Domain-Driven Design

Back to the roots
Retour aux sources

Cédric Pontet

CTO @ Agile Partner, Founder @ #play14, Agile Coach, Software & Cloud Architect, EventStormer, and Sketchnoter

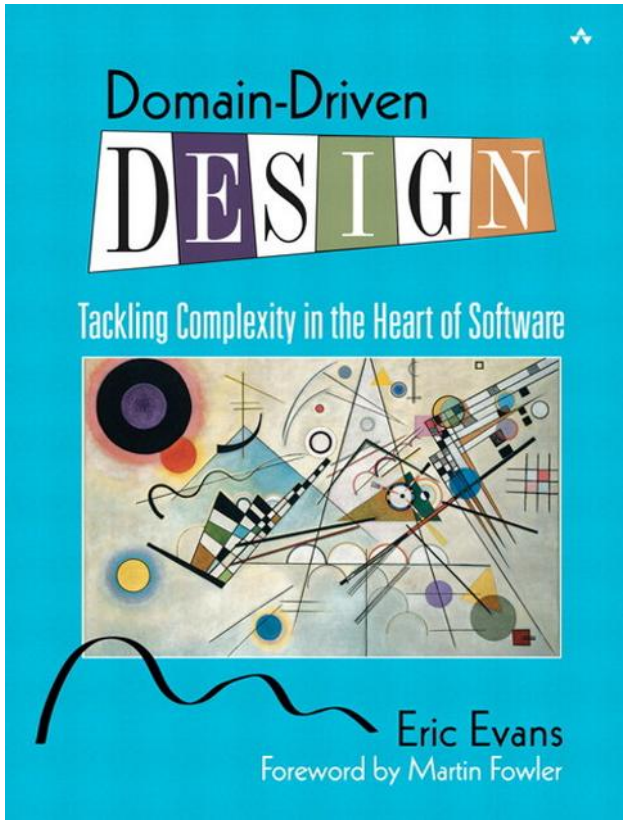
Luxembourg · 500+ connections · Contact info

Agile Partner S.A.

Université Henri Poincaré (Nancy I)

I could start like this

Eric Evans is a genius



1st edition - August 20, 2003

But I won’t to that

I have seen that talk before

I have given that talk before

You might have seen that talk before

DDD

It’s complicated

It takes time to undestand

It’s not for beginners

My goal

It’s not “so” complicated

You already know the concepts

You are probably already doing it

Let’s do something different

Let’s experiment

Let’s start from an example

Congratulations

You have all been hired by the best new bank in Luxembourg

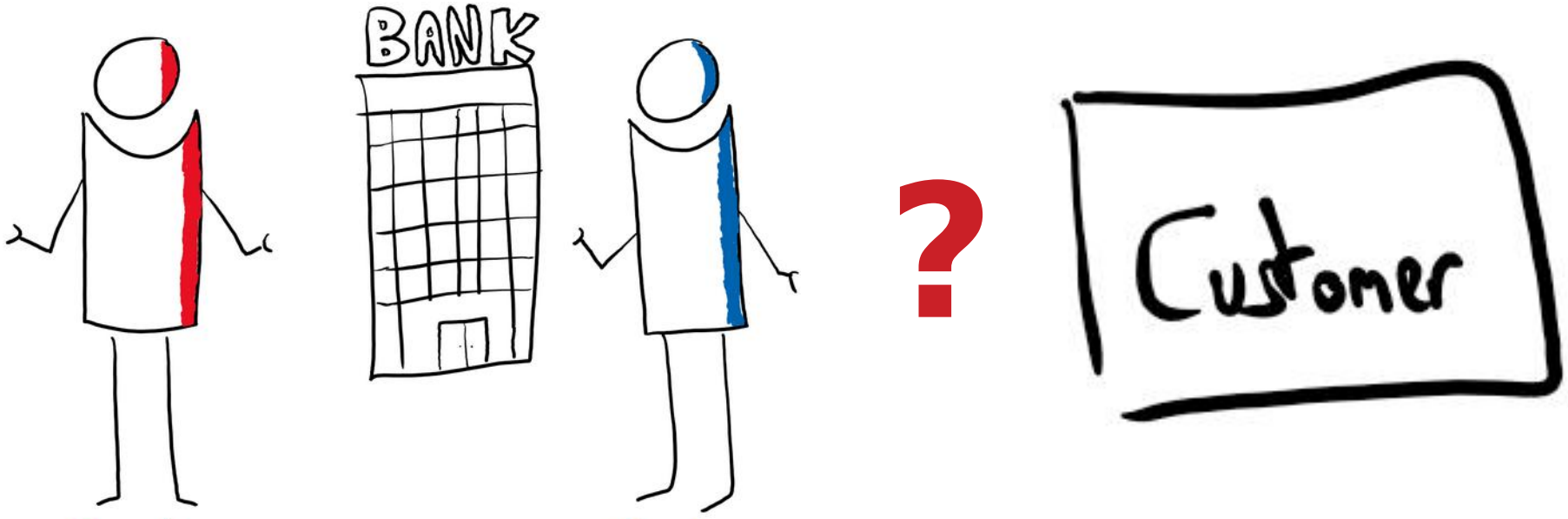
Welcome to BXL

Banque eXtraordinaire de Luxembourg

“Make your daily life exceptional”
“Faire de l’excepionnel votre quotidien”

CC BY-SA

It’s your first day



Your boss

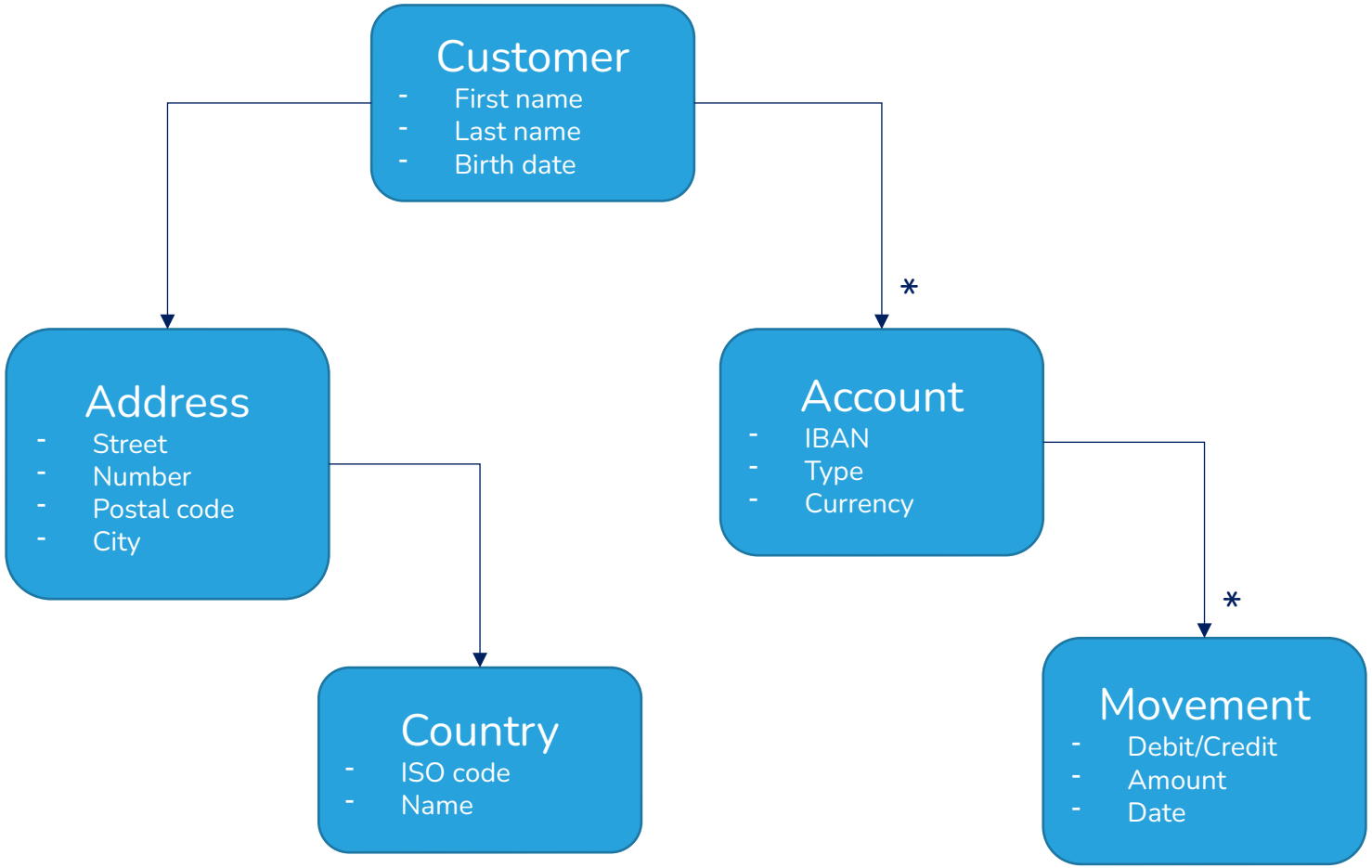
You

What is a customer?

Talk to your neighbor

Try to come up with a definition for a customer

Customer


Customer
- First name
- Last name
- Birth date
*
Address
- Street
- Number
- Postal code
- City
Account
- IBAN
- Type
- Currency
*
Country
- ISO code
- Name
Movement
- Debit/Credit
- Amount
- Date

It all makes sens

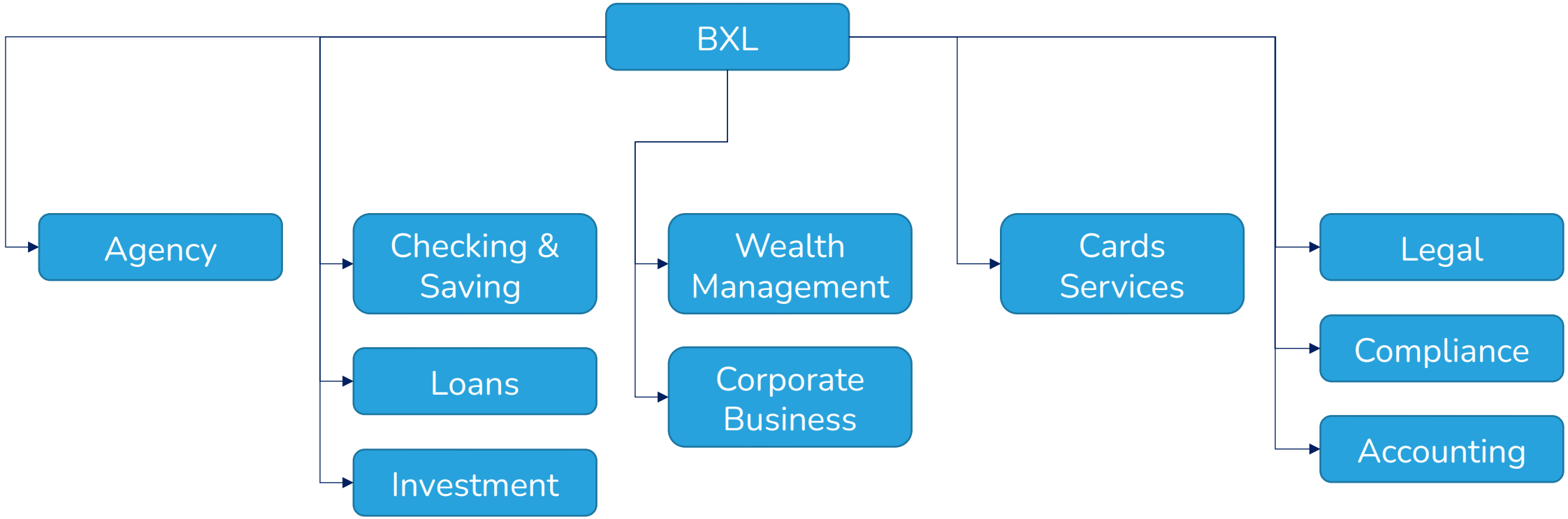
We can go on like that

We will end up with a pretty decent idea of what a customer is

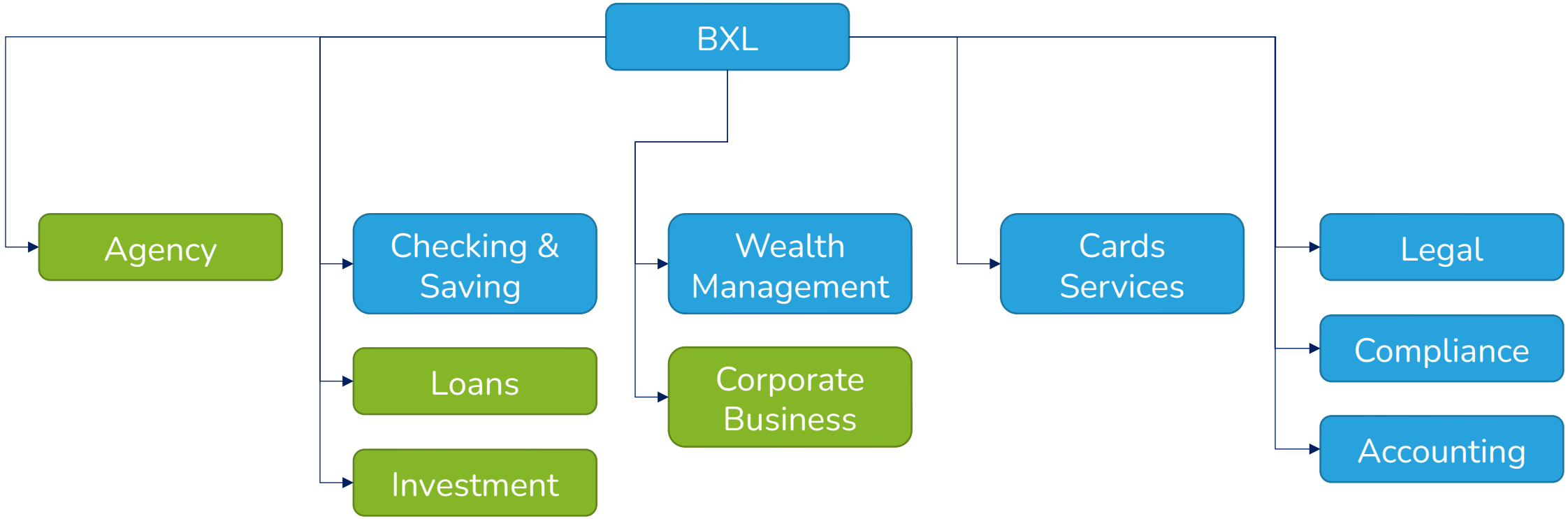
Does it though?

Does it really make sens?
Are we looking for the right thing?
Are we talking to the right people?

The bank has departments



The bank has departments



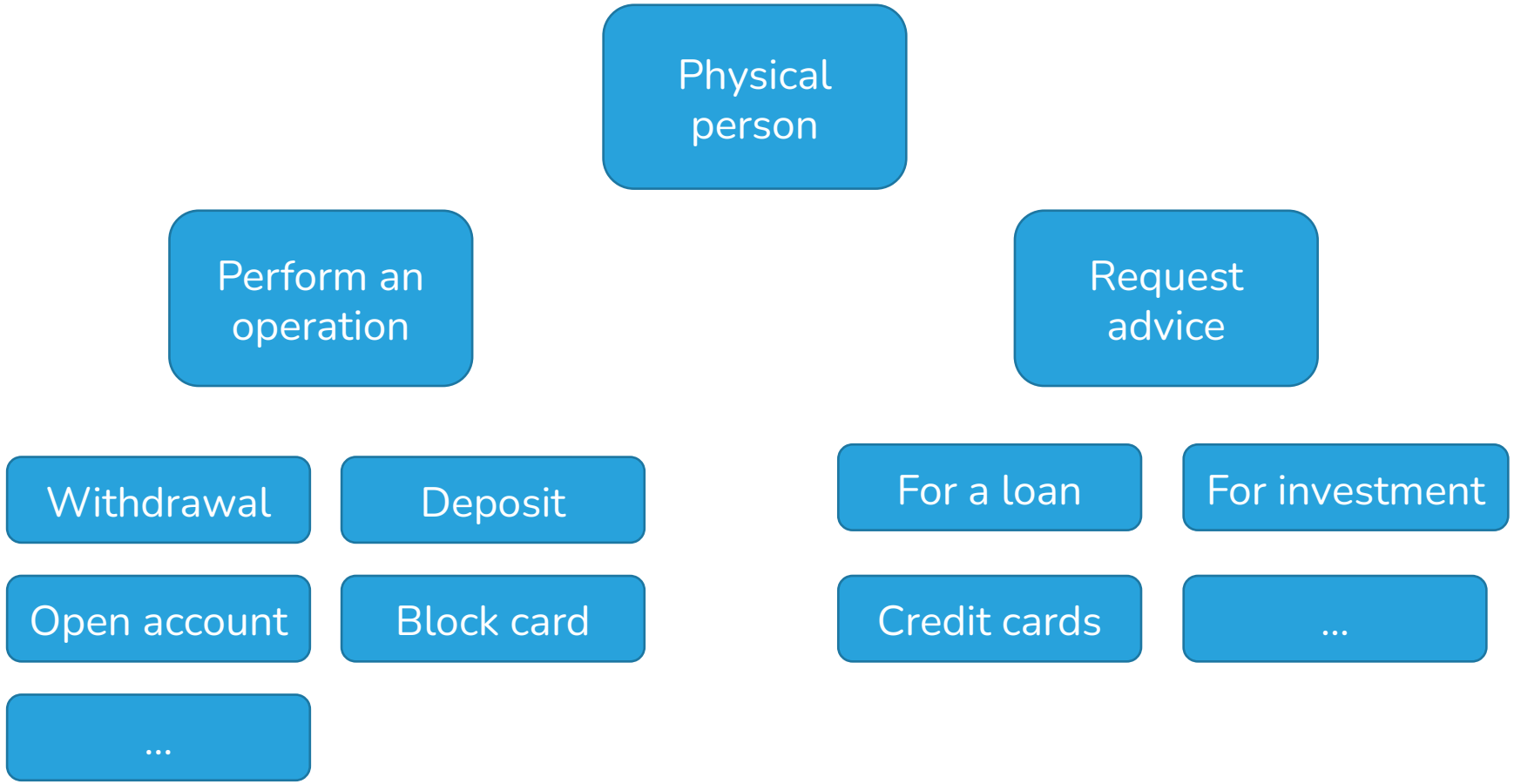
What is a customer?

Talk to your neighbor again

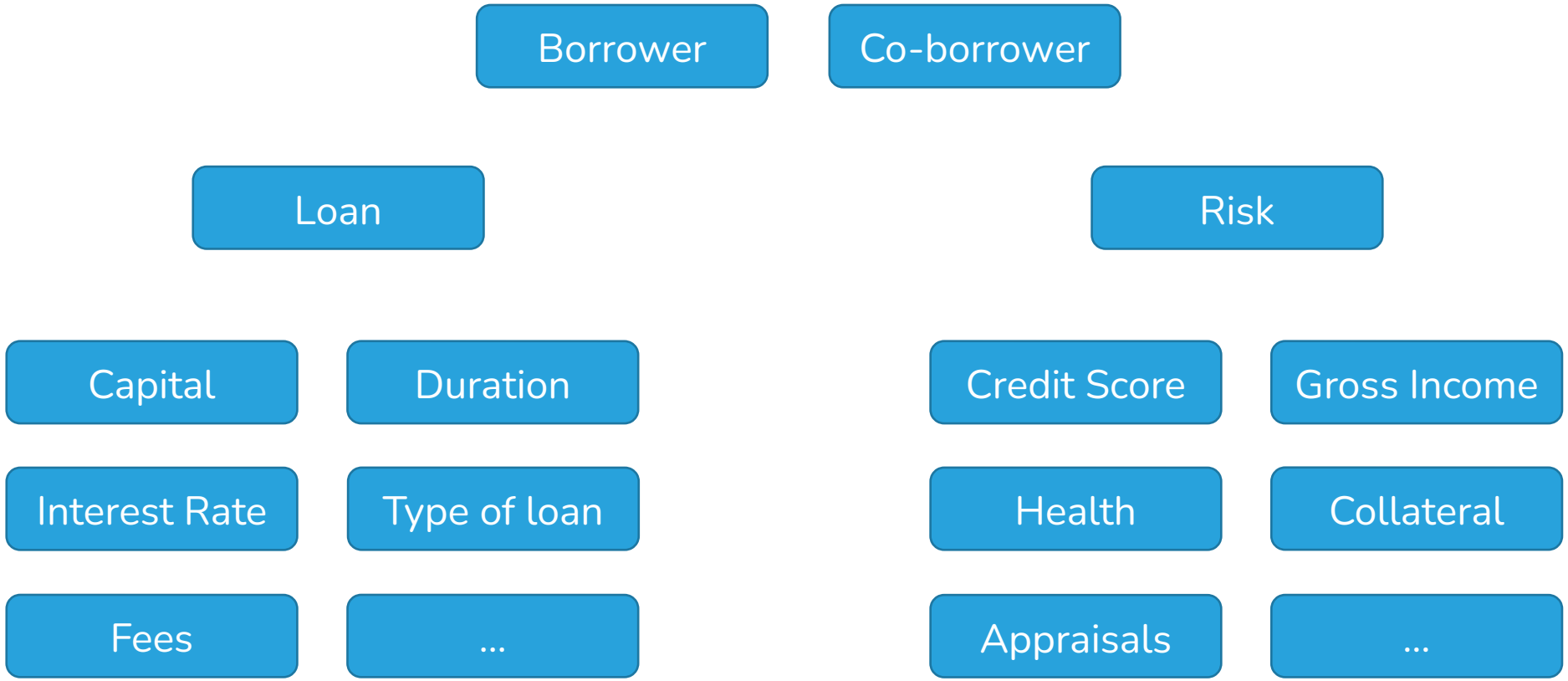
Imagine you ask that question to

- A teller in an agency
- A loan department employee
- An investment advisor
- A corporate banker for SMEs

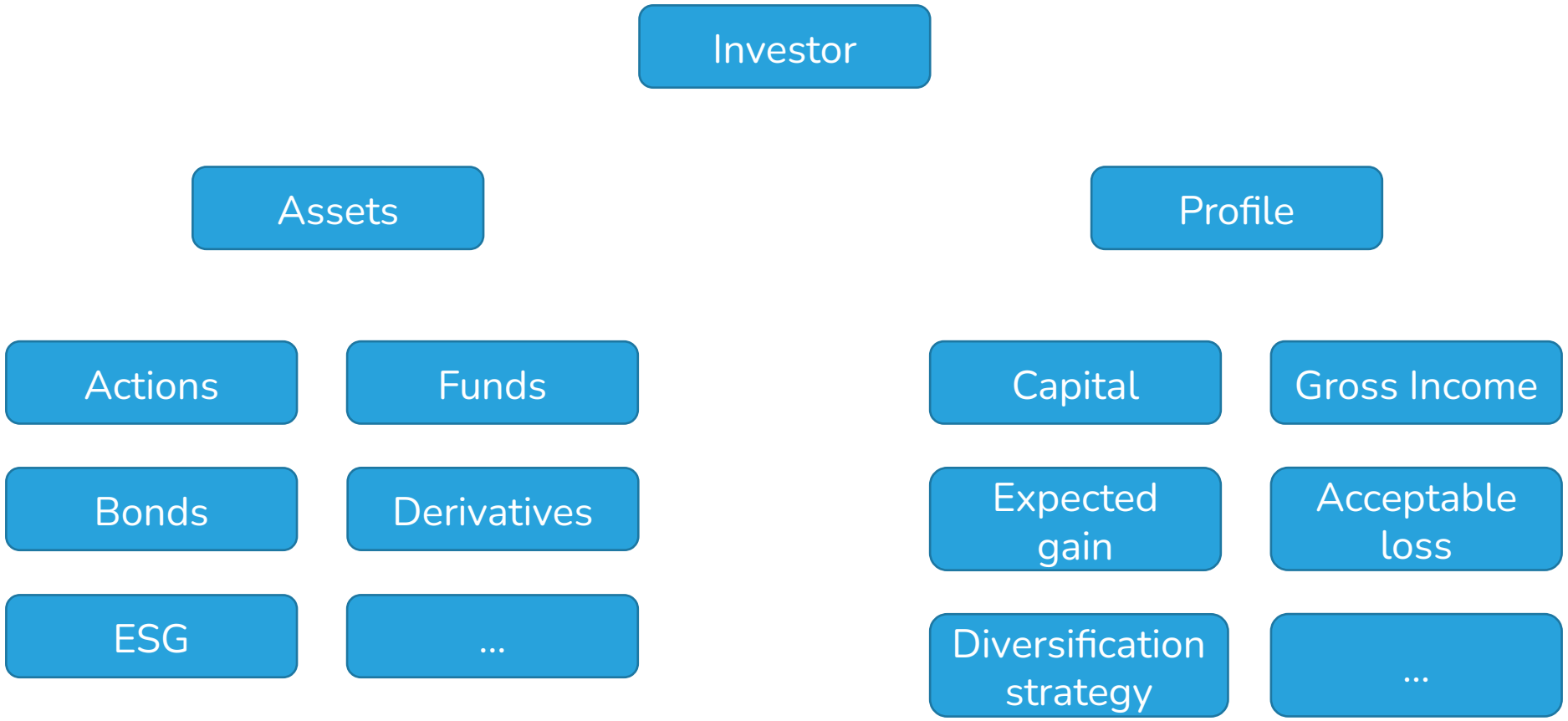
Customer for an agency teller



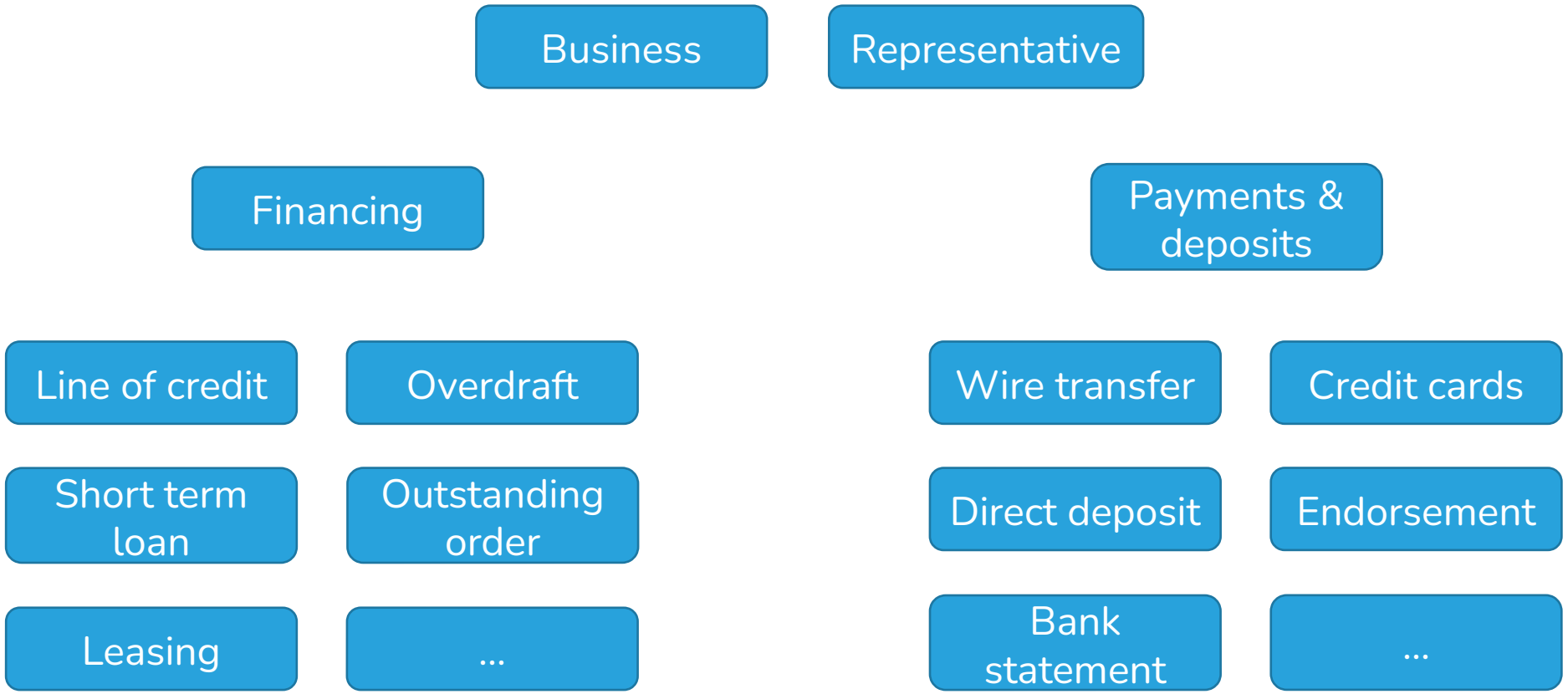
Customer for a loan department employee



Customer for an investment advisor



Customer for a corporate banker for SMEs



Is it be possible that the same word has different semantics?

We now have different definitions of a customer

A customer now has different names and properties

A customer now has different expected behaviors

Semantics depend on the context

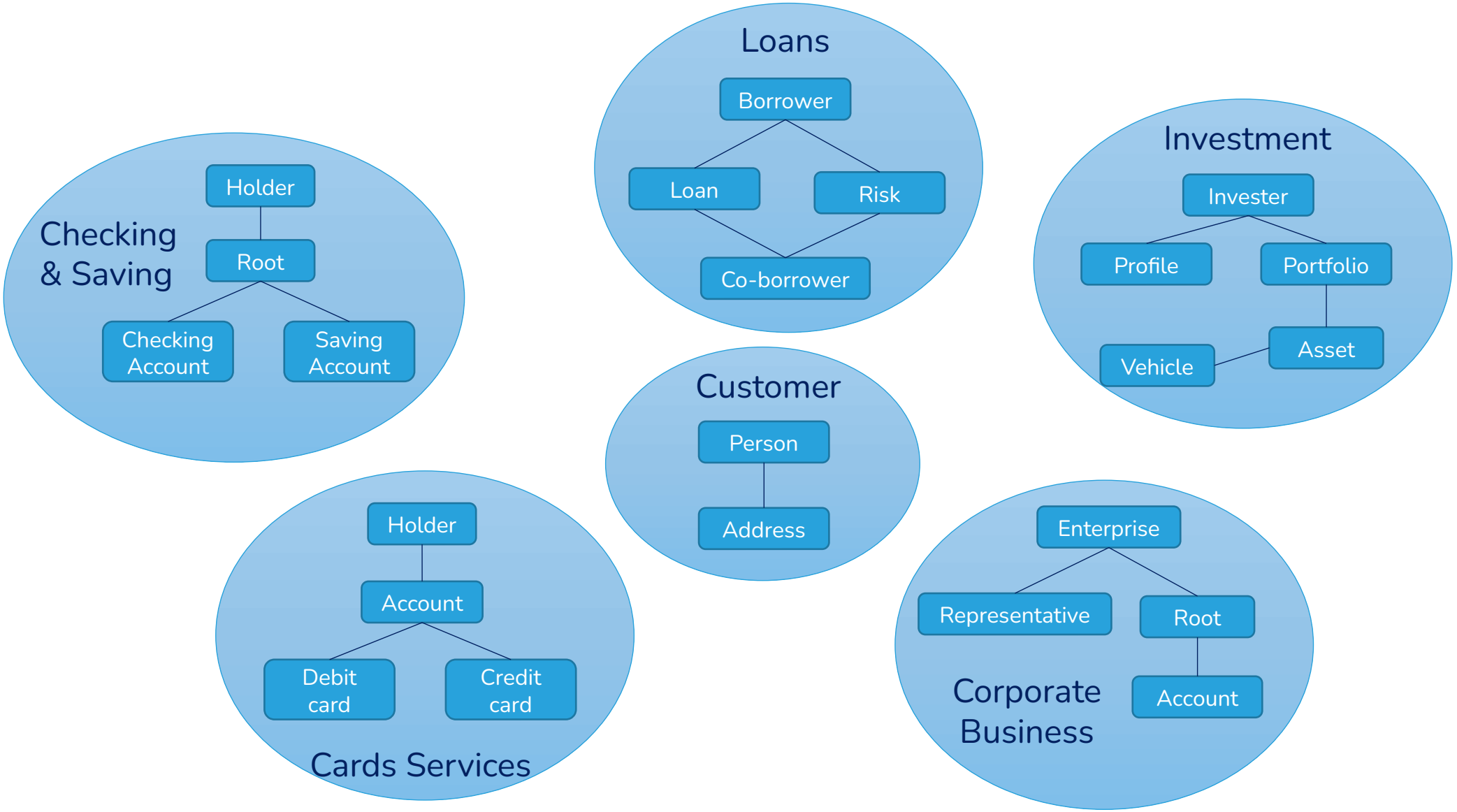
All customers are not the same

The same customer has different needs in different contexts

The bank offers different services

The bank has different obligations

The different domain experts have their own language


Loans
Borrower
Loan
Risk
Co-borrower
Investment
Invester
Profile
Portfolio
Vehicle
Asset
Checking & Saving
Holder
Root
Checking Account
Saving Account
Customer
Person
Address
Holder
Account
Debit card
Credit card
Cards Services
Enterprise
Representative
Root
Account
Corporate Business

Bounded Context

A semantic contextual boundary

A bubble of responsibility

A model of the functional sub-domain

Exposes behavior to the outside world

Hides internal state and functionality

Does that remind you of something?

What concept do you know that

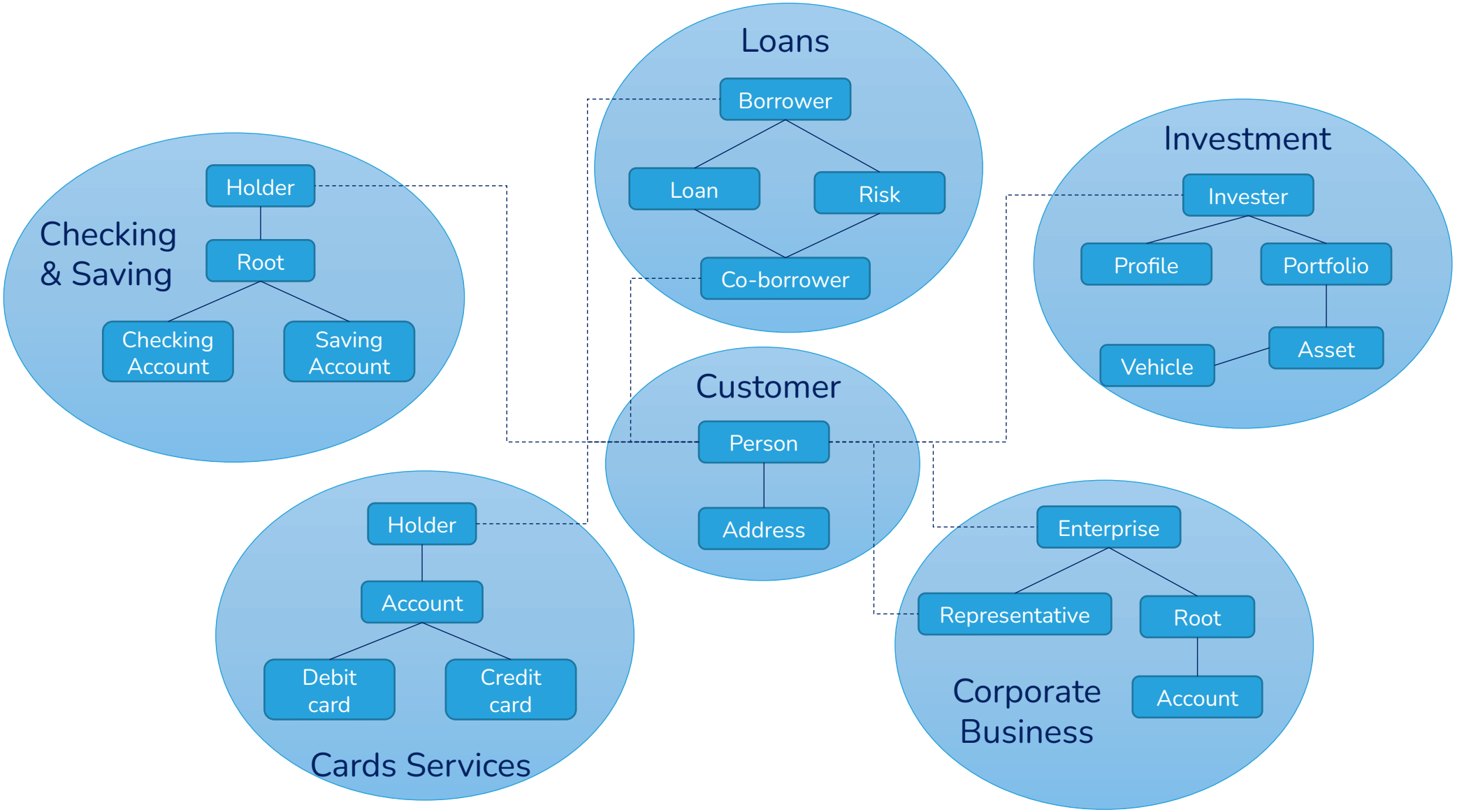
- Provides explicit boundaries
- Exposes behavior
- Hides internal state and functionality

Encapsulation at scale

A Bounded Context provides encapsulation at the strategic architectural level
It provides an abstraction that models a functional sub-domain

How do you maintain integrity?

What concept can we use to make sure that we speak of
the same customer in different contexts ?


Loans
Borrower
Loan
Risk
Co-borrower
Checking & Saving
Holder
Root
Checking Account
Saving Account
Investment
Invester
Profile
Portfolio
Asset
Vehicle
Customer
Person
Address
Holder
Account
Debit card
Credit card
Cards Services
Enterprise
Representative
Root
Account
Corporate Business

Identity

Shared identity between Bounded Contexts

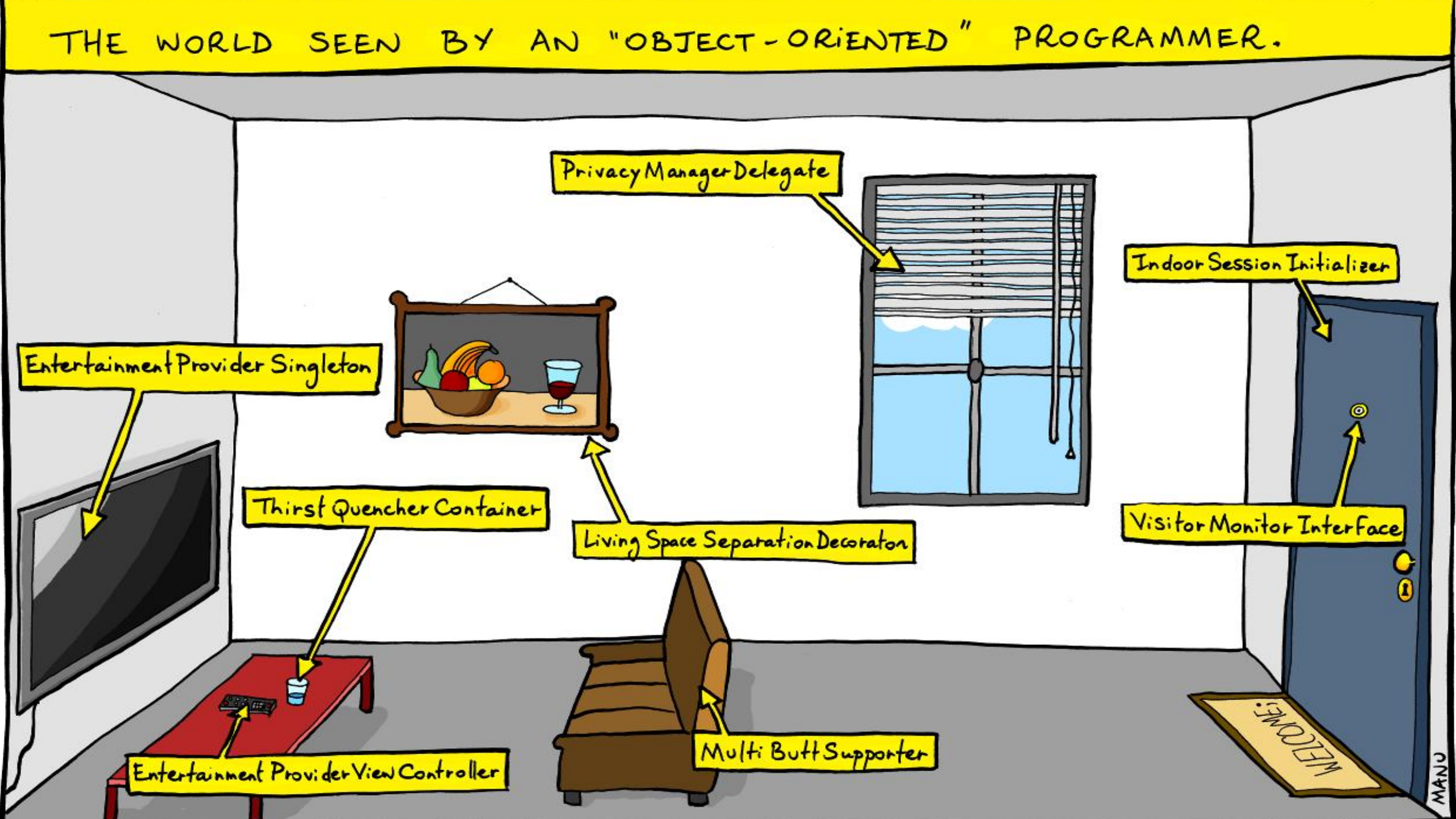
Different behaviors encapsulated in different Bounded Contexts

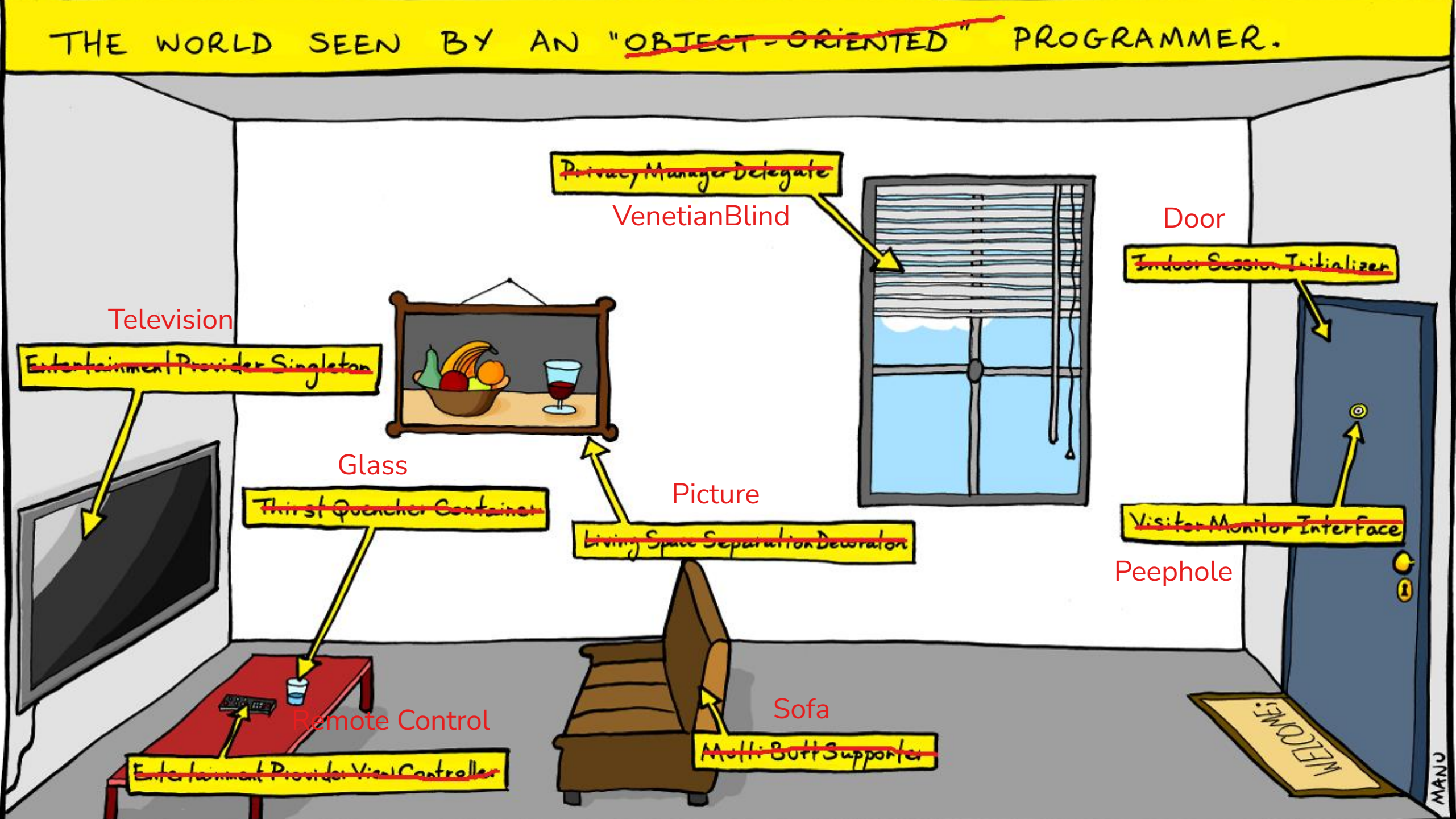
Different facets of the same person

What about the language?

What type of language do you use

- When you talk to customers / users?
- When you talk with your colleagues?
- When you talk with your family?


THE WORLD SEEN BY AN "OBJECT-ORIENTED" PROGRAMMER.
PrivacyManagerDelegate
IndoorSessionInitializer
EntertainmentProviderSingleton
ThirstQuencherContainer
LivingSpaceSeparationDecoraton
VisitorMonitorInterface
MultiButtSupporter
EntertainmentProviderViewController
WELCOME!
MANU


THE WORLD SEEN BY AN "OBJECT-ORIENTED" PROGRAMMER.
PrivacyManagerDelegate
VenetianBlind
Door
IndoorSessionInitializer
Television
EntertainmentProviderSingleton
Glass
ThirstQuencherContainer
Picture
LivingSpaceSeparationDecorator
VisitorMonitorInterface
Peephole
Remote Control
EntertainmentProviderViewController
Sofa
MultiButtSupporter
WELCOME.
MANU

Ubiquitous Language

The language is ubiquitous within the boundary of the context

Words used by the domain experts

One word describes only one business concept

A business concept is only described by one word

No ambiguity

The definition is shared by all within the context

Who is in charge of the Bounded Contexts?

How do you call a group of people who share practices and tools in order to achieve a common goal?

Bounded Contexts and teams

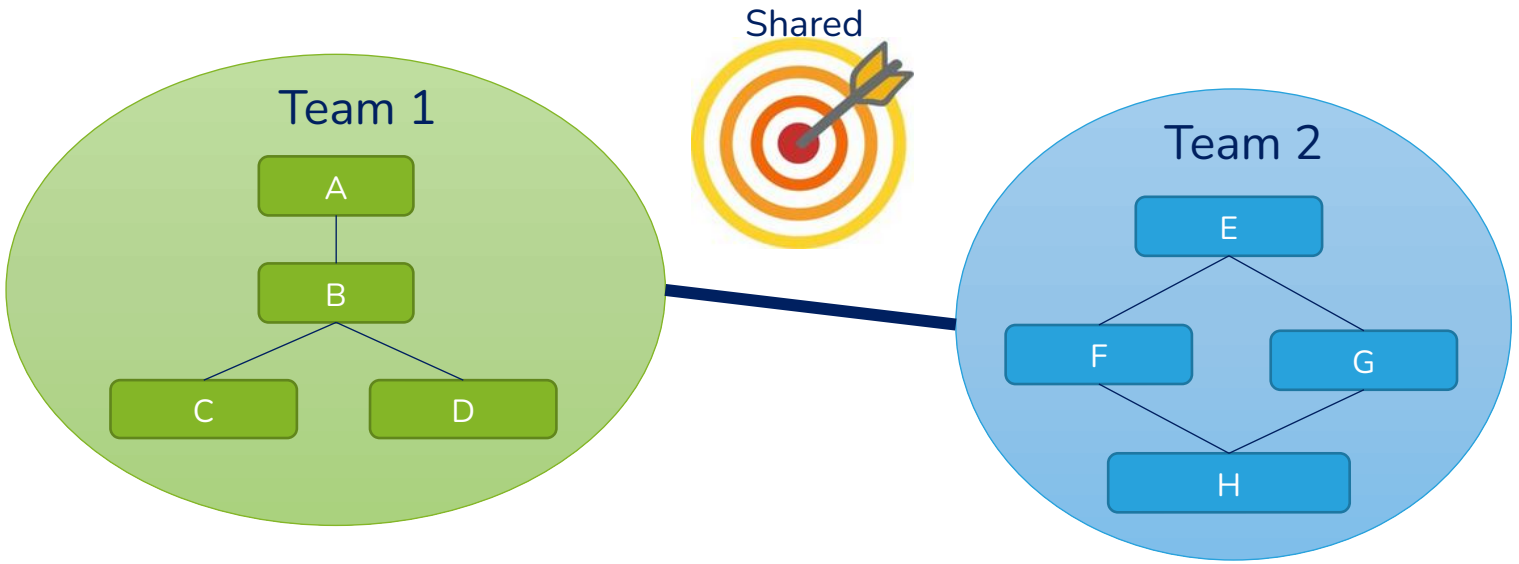
A bounded context belongs to one team only

A team may own more than one Bounded Context

Coupling

A & B are coupled if
when you want to change A
you also need to change B

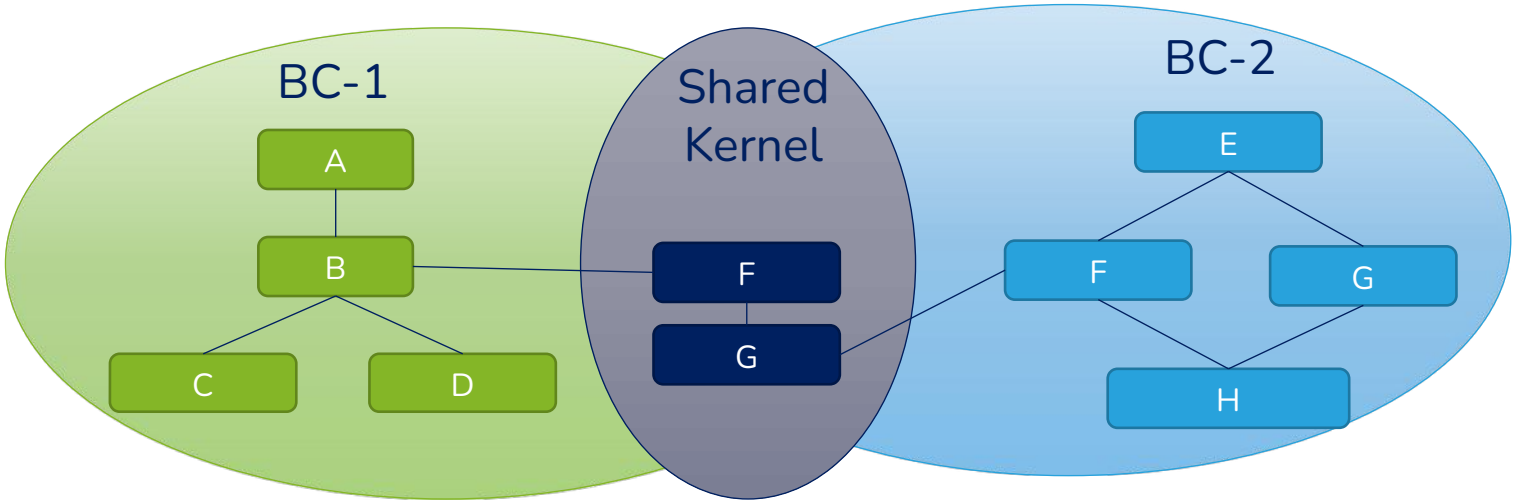
Partnership



the two teams will succeed or fail together

Coupling?

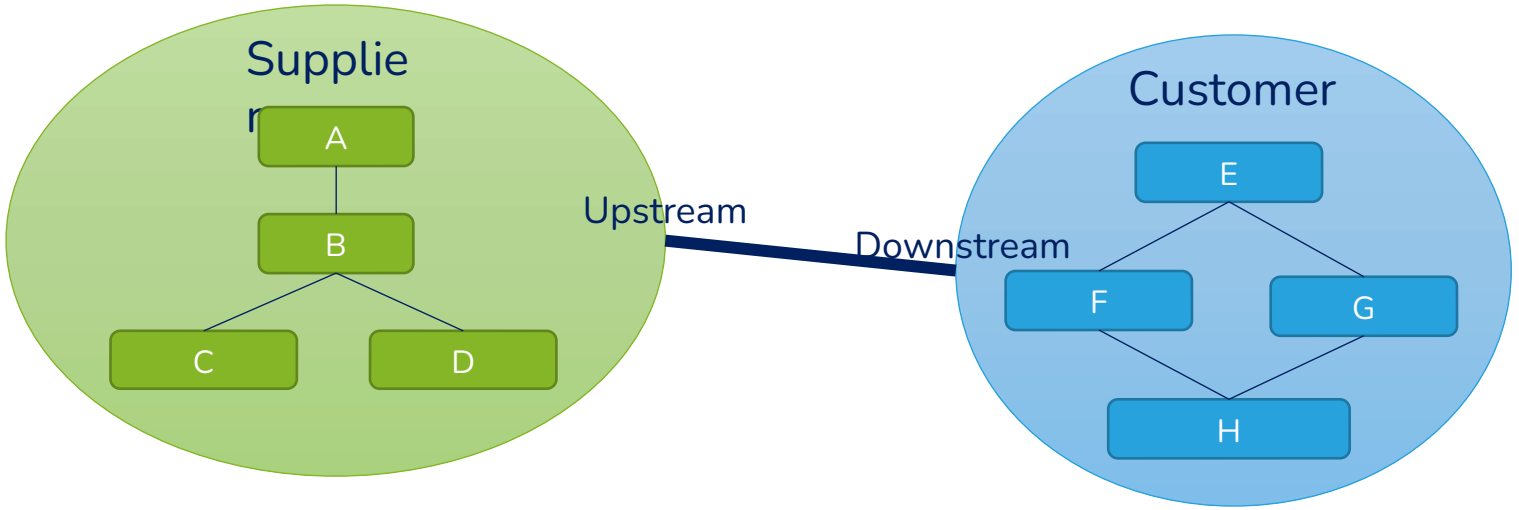
Shared kernel



the teams must agree on what model elements they are to share

Coupling?

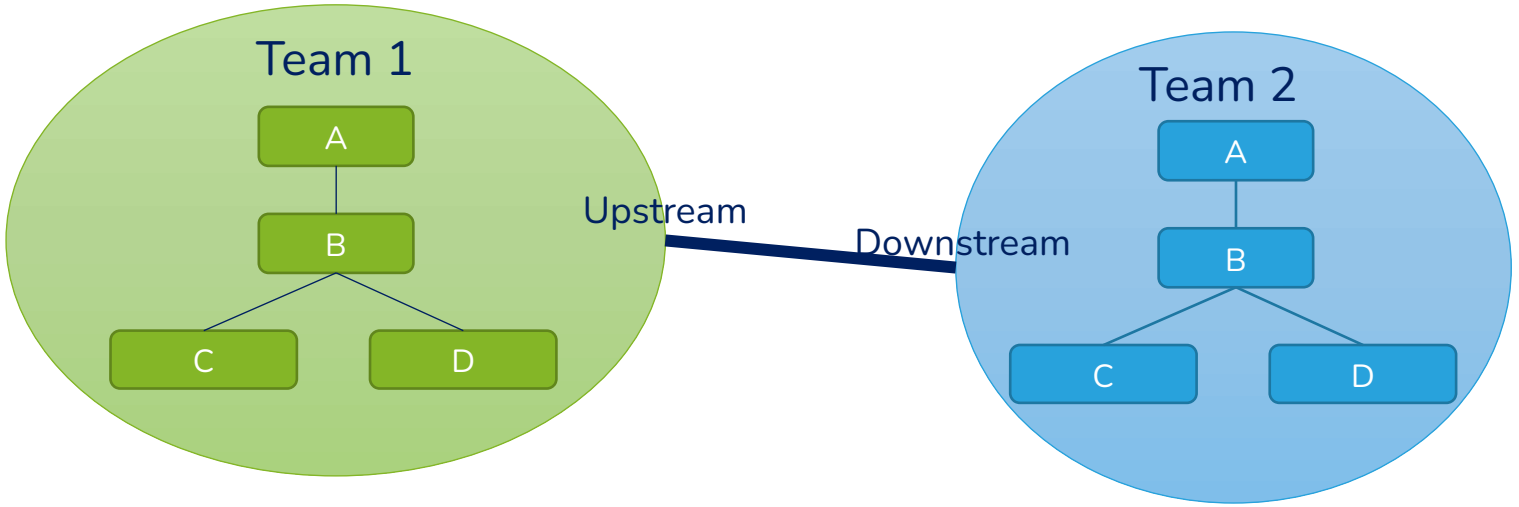
Customer-Supplier



the *Supplier* determines what the *Customer* will get and when

Coupling?

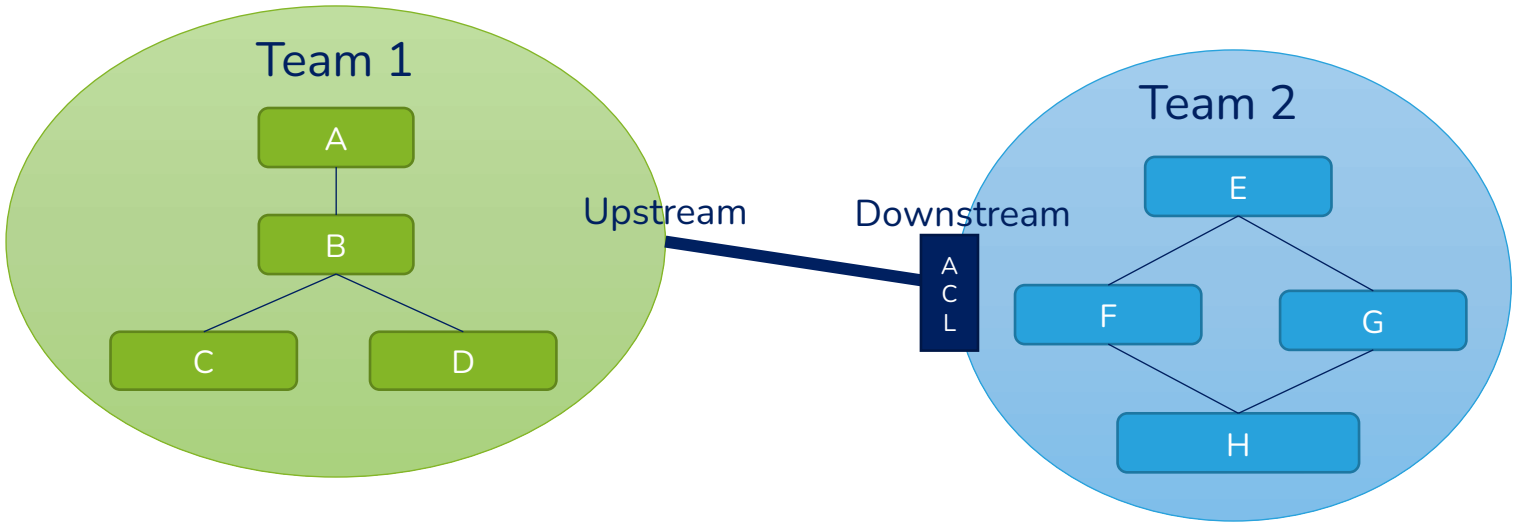
Conformist



the downstream team conforms to the upstream model

Coupling?

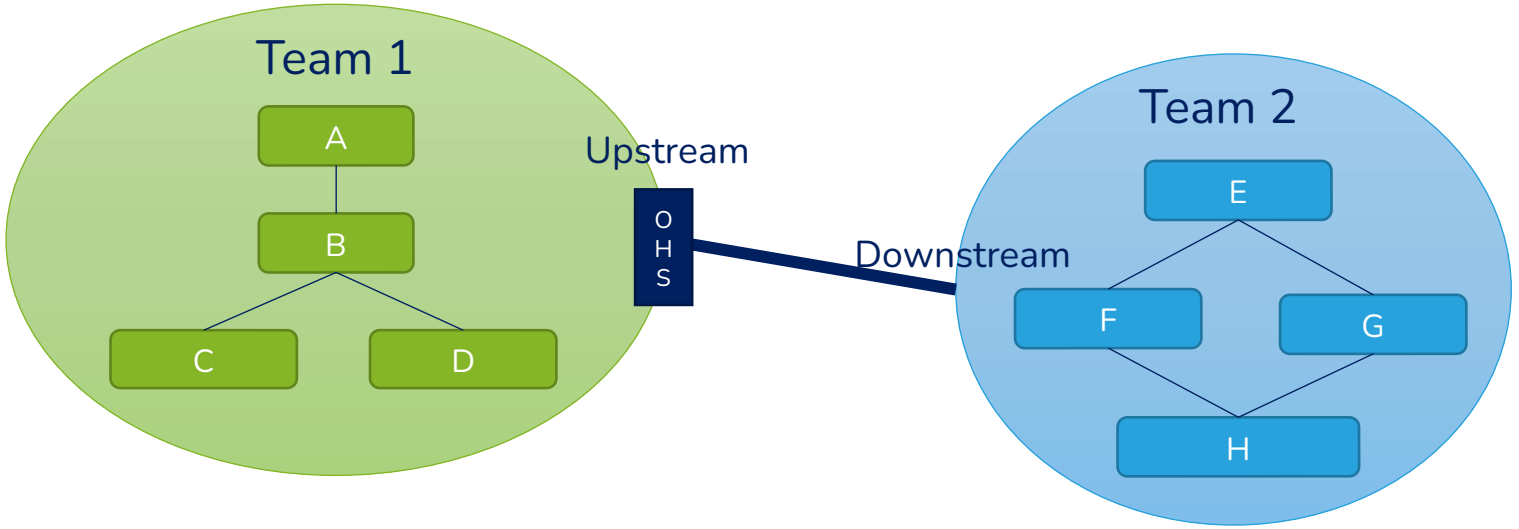
Anticorruption Layer



the anticorruption layer isolates the downstream model
from the upstream model and translates between the two

Coupling?

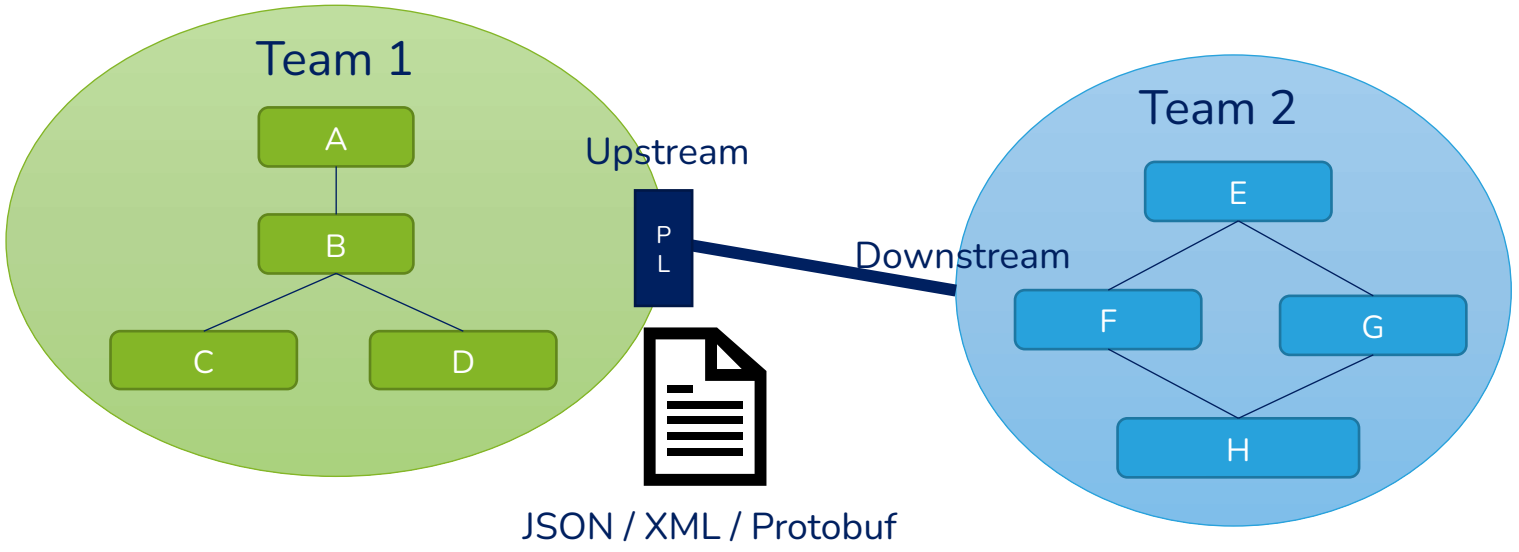
Open Host Service



an “open” protocol or interface gives access to your *Bounded Context* as a set of services so that all who need to integrate with it can with relative ease

Coupling?

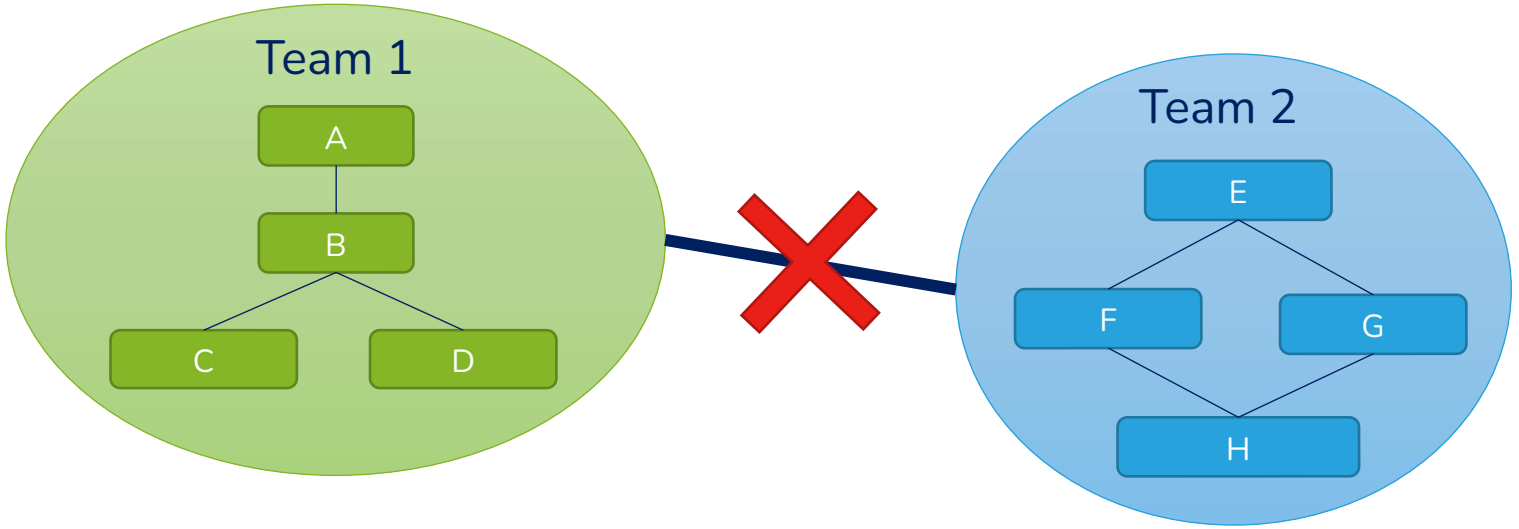
Published Language



a well-documented information exchange language enabling simple consumption and translation by any number of consuming *Bounded Contexts*

Coupling?

Separate ways



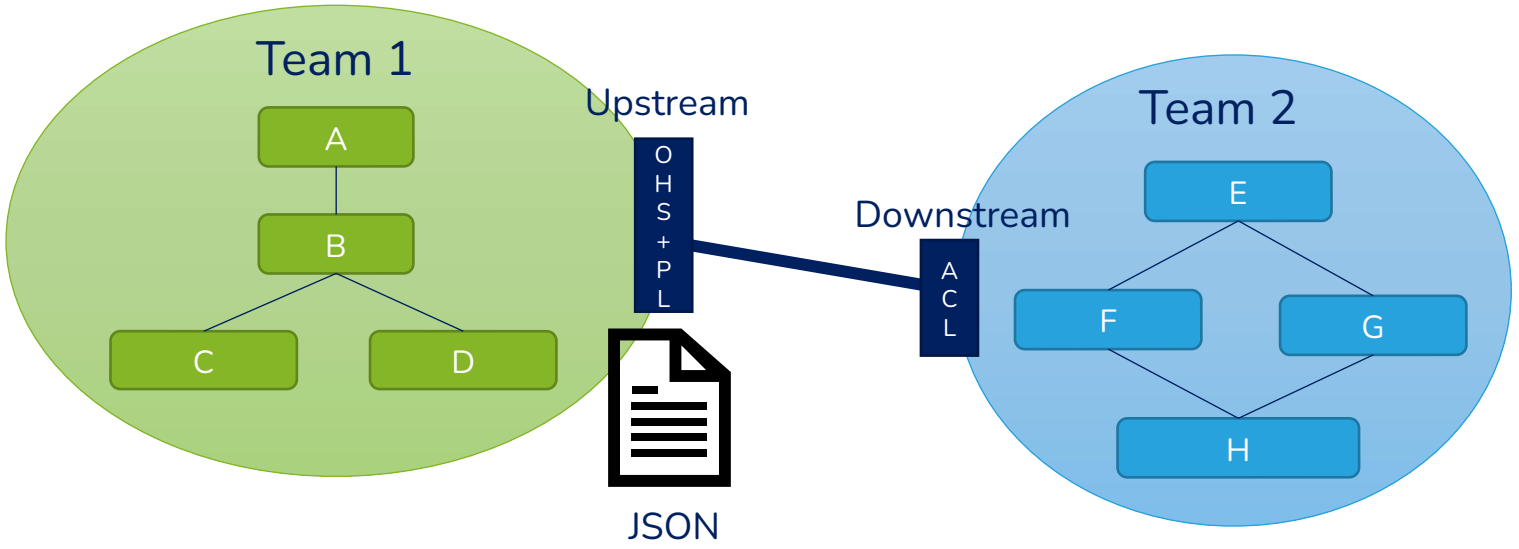
Team 2 produces its own specialized solution in its *Bounded Context* and forgets integrating with team 1

Coupling?

Does that remind you of something?

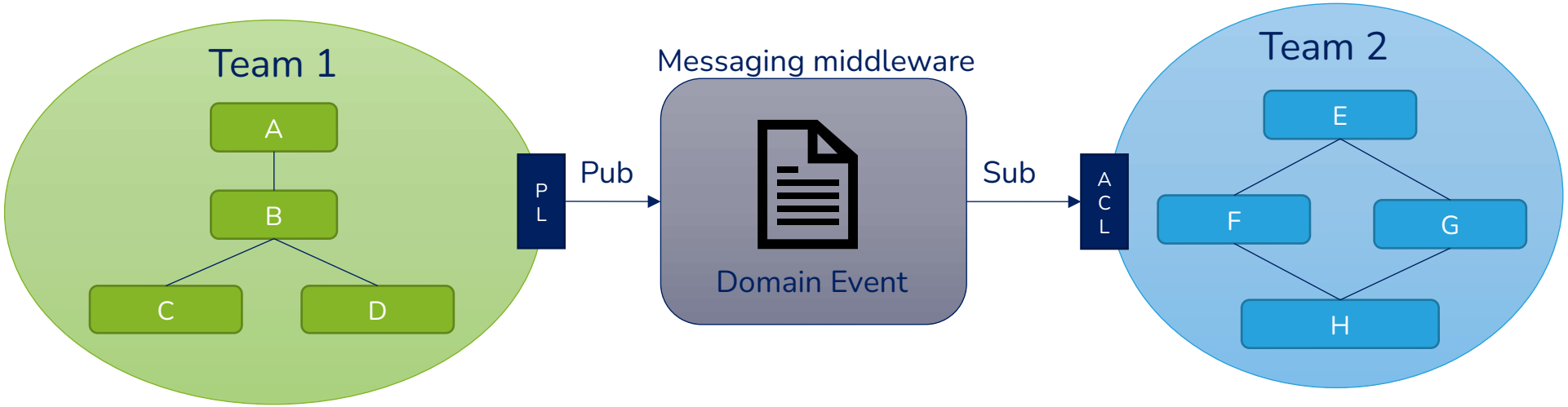
What mechanisms do you know that would ease the integration between two Bounded Contexts?

REST



Coupling

Pub/Sub



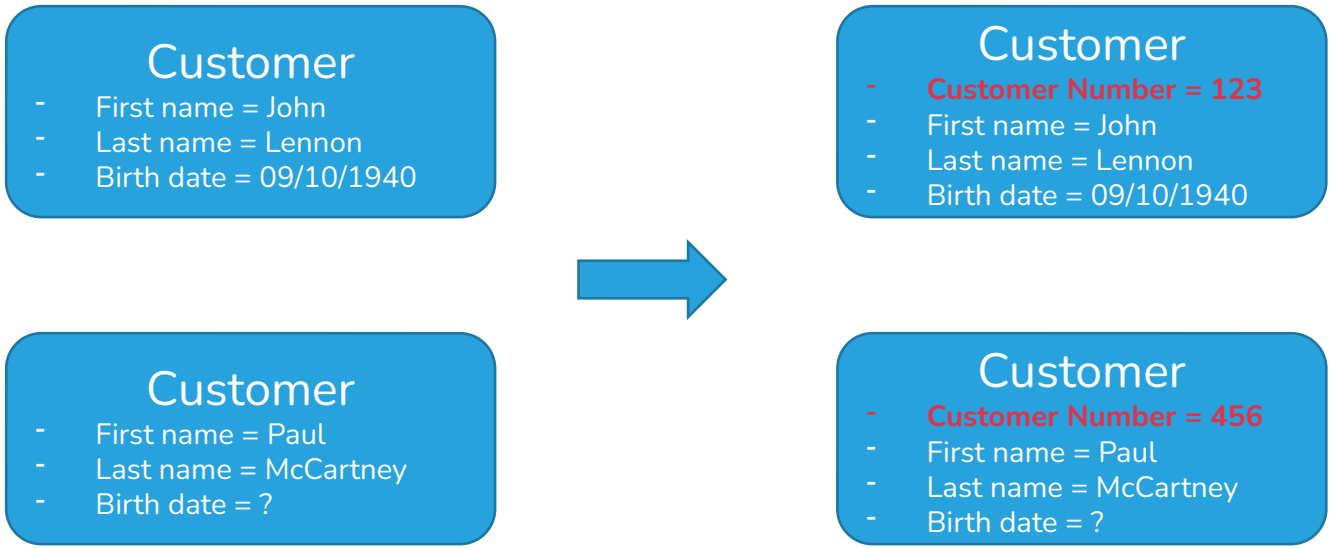
Coupling

How would you create the model within a given Bounded Context?

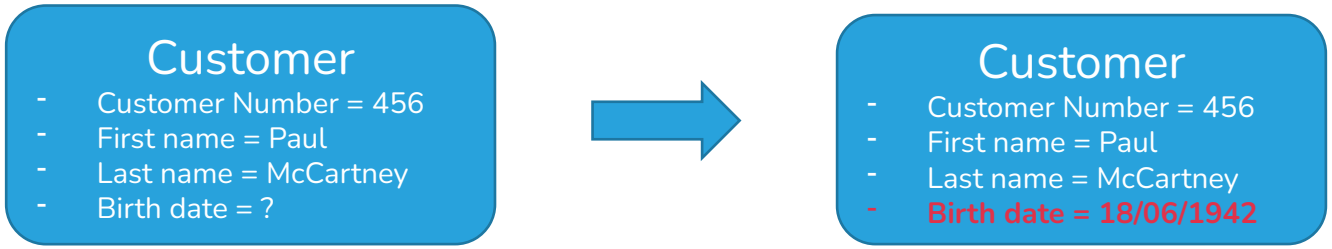
What tools do you know that allow you go from domain to model?

How do you go from problem space to solution space?

How do we distinguish these two customers?



Do you know Paul's birth date?



Entity

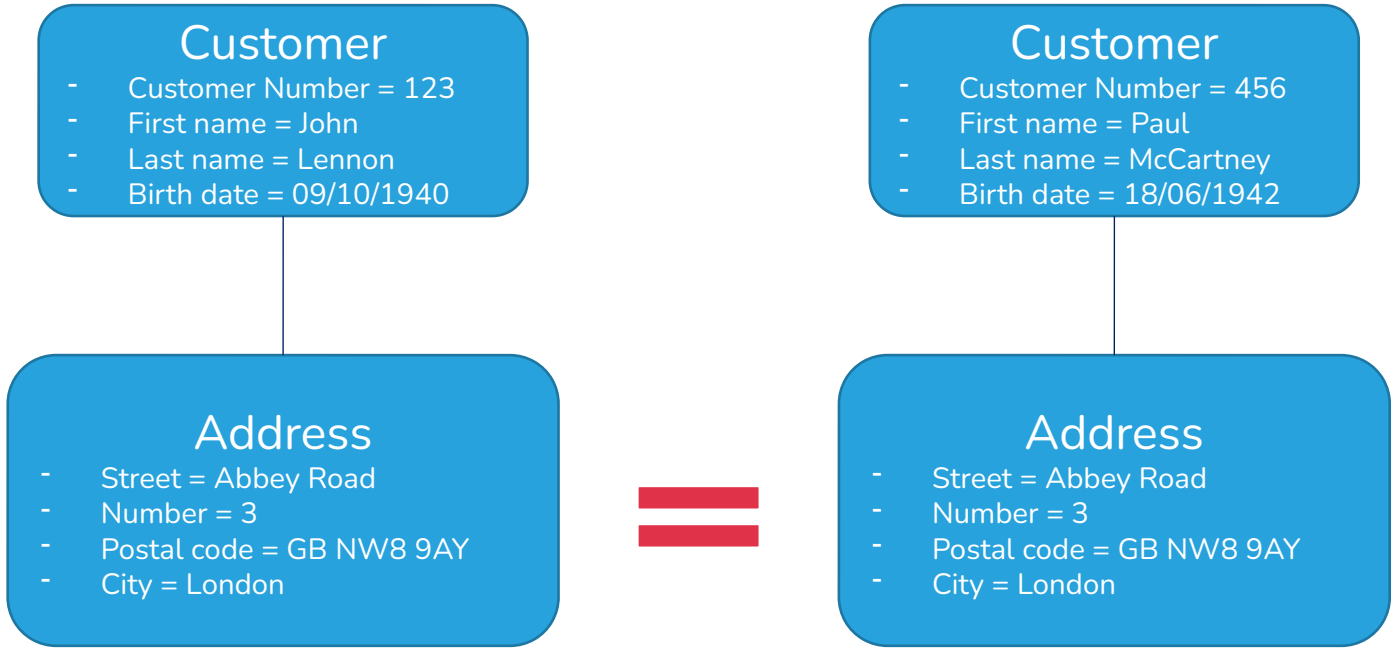
An individual thing

It has a unique identity

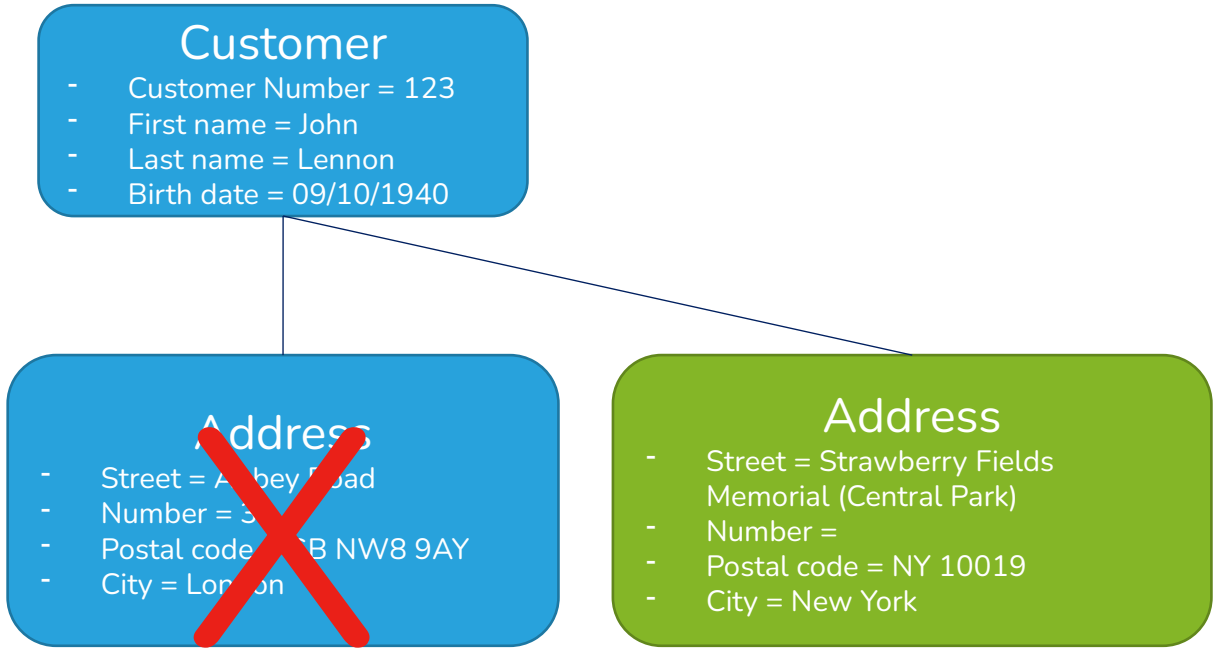
Usually mutable

Changing over time

Do these two customer leave at the same address?



What happens when the customer moves?



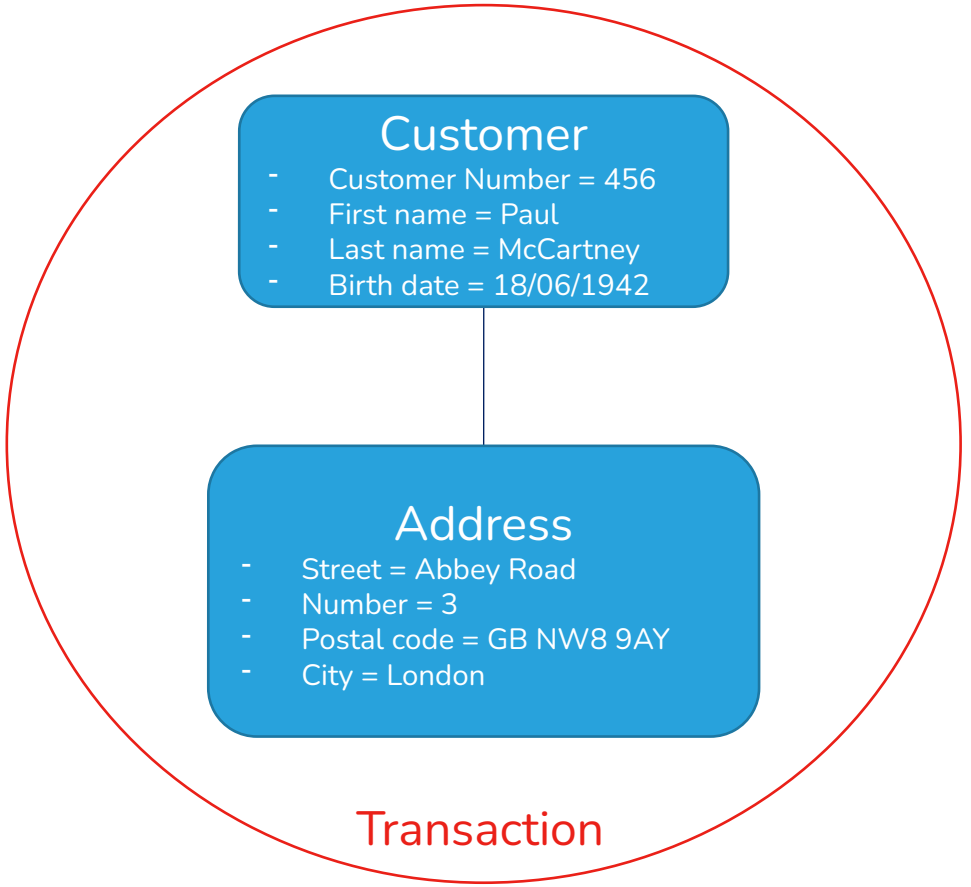
Value object

An immutable conceptual whole

It does not have a unique identity

Equivalence is determined by comparing the attributes it encapsulates

What happens when you persist this in a database?



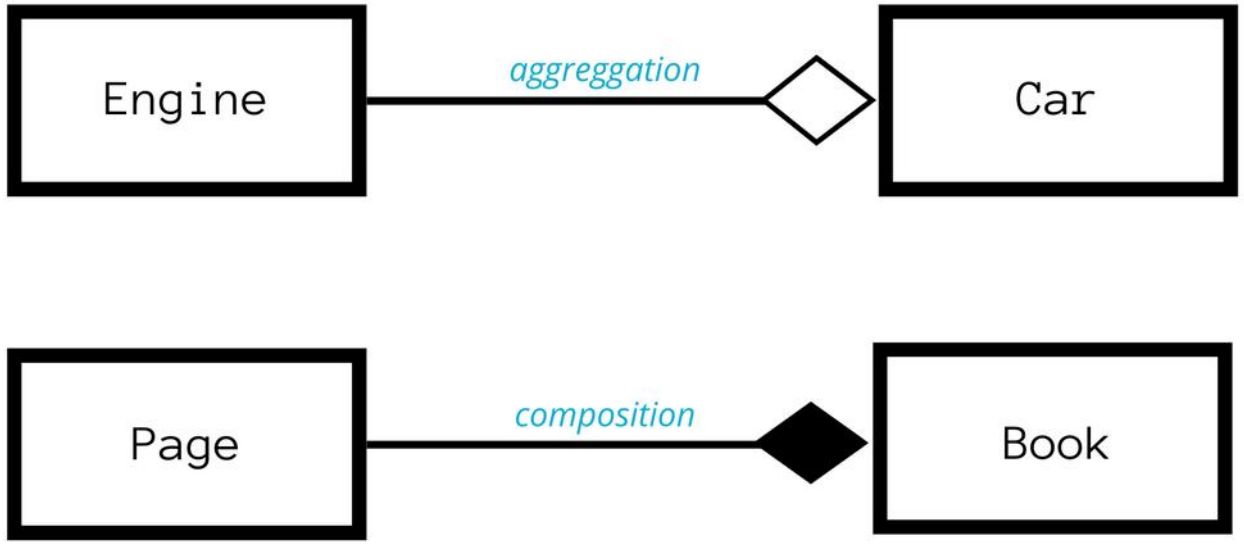
Aggregate

A transactional consistency boundary

Loaded and persisted as a whole

All composed parts must be consistent according to business rules

Aggregation vs Composition



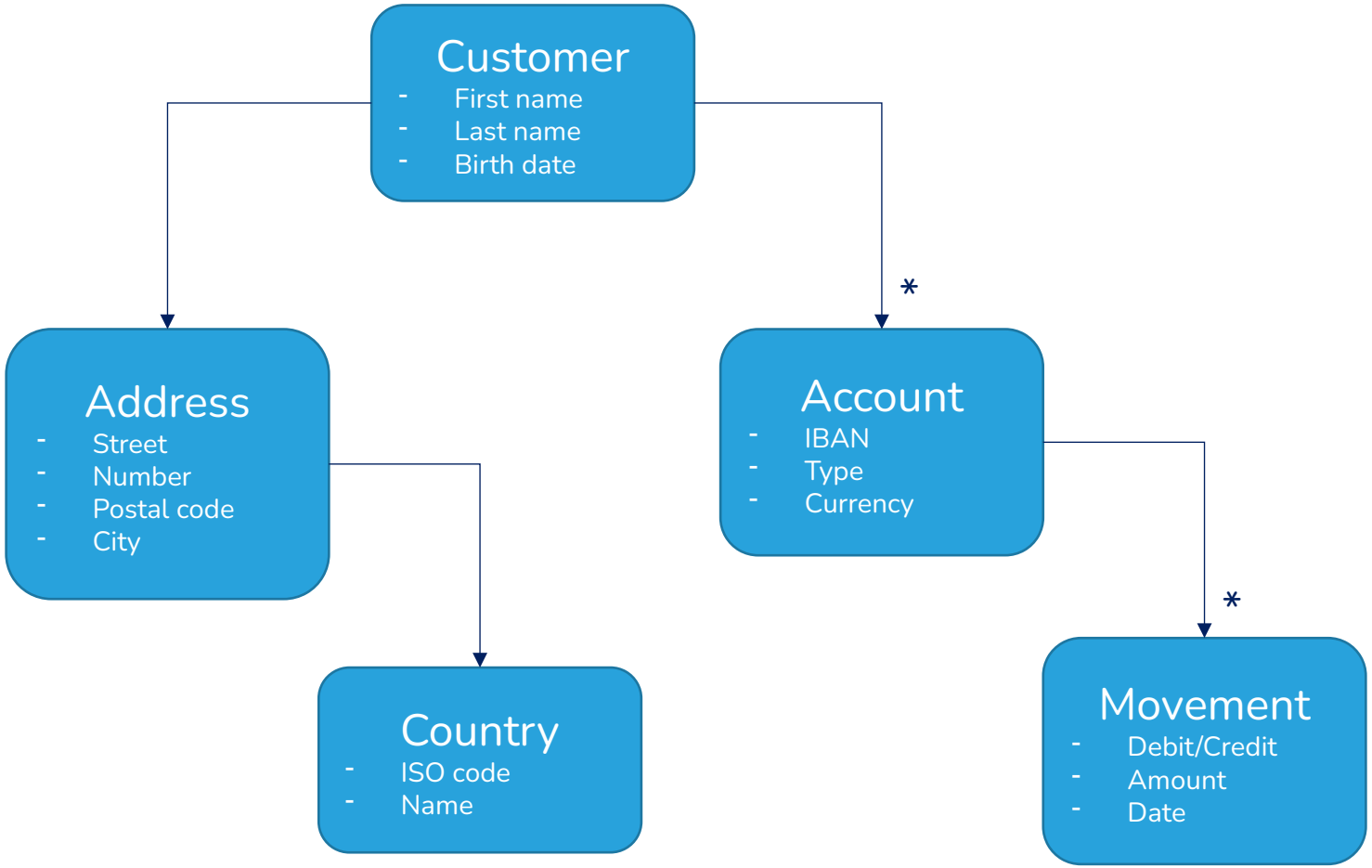
In an aggregation, the part may have an independent lifecycle, it can exist independently.

When the whole is destroyed, the part may continue to exist.

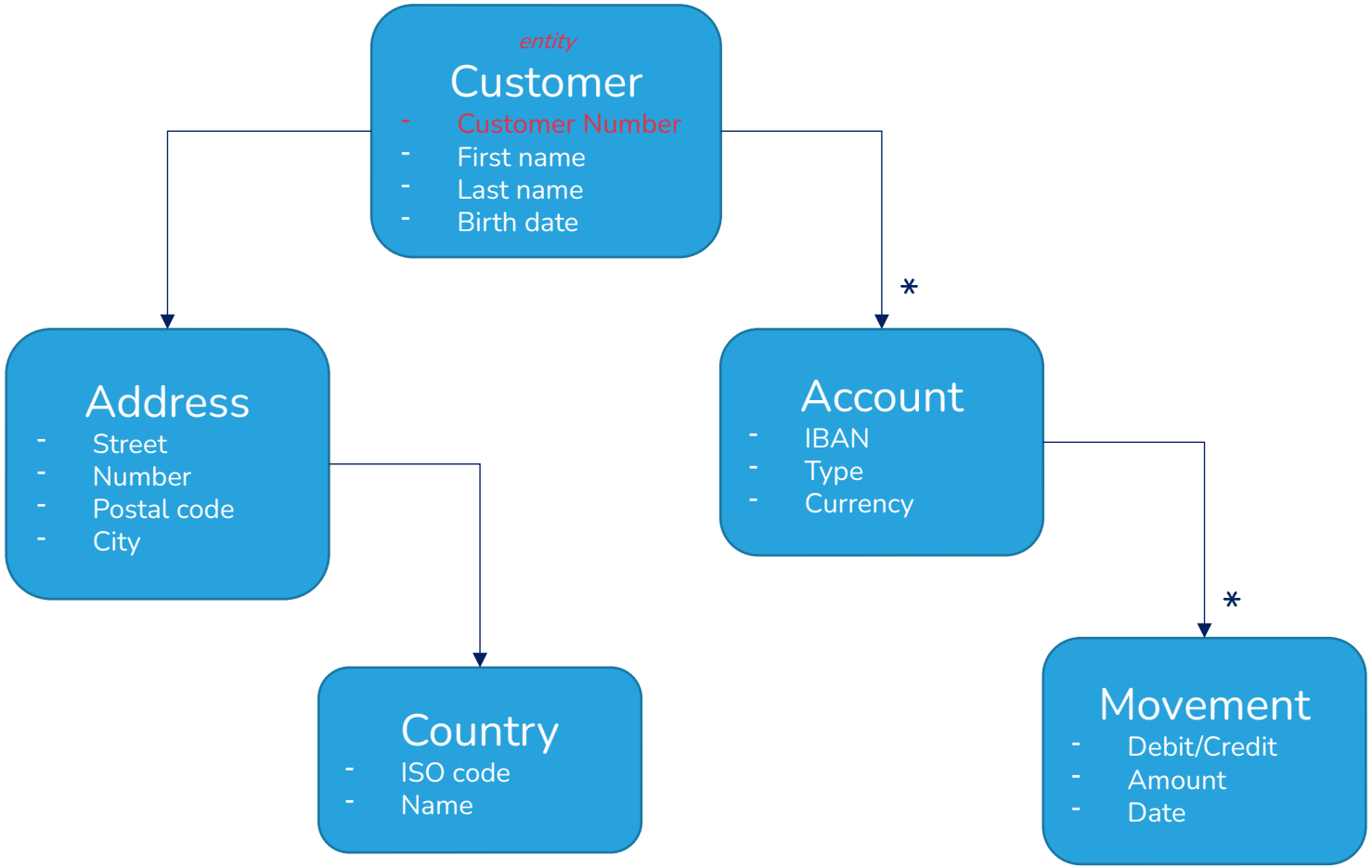
Knowing all this...

how would we adapt our “naive” Customer model?

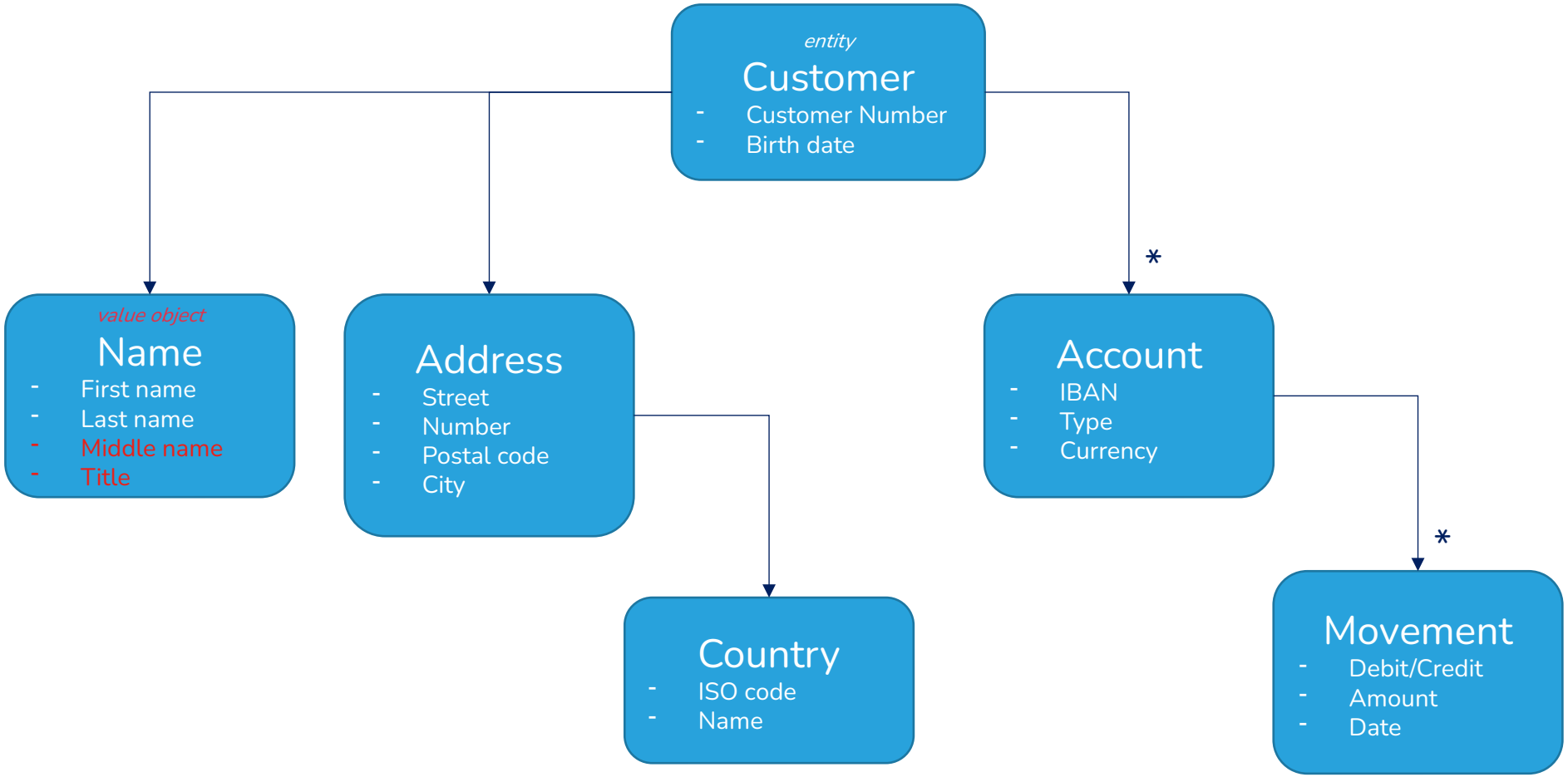
How would we adapt this model?



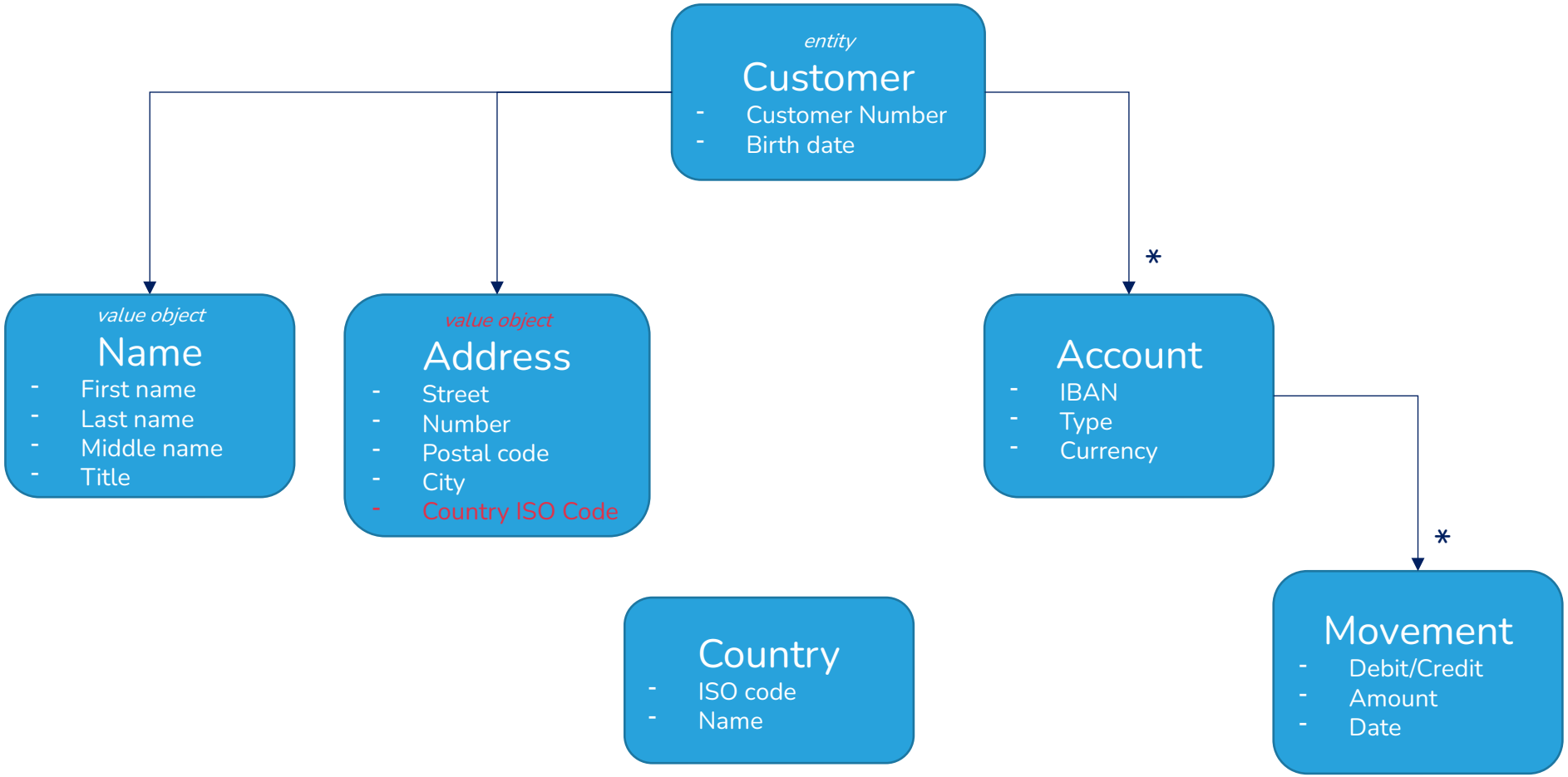
How would we adapt this model?



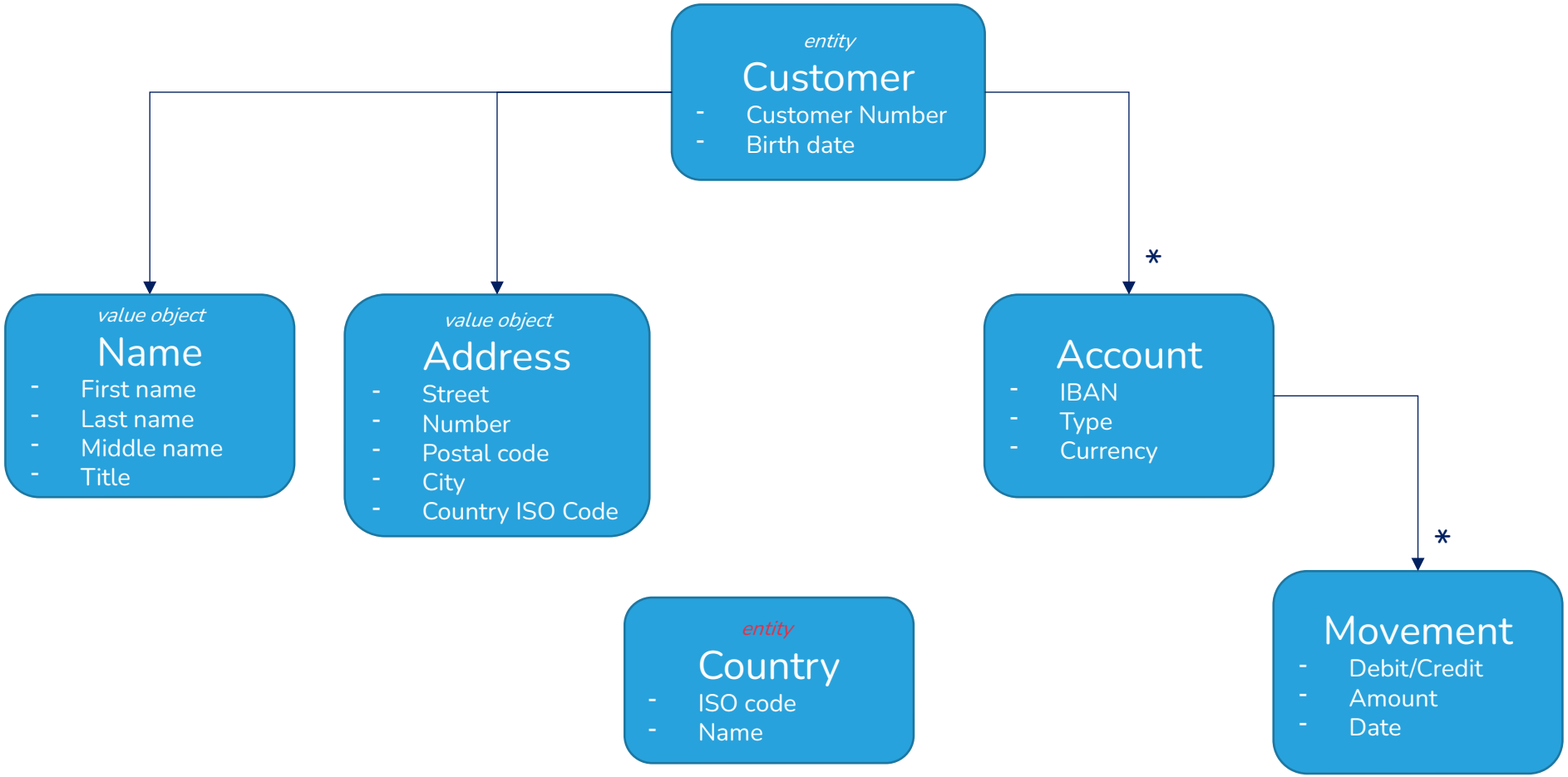
How would we adapt this model?



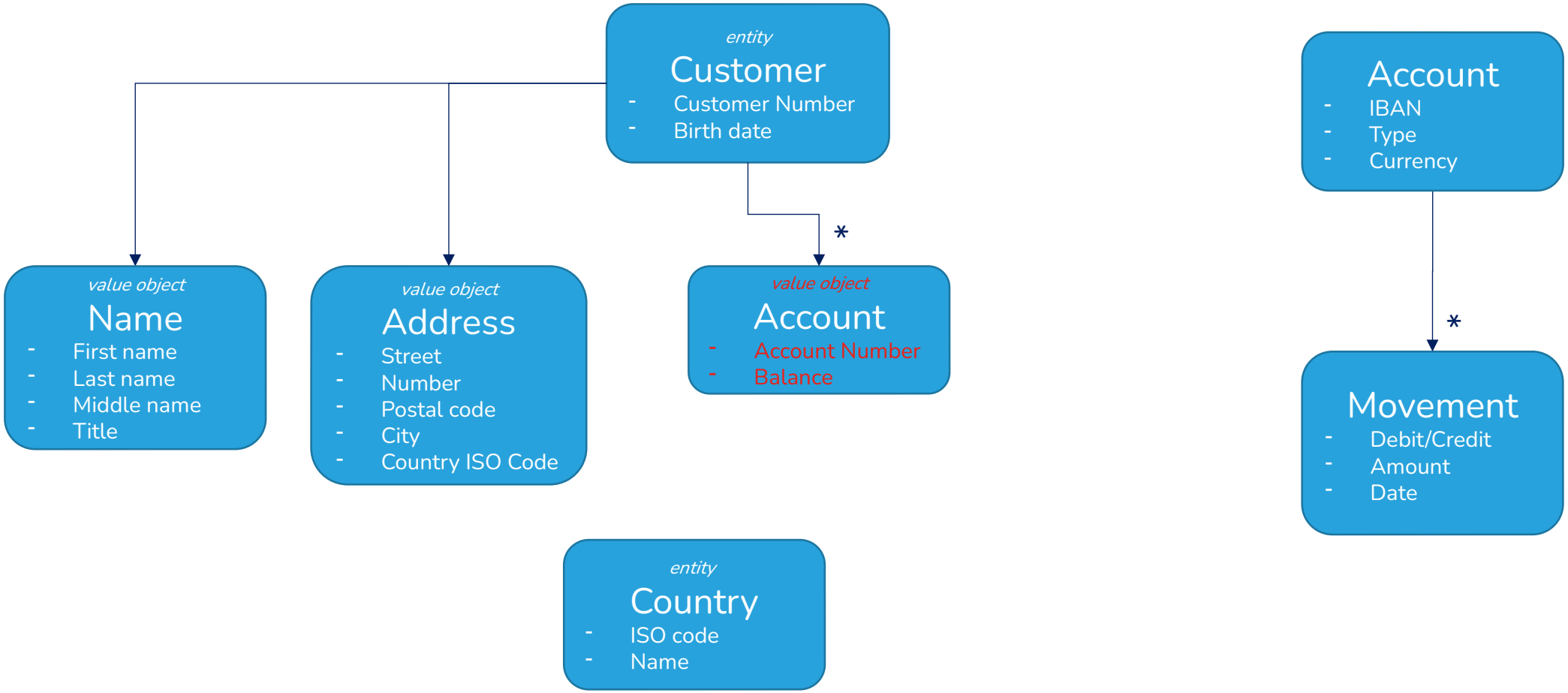
How would we adapt this model?



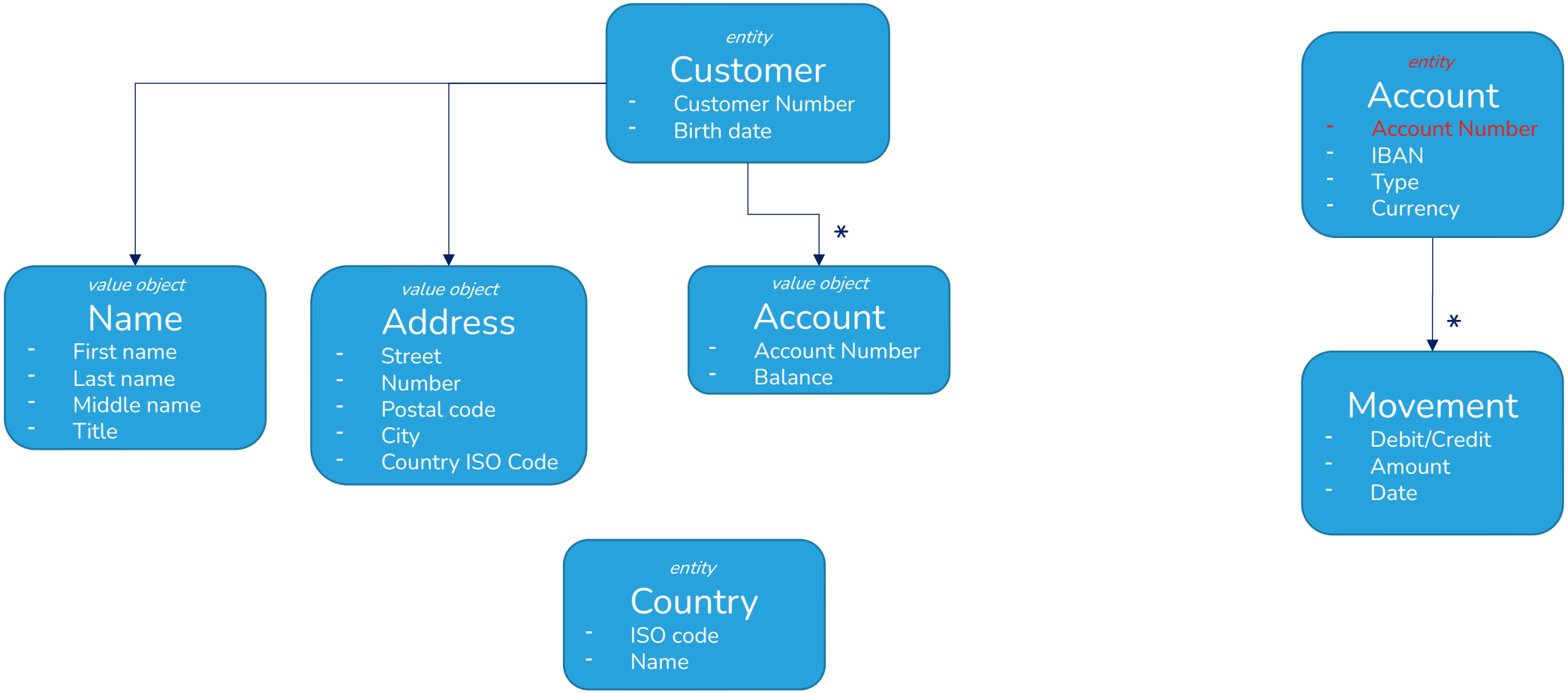
How would we adapt this model?



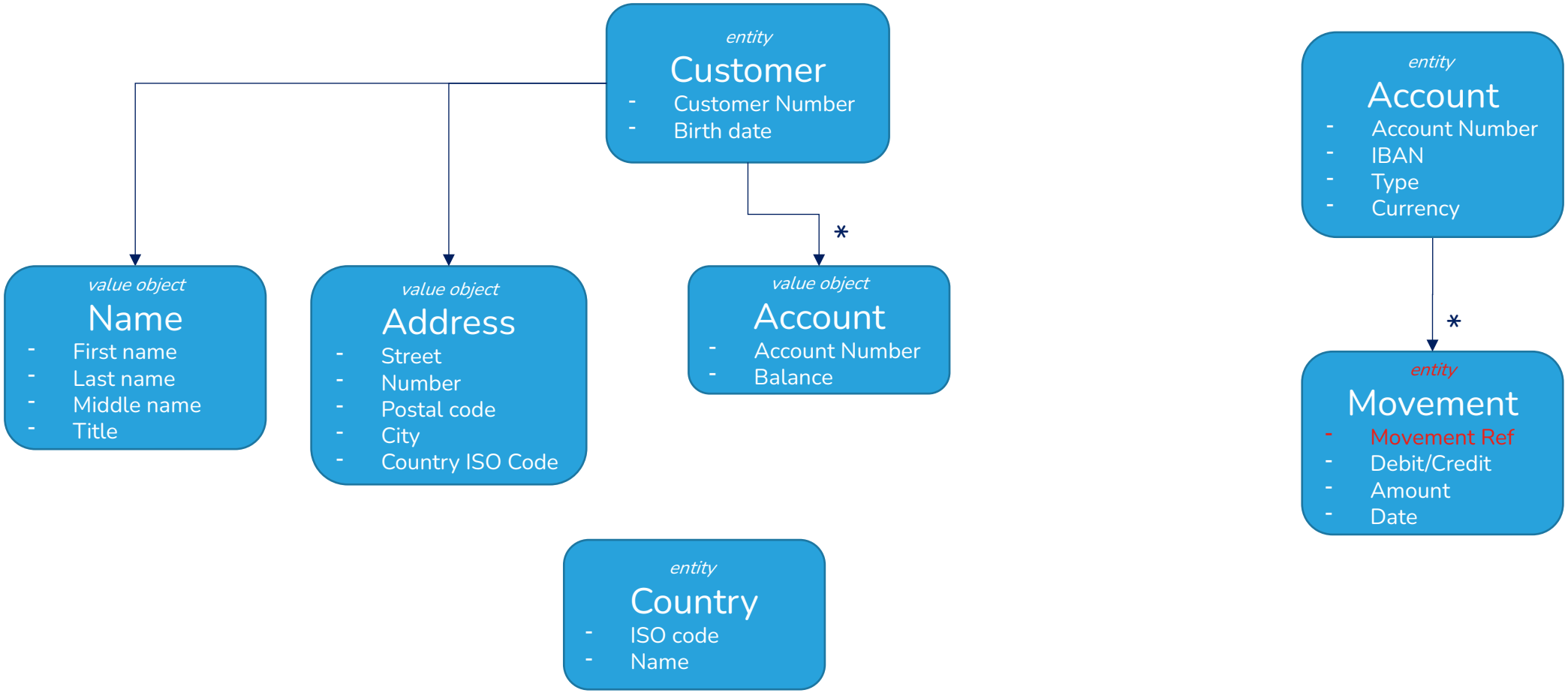
How would we adapt this model?



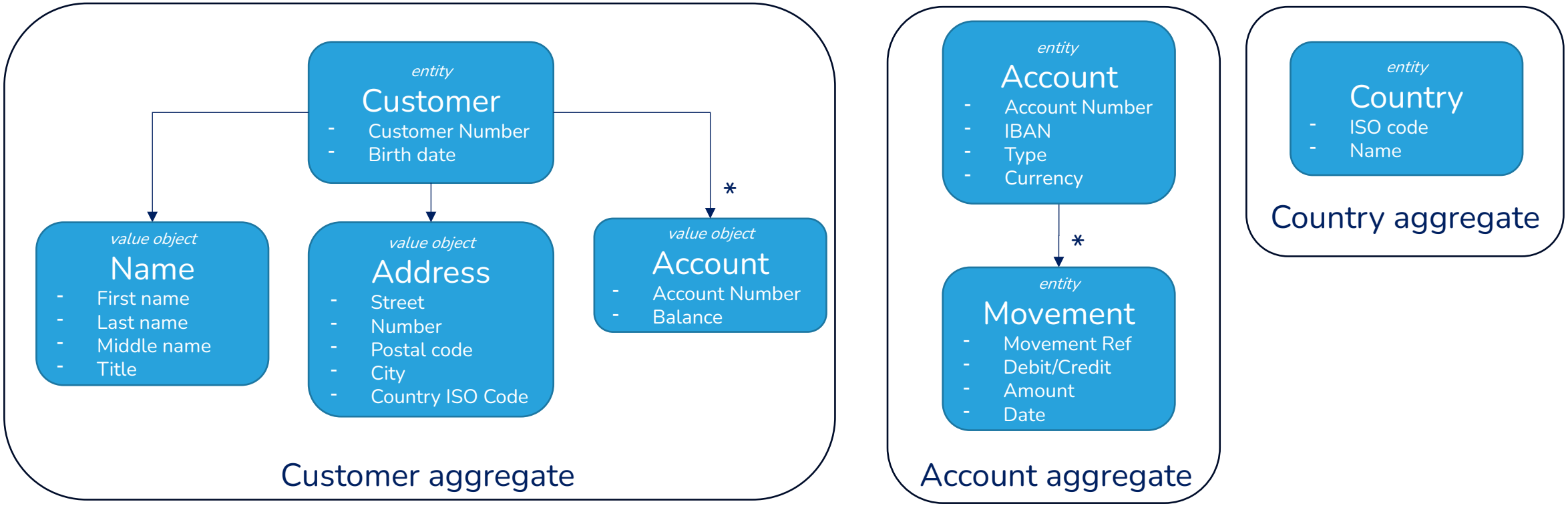
How would we adapt this model?



How would we adapt this model?



How would we adapt this model?



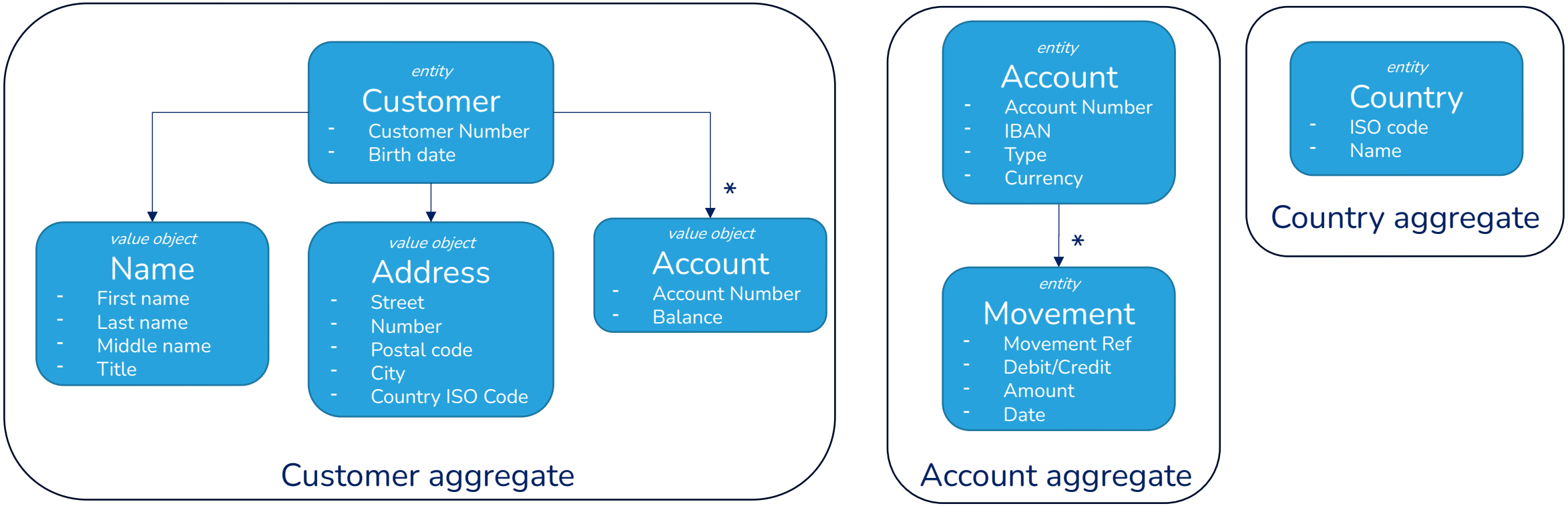
The question is not about finding the right model from the start

The question is about being able to change the model over time

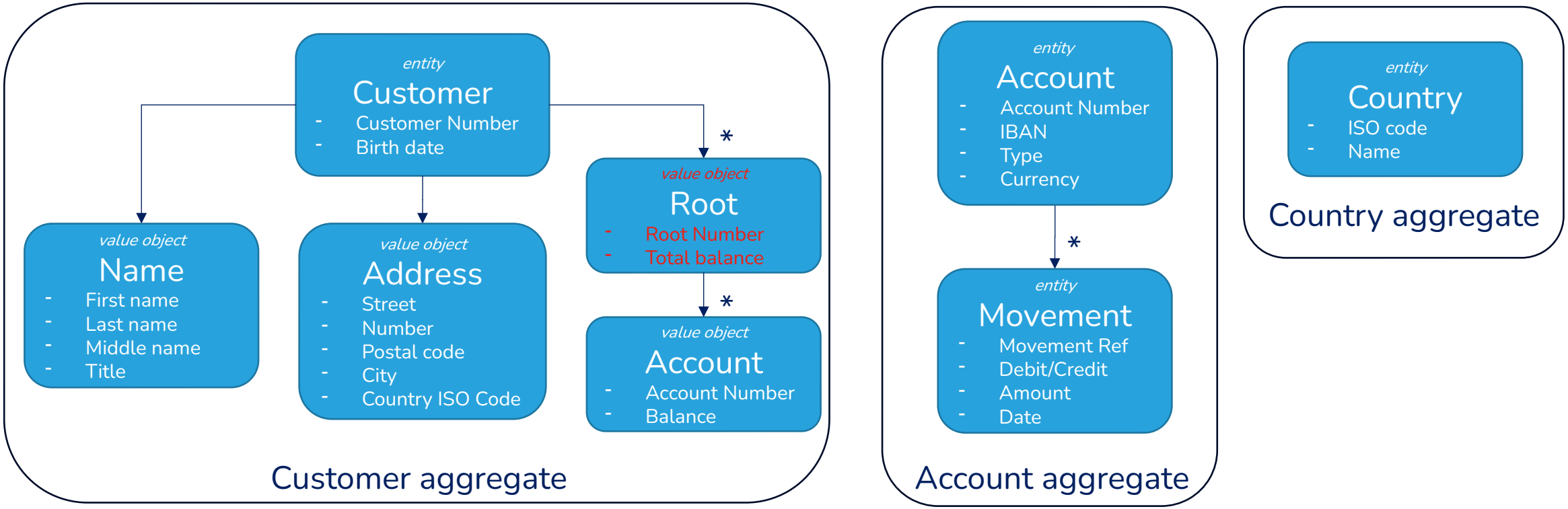
Can we easily add a new notion?

Let’s try to add the notion of Root for a bank account

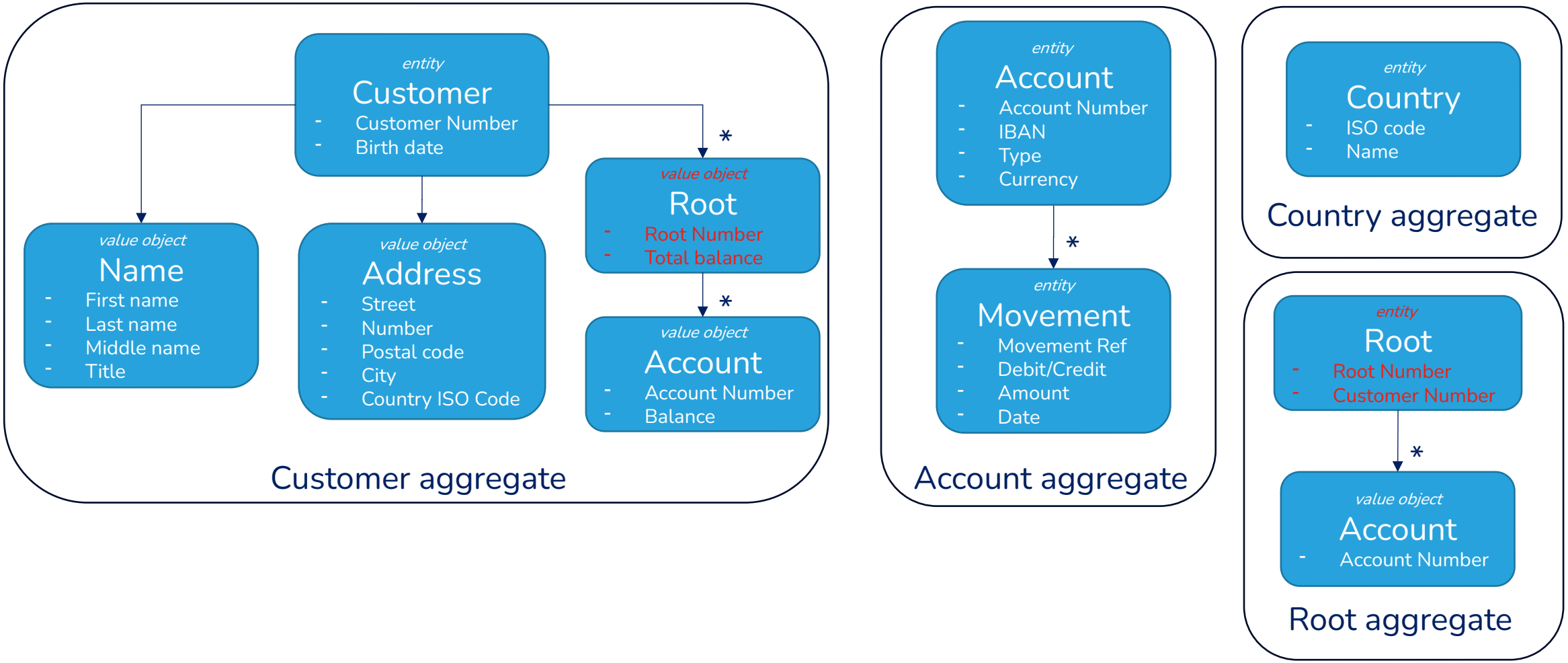
How would we adapt this model?



How would we adapt this model?



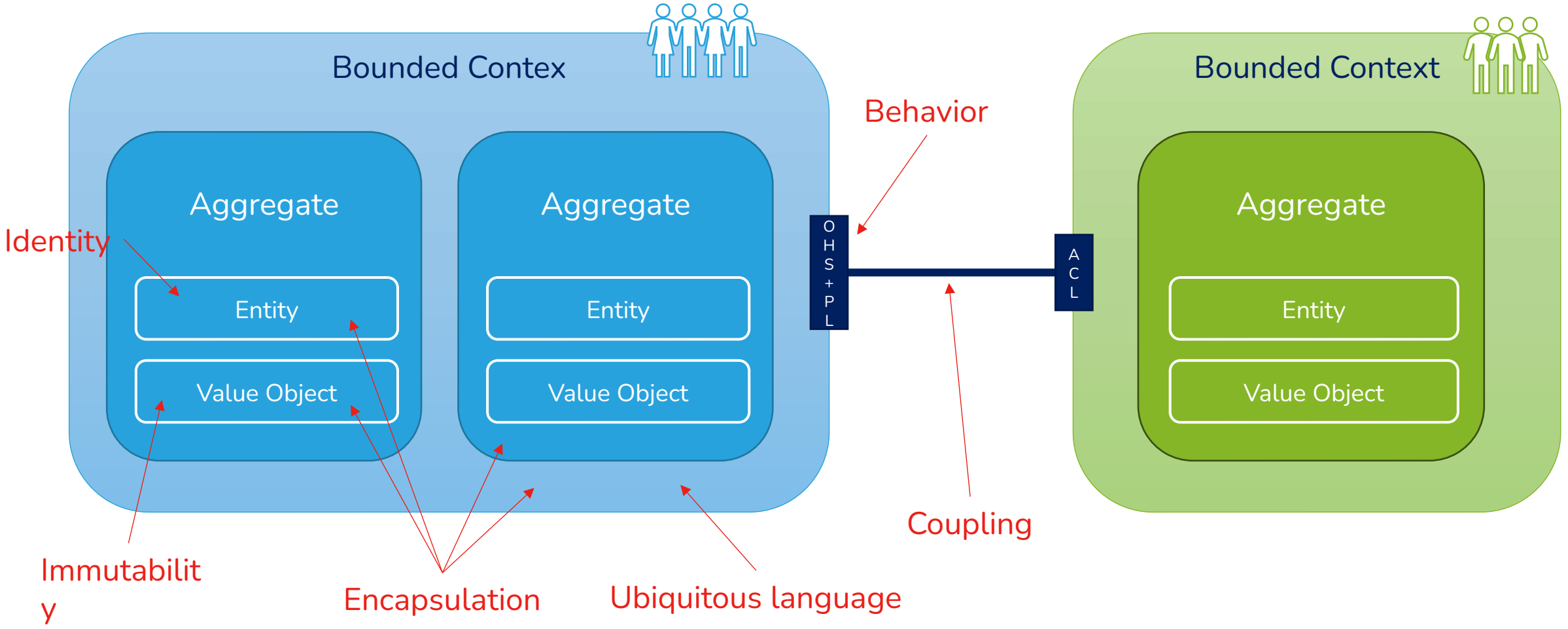
How would we adapt this model?



Let’s summarize

Let’s do it bottom-up

Summary


Bounded Contex
Aggregate
Entity
Value Object
Aggregate
Entity
Value Object
Identity
Behavior
O H S + P L
A C L
Bounded Context
Aggregate
Entity
Value Object
Coupling
Immutabilit y
Encapsulation
Ubiquitous language

Was that so complicated?

Now, it’s time for questions

Merci !