# Software Architectures

Domain Driven Design II

## Content

# Terminology

## Ubiquitous Language

- a common, rigorous language between developers and users
- our domain model is based on this language

## Bounded Context

- context related to one specific area (product in warehouse vs in basket)

## Context Map

- diagram that shows different bounded contexts and their connections

## Aggregate

Aggregate is a cluster of domain objects that can be treated as a single unit.
Transactions should not cross aggregate boundaries.

## Domain Event

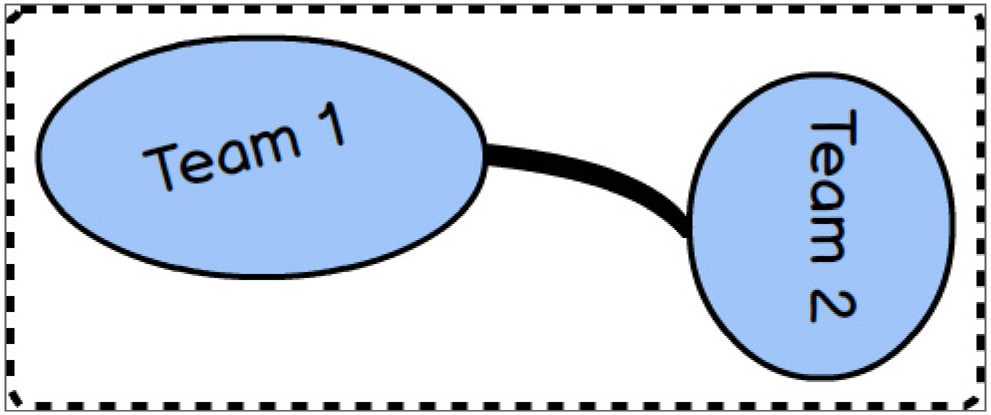
An event is something that has happened in the past.

## Command

Command is a instuction to perform an operation between aggregates.

## Strategic Design with Context Mapping

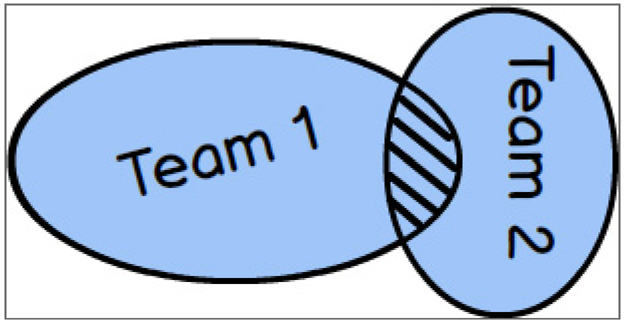
Partnership



## Partnership

- align the two teams with a dependent set of goals
- he two teams will succeed or fail together
- they will meet frequently to synchronize schedules and dependent work
- synchronization is represented by the thick mapping line between the two teams
- challenging to maintain a Partnership over the long term
- should last only as long as it provides an advantage
- should be remapped to a different relationship when the advantage is gone

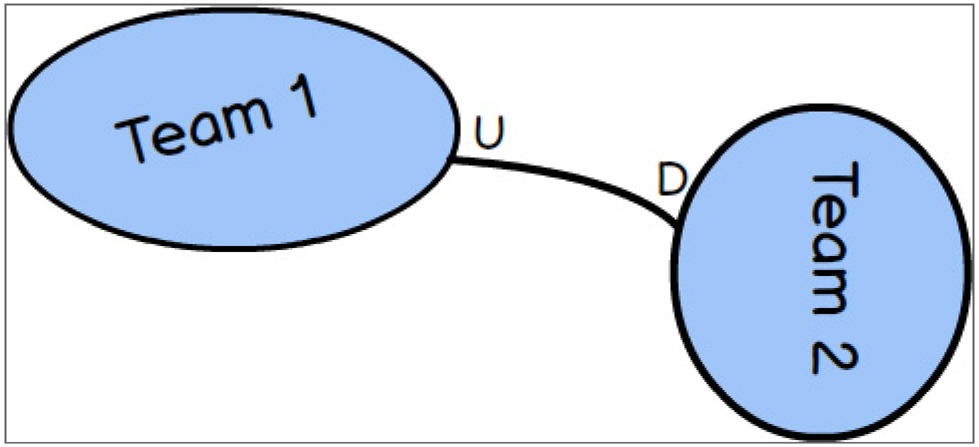
Shared Kernel



## Shared Kernel

- intersection of the two Bounded Contexts
- describes the relationship between two (or more) teams that share a small but common model
- possible that only one of the teams will maintain the code, build, and test for what is shared
- often very difficult to conceive, and difficult to maintain
- must have open communication between teams and constant agreement

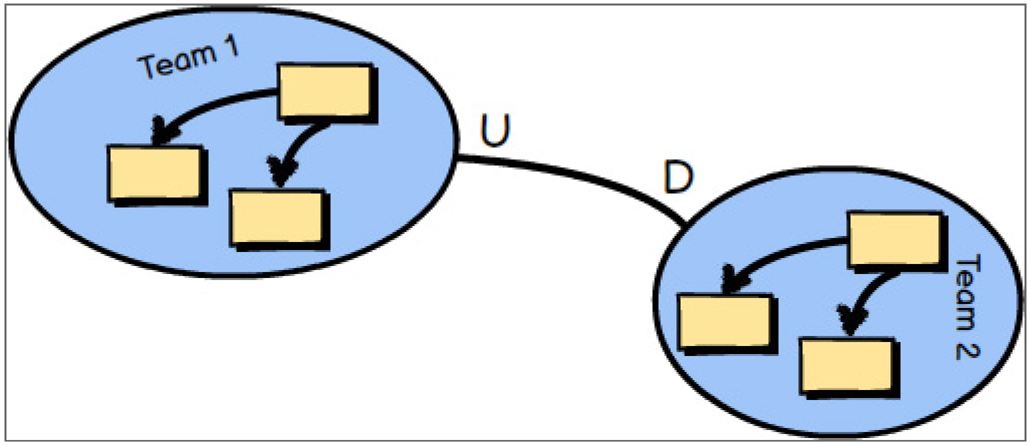
Customer-Supplier



## Customer-Supplier

- very typical and practical relationship between teams
- Supplier is upstream (the U in the diagram) and the Customer is downstream (the D in the diagram)
- It’s up to the Customer to plan with the Supplier to meet various expectations
- in the end the Supplier determines what the Customer will get and when
- works as long as corporate culture does not allow the Supplier to be completely autonomous and unresponsive to the real needs of Customers

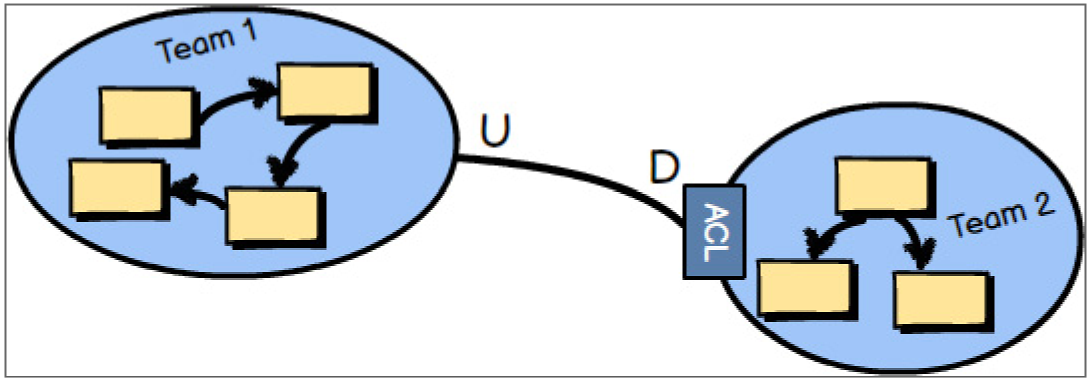
Conformist



## Conformist

- the upstream team has no motivation to support the specific needs of the downstream team
- downstream team cannot sustain an effort to translate the Ubiquitous Language of the upstream model to fit its specific needs
- so the team conforms to the upstream model as is
- for example, when integrating with a very large and complex model that is well established

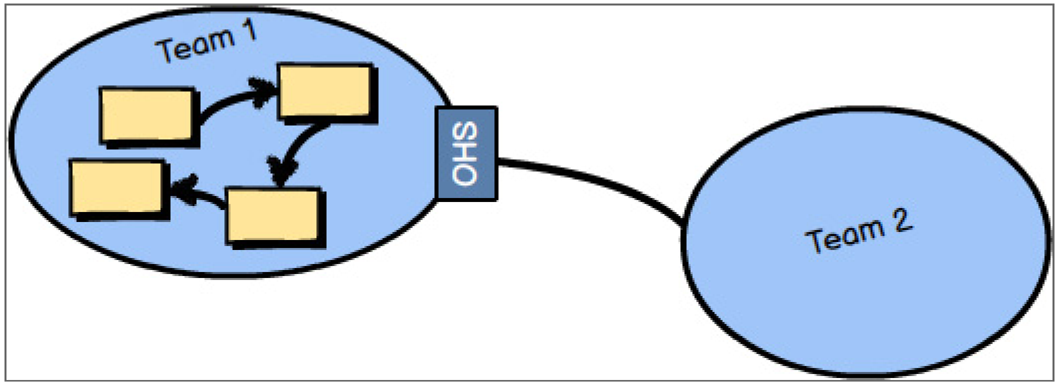
# Anticorruption Layer



## Anticorruption Layer

- the most defensive Context Mapping relationship
- the downstream team creates a translation layer between its Ubiquitous Language (model) and the Ubiquitous Language (model) that is upstream to it
- this is also an approach to integration
- Whenever possible, you should try to create an Anticorruption Layer between your downstream model and an upstream integration model
- keep you completely isolated from foreign concepts

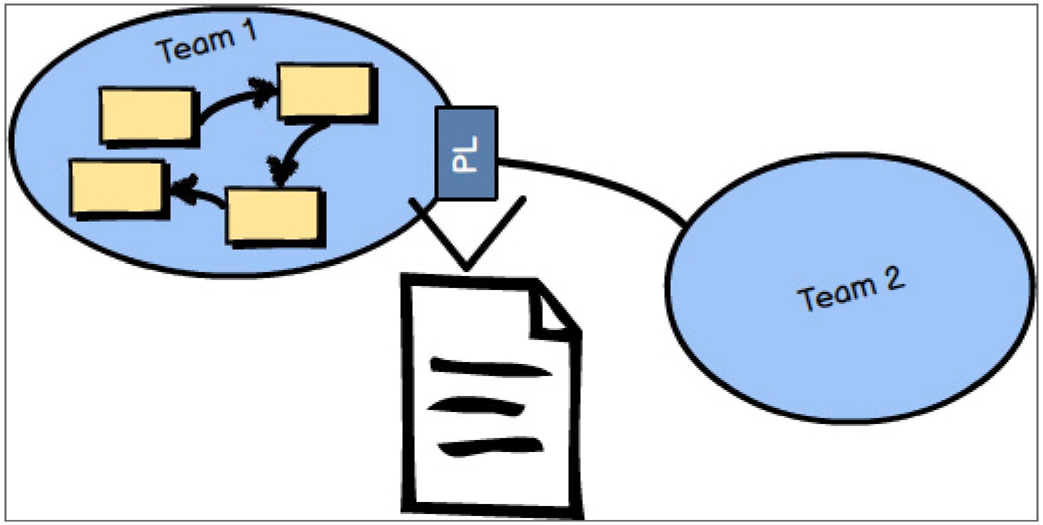
## Open Host Service



## Open Host Service

- Open Host Service defines a protocol or interface that gives access to your Bounded Context as a set of services
- the protocol is “open” so that all who need to integrate with your Bounded Context can use it with relative ease
- services offered by the application programming interface (API) are well documented and a pleasure to use
- it would be much more tolerable to be a Conformist to this model

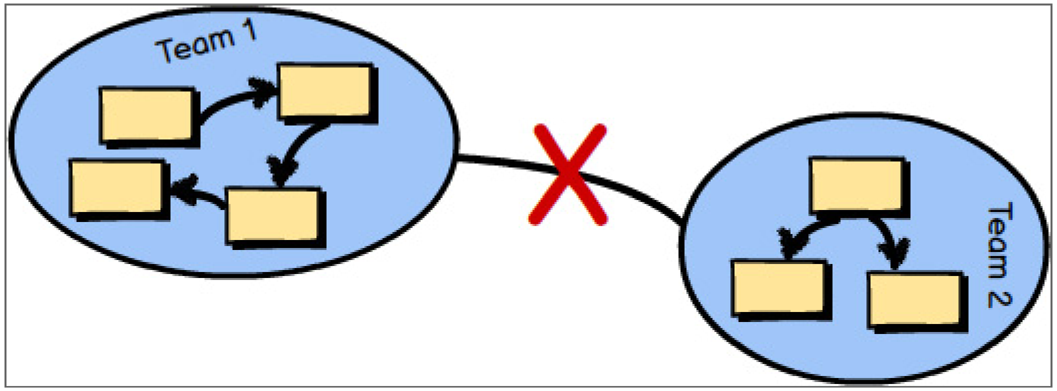
## Published Language



## Published Language

- well-documented information exchange language enabling simple consumption and translation
- can be defined with (XML Schema,) JSON Schema, Swagger, or a more optimal wire format, such as Protobuf or Avro

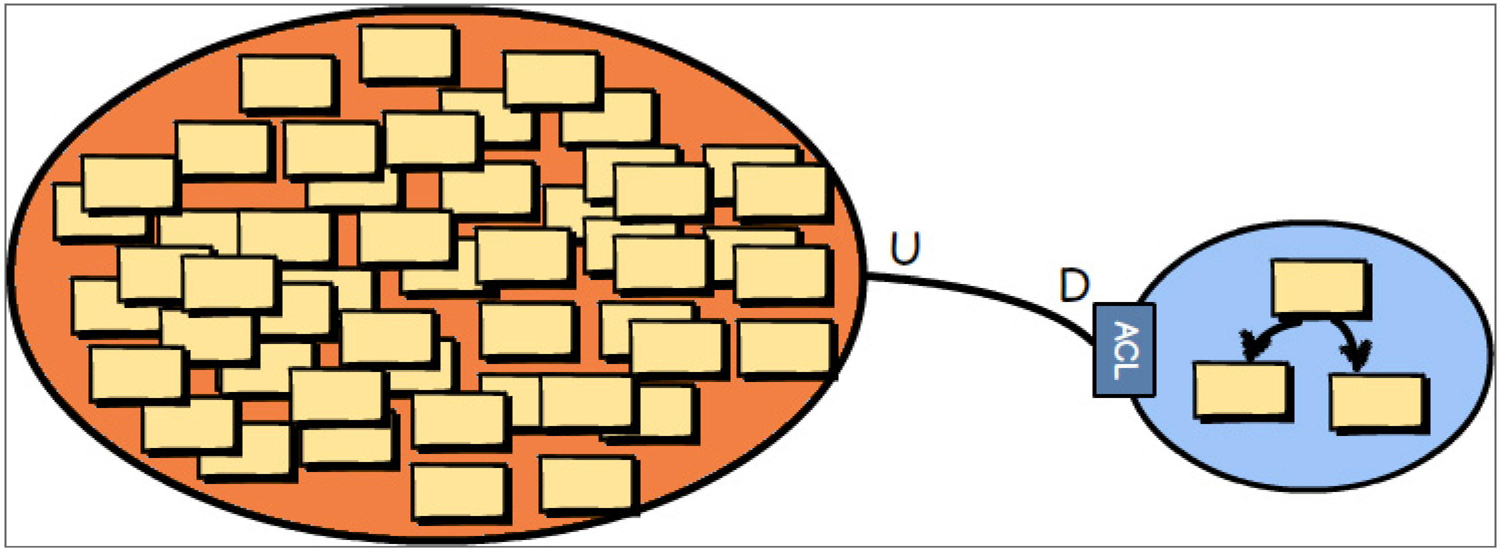
## Separate Ways



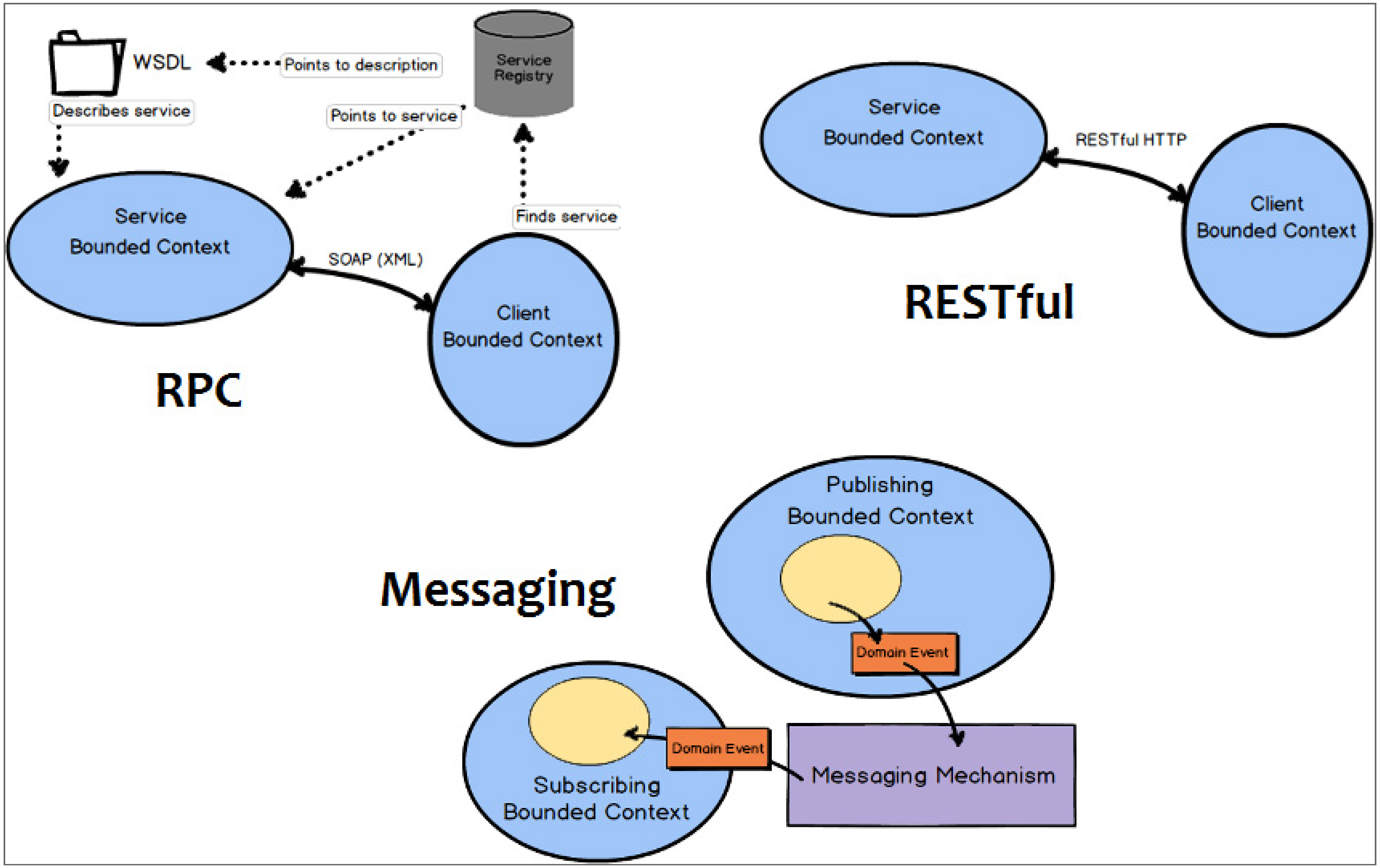
## Separate Ways

- where integration will not produce significant payoff

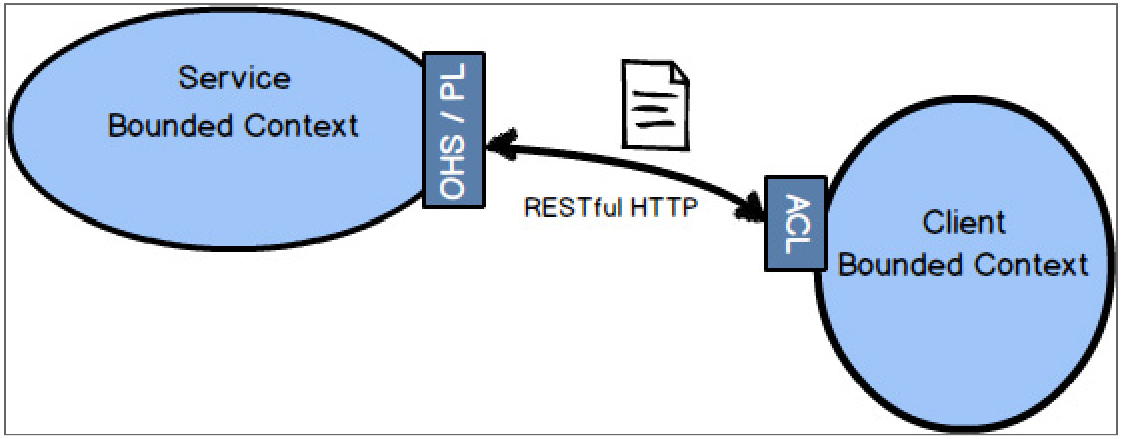
## Integration with Big Ball of Mud



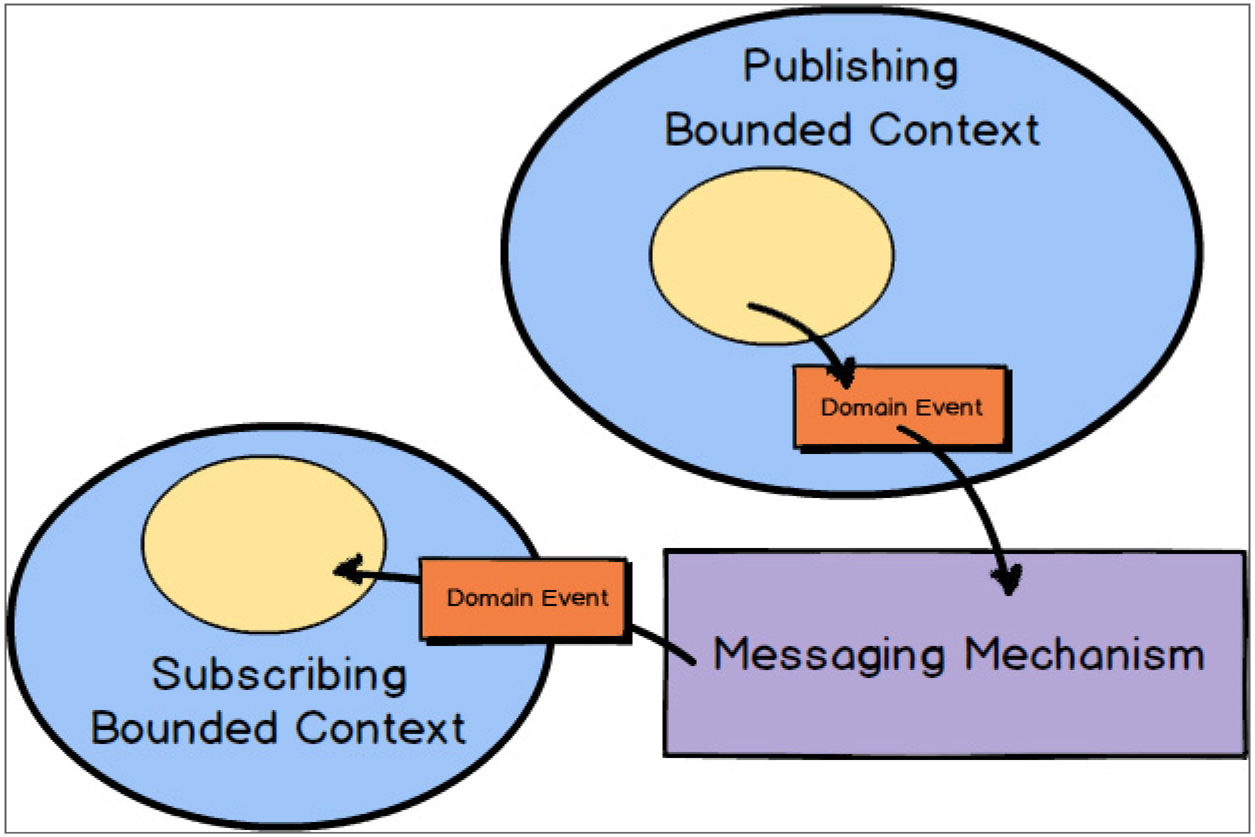
# Differents architectures using Bounded Context



# Modern microservices using DDD



## Messaging architecture



Questions?

That's it for today.

Speaker notes