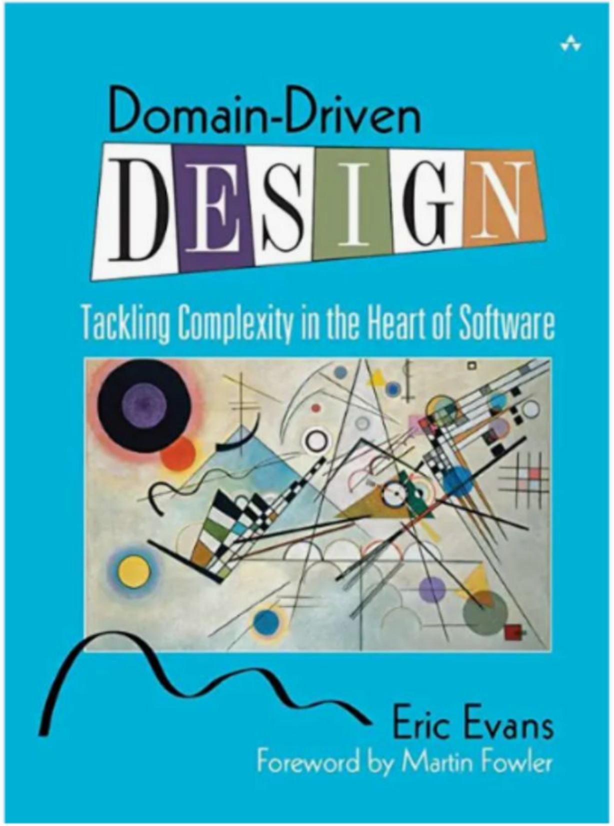
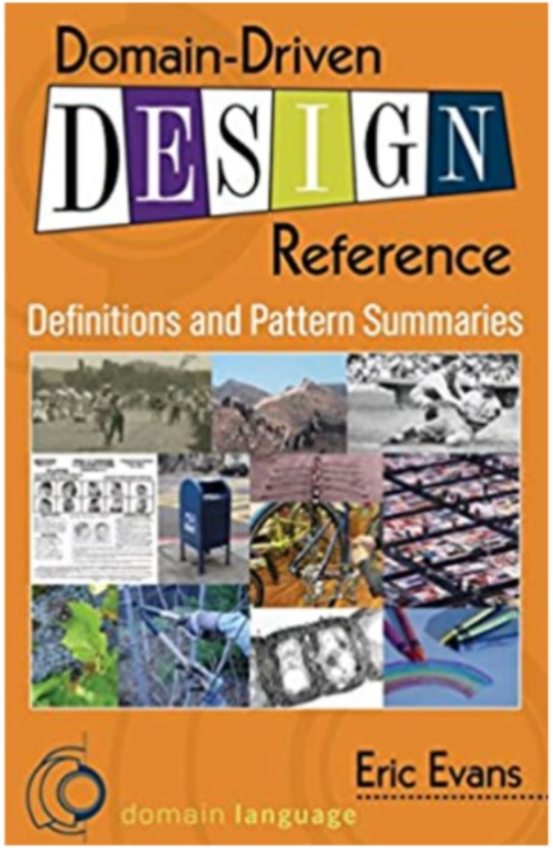
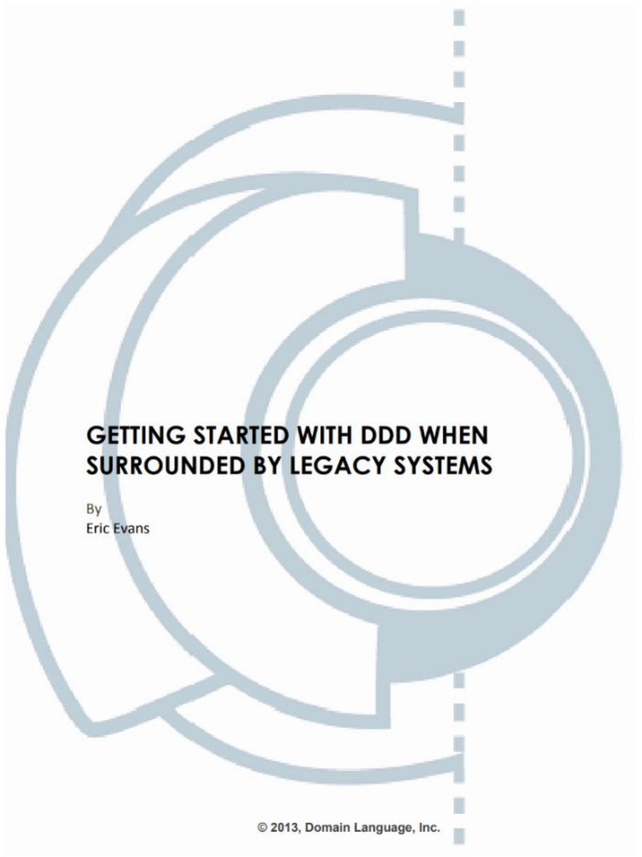

Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler



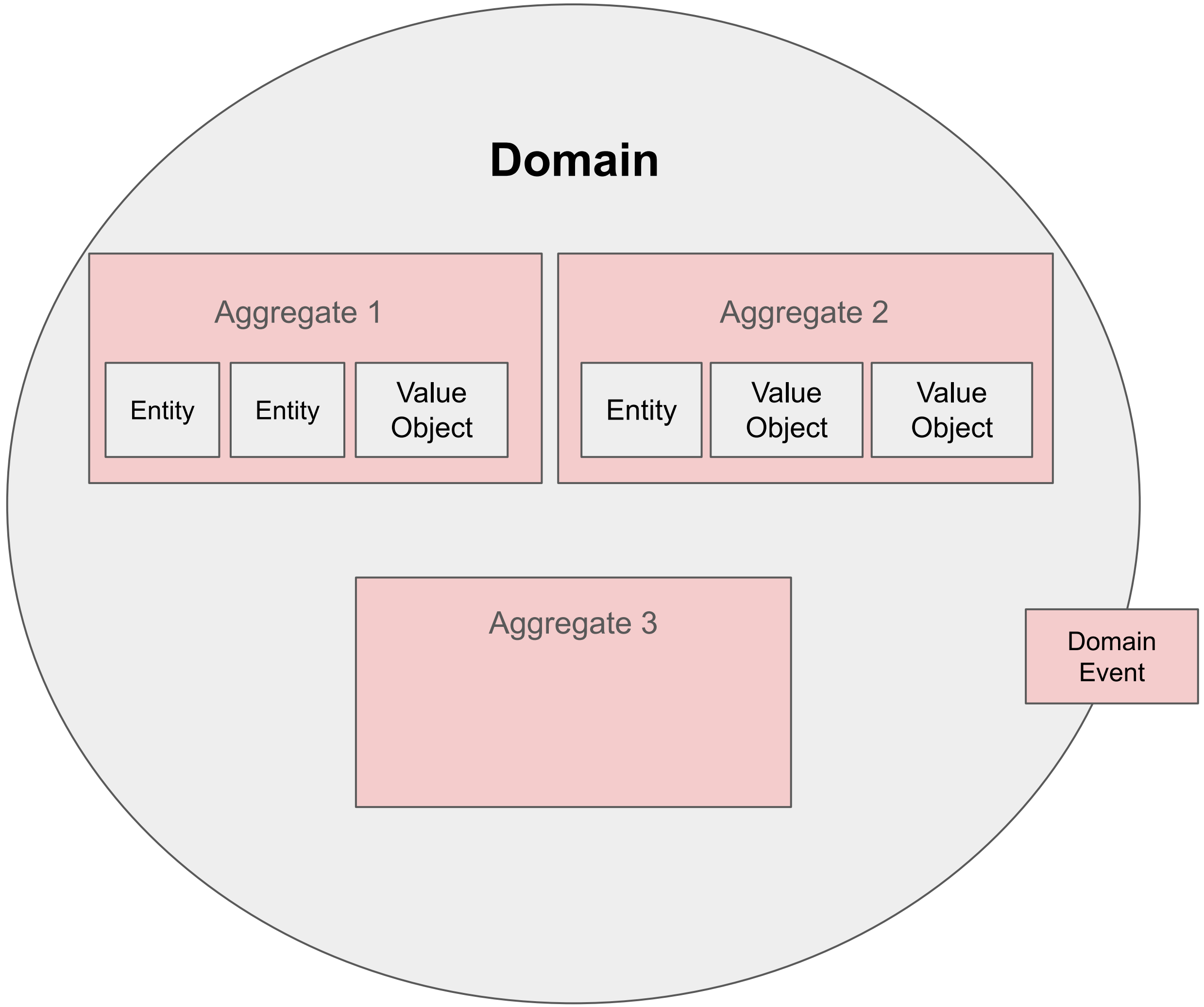

GETTING STARTED WITH DDD WHEN SURROUNDED BY LEGACY SYSTEMS

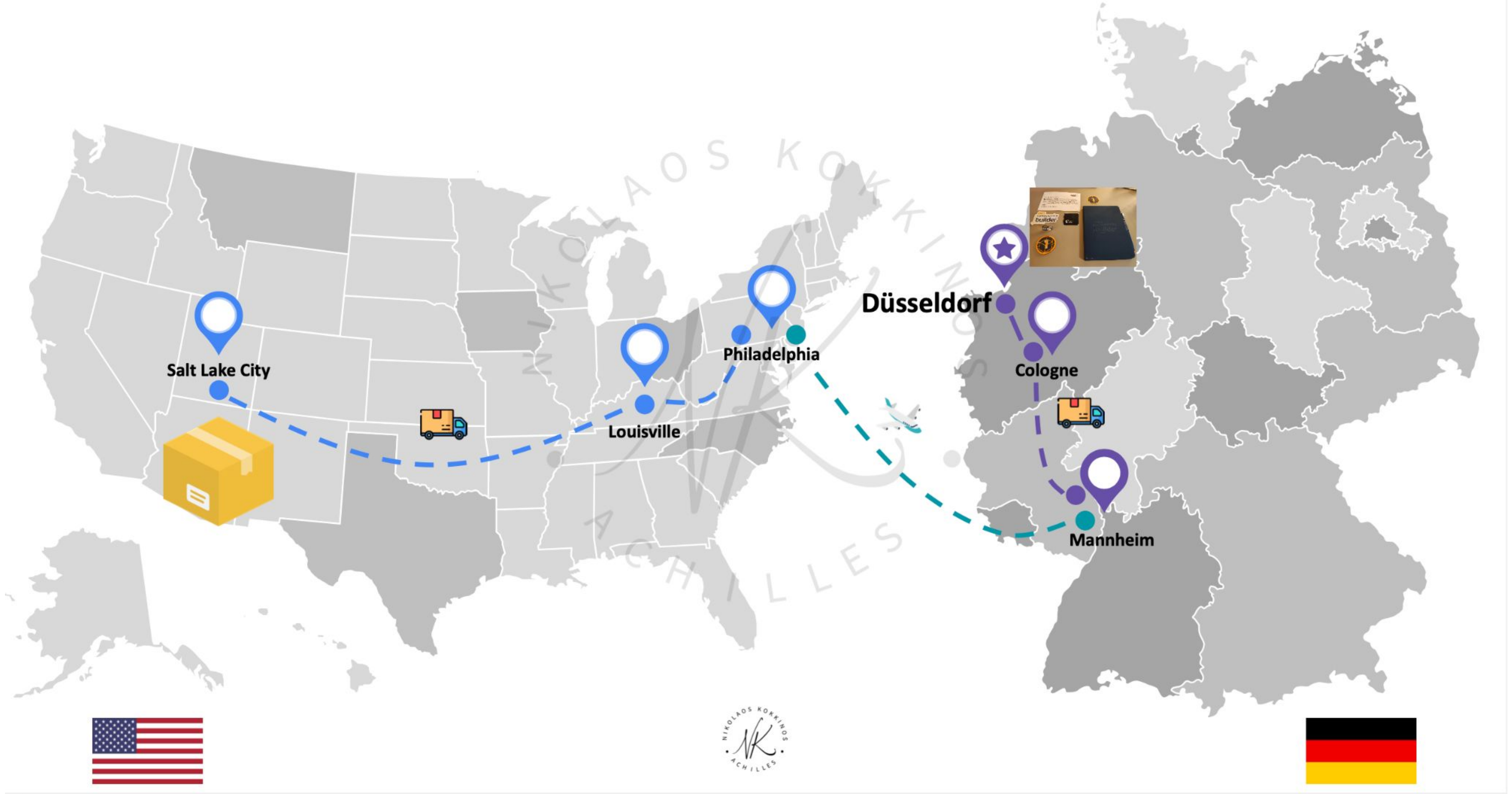
By
Eric Evans

© 2013, Domain Language, Inc.

DDD Cologne Aug 26, 2024

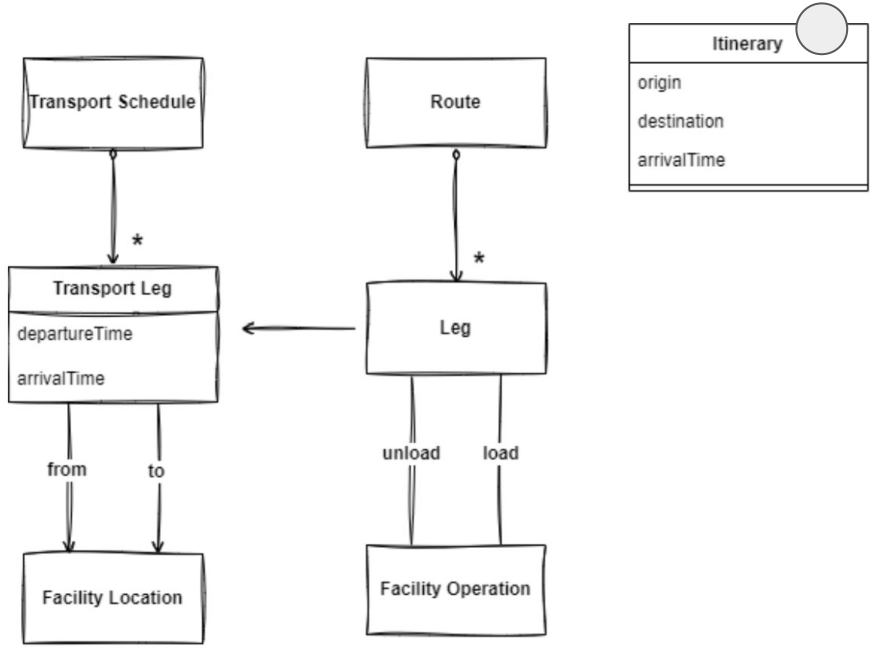
Basic patterns


Domain
Aggregate 1
Entity
Entity
Value Object
Aggregate 2
Entity
Value Object
Value Object
Aggregate 3
Domain Event

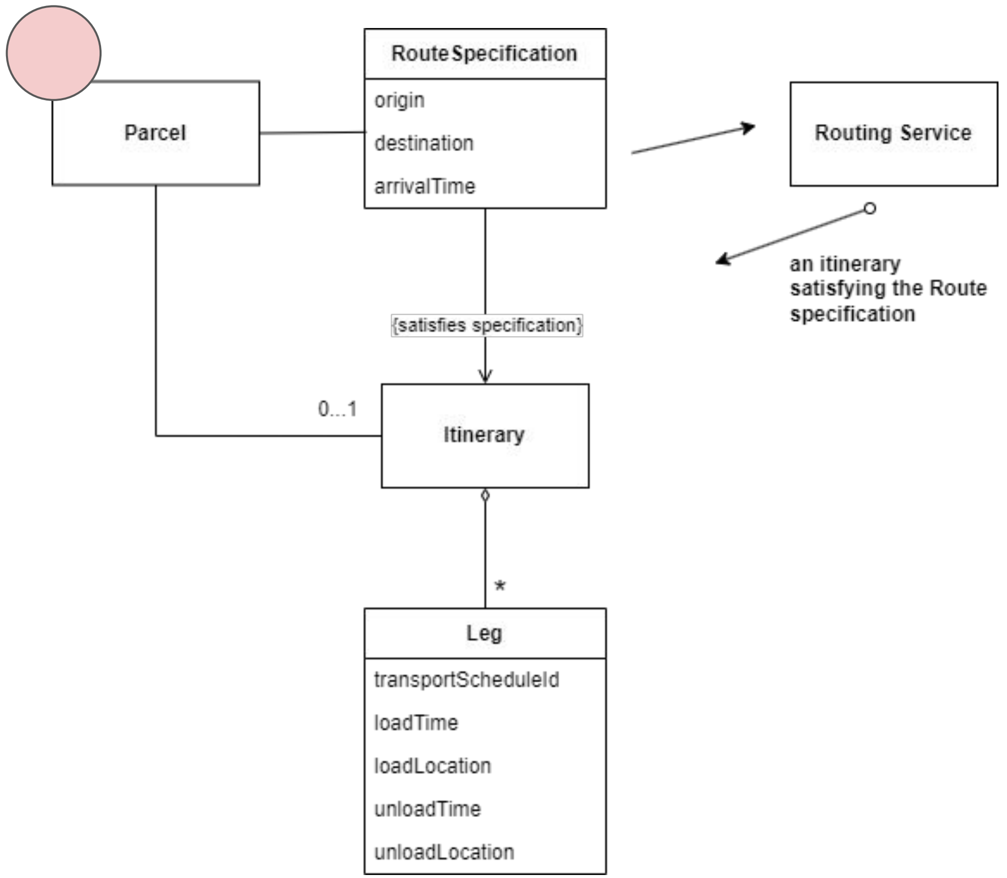

Salt Lake City
Louisville
Philadelphia
Düsseldorf
Cologne
Mannheim

Power Words

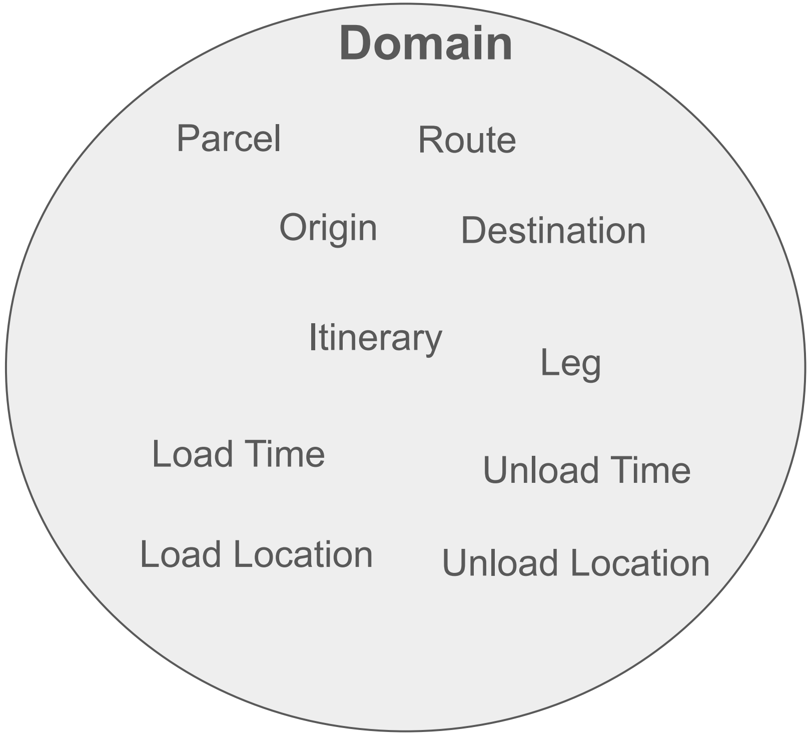
Parcel
Route
Leg
Facility Operation
Transport Leg
Load Time
Origin
Unload Time
Arrival Time
Departure Time
Itinerary
Facility Location
Destination
Unload Location
Load Location



Aggregate



Parcel
Route Specification
origin
destination
arrivalTime
Routing Service
an itinerary satisfying the Route specification
{satisfies specification}
0...1
Itinerary
*
Leg
transportScheduleId
loadTime
loadLocation
unloadTime
unloadLocation



Domain
Parcel
Route
Origin
Destination
Itinerary
Leg
Load Time
Unload Time
Load Location
Unload Location

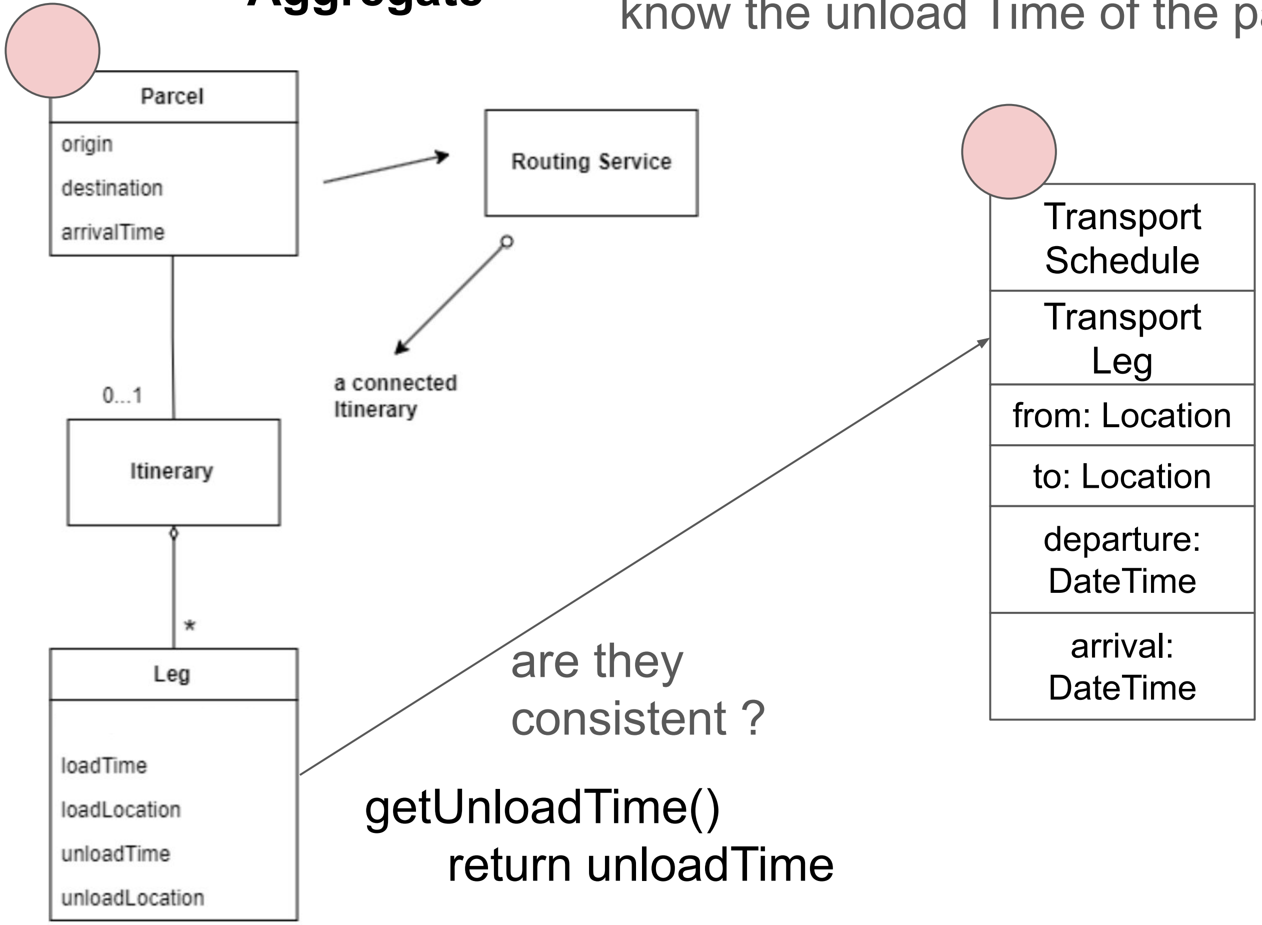
Design Moment

Aggregate Rules:

- **Reference other aggregates by id**
- Changes are committed and rolled back as a hole
- Changes to an aggregate are done via the root

Aggregate

use case: the Operation users would like to know the unload Time of the parcel.

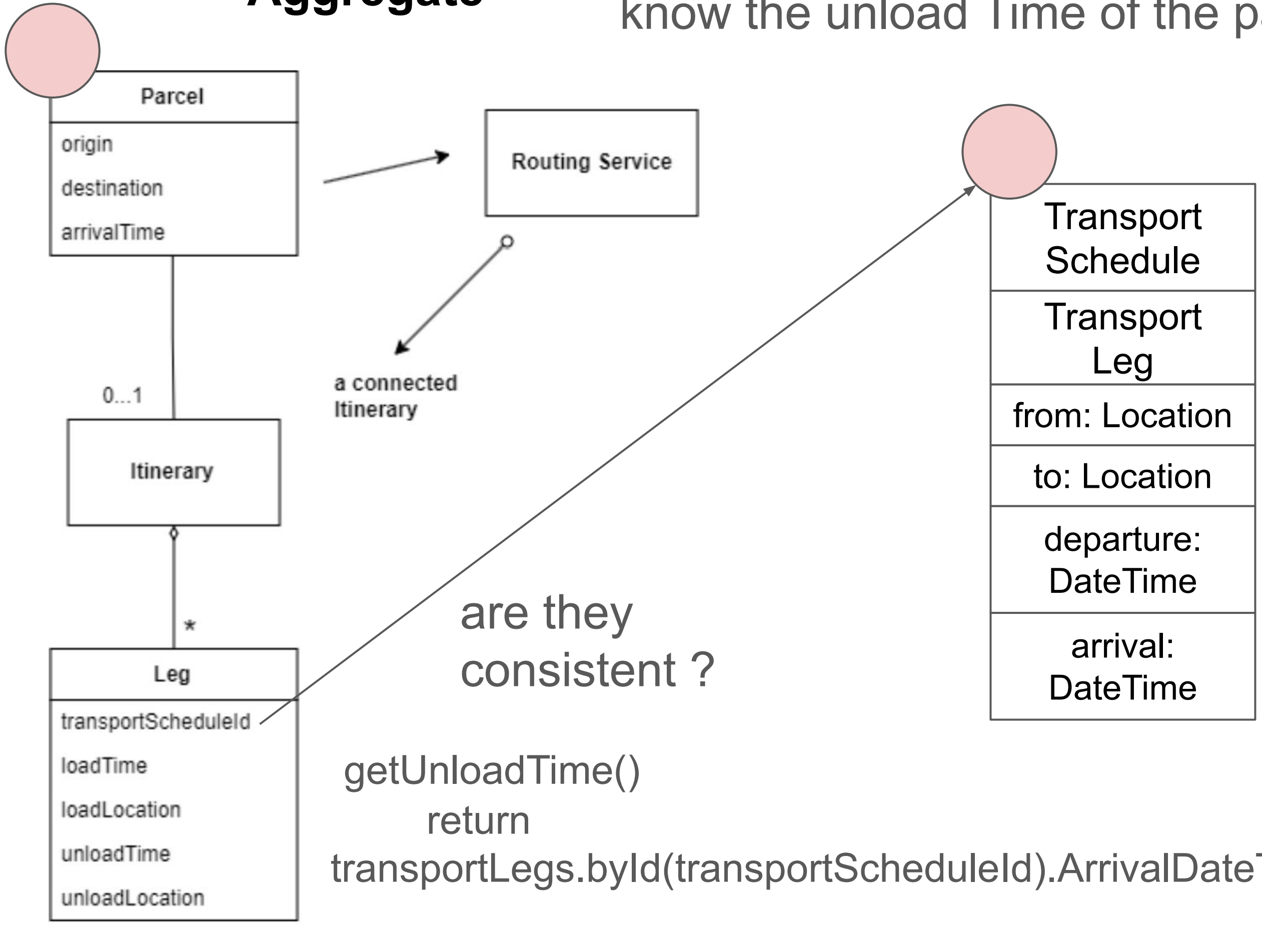


are they consistent ?

```
getUnloadTime()
    return unloadTime
```

Aggregate

use case: the Operation users would like to know the unload Time of the parcel.



are they consistent ?

```
getUnloadTime()
    return
transportLegs.byId(transportScheduleId).ArrivalDateTime()...
```

The
Collaborative Modeling
Unconference

COMO
CAMP

Vienna
May 7th-10th, 2025

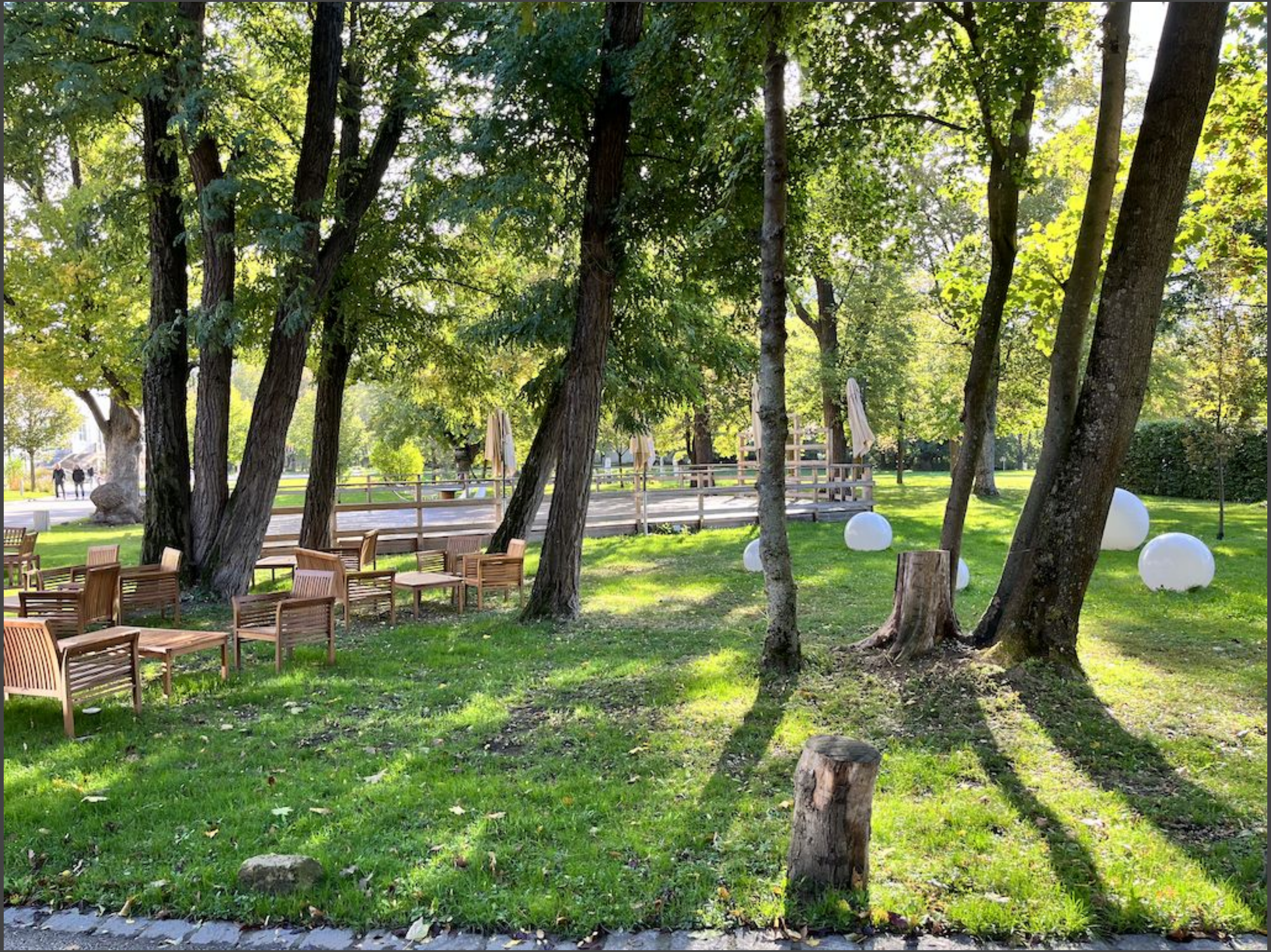
COMO
IDEAS FOR Tomorrow
LUNCH
LUNCH
LUNCH
LUNCH
LUNCH
LUNCH
LUNCH
EUROPAHAUS
WWW.EUROPAHAUS

Richard
COMO CAMP

Rocket launched to Orbit
Additional funding secured
Seed secured
Mission crew selected
Spacecraft built
Astronauts selected
COMO CAMP

OPAHAUS WIEN

OUTDOOR
A
COMO
COMO CAMP



ROADS? WHERE WE'RE GOING WE DON'T NEED ROADS
COMO CAMP

The
Collaborative Modeling
Unconference

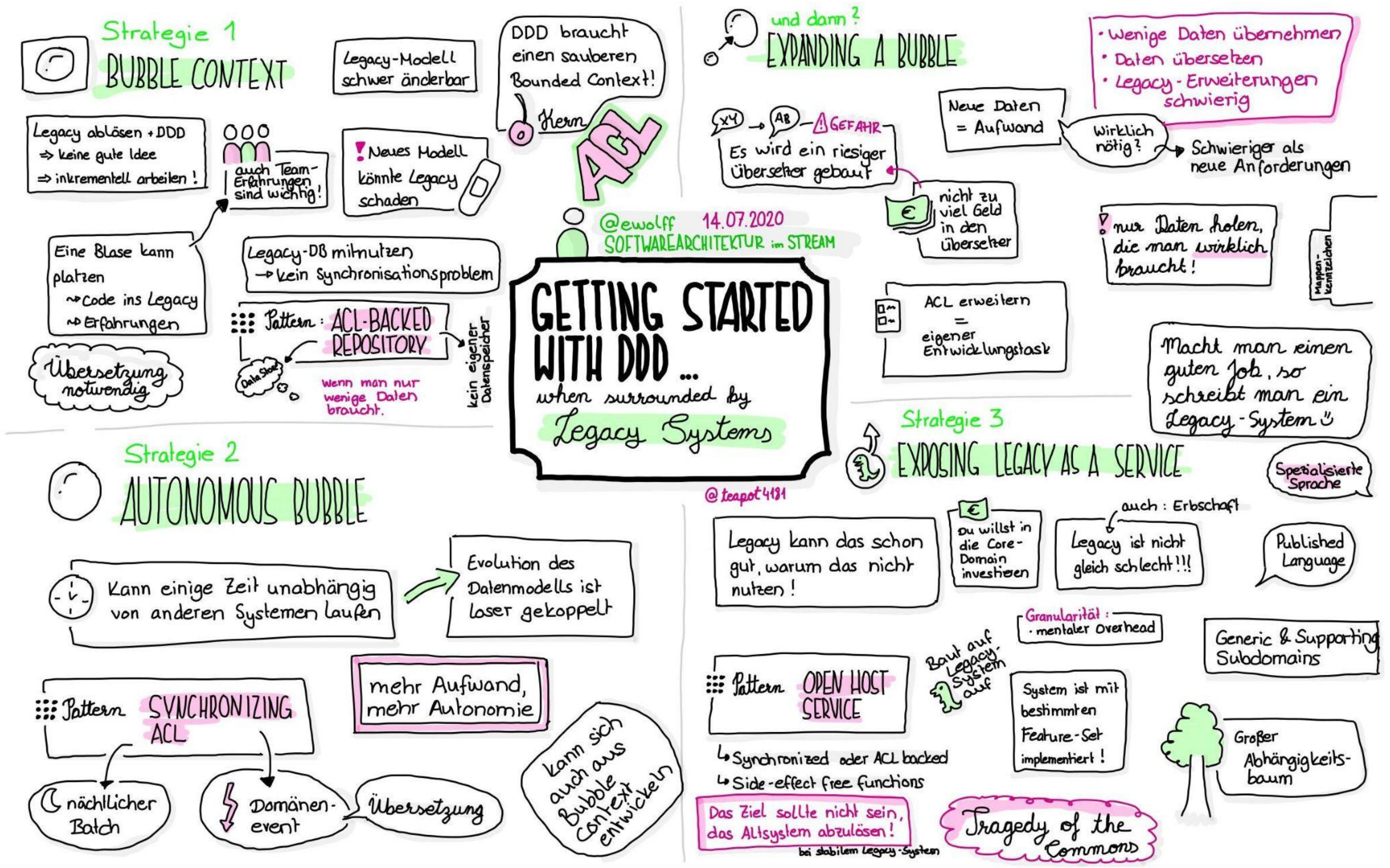
COMO CAMP

Vienna
May 7th-10th, 2025

Getting started with DDD when surrounded by Legacy Systems

Revisiting a 2013 paper by Eric Evans

Christoph Baudson


GETTING STARTED WITH DDD ...
when surrounded by
Legacy Systems
@ewolff 14.07.2020
SOFTWAREARCHITEKTUR im STREAM
@teapot4181
Strategie 1
BUBBLE CONTEXT
Legacy-Modell schwer änderbar
DDD braucht einen sauberen Bounded Context!
Kern
ACL
Legacy ablösen + DDD
⇒ keine gute Idee
⇒ inkrementell arbeiten!
auch Team-Erfahrungen sind wichtig!
! Neues Modell könnte Legacy schaden
Eine Blase kann platzen
↝ Code ins Legacy
↝ Erfahrungen
Legacy-DB mitnutzen
→ kein Synchronisationsproblem
Pattern: ACL-BACKED REPOSITORY
Data Store
kein eigener Datenspeicher
wenn man nur wenige Daten braucht.
Übersetzung notwendig
Strategie 2
AUTONOMOUS BUBBLE
Kann einige Zeit unabhängig von anderen Systemen laufen
Evolution des Datenmodells ist loser gekoppelt
Pattern SYNCHRONIZING ACL
mehr Aufwand, mehr Autonomie
nächtlicher Batch
Domänen-event
Übersetzung
kann sich auch aus Bubble Context entwickeln
und dann?
EXPANDING A BUBBLE
XY → AB
⚠ GEFAHR
Es wird ein riesiger Übersetzer gebaut
nicht zu viel Geld in den Übersetzer
Neue Daten = Aufwand
Wirklich nötig?
Schwieriger als neue Anforderungen
· Wenige Daten übernehmen
· Daten übersetzen
· Legacy-Erweiterungen schwierig
! nur Daten holen, die man wirklich braucht!
Mappen-kennzeichen
ACL erweitern = eigener Entwicklungstask
Macht man einen guten Job, so schreibt man ein Legacy-System ☺
Strategie 3
EXPOSING LEGACY AS A SERVICE
Spezialisierte Sprache
Legacy kann das schon gut, warum das nicht nutzen!
Du willst in die Core-Domain investieren
auch: Erbschaft
Legacy ist nicht gleich schlecht!!!
Published Language
Granularität:
· mentaler Overhead
Generic & Supporting Subdomains
Pattern OPEN HOST SERVICE
Baut auf Legacy-System auf
↳ Synchronized oder ACL backed
↳ Side-effect free functions
System ist mit bestimmten Feature-Set implementiert!
Großer Abhängigkeitsbaum
Das Ziel sollte nicht sein, das Altsystem abzulösen!
bei stabilem Legacy-System
Tragedy of the Commons

Software Architektur TV Folge 6

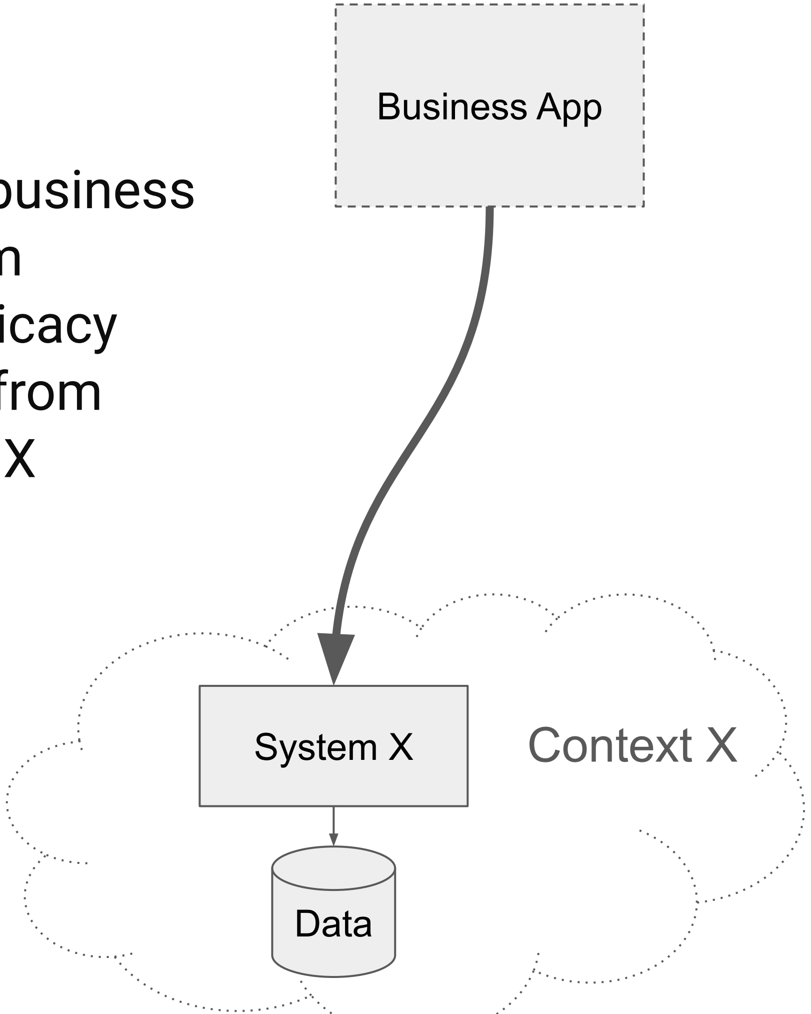
Sketch notes by Lisa Maria Moritz

Core ideas

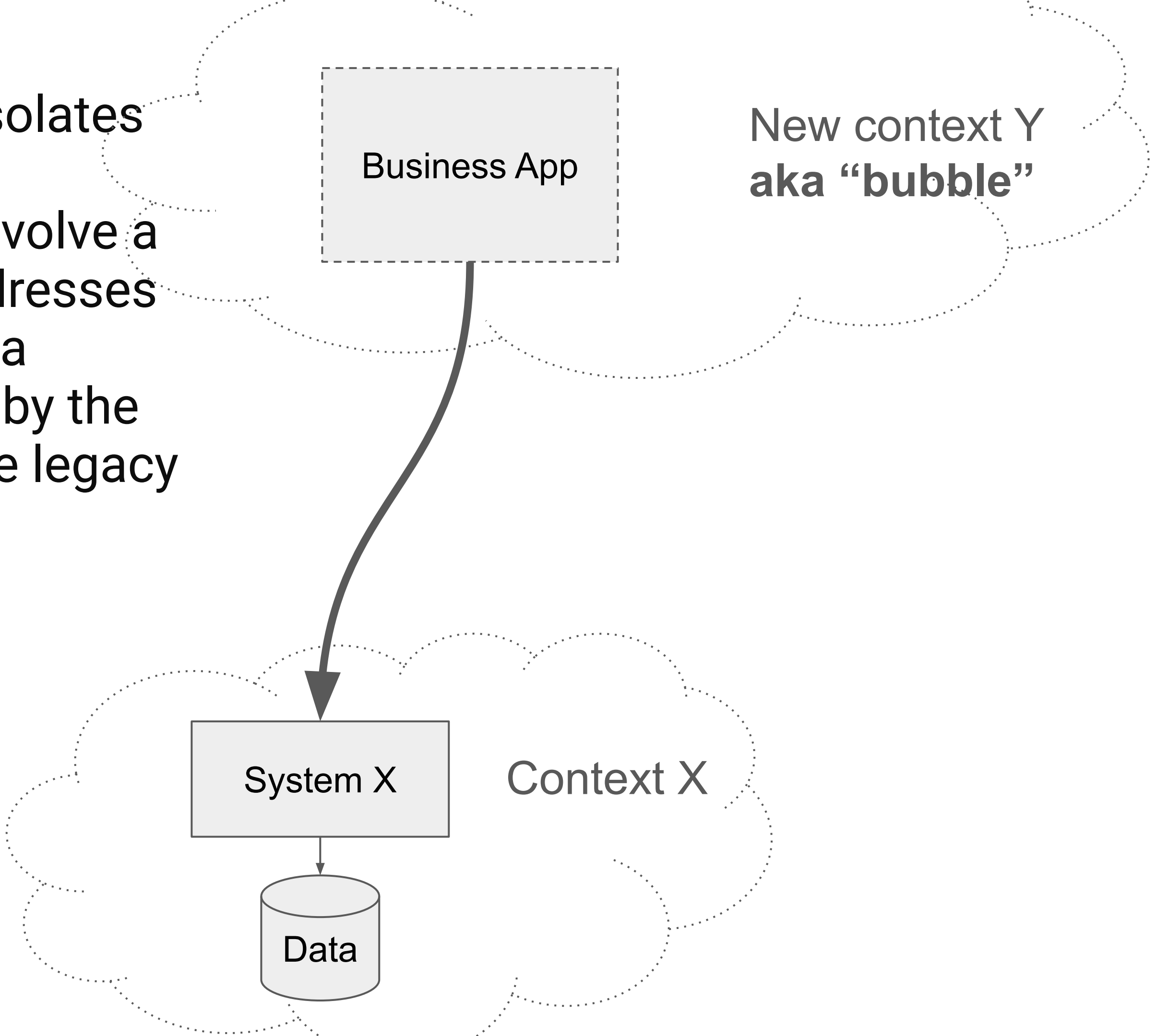
- ❏ Legacy represents value - don’t fall into the trap of replacing legacy systems by default!
- ❏ Make it easy for people to apply DDD patterns in brownfield situations!
- ❏ Remember some solutions presented here are temporary or steps in an evolution!

Strategy 1:
Bubble context

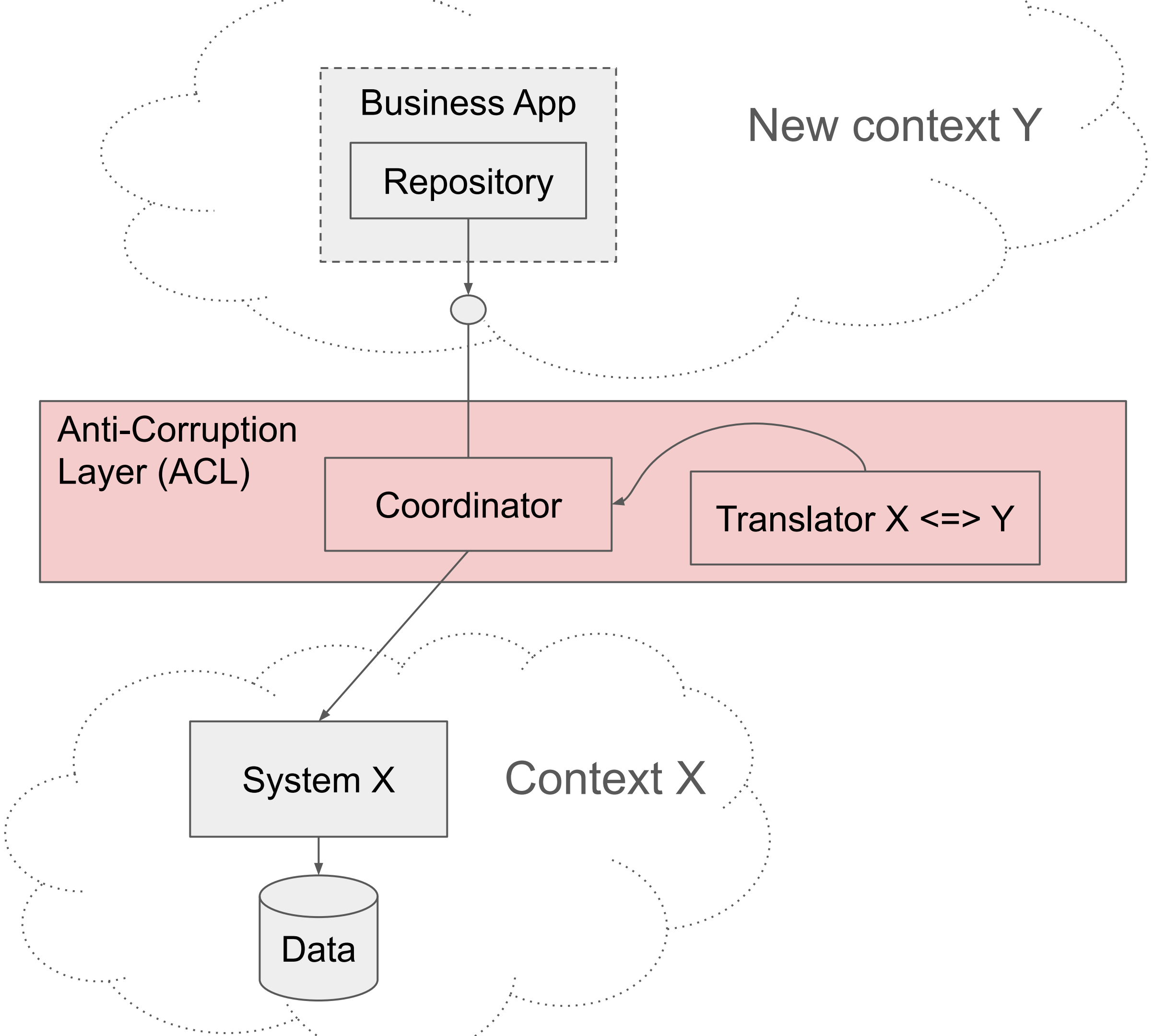
- ❏ Solution to an important, yet modest-sized business related problem
- ❏ With some intricacy
- ❏ Requires data from legacy system X



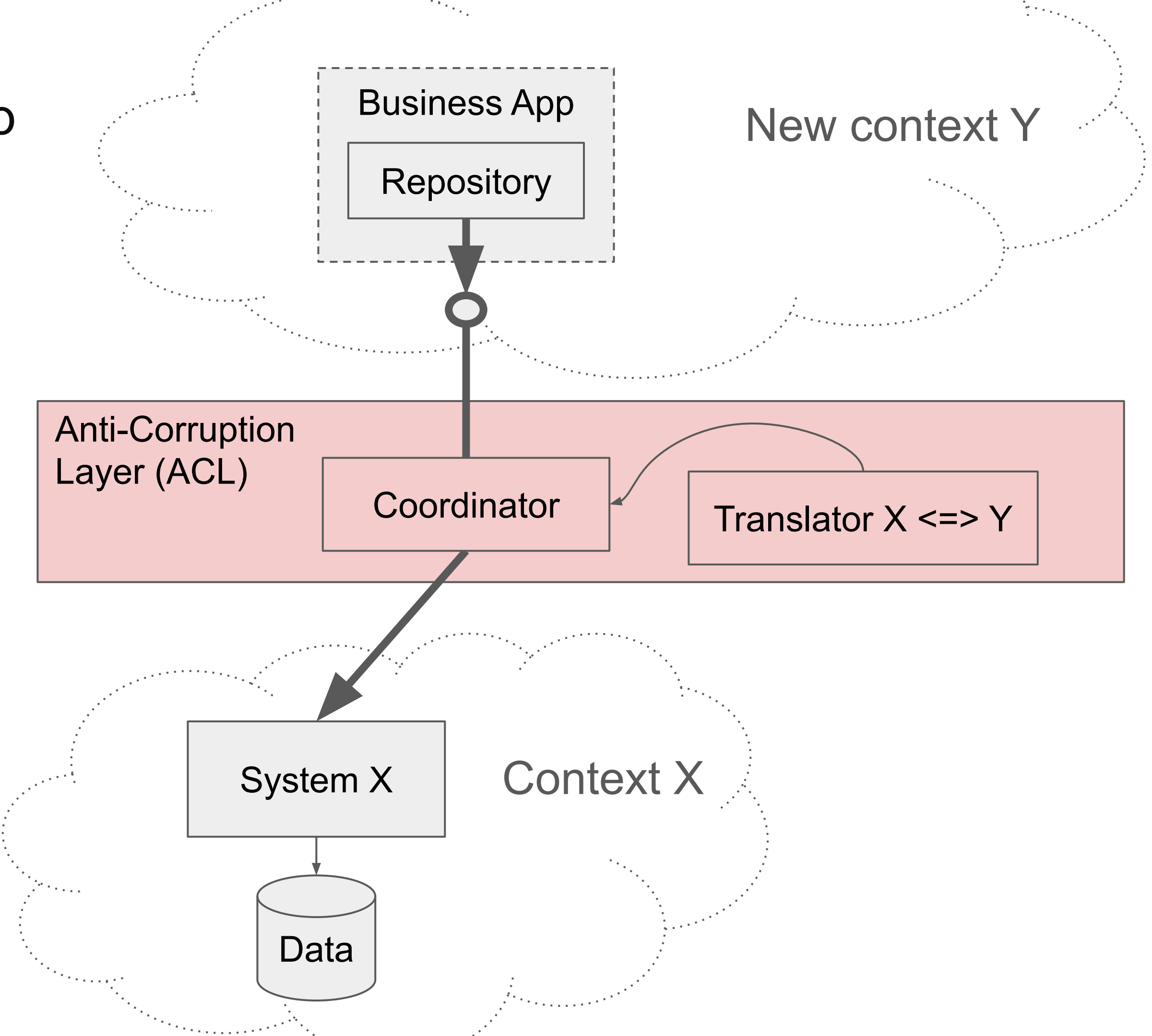
- The “bubble” isolates that work
- the team can evolve a model that addresses the chosen area
- unconstrained by the concepts of the legacy systems.



- Create the **illusion** that bubble context Y has its own data store
- No new DB in the context Y, query the legacy DB **synchronously** through ACL
- translate data in the ACL into Y's model

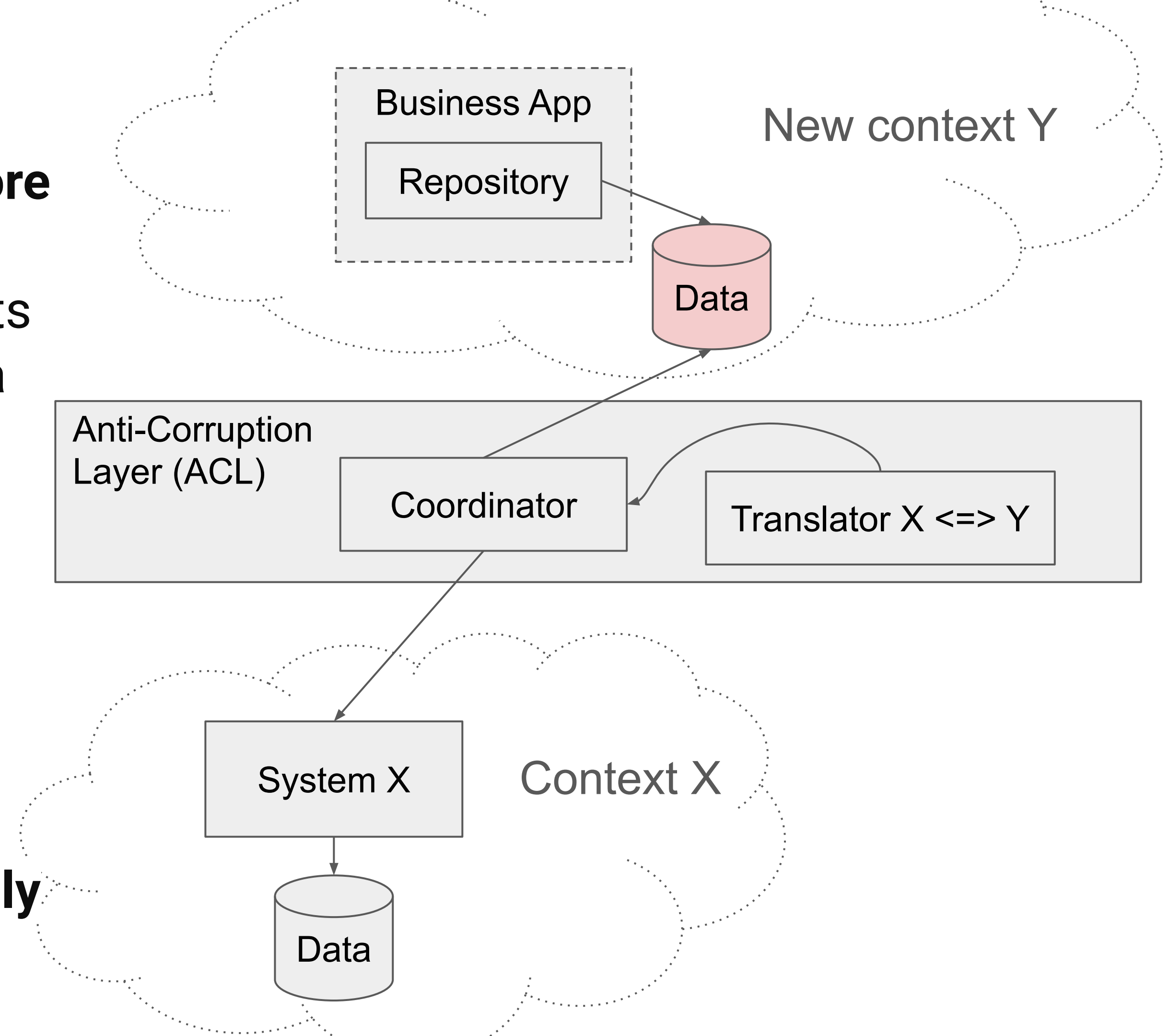


- ACL is closer to context Y than to context X
- The bubble context Y is **completely dependent** on the parent context X!
- ACL acts like an **umbilical cord**
- Context Y can't survive alone!



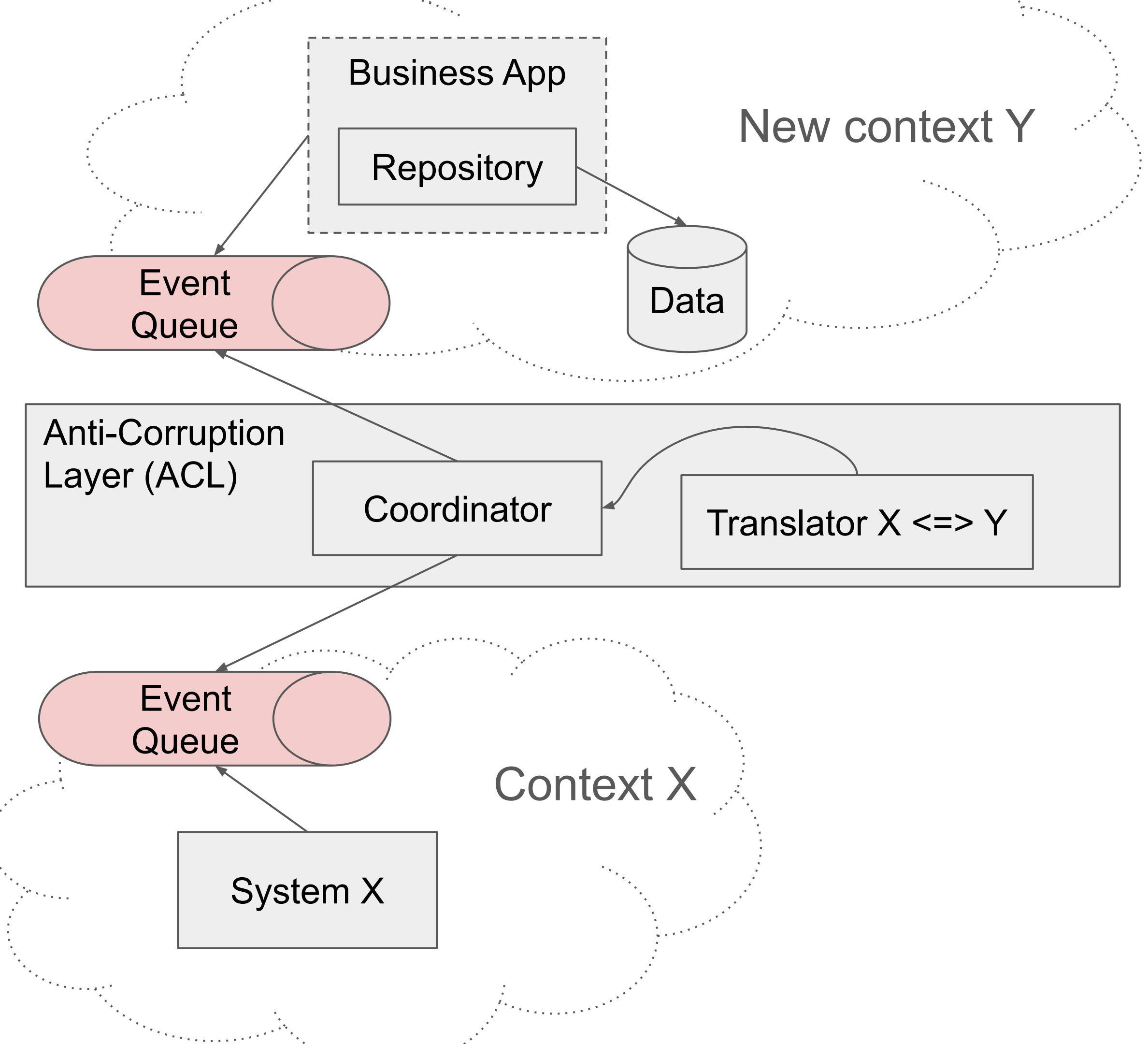
Strategy 2:
Autonomous Bubble

- ❑ Typically has some **data store of its own**
- ❑ Ability to run its software, for a time, **cut off** from other systems
- ❑ ACL syncs between data stores in two contexts **asynchronously**



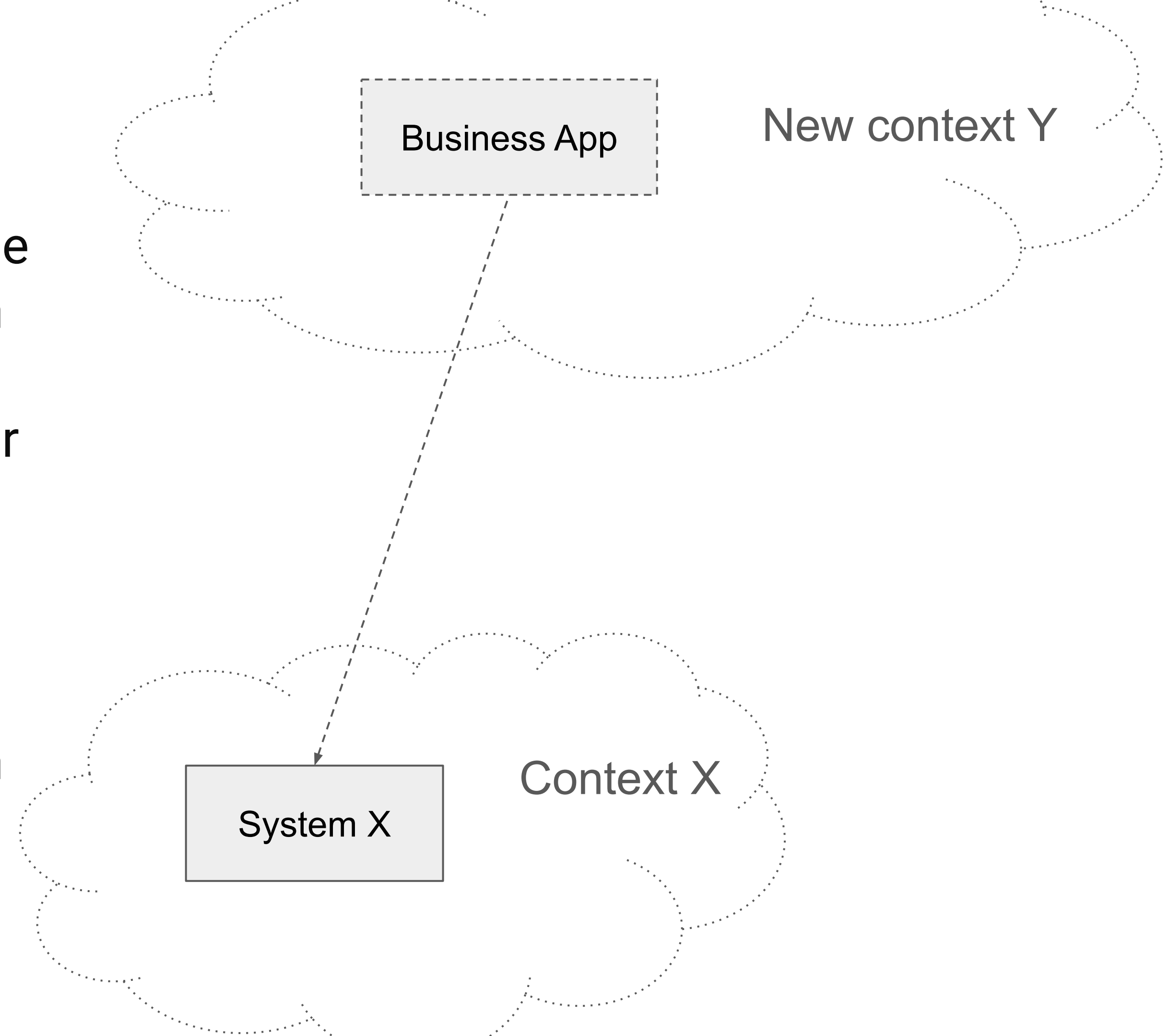
Strategy 2b: Autonomous Bubble with Domain Events

- ❑ Fully asynchronous
- ❑ **Messages are not neutral!** They are always expressed in some language based on a model
- ❑ **Don’t let them enter a context that uses a different model**

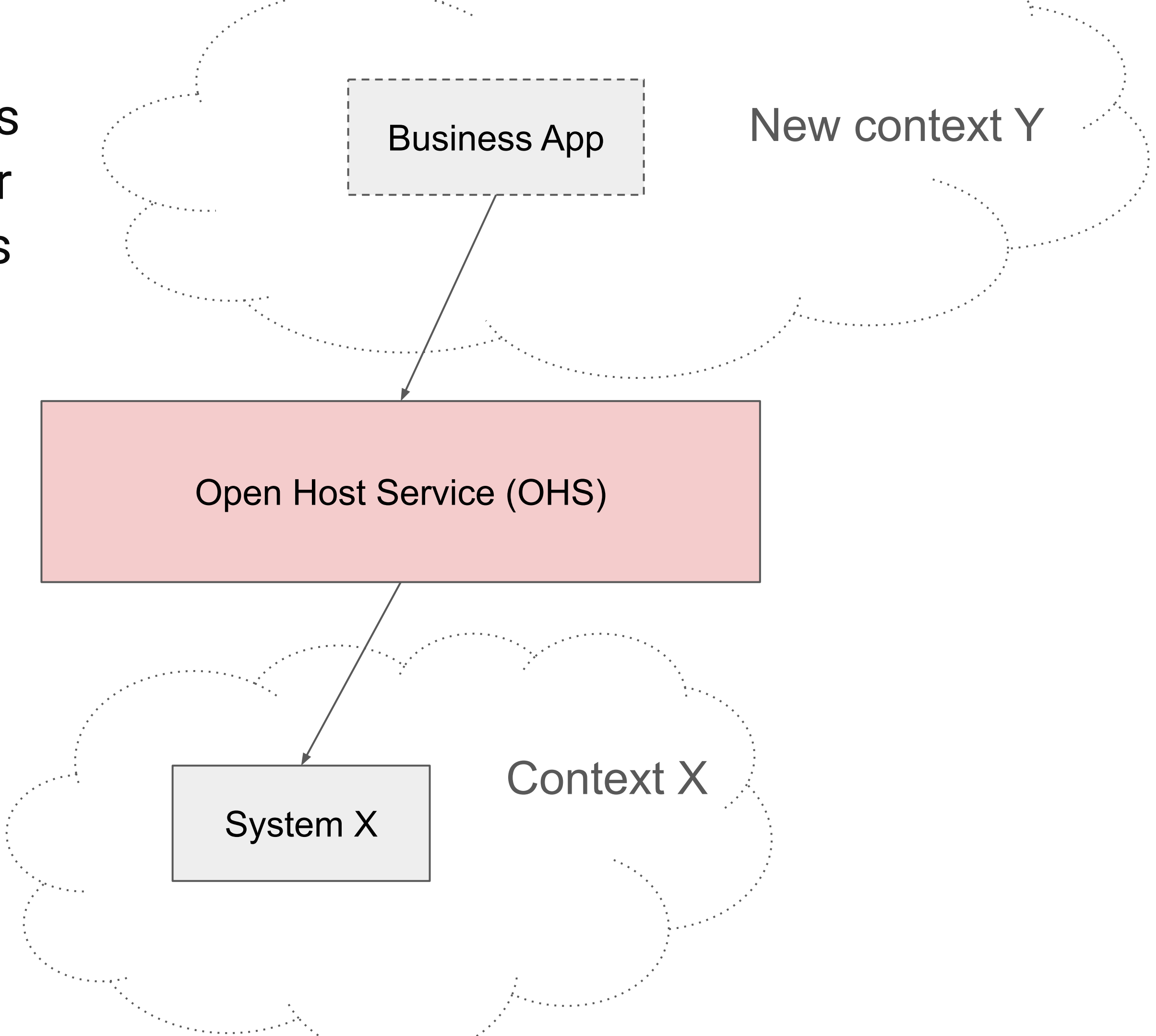


Strategy 3:
Exposing Legacy Assets as a service

- ❏ New context needs data from a valuable legacy system
- ❏ Possibly complicated or inaccessible interfaces
- ❏ No value in replacing the legacy system

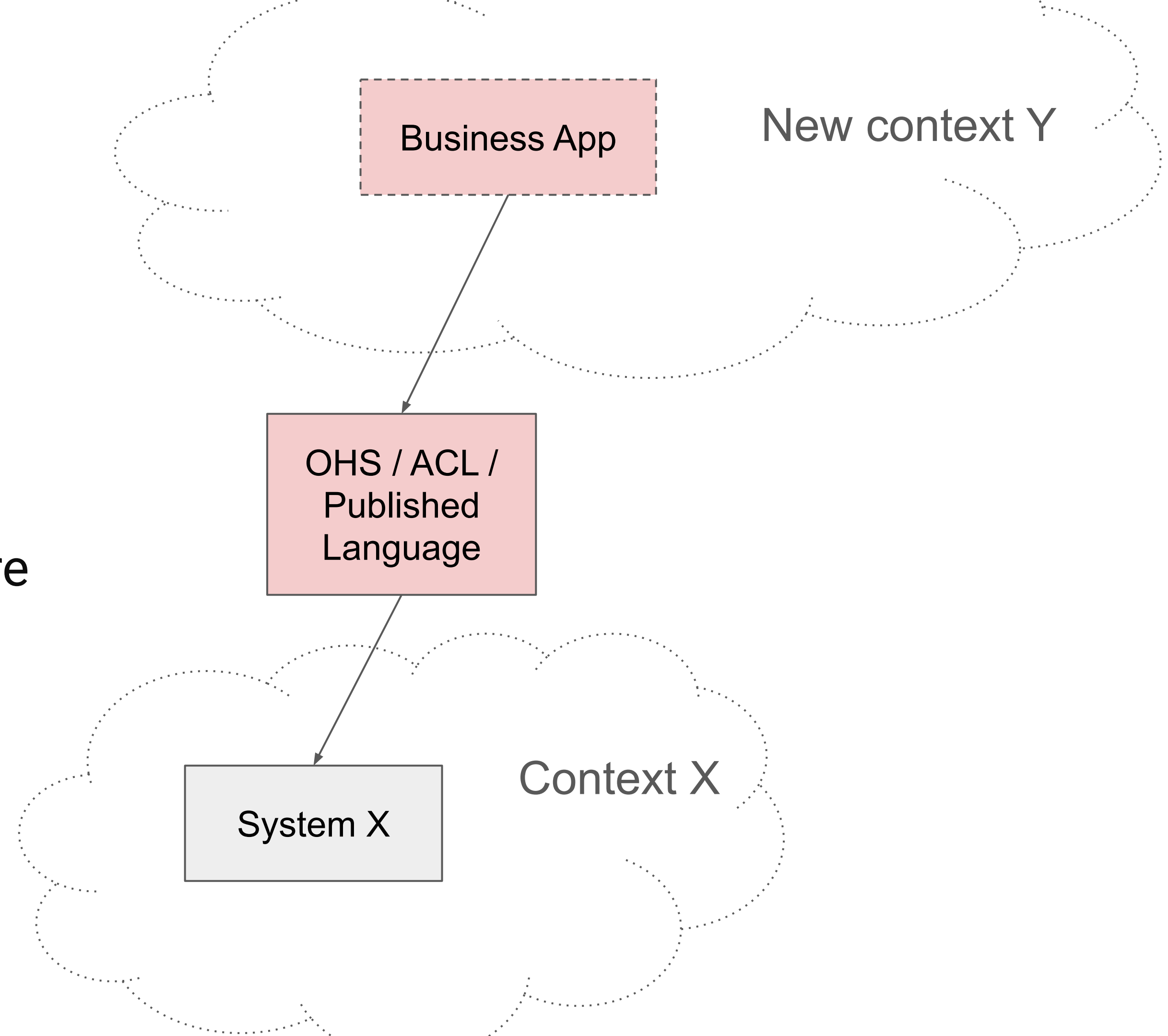


- OHS introduces an interface for newer contexts
- In an OHS, the model is more **general**
- It is closer to context X than to context Y
- Potentially introduces a **Published Language**

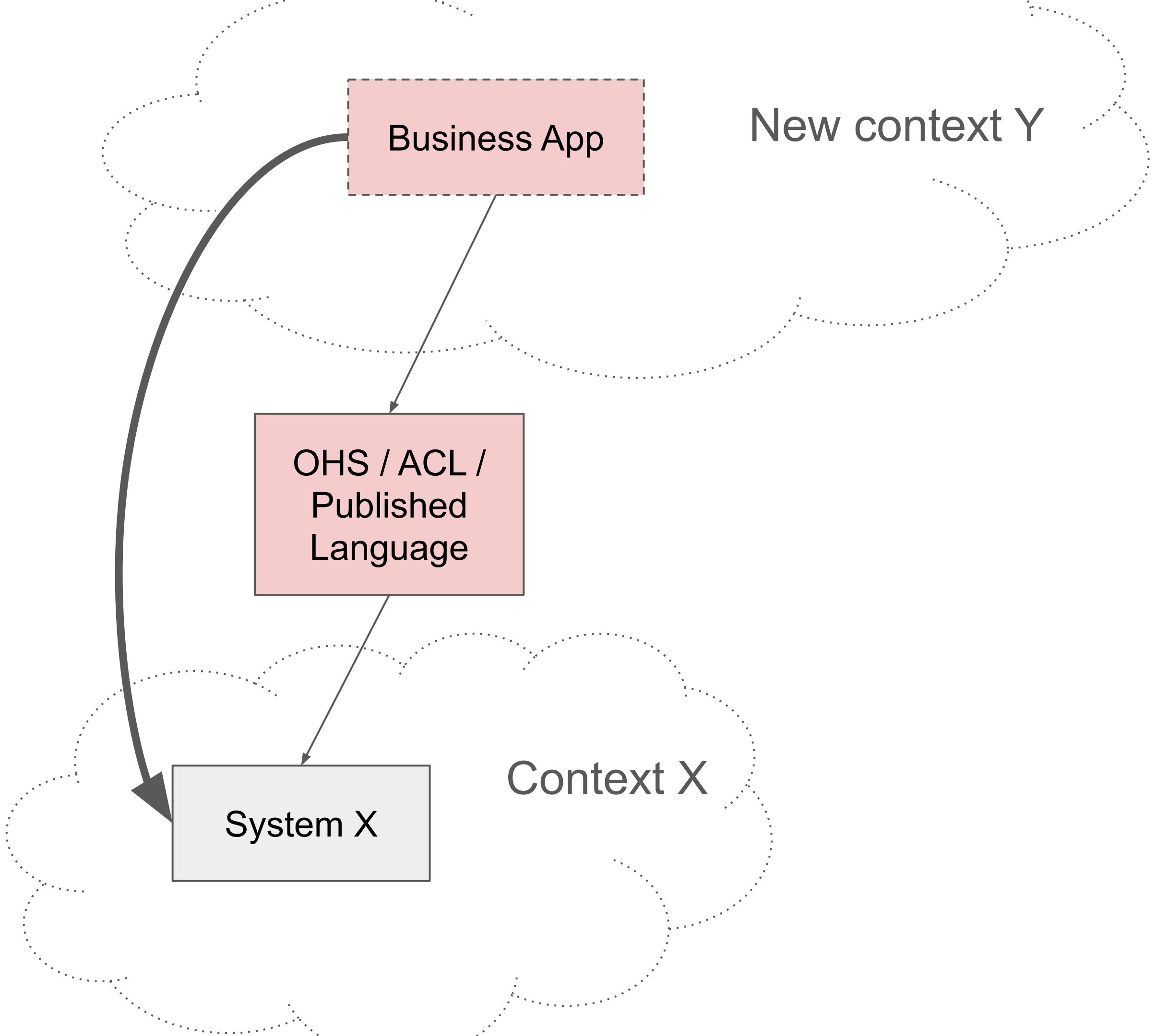


Strategy 4:
Expanding the bubble

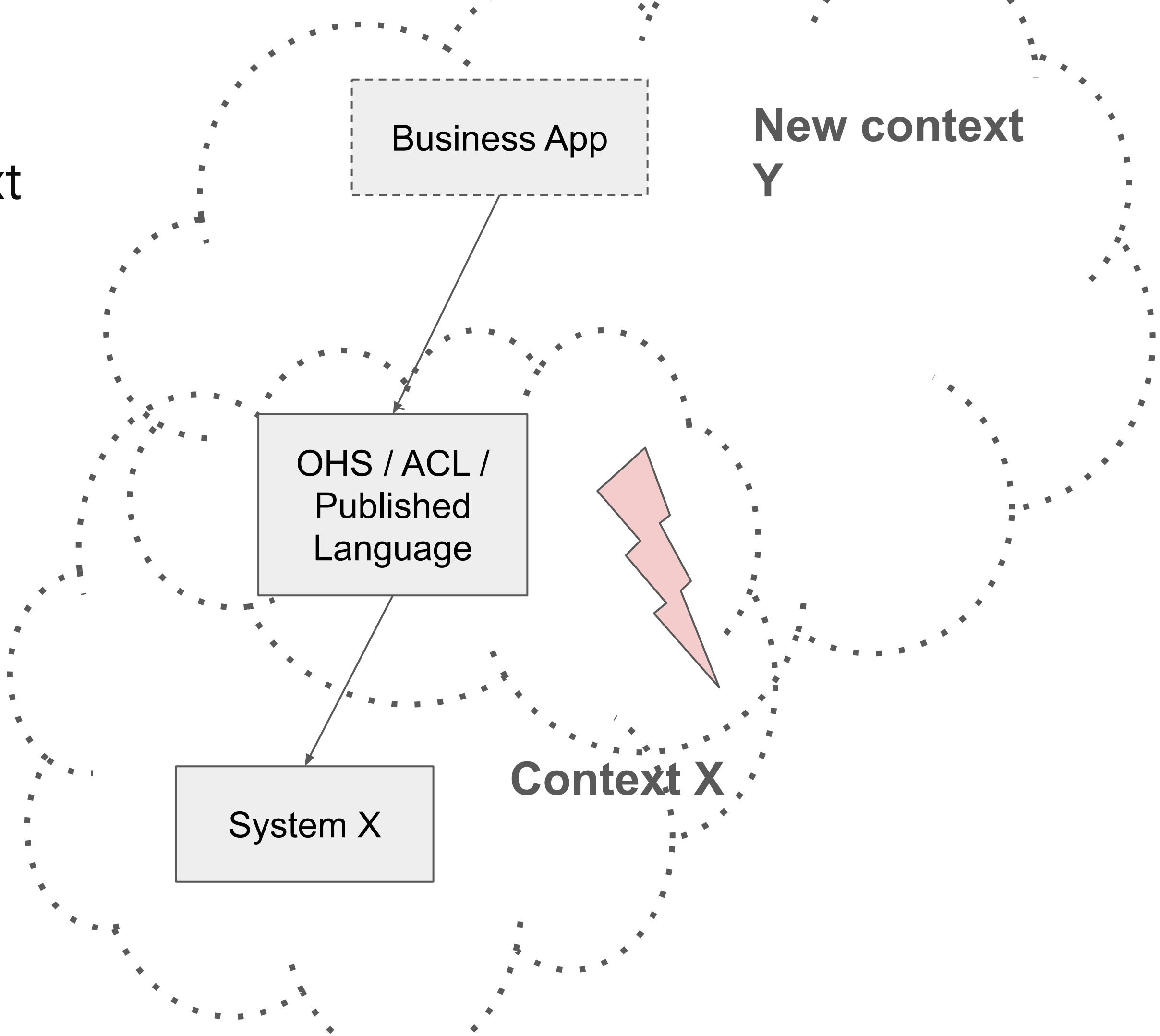
- New context and OHS/ACL both need to evolve
- Often, the OHS/ACL is neglected and new feature are prioritised
- It isn’t just “a new field” - do serious modeling

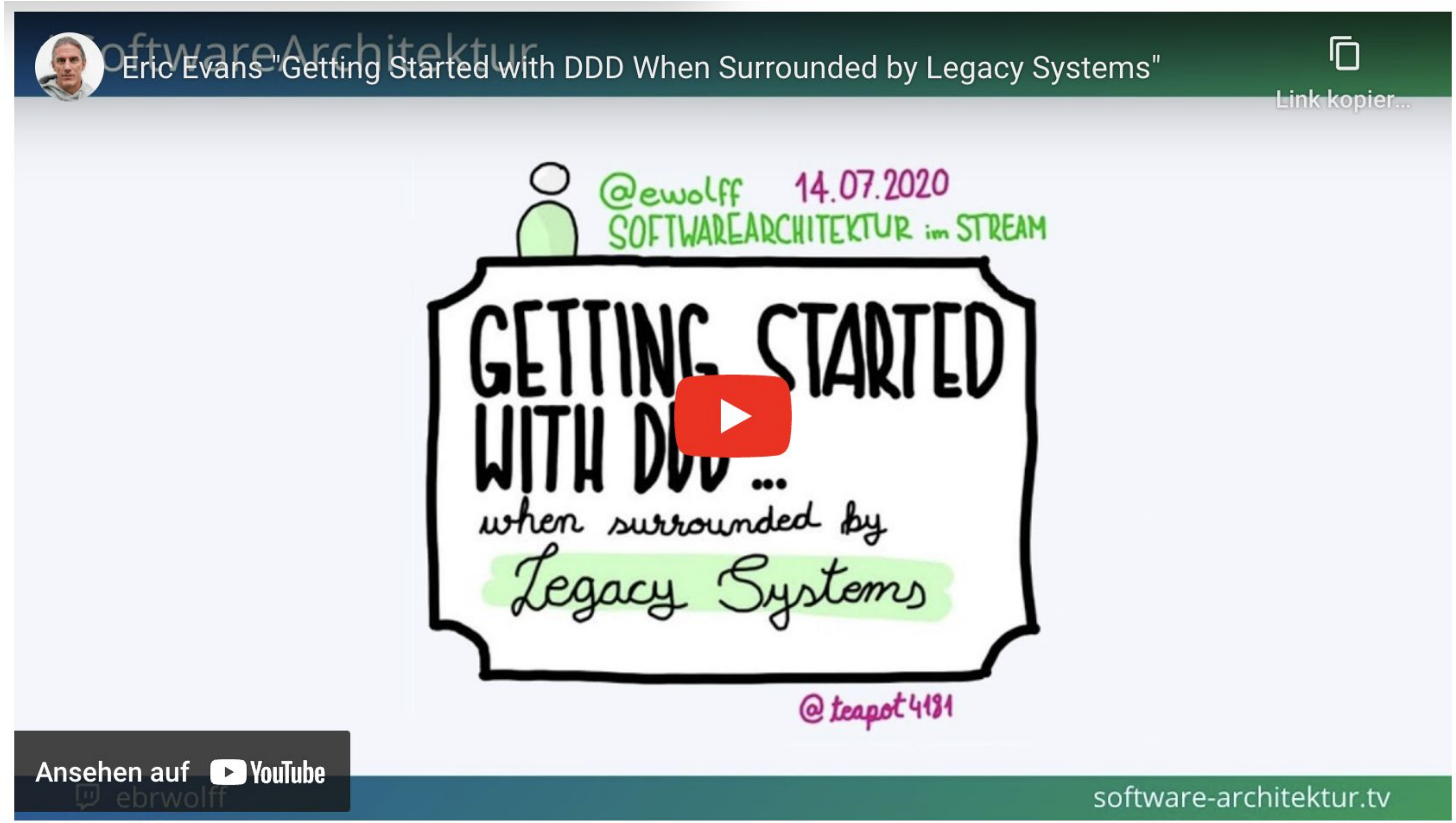


- ❏ If you fail to do this, people might take shortcuts
- ❏ this sabotages the OHS / ACL



- it also leads to eroding context boundaries





Software Architektur TV Folge 6

Sketch notes by Lisa Maria Moritz