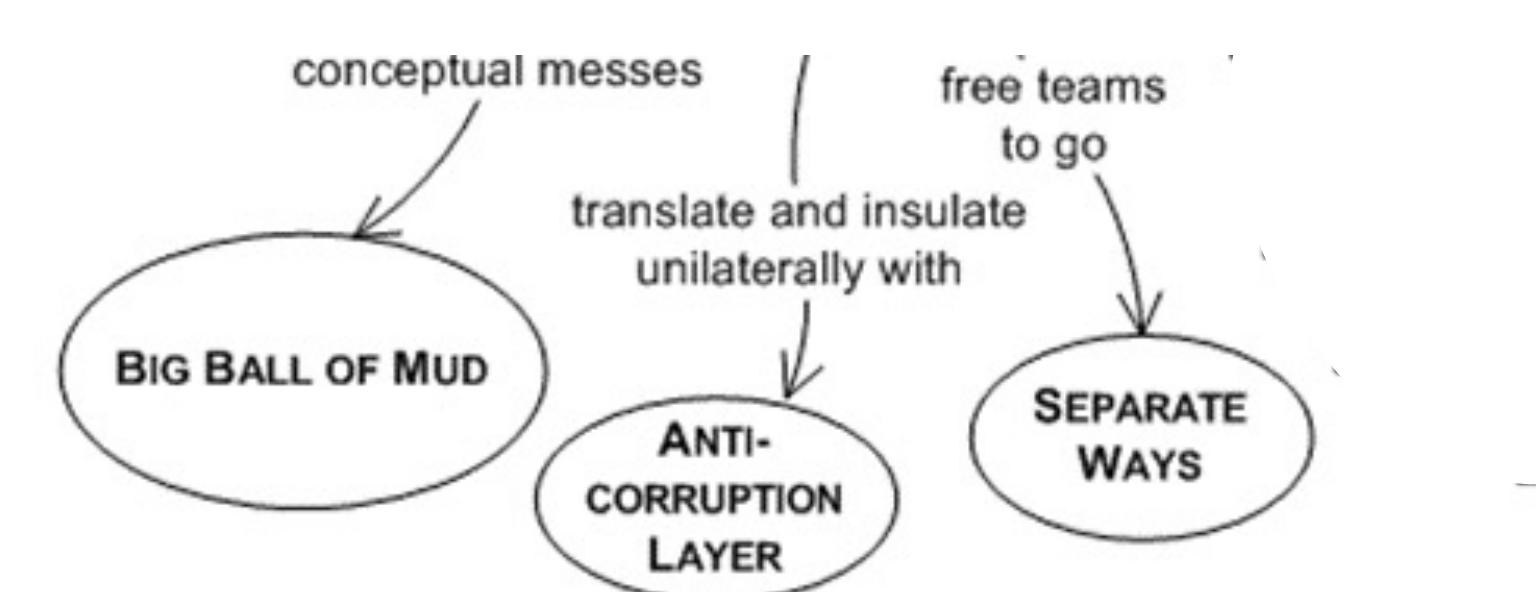
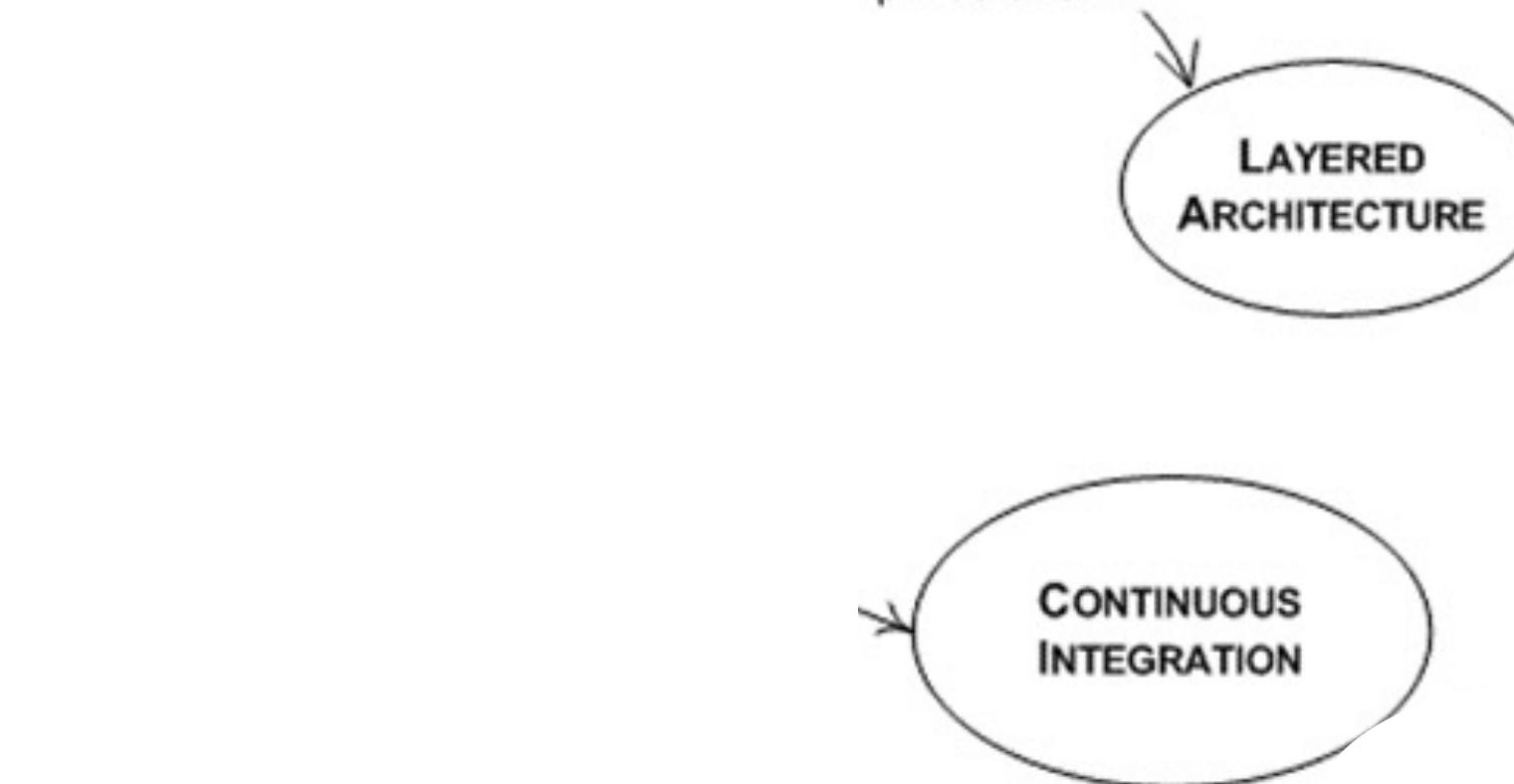
DDD Mind Map

Communication



DDD Mind Map

Development
Process

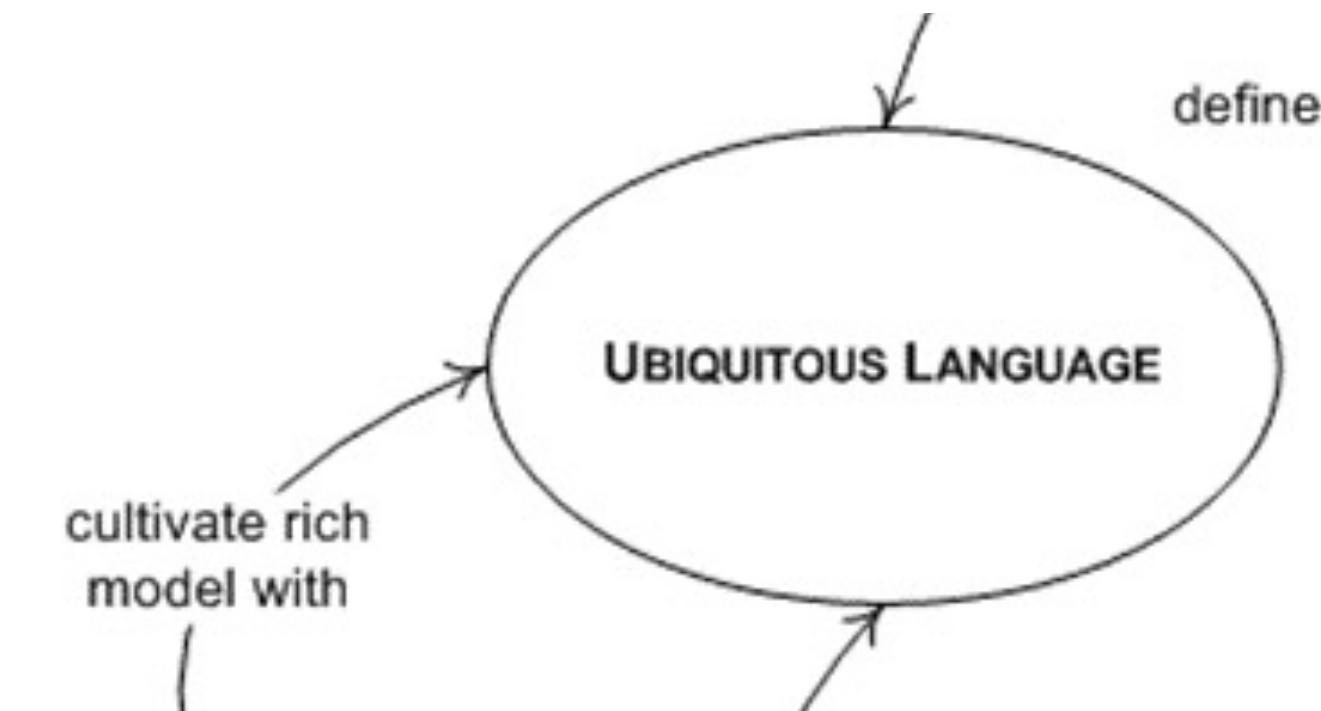


DDD Mind Map



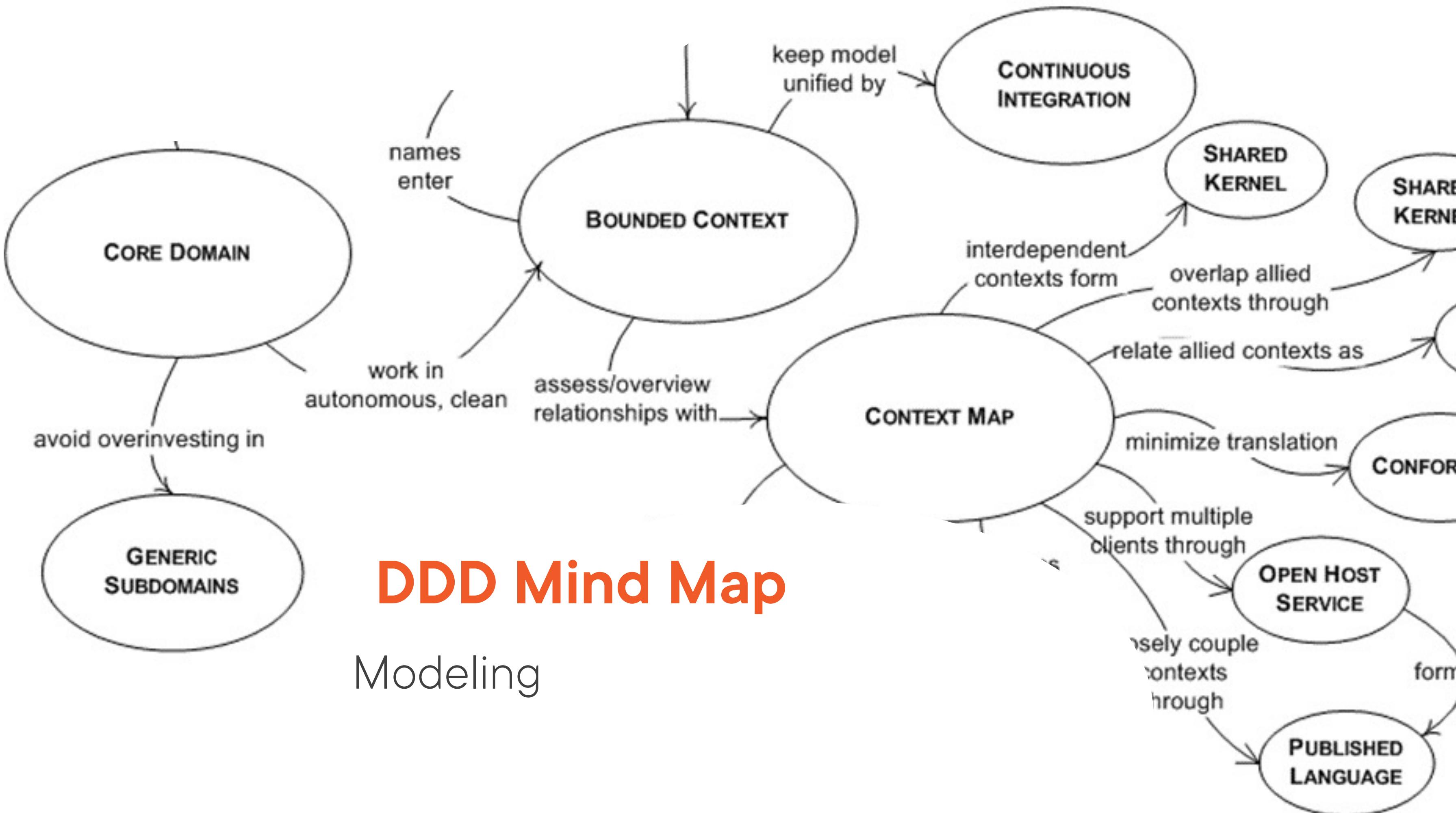
DDD Mind Map

Communication

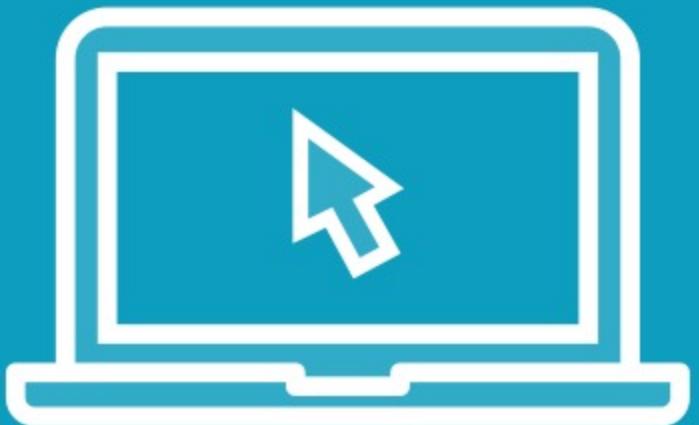


DDD Mind Map





Demo

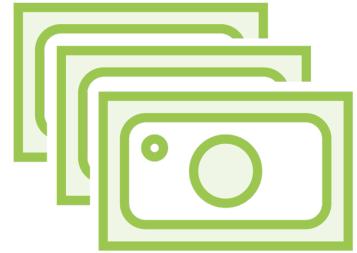


Introducing Our Sample Application

The Software Does More than Scheduling



Staff



Accounting



Client and patient records



Visit records



Appointment scheduling

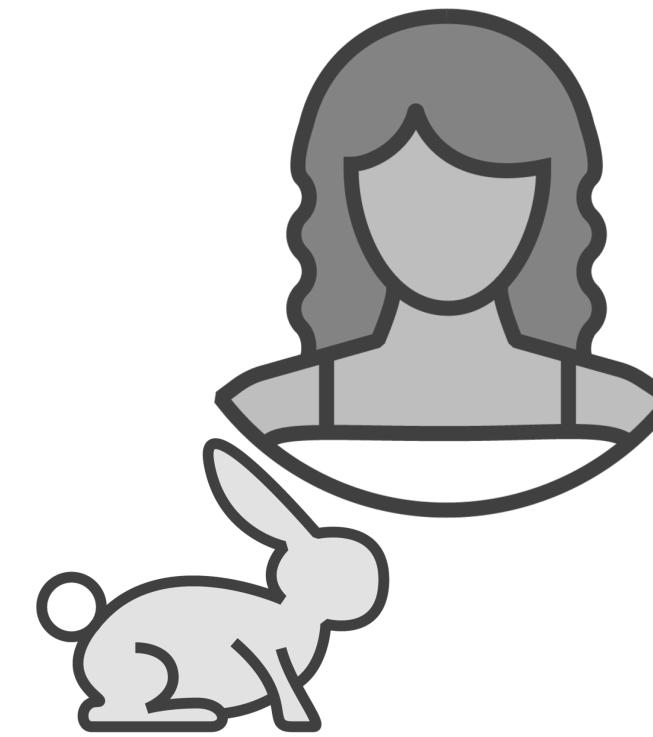


Sales

Some Logic is Complex, Some is Simple CRUD



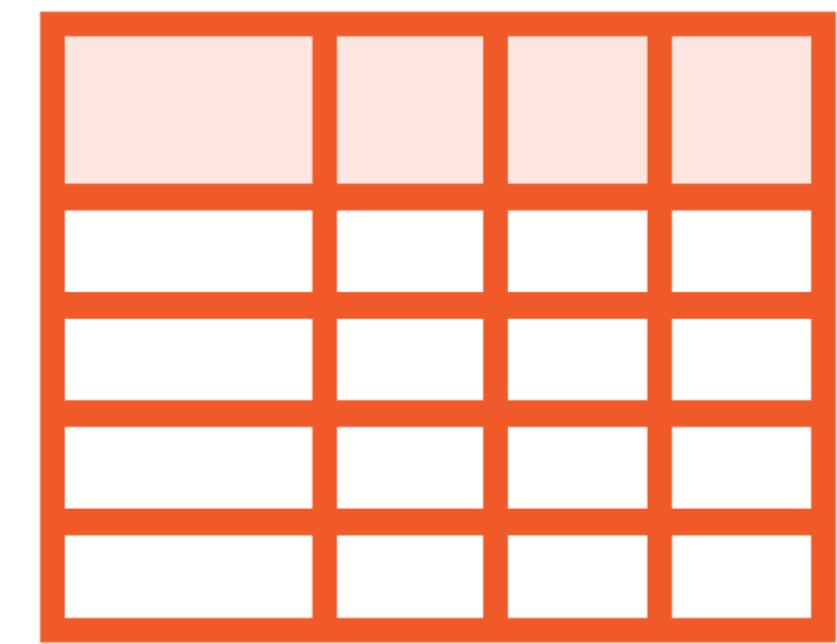
**Appointment
scheduling is
complex!**



**Manage client &
patient data**



**Manage staff
and facilities**



**Other simple
data
management**



Pluralsight Course

Creating N-Tier Applications in C#, Part 1 & 2

Steve Smith

Exploring the Sample App's High-Level Structure

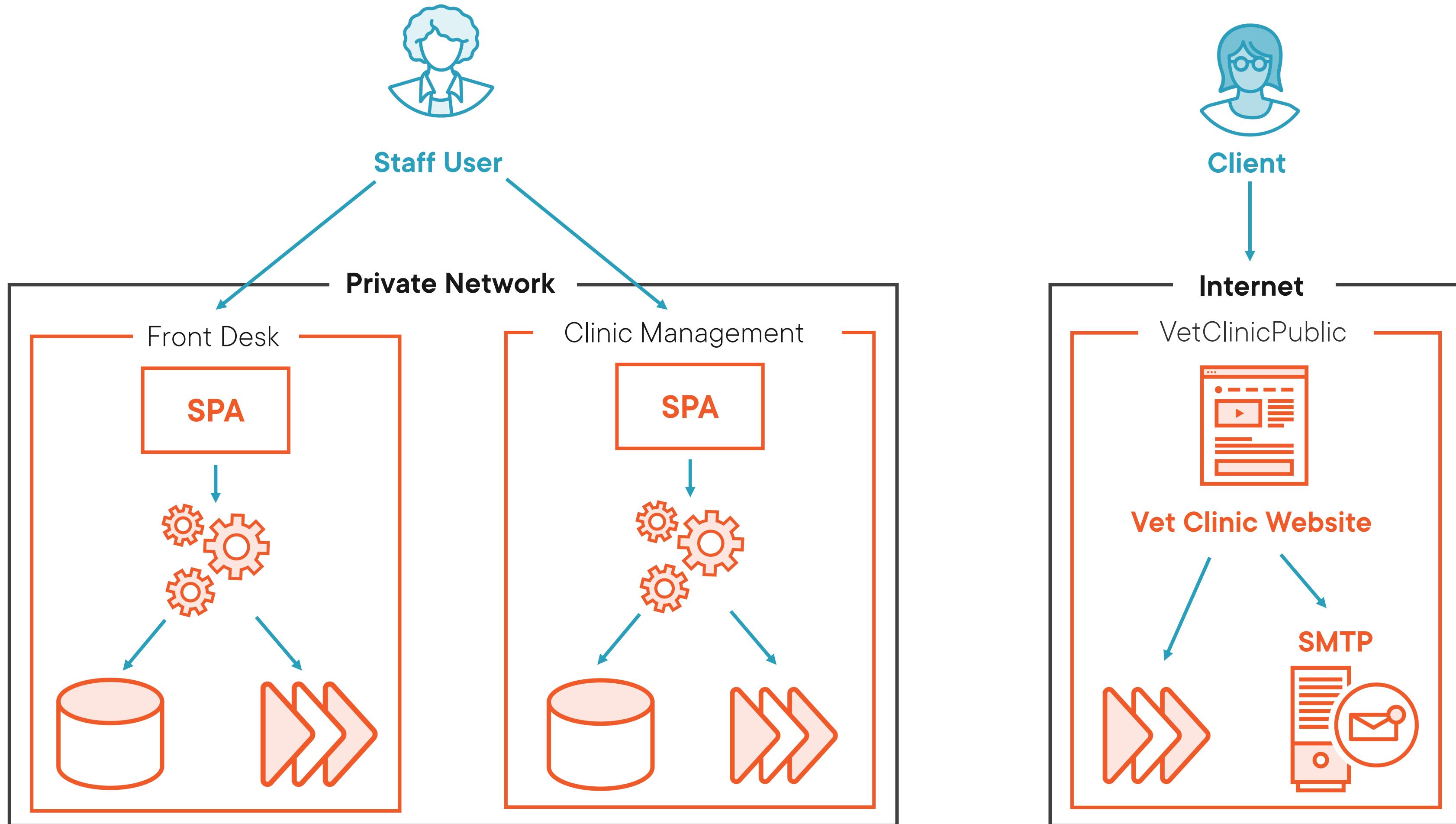
Our Distributed Web Apps

**Single Page App (SPA)
using
Blazor Web Assembly**

**APIs
built with
ASP.NET Core**

Front End

Back End



DDD Mind Map



Key Takeaways



Solving problems, not just coding

Collaborate with domain experts

Break domain into smaller units

Efficient and effective path to success

Teamwork!

Up Next:
Modeling Problems in Software

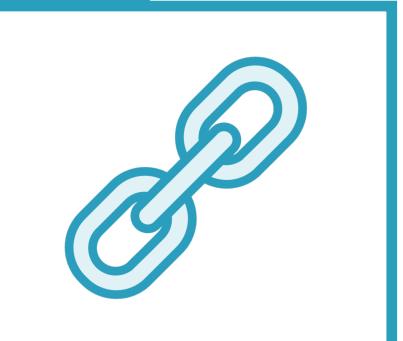
Resources Referenced in this Module



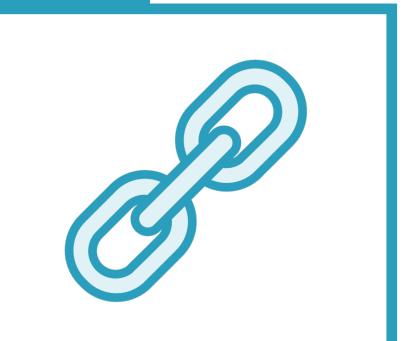
Domain-Driven Design, Eric Evans, 2003, Addison-Wesley Professional amzn.to/3k7g948



Patterns, Principles, and Practices of Domain-Driven Design
Scott Millett, Nick Tune, 2015, Wrox amzn.to/2ZyaNH8



Domain-Driven Design, The First 15 Years, Essays from the community
leanpub.com/ddd_first_15_years

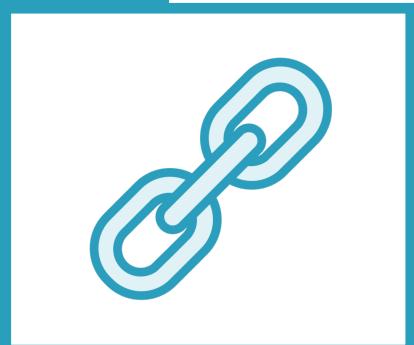


VirtualDDD Meetup with recorded sessions: virtualddd.com

Resources Referenced in this Module



Getting to DDD: Pragmatic or Principled
Julie Lerman, KanDDDinsky 2018
youtu.be/CPd2BVR3hEM



Pluralsight DDD Fundamentals Sample for this course
github.com/ardalis/pluralsight-ddd-fundamentals

Domain-Driven Design Fundamentals

Introducing Domain-Driven Design



Steve Smith
Force Multiplier for
Dev Teams
[@ardalis](https://twitter.com/ardalis) ardalis.com



Julie Lerman
Software coach,
DDD Champion
[@julielerman](https://twitter.com/julielerman) thedatafarm.com