DATA MESH ARCHITECTURE

MINI-PROJECT REPORT - TOPIC 8

DATA ENGINEERING SPECIALIZATION

RESIDENT: Tạ Ngọc Minh
BSc. student in Data Science & AI @ HUST

SUPERVISOR: Mr. Nguyễn Chí Thanh

Hanoi - 2023

PREFACE

Dear the board of directors of Viettel Group, the mentors, and all of my friends,

This document is a part of my research mini-project as a data engineer intern for the first period of Viettel Digital Talent 2023 Program. The completion of this research report does not mean the end of my duties as learners. In fact, this report is a first step towards consistently studying what I have reviewed and written here.

Data has grown up so quickly recently. The term 'big data' has become familiar with every tech-lovers is the clearest evidence of the explosion of data. This leads to many requirements of upgrading the current system and management, along with finding new architecture to store, process and analyze data. Once the central data lake has been overloaded, we need to design a decentralized architecture. This report is written under the research about the Data Mesh architecture, a new design that was first defined by Zhamak Dehghani in 2019.

To write this document, I would like to give many thanks to Viettel Group, and Viettel Digital Talent program for giving me chance to do my research. I want to send a big thanks to Mr. Nguyễn Chí Thanh, and all the mentors of the first period of the program, who gave me many useful lectures and guidance. Also, I want to say thanks to professor Huỳnh Thị Thanh Bình, professor Đỗ Tuấn Anh for their initial supports, and many other professors at the School of Information and Communication Technology, Hanoi University of Science and Technology about their public documents about data engineering. Last, I would like to say thanks to all of my friends, who always read my documents, comment about them and support me whenever I need.

Because this document is written in such a short time and some of the concepts are so new and hard to understand for a fresher like me, it is so hard to make sure it is completely correct, including the grammar errors and formatting errors in LaTeX. I would love to receive your contributions via **ngocminhta.nmt@gmail.com**.

Thank you for reading this report and looking forward to getting many comments from you.

Ngoc-Minh Ta

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Problems and Initial Approach

1.1 Real-life Problem

1.1.1 A problem from history

Many organizations have invested in building a central data lake. To make data-driven business, there is a central data administration team take the responsibility for this.

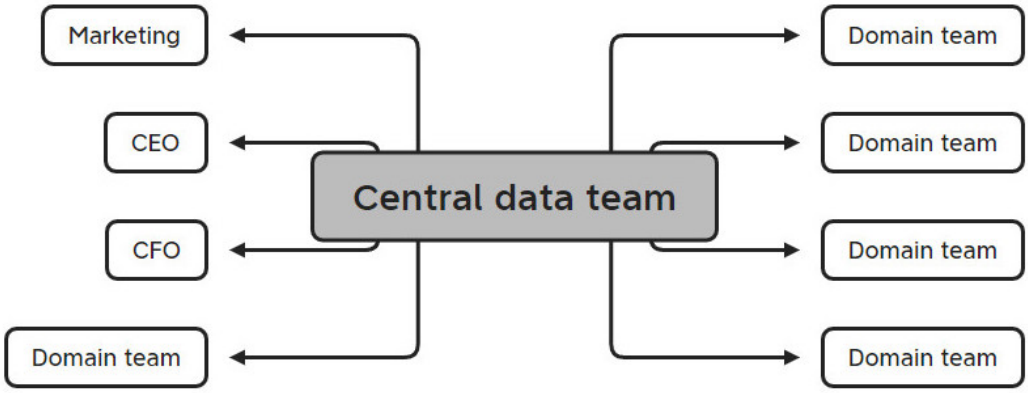

Figure 1.1: Central data team in a data-driven business

This model works quite well at the first time, however, after a while, they noticed that **the team often became bottleneck**. The team were overloaded with thousands of tasks and questions from multiple teams, along with the requests of time and accuracy. This lead to a massive problem since the competitiveness of business depends much on the speed of analysis. For example, when we create a landing page for a new product, how does this change the influence clicking and buying rate?

But, why the response of the central data team is so slow and struggle? Actually, once the operational database changes, the team has to spend much of their time fixing the broken data pipelines. The little remaining time is insufficient for them to discover and understanding all the necessary domain data **for each question**. Getting the required domain expertise is a daunting task. [1, 2]

In contrast, some firms employ domain-driven design strategies that include independent domain teams and a decentralized micro-service architecture to relieve burden on the central data team. These teams exclusively control and understand their domain, including the company's information demands. The issue hasn't entirely been resolved, though. The domain teams must contact the overloaded central data team despite being aware of the key information needs and the domain, in order to obtain the essential data-driven insights. [1]

1.1.2 Demands from recent developments

Let's consider the scale-up of the software development over time. When one task grow bigger and bigger, we have to decentralize it into many smaller tasks, otherwise, all the organization will overload and lost of control. Take into consideration what we have done for the scale-up of software development:

- Decentralize business into domains;
- Decentralize engineering into autonomous teams;
- Decentralize monolith into micro-services;
- Decentralize operations into DevOps teams.

Hence, the next thing we need to do is scaling up data analytics by decentralizing data lake into data mesh.[2]

The position between domain teams and the central data team gets worse as the organization eventually grows. Transferring data management responsibilities from the central data team to the domain teams can help. Domain-oriented decentralization for analytical data is the central notion of the data mesh concept. Similar to APIs in a micro-service design, a data mesh architecture allows domain teams to conduct cross-domain data analysis on their own and connects data.

1.2 INITIAL APPROACH

1.2.1 What is Data Mesh?

The term data mesh was first stated by Zhamak Dehghani in 2019 and is based on four fundamental principles, which will be discussed in depth later:

- Principle of Domain Ownership;
- Principle of Data as a Product;
- Principle of the Self-Serve Data Platform;
- Principle of Federated Computational Governance.

1.2.2 What will changes after data mesh?

Data mesh is a fresh method for business intelligence, data analysis and management that is based on an innovative distributed architecture. Along with that, data lake and data warehouse do not disappear they just become nodes in the mesh. [3, 4]

Data mesh ensures organizations to continue to apply some data lake principles, such as making immutable data available for exploration or analytical use, and data lake tooling for internal implementation of data products or as part of the shared data infrastructure. Data lake would not be the centerpiece anymore.

It is an implementation detail sub-serving the idea of domain data product as the first-class concern. The same applies to data warehouse in terms of business reporting and visualization. [4]

Implementing a data mesh is not a purely technical project that we can implement in isolation from the rest of the business. It's not some thing that we can just start with, and then it works from the first try, but we have to grow and develop with it. It also cause a cultural shift in the organization.

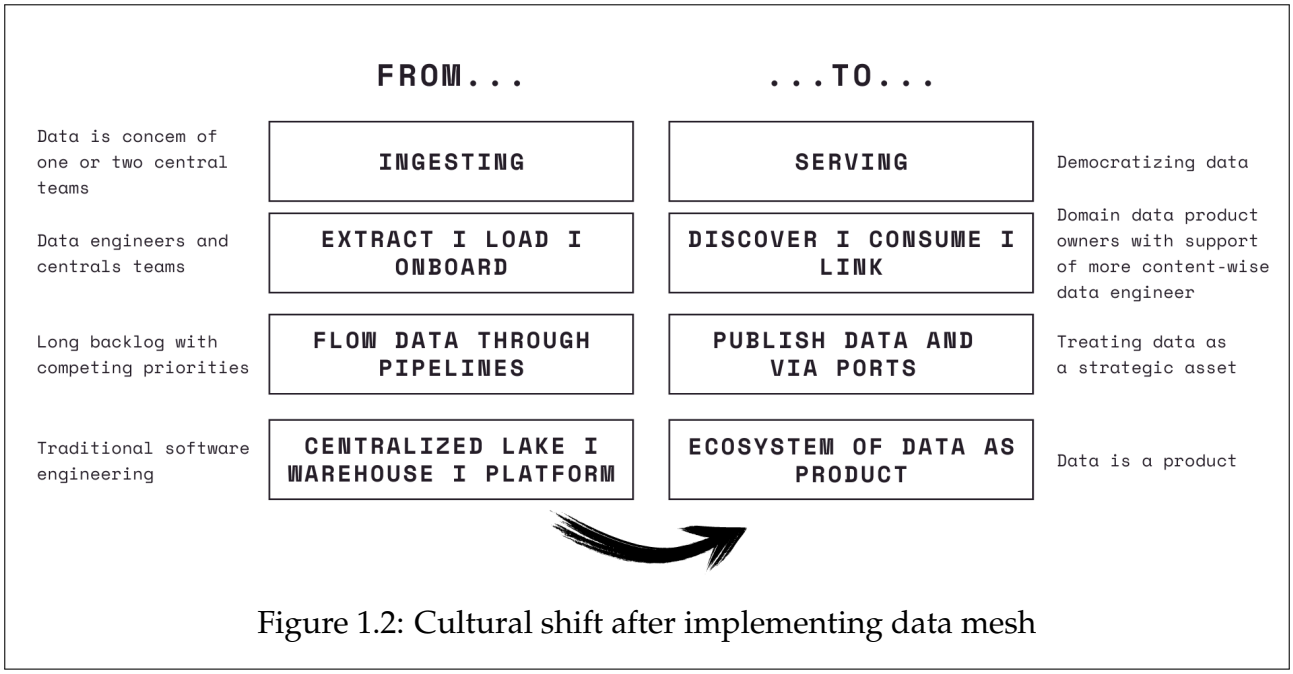

Figure 1.2: Cultural shift after implementing data mesh

Also, data mesh calls for a fundamental shift in the assumptions, architecture, technical solutions, and social structure of our organizations, that means, how we manage, use and own analytical data. [5]

Data mesh can be used as part of an enterprise data strategy, articulating the target state of the enterprise architecture as well as an organizational operating model with an iterative execution plan.

In its most basic form, it can be characterized by four interacting principles, which will be discussed in depth in section 2.1.

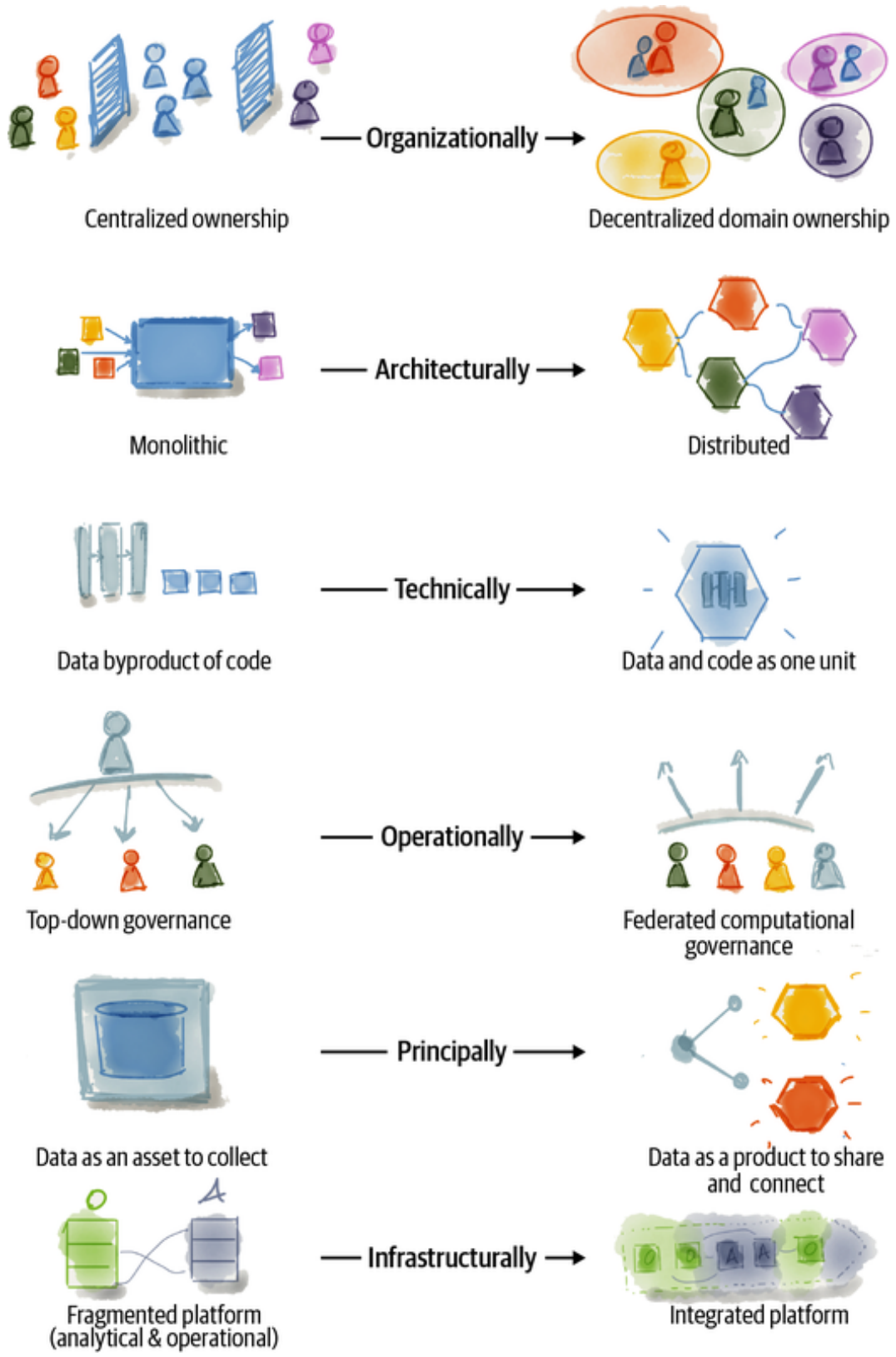

Figure 1.3: Data mesh dimensions of organizational changes

Chapter 2

DATA MESH ARCHITECTURE DESIGN

2.1 FOUR FUNDAMENTAL PRINCIPLES OF DATA MESH

Four simple principles can represent the logical architecture and operating model of data mesh. They are intended to move us closer to the goals of data mesh: increasing the value of data at scale, maintaining agility as an organization grows, and embracing change in a complex and turbulent business context.

2.1.1 Principle of Domain Ownership

Data mesh, at its core, is founded in decentralization and distribution of data responsibility to people who are closest to the data. This is to support a scale-out structure and continuous and rapid change cycles.

However, in contrast to traditional data structures with technological partition (e.g. data warehouse, data lake), data mesh follows *the seams of organizational units*. It follows the lines of division of responsibility aligned with the business using *domain-driven design (DDD) strategies*. Data mesh also gives the data sharing responsibility yo each of the business domains, each domain is responsible for the data it is most familiar with.

DDD is an approach to decomposition of software design and team allocation, based on the seams of a business. It defines a *domain* as "a sphere of knowledge, influence or activity." DDD's Strategic Design embraces modeling based on multiple models each contextualized to a particular domain, called a bounded context[1]. As Z. Dehghani's recommendation, data mesh adopts the boundary of bounded contexts to individual data products - data, its models, and its ownership.

Domain data ownership is the foundation of scale in a complex system like enterprises today. When we map the data mesh to an organization and its domains, we discover a few different archetypes of domain-oriented analytical data:

- Source-aligned domain data (native data product): Analytical data reflecting the business facts generated by the operational systems.

[1] A bounded context is the delimited applicability of a particular model that gives team members a clear and shared understanding of what has to be consistent and what can develop independently. [6]

- Aggregate domain data: Analytical data that is an aggregate of multiple upstream domains.
- Consumer-aligned (fit-for-purpose) domain data: Analytical data transformed to fit the needs of one or multiple specific use cases.

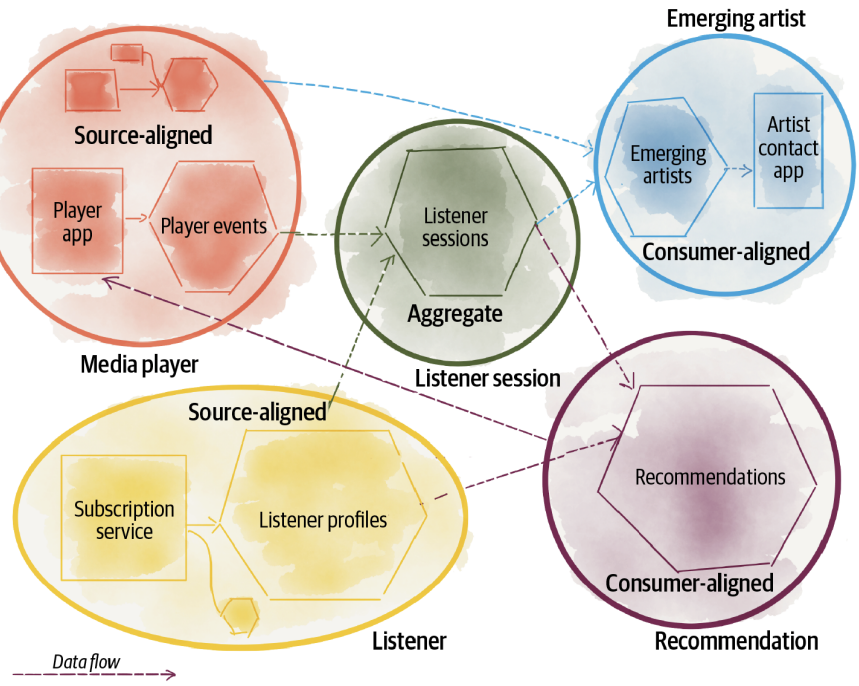

Figure 2.1: Decomposing the analytical data ownership and architecture, aligned with business domains, along with their data archetypes.

The shift toward domain-oriented data ownership leads to accepting and working with real-world messiness of data, particularly in high-speed and scaled environments:

- **Push Data Ownership Upstream:** Data can be consumable and useful right at the source analytical domain (source-aligned domain data). At a later point downstream, source-aligned domain data can be aggregated and transformed to create a new higher order insight (aggregate domain data or fit-for-purpose analytical data).
- **Define Multiple Connected Models:** We implement multiple models of polysemes[2]. In data mesh, each domain can model its data according to its context, share this data and its models with others, and identify how one model can relate and map to others.
- Data mesh does not enforce the idea of searching for the single source of truth for each shared business concept. However, it places multiple practices in place that reduces the likelihood of multiple copies of out-of-date data. Long-term domain-oriented ownership with accountability to share discoverable, high-quality, and usable data in multiple modes for analysts and scientists.
- **Hide the Data Pipelines:** Data pipelines are first-class architectural concerns in traditional data architectures, composing more complicated data processing and transportation. A data pipeline in data mesh is just an internal implementation of the data domain that is handled within the domain.

[2]Polysemes are shared concepts across different domains. They point to the same entity, with domain-specific attributes. [7]

Domains are taking up extra data responsibilities with data mesh. To acquire agility and authenticity, responsibilities and efforts transfer from a centralized data team to domains. [7]

2.1.2 Principle of Data as a Product

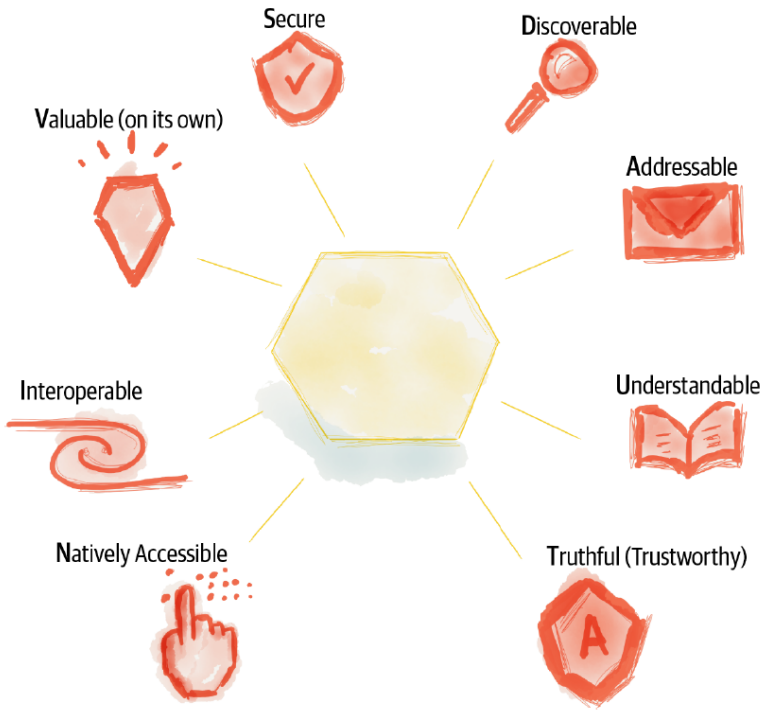

Figure 2.2: The baseline usability attributes of data products (DAUTNIVS)

The principle of data as a product is a response to the data siloing challenge that may arise from the distribution of data ownership to domains. It is also a shift in the data culture toward data accountability and data trust at the point of origin. The ultimate goal is to make data simply usable.

In approach to data product, there is a set of non-negotiable baseline characteristics in figure 2.2 that a data product incorporates to be considered useful. These characteristics apply to all data products, regardless of their domain or archetype. We call these baseline data product usability attributes. Every data product incorporates these characteristics to be part of the mesh. These are an addition to what has been known as **FAIR** data in the past—data that meets the principles of **D**indability, **A**ccessibility, **I**nteroperability, and **R**eusability. [8]

The introduction of analytical data as a product adds to the list of existing responsibilities of cross-functional domain teams [9] and expands their roles to:

- Data product developer: The role responsible for developing, serving, and maintaining the domain's data products as long as the data products live and are being used. They will be working alongside their fellow application developers in the domain.
- Data product owner: The role accountable for the success of a domain's data products in delivering value, satisfying and growing the data users, and maintaining the life cycle of the data products. He assure continuity of ownership of data and accountability

of success metrics such as data quality, decreased lead time of data consumption, and in general data user satisfaction through net promoter score.

Define these roles for each domain and allocate one or multiple people to the roles depending on the complexity of the domain and the number of its data products. Moreover, to make a data as a product, we need to satisfy these conditions:

- Reframe the Nomenclature: Data mesh suggests reframing receiving upstream data from ingestion to consumption. The minor distinction is that the upstream data has already been cleansed, processed, and is ready for consumption.
- Think of Data as a Product, not a mere asset.
- Establish a Trust-But-Verify Data Culture: The data as a product principle entails a variety of actions that contribute to a culture in which data users can trust the veracity of the data and focus on proving its suitability for their use cases. Data-as-a-product practices strive to create a new culture, moving away from presumption of guilt.
- Join Data and Compute as One Logical Unit: Coexistence of data and code is not a novel concept. The expansion of operational systems has resulted in a model in which each service handles its own code and data, as well as schema definition and upgrades. The link between the code and its data distinguishes an operational system.

2.1.3 Principle of the Self-Serve Data Platform

The principle comes to the rescue in order to reduce the cognitive load imposed on the existing domain teams by the other two principles: own your analytical data and distribute it as a product.

It shares several of the same features as existing platforms, such as access to polyglot storage, data processing engines, query engines, streaming, and so on. It differs from previous platforms, however, in its users: autonomous domain teams comprised mostly of generalist technologists. It maintains a higher-level data product architecture that encapsulates data, metadata, code, and policy as a single entity.

Its goal is to empower domain teams by hiding low-level complexity behind simpler abstractions and minimizing friction from their journeys toward their goal of exchanging data products as a unit of value. It prefers decentralized, interoperable methods to expand out data exchange beyond a particular deployment environment or organizational unit.

2.1.4 Principle of Federated Computational Governance

The response to the questions about the data mesh governance model's change from centralized to decentralized governance, known as federated computational governance. Data mesh, like lake and warehouse, serves a similar set of governance goals. However, it differs in its operating strategy and how these goals are attained.

The data mesh governance model consists of three complementary pillars:

- **System thinking:** Consider the mesh to be an ecosystem of interconnected data product and platform systems, with independent and yet linked teams. Then, try to locate the leverage points and feedback loops that will allow you to manage the behavior of

the mesh as a whole toward the goal of creating value by exchanging data products at scale.

- **Apply a federated operating model:** Create a federated team of individual domains and platform representatives from a social and organizational standpoint. Create incentives that are aligned with the performance of both domains' data products and the overall success of the ecosystem. Allow domains autonomy and accountability for the vast majority of policies within their influence and control, while leaving cross-functional and a small number of policies to be developed worldwide.
- **Embed the governance policies** into each data product in an automated and computational fashion.

To bring these three pillars together, Figure 2.3 shows an example of this model.

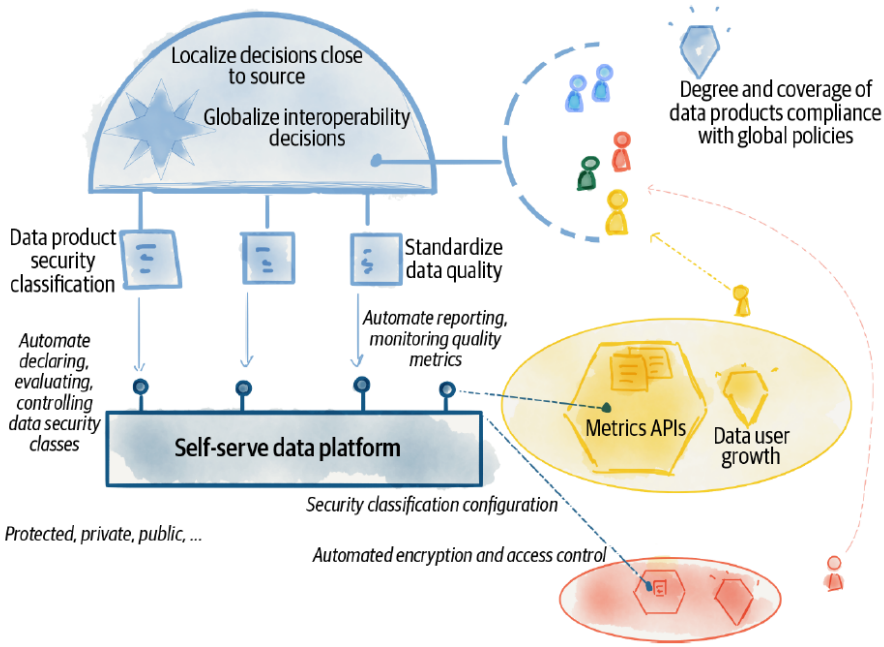

Figure 2.3: Example of data mesh governance operating model

Data mesh governance seeks to improve the existing approach to data governance at the intersection of a decentralized value system meeting automation and computation.

Data mesh governance is the method by which we define and apply what is right and how to do the right thing in an ecosystem of data product teams. It defines an approach that attempts to continuously find the dynamic equilibrium between localization of decisions so that the domains can go fast versus globalization and centralization of decisions for everyone to go far.

However, the mesh will not reach an optimal state unless the majority of domains become intelligently augmented - with embedded ML-based systems in each of their products and systems. The continuous need for trustworthy and useful data across multiple domains to train ML-based solutions will be the ultimate motivator for the adoption of data mesh governance and doing the right thing.

2.2 WHY WE NEED DATA MESH?

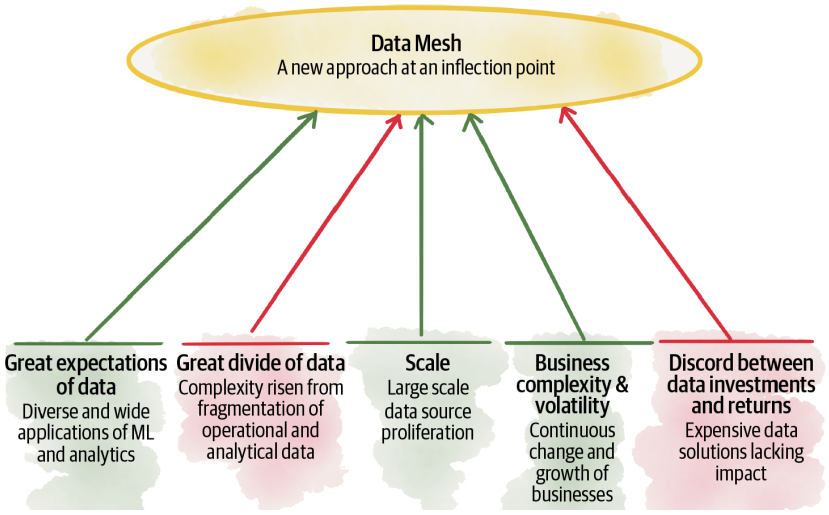

Figure 2.4: Macro drivers for the creation of data mesh

Data mesh is what comes after an inflection point, shifting our approach, attitude, and technology toward data. Data mesh assumes a new default starting state: proliferation of data origins within and beyond organizations' boundaries, on one or across multiple cloud platforms.

After the inflection point, data mesh make us reimagine data, specifically how to design solutions to manage it, how to govern it, and how to structure our teams: [10]

Table 2.1: Summary of after the inflection point with data mesh

Goal	What to do	How to do it
Manage changes to data gracefully in a complex, volatile, and uncertain business environment	Align business, tech, and data	Create cross-functional business, tech, and data teams each responsible for long-term ownership of their data *Principle of domain data ownership*
	Close the gap between the operational and analytical data planes	Remove organization-wide pipelines and the two-plane data architecture Integrate applications and data products more closely through dumb pipes *Principle of data as a product*
	Localize data changes to business domains	Localize maintenance and ownership of data products in their specific domains Create clear contracts between domain-oriented data products to reduce impact of change *Principle of data as a product*

continued on next page –

Table 2.1 – *continued from previous page*

Goal	What to do	How to do it
	Reduce the accidental complexity of pipelines and copying of data	Breakdown pipelines, move the necessary transformation logic into the corresponding data products, and abstract them as an internal implementation *Principle of data as a product* *Data product quantum architectural component*
Sustain agility in the face of growth	Remove centralized architectural bottlenecks	Remove centralized data warehouses and data lakes Enable peer-to-peer data sharing of data products through their data interfaces *Principle of domain ownership* *Principle of data as a product*
	Reduce the coordination of data pipelines	Move from a top-level functional decomposition of pipeline architecture to a domain-oriented decomposition of architecture Introduce explicit data contracts between domain-oriented data products. *Principle of domain ownership* *Principle of data as a product*
	Reduce coordination of data governance	Delegate governance responsibilities to autonomous domains and their data product owners Automate governance policies as code embedded and verified by each data product quantum *Principle of federated computational governance*
	Enable team autonomy	Give domain teams autonomy in moving fast independently. Balance team autonomy with computational standards to create interoperability and a globally consistent experience of the mesh. Provide domain-agnostic infrastructure capabilities in a self-serve manner to give domain teams autonomy. *Principle of federated computational governance* *Principle of the self-serve data platform*
Increase value from data over cost	Abstract complexity with a data platform	Create a data-developer-centric and a data-user-centric infrastructure to remove friction and hidden costs in data development and use journeys Define open and standard interfaces for data products to reduce vendor integration complexity *Principle of data as a product* *Principle of the self-serve data platform*
	Embed product thinking everywhere	Focus and measure success based on data user and developer happiness Treat both data and the data platform as a product *Principle of the self-serve data platform* *Principle of data as a product*

continued on next page –

Table 2.1 – *continued from previous page*

Goal	What to do	How to do it
	Go beyond the boundaries of an organization	Share data across physical and logical boundaries of platforms and organizations with standard and internet-based data sharing contracts across data products *Principle of data as a product* *Principle of the self-serve data platform*

While the evolution of data architecture has been necessary and an improvement, all of the existing analytical data architectures share a common set of characteristics that inhibit them from scaling organizationally. They are all monolithic with centralized ownership and technically partitioned. It is no longer valid when data gets sources from hundreds of microservices and millions of devices from within and outside of enterprises.

2.3 Data Mesh Architecture Design

2.3.1 The Logical Architecture

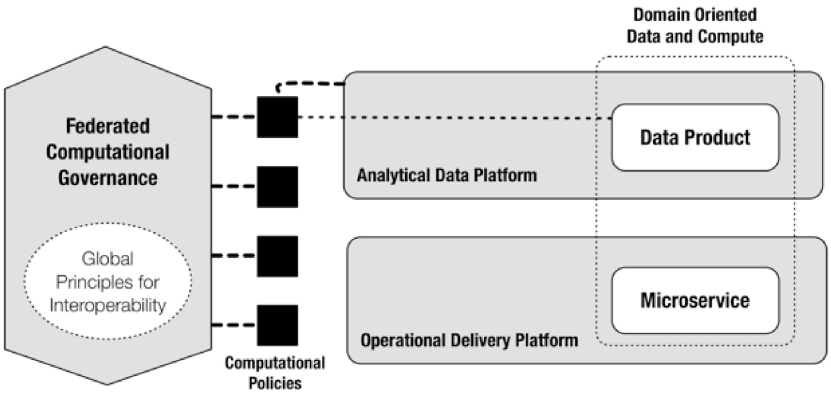

Figure 2.5: Data Mesh Logical Architecture [3]

Table 2.2: Data mesh logical architectural components and their maturity

Aspect	Architectural component	Description
Domain-Oriented Data Sharing Interfaces	Domain	Systems, data products, and a cross-functional team aligned to serve a business domain function and outcomes and share its analytical and operational capabilities with the wider business and customers. *This is a well-established concept.*

continued on next page –

Table 2.2 – *continued from previous page*

Aspect	Architectural component	Description
	Domain analytical data interfaces	Standardized interfaces that discover, access, and share domain-oriented data products. *Up to now, the implementation of these APIs is custom or platform specific.* The proprietary platforms need to offer open interfaces to make data sharing more convenient and interoperable with other hosting platforms.
	Domain operational interfaces	APIs and applications through which a business domain shares its transactional capabilities and state with the wider organization. *This concept has mature implementations.* It is supported by de facto standards such as REST, GraphQL, gRPC, etc.
Data product quantum	Data product structural components	Data product implemented as an architecture quantum that encapsulates all the structural components it needs to do its job—code, data, infrastructure specifications, and policies[3]. It is referred to in architectural discussions. It is used interchangeably with data products. *At the time of writing this is an experimental concept with custom implementations.*
	Data Product Data Sharing Interactions	There are many technologies for implementing a data product's input and output data port of data mesh, but it shares the common properties: • **Input data port:** A data product's mechanisms to continuously receive data from one or multiple upstream sources. *At the time of writing this has custom implementations with existing event streaming and pipeline management technologies.* • **Output data port:** A data product's standardized APIs to continuously share data. *At the time of writing this has vendor-specific custom implementations.* *A mature implementation of the concept requires open data sharing standards with support for multiple modes of access to temporal data.*

continued on next page –

Table 2.2 – *continued from previous page*

Aspect	Architectural component	Description
	Discovery and observability APIs	A data product's standard APIs to provide discoverability information - to find, address, learn, and explore a data product - and observability information such as lineage, metrics, logs, etc. *At the time of writing custom implementations of these APIs have been built.* *A mature implementation requires open standards for discoverability and observabilityinformation modeling and sharing. Some standards[4] are currently under development.*
The Multiplane Data Platform	Platform plane	A group of self-serve platform capabilities with high functional cohesion surfaced through APIs. *This is a general concept and well established.*
	Data infrastructure (utility) plane	Platform plane providing low-level infrastructure resource management - compute, storage, identity, etc. *At the time of writing the services that constitute the infrastructure plane are mature and provided by many vendors with support for automated provisioning.*
	Data product experience plane	Platform plan providing operations on a data product. *At the time of writing, custom implementation of services constituting a data product experience plane has been implemented. No reference implementation publicly exists.*
	Mesh experience	Platform plane providing operations on the mesh of connected data products. *At the time of writing, custom implementations of some of the services constituting a mesh experience plane, such as discovery and search services, have been implemented. No reference implementation publicly exists.*
Embedded Computational Policies	Data product container	A mechanism to bundle all the structural components of a data product, deployed and run as a single unit with its sidecar. *At the time of writing this is an experimental concept with custom implementations.*
	Data product sidecar	The accompanying process to the data product. It runs with the context of a data product container and implements cross-functional and standardized behaviors such as global policy execution. *At the time of writing this is an experimental concept with custom implementations.*
	Control port	A data product's standard APIs to configure policies or perform highly privileged governance operations. *At the time of writing this concept is experimental.*

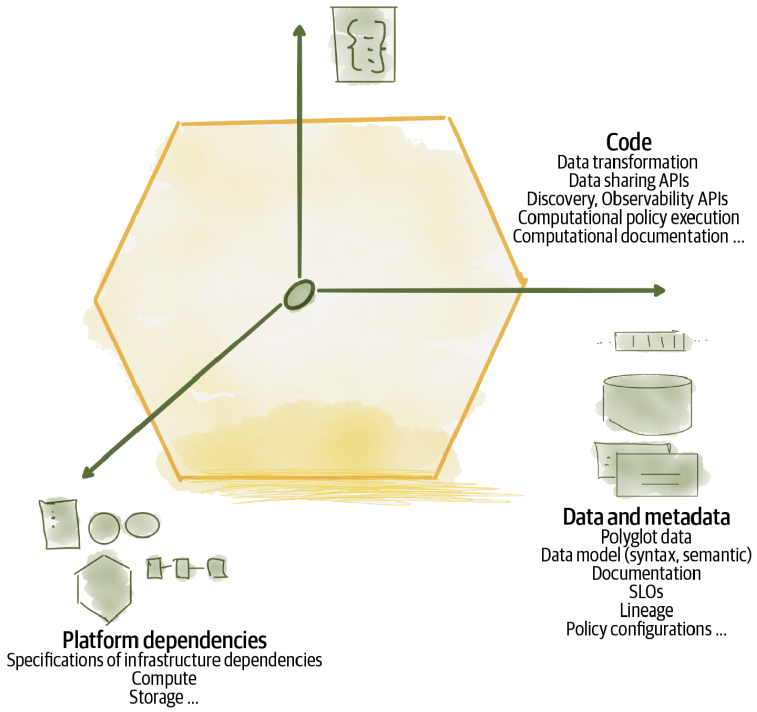

Figure 2.6: Types of structural elements of a data product

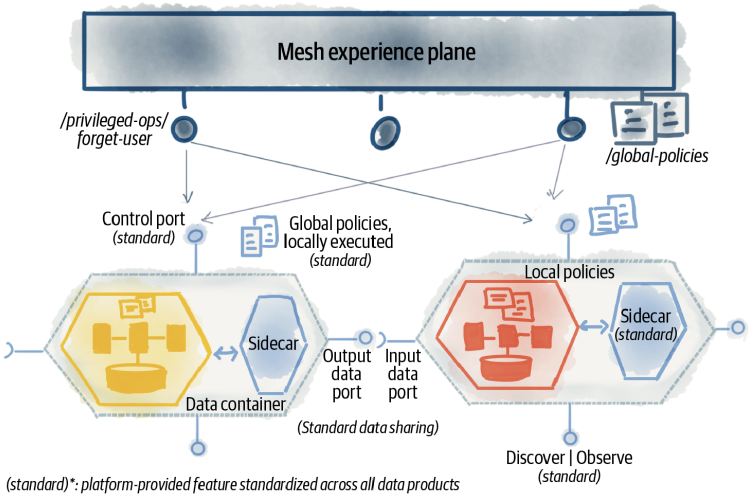

Figure 2.7: The logical architectural components of embedded computational policies

It is intentionally that the technology evolves to the point that we can get the logical architecture and its physical implementation as close to each other as possible. [11]

2.3.2 The Multiplane Data Platform Architecture

As we have discussed previously, the multiplane data platform consists of three planes, with the interactive diagram as figure 2.8.

[3]See figure 2.6
[4]OpenLineage is an example of an observability open standard that data products can adopt.

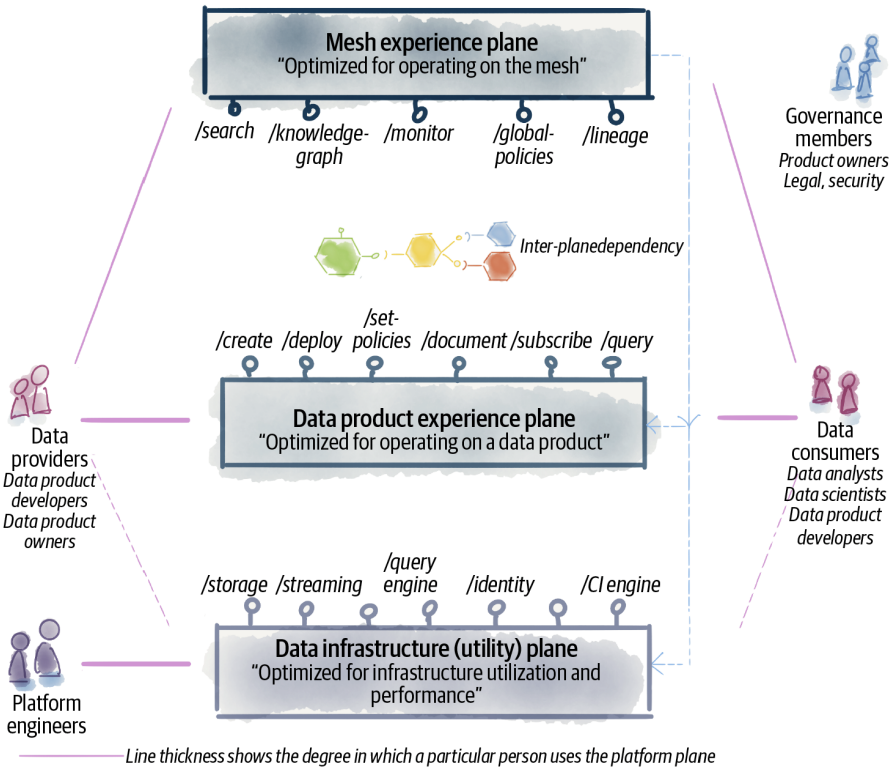

Figure 2.8: Multiplane self-serve platform and platform users

The data product experience plane is used by data product consumers and providers to discover, learn, understand, consume, build, and maintain data products. It provides a set of operations on a data product and manages the complexity of provisioning its underlying infrastructure. The data infrastructure utility plane optimizes the resources' performance and utilization to get the best out of what the underlying infrastructure providers offer. It is organized around the underlying resources and its users, with many of the data infrastructure plane services shared with the operational systems.

The most important idea is to understand the main journeys of the platform users and evaluate how to make it easy for them to complete their journeys. There are a few main high-level personas in the data mesh ecosystem, but we only discussed about journey of a data product developer and a data scientist as a data consumer.

Data Product Developer Journey

Creating and operating a data product is one of the most important journeys of a data product developer. This is a comprehensive and long-term obligation that adheres to the continuous delivery premise of "everyone is responsible." A data product development journey interacts with other journeys. Let's look at the figure 2.9 for an example of high-level stages of data product development and how the platform interfaces are designed to support them.

We should go deeper to consider each stage of the platform planes and interfaces

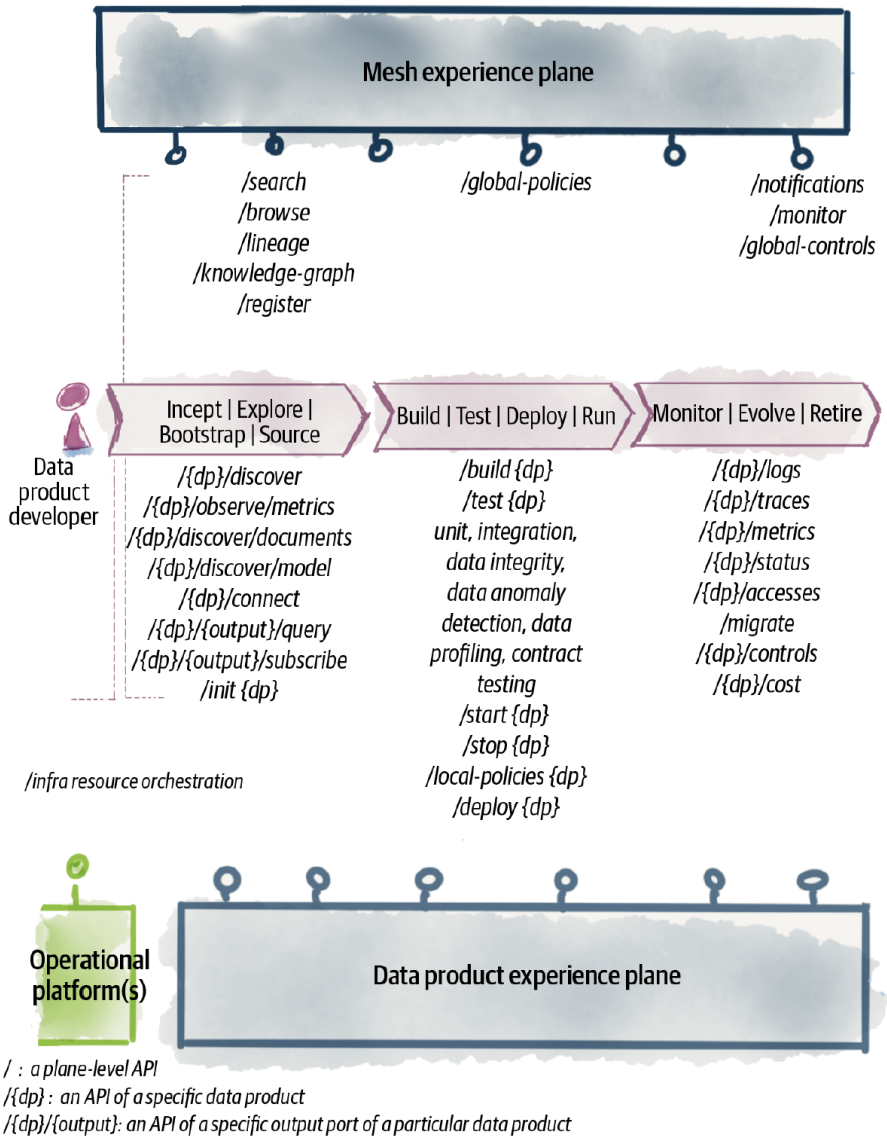

Figure 2.9: High-level example of data product development journey using the platform

Table 2.3: Example interfaces provided by the platform planes

Phase	Platform plane	Platform interface	Platform interface description
Data product inception			
Incept | Explore	Mesh experience	`/search`	Search the mesh of existing data products to find suitable sources. The search can be based on various parameters such as source operational systems, domains, and types of data.
		`/knowledge -graph`	Browse the mesh of related data products' semantic models. Traverse their semantic relationship to identify the desired sources of data.

continued on next page –

Table 2.3 – *continued from previous page*

Phase	Platform plane	Platform interface	Platform interface description
		`/lineage`	Traverse the lineage of input-output data between different data products on the mesh to identify the desired sources based on the origin of the data and the transformations the data has gone through.
Incept | Explore	Mesh experience	`/search`	Search the mesh of existing data products to find suitable sources. The search can be based on various parameters such as source operational systems, domains, and types of data.
		`/knowledge -graph`	Browse the mesh of related data products' semantic models. Traverse their semantic relationship to identify the desired sources of data.
		`/lineage`	Traverse the lineage of input-output data between different data products on the mesh to identify the desired sources based on the origin of the data and the transformations the data has gone through.

Chapter 3

DATA PRODUCT DESIGN & IMPLEMENTATION

Chapter 4

DATA MESH IN USE

4.1 DATA MESH IN COMBINATION WITH DATA LAKEHOUSE

4.2 FRAMEWORKS AND TECHNOLOGIES FOR DATA MESH

4.3 CASE STUDY AND DEMO

REFERENCES

[1] Z. Dehghani, "Prologue: Imagine data mesh," in *Data Mesh: Delivering Data-Driven Value at Scale*. O'Reilly Media, 2022, pp. xxv–xxxviii.

[2] C. Jochen, V. Larysa, and S. Harrer. (2023) Data mesh from an engineering perspective. [Online]. Available: https://www.datamesh-architecture.com/

[3] I. A. Machado, C. Costa, and M. Y. Santos, "Data mesh: Concepts and principles of a paradigm shift in data architectures," *Procedia Computer Science*, vol. 196, pp. 263–271, 2022.

[4] D. Nicolas. (2023) Will the buzz around data mesh really help solve my data challenges? [Online]. Available: https://kpmg.com/be/en/home/insights/2023/03/lh-the-impact-of-data-mesh-on-organizational-data.html

[5] Z. Dehghani, "Data mesh in a nutshell," in *Data Mesh: Delivering Data-Driven Value at Scale*. O'Reilly Media, 2022, pp. 3–14.

[6] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional, 2003.

[7] Z. Dehghani, "Principle of domain ownership," in *Data Mesh: Delivering Data-Driven Value at Scale*. O'Reilly Media, 2022, pp. 15–28.

[8] M. D. Wilkinson *et al.*, "The fair guiding principles for scientific data management and stewardship," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[9] Z. Dehghani, "Principle of data as a product," in *Data Mesh: Delivering Data-Driven Value at Scale*. O'Reilly Media, 2022, pp. 29–46.

[10] ——, "Why data mesh?" in *Data Mesh: Delivering Data-Driven Value at Scale*. O'Reilly Media, 2022, pp. 93–137.

[11] ——, "How to design the data mesh architecture," in *Data Mesh: Delivering Data-Driven Value at Scale*. O'Reilly Media, 2022, pp. 139–190.