Data Mesh Information Architecture

Modeling data products and domains

About me

Juha Korpela

- Based in Helsinki, Finland
- 15+ years in Data Management in a variety of industries with diverse expertise:
  - Data Products
  - Data Governance
  - Data Modeling
  - Data Engineering
  - Data Strategy
- Entrepreneur & Consultant @ Datakor Consulting
- Founder & Chairman @ Helsinki Data Week
- Podcast host @ Helsinki Data Mafia
- Speaker: Data Day Texas, IRM UK Enterprise Data, Data Mesh Live...
- ex-CPO @ Ellie Technologies Inc.
- ex-Head of Data Platform @ UPM-Kymmene Oyj

DATAKOR CONSULTING

Follow me on LinkedIn:
linkedin.com/in/jkorpela

Agenda

- Data Mesh basics
- Conceptual models for cross-domain understanding
- Hands-on exercise: modeling a domain
- Data modeling as part of data product design
- Ensuring semantic interoperability at the domain boundary
- Data Mesh information architecture operating model
- Conclusions

Goals

- Understand the basics of the Data Mesh paradigm and its challenges relating to information architecture and semantics
- Learn the basics of conceptual modeling as a method of defining the business context of domains and data products
- Learn the basics of logical modeling as a part of data product design process
- Learn how solution-level metadata (e.g. data contracts) can expose domain-level context across domain boundaries
- Understand the basic operating model of information architecture management in the context of independent domain teams within a Data Mesh setup

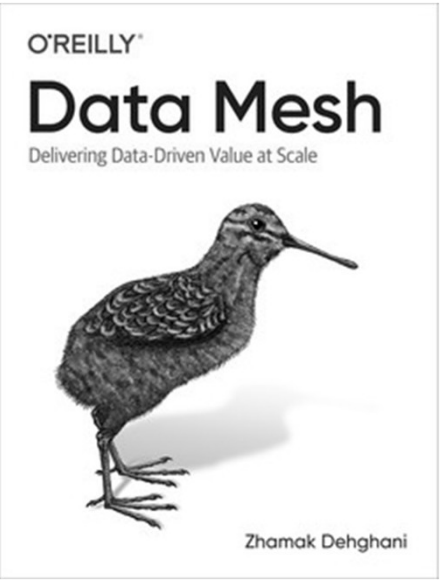
Data Mesh basics

Data Mesh basics

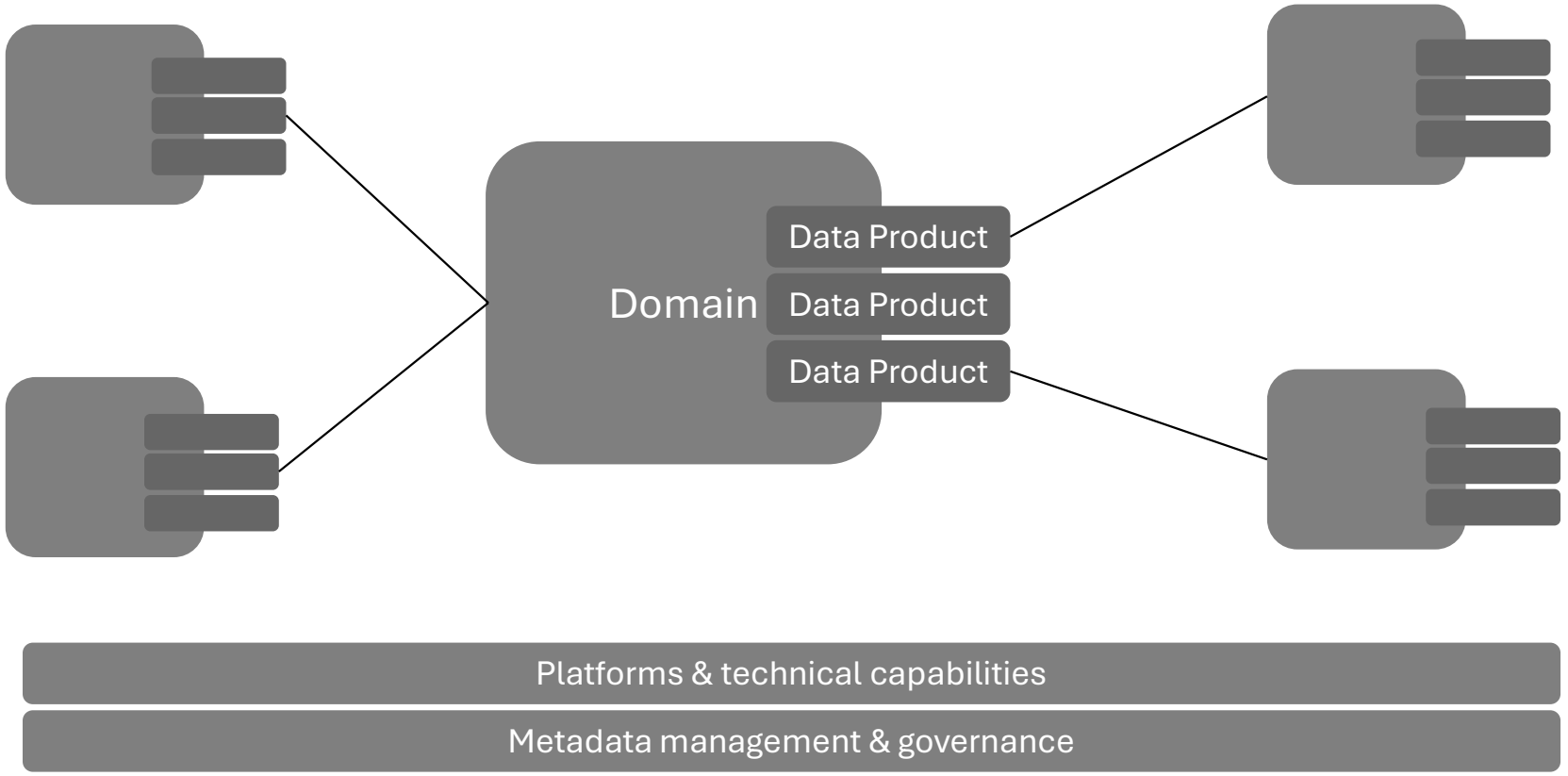
- General idea
- Four pillars of Data Mesh according to Dehghani
- Domains and domain teams
- Data products
- The interoperability challenge

Data Mesh: background

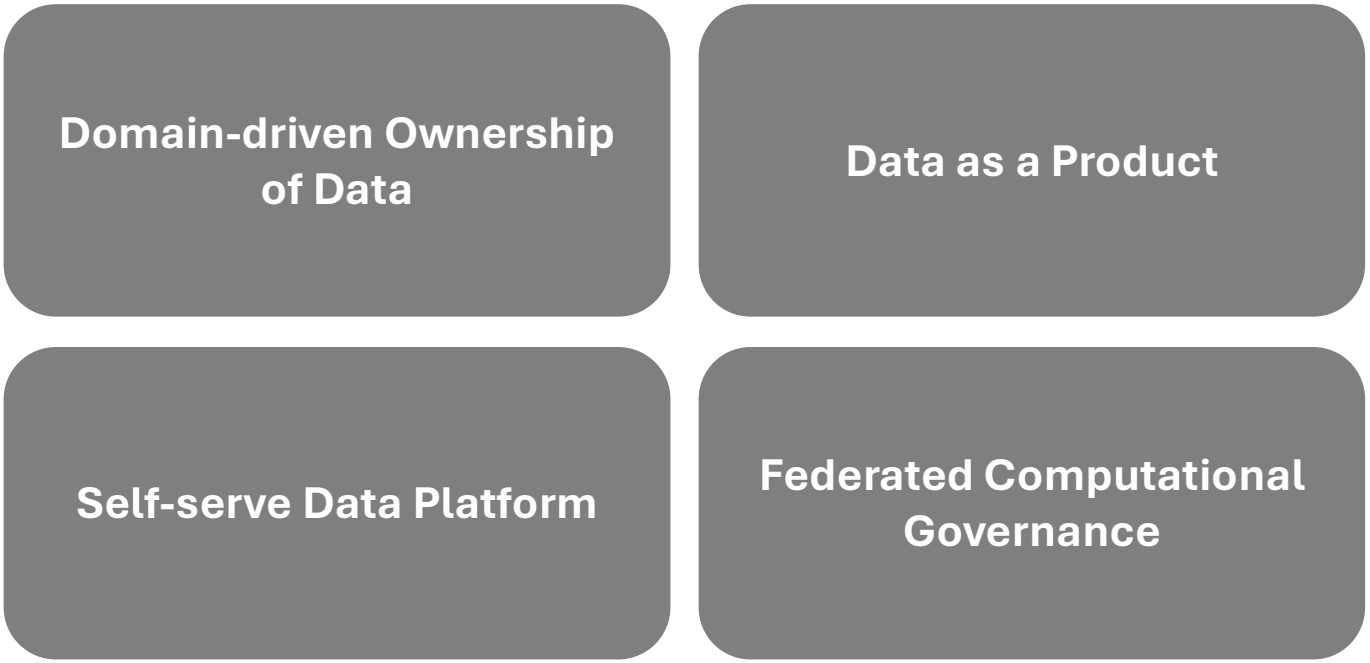
- A framework developed by **Zhamak Dehghani**
- **Domain-oriented**, decentralized model
- Domain teams are responsible for all the data within a domain & distributing it outside the domain
- All data distribution & consumption done through **data products**
- Domains and their data products form a **mesh**
- The Data Mesh is **supported by shared capabilities**: technology platforms + governance model



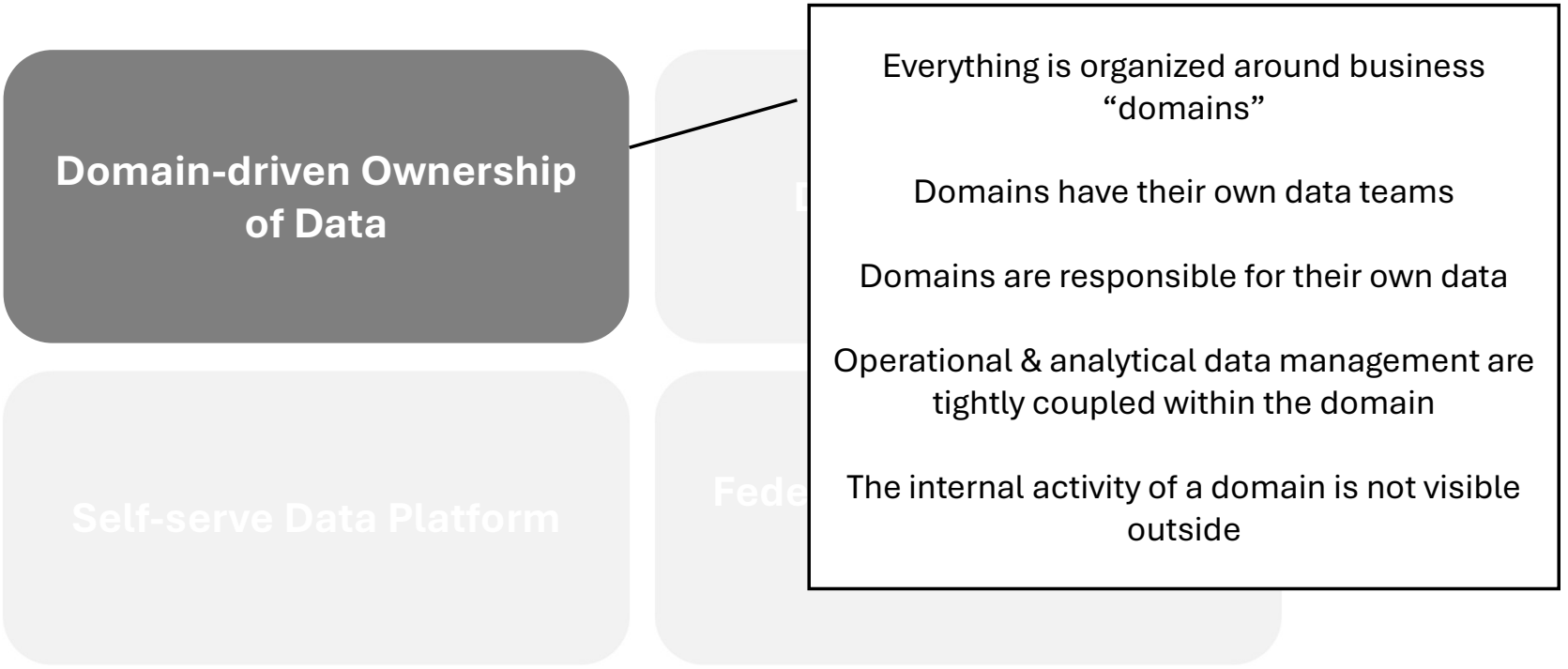
Data Mesh in a nutshell



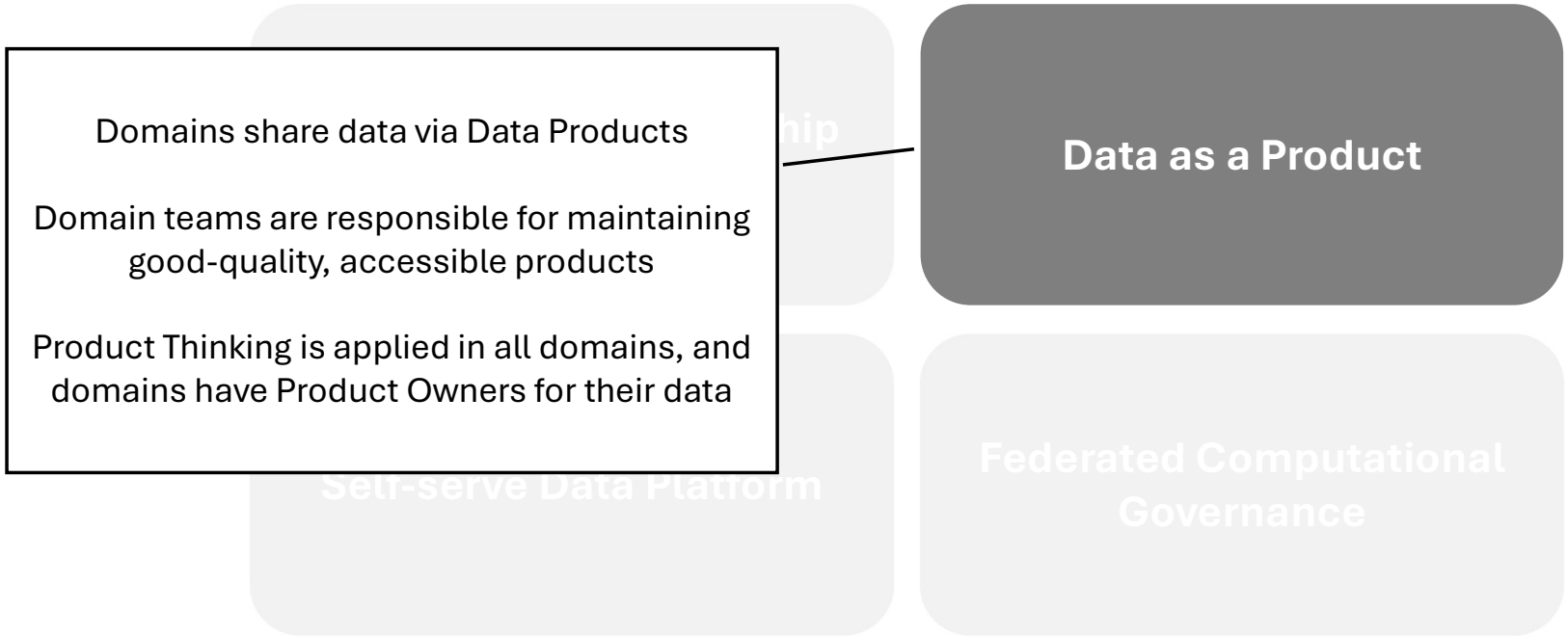
Four principles of Data Mesh



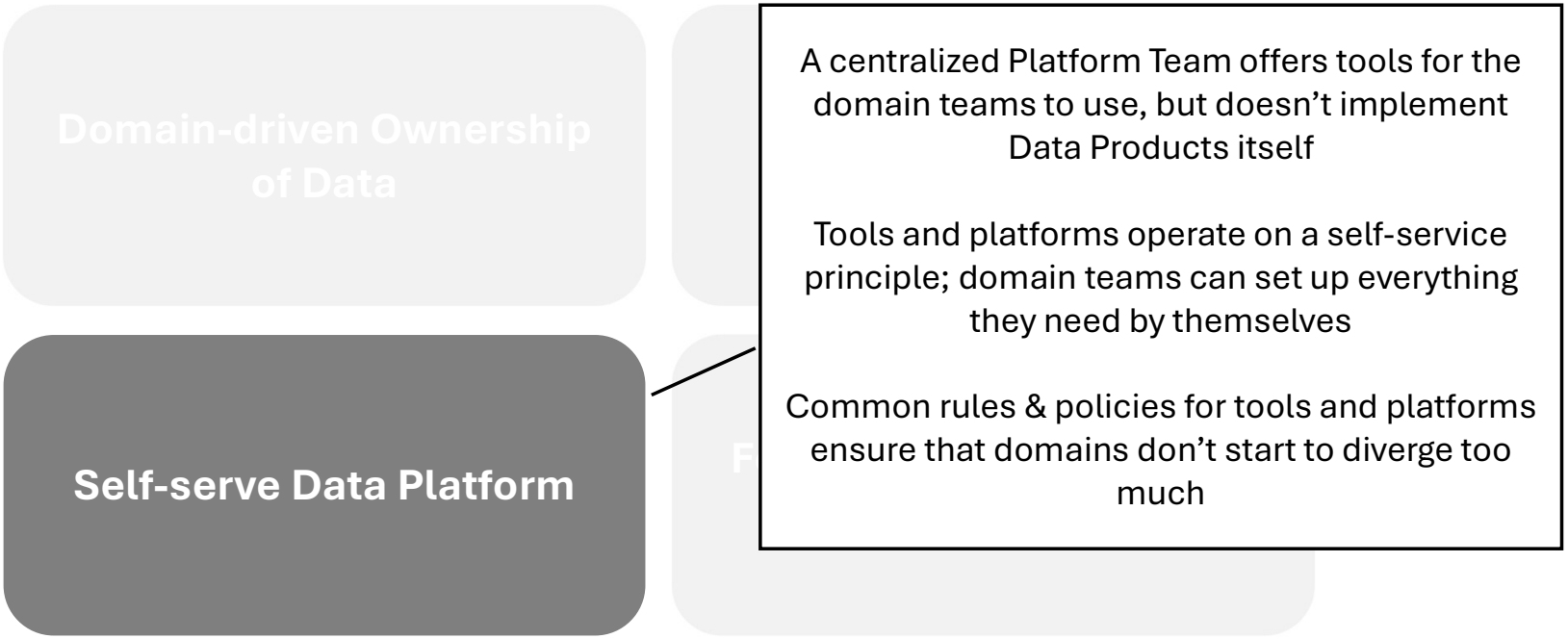
Four principles of Data Mesh



Four principles of Data Mesh



Four principles of Data Mesh

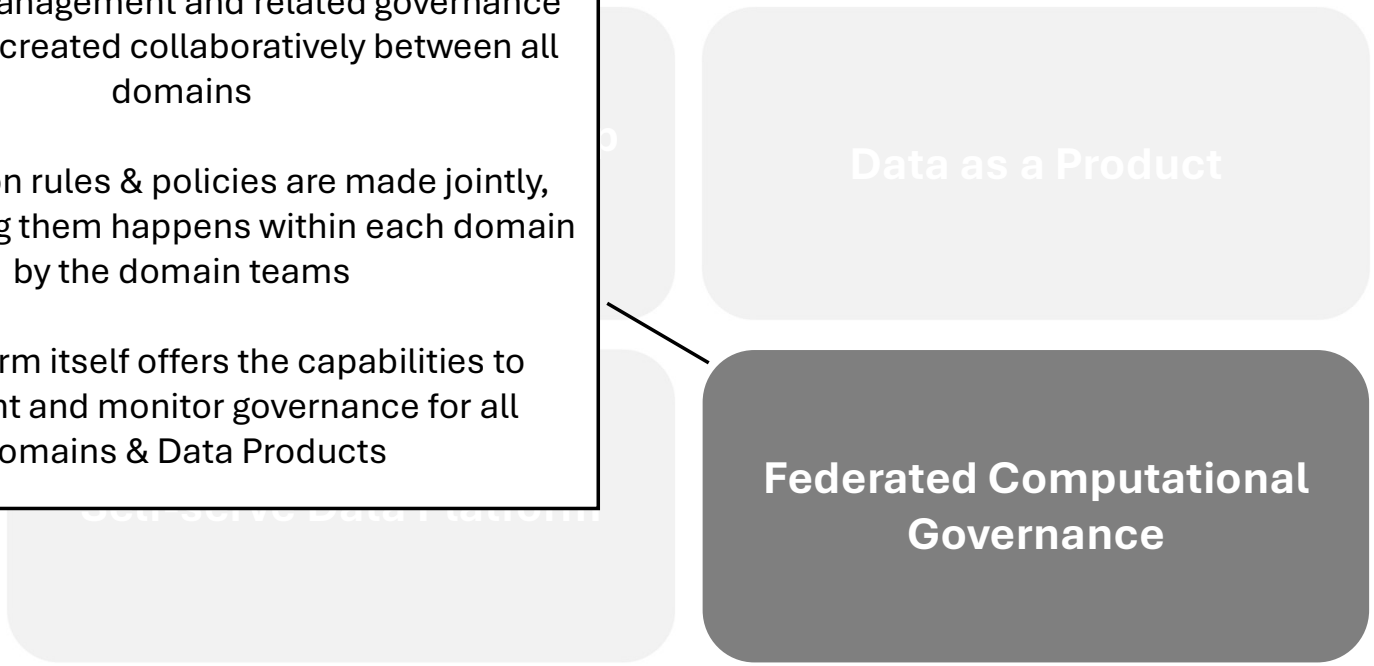


Four principles of Data Mesh

Metadata management and related governance policies are created collaboratively between all domains

Decisions on rules & policies are made jointly, implementing them happens within each domain by the domain teams

The platform itself offers the capabilities to implement and monitor governance for all domains & Data Products



Data Mesh in reality

- 100% pure Data Mesh implementation is **rare**
- Significant technology effort
- **Massive** organizational effort
- Benefits from **scale** - example organizations: PayPal, Spotify, Roche...

HOWEVER...

- Real-life benefits in **shifting data responsibilities closer to value-creation**
- **"Data as a Product" –thinking** valuable for everyone
- Separating **Platform teams** (building capabilities) and **Domain teams** (building solutions) clarifies roles and boundaries

➢ ***If you want to do decentralized data products, you will end up scaling to Data Mesh!***

What is a “domain”?

- A problem space within business reality
- Based on Domain-Driven Design (DDD) thinking (Eric Evans)
- Domains can be organized along different lines:
  - Business capability
  - Value chain
  - Core process
  - Organizational unit
- Domains should **never** be based on technology!

Example: business capability domains

Piethein Strengholt
https://towardsdatascience.com/data-domains-where-do-i-start-a6d52fef95d1

Cross-domain data sharing

Domains share data via Data Products

Domain teams are responsible for maintaining good-quality, accessible products

Product Thinking is applied in all domains, and domains have Product Owners for their data

Data as a Product

Product thinking

- Agile methods bring extra focus on ***products***
- Products are meant to ***deliver value for known users***
- Products are designed for ***known use cases***
- Products have ***clear boundaries***
- Tech and platforms are ***secondary*** enablers; products are the ***primary*** output
- Products have ***Product Owners***
- Products have a ***description, product management processes,*** and ***lifecycle***

"A product is a vehicle to deliver value. It has a clear boundary, known stakeholders, well-defined users or customers. A product could be a service, a physical product, or something more abstract."

Scrum Guide
(Schwaber & Sutherland)

WE HAVE NOT BEEN DOING THIS FOR DATA & ANALYTICS

From projects to products

Demand-driven value delivery

Build for identified use cases

Know your users

Explicit UX design

Product ownership

Governance for products

Lifecycle management

Prioritization by the owner

Loosely coupled architecture

Clear boundaries

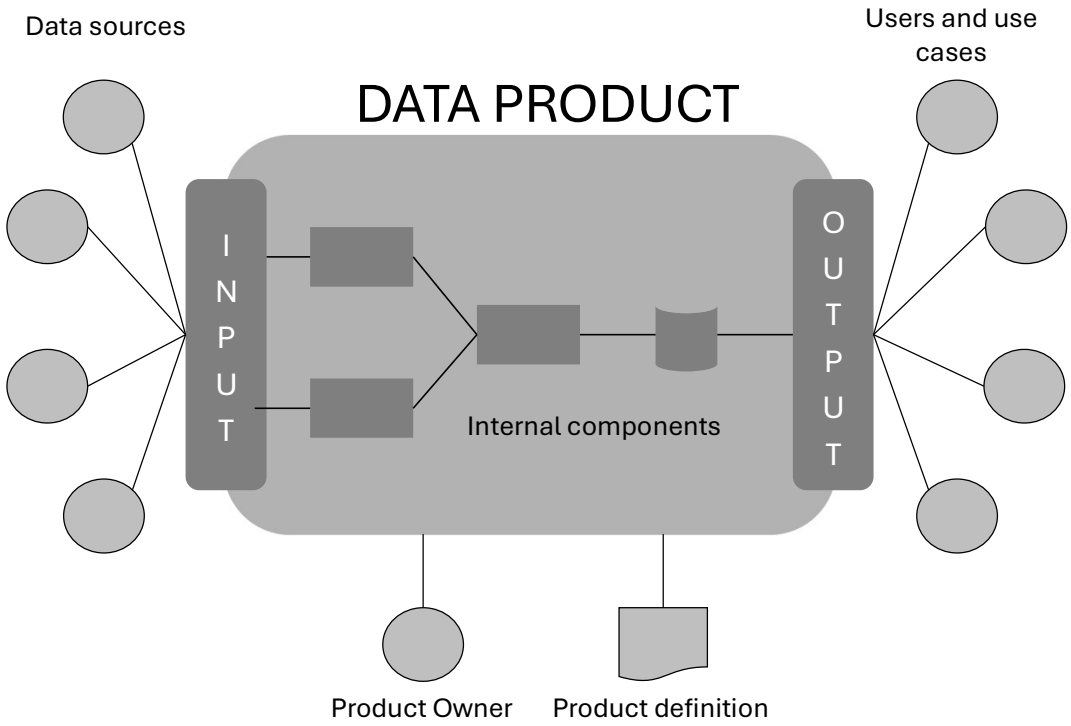
Shared platforms

Independent development

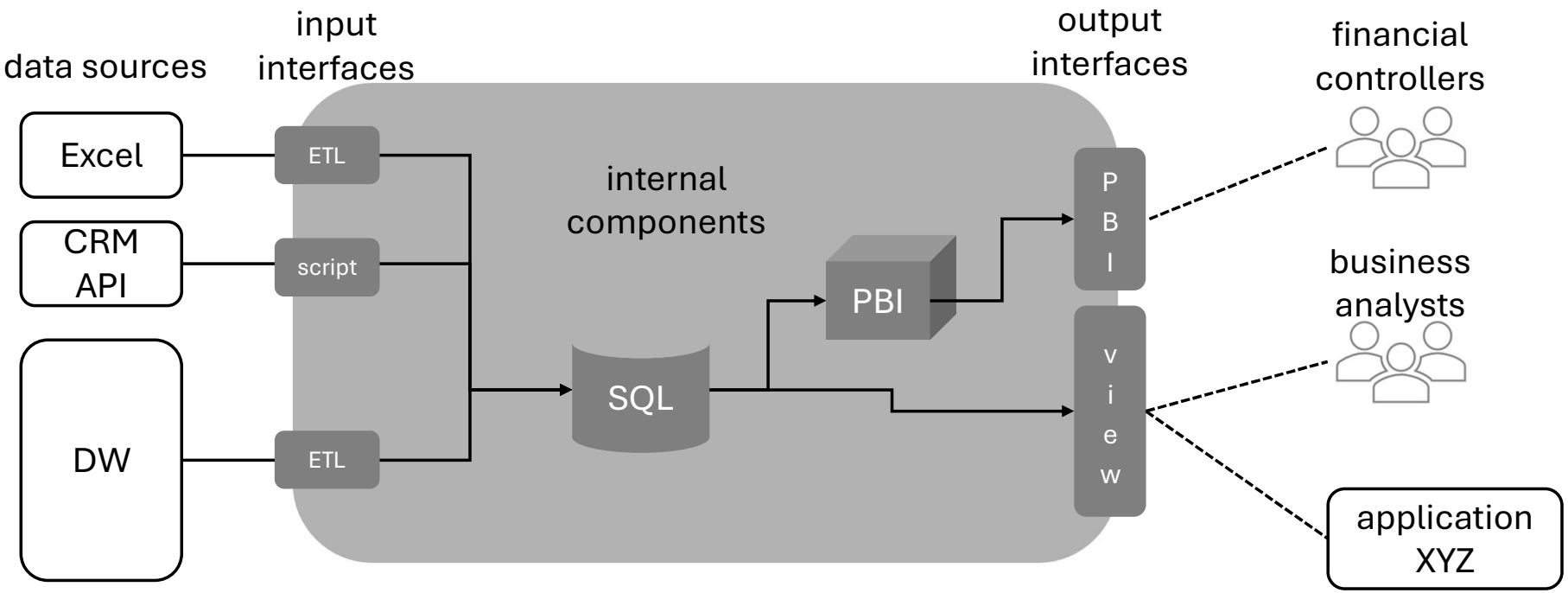
What is a “Data Product”? My definition!

A “packaged” set of data and components, that has:

- Clear boundaries
- One owner
- A lifecycle
- Identified users and benefits
- Input and output interfaces
- Understandable definition
- “Data and code”!



A Data Product under the hood



Types of Data Products

Source-aligned

- Offers unmodified “raw” data from an operational source system
- Safe way to utilize source data without affecting operational use
- Users: other systems, other data products

Aggregate

- Re-models and integrates data from one or more source systems or source-aligned data products
- Aims at maximum reusability, acts as a source for other data products – modeled according to domain entities
- Users: consumer-aligned data products

Consumer-aligned

- Specifically aimed at and optimized for a known use case
- Limited, “tight” scope, maximum use case fit for maximum business impact
- Users: humans, various applications

Data

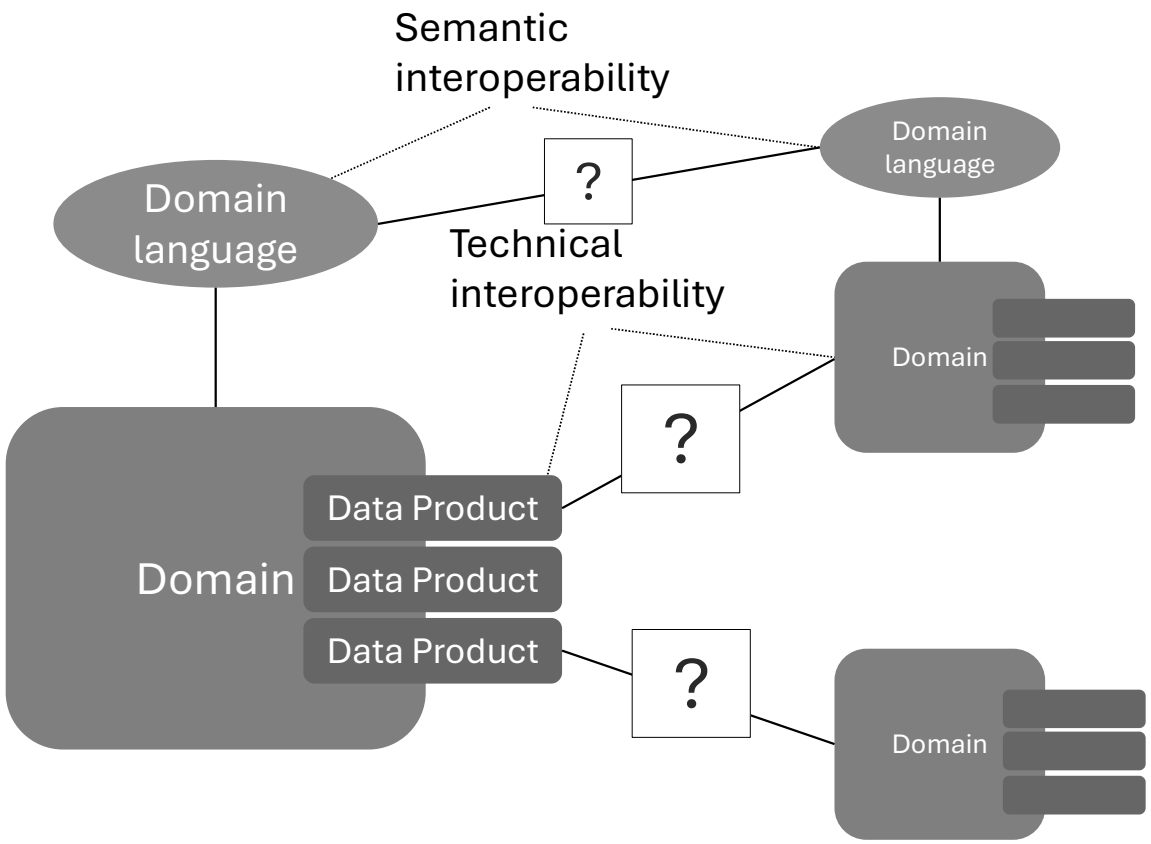
”Productized” data

The interoperability challenge

What Data Products are available?

How can they be accessed?

What does the data mean?



Conceptual models for cross-domain understanding

Conceptual models for cross-domain understanding

- Basics of conceptual modeling: entities, relationships, and attributes
- How to identify the real business objects
- Building definitions and glossaries

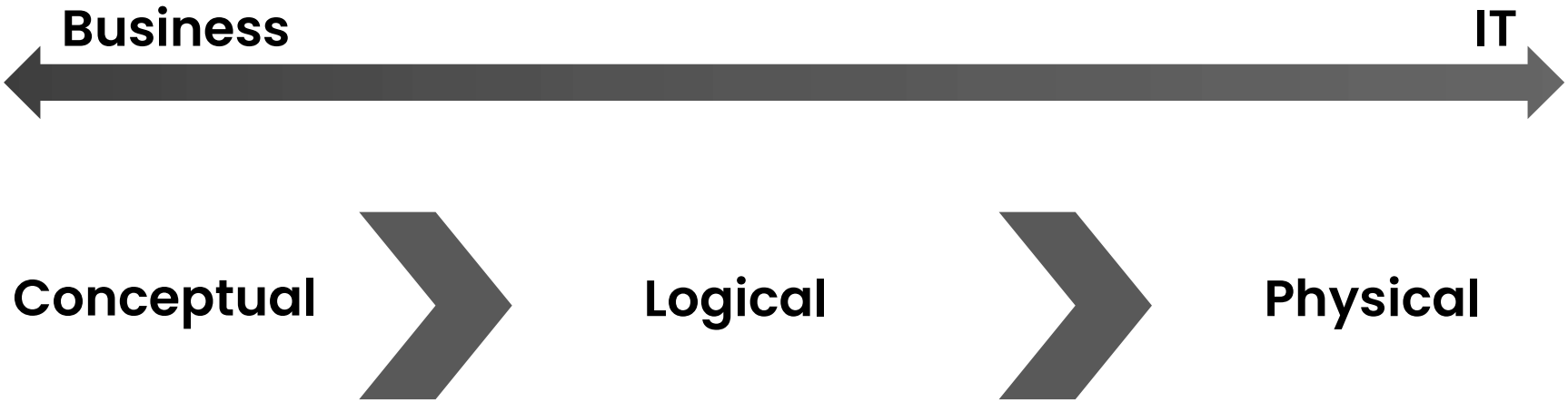
What is data modeling?

Data modeling is a way to describe reality in a structured format.

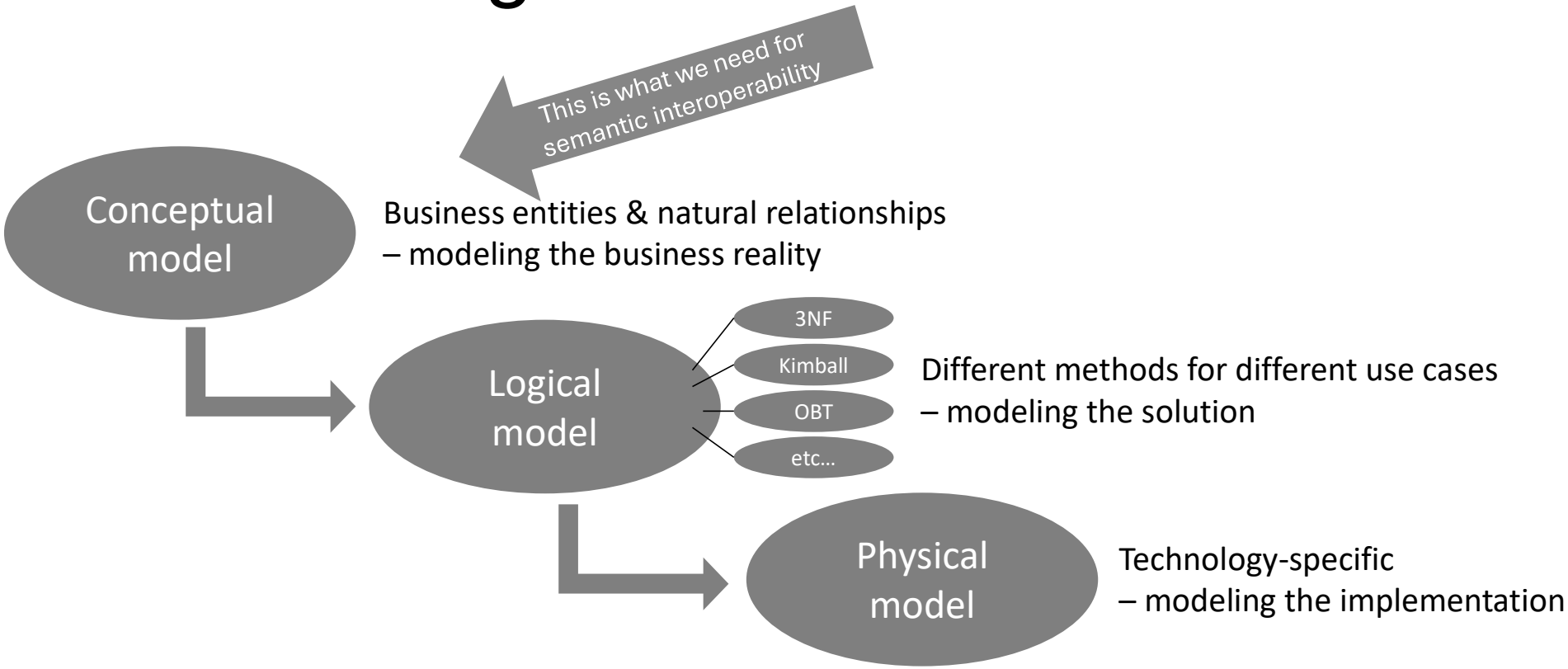
“A data model is a wayfinding tool for both business and IT professionals, which uses a set of symbols and text to precisely explain a subset of real information to improve communication within the organization and thereby lead to a more flexible and stable application environment.”
- Steve Hoberman

“A data model is a structured representation that organizes and standardizes data to enable and guide human and machine behavior, inform decision-making, and facilitate actions.”
- Joe Reis

Three levels of data modeling



Data modeling levels & methods



What is a Conceptual Data Model?

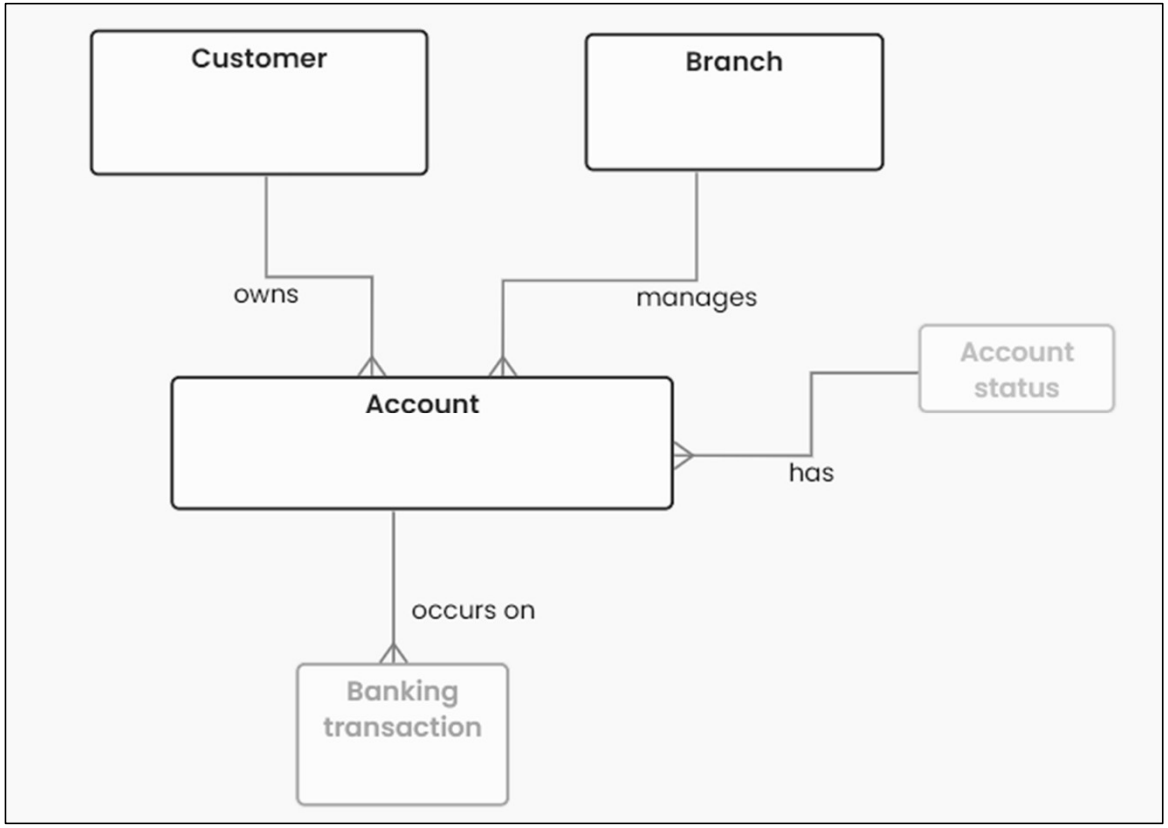
- A business-language description of some part of business reality
- A narrative of “what’s going on” represented by a diagram
- Depicts *things the business needs to know about* and *the associations between those things*
  - “Entities” and “relationships” → “Entity-Relationship Diagram”, ERD
  - Note! Not technical components, but *things!*
- Is NOT about systems or reports, but the *reality* behind the system or report

Therefore, a conceptual model is…

"a description of the business in terms of the things it needs to know about"

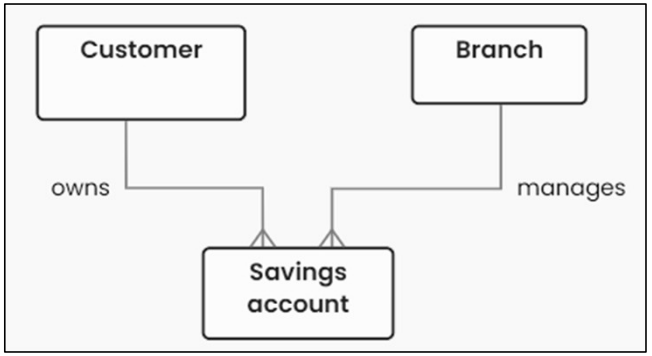
- Alec Sharp

Conceptual model: a simple example



Entities

- An entity is a noun in a sentence
- Narrative: *“A customer owns a savings account which is managed by a branch”*
  - Entities: “Customer”, “Savings account”, “Branch”
- Good practice: always use singular nouns (“Customer” instead of “Customers”)
- Listen to the business expert and write down all nouns → entity candidates

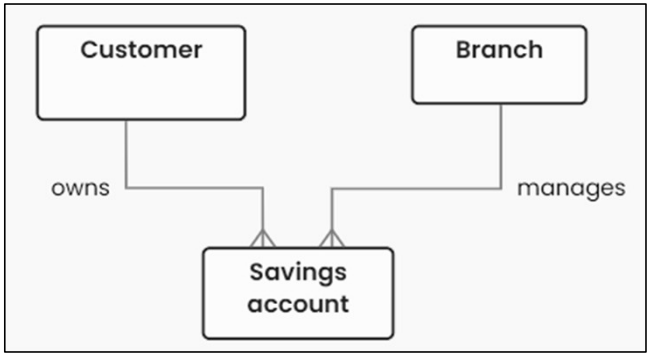


Attributes

- An attribute is information about an entity
  - “Customer” = entity
  - “Customer name” = attribute of the “Customer” entity
- Attributes can’t exist on their own!
- In conceptual models, attributes are often not included
- Sometimes useful to list a few important attributes for the entities already in early stages
  - E.g. “Customer Lifetime Value” is an attribute of “Customer”

Relationships

- An association between two entities that we want to maintain information about
- Based on *real-life* associations
  - It's possible that a relationship exists in real life, but it's not (yet) recorded in any system!
- Relationships are named with verbs
  - Narrative: "*A customer owns a savings account which is managed by a branch*"
  - Find the verb describing the real-life association – maintain the narrative!



Identifying entities: common pitfalls

Most data modeling failures are caused by poorly chosen entities!

- Technical objects instead of real-life entities
  - Avoid applications, reports, tables, components, or system messages – find the business entities behind these
- Attributes as entities
  - “Customer name” is not an entity but an *attribute* of the “Customer” entity (more about attributes later)
- Non-business language
  - Don’t think about system names or official naming – use whatever the business uses!
- Non-countable abstractions
  - Find the countable things: not “Staff” but “Employee”, not “Bank statement” but “Financial transaction”
- Too much generalization too soon
  - “Object instance type” is not a business entity!

Entity definitions & Glossary

- Every entity should have an understandable, business-language definition
- Answers the question: “what do we mean by <entityname>?”
- Entity definitions form a Glossary
- Often domain-specific: “Account” for Sales is different from “Account” for Finance

➢Need to be written down so that they’re available for everyone

➢Domain Glossaries capture the domain’s language

Example: entity definitions

Debit Card

GOOD

- “Payment card issued by a financial service provider that enables the cardholder to access funds in a demand deposit account.”

Credit Card

GOOD

- “Card issued by a financial service provider that enables the cardholder to borrow funds.”

Savings Association

NOT SO GOOD?

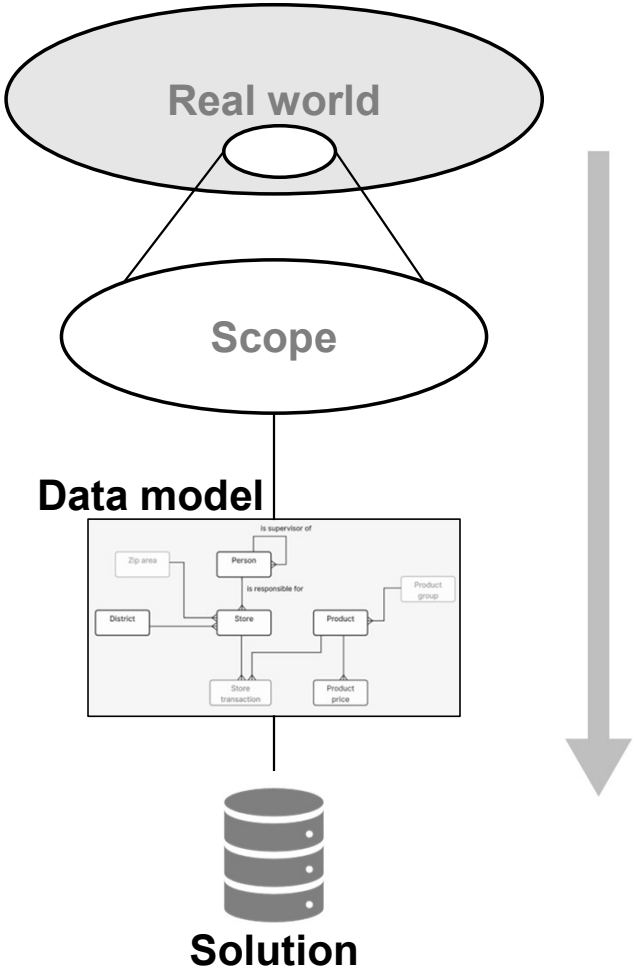
- “Depository institution that is (a) any federal savings bank or association chartered under section 1464 of the Federal Deposit Insurance Act; (b) any state chartered building and loan association, savings and loan association, or homestead association; or (c) any cooperative bank (other than a cooperative bank which is a state bank as defined in subsection (a)(2)) of the Federal Deposit Insurance Act, which is organized and operating according to the laws of the State (as defined in subsection (a)(3)) in which it is chartered or organized; and (c) any corporation (other than a bank) that the board of directors and the comptroller of the currency jointly determine to be operating in substantially the same manner as such a depository institution.”

The scope of a conceptual model

- What are we *really* modeling?
- The scope of a conceptual model must always be a **slice of reality** – the part of the real world *behind* our data needs
- Boundaries often defined by:
  - Business process area
  - Current business problem
  - Departmental responsibility
  - What is covered by another technical solution…
- Important: we are not modeling a solution, we are modeling reality

➢ **"Slice of reality" can be a Domain!**

➢ **"Solution" can be a Data Product!**



Hands-on exercise: modeling a domain

Hands-on exercise: modeling a domain

- Domain boundaries
- Identifying entities within the domain
- Definitions and Domain Glossary

Introducing “Storefront”

Storefront is a mid-sized online retailer specialising in consumer electronics. The company sells directly to consumers through its website and mobile app, handling everything from product discovery to doorstep delivery.

EXERCISE

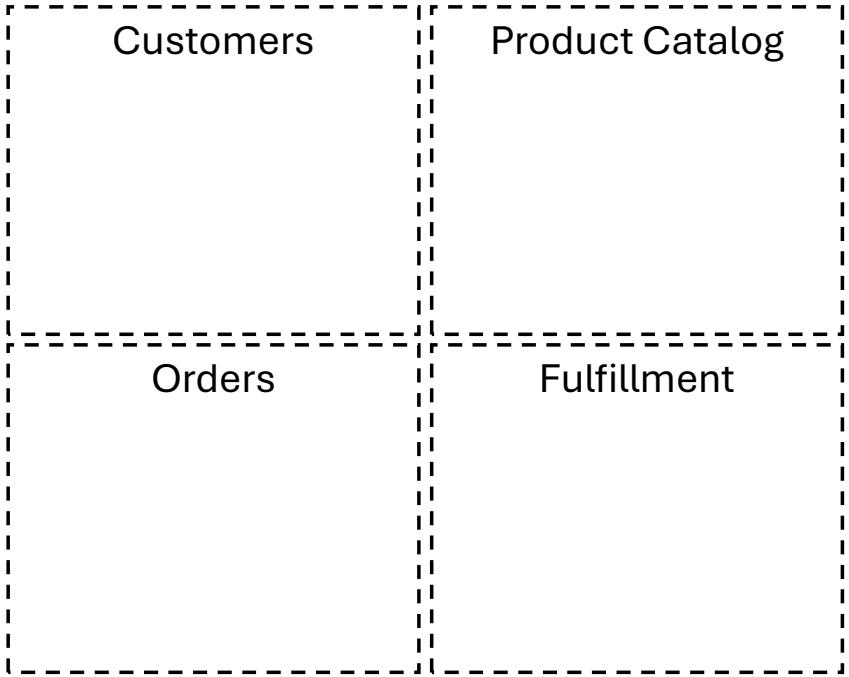
Storefront's domain structure

Let's come up with some example domains for Storefront

Choose according to e.g.

- Business capability
- Value chain
- Core process
- Organizational unit

There is no single right answer – we'll continue with these four:



storefront

You are now responsible for “Orders”

The Orders domain team owns everything that happens between a customer deciding to buy something and the payment being settled. This is the commercial heartbeat of the business.

Your mission: Capture, track, and complete every customer purchase from the moment of intent to the moment of payment.

Your responsibilities

- Accepting and recording customer orders
- Applying pricing rules, promotions, and discount codes
- Processing and confirming payments
- Managing order status through the purchase lifecycle
- Handling cancellations and returns

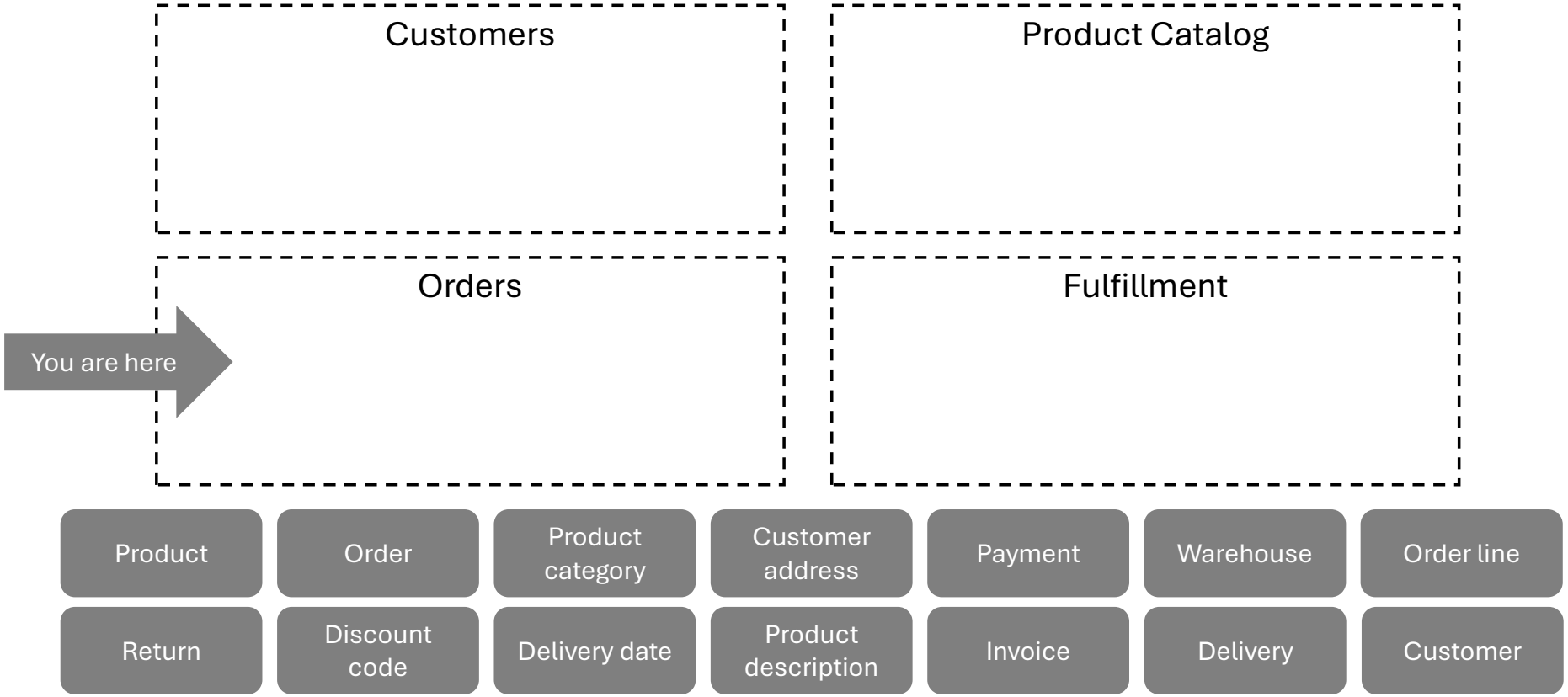
Business questions

- How many orders were placed today, and what is their total value?
- Which orders are awaiting payment confirmation?
- What is the return rate for orders placed in the last 30 days?

storefront

EXERCISE

Domain boundaries – who owns what?



EXERCISE

Conceptual model for the Orders domain

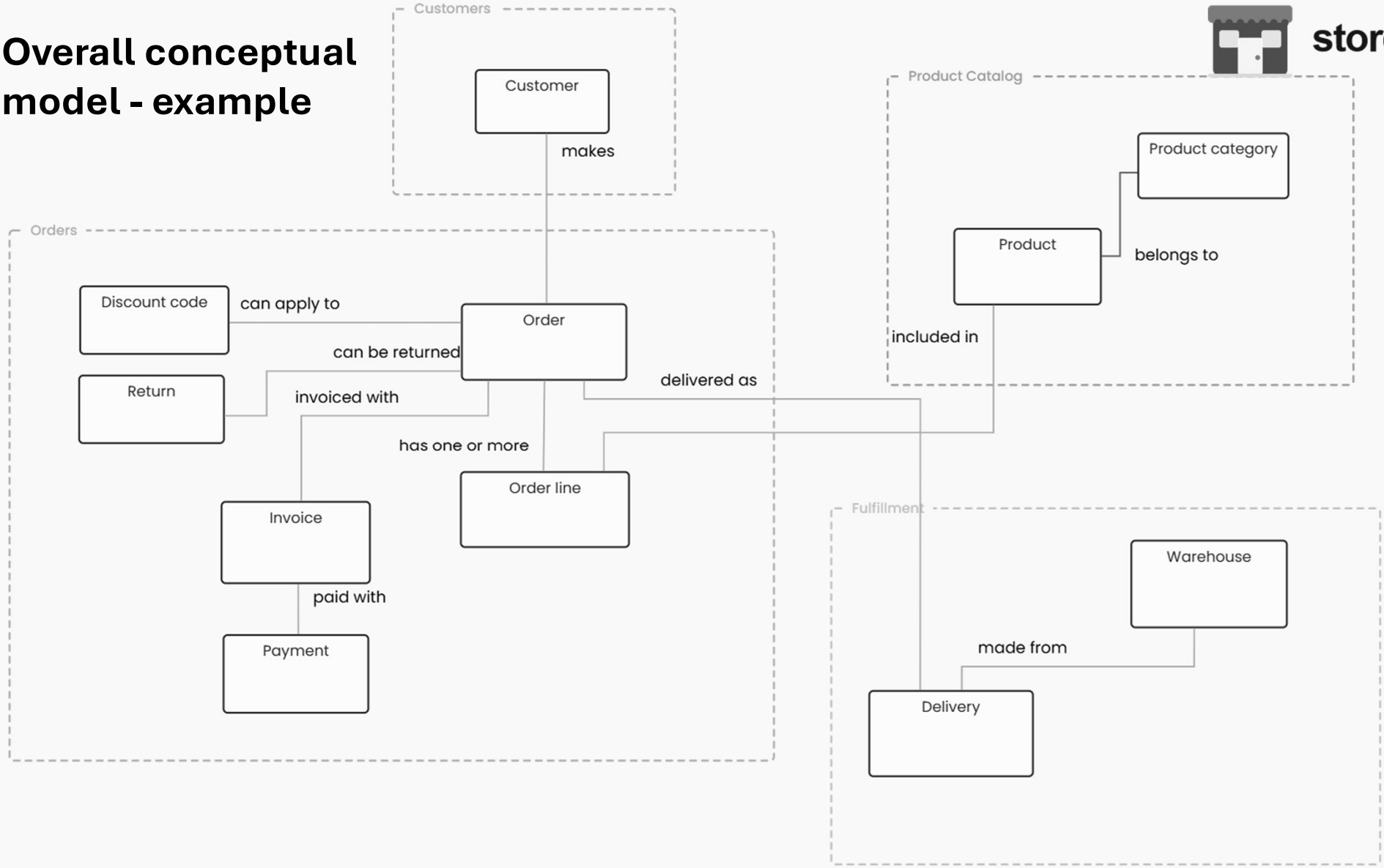
Orders

Let’s build a model!

- Include all YOUR data
- What is an ENTITY, what is an ATTRIBUTE?
- Add named RELATIONSHIPS
- Identify where you need to REFERENCE some other domain’s data

Overall conceptual model - example

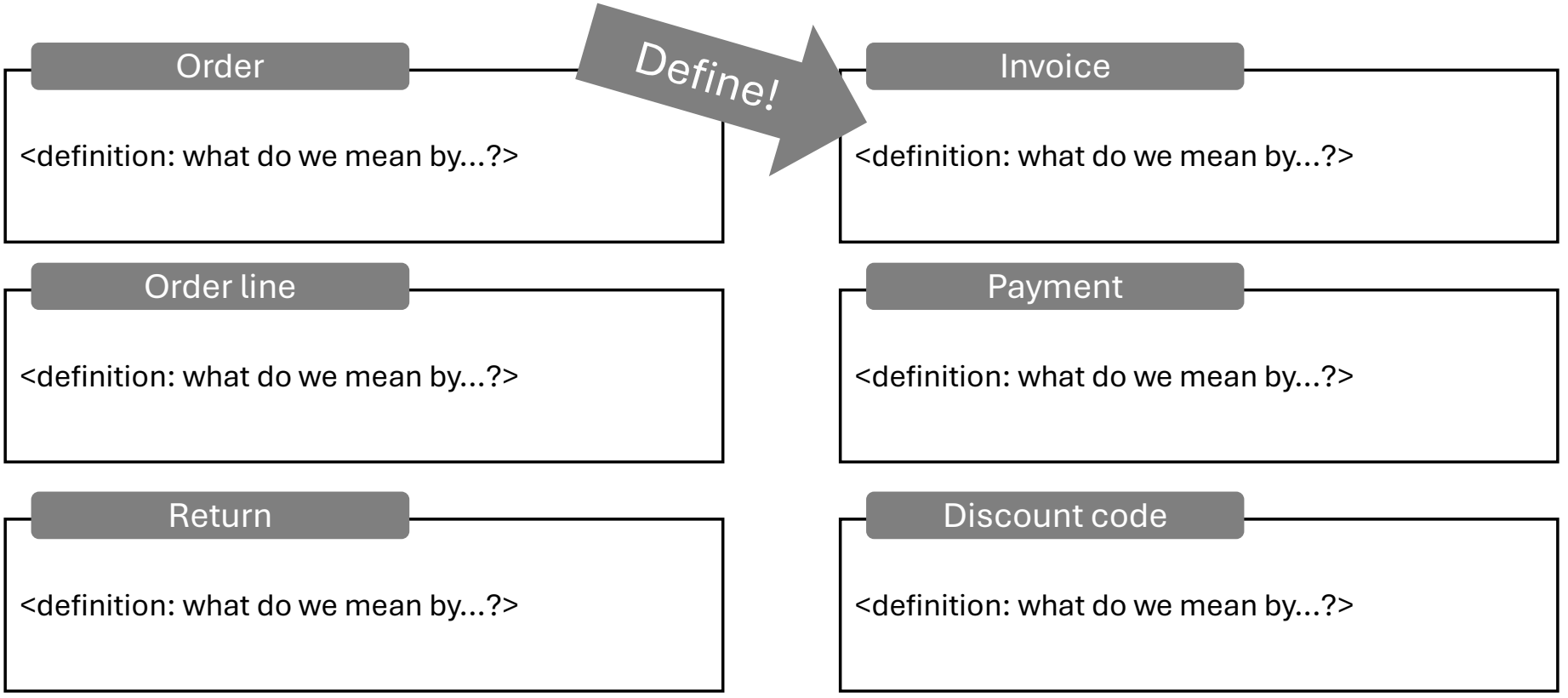
storefront



storefront

EXERCISE

Creating a Domain Glossary for Orders



storefront

Domain Glossary for Orders: examples

Order

A customer's confirmed intent to purchase one or more products from Storefront, creating a commercial commitment that the business is obligated to fulfill.

Order line

A single product and quantity within an order, capturing what was bought and at what price at the moment of purchase. Each order contains at least one order line; the order line is the atomic unit of what was sold.

Return

A customer's request to reverse all or part of a fulfilled order, triggering a commercial process that typically results in a refund or exchange.

Invoice

The formal record of the financial obligation arising from an order, stating what was purchased, at what agreed price, and what payment is due.

Payment

A record of a customer’s settlement of an invoice, whether it is settled fully or partially.

Discount code

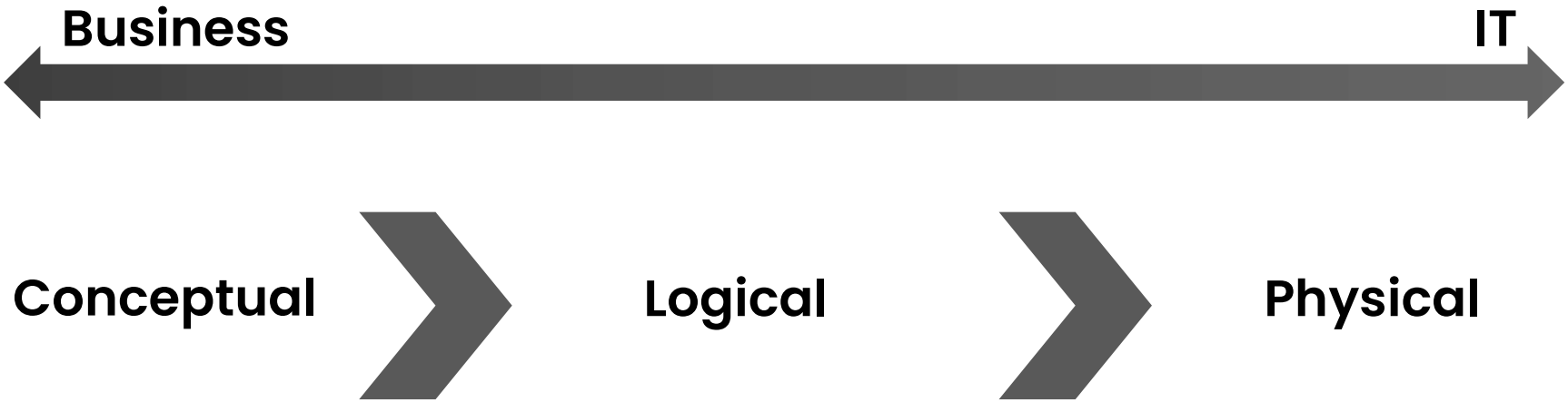
A code applied by a customer at checkout that reduces the payable value of an order, as authorised by a promotion.

Data modeling as part of data product design

Data modeling as part of data product design

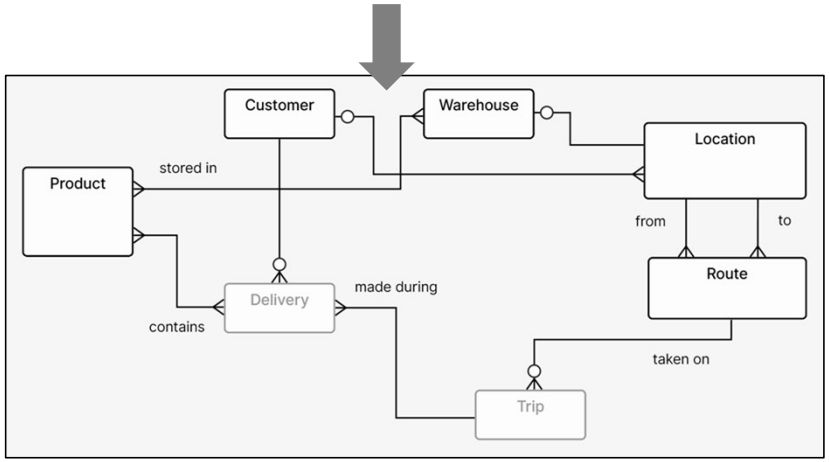
- Understanding product scope as part of the domain model
- Logical model as product-level design & documentation
- Deriving logical models from conceptual model
- Maintaining links with the domain model

Recap: three levels of data modeling



Capturing context with data models

Business entities and relationships of the domain



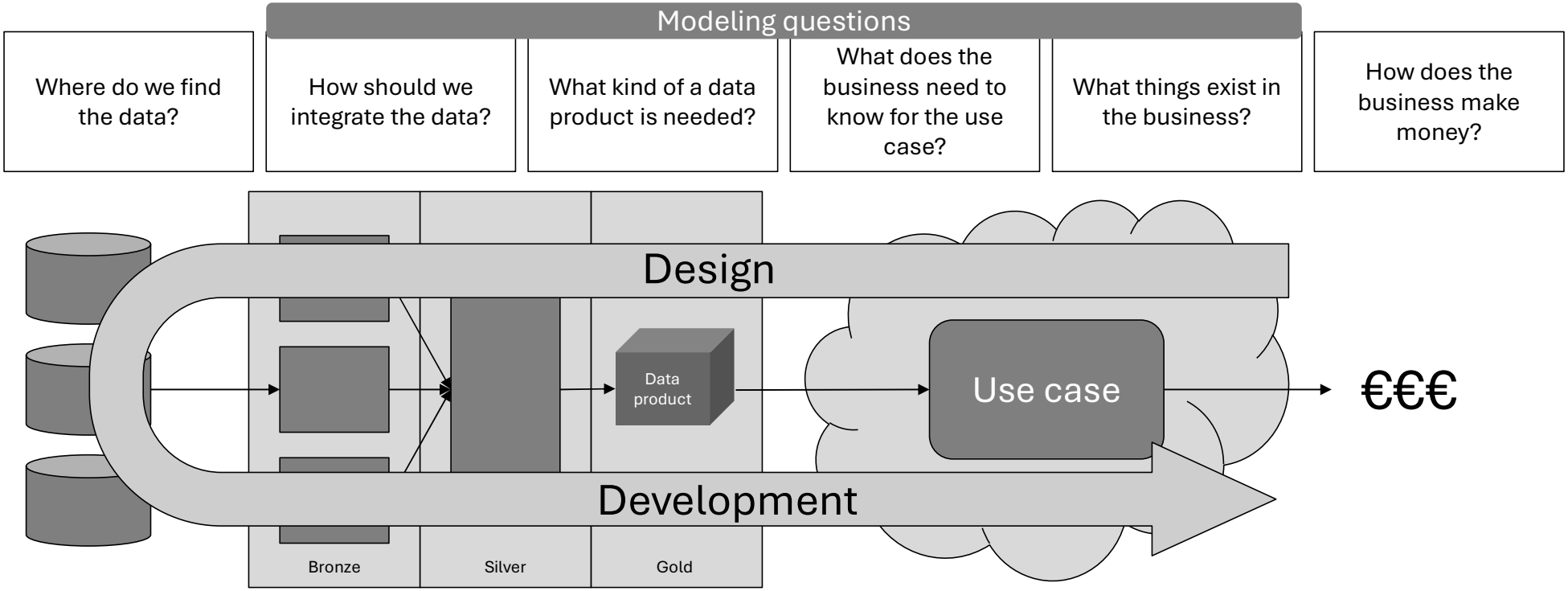
Derived design of a Data Product

Conceptual model: semantics

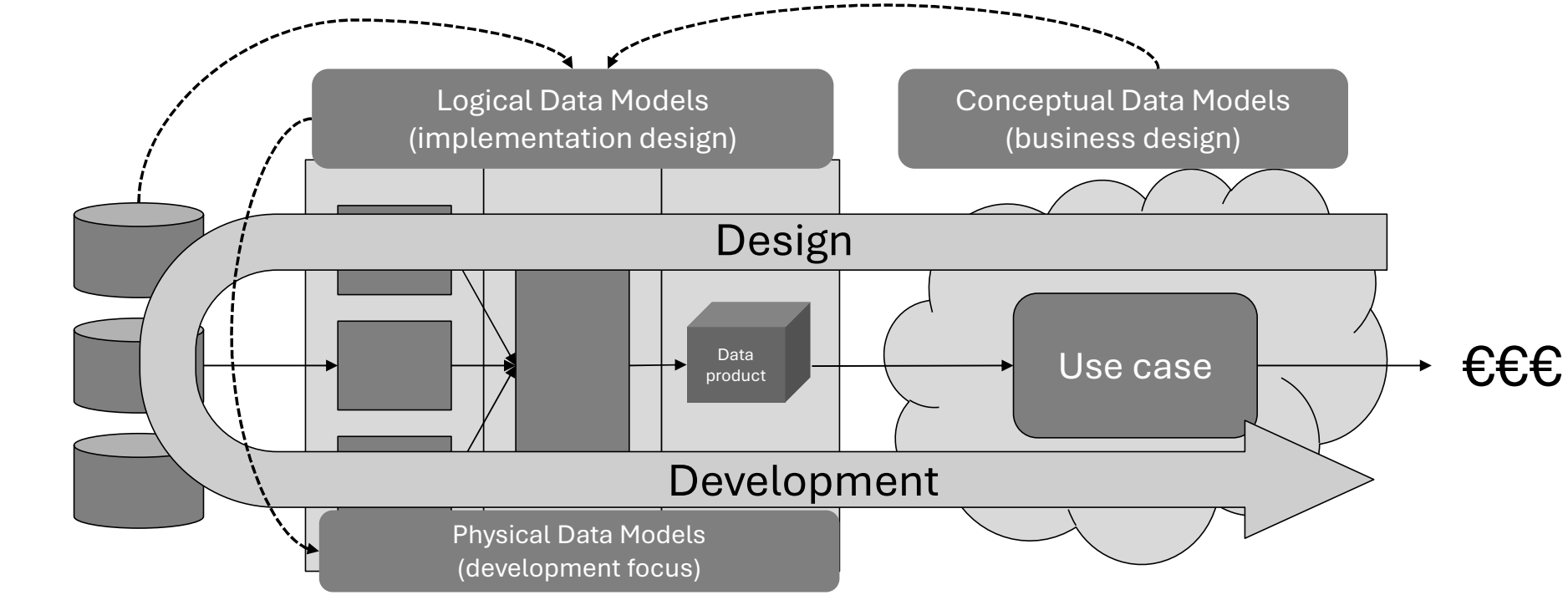
Often the domain-level model is enough, sometimes a product-level conceptual model is needed!

Logical model: use case-specific structure

Business-driven data design & development



How modeling happens during the process



From conceptual to logical modeling

CONCEPTUAL: FOCUS ON BUSINESS OBJECTS

- Understand the **DOMAIN**
- Understand what **THINGS** people need data **ABOUT**
- Understand what the **THINGS** are
- Understand how the **THINGS** are related in the real world

Result:

- A **RE-USABLE** data model
- A model that is **true regardless of implementation**
- A model that **stays true** (unless the business itself changes)
- A model that **describes reality**

LOGICAL: FOCUS ON TECHNICAL OBJECTS

- Understand the **USE CASE**
- Design a structure for a suitable **DATA PRODUCT**
- Understand what information is available about the business objects
- Design how that information fits the chosen structure

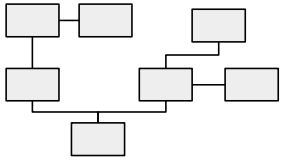
Result:

- A **DATA PRODUCT**-specific data model
- A model that is **only true** for a **specific implementation**
- A model that is **directly usable** as implementation blueprint
- A model that **describes a single solution**

Ask: *"What is a suitable data structure for my product, given a) the existing conceptual model and b) the needs of my use case?"*

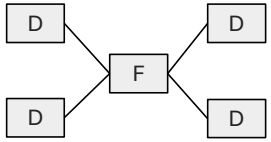
Logical model: the “shape” of data

Some examples:



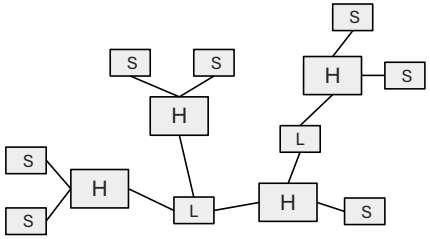
3NF
“normalized”

- Each piece of information only exists once
- Many tables and joins
- Reusable
- Easy to update
- Complex queries



Star Schema
“dimensional”

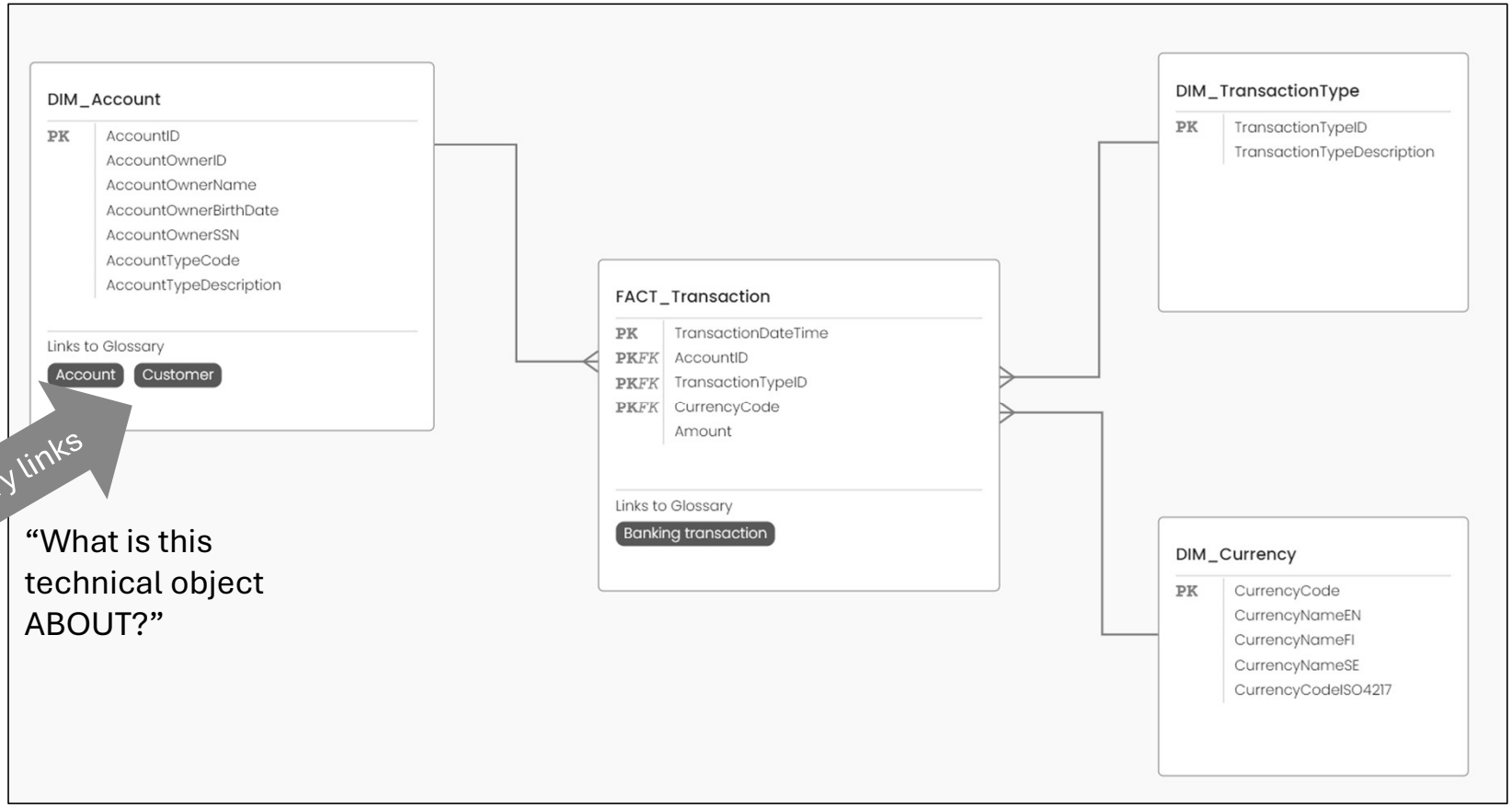
- “Facts” and “Dimensions”
- Few tables
- Easy to query
- Sometimes complex to update
- Limited reusability
- Good for reporting



Data Vault
“DV” and “DV2.0”

- “Hubs”, “Links”, and “Satellites”
- Very large number of tables and joins
- Easy to update (insert-only)
- Highly complex queries
- Self-historizing
- Works well with automation

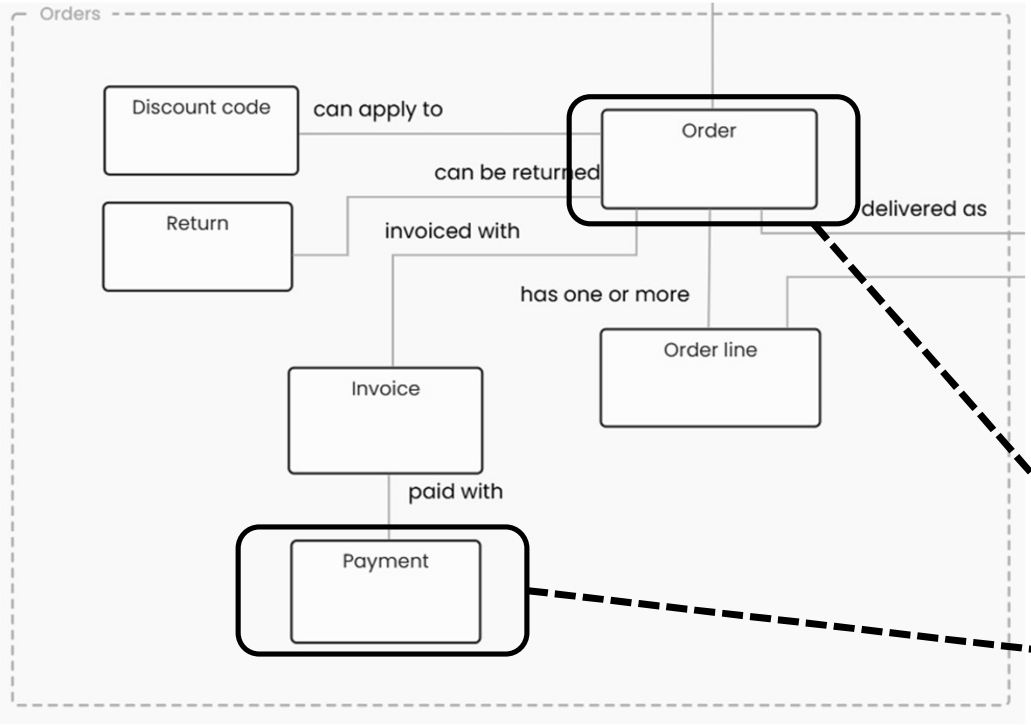
Logical model example: star schema



Note: glossary links

"What is this technical object ABOUT?"

Connecting the Product and its context

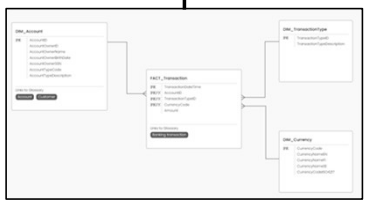


Domain conceptual model

Business questions

- Which orders are awaiting payment confirmation?

Data Product



Data Product logical model

Ensuring semantic interoperability at the domain boundary

Ensuring semantic interoperability at the domain boundary

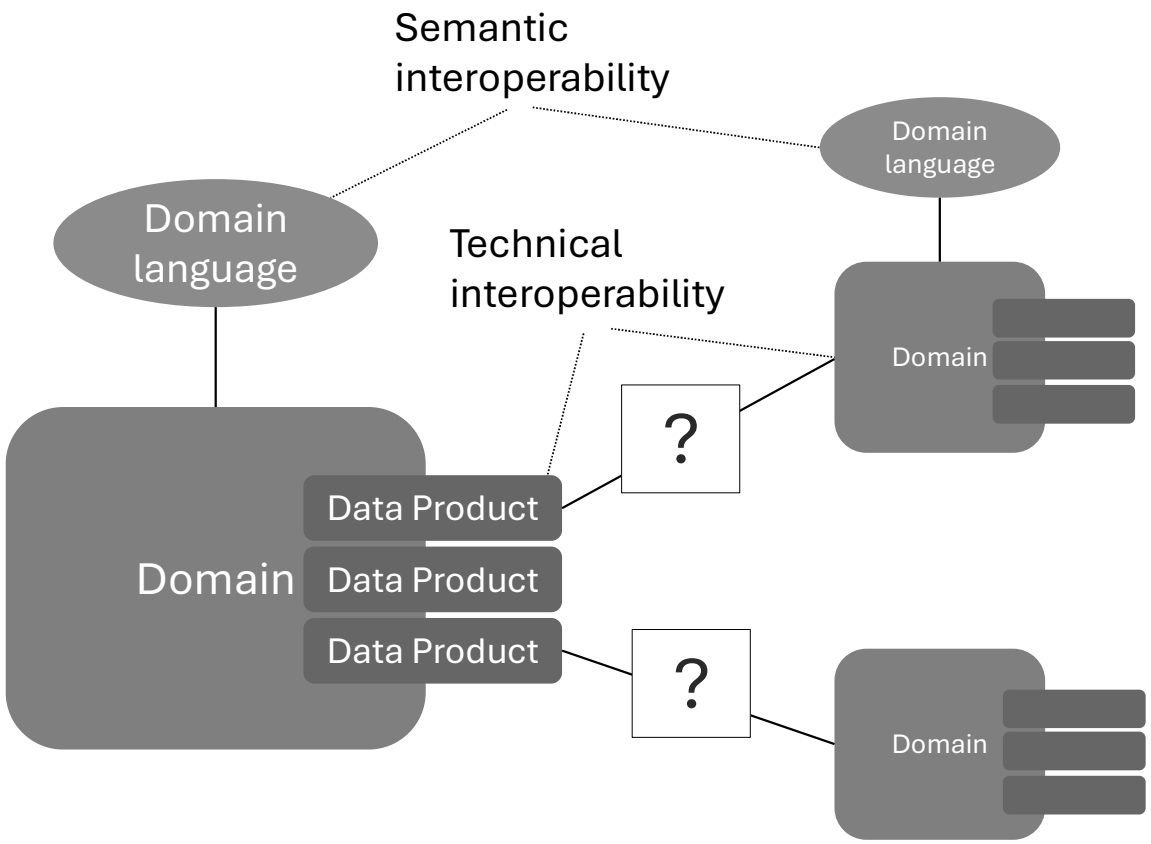
- Exposing metadata from domains and data products
- Data contract basics
- Domain glossaries vs. shared enterprise glossaries
- Dealing with polysemes

Recap: the interoperability challenge

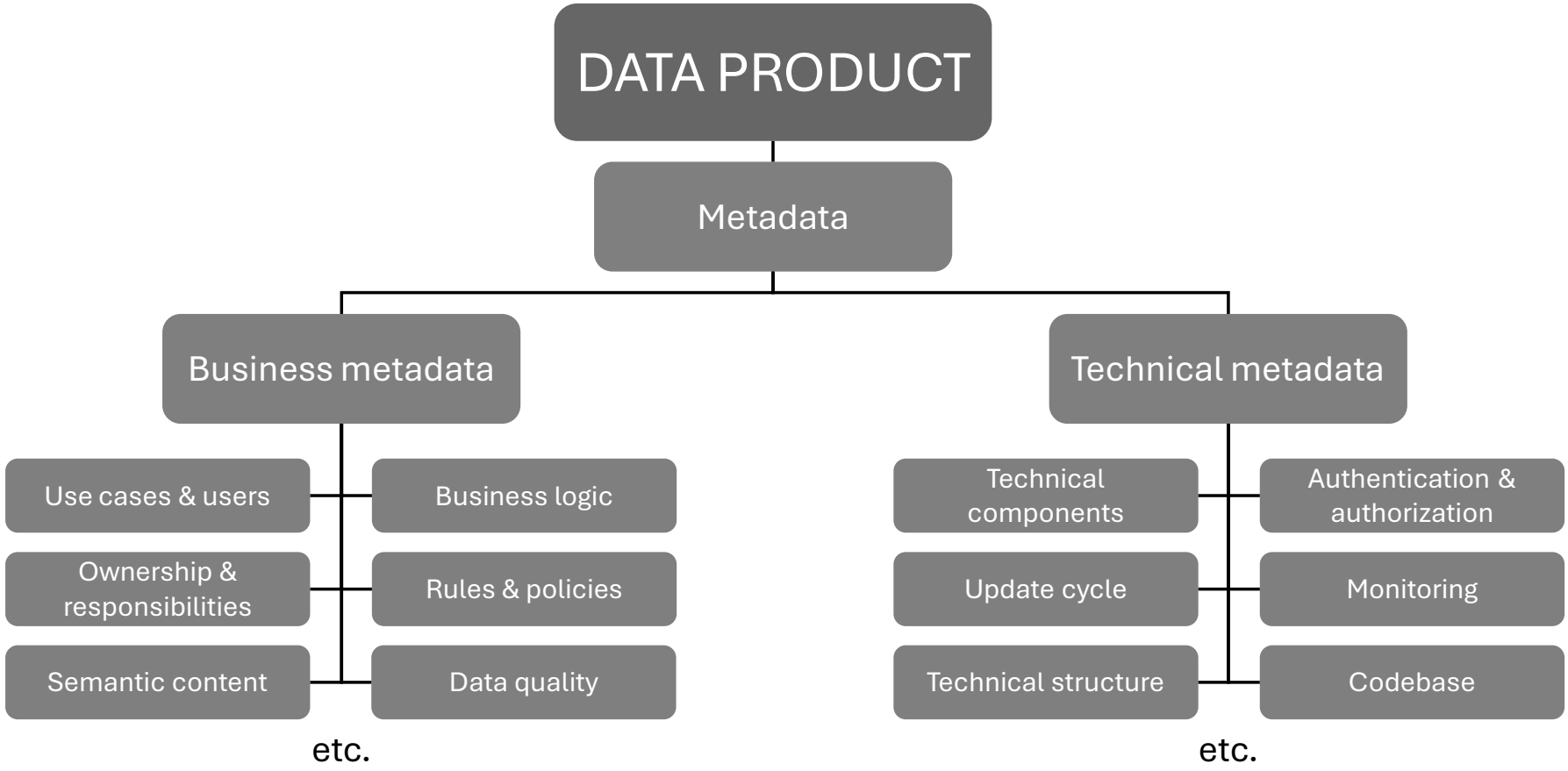
What Data Products are available?

How can they be accessed?

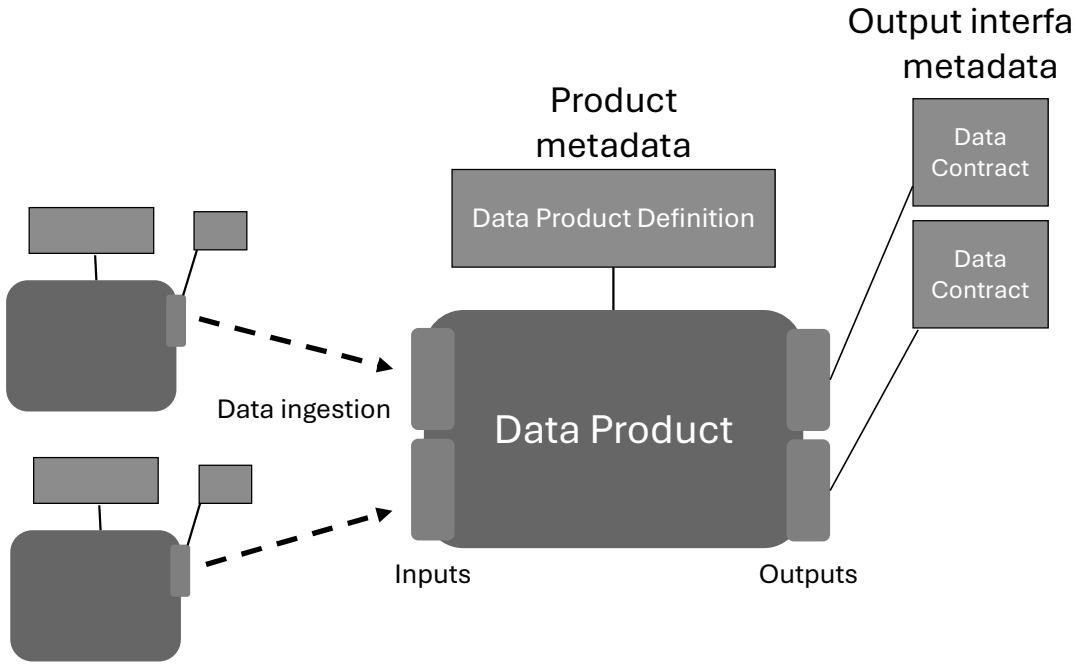
What does the data mean?



Data Product Definition = collection of metadata



Data Product Definition & Data Contracts



Data Contracts for data producers:

- A "promise" of what an interface is and how it behaves
- Must be available programmatically (exposes metadata in a machine-readable format)
- Should enforce schema compliance in development process (check changes against existing contract on deploy)
- Should be versioned (communicate upcoming changes to data consumers!)

Data Contracts for data consumers:

- Specification for technical data interface
- Can be used to double-check against incoming data (ensure no unexpected changes)

Example standards

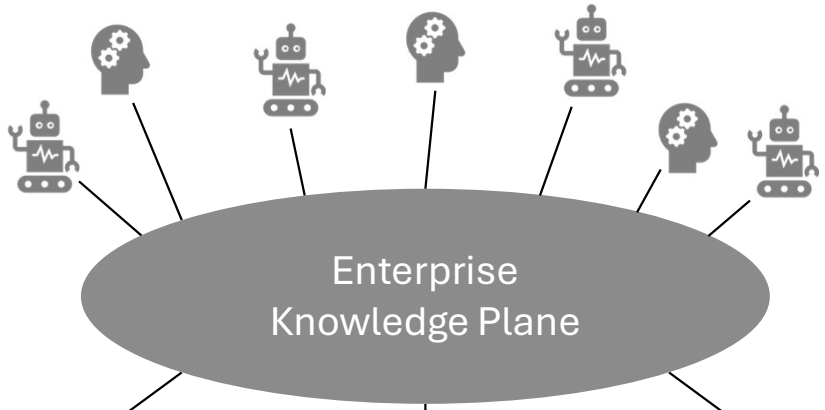
Open Data Product Standard (ODPS)

- Fundamentals
- Product Information
- Management Ports
- Support and Communication Channels
- Team
- Ancillary Objects: Custom Properties
- Ancillary Objects: Authoritative Definitions
- Other Properties

Open Data Contract Standard (ODCS)

- Fundamentals
- Schema
- References
- Data Quality
- Support & Communication Channels
- Pricing
- Team
- Roles
- Service-Level Agreement
- Infrastructures & Servers
- Custom & Other Properties

Enterprise view on semantics



Data Product Definitions need to expose semantics for enterprise-wide discoverability *across domains*

"This product is about Customers and Prospects"

Data Product Definition

DATA PRODUCT

Data Product Definition

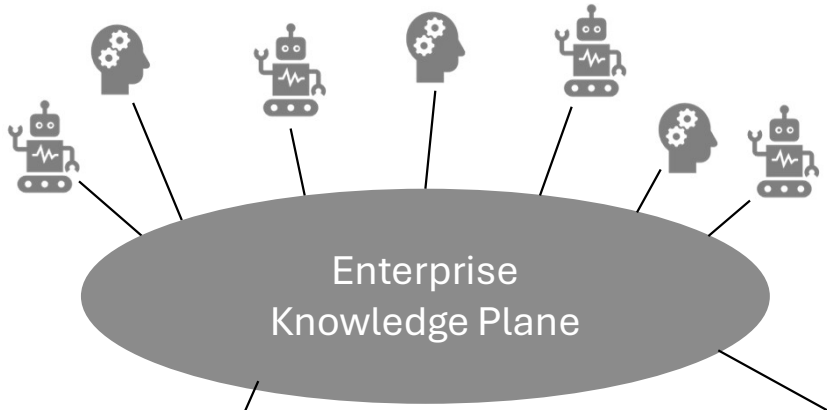
DATA PRODUCT

Data Product Definition

DATA PRODUCT

"This product is about Customers and Deliveries"

Domain vs. Enterprise Glossaries



Enterprise Knowledge Plane

Domain-specific glossaries are needed for domain-specific language, but everything must connect at enterprise level

"This is what Domain 1 means by Customer"

Domain Glossary

Domain Glossary

"This is what Domain 2 means by Customer"

"This product is about Customers and Prospects"

Data Product Definition

Data Product Definition

Data Product Definition

"This product is about Customers and Deliveries"

DATA PRODUCT

DATA PRODUCT

DATA PRODUCT

Language problems

Synonym

Different word, same meaning

Cross-domain and within-domain

“Customer” – “Client”

Homonym

Same word, different meaning

Cross-domain and within-domain

“Account” (sales) – “Account” (finance)

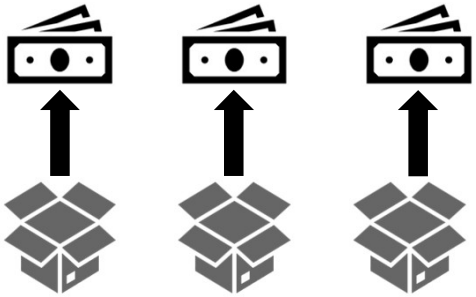
- Use “preferred labels” and “alternative labels”
- Map cross-domain connections at enterprise level
- Prioritize within-domain clarity

Data Mesh information architecture operating model

Data Mesh information architecture operating model

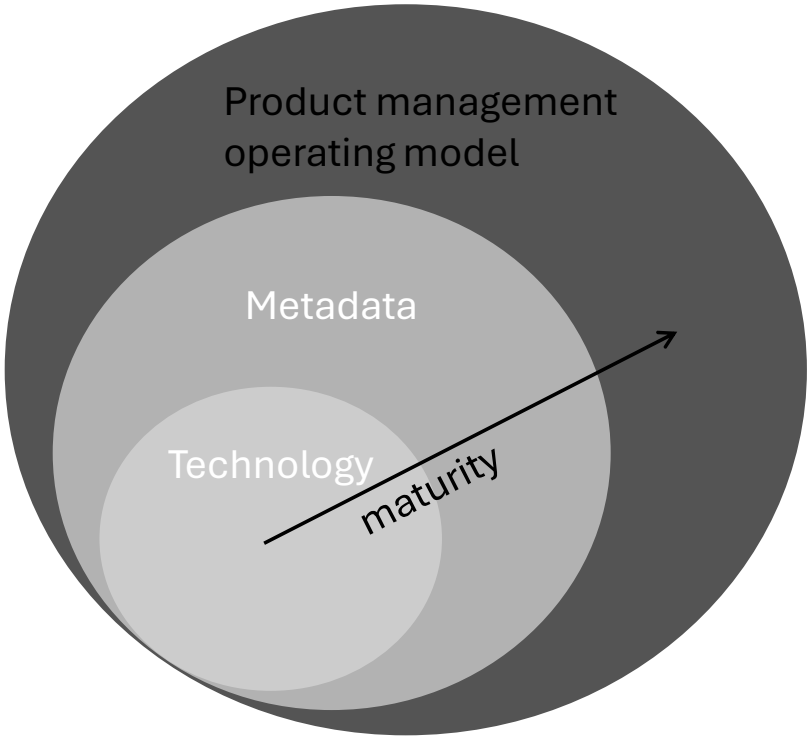
- Domain team responsibilities
- Data product owner responsibilities
- Platform team responsibilities
- Federated governance

Scaling value

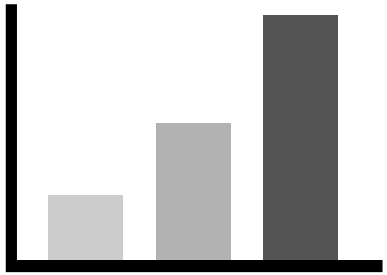


Individual data products
create direct business value

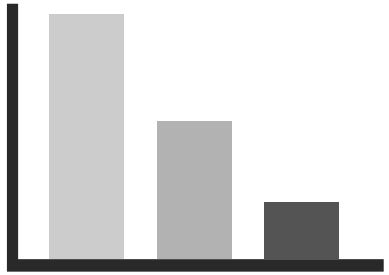
Enterprise-level approach
enables value at scale



Value to organization



This is often the focus
of the organization
– road to failure!



Roles and teams

Data Product Owner

- Responsible for a Data Product throughout its lifecycle
- Decides on changes and prioritization (backlog)
- Primary contact for everything related to the particular Data Product
- Often responsible for specifications

Domain Owner

- Responsible for all data within a domain
- Often the “official” data owner across products
- Decides on general principles and policies within the domain
- Owns the Domain Glossary

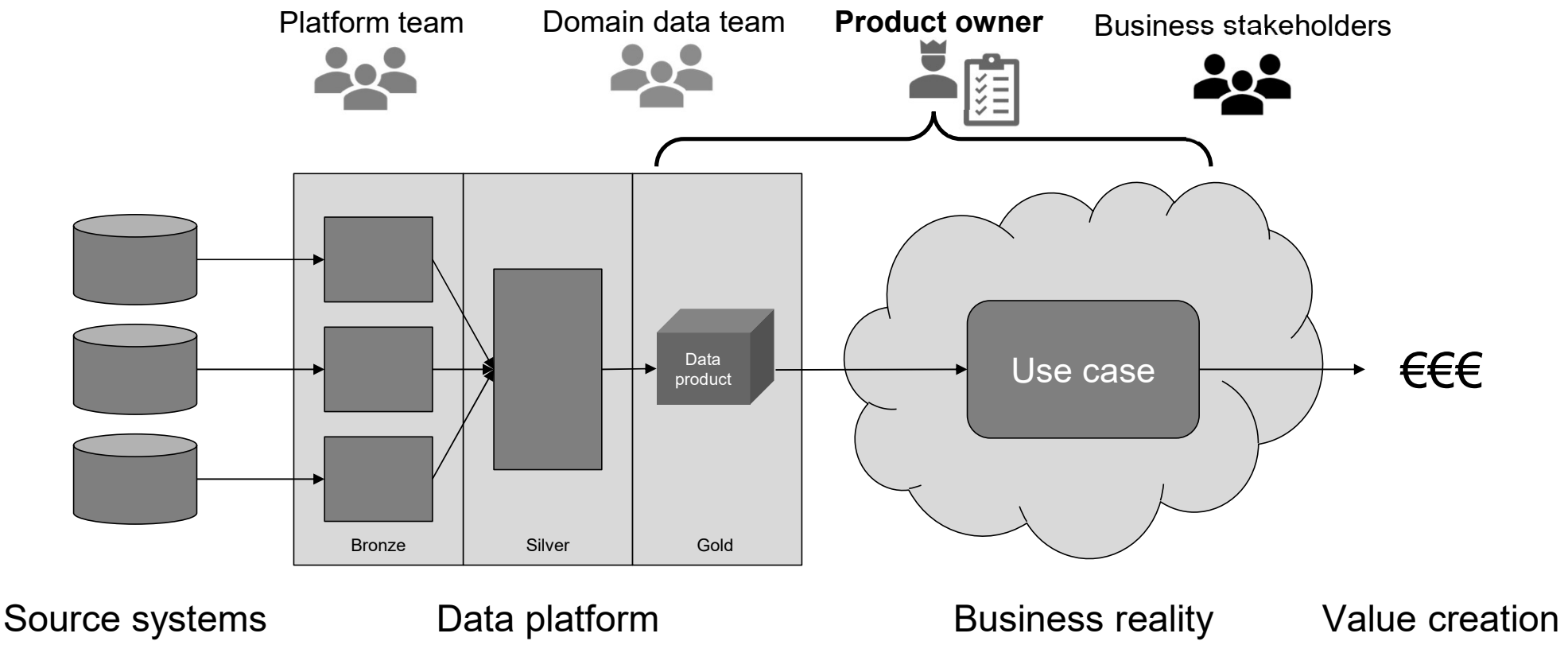
Platform Team

- Centralized team, responsible for a technical platform service
- Doesn’t do actual Data Product implementation work!
- Ensures that Domain Teams have everything they need to create Data Products
- Often includes a Platform Owner role

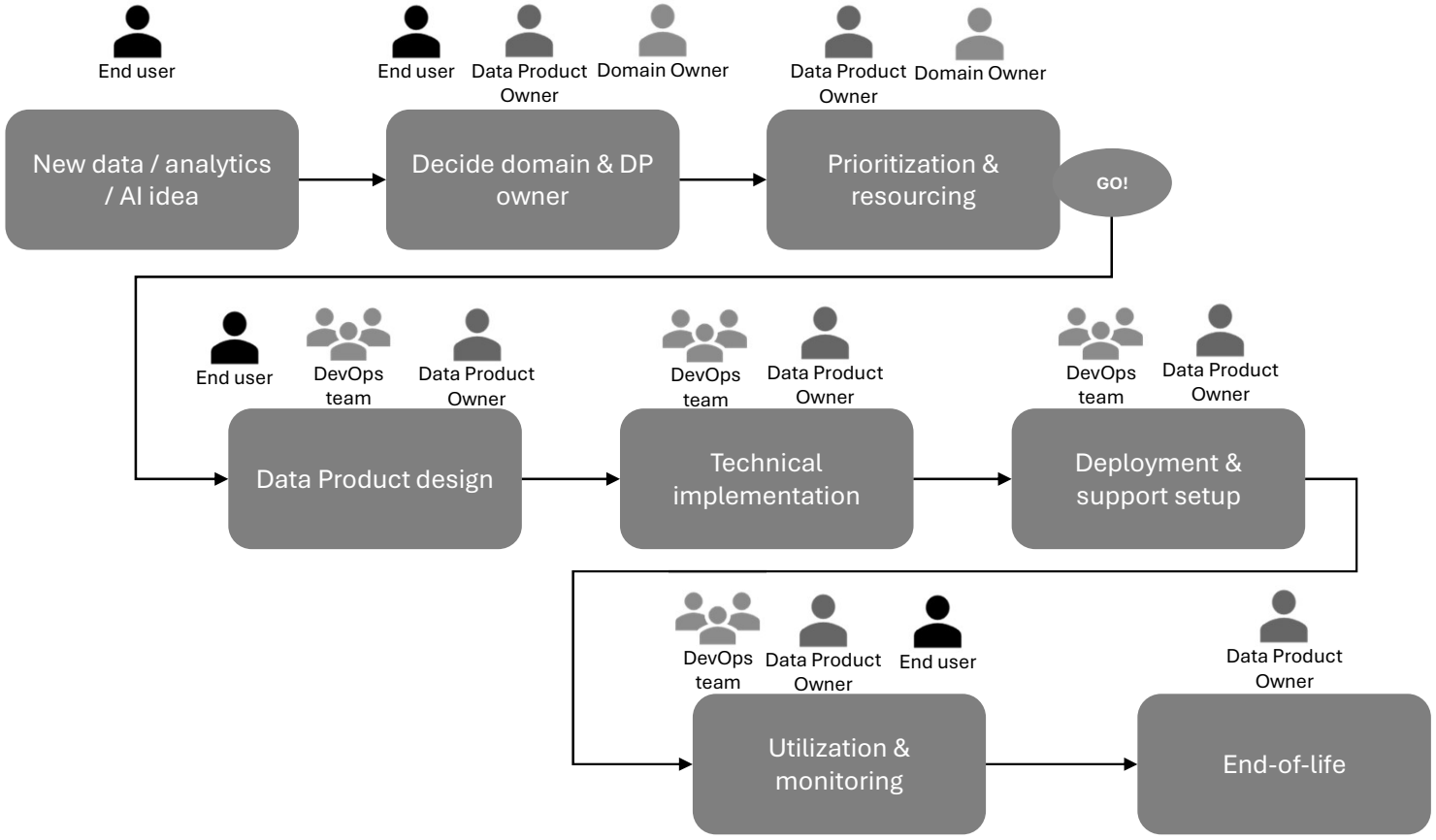
Domain DevOps Team

- Federated team, responsible for one domain and all the Data Products in it
- Can develop new Data Products independently (all necessary skills included)
- Good domain knowledge
- Sometimes incl. both analytical and operational development

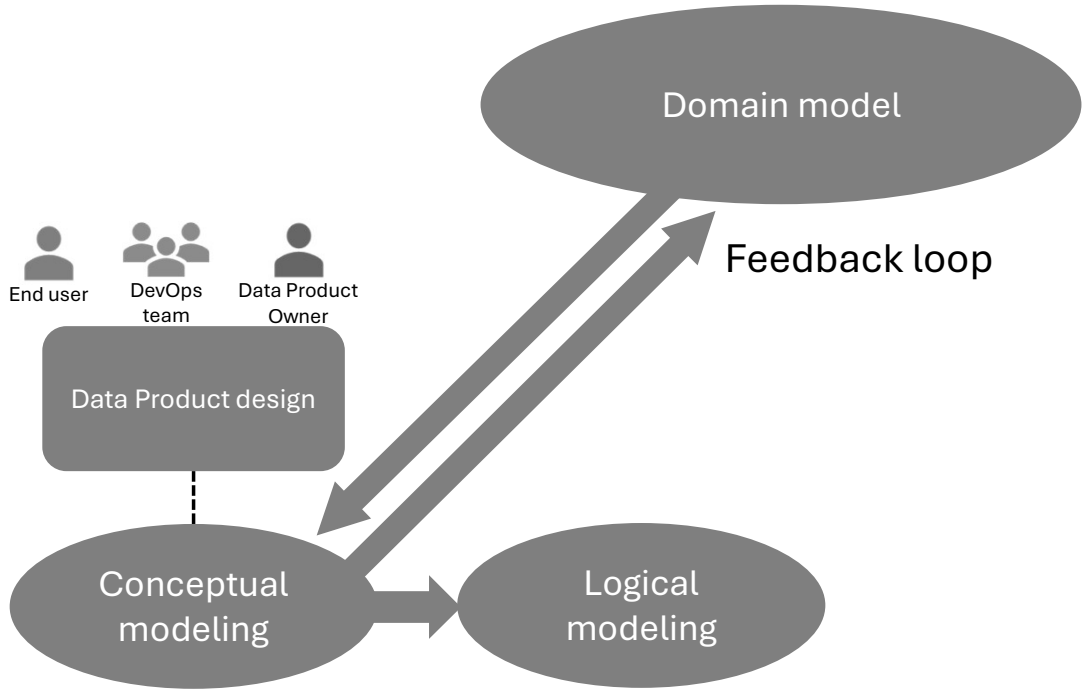
Product ownership & backlog management



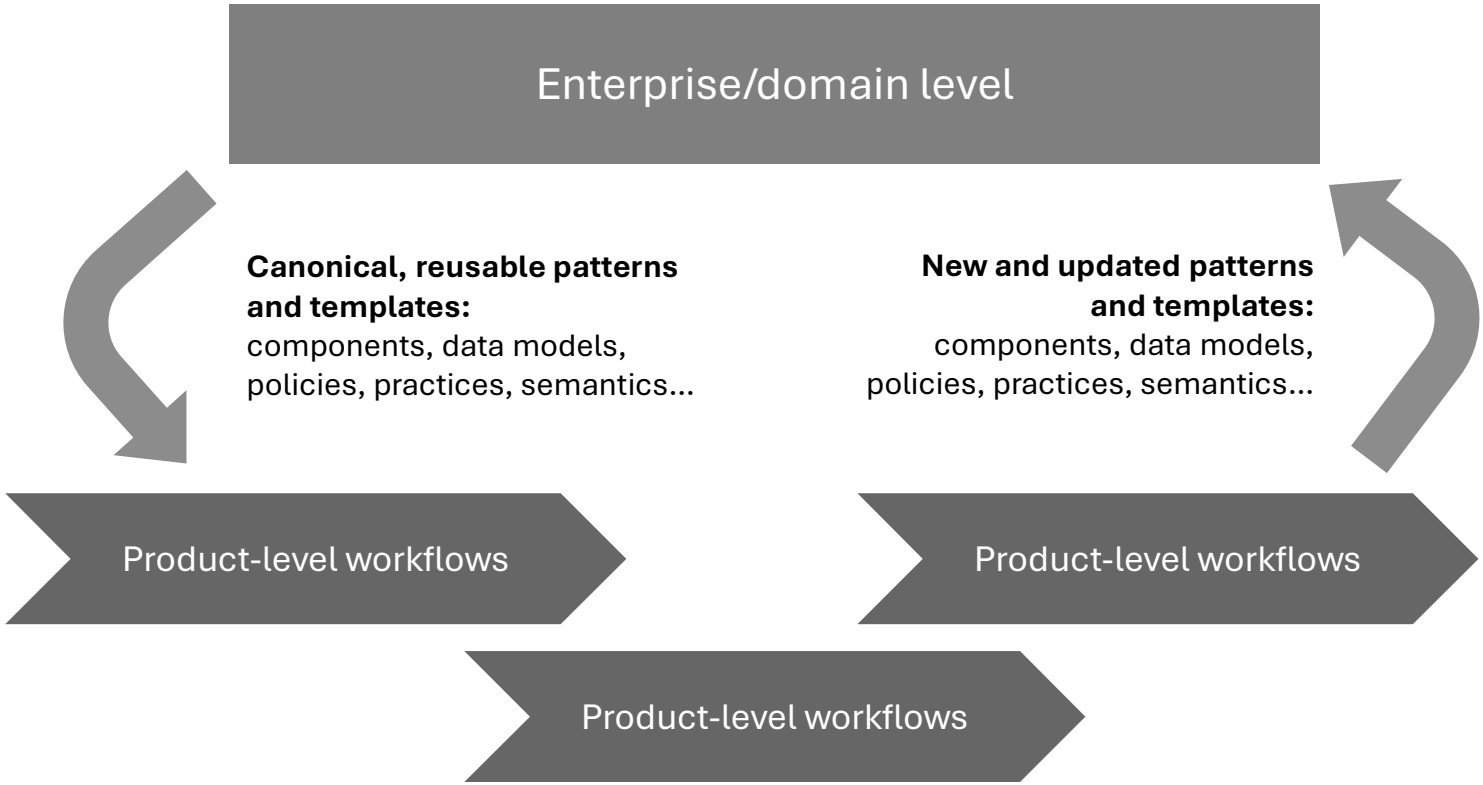
Data Product lifecycle management



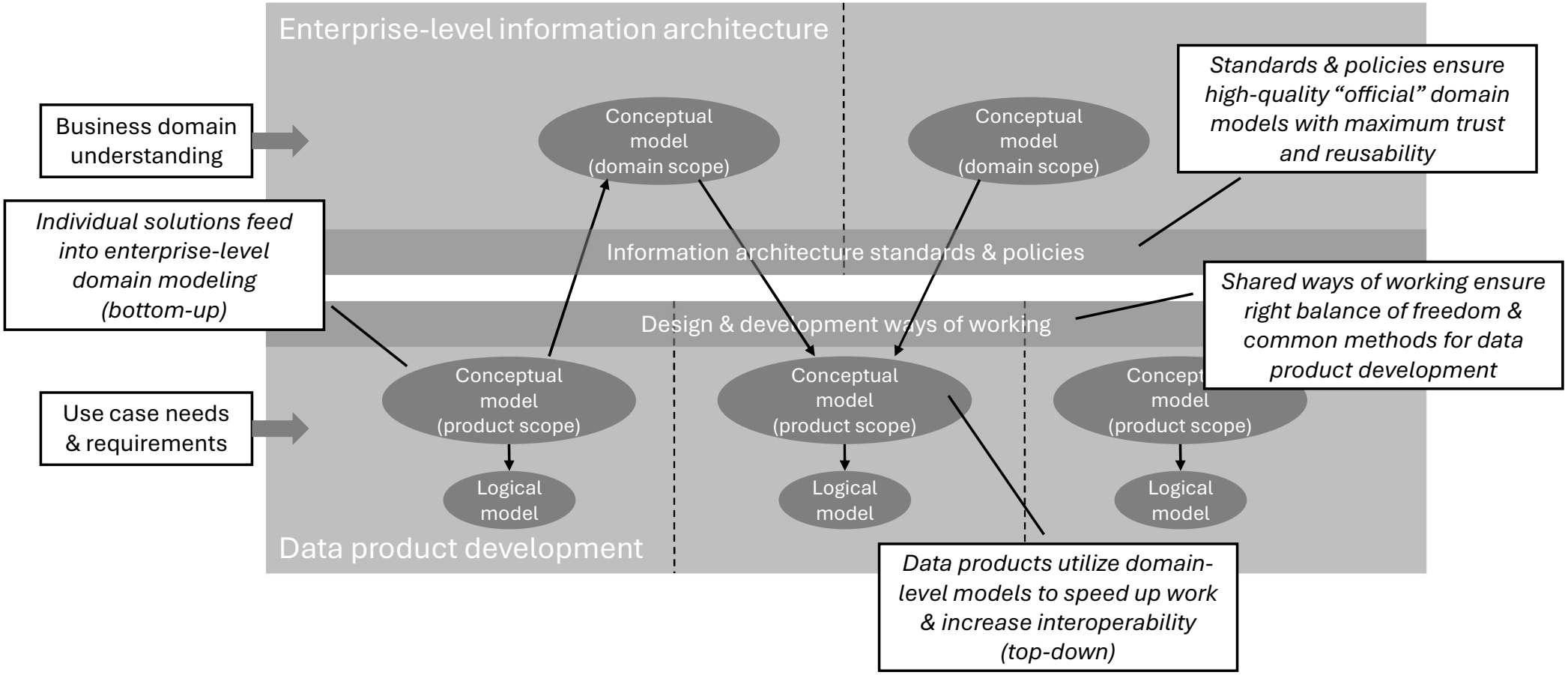
Modeling at design stage



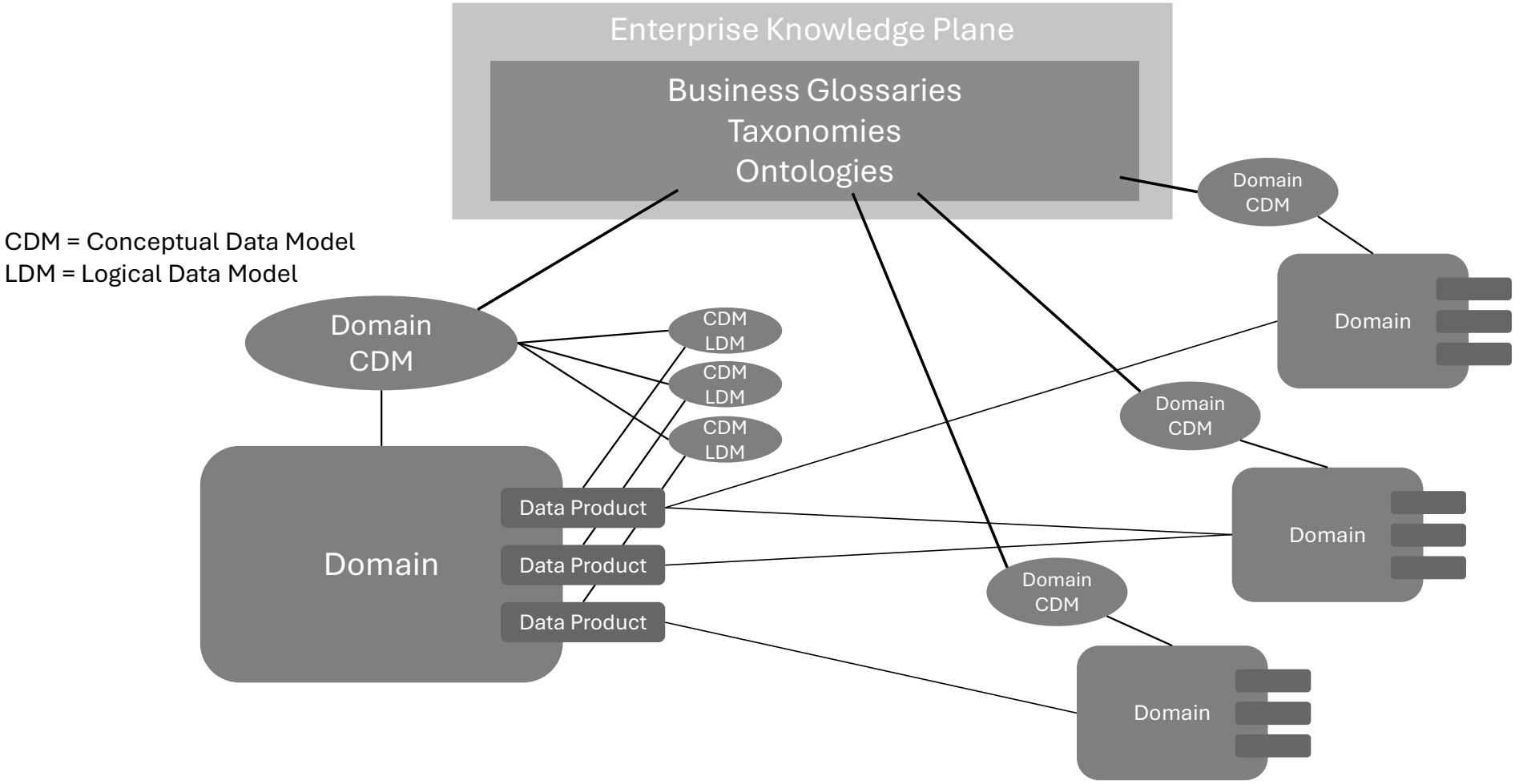
Key success factor: feedback loops



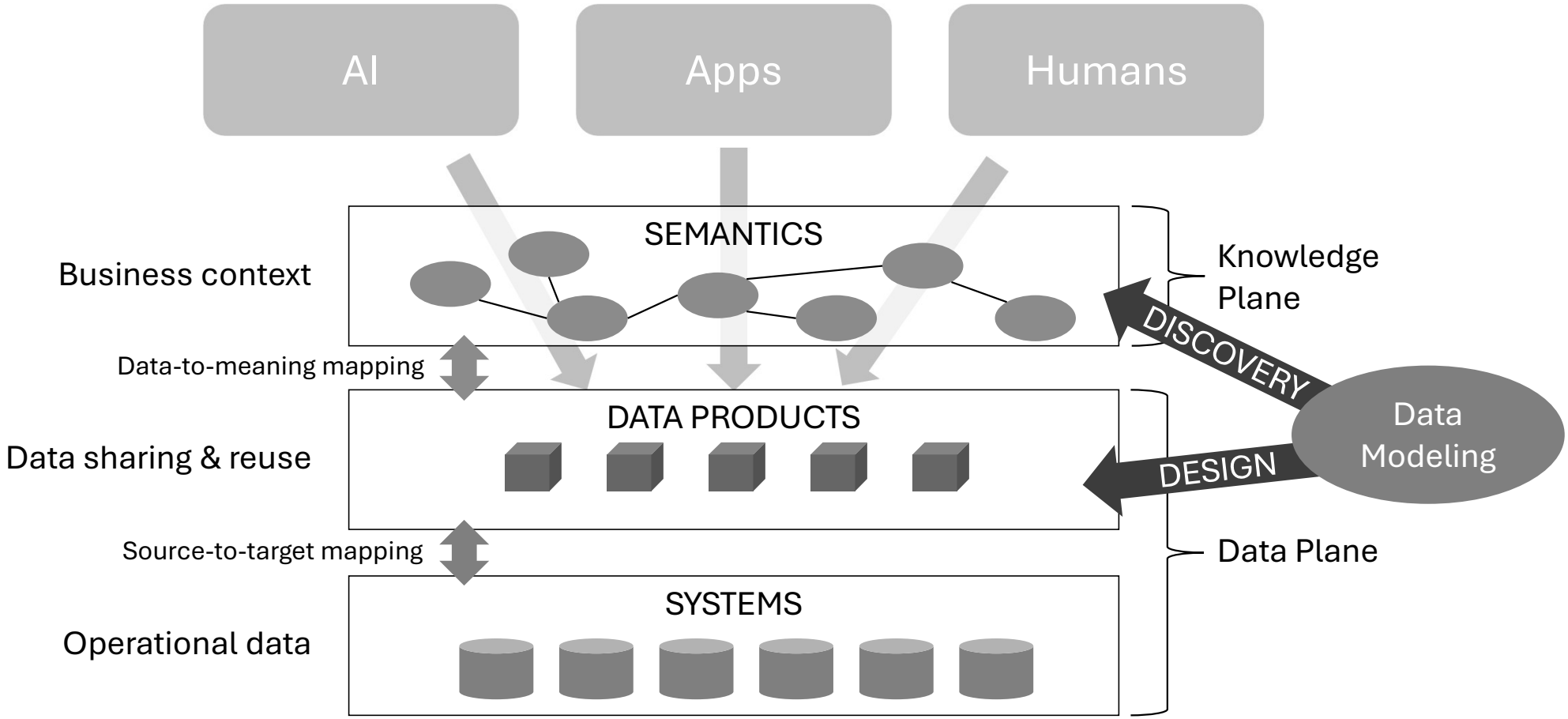
Organizing data modeling on two levels



Cross-domain interoperability



Goal: context-aware data utilization



Conclusions

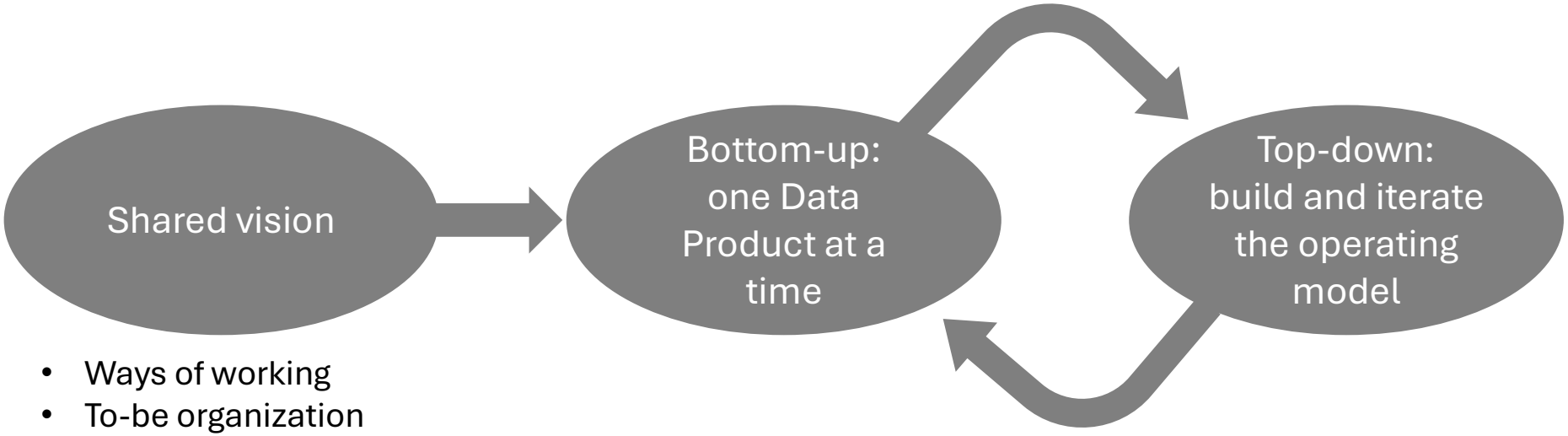
Conclusions

- Key takeaways
- Where to start in your organization
- How to learn more

Key takeaways

- Federated operating model in Data Mesh creates challenges at the domain boundary (semantic interoperability)
- Data modeling helps with design and documentation of both domains and data products
- Conceptual data modeling discovers semantics – the meaning of data
- Metadata from data products (Data Product Definition) and product interfaces (Data Contract) need to be made available & connected with the domain models
- Enterprise-level management of semantic information (Knowledge Plane) connects domains and explains polysemes

Path forward

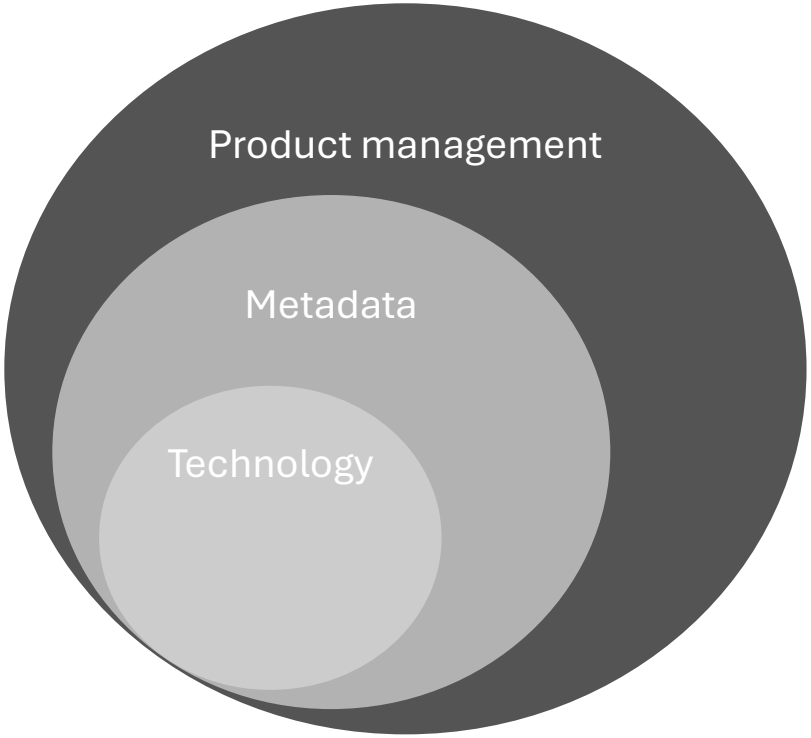


- Ways of working
- To-be organization
- Metadata management

Data Product Management training & advisory services

DATAKOR CONSULTING

Training, consulting & advisory services on Data Products, Data Modeling, and Information Architecture



More thoughts on Substack!