Data mesh

by Nicola Orecchini, 24/07/2025

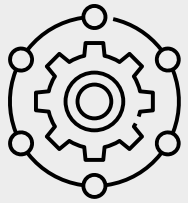
What you'll find in this presentation

Chapter	Content
1. Challenges of monolithic data achitectures	Monolithic data architectures & their main points of failure
2. Data mesh: a new decentralized data architecture	A proposed evolution to solve some challenges

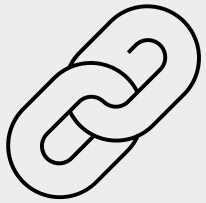
1. Challenges of monolithic data achitectures

Most of the modern data architectures (such as data lake) are composed by a single, centralized, monolithic piece

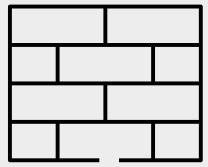
There are 3 main failure modes in such an architecture



Centralized and monolithic

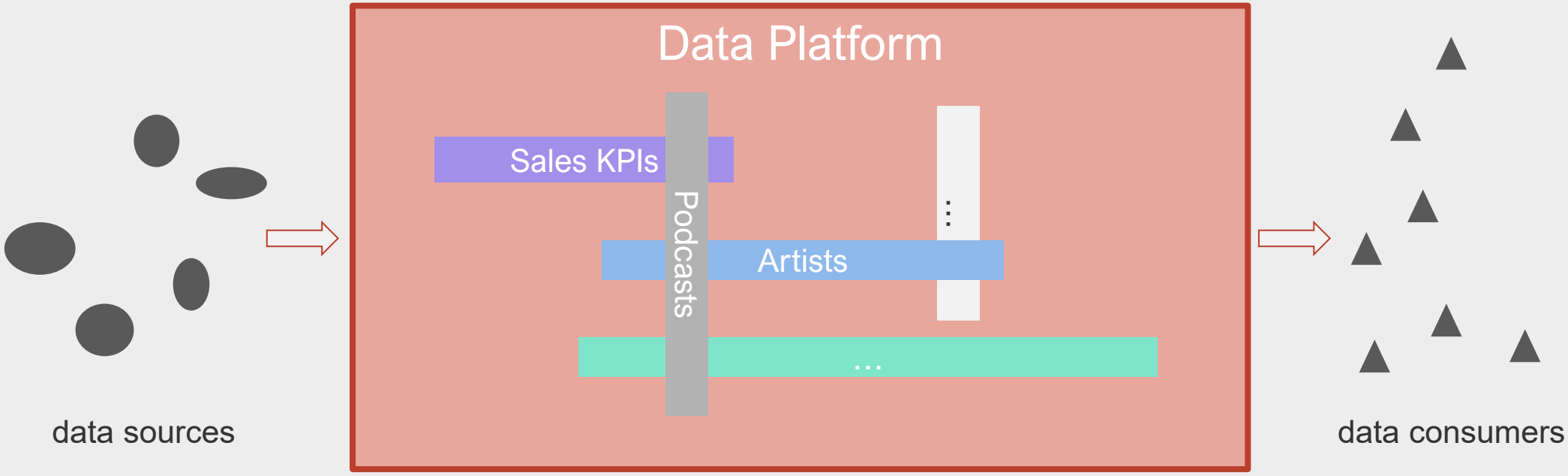


Coupled pipeline decomposition



Siloed and hyper-specialized ownership

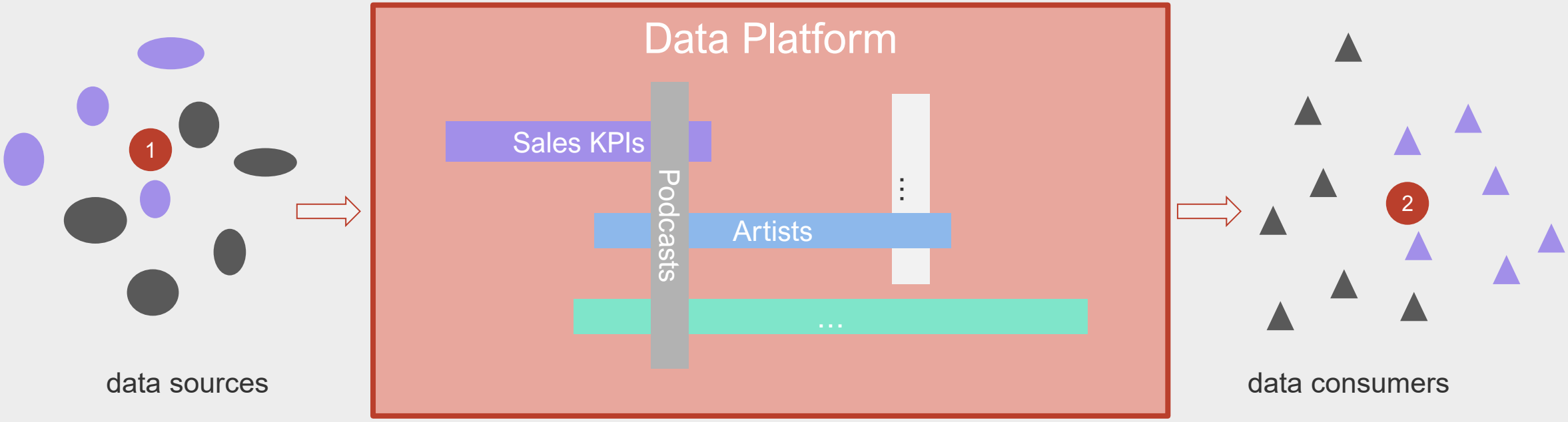
Inside the data platform, there are data from many disparate domains with no clear boundaries and ownership



It is an accepted convention that the monolithic data platform hosts and owns **all the data in the organization**, even if that logically belong to different domains

Furthermore, **data from different** domains are often in **overlap**

2 pressure points threaten the efficiency of a centralized platform



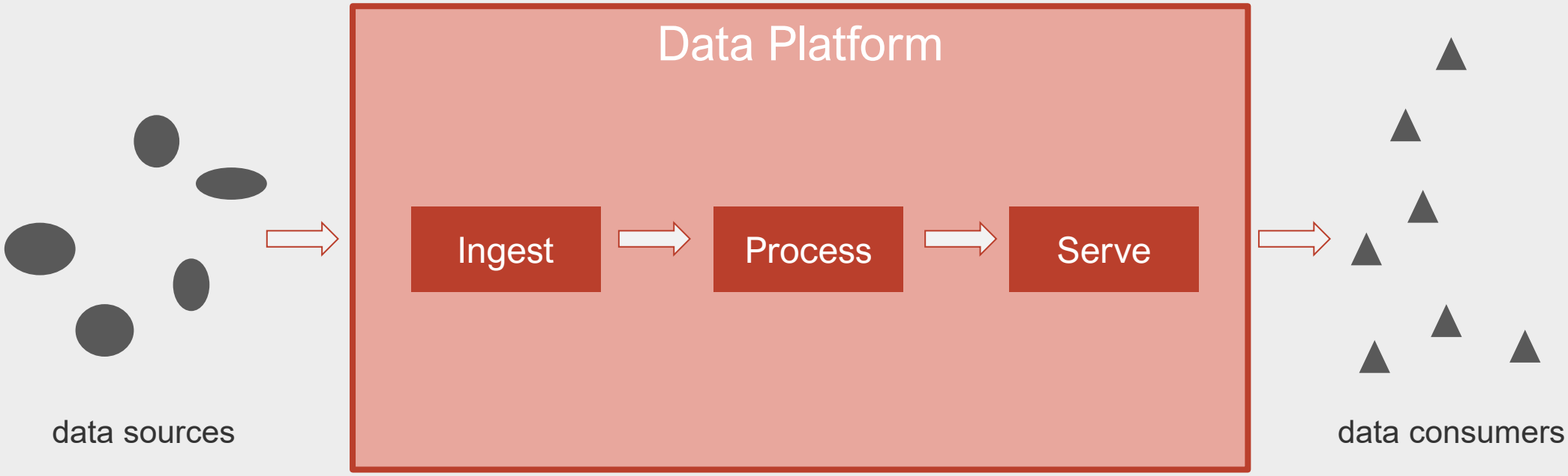
1 Data source proliferation

If the data sources start to become more and more, the assumption that we need to ingest and store all the data in one place will hinder our ability to timely & efficiently integrate the new data sources

2 Consumer proliferation

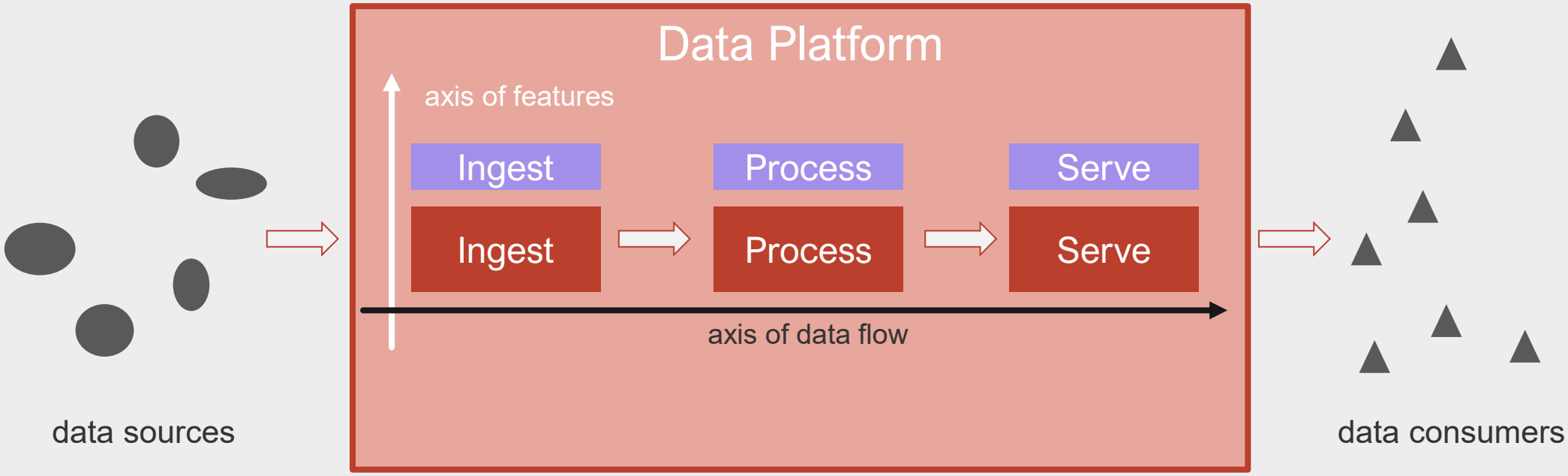
Organizations' need for rapid experimentation introduces a large number of use-cases for consumption of the data from the platform, which imply a growing number of transformations on the data, increasing the response time of the engineers

If we zoom in on the architecture, we find the data platform is decomposed to a pipeline of data processing steps



The grounding idea is that by decomposing the data platform architecture into separated capabilities, **teams can work parallely** on different stages of the processing, and the work can **scale**

However, the high level of coupling between stages slows the delivery of new features



On paper, with pipeline decomposition, it appears that we have achieved **modular components** that represent the **smallest units of change**

From the point of view of implementing a new feature, though, the **smallest unit of change** is still the **entire pipeline**: if you introduce a new feature, you must **change every component of the** pipeline. Thus, the pipeline decomposition is still **coupled**

We could say that the pipeline decomposition acts on the **axis of data** flow, which is orthogonal to the **axis of features**

If you decompose the pipeline along the axis of data flow, you will achieve **benefits** on **parallelizing stages implementation**, but **no benefits** on **new features implementation**

Centralized platforms are built and owned by… guess who? If you said *one single centralized team*, you're right!

Source teams

Purchase events team

Song play events team

…

Cross-functional

Domain-oriented

- They often become disconnected

Platform Engineers

Data & ML Platform Engineers

Hyper-specialized

Over stretched

Siloed from the operational teams that create or consume data

- They have very little understanding of the source domains that generate the data and lack the domain expertise
- They need to provide data for a diverse set of needs, without a clear understanding of the application of the data and access to the consuming domain's experts

Consumer Teams

Sales team

Artists paying team

…

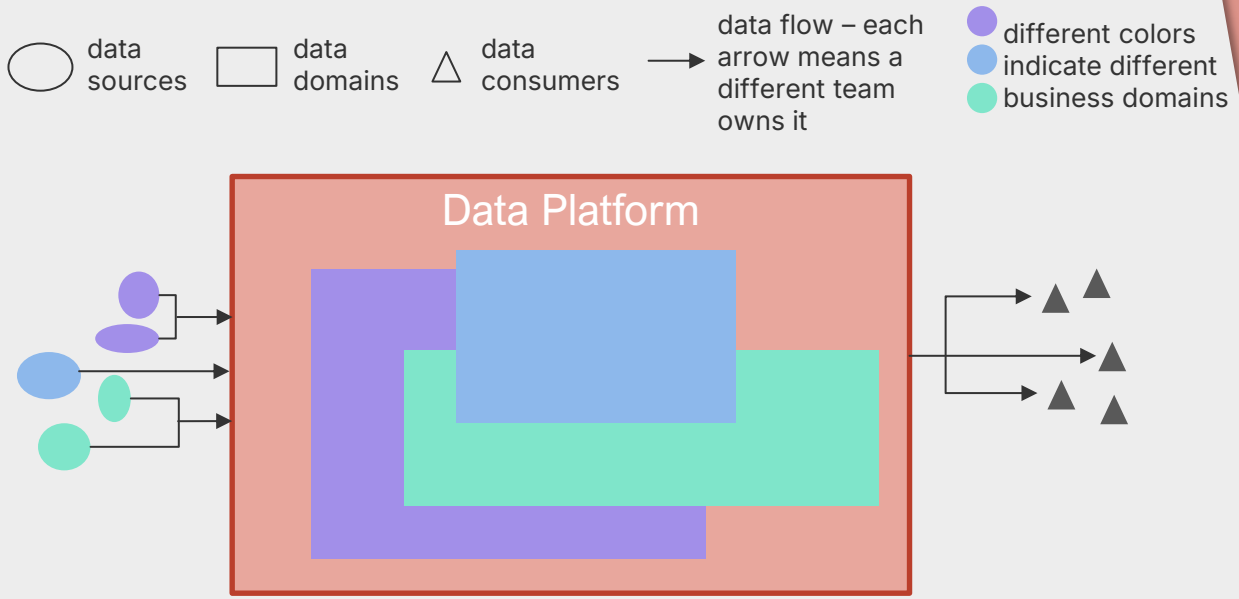
Cross-functional

Domain-oriented

- They start fighting for a spot on top of the data platform team backlog

2. Data mesh: a new decentralized data architecture

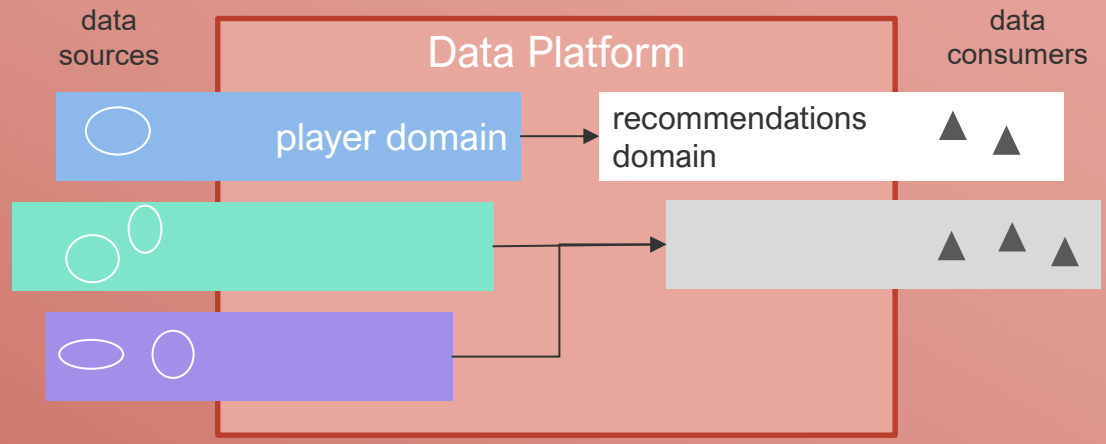
To decentralize the monolithic data platform…



Push & ingest model

- each source operational system (managed by a domain team) feeds data organized by domains into the platform, not caring where these data will be consumed
- the centrally owned data platform ingests the data
- after ingestion, the concept of domain is lost, and one single platform team is responsible of providing data from the platform to consumers
- the architectural quantum is represented by a stage of the pipeline (e.g., ingest/process/serve)

…a paradigm shift is required to a domain-oriented data platform



Serve & pull model

- each domain owns, hosts and serves their datasets for access by any team downstream
- the physical location where the datasets actually reside and how they flow is a technical implementation of the *player* domain
  - the physical storage could still be centralized (e.g., Amazon S3 buckets), but domain datasets content and ownership remains within the domain generating them
- the architectural quantum is represented by a domain (e.g., player, recommendations, etc.)

1) Domain Driven Design

Disclaimer

This is my personal re-elaboration of Zhamak Dehghani's (the inventor of Data Mesh concept) article *How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh* (https://martinfowler.com/articles/data-monolith-to-mesh.html). I do not own the rights about the idea. For what concerns this deck, most of the sentences are copy-pasted from the article, and the graphics are also very similar to those of the article. But both the text and the graphics have been slightly re-elaborated by me, with the goal of making them more accessible and easier to understand.