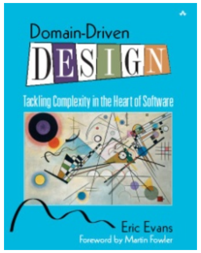
Recovering the Ability to Design when Surrounded by Messy Legacy Systems

Eric Evans

domainlanguage.com

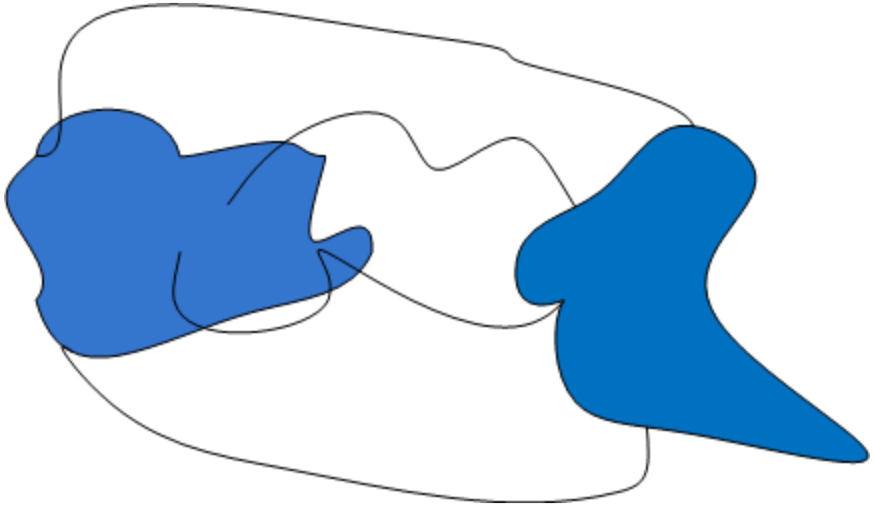
Twitter: @ericevans0 #DDDesign

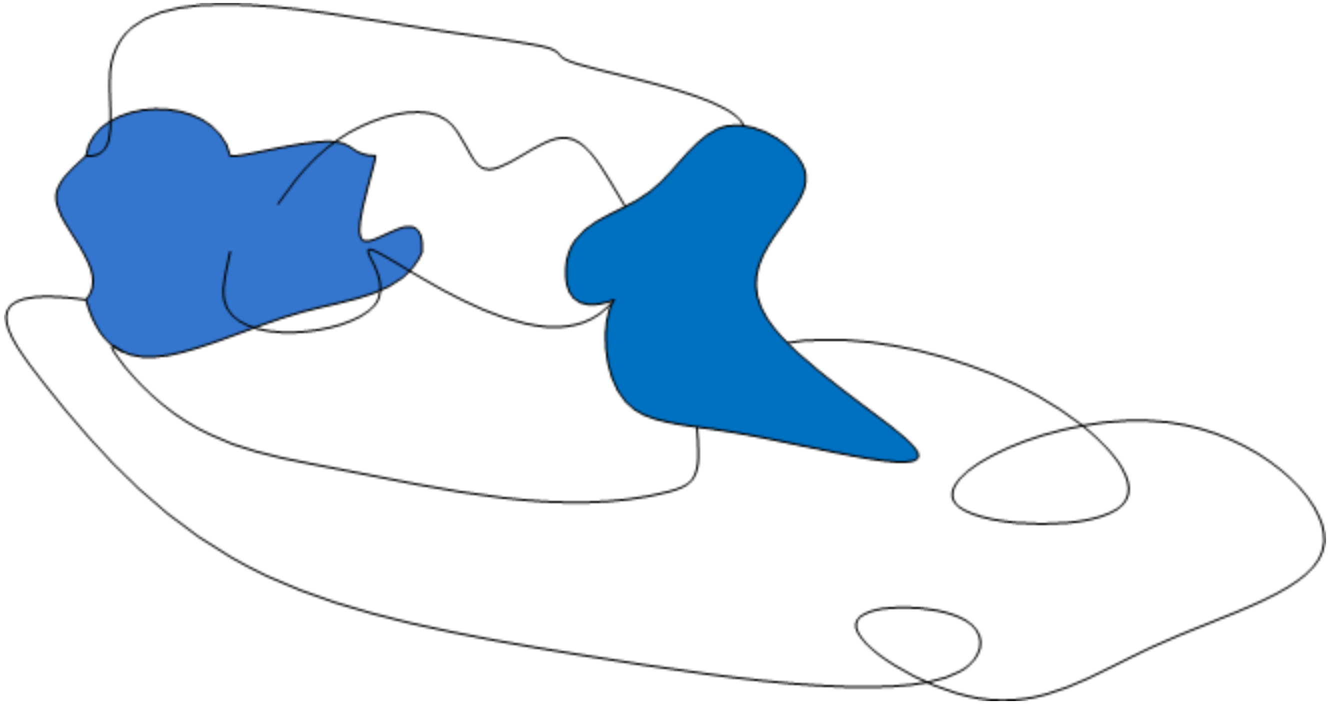
domain language


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

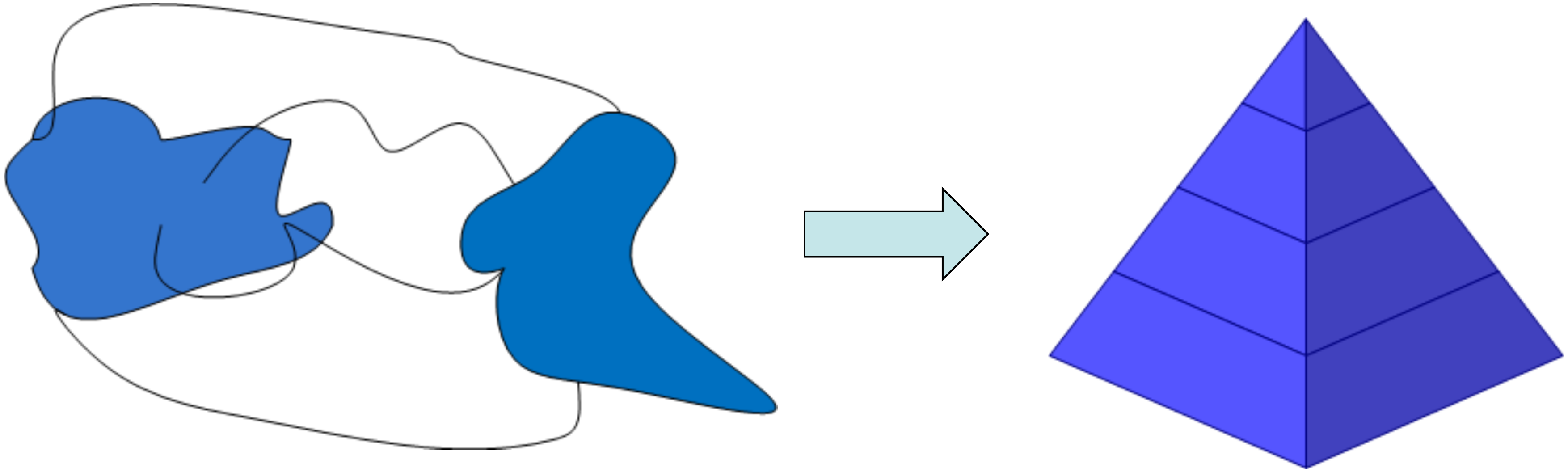
Not all of a large system will be well designed.

Part 1: Why We Lose the Ability to Design

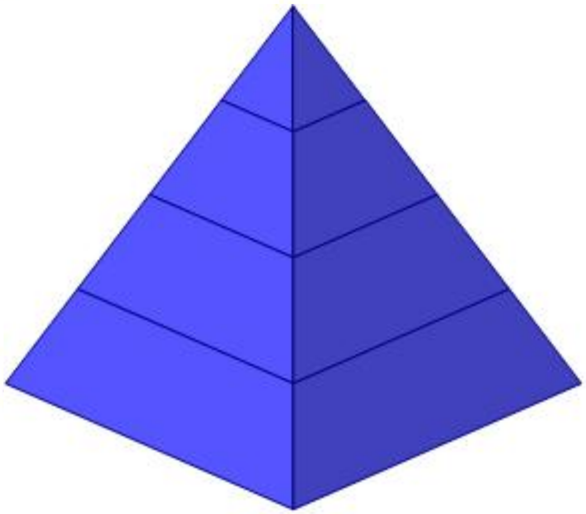




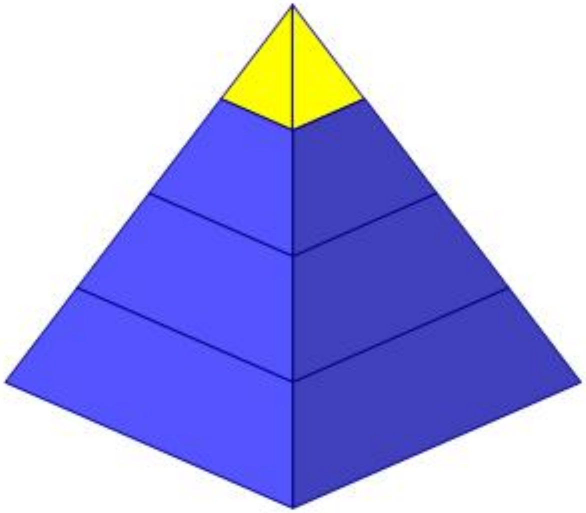
We Dream



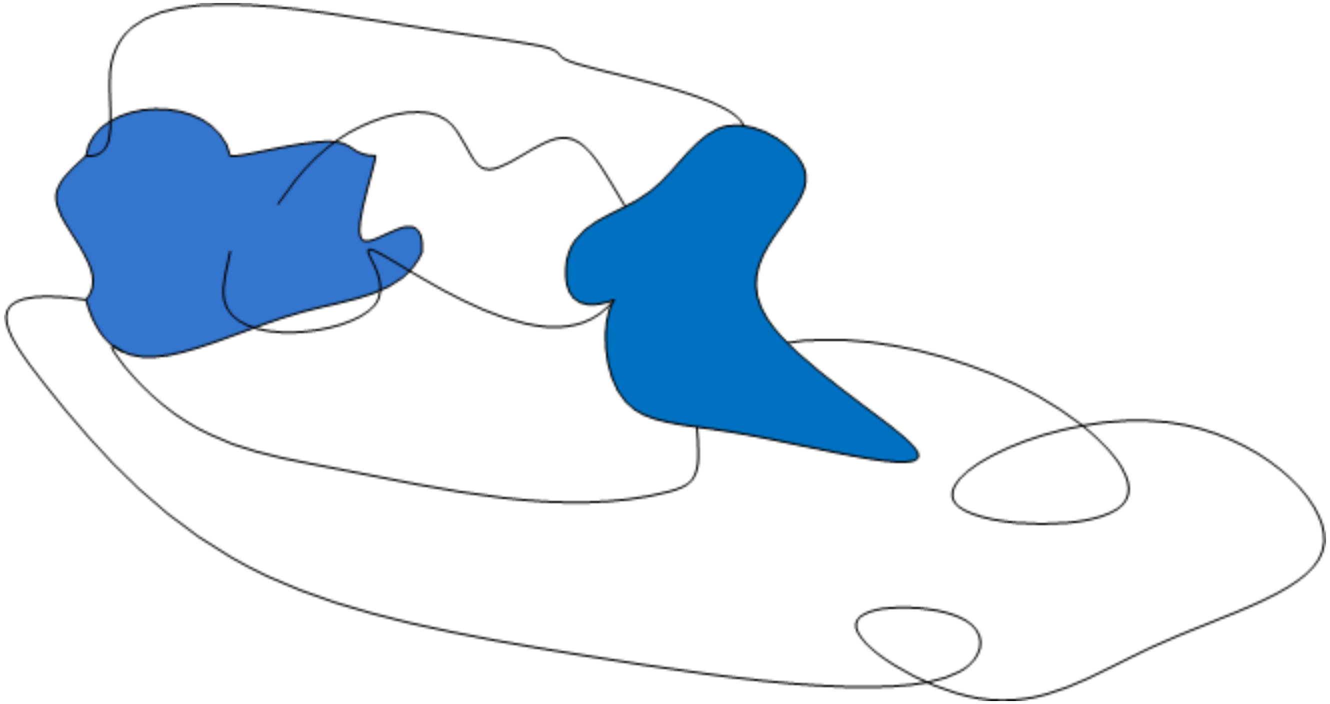
Where is Strategic Value?



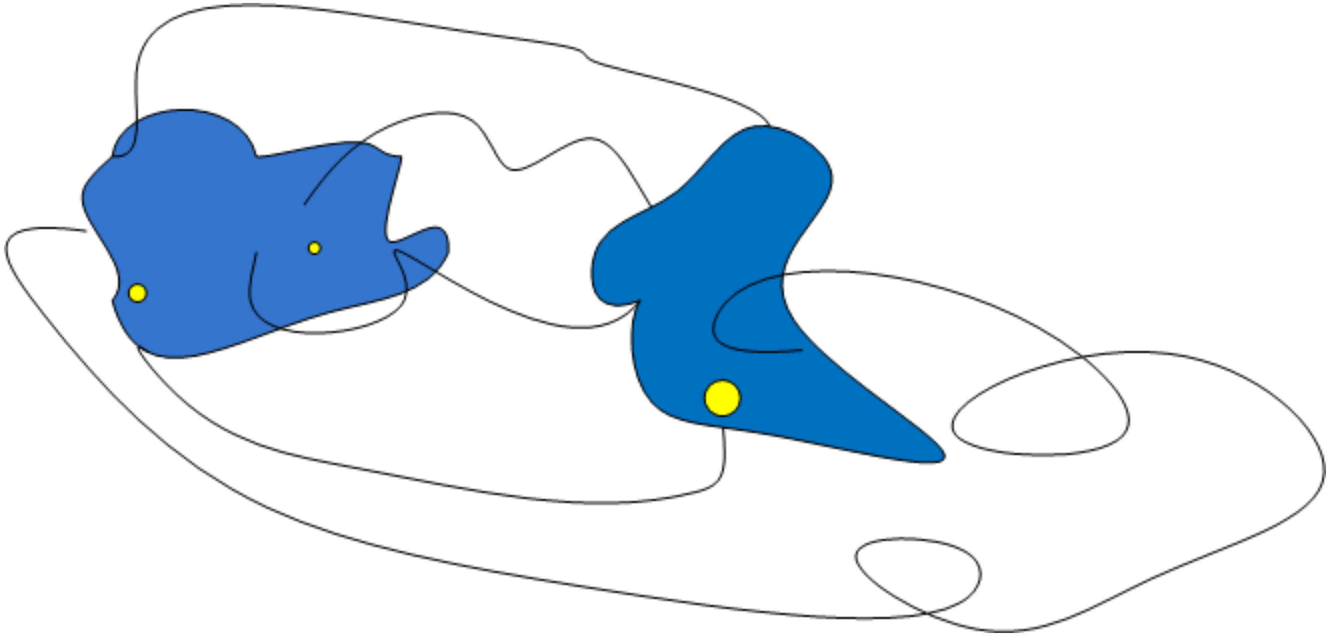
Where is Strategic Value?



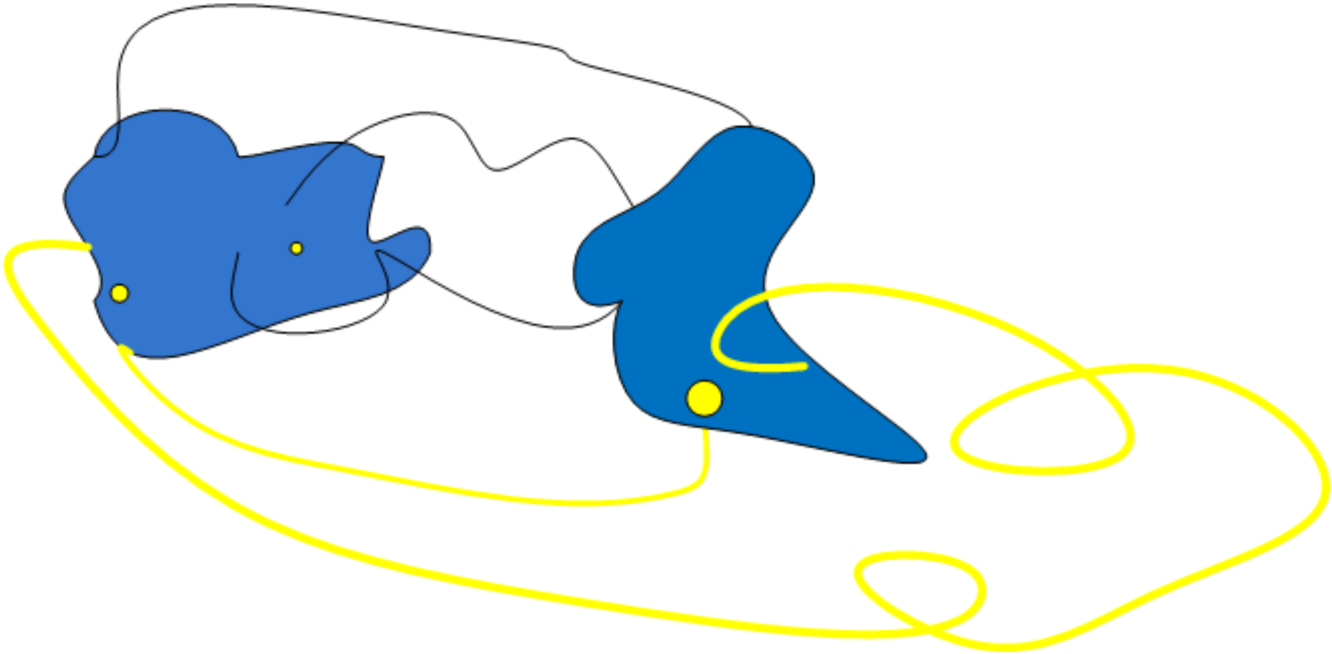
Where is Strategic Value?



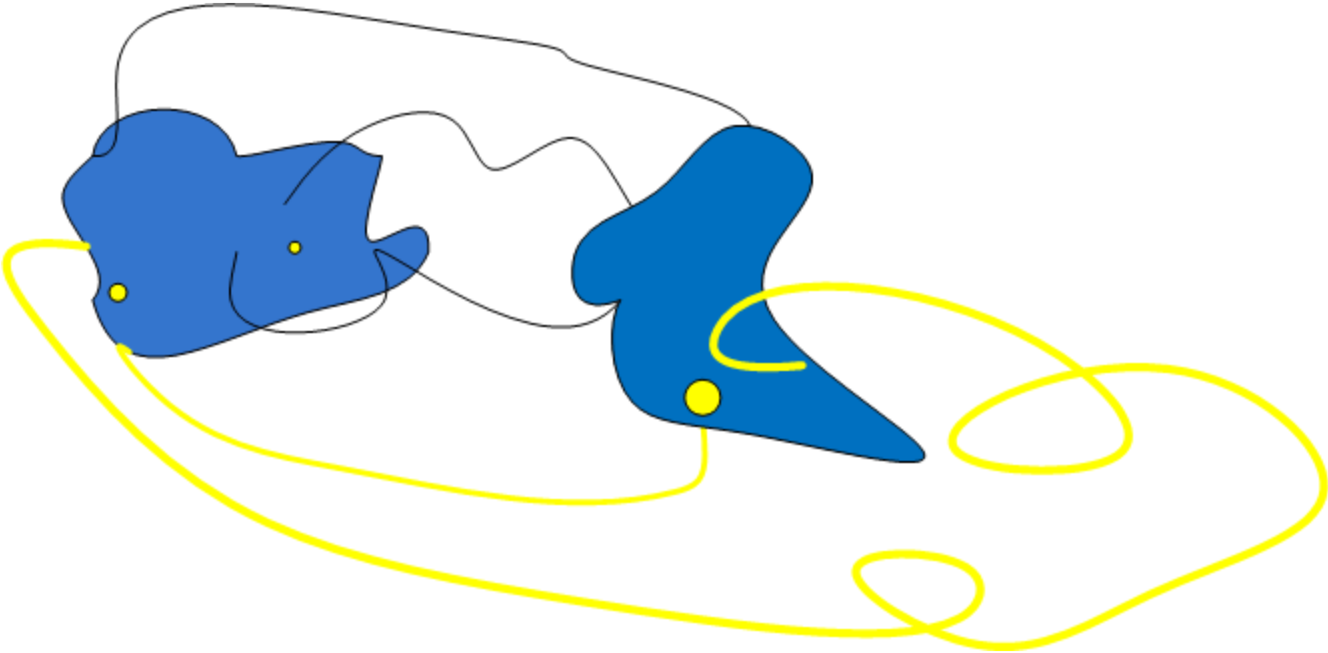
Where is Strategic Value?



Where is Strategic Value?

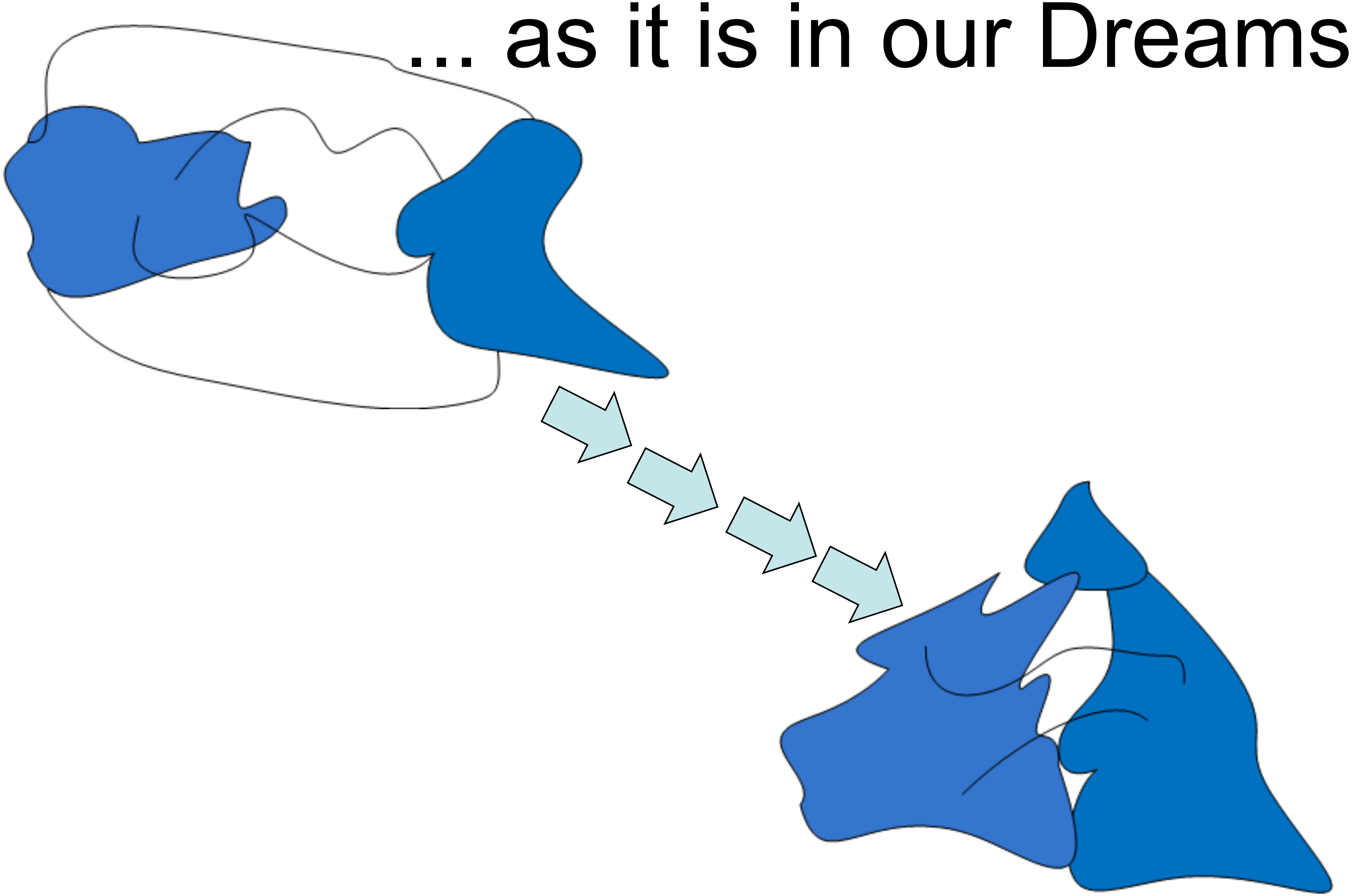


Where is Strategic Value?

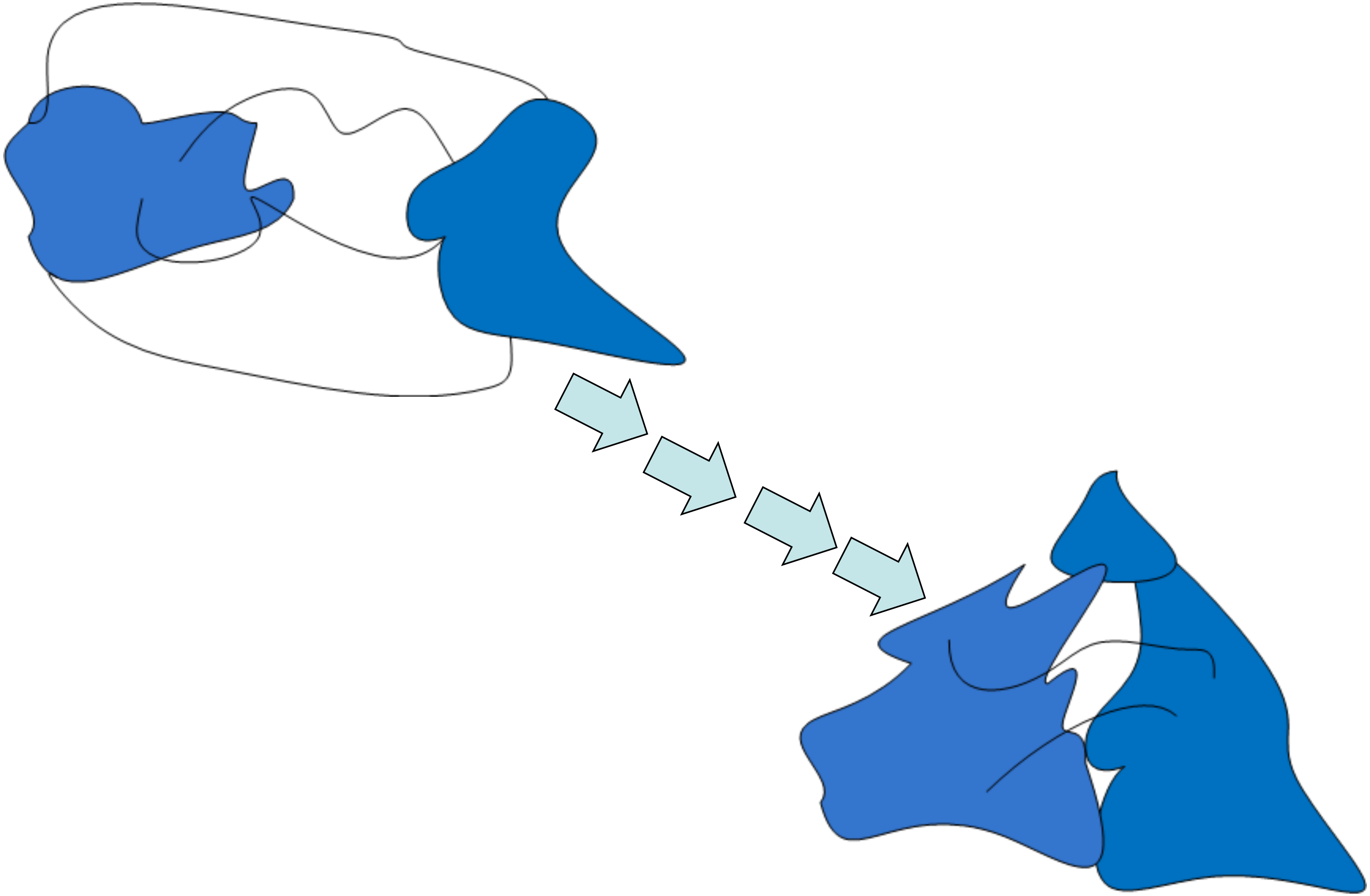


No place to express a cohesive model!

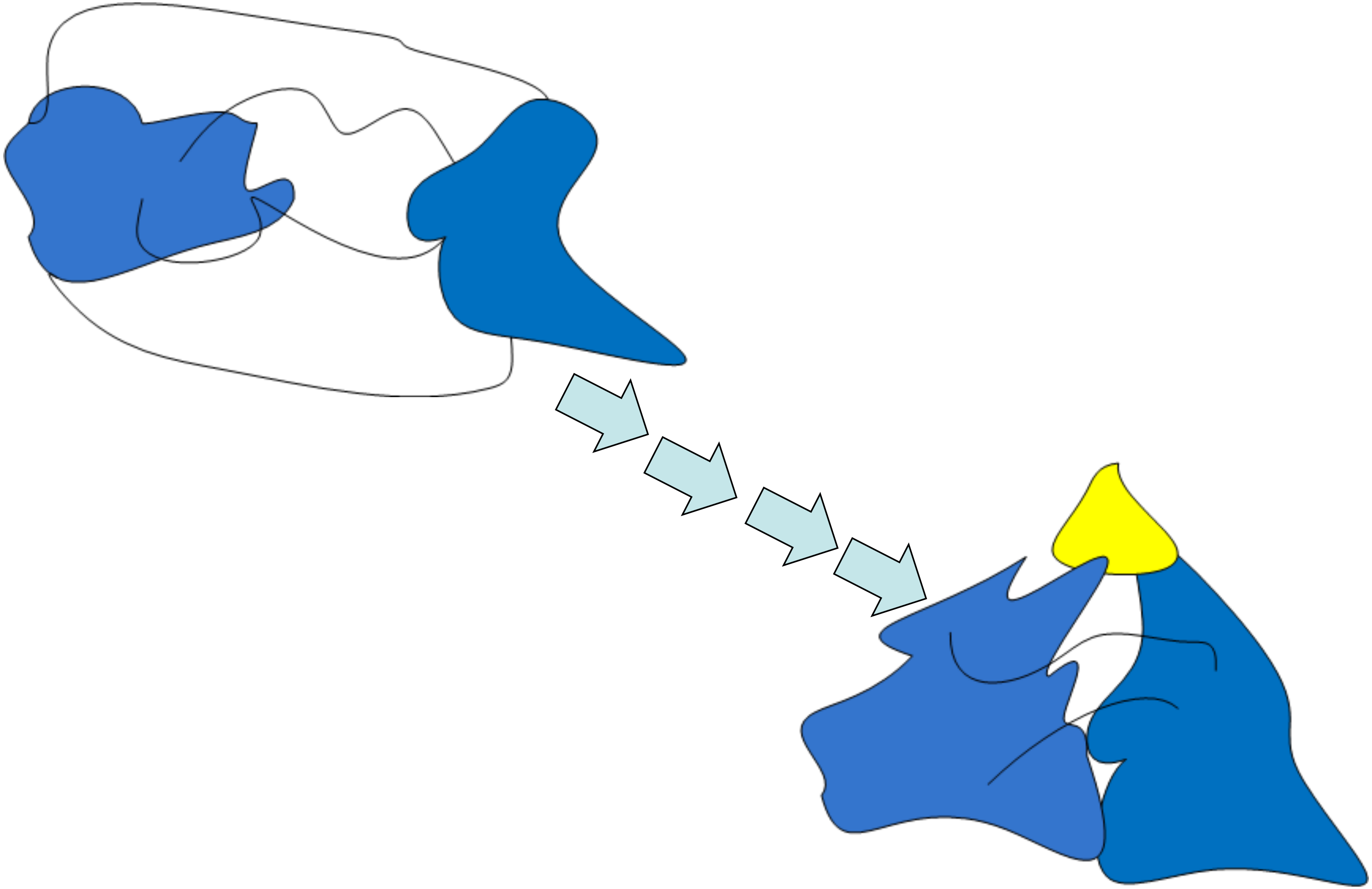
Refactoring
... as it is in our Dreams



Where is the Strategic Value?



Where is the Strategic Value?



Not all of a large system will be well designed.

Distinguish Needs

- Old system does fine
- Old feature ok, but unmaintainable or unstable
- Need new or modified features within framework of old domain concepts.
- Need different approach to *domain* – business innovation!

Part 2: A *really* quick introduction to DDD

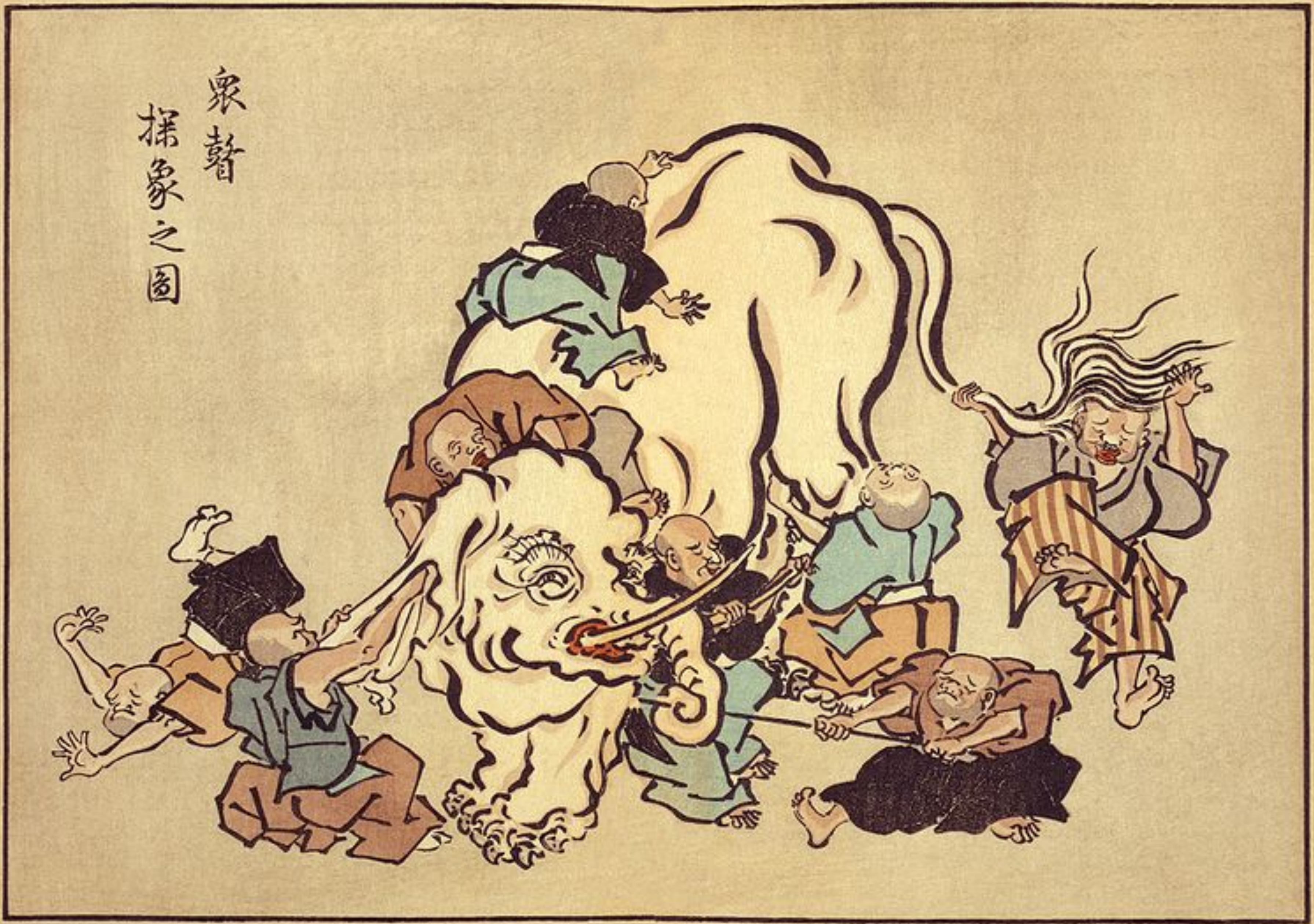
What is a model?

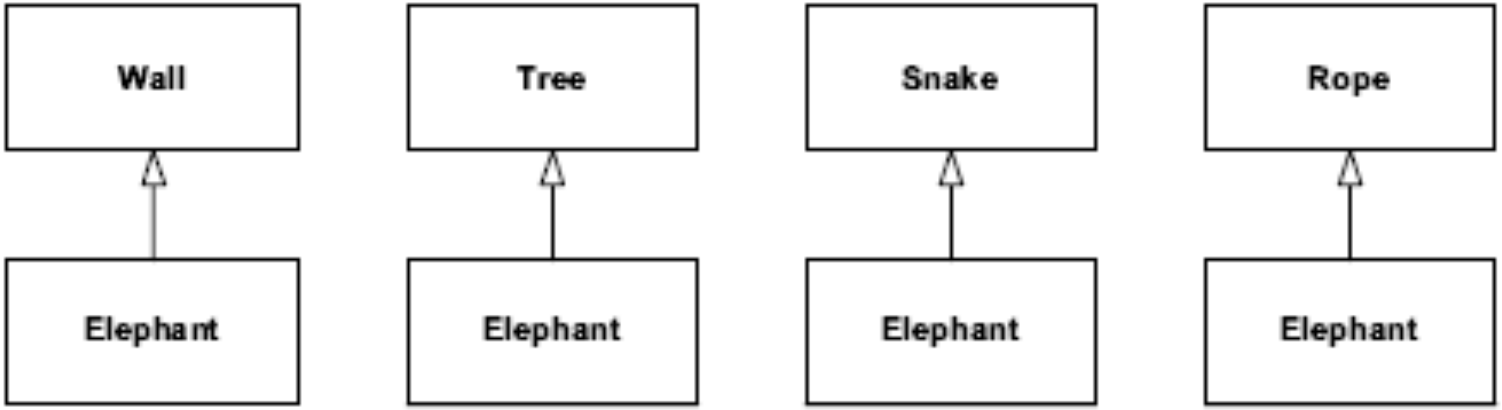
How do we know what something means?

domain A sphere of knowledge or activity.

model A system of abstractions representing selected aspects of the domain.

context The setting in which a word or phrase appears that determines its meaning.


衆瞽
探象之圖



Wall

Elephant

Tree

Elephant

Snake

Elephant

Rope

Elephant

Two Modelling Mistakes

- They keep trying to figure out the “true” nature of an elephant.

DDD modelers don’t seek “realism”.

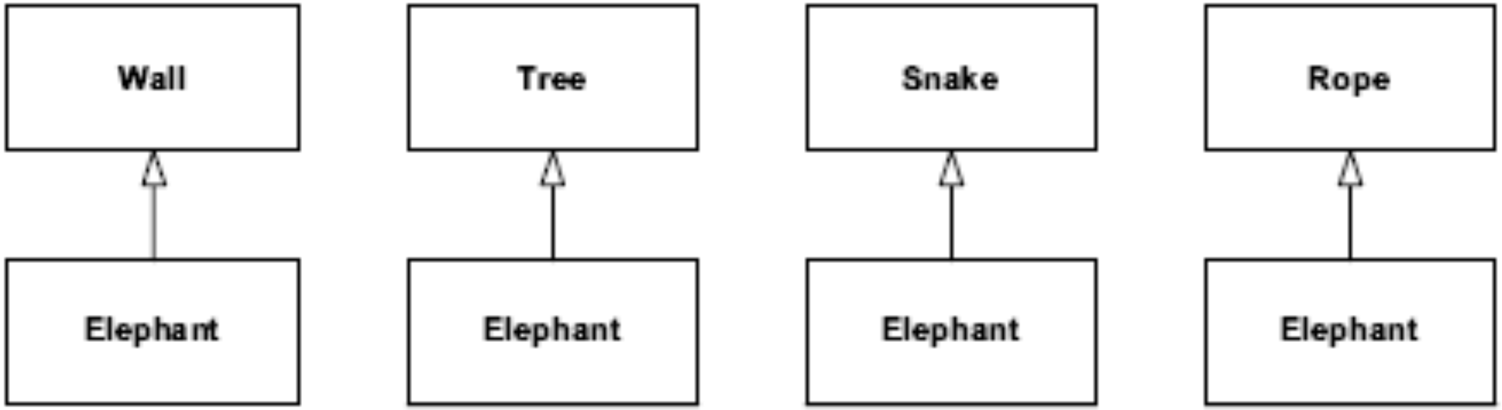
They should be useful for specific purposes.

Two Modelling Mistakes

- They keep trying to figure out the “true” nature of an elephant.
- **They insist on just one model, in spite of incomplete information and possibly different goals.**

There are always multiple models.

bounded context An explicit definition of where a particular model is well-defined and applicable. (Typically a subsystem, or the work owned by a particular team).



Wall

Elephant

Tree

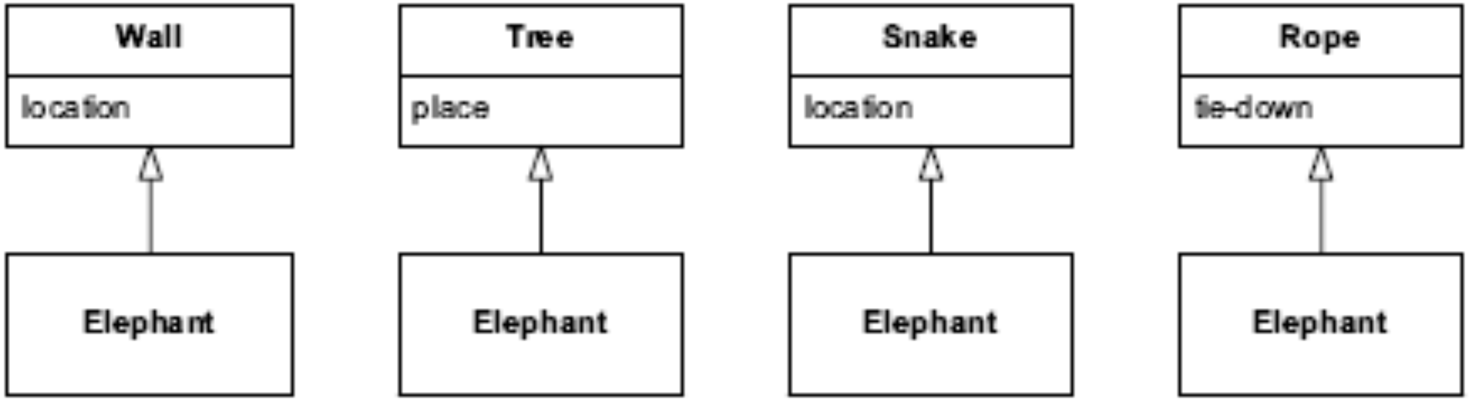
Elephant

Snake

Elephant

Rope

Elephant



Wall
location

Tree
place

Snake
location

Rope
tie-down

Elephant

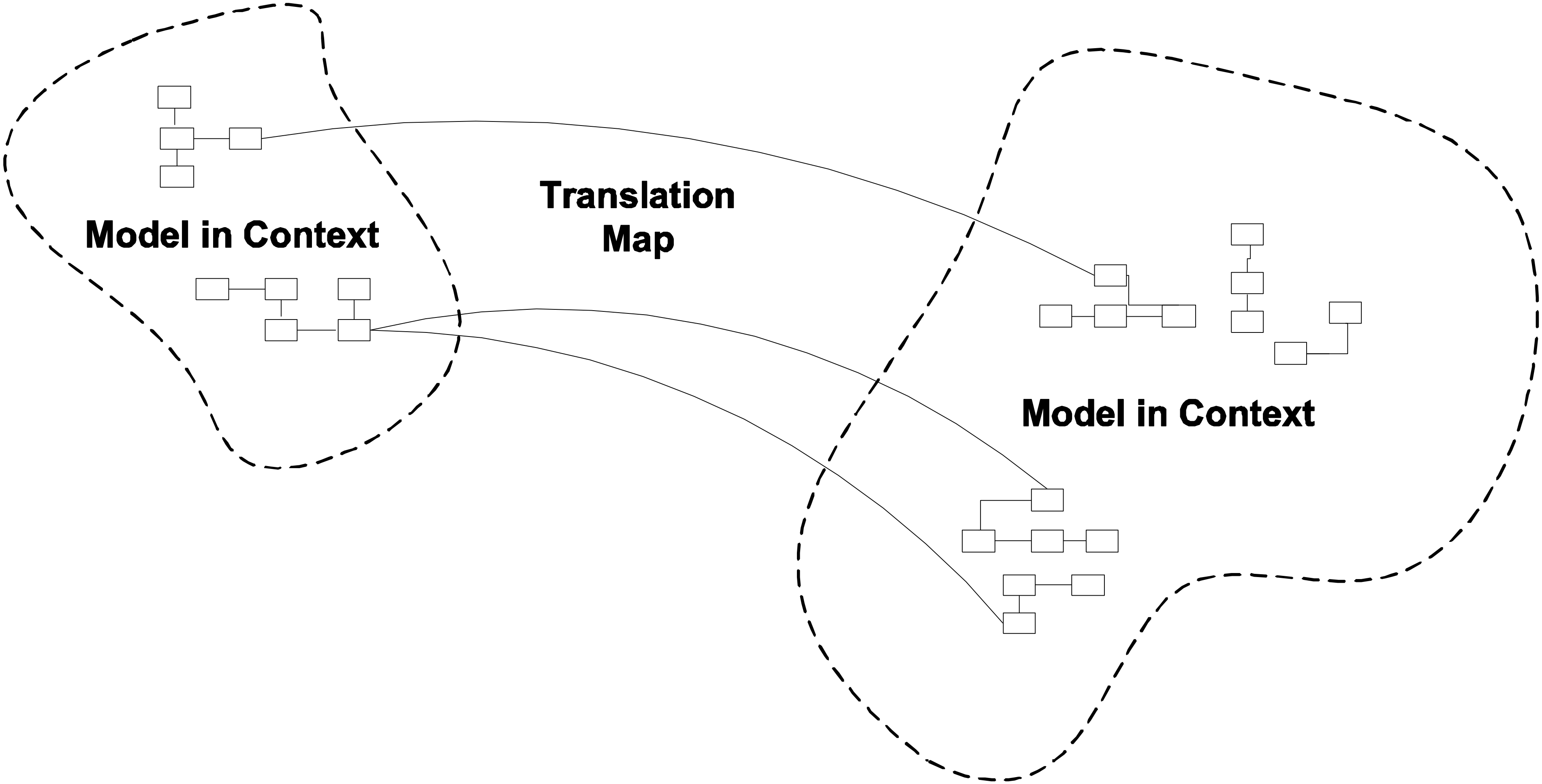
Elephant

Elephant

Elephant

Translations: {Wall.location ↔ Tree.place ↔ Snake.location ↔ Rope.tie-down}

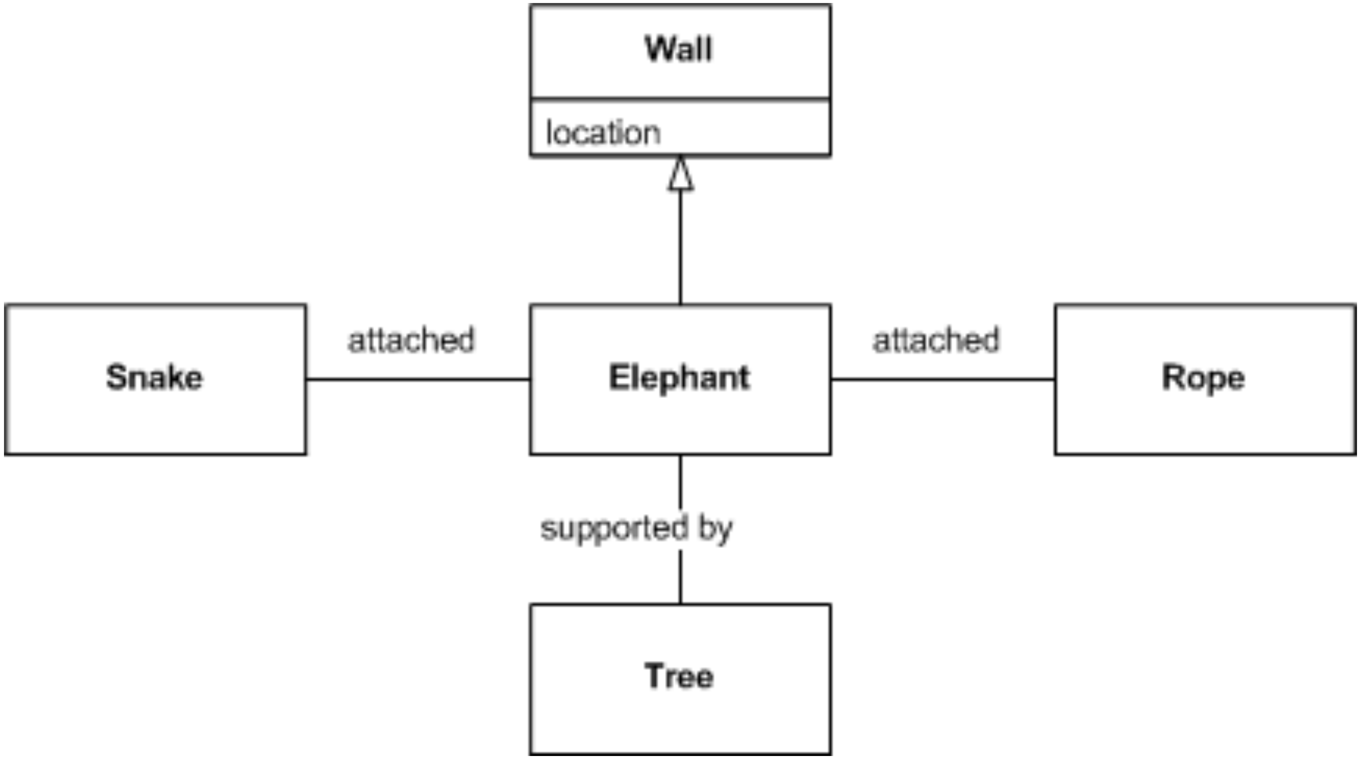
Bounded Contexts



Translation Map

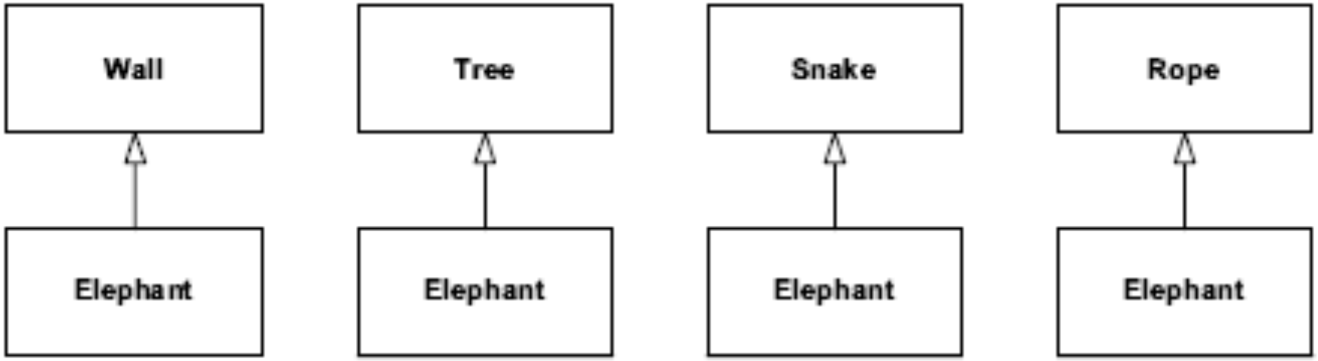
Model in Context

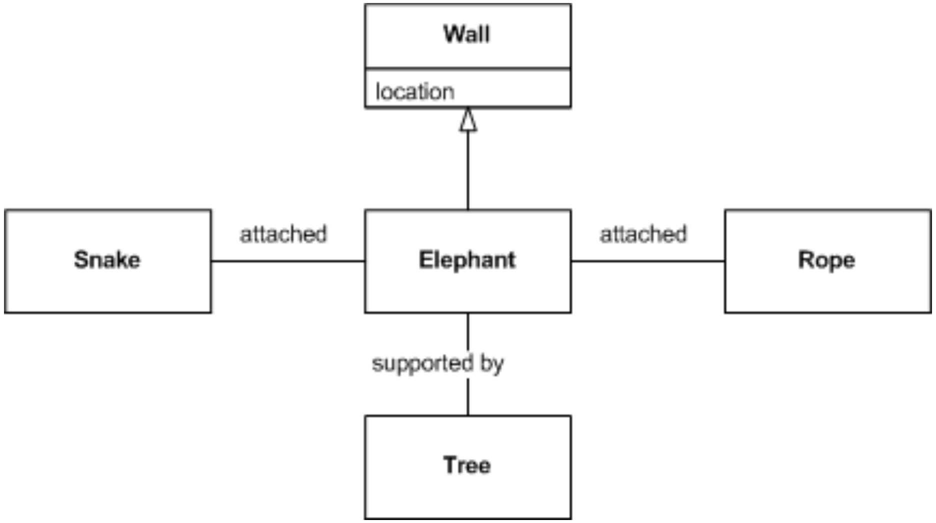
Model in Context

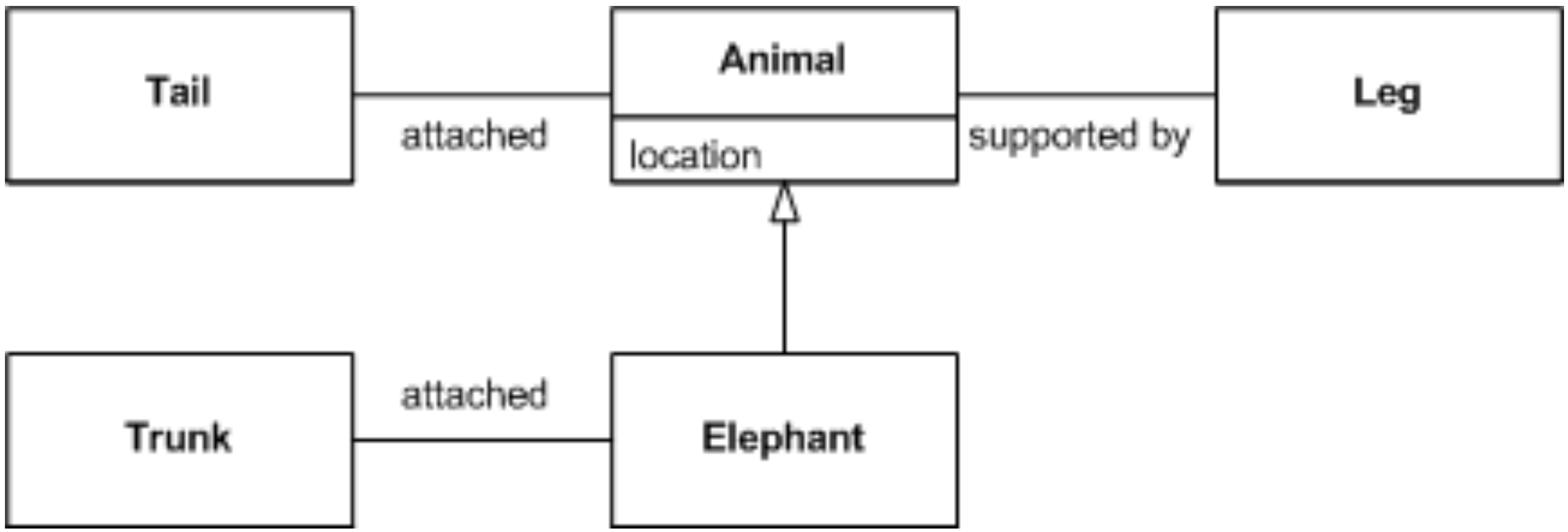

Wall
location
Snake
attached
Elephant
attached
Rope
supported by
Tree

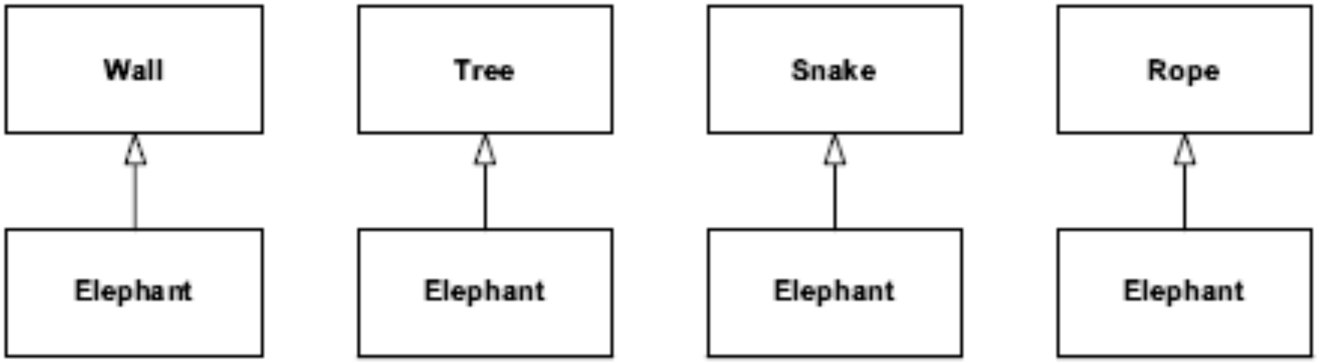
A Bad Strategy:
The Enterprise Model

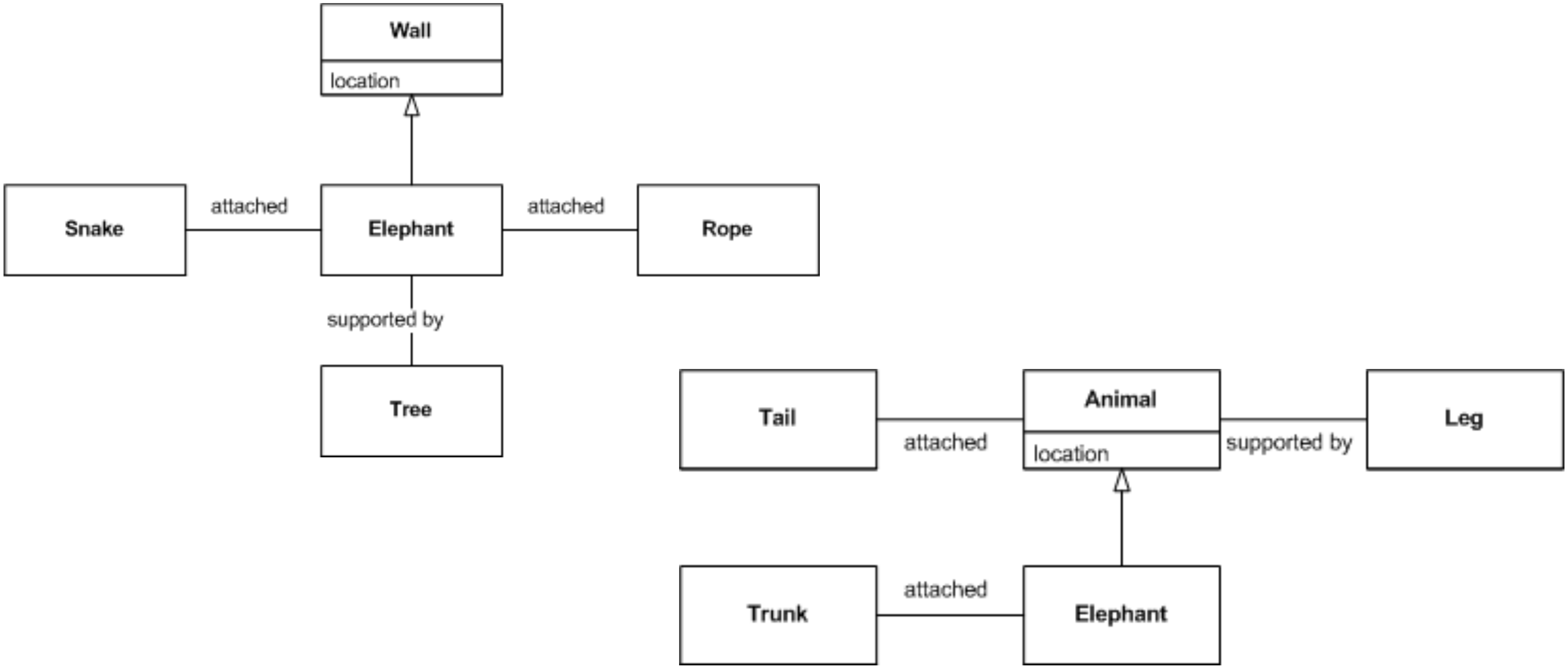
One Ring to rule them all, One Ring to find them, One Ring to bring them all, and in the darkness bind them

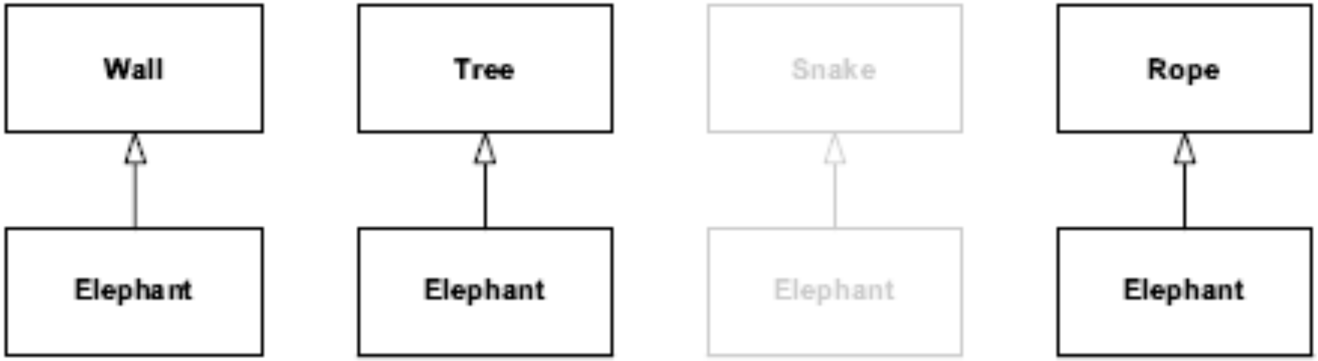

Wall
Tree
Snake
Rope
Elephant
Elephant
Elephant
Elephant

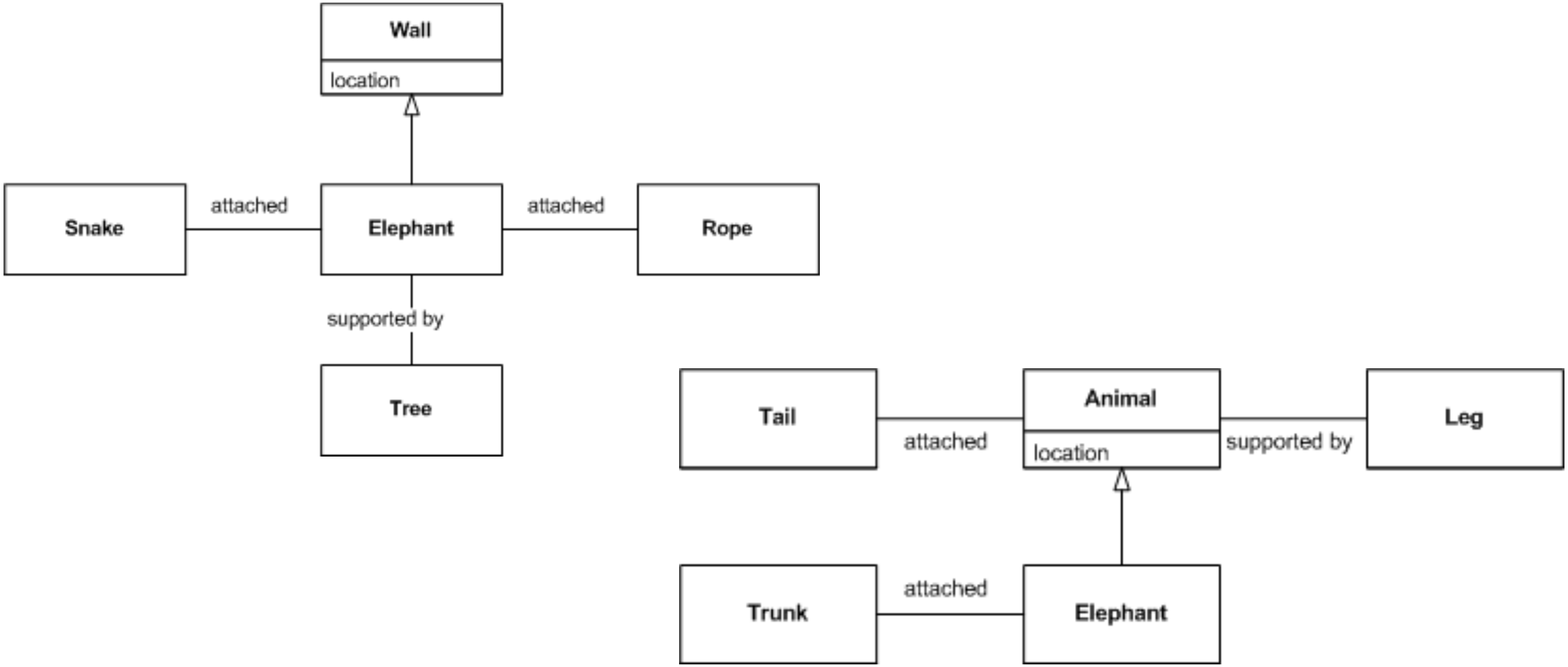

Wall
location
Snake
attached
Elephant
attached
Rope
supported by
Tree


Tail
attached
Animal
location
supported by
Leg
Trunk
attached
Elephant


Wall
Elephant
Tree
Elephant
Snake
Elephant
Rope
Elephant


Wall
location
Snake
attached
Elephant
attached
Rope
supported by
Tree
Tail
attached
Animal
location
supported by
Leg
Trunk
attached
Elephant


Wall
Elephant
Tree
Elephant
Snake
Elephant
Rope
Elephant


Wall
location
Snake
attached
Elephant
attached
Rope
supported by
Tree
Tail
attached
Animal
location
supported by
Leg
Trunk
attached
Elephant

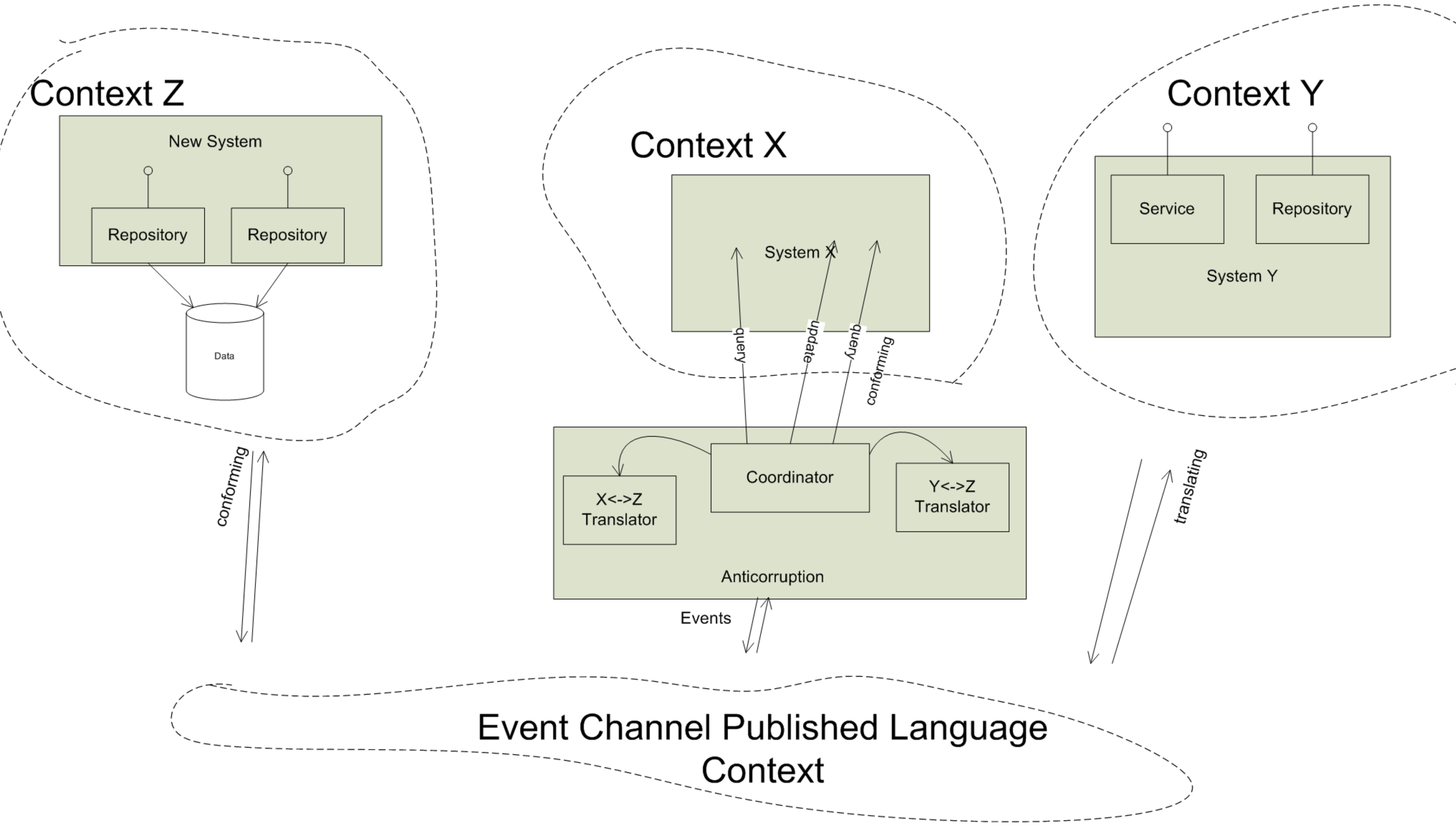
Models that allow us to make simple, clear assertions are only possible within a bounded context.

Sophisticated design is wasted in a big ball of mud.

Changing the fundamental concepts of an established system is unlikely.

Part 3: Sample Strategies

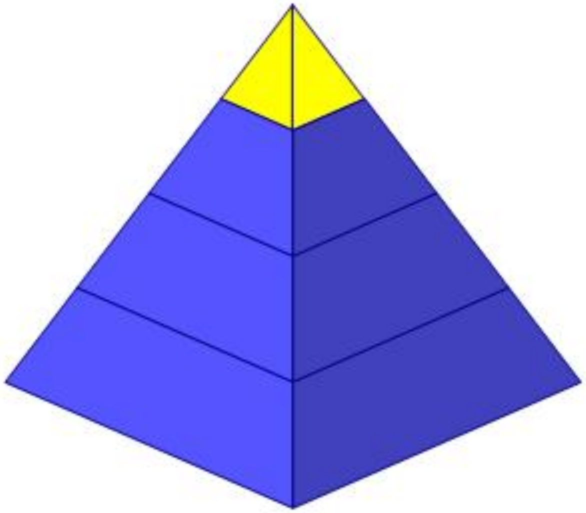
Event Channel

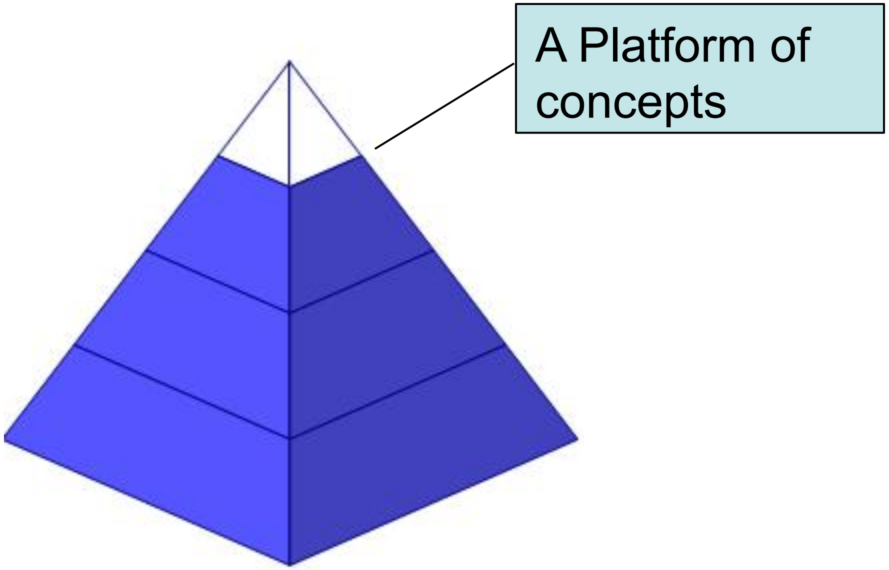


~~Grandiosity~~

Humility

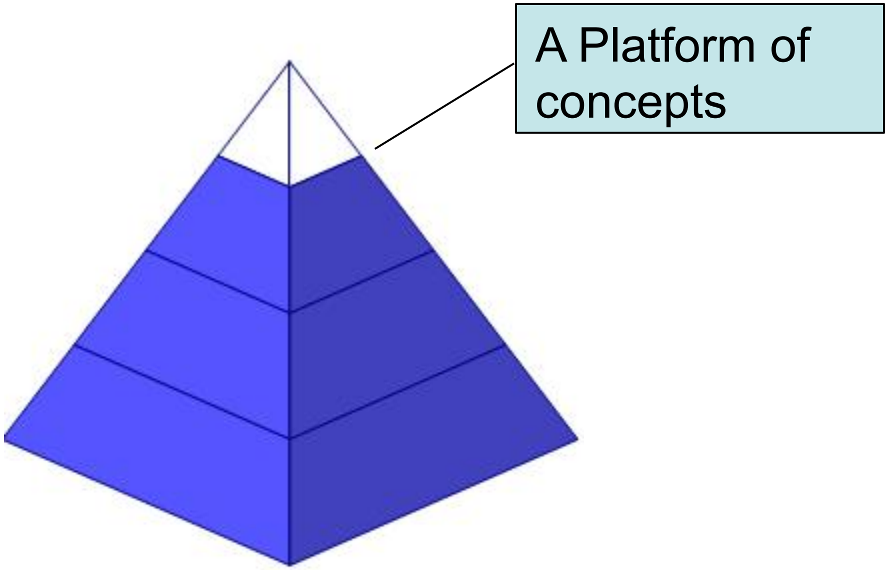
How could we just build the strategic part?





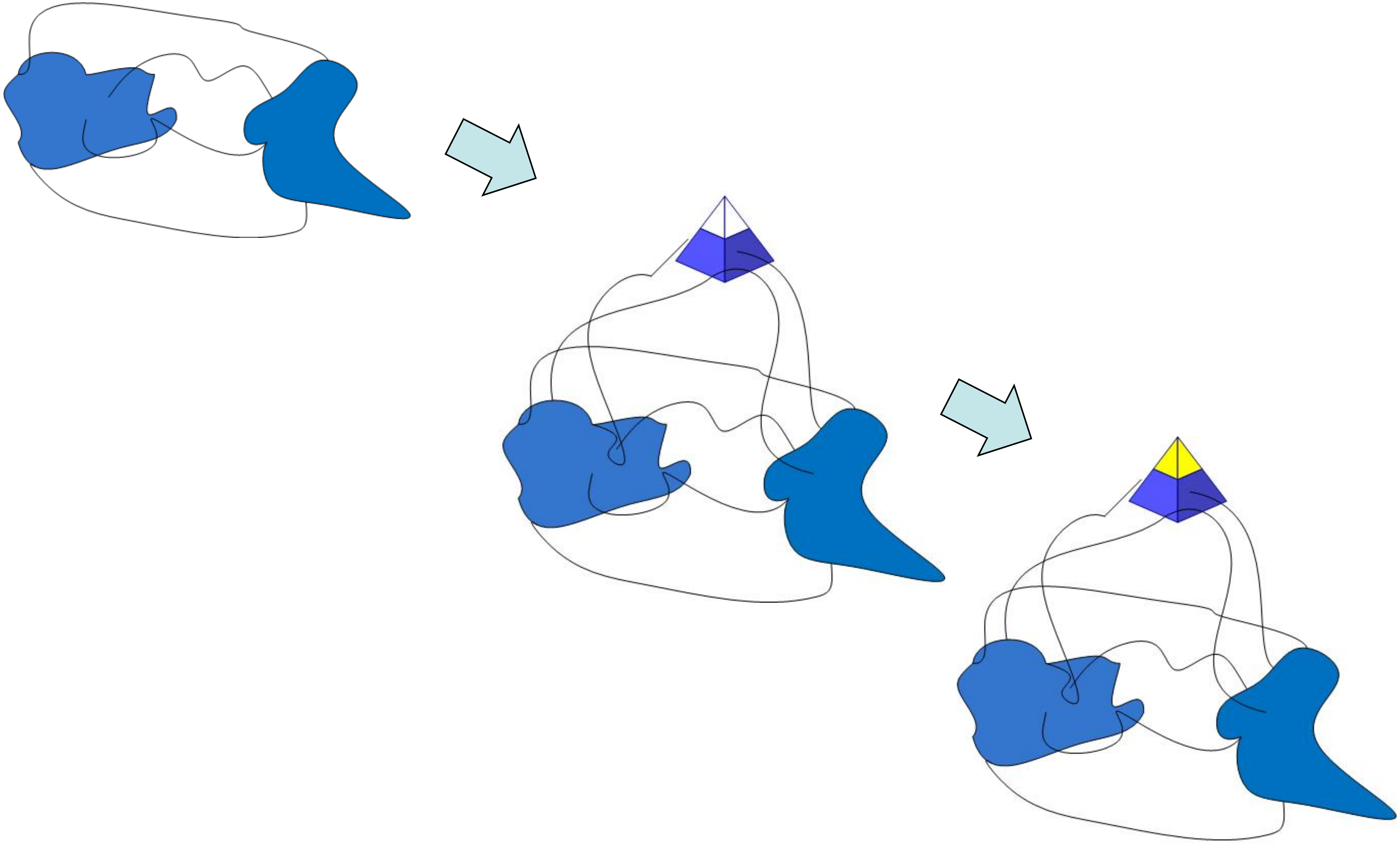
A Platform of concepts

All we really see of the lower layers is the platform *interface*.



A Platform of concepts

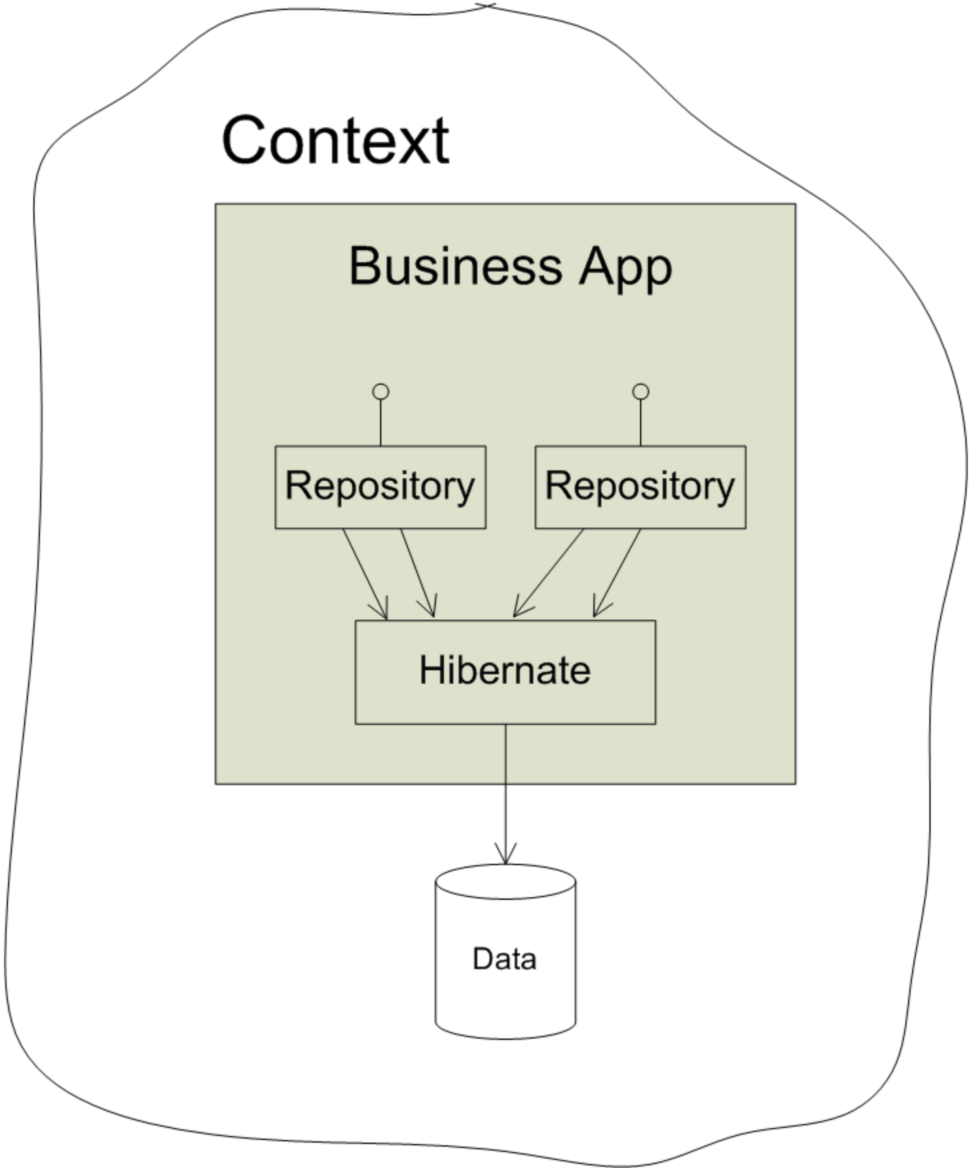
Build Platform Based on Model



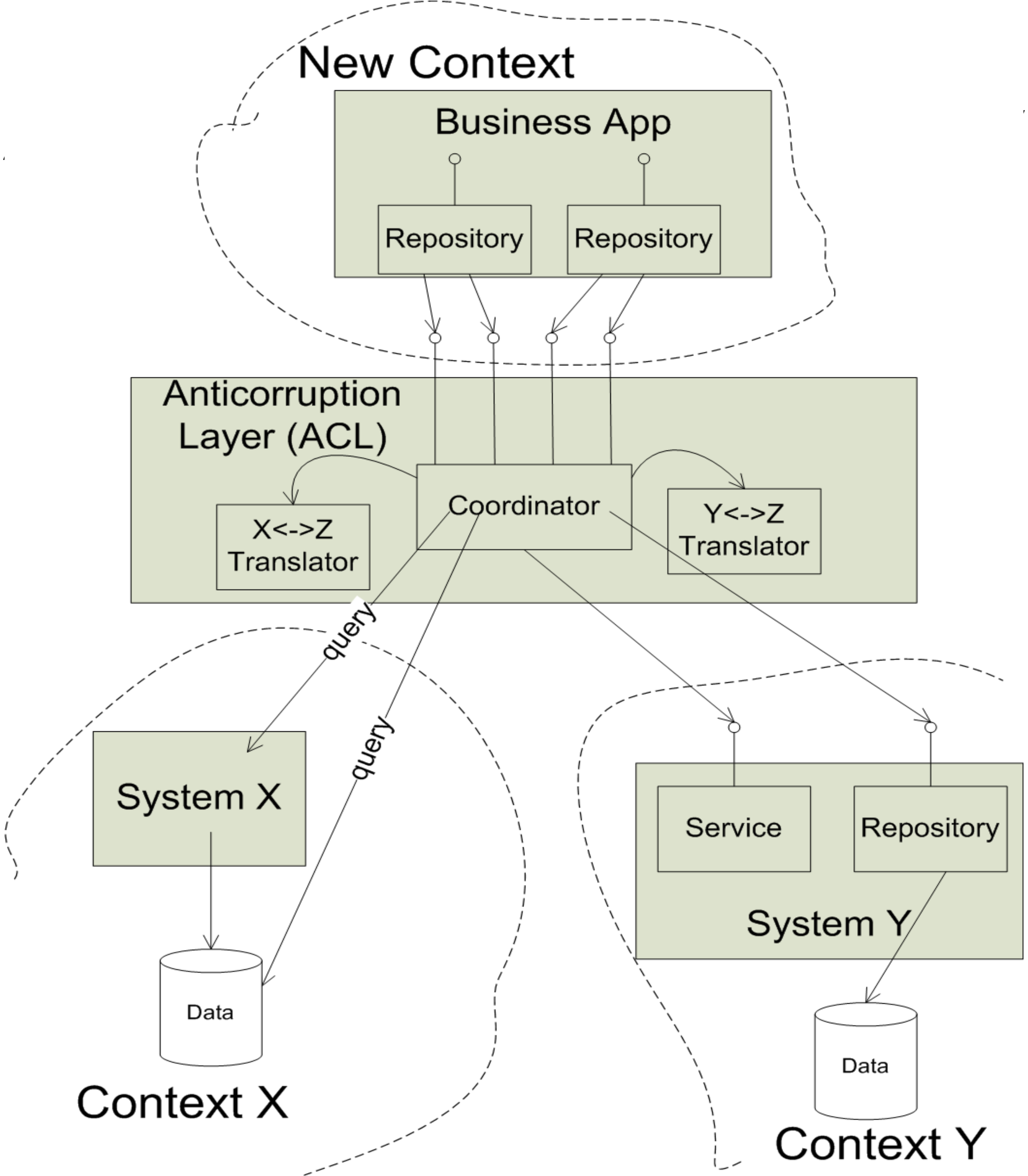
Strategy 1: Bubble Context

- Modest commitment to DDD
- Limited range of data needed from legacy
- No synchronization risk

Classic Repository



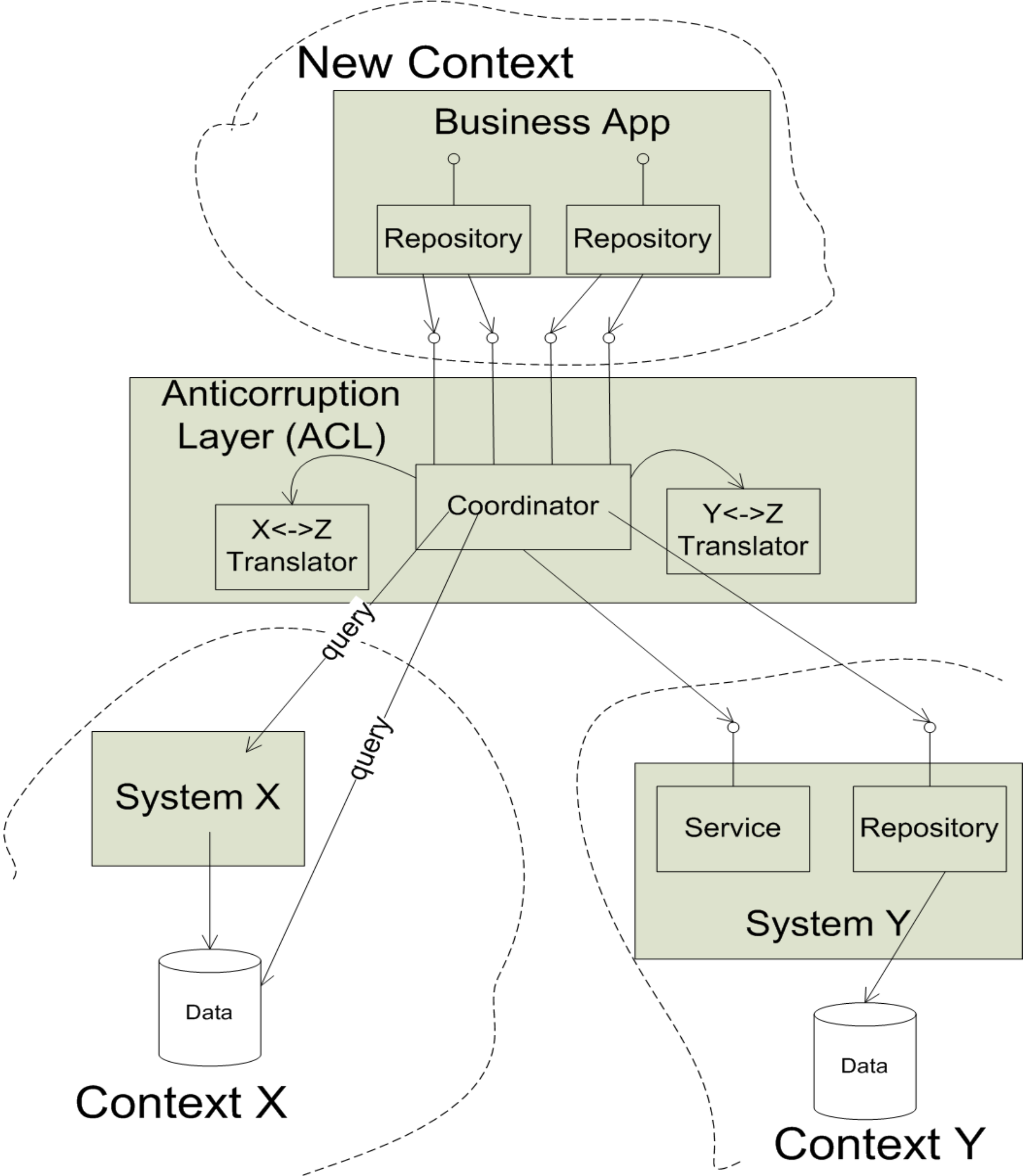
ACL-Backed Repository



Strategy 2: ACL Synchronization

- Starting a new chapter
- Organization committed to major new development with new design approach
- May progress from Bubble Context as pilot project

ACL-Backed Repository



New Context

Business App

Repository

Repository

Anticorruption Layer (ACL)

Coordinator

X<->Z Translator

Y<->Z Translator

query

query

System X

Data

Context X

Service

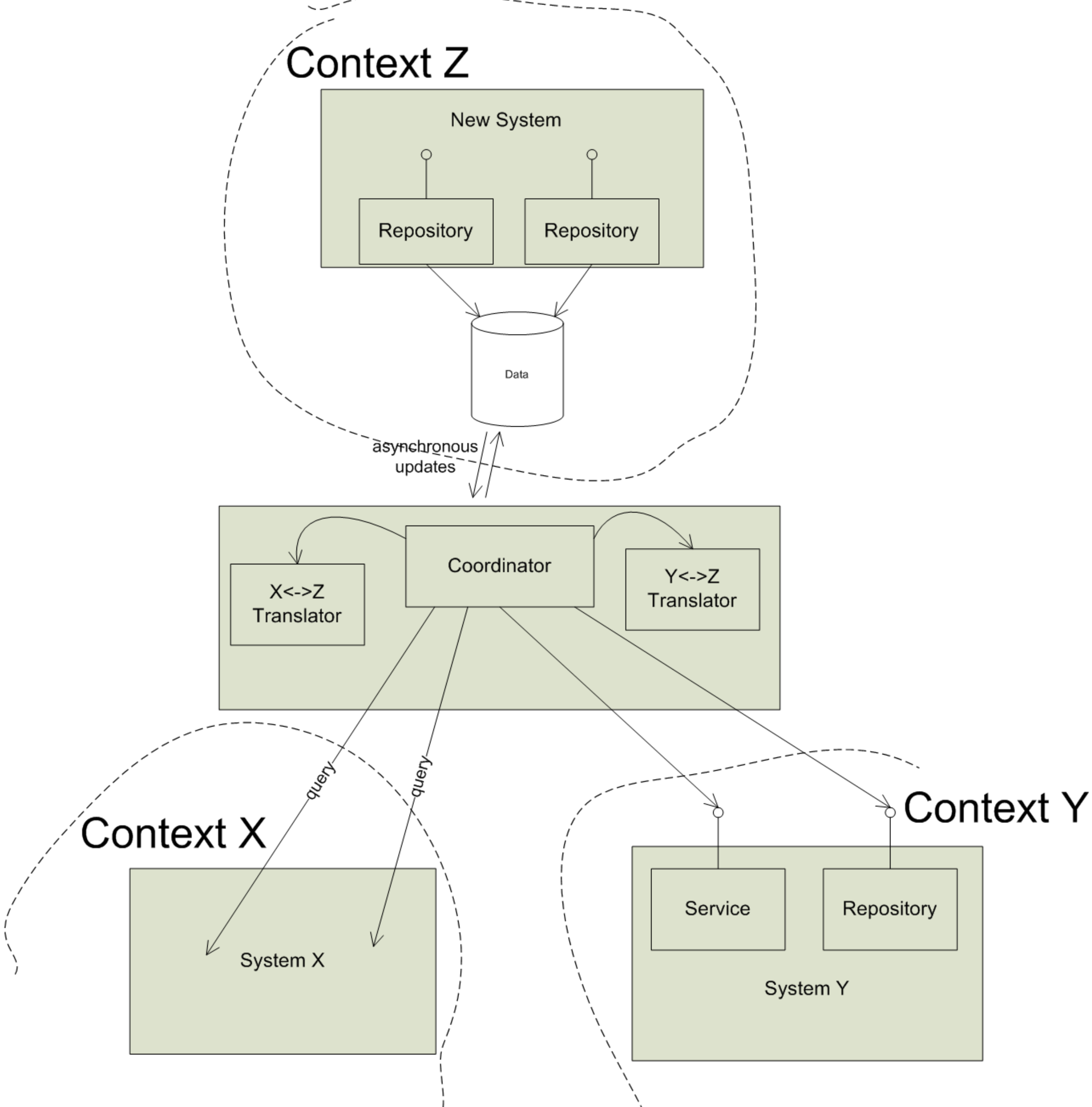
Repository

System Y

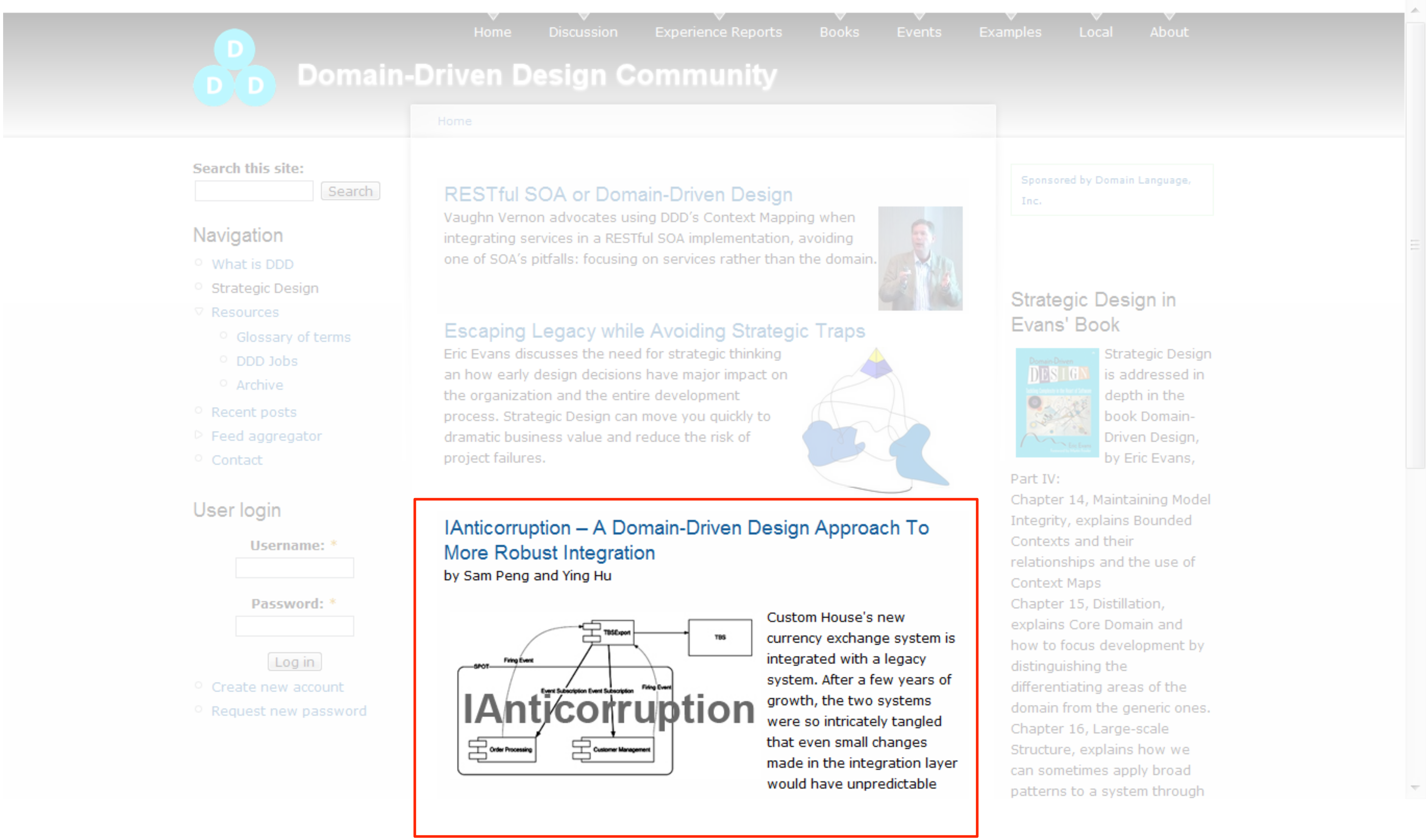
Data

Context Y

Synchronizing ACL



dddcommunity.org/strategicdesign



Domain-Driven Design Community

Home

Search this site:

Navigation

- What is DDD
- Strategic Design
- Resources
  - Glossary of terms
  - DDD Jobs
  - Archive
- Recent posts
- Feed aggregator
- Contact

User login

Username: *

Password: *

- Create new account
- Request new password

RESTful SOA or Domain-Driven Design

Vaughn Vernon advocates using DDD's Context Mapping when integrating services in a RESTful SOA implementation, avoiding one of SOA's pitfalls: focusing on services rather than the domain.

Escaping Legacy while Avoiding Strategic Traps

Eric Evans discusses the need for strategic thinking an how early design decisions have major impact on the organization and the entire development process. Strategic Design can move you quickly to dramatic business value and reduce the risk of project failures.

IAnticorruption – A Domain-Driven Design Approach To More Robust Integration

by Sam Peng and Ying Hu

Custom House's new currency exchange system is integrated with a legacy system. After a few years of growth, the two systems were so intricately tangled that even small changes made in the integration layer would have unpredictable

Sponsored by Domain Language, Inc.

Strategic Design in Evans' Book

Strategic Design is addressed in depth in the book Domain-Driven Design, by Eric Evans, Part IV:
Chapter 14, Maintaining Model Integrity, explains Bounded Contexts and their relationships and the use of Context Maps
Chapter 15, Distillation, explains Core Domain and how to focus development by distinguishing the differentiating areas of the domain from the generic ones.
Chapter 16, Large-scale Structure, explains how we can sometimes apply broad patterns to a system through

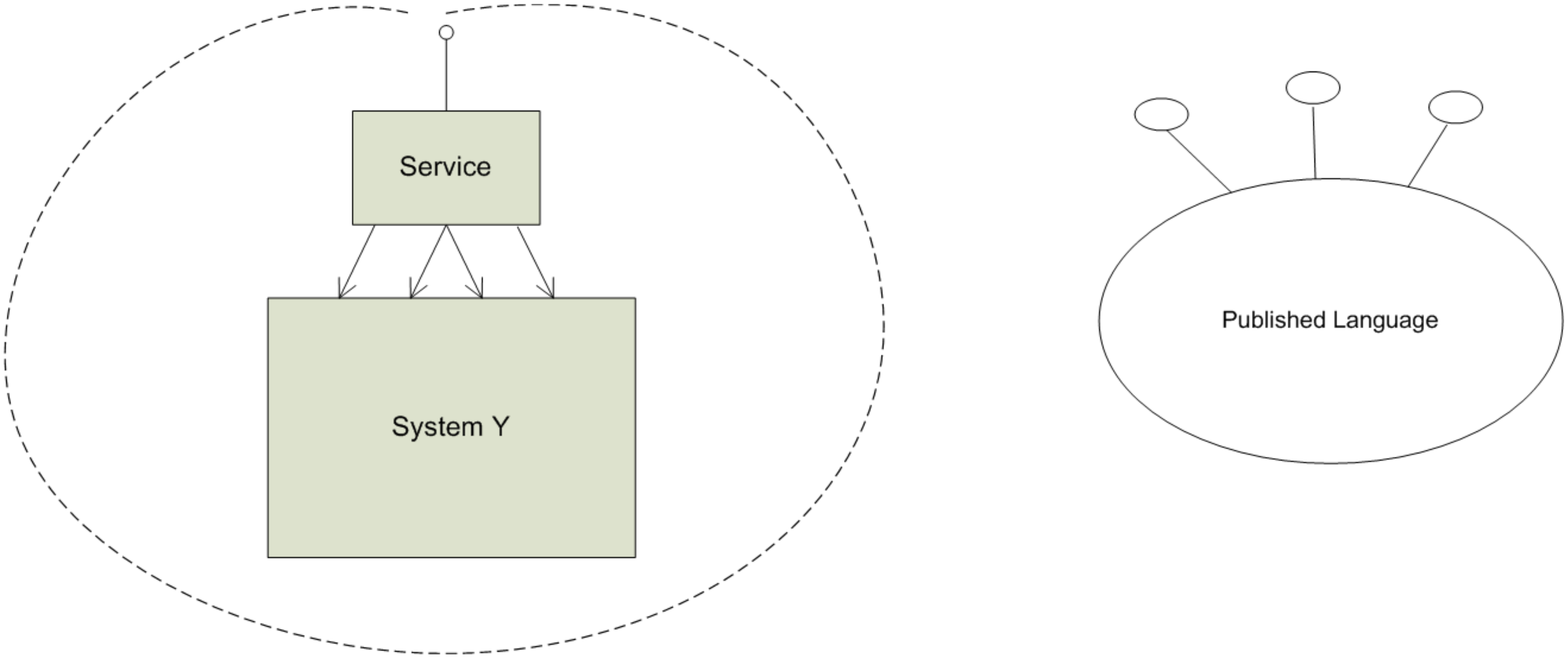
Strategy 3: Expose Legacy Asset in terms of a Published Language

- Modest Commitment to DDD

- Specific functionality/data of legacy needed elsewhere

- Reduce coupling with dependent systems and with other legacy capabilities

Classic Open-Host Service



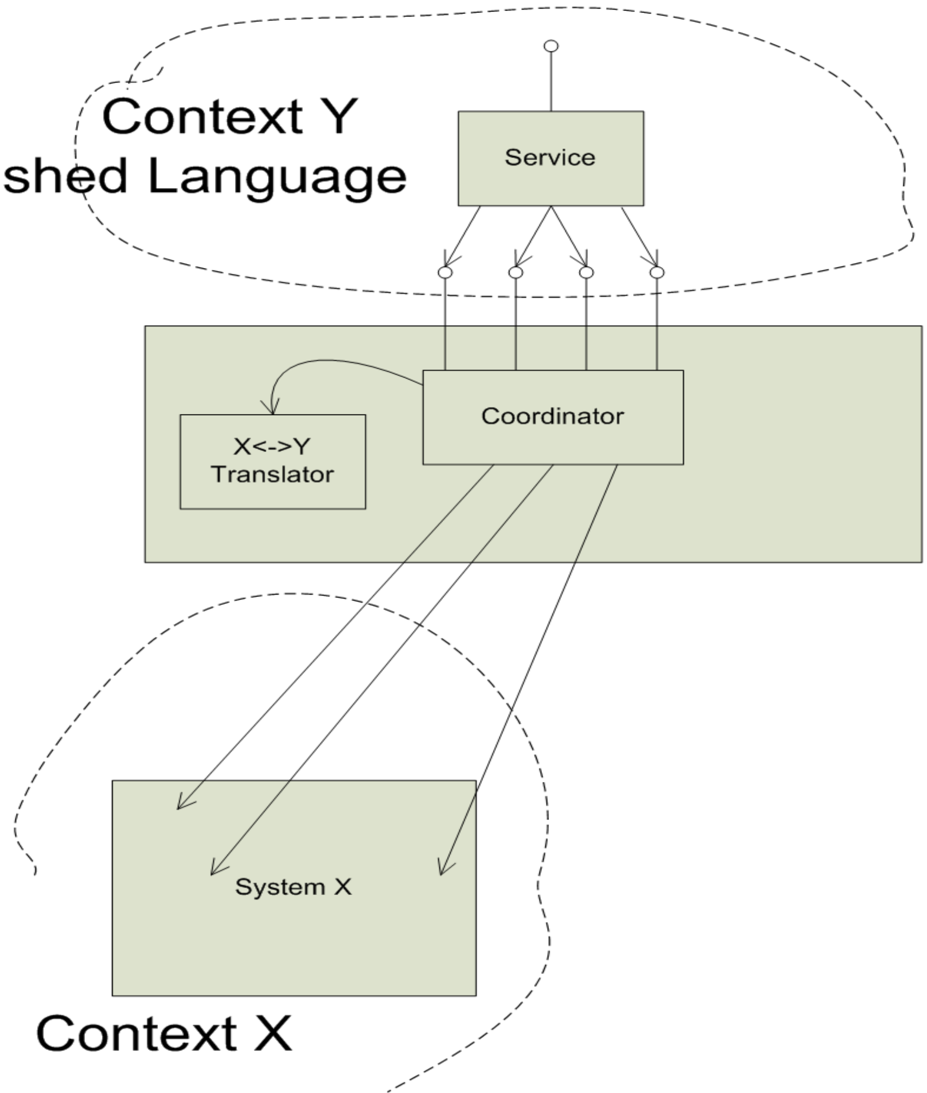
Service

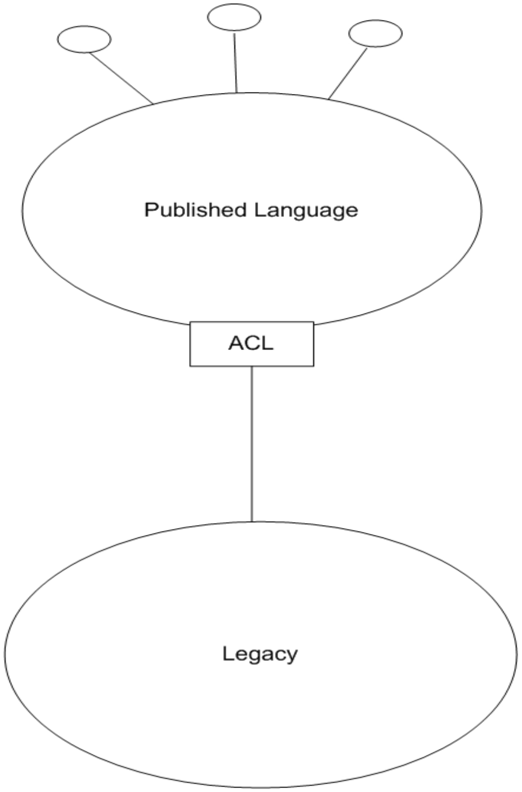
System Y

Published Language

SOA + Bounded Context

ACL-Backed Open-Host Service


Context Y
Published Language
Service
Coordinator
X<->Y
Translator
System X
Context X


Published Language
ACL
Legacy

Published Language Commons

- A suite of services, events, etc. with interfaces expressed in terms of integration model
- Implementations typically in distinct contexts
- Consumers typically in distinct contexts

Strategy 4: Domain Event Channel in Published Language Commons

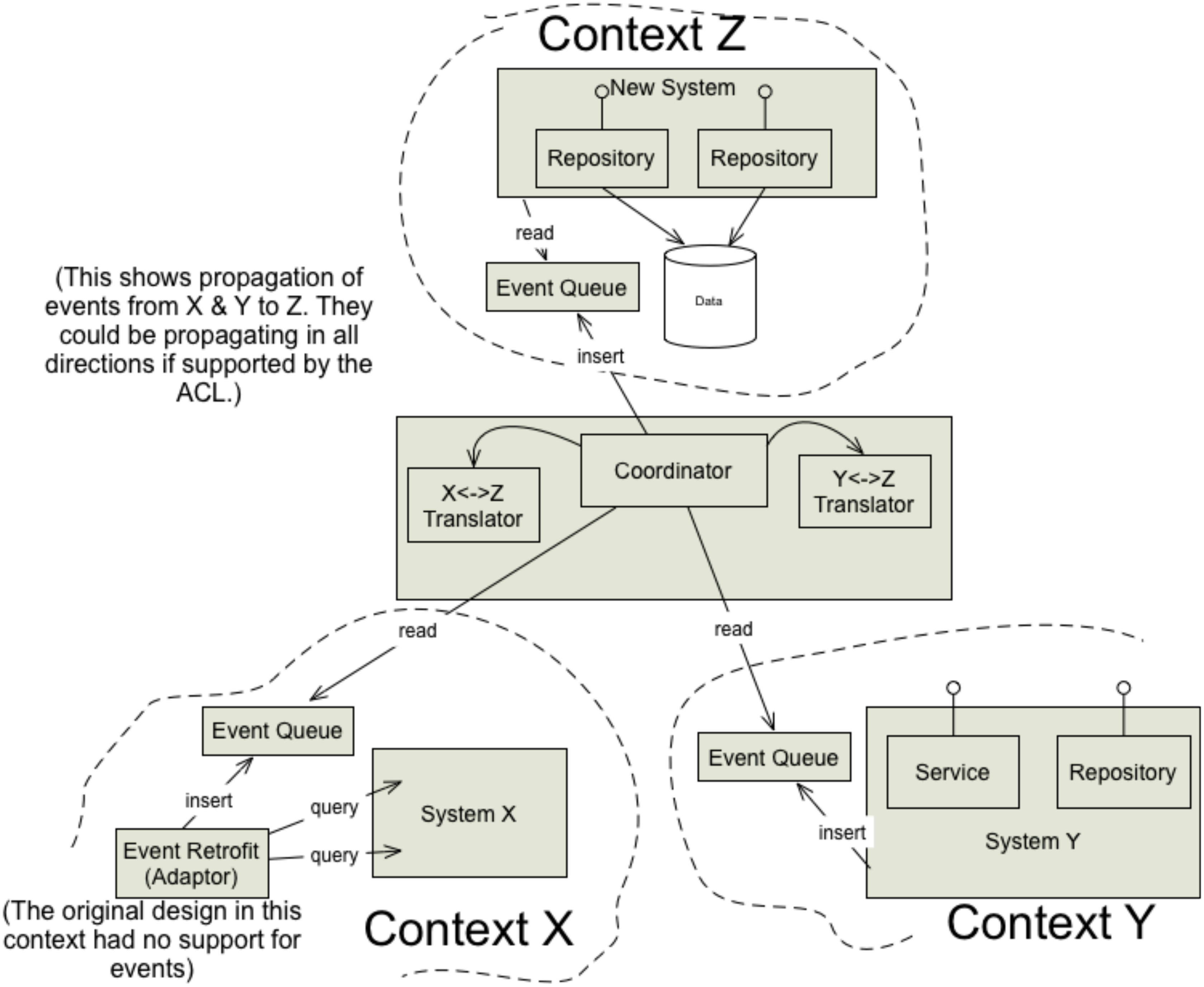
- Organization committed to major new development with new architecture
- Relatively independent development in distinct contexts
- More robust and verifiable system integration

What it Looks Like

- Domain Events defined in terms of published language “integration model”

- Systems integrated through creating or consuming Domain Events

- Legacy system modified to create events or
  ACL tells story of events in legacy



(This shows propagation of events from X & Y to Z. They could be propagating in all directions if supported by the ACL.)

(The original design in this context had no support for events)

Four Example Strategies

- Bubble context for a limited objective
  - ACL-backed Repository (umbilical)
- Autonomous Bubble
  - Self-contained context is peer to legacy
  - Sychronizing ACL between two data stores
- Expose Legacy Assets
  in Published Language Commons
  - ACL Backed Open-host Service
- Domain Events Channel
  in Published Language Commons

Where to look for more

- Context Mapping
  - Chapter 14 in DDD book
- DDD Community (dddcommunity.org)
  - dddcommunity.org/strategicdesign
- My evolving thoughts
  - domainlanguage.com/newsletter