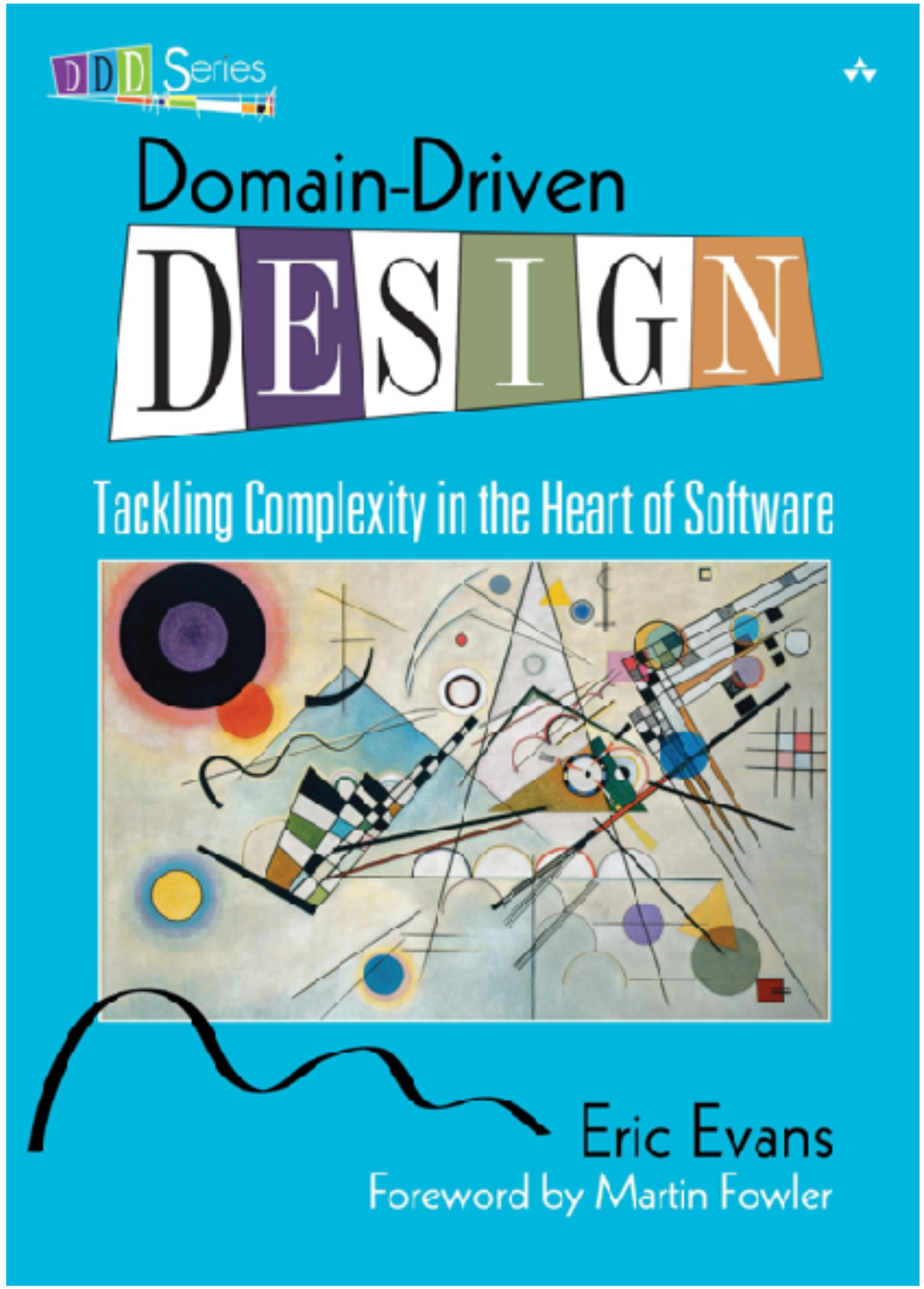
สยามชำนาญกิจ
SIAM CHAMNAN KIT

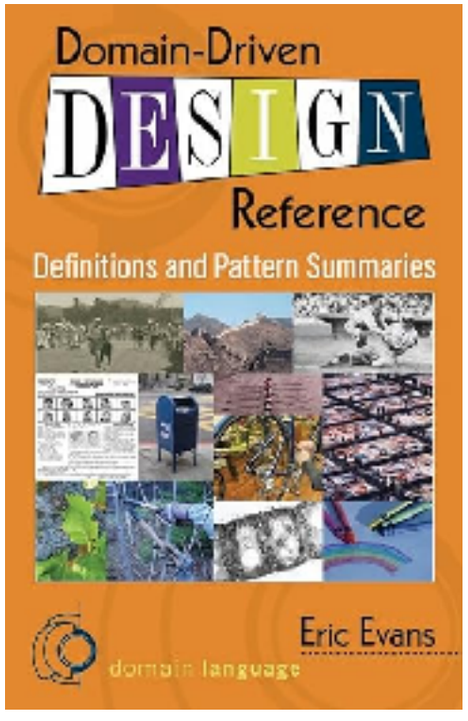
Microservices

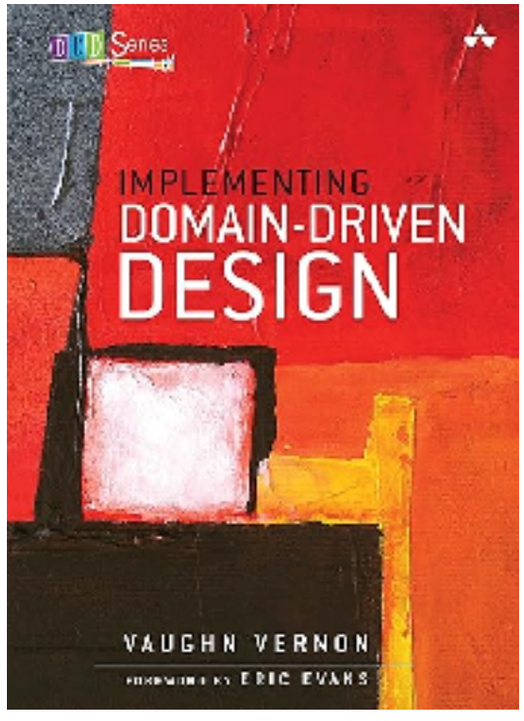
DDD

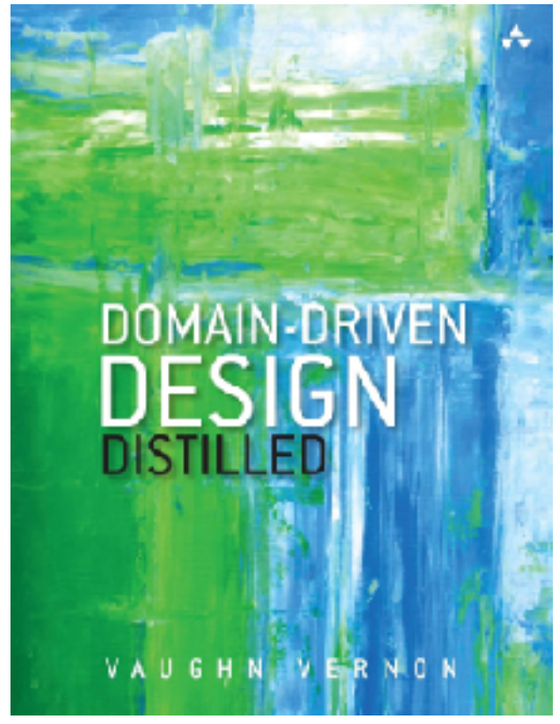
Deadline Driven Design

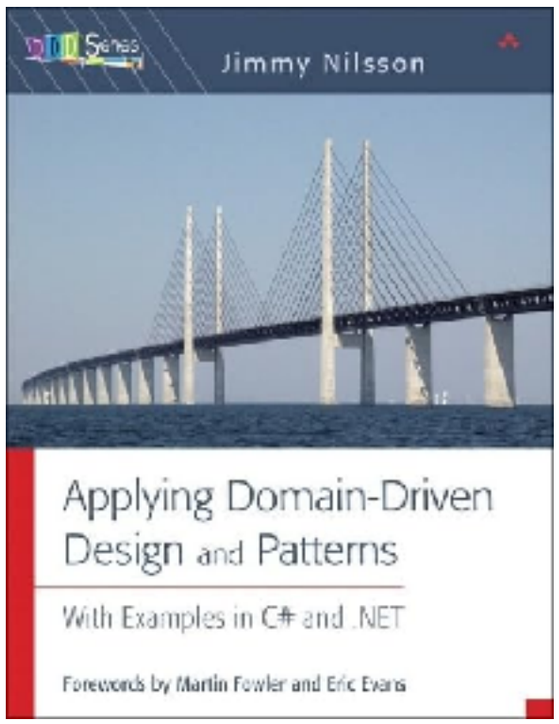
Domain Driven Design (DDD)

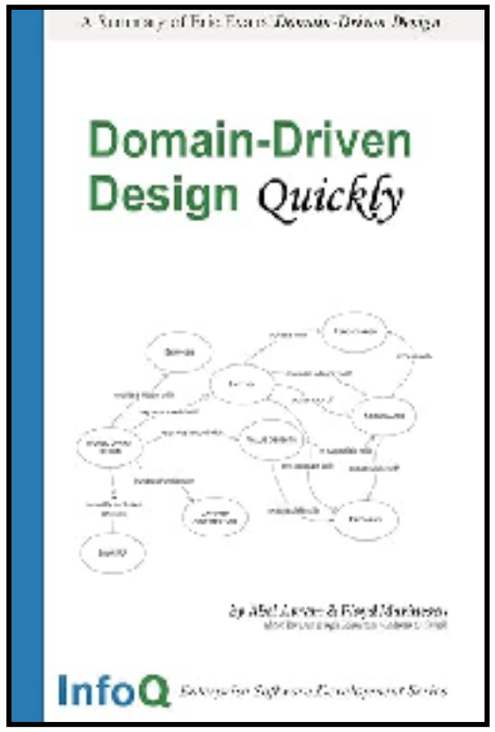


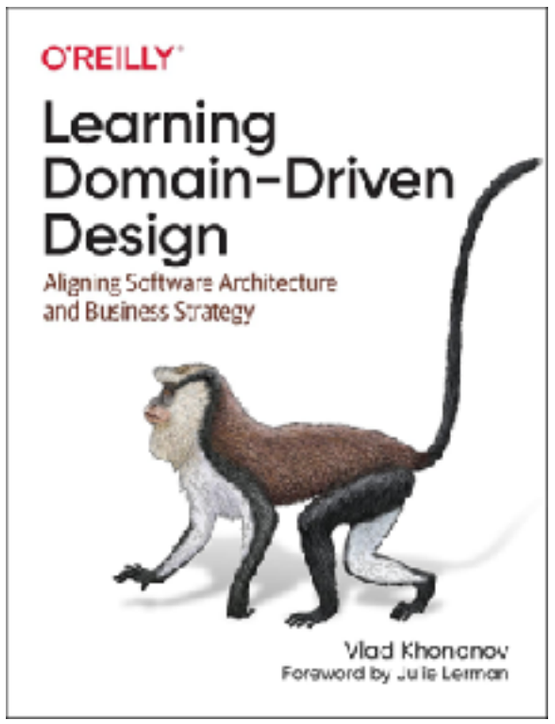










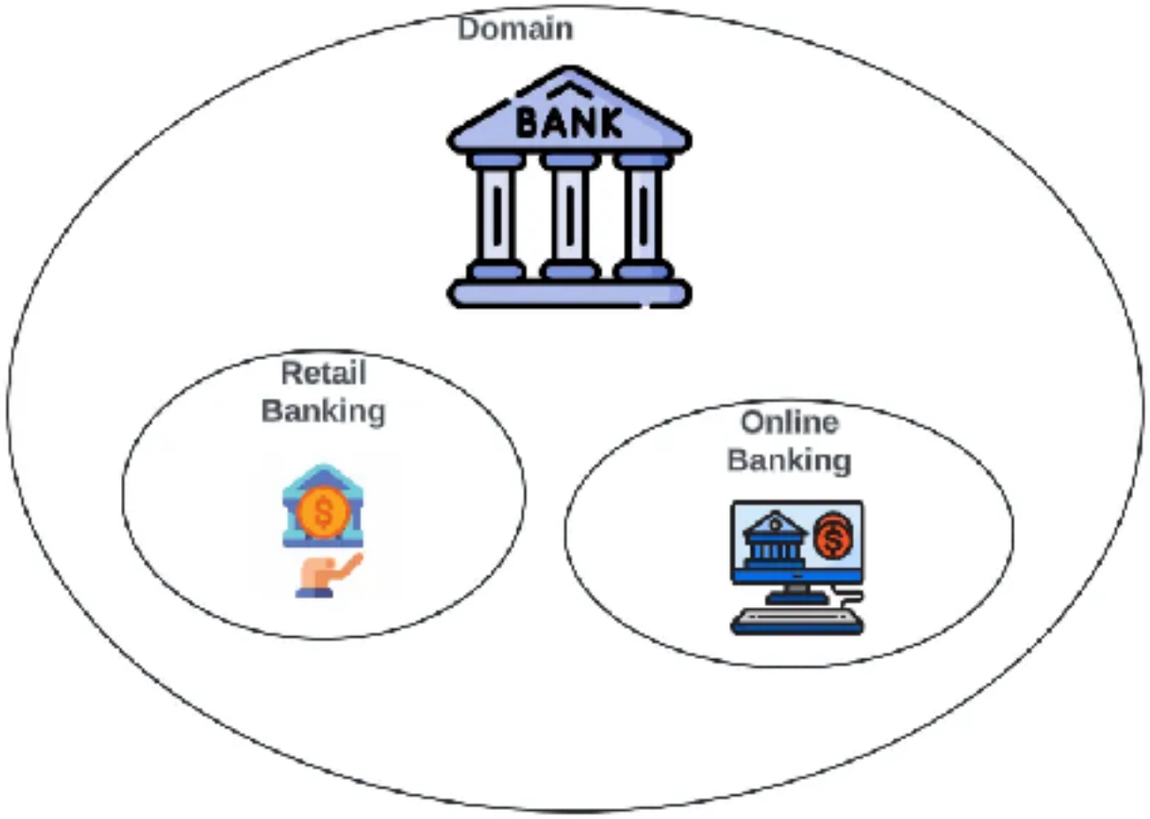


Domain ?

Domain ?

The area of knowledge or activity around which the application is centered

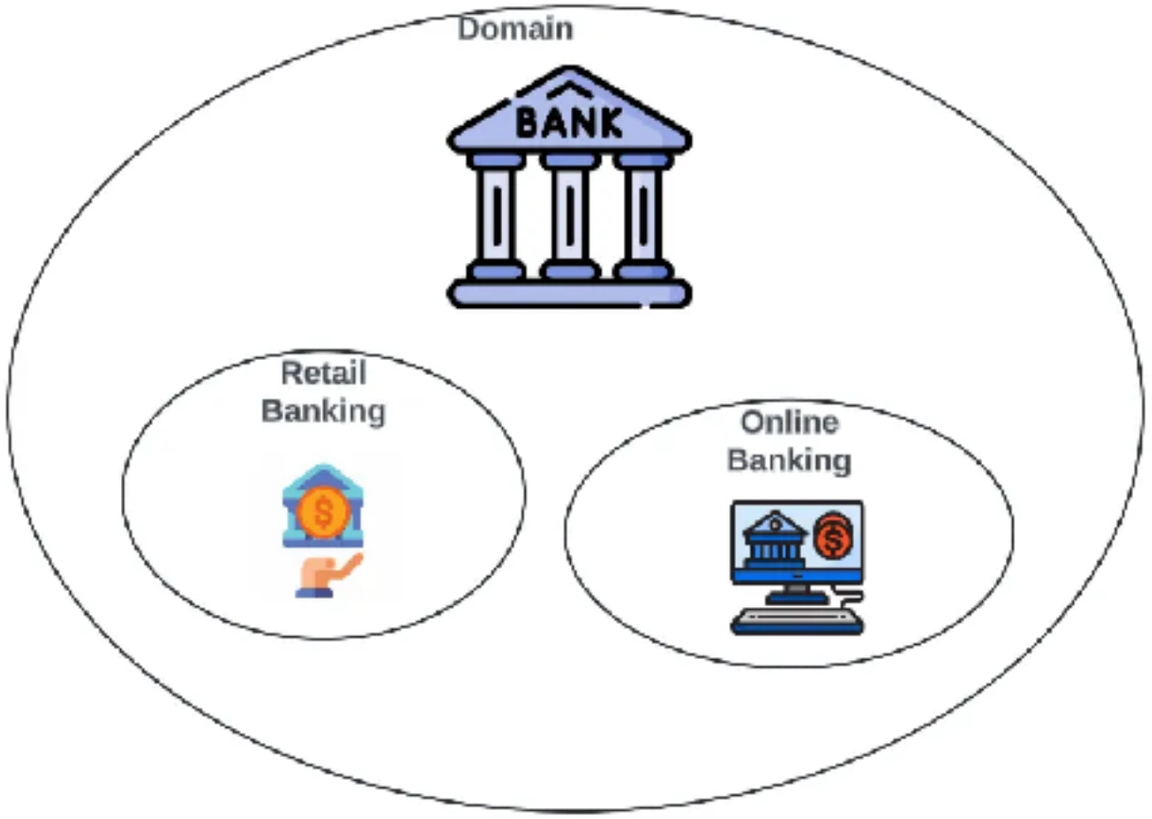
Bank Domain



Subdomain ?

Specific areas within the domain
Core, Supporting and Generic (important and role)

Bank Domain



Core Subdomain

Heart of business (competitive advantage)

Focus on delivering **business value** that matter the most for the **customer**

Supporting Subdomain

Crucial for the car subdomain to operate effectively

But don’t provide the level of differentiation and not unique to the company

Generic Subdomain

Common functionality
Not specific to the business domain
Often be addressed with readily available solutions

Subdomain

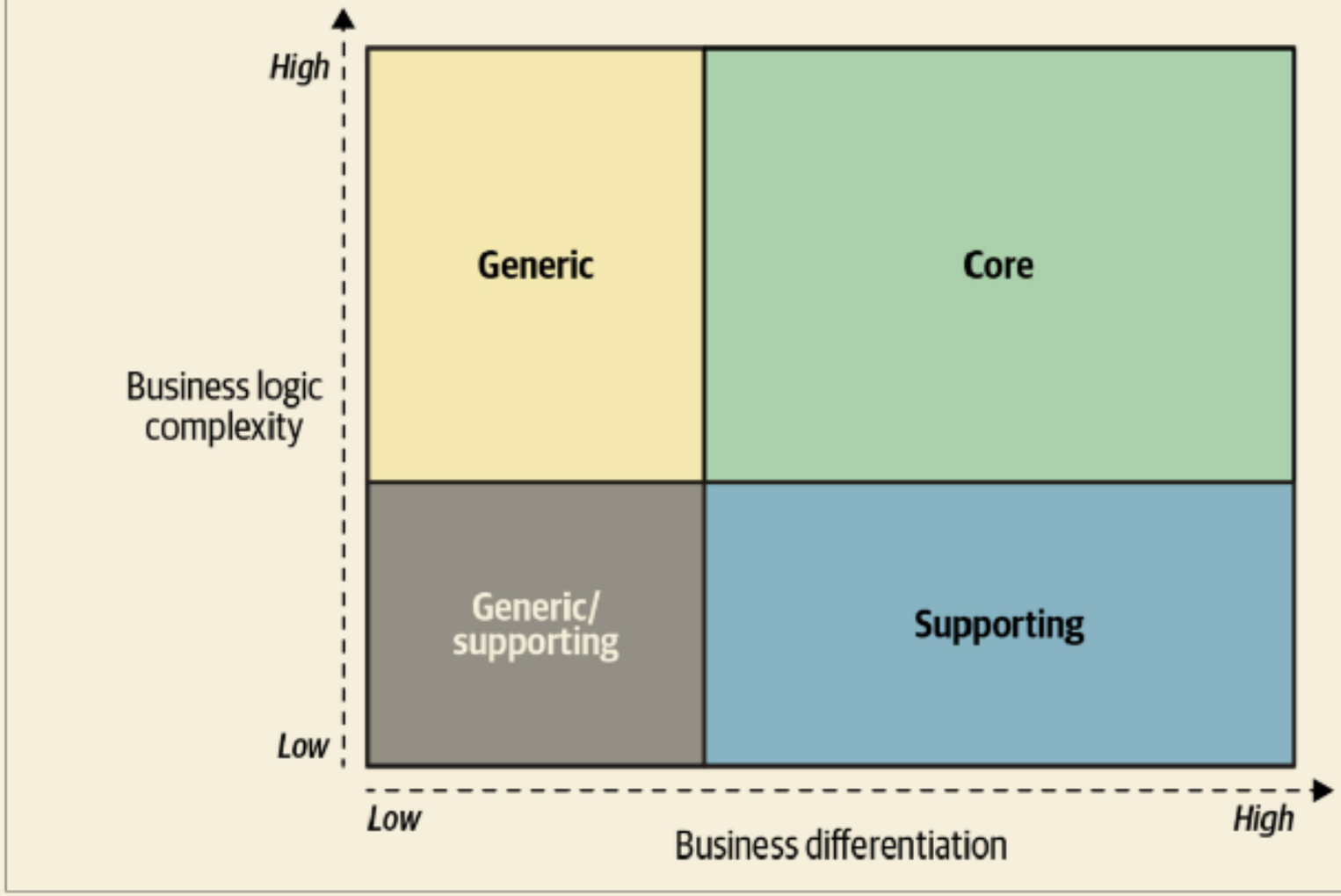
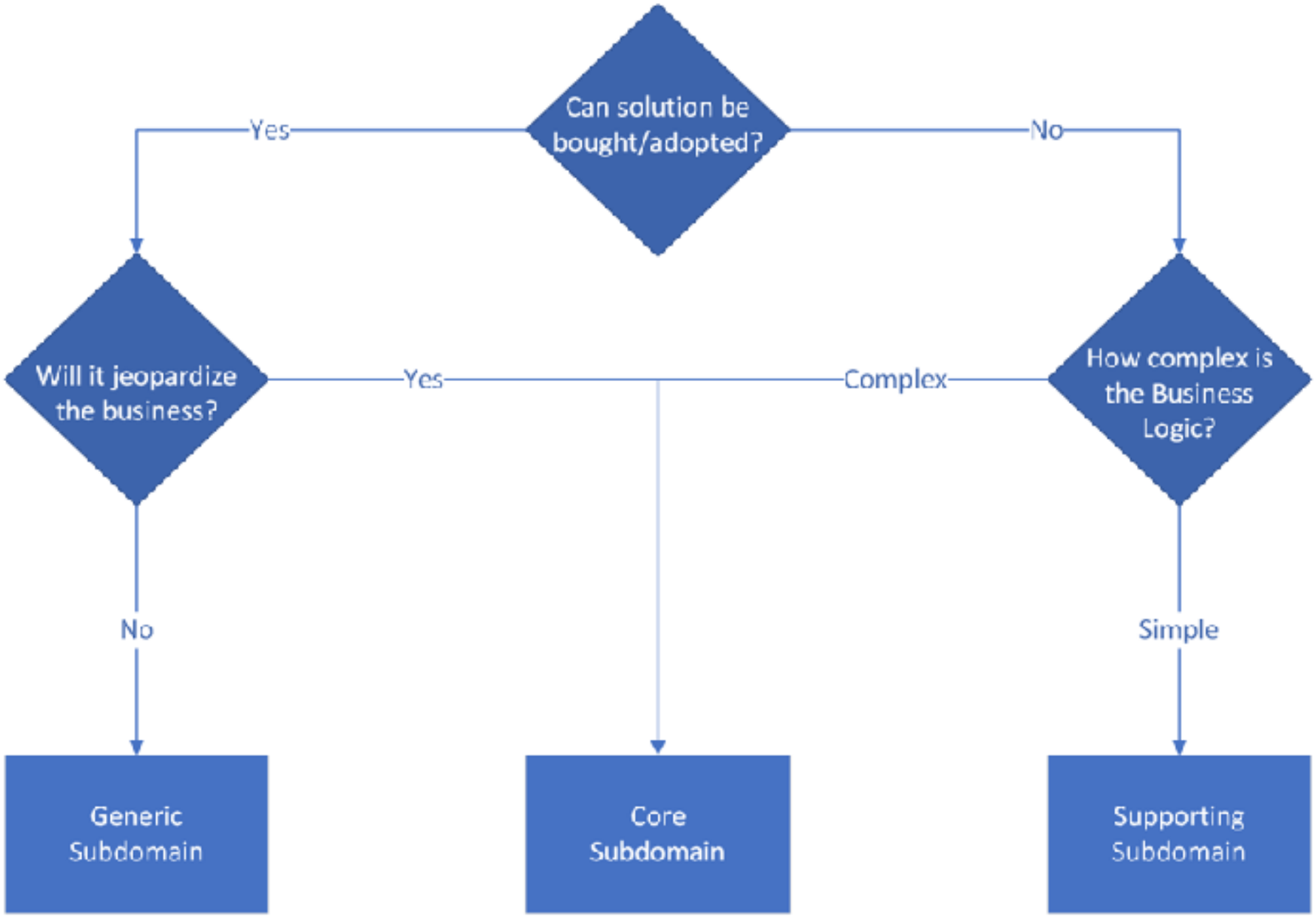


Figure 1-1. The business differentiation and business logic complexity of the three types of subdomains

https://datahonor.com/se/ddd/learning-ddd/#chap1-analyzing-business-domains

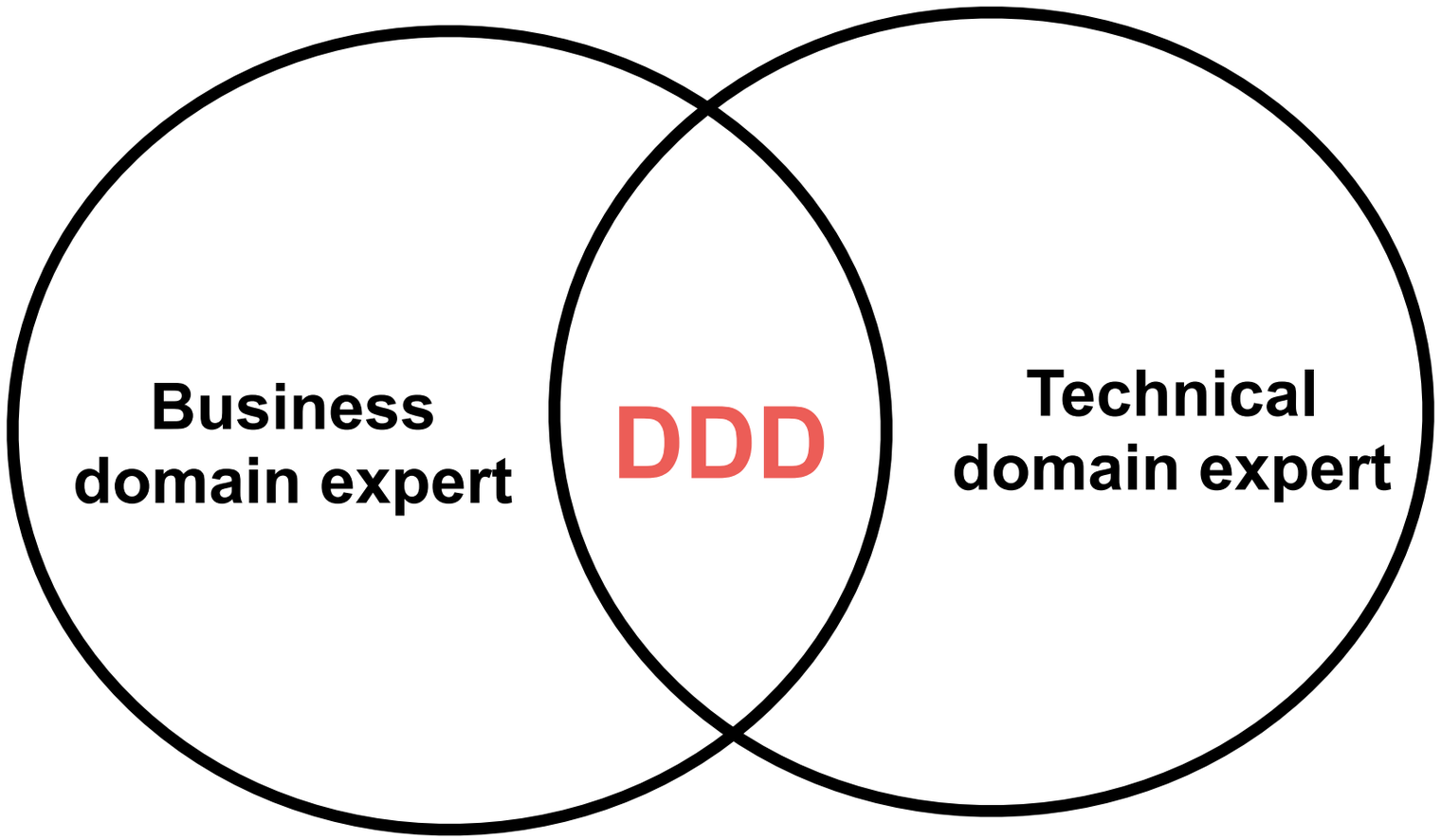
Categorize Subdomain ?



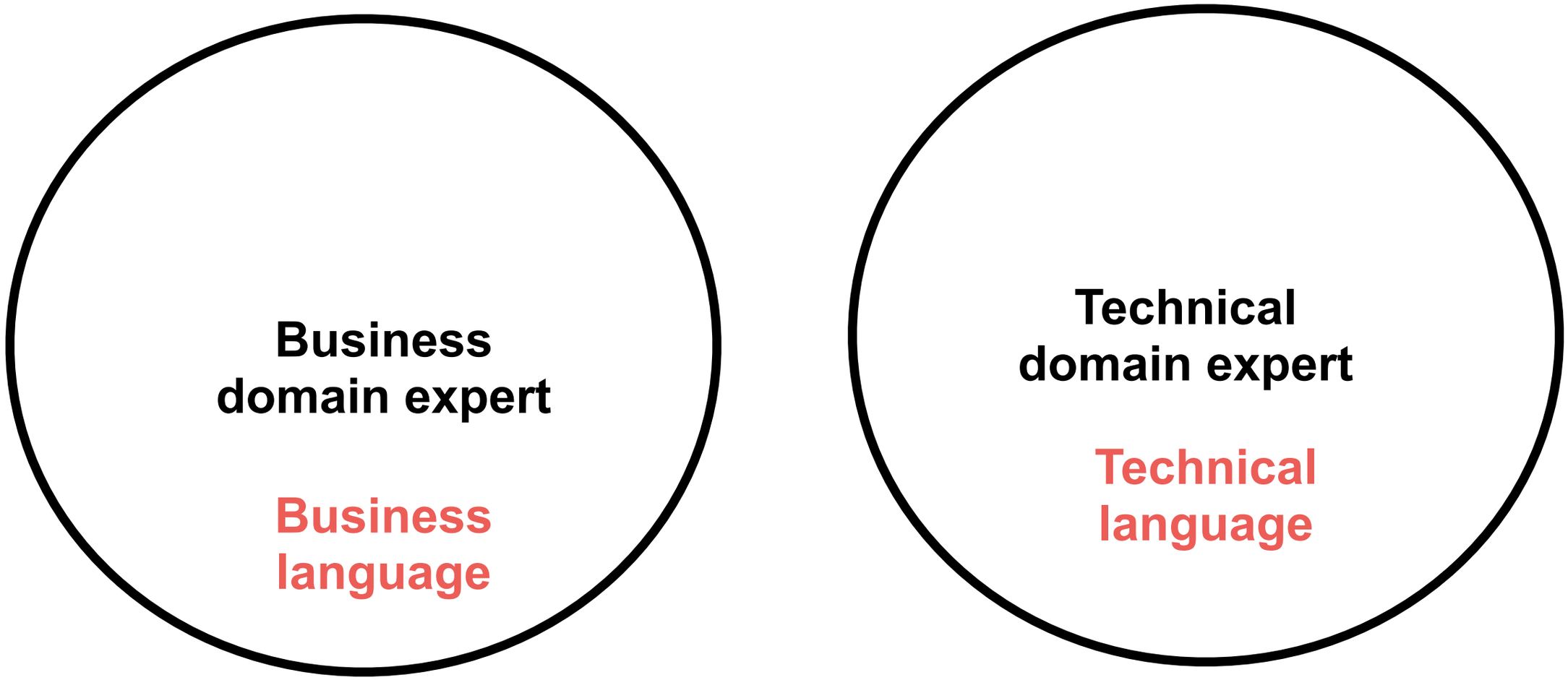
https://vladikk.com/2018/01/26/revisiting-the-basics-of-ddd/

Domain Driven Design
is not
About technologies

DDD



Problem ?



Problem with language ?

Business domain expert

Business language

Technical domain expert

Technical language

Parcel sent

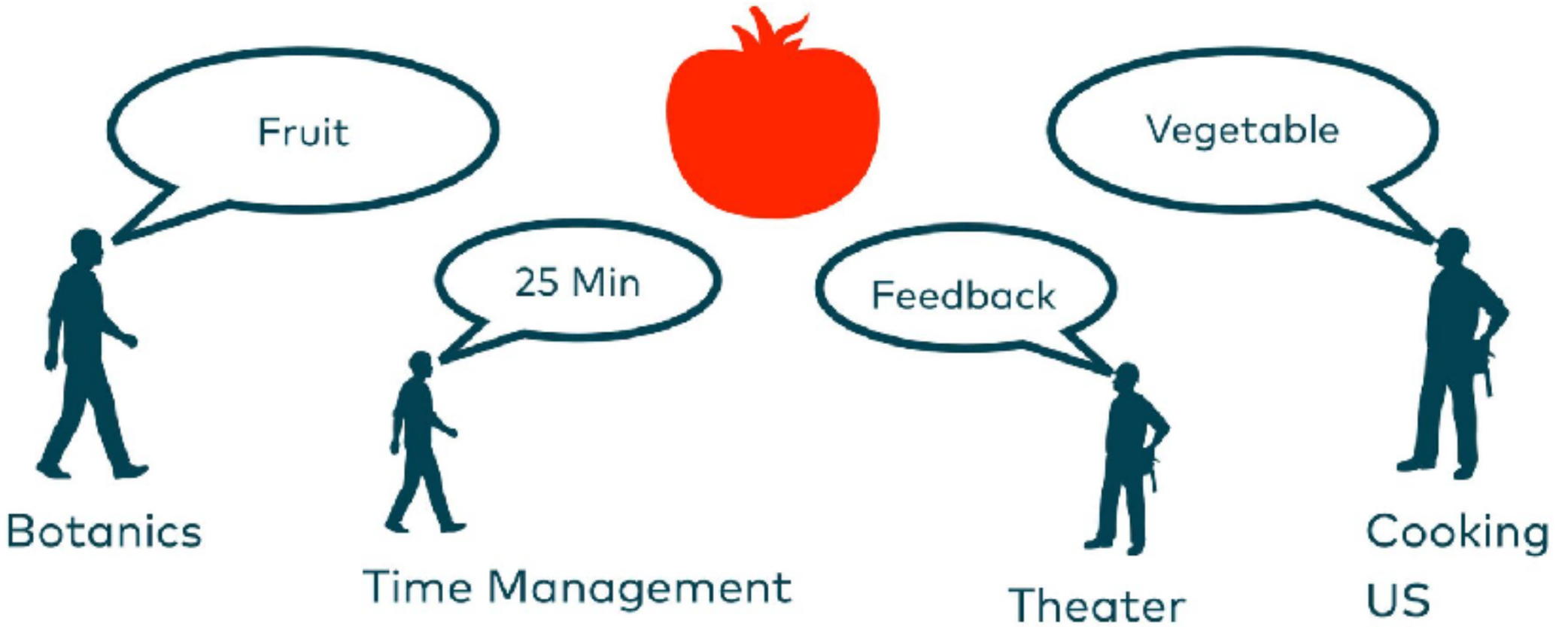
Predict delivery

Recipient notified

Results MySQL

Messaging RabbitMQ

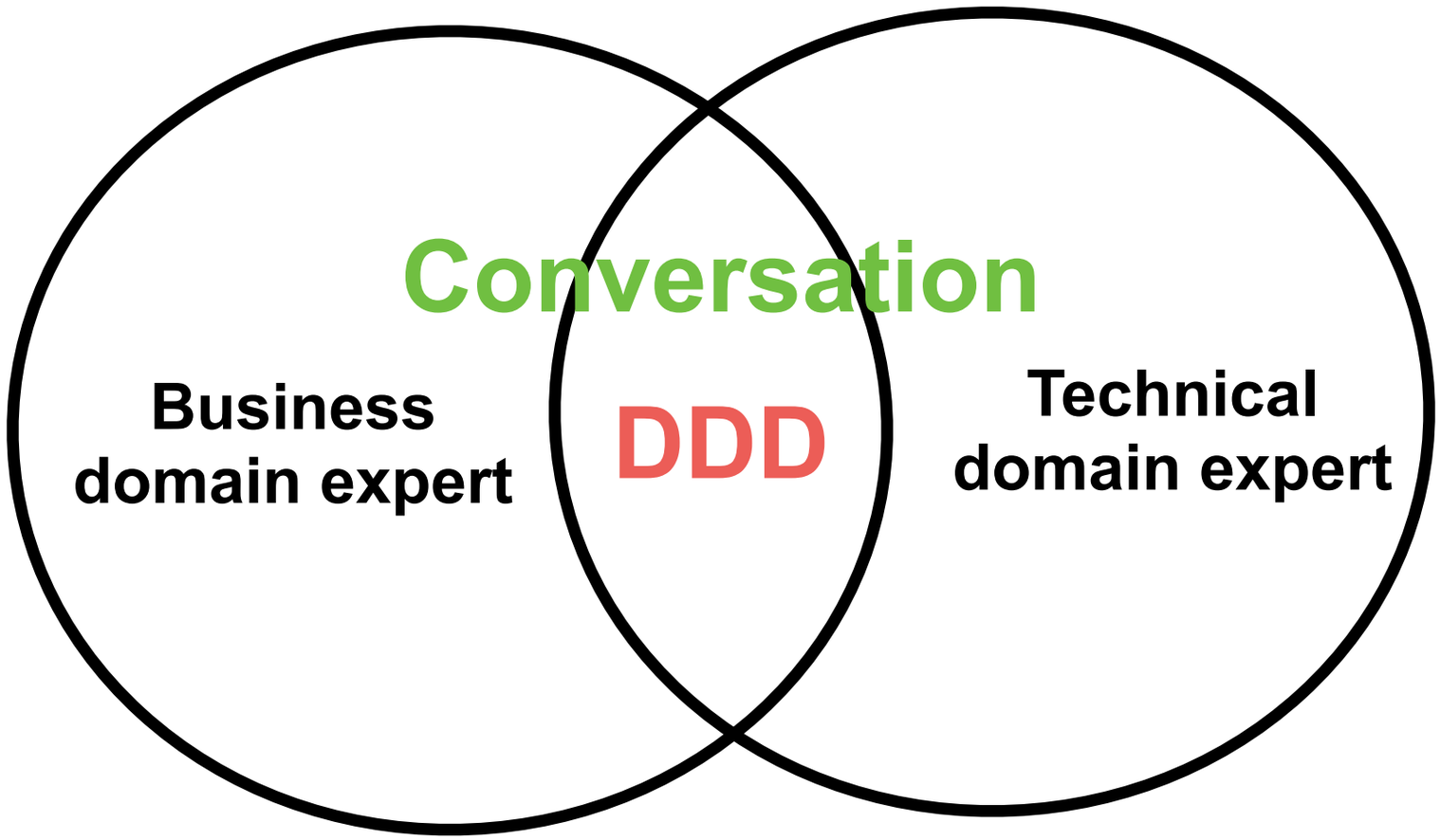
AI Engine

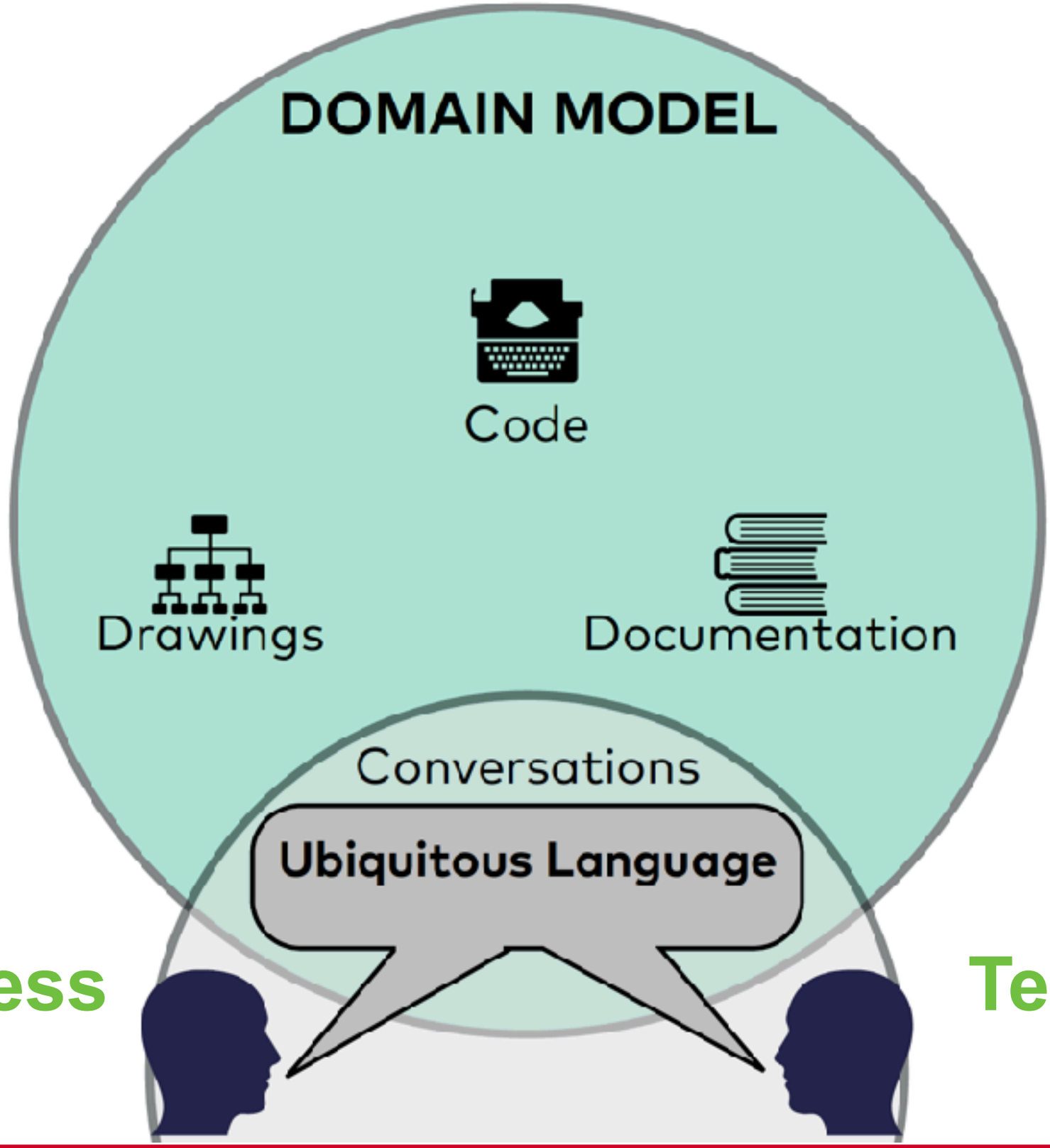


What is a tomato?

Need Same Language ?

Ubiquitous Language





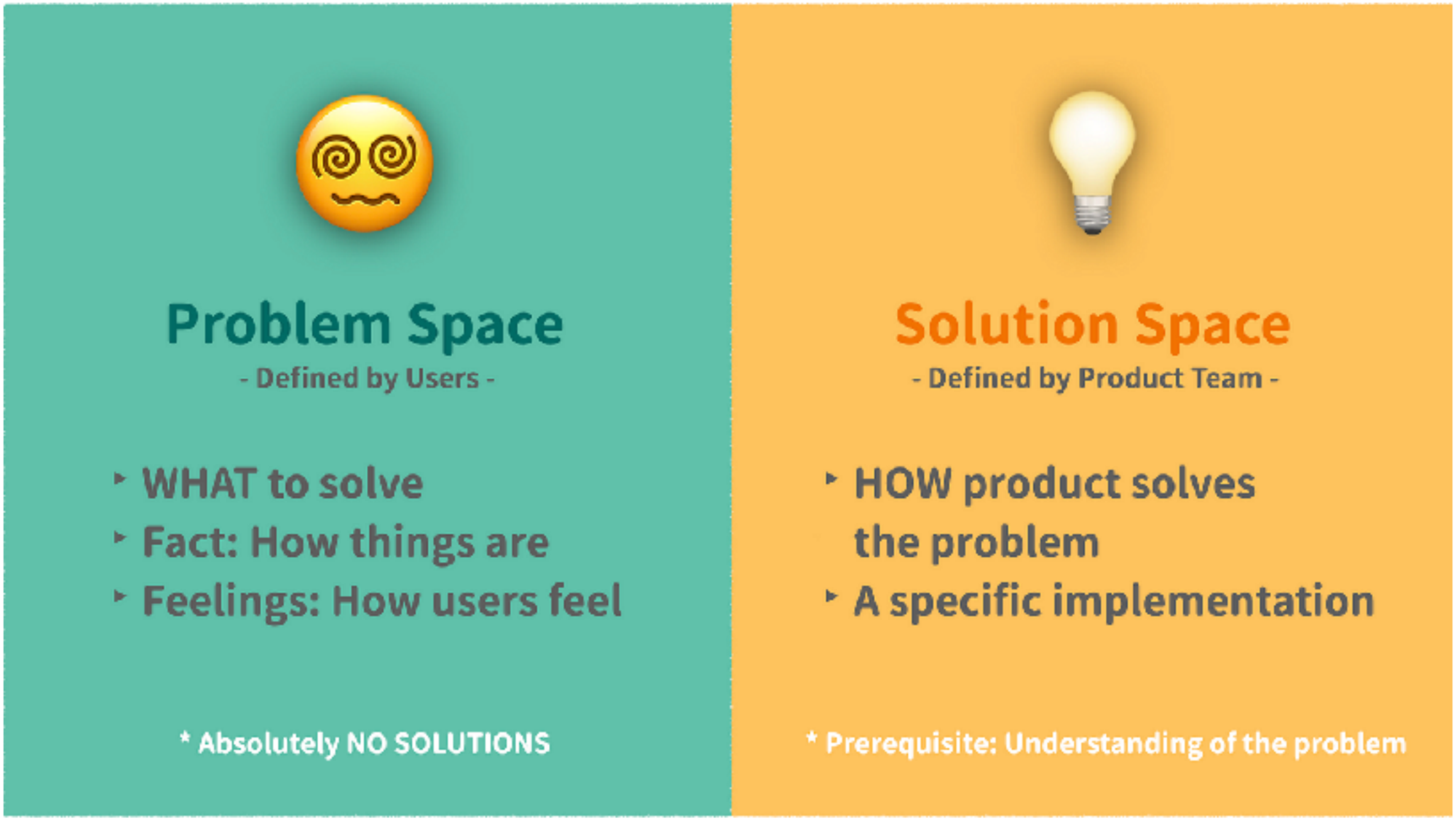
Business **Technical**

Need Same Direction ?

Need for DDD

Top-down approach
Aligning with business
Communication (ubiquitous language)
Remaining flexible and scalable for biz changing
Mapping dependencies (clear boundary)

Start with Problem Space



Start with Problem ?

Ubiquitous language

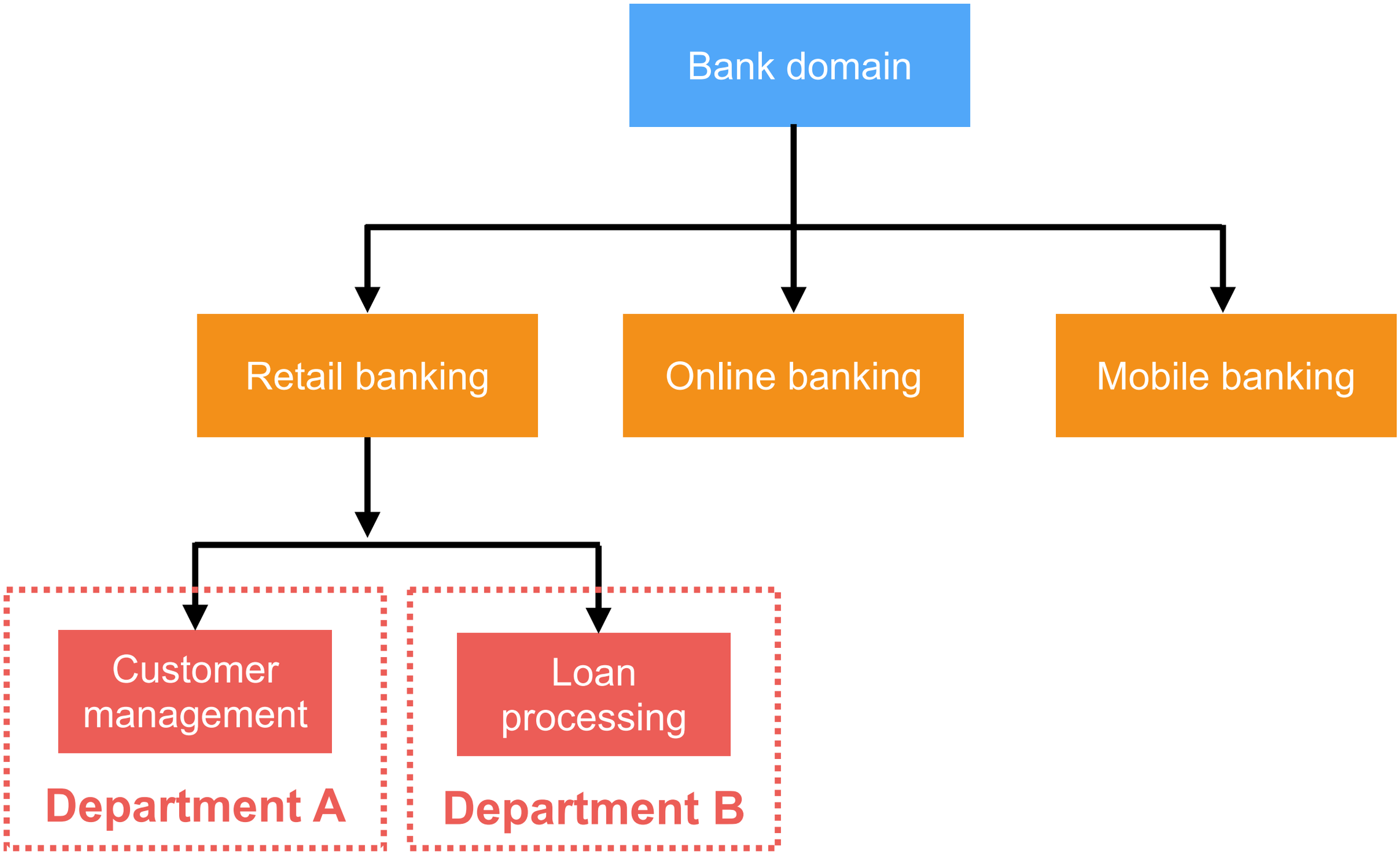
Effective communication

Knowledge Crunching

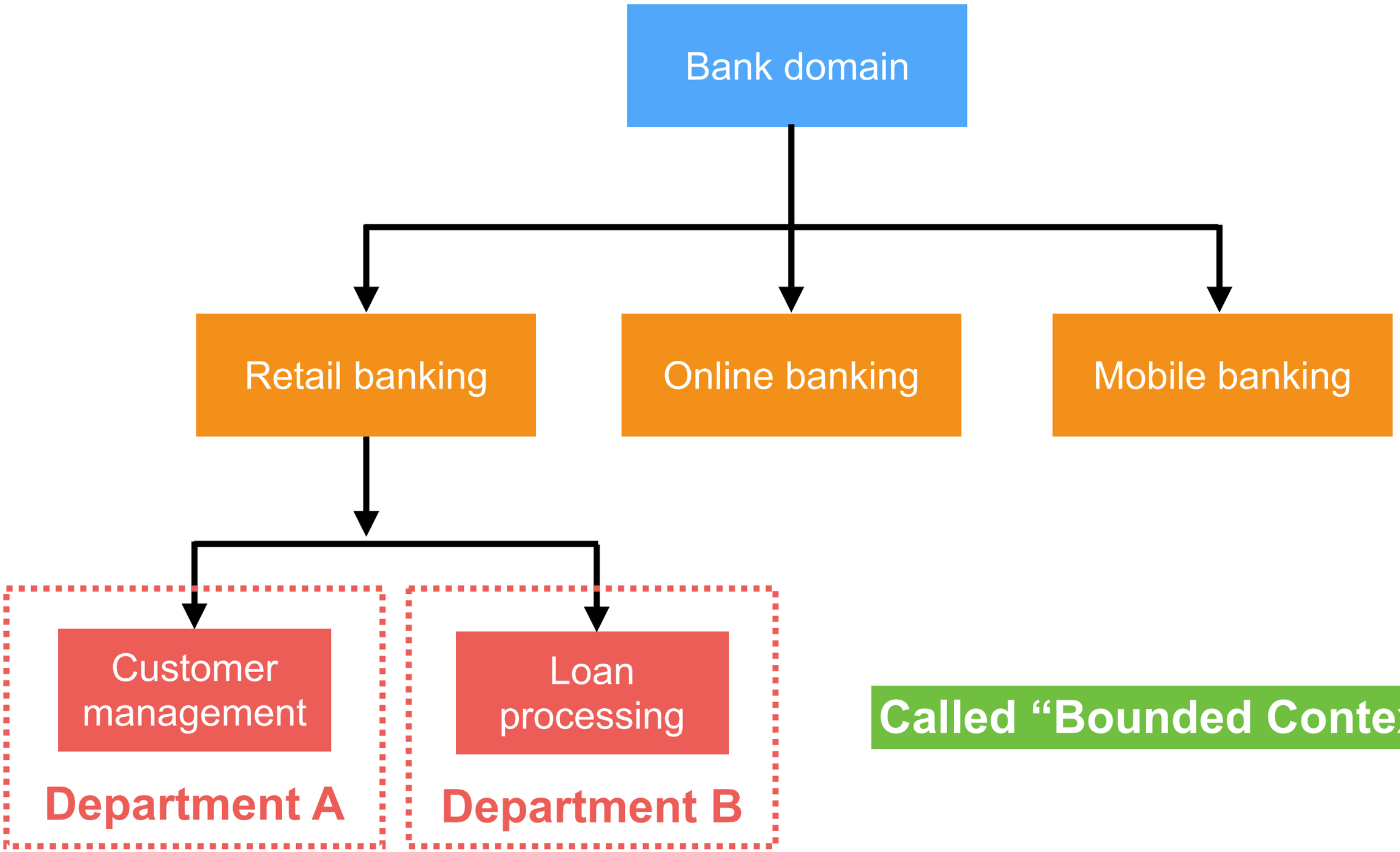
Business rules

Features

Example of Bank



Example of Bank



Called “Bounded Context”

Retail Banking

Bounded Context !!

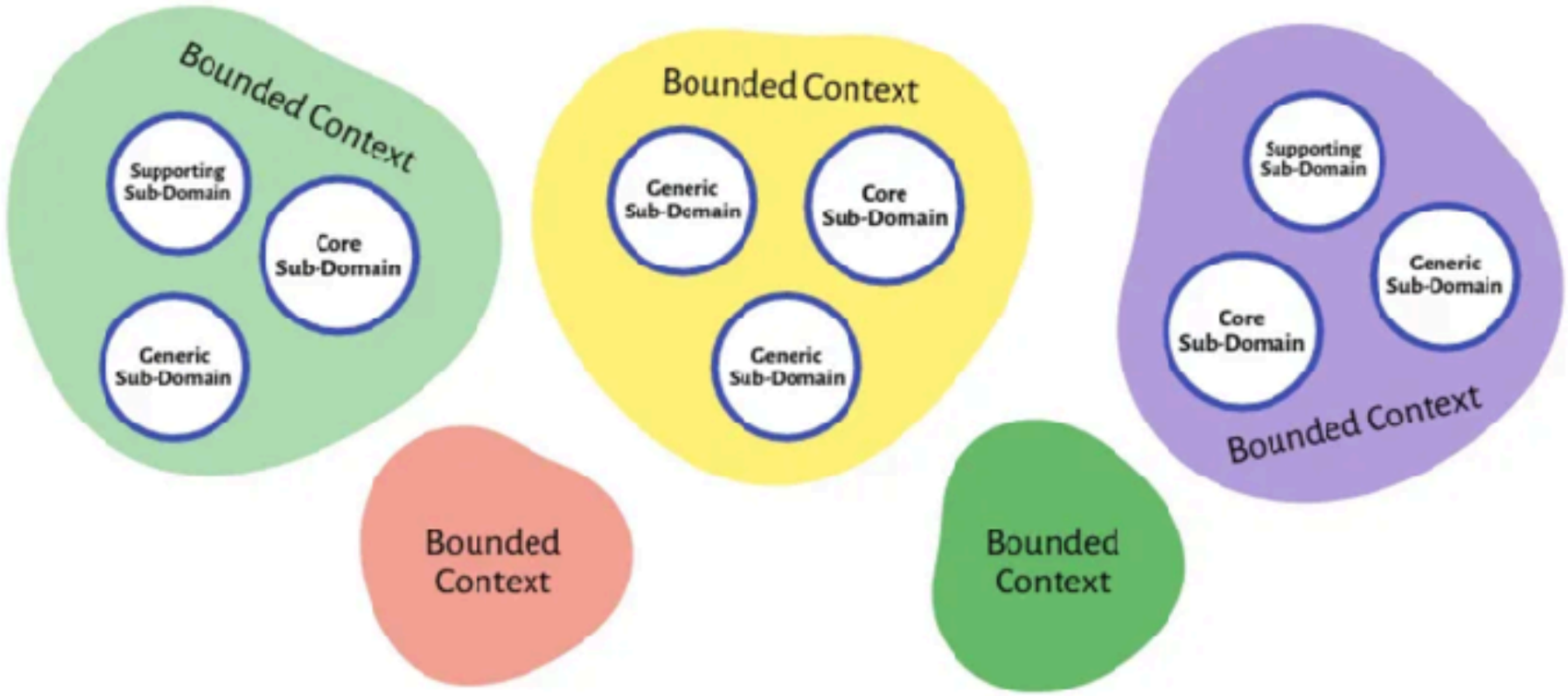
Customer management

Delivery

Loan processing

Finance

Bounded Context



Retail Banking

Language in each Bounded Context !!

Customer management

Delivery

Loan processing

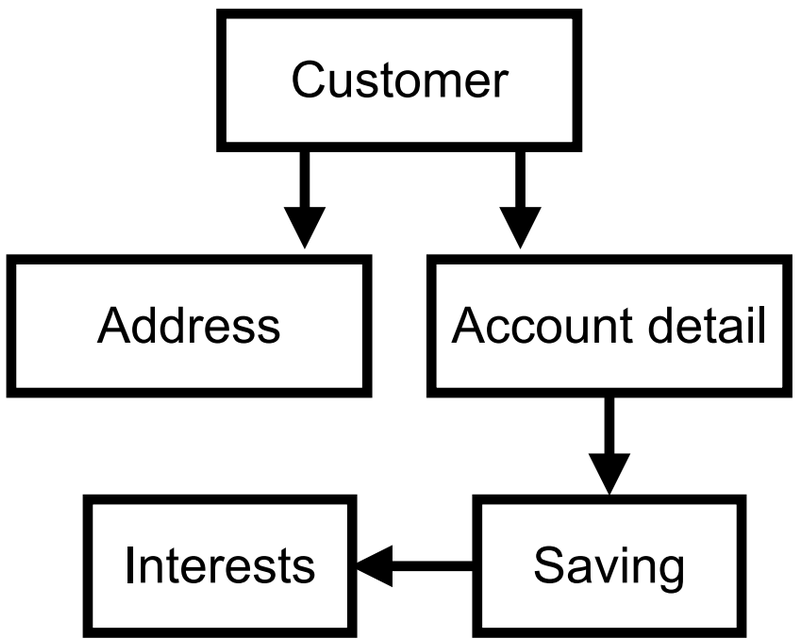
Finance

Each context has own language

Need “Domain Expert”

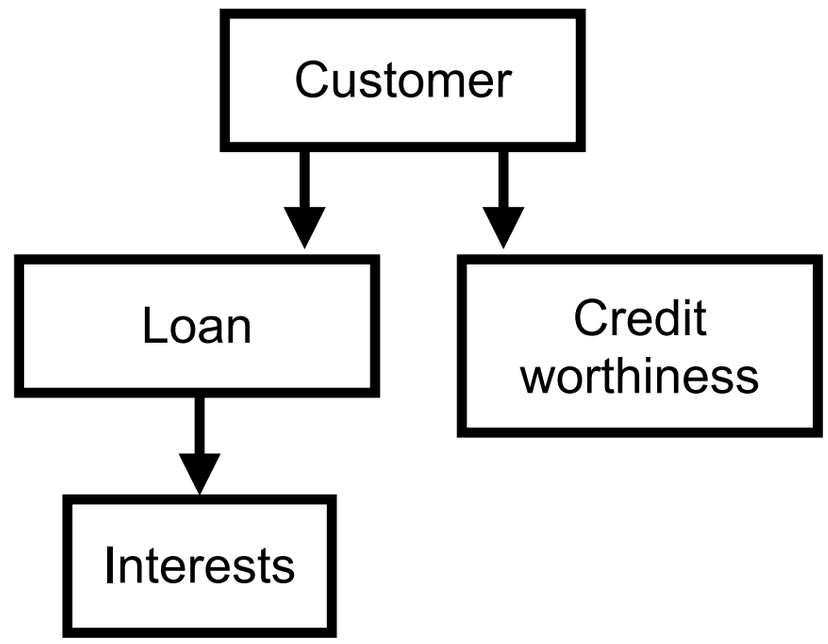
Customer management

Delivery



Loan processing

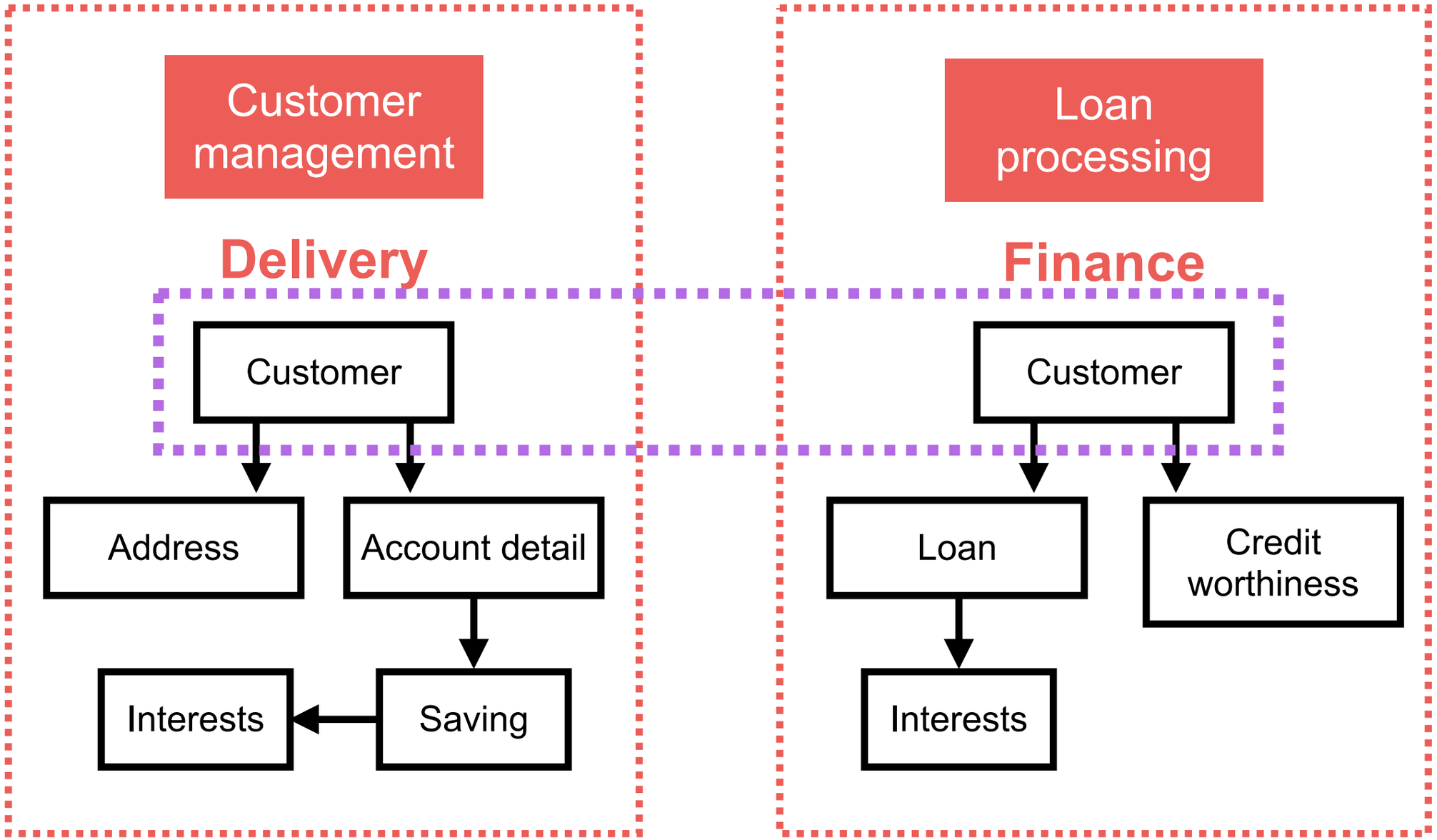
Finance



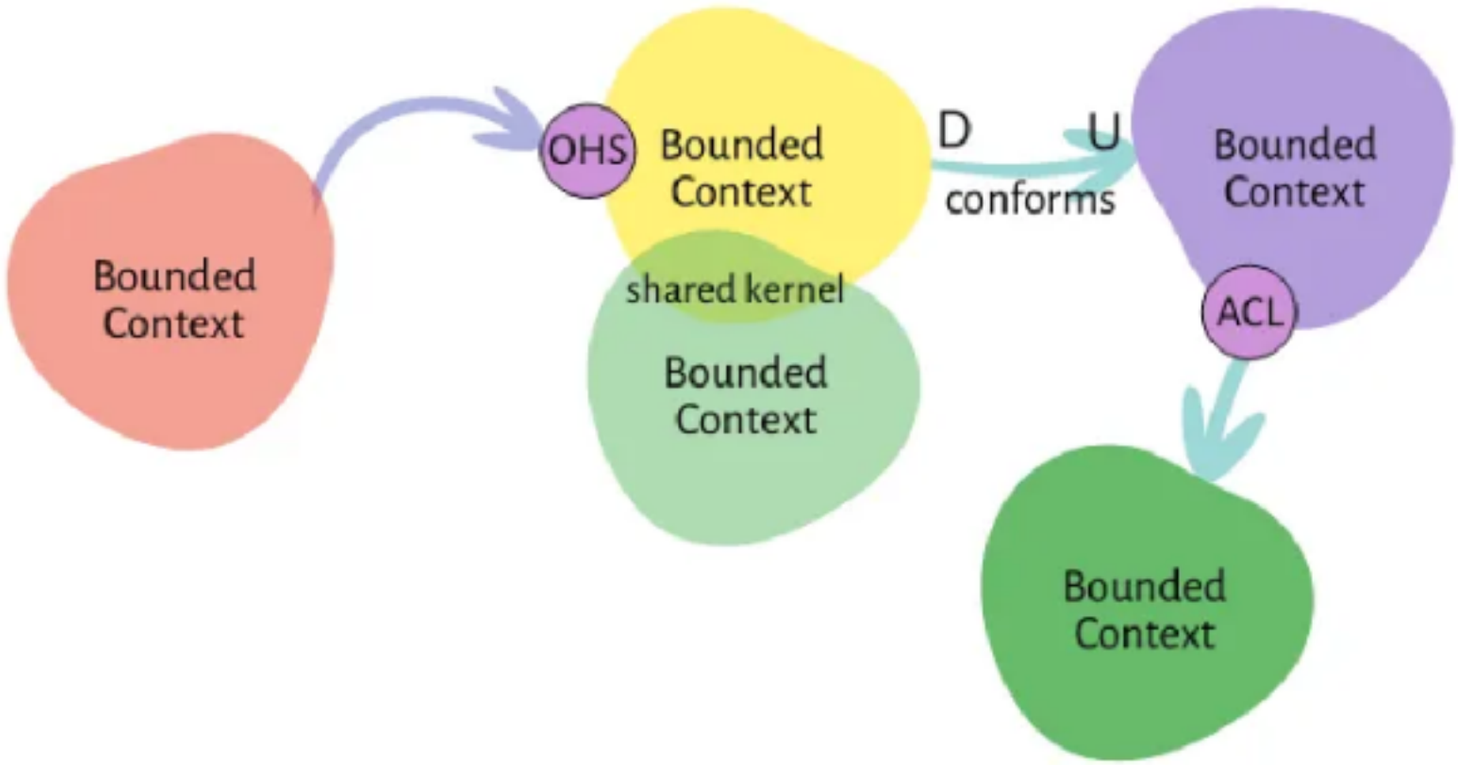
Relationship
between bounded contexts ?

Relationship ?

Called “Context Mapping”



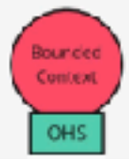
Context Mapping



Context Mapping

Open / Host Service

A Bounded Context offers a defined set of services that expose functionality for other systems. Any downstream system can then implement their own integration. This is especially useful for integration requirements with many other systems. Example: public APIs.



Conformist

The downstream team conforms to the model of the upstream team. There is no translation of models. Couples the Conformist's domain model to another bounded context's model.

Anticorruption Layer

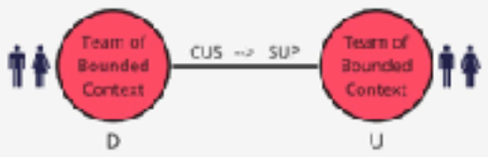
The anticorruption layer is a layer that isolates a client's model from another systems model by translation. Only couples the integration layer (or adapter) to another bounded context's model but not the domain model itself.

Shared Kernel

Two teams share a subset of the domain model including code and maybe the database. Typical examples: shared JARs, DLLs or a shared database schema. Teams with a Shared Kernel are often mutually dependent and should form a Partnership.

Customer / Supplier

There is a customer / supplier relationship between teams. The downstream team is considered to be the customer. Downstream requirements factor into upstream planning. Therefore, the downstream team gains some influence over the priorities and tasks of the upstream team.



Partnership

Partnership is a cooperative relationship between two teams. These teams establish a process for coordinated planning of development and joint management of integration.

Published Language

A Published Language is a well documented shared language between Bounded Contexts which can translate in and out from that language. Published Language is often combined with Open Host Service. Typical examples are iCalendar or vCard.

Separate Ways

Bounded Contexts and their corresponding teams have no connections because integration is sometimes too expensive or it takes very long to implement. The teams chose to go separate ways in order to focus on their specific solutions.

Big Ball Of Mud

A (part of a) system which is a mess by having mixed models and inconsistent boundaries. Don't let this lousy model propagate into the other Bounded Contexts. Big Ball Of Mud is a demarcation of a bad model or system quality.

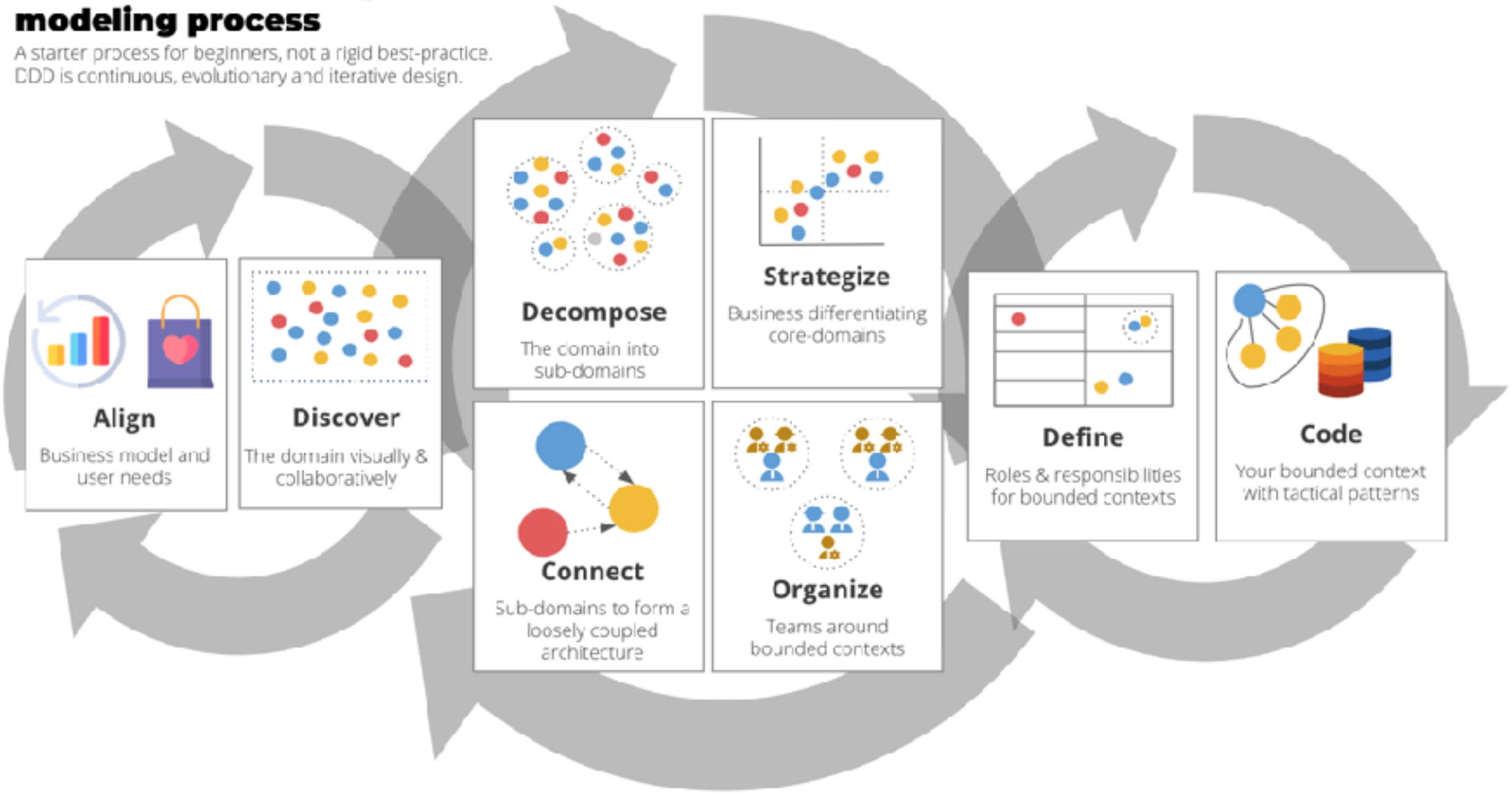
https://github.com/ddd-crew/context-mapping/

DDD Modeling process

Continuous, evolutionary and iterative design

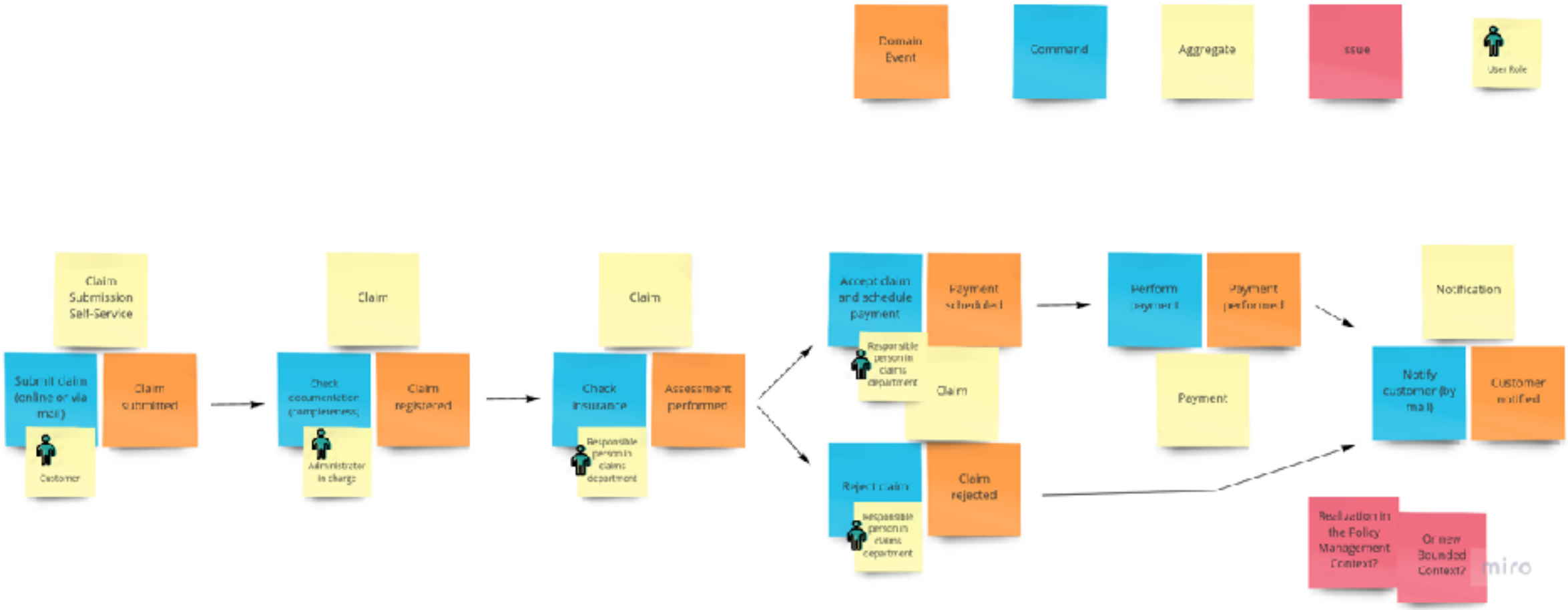
Domain-Driven Design starter modeling process

A starter process for beginners, not a rigid best-practice.
DDD is continuous, evolutionary and iterative design.



https://github.com/ddd-crew/ddd-starter-modelling-process

Event Storming



https://contextmapper.org/docs/event-storming/

Workshop