Domain Driven Design

Distillation - Chapter 15

DDD-Meetup Cincinnati
2019-09-15

https://github.com/mwindholtz/presentations/tree/master/DDD

Domain-driven design (DDD)

An approach to software development

- for complex needs
- by connecting the implementation
- to an evolving model.

The term was coined by Eric Evans

A Short Review or Overview …

Meeting Topic Areas

1. **Strategic Patterns**
    - Context Maps, Sub Domains, etc
2. **Tactical Patterns**
    - DomainEvents, Aggregates, etc
3. **Communication Tips**
    - Whirlpool, Knowledge Crunching, Event Storming
4. **Code Examples**
    - Build In Your Own Language: The Cargo Shipping Example

When to Apply Domain Design

- **For <u>Simple</u> systems**
  - No worries. It fits inside a person's head.
- **For <u>Medium</u> systems**
  - No worries. Hire smart people so that ..
  - It fits inside a person's head. Oh and write loads of *Documentation! ***
- **For <u>Complex</u> systems**
  - Starting is ok. It initially still fits inside a person's head.
  - Then Documents help a while
  - But As It Grows …

** Documentation has an unknown expiration date.
And may be wrong to begin with.
Other restrictions may apply.

Typical “Agile” project progression

- Feature story
- Design, design, design *:-)*
- Feature story, Feature story
- Design. *:-/*
- Feature story, Feature story, Feature story, Feature story, Feature story, Feature story *:-(*
- Feature story, Feature story, Feature story, Feature story, Feature story, Feature story *:-o*

Code Structure:
Big Ball Of Mud

---

http://www.laputan.org/mud/

Process Diagnosis:
Featureatitis

Software Craftsmanship — IS NOT ENOUGH —

- Refactoring
- Better names
- Test Driven Design
- Continuous Single Integration
- Something is still missing

Kent Beck @KentBeck May 28

Software development is a leaky rowboat. Behavior changes are rowing--making progress toward a goal, however dimly glimpsed. Structure changes are bailing--not progress in a measurable sense but absolutely necessary for progress.

DDD Europe @ddd_eu 5d

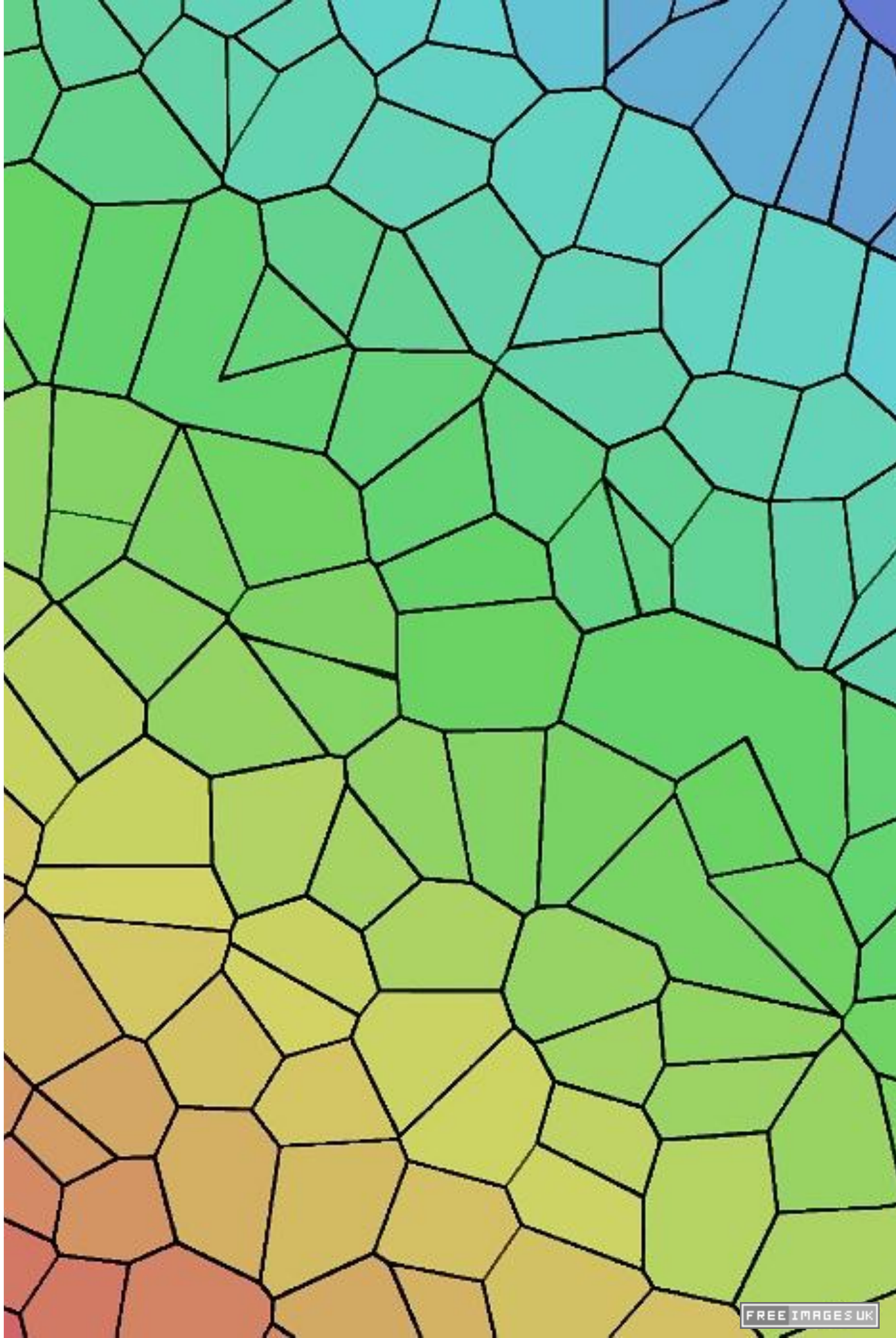
"No refactoring without remodelling. Clean Code by itself cannot save a rotten model." From "Technical debt isn't technical" by Einar Høst @einarwh at #DDDEU 2019 buff.ly/2WGYyss

17 35

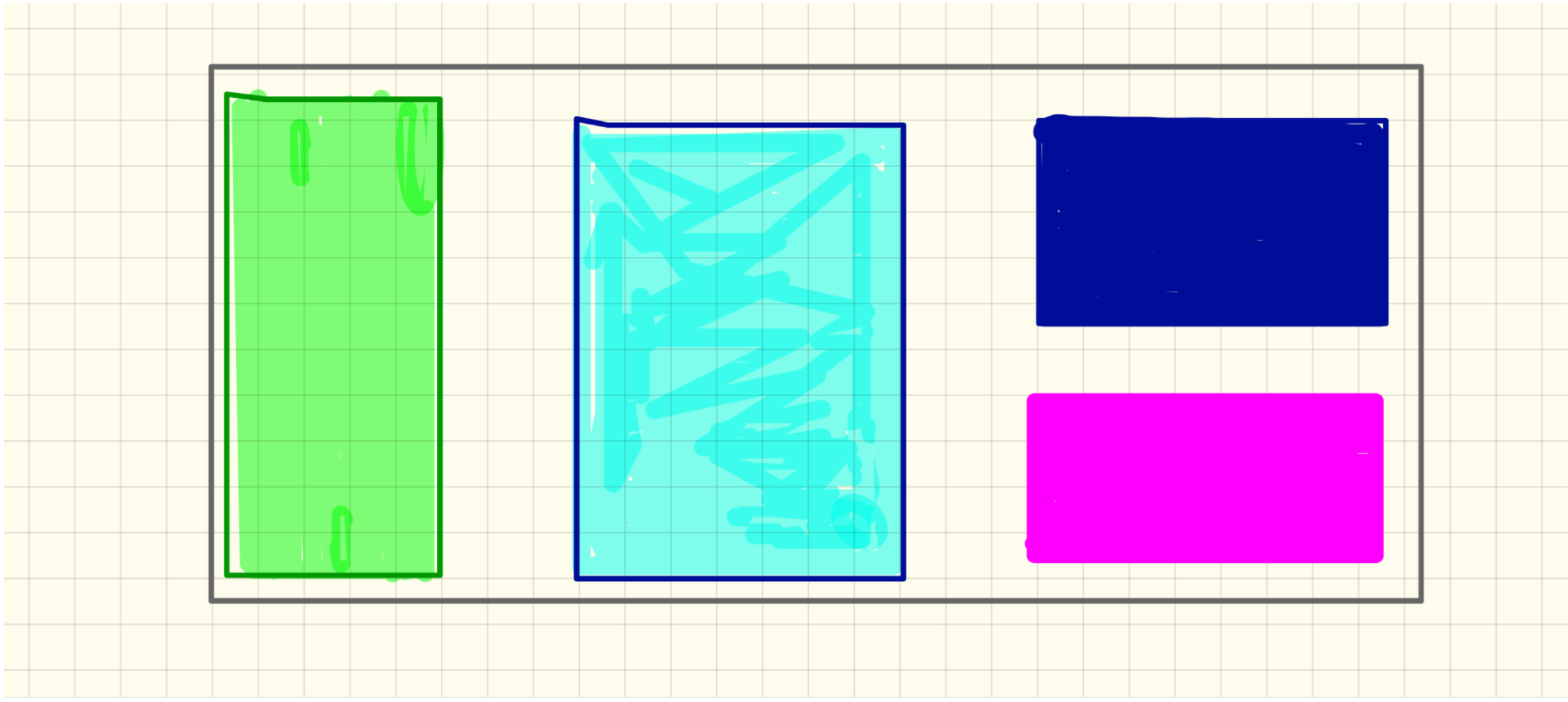
DDD Europe Videos 1999

“Doing” DDD

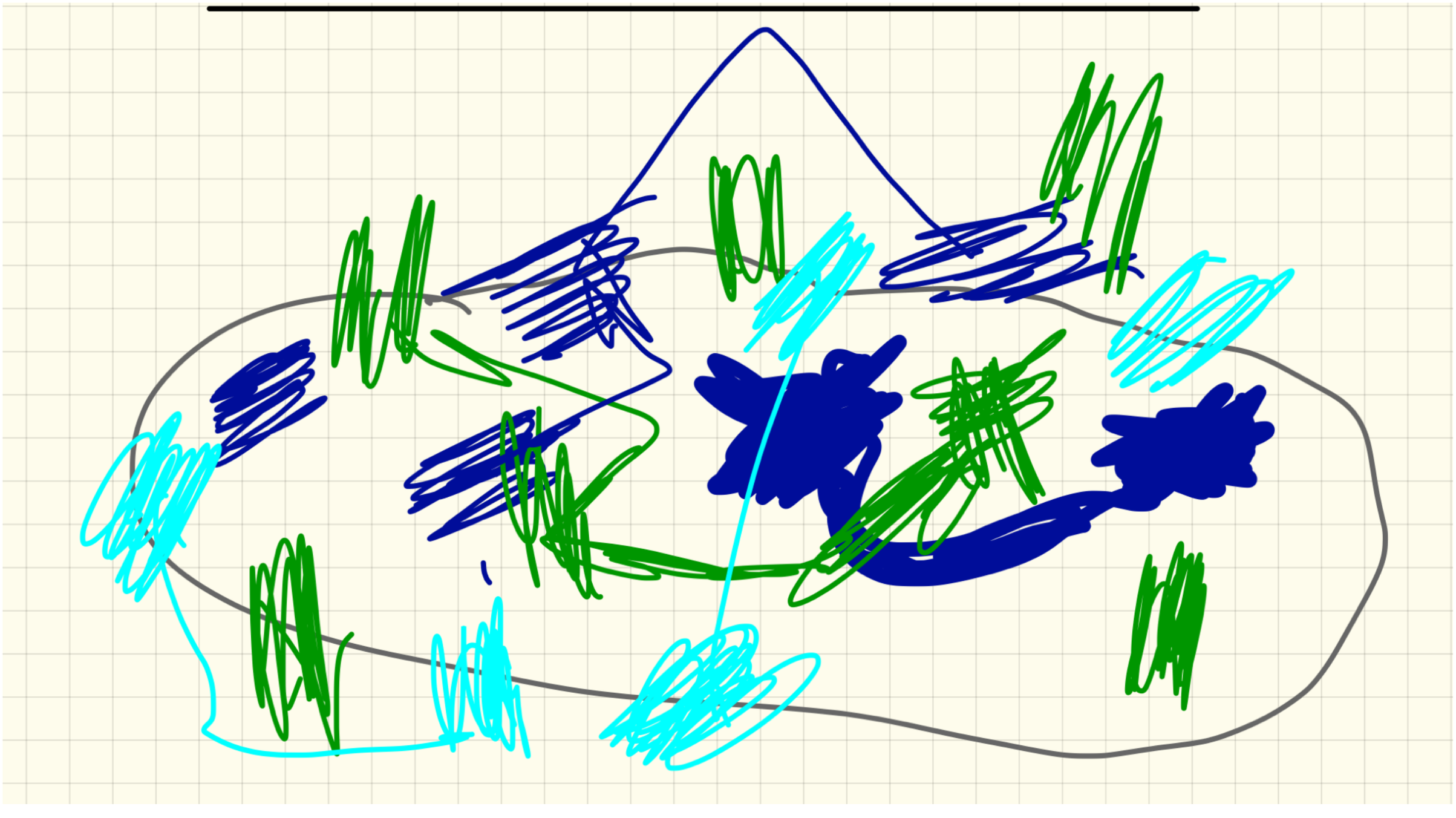
- Find ***Bounded Contexts***
- Build ***Context Map***
- Focus on ***Core Domain***
- Apply ***Building Blocks***
- Engage Domain Experts to build ***Ubiquitous Language***
- Repeat … and Revisit :||



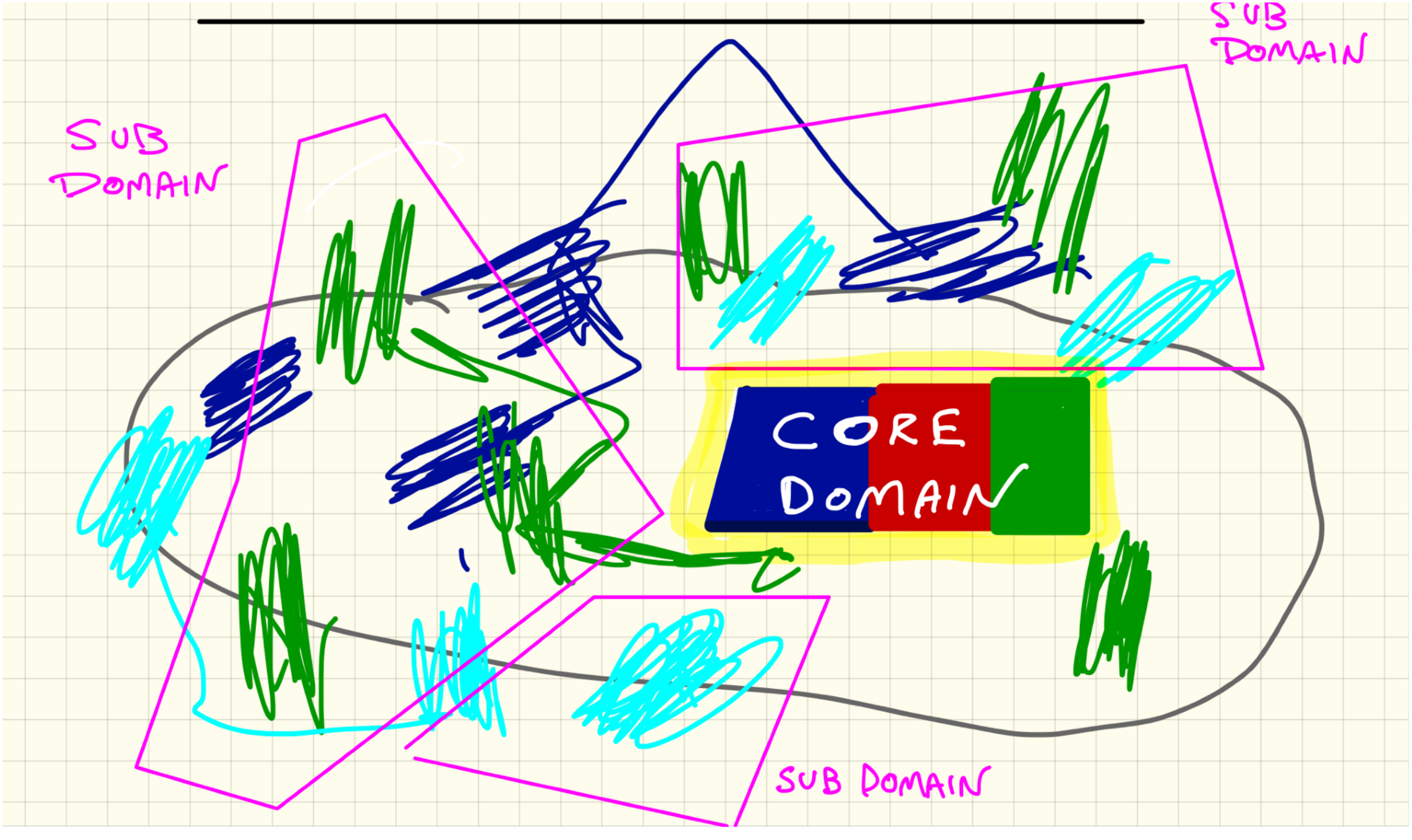
Application, as we Imagine it



Application, in Reality



Apply Strategic Distillation



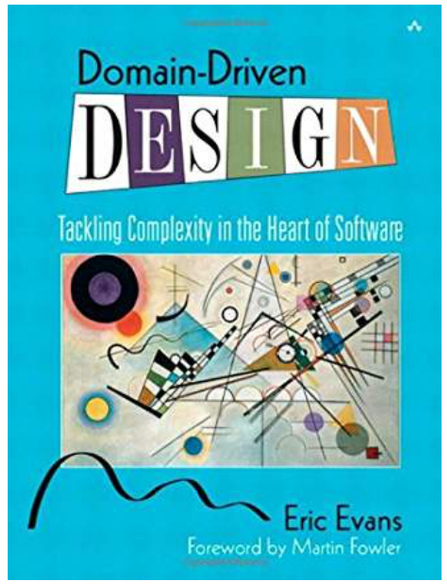
Domain

- **A sphere of knowledge, influence, or activity.**

- The subject area to which the user applies a program is the domain of the software.

Why Now ?

- **Domain Driven Design, “Big Blue Book”, 2003**
  - Tactical Patterns get most of the attention



- **Micro-Services, 2012**
  - Bounded Context and Context Mapping for organization
  - Content of each Micro-Service
  - Relationships among Micro-Services

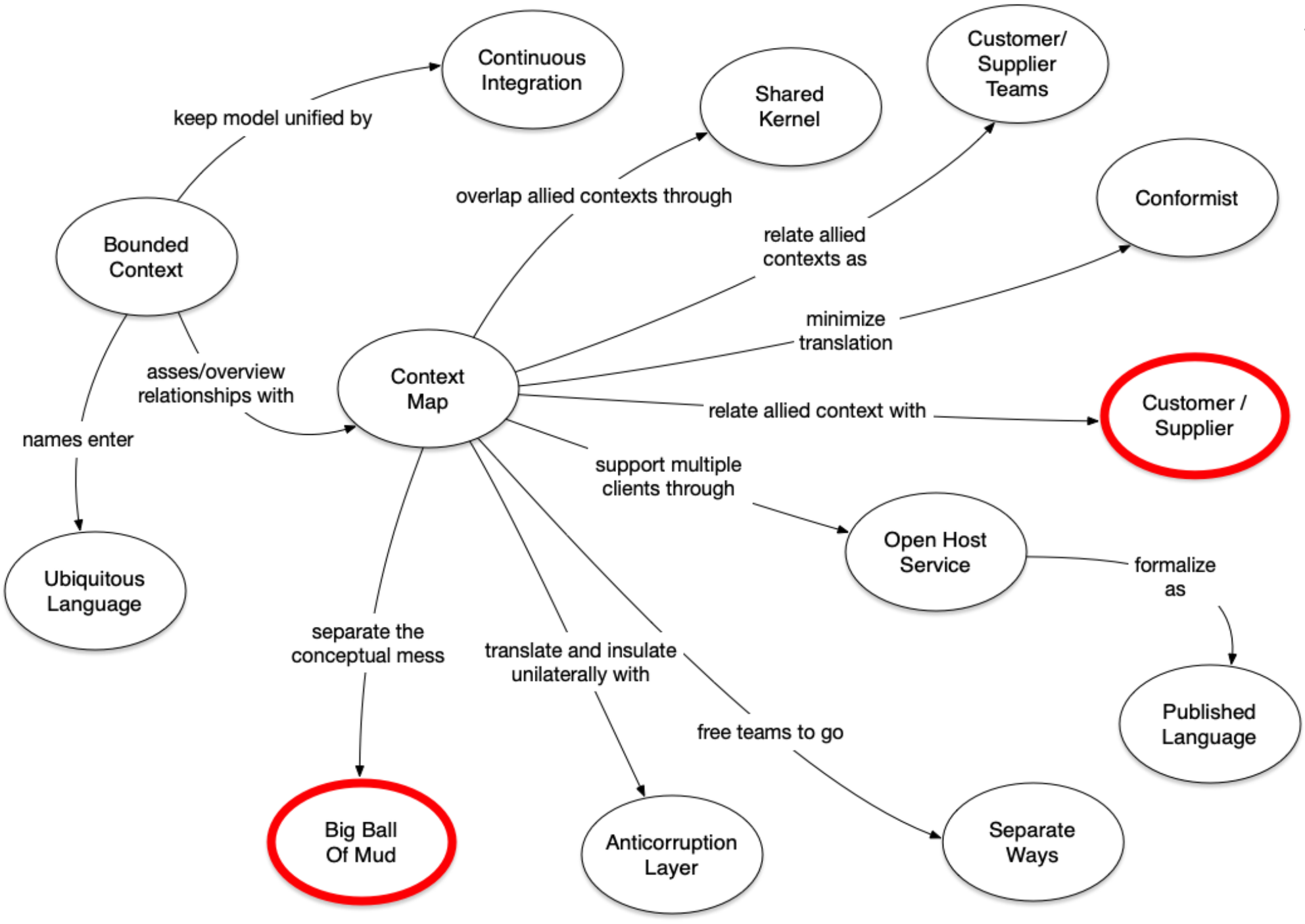
Why is DDD Difficult to Explain?

- **Since the *Problem* is Complex and Subtle**
- **The *Solution* is also Complex and Subtle**
  - Difficult to *scale down* into examples
- **Large Vocabulary of Interrelated Patterns**
  - Pattern Languages

Building Blocks

- Layered Architecture
- Value Objects
- Entities
- Factories
- Repositories
- **Aggregates**
- **Services**
- **Domain Events**

Strategic Patterns



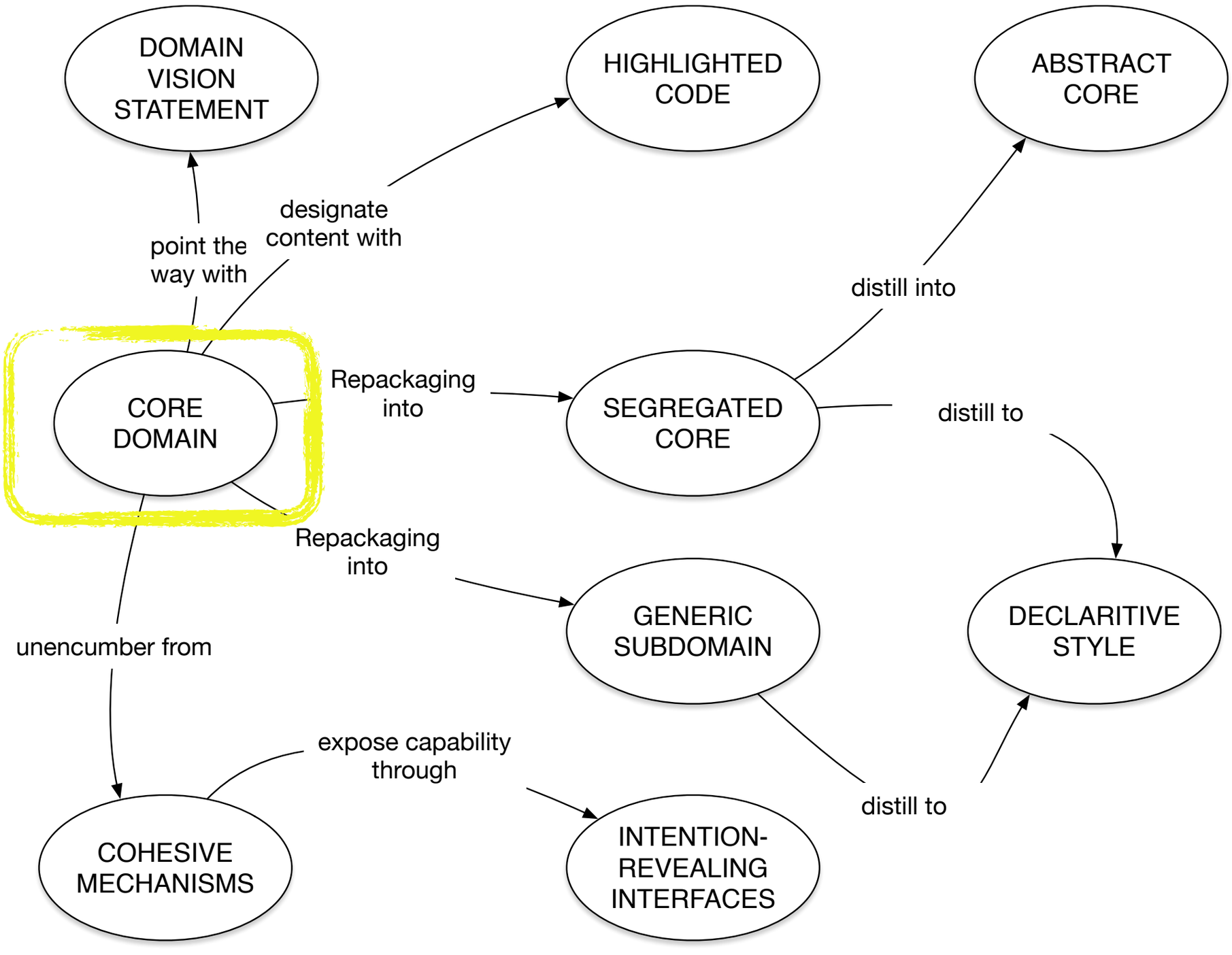
Strategic Patterns

CH 15 - DISTILLATION

Finding - Separating
- The Core Domain

Wikimedia Commons

Strategic Distillation



CORE DOMAIN

- A System that is Hard to understand is hard to change
- Not all parts of a system are going to be equally refined
- Skilled developers are drawn to the new technical areas
- The specialized core less experienced devs
- Boil down the model.
- Apply top talent to the CORE DOMAIN

GENERIC SUBDOMAIN

- Some Parts of Model add complexity without capturing special knowledge
- Don’t Clutter the CORE DOMAIN
- Identify Cohesive subdomains that are not the target of the project
- Examples: Email Delivery, User Management, Billing

DOMAIN VISION STATEMENT

- Write a short (one page) description of the CORE DOMAIN
- Leave out technical details, UI, DB, Tooling, etc
-

HIGHLIGHTED CORE

- Write a short 3-7 page document
- Describe the main MODEL elements and their interactions

COHESIVE MECHANISMS

- Some parts of the MODEL get complex
  - example: Algorithms
- CM is part of the MODEL but a complex part that internally works together
- Partition those parts in COHESIVE MECHANISMS
- Provide an INTENTION-REVEALING INTERFACE

SEGREGATED CORE

- ?

ABSTRACT CORE

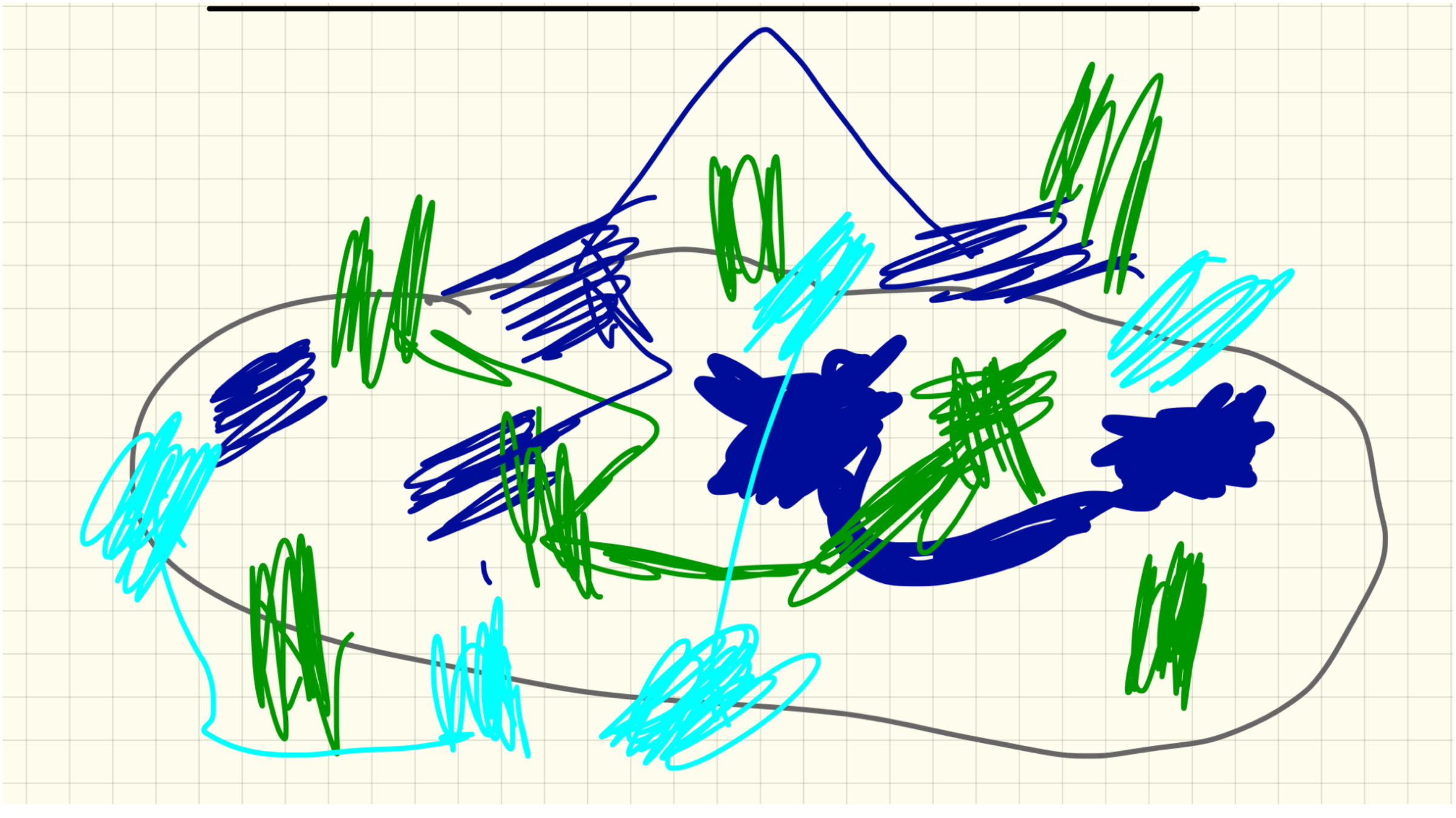
- Identify the most fundamental concepts in the model and create distinct abstract interfaces for them.

- Create interactions between these abstract model elements

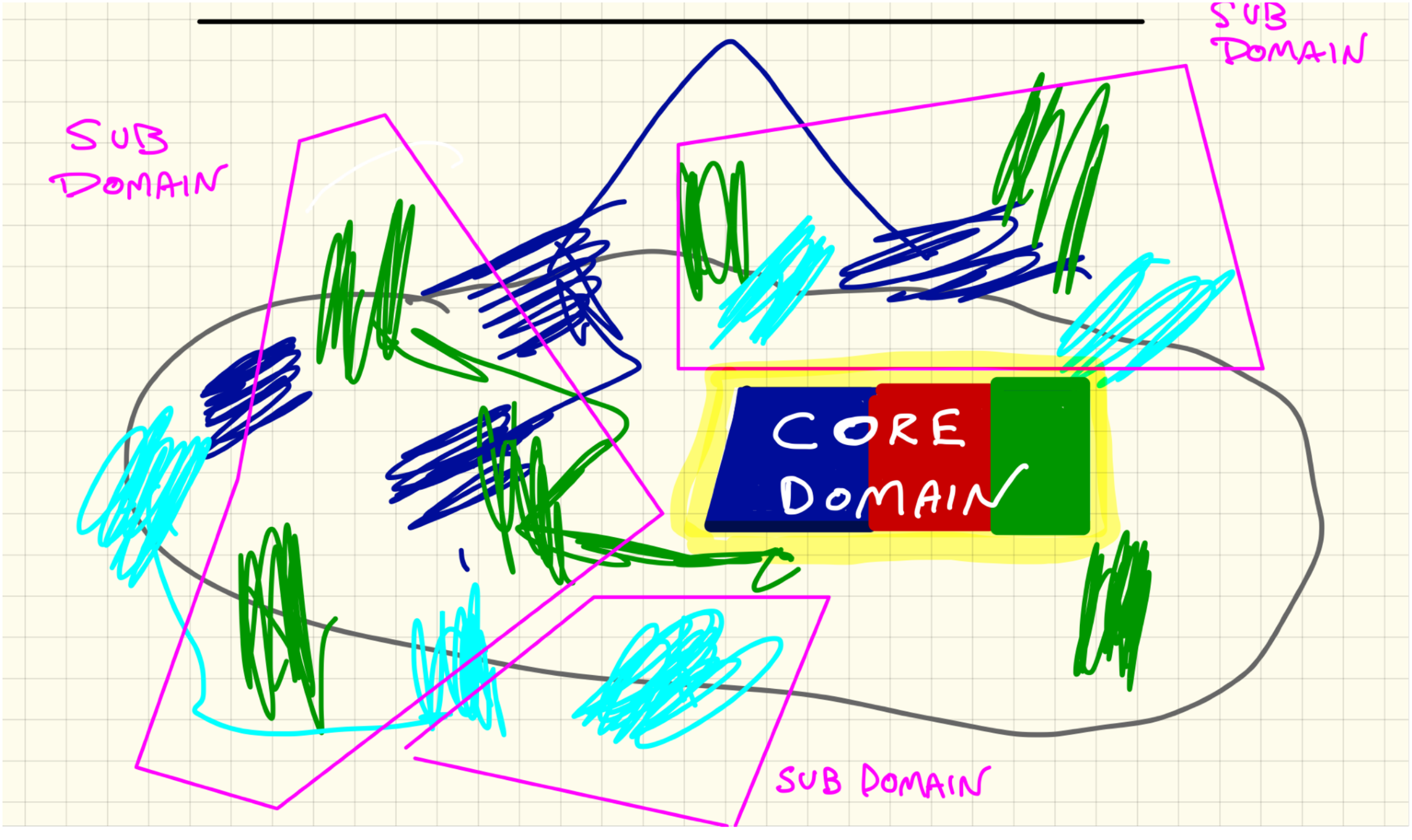
DECLARITIVE STYLE p.270

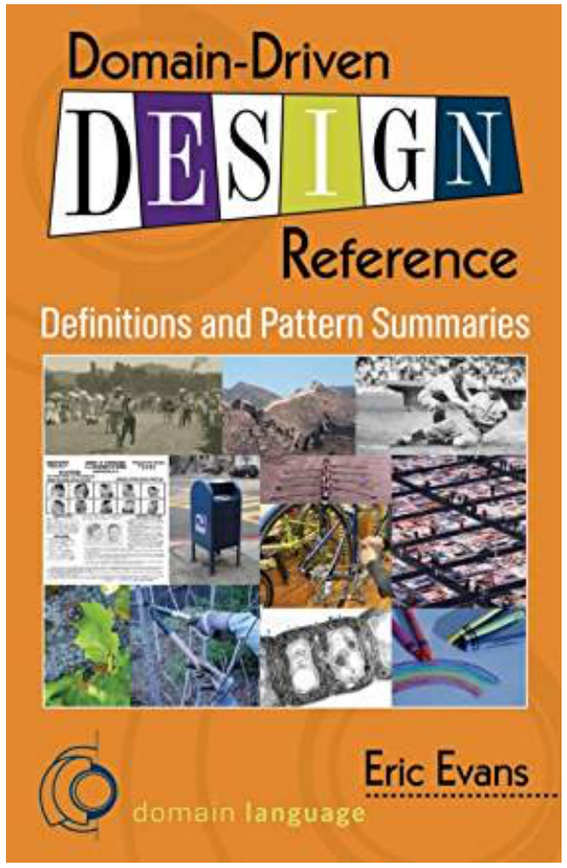
- Code Generator, Framework Generator
- Specifications, Rules,
- Metalevel, macros, templates(?)
- Example:

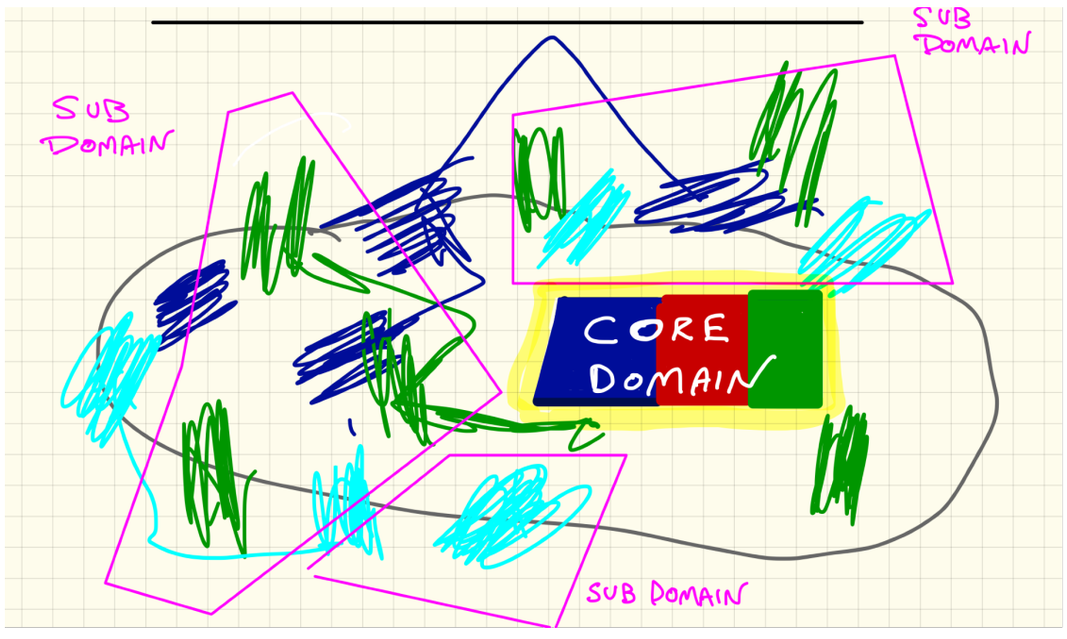
Application, in Reality

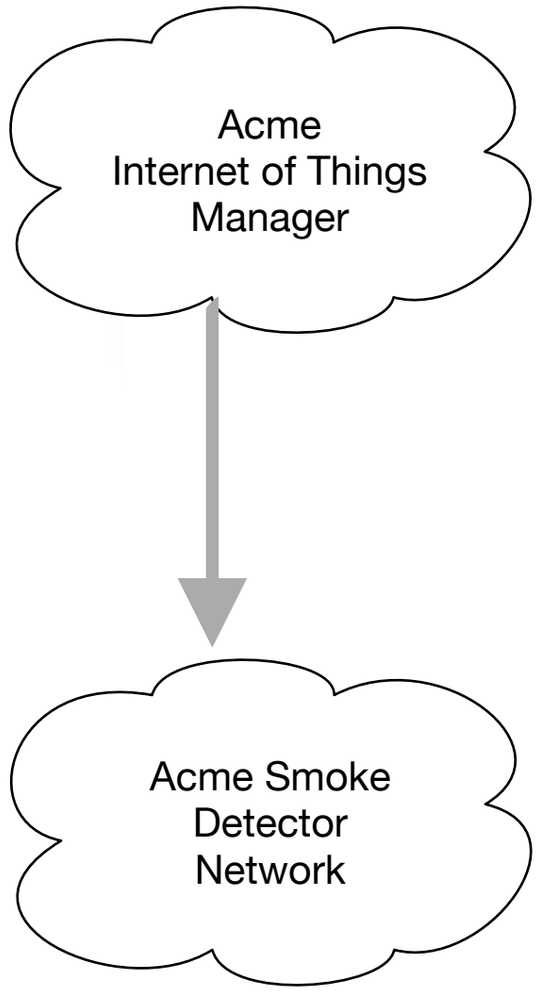


Apply Strategic Distillation




Domain-Driven
DESIGN
Reference
Definitions and Pattern Summaries
Eric Evans
domain language


SUB DOMAIN
SUB DOMAIN
CORE DOMAIN
SUB DOMAIN


Acme
Internet of Things
Manager
Acme Smoke
Detector
Network

The End