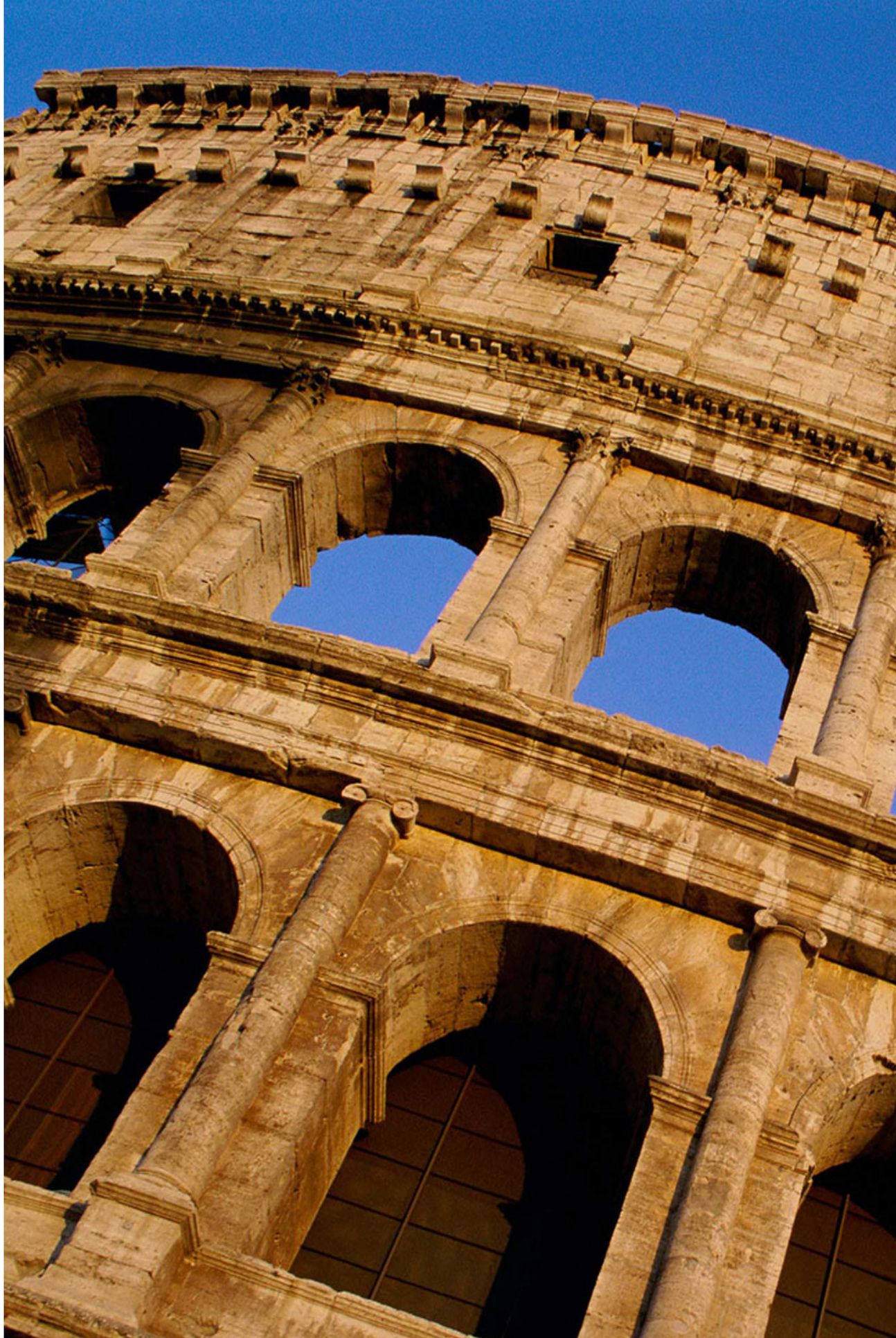
Domain Driven Design

Introduction

DDD-Meetup Cincinnati
2019-06-25



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

Areas

1. **Strategic Patterns**
    - Context Maps, Sub Domains, etc
2. **Tactical Patterns**
    - DomainEvents, Aggregates, Factories, etc
3. **Communication Tips**
    - Whirlpool, Knowledge Crunching, Event Storming

12 Original Practices of Extreme Programming

1. Planning Game
2. Small Releases
3. Testing
4. On Site Customer
5. Continuous Integration
6. Refactoring
7. Pair Programming
8. Coding Standards
9. Collective Ownership
10. Simple Design
11. Sustainable Pace
12. 

"Guide all development with a simple shared story of how the whole system works" -XP explained p54

When to Apply Domain Design

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

Typical “Agile” project progression

- Feature story
- ***:-)*** Design, design, design
- Feature story, Feature story

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

Code Structure:
Big Ball Of Mud

---

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

Process Diagnosis:
Featureatitis

Software Craftsmanship

- Refactoring
- Better names
- Test Driven Design
- Continuous Single Integration
-

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

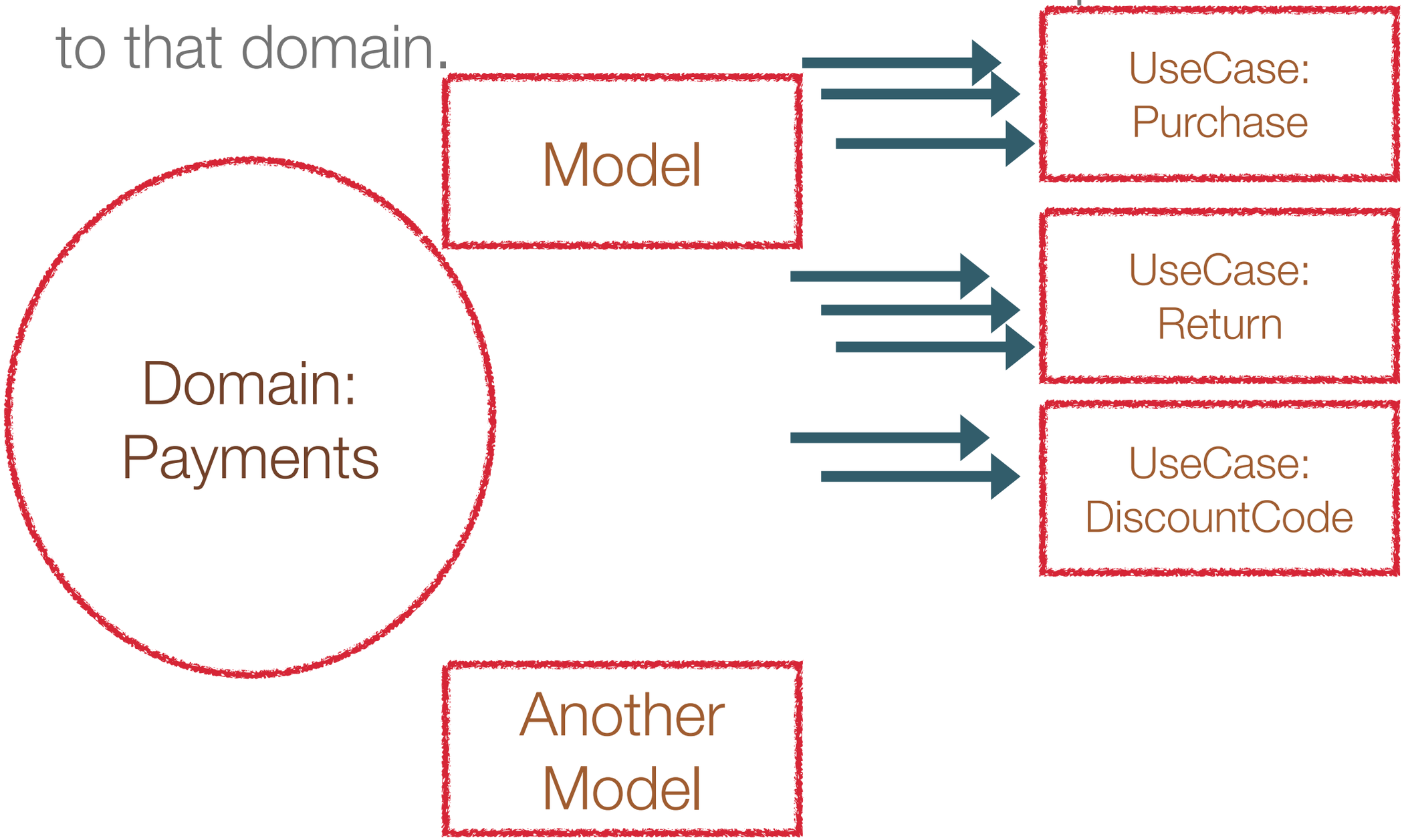
Domain

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

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

Model

- A system of abstractions that describes selected aspects of a domain and can be used to solver problems related to that domain.



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
  - What to put into a micro Service
  - How are the relationships among Micro-Services

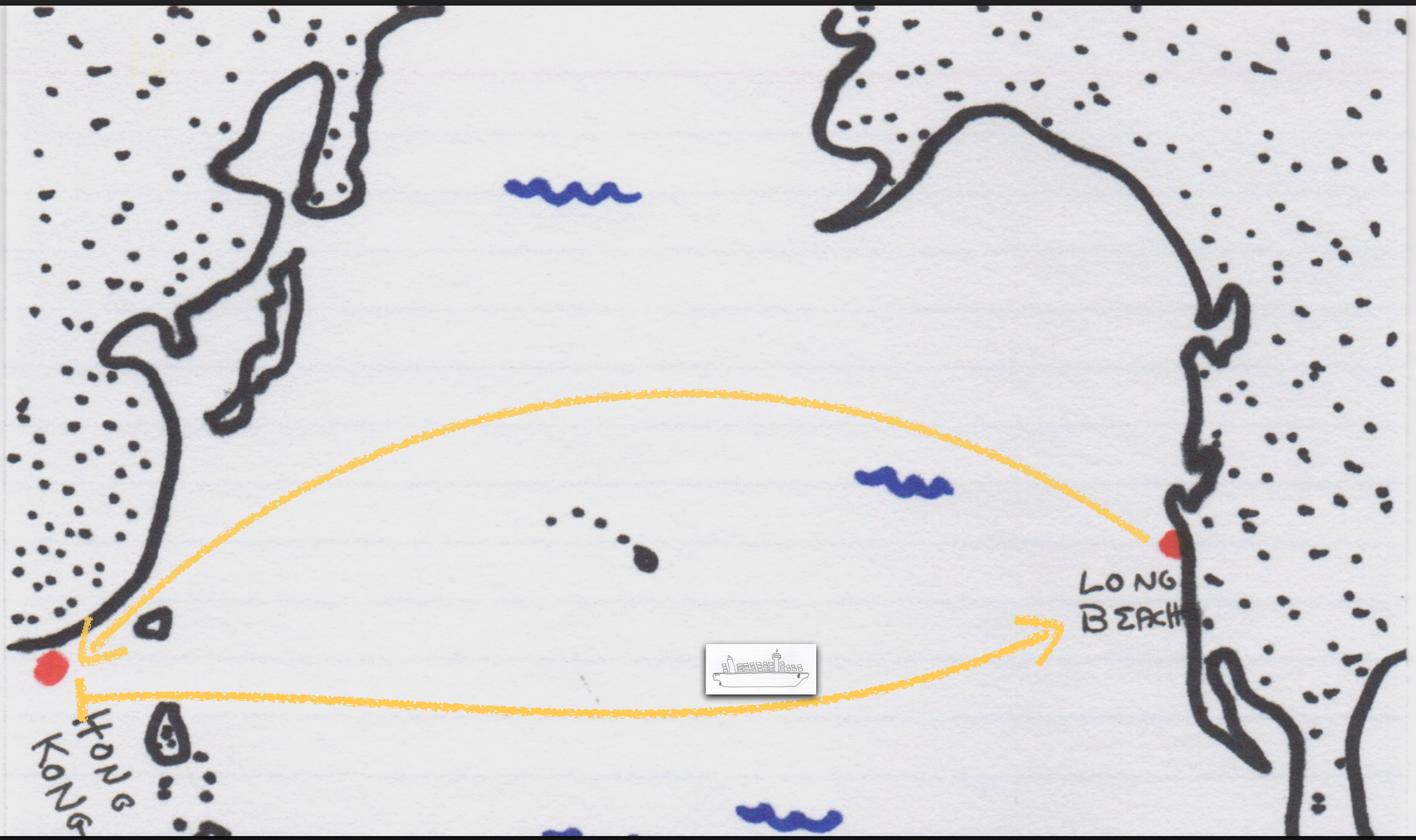
Why is it Difficult to Explain?

- **Complex, Subtle Solutions**
    - Difficult to *scale down* into examples
- **Large Vocabulary of Interrelated Patterns**
    - Pattern Language

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

CARGO SHIPPING



Discussion

- **Meetings will have a topic area**
  - Tactical, Strategic, Communication, Code
- **Topic**
  - Announced early as possible so folks can read-up
  - Motivated by common IT problem
- **Bring real life Modeling examples to the meetings**