future internet

MDPI

Article

Implementing Federated Governance in Data Mesh Architecture †

Anton Dolhopolov, Arnaud Castelltort and Anne Laurent *

LIRMM, University of Montpellier, CNRS, 34095 Montpellier, France; anton.dolhopolov@lirmm.fr (A.D.); arnaud.castelltort@lirmm.fr (A.C.)

* Correspondence: anne.laurent@umontpellier.fr

† This article is an extended version of our paper published in 5th Congress on Blockchain and Applications.

Abstract: Analytical data platforms have been used for decades to improve organizational performance. Starting from the data warehouses used primarily for structured data processing, through the data lakes oriented for raw data storage and post-hoc data analyses, to the data lakehouses—a combination of raw storage and business intelligence pre-processing for improving the platform's efficacy. But in recent years, a new architecture called Data Mesh has emerged. The main promise of this architecture is to remove the barriers between operational and analytical teams in order to boost the overall value extraction from the big data. A number of attempts have been made to formalize and implement it in existing projects. Although being defined as a socio-technical paradigm, data mesh still lacks the technology support to enable its widespread adoption. To overcome this limitation, we propose a new view of the platform requirements alongside the formal governance definition that we believe can help in the successful adoption of the data mesh. It is based on fundamental aspects such as decentralized data domains and federated computational governance. In addition, we also present a blockchain-based implementation of a mesh platform as a practical validation of our theoretical proposal. Overall, this article demonstrates a novel research direction for information system decentralization technologies.

Keywords: decentralized systems; federated governance; metadata management; data mesh; blockchain

Citation: Dolhopolov, A.; Castelltort, A.; Laurent, A. Implementing Federated Governance in Data Mesh Architecture. *Future Internet* **2024**, *16*, 115. https://doi.org/10.3390/fi16040115

Academic Editors: Javier Prieto and Ricardo Alonso

Received: 17 February 2024
Revised: 17 March 2024
Accepted: 27 March 2024
Published: 29 March 2024

1. Introduction

The majority of modern organizations that deal with data processing are constantly facing the challenges of big data to some extent. These challenges include: volume, velocity, veracity, variety, and value of the processed data [1]. To cope with these, we distinguish two important aspects used by organizations, namely *data platforms* and *organizational structure*. The platform serves the utility purpose, which is facilitating operations over data assets. Meanwhile, the org structure defines the user's responsibilities, roles, and rights needed to perform the necessary operations. In this article, we focus specifically on the data platform aspect that is required for implementing the decentralized platform architecture.

Data warehouses [2] are considered to be the first type of platform to deal with big data. Being oriented to managing the structured data, they provide tools for business schema enforcement on the collected data, user query processing engines, and analytical online data processing (OLAP). This kind of solution is widely applied today, but it may struggle with such big data requirements as velocity and variety of data.

To overcome this, *data lakes* were proposed as a second type of platform [3]. They implement a *schema-on-read* functionality, which is opposite to schema-on-write seen in the data warehouses. In the data warehouse architecture, incoming data (application telemetry, operational database changes, etc.) are transformed during the ingestion phase. In the data lake architecture, we keep the ingested data in their raw format, usually by using large object storage. The data processing is postponed until the querying phase, when the value extraction is performed by respective professionals, like data scientists. This facilitates the

speed (velocity) of batch and real-time acquisition directly from the data sources to the support of a variety of data types, including structured and unstructured information.

The synergy of these two platforms was introduced in [4]. *Data lakehouses* preserve the benefits of both solutions. On one side, analytical data pre-computation enables fast business queries (e.g., business intelligence cubes); on the other side, raw data storage enables the discovery of valuable data insights in the future.

Even with the strong support of distributed technologies, all these three kinds of platforms are designed and developed as centralized solutions from a logical point of view. The centralization comes from the fact that usually, data value extraction is performed by analytical teams that are detached from the operational activities of the company. Data scientists and engineers are gathered under a single data division, tasked with performance evaluation, and serve as a mediator between the business people and developers.

Nonetheless, with the growing popularity of decentralized systems in recent years, a new data platform architecture called Data Mesh [5] has emerged. Building on the lasting works around the domain-driven design [6] and data markets [7], it manifests four core principles:

- Distributed data domains;
- Data-as-a-product;
- Self-serve data infrastructure;
- Federated computational governance.

In contrast to the previous platform types that envision the centralization of the data ownership and value extractions under the command of a dedicated team, the mesh architecture aims to remove the segmentation of the holistic business domains across the operational and analytical teams.

The core component of any big data platform is a metadata management (MDM) system. This makes it possible to account for, link, comprehend, integrate, and control the existing data assets. MDM can be seen as an essential part of the federated governance [8]. From a high-level perspective, we can split the last data mesh principle into two modules:

1. Definition, access, and collaboration on schemes, semantic knowledge, lineage, etc.;
2. Tools to automatically enforce security, transparency, legal policies, etc.

We see that the first part (governance) is well-aligned with metadata management, while the second part (computability) represents the infrastructure platform functions.

However, the main challenge appears when building the federated computational governance with the existing products or research tools. The data mesh platforms either fall back to using the centralized systems, like Apache Atlas [9,10], cloud data catalogs [11,12]; or represent only a conceptual architecture of the federated MDM system [13–15].

In this article, we demonstrate the further development of our previous metadata system described in [16]. At first, we consider the problem domain in more detail by showcasing a virtual company running example. Then, we outline the MDM properties and challenges of the mesh platform governance. On the way to overcome the limitations, we show how the operating system level virtualization (e.g., Kubernetes) and blockchain technologies fulfill those requirements. As a final part, we present our platform prototype, which aims to implement federated computational governance. The article concludes with a system evaluation and future research directions.

2. Background

In this section, we recall the data mesh principles with more descriptive details and provide a running example of a virtual company to illustrate their application. We also walk through the existing related research and proceed with the requirements of data mesh governance.

2.1. Data Mesh Principles Explained

In 2019 (https://martinfowler.com/articles/data-monolith-to-mesh.html (accessed on 16 February 2024)), Zhamak Deghani introduced a novel approach for building organizationally scalable data platforms that are based on four principles: distributed data domains, data-as-a-product, self-serve data platform, and federated computational governance.

2.1.1. Distributed Data Domains

The data mesh takes its origin from the domain-driven design and service-oriented architecture (SOA) in particular. In SOA, the monolithic operational platform is broken up into independently deployable modules, like micro-services [17]. In the same way, we can divide the analytical platform into self-contained data domains. *Distributed data domains* constitute the first principle of a mesh. Instead of forcing centralized data ownership onto a single analytical team, we want to benefit from forming domain-aware groups of engineers.

2.1.2. Data-as-a-Product

The domain-driven methodology uses the notion of the software products [6]. In data mesh, this means that the culture of building the *data products* within the domains is required. Most of the time, the data users are not only the external customers, but other domain teams within a single organization too. Making a reusable data product requires designing, delivering, measuring metrics, and constantly improving the provided data [12]. As a consequence, it also results in a new zone of responsibility for data product owners: SLA guarantees, trustworthy data delivery, tools for data discovery, comprehension, consumption, etc.

2.1.3. Shared Data Infrastructure

Following the wisdom of a popular software pattern that states: "Don't Repeat Yourself", we should avoid building multiple data platforms in parallel by each distributed domain team. In reality, the mesh platform is required to provide a way to automate the provisioning and usage of identical or similar functionality like data acquisition, processing, and serving [5]. It is possible to achieve this with a *shared data infrastructure*. Its main benefits are similar to that of a *data fabric* [18], which is a set of common and interchangeable tools like data source connectors and sinks, transformation jobs, automatic logs, lifecycle configuration, etc. Eliminating the basic day-to-day engineering operations behind an easy-to-use platform interface helps to reduce production errors and enable a fast release cycle.

2.1.4. Federated Computational Data Governance

Governance centralization dictates the common practices that also enable the interoperability of the systems [10]. But it has organizational scaling issues. As was mentioned in the introduction, federated governance should provide the tools that help to collaborate and ease the definition and access to data schemes, semantic knowledge, and lineage, but also to automatically enforce security, transparency, and legal policies. Meanwhile, it also has to leave the necessary freedom to domain teams to ensure fast product evolution and development. The common policies must be automatically enforced by the infrastructure platform. At the same time, the committee of data domain representatives can define these rules via the appropriate tools.

Now, we illustrate the application of these principles with our running example.

2.2. Running Example

In Figure 1, we present a running example of a virtual e-commerce company adopted from [14] that comprises Human Resources, Suppliers, and Sales departmental domains.

In this example, the engineers within the Supplier's team are occupied with both transactional and analytical online processing. The domain output is polyglot data prod-

ucts, which are actual or derived data, like business intelligence reports with aggregated information on regional supply chain state or predicted raw material delivery delays.

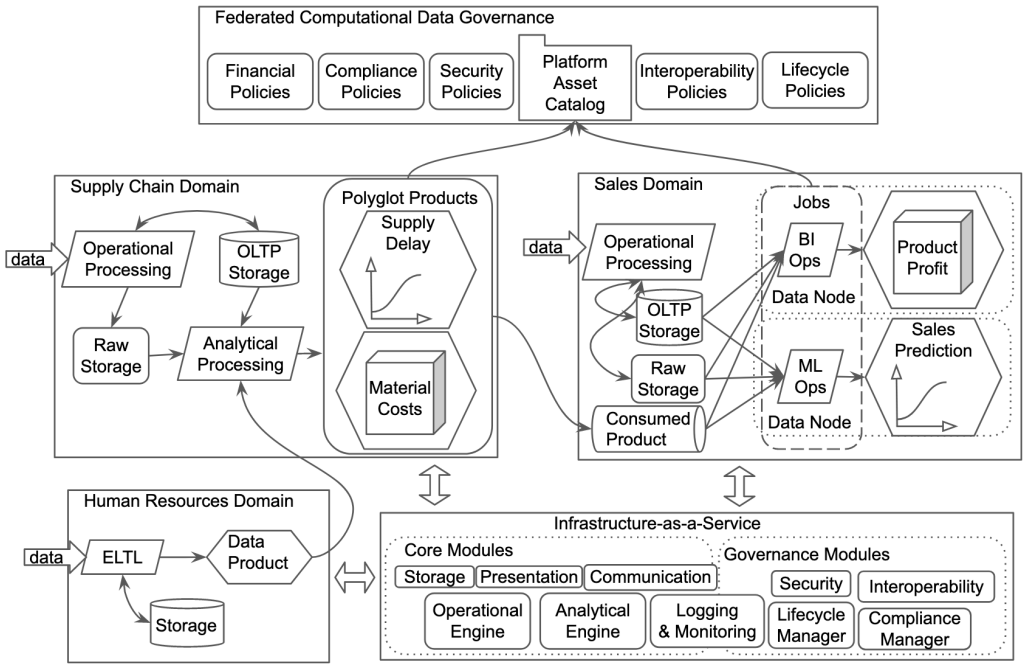

Figure 1. Data mesh running example architecture overview.

By consuming those reports, analysts of the Sales department will improve the accuracy of its own forecasts, like adjusting the expected revenue of the company.

The shared infrastructure platform provides a comfortable interface over the utilized computing resources. This helps to put in place the monitoring of the newly released reports and to automate the updates of the product prices according to the selected policies. In general, the common platform assures interoperability and helps to avoid over-provisioning of resources when working with different technological stacks.

The federated computational governance is an essential component for "maintaining a dynamic equilibrium between domain autonomy and global interoperability" [5]. Here, governance is a collection of policies, standards, objectives, metrics, and roles aimed to maximize the value of data, and computational property ensures its automatic enforcement.

Recalling the metadata management system as a core part of the governance, we also provide a specialized Platform Asset Catalog that enlists asset metadata such as ownership, purpose, history, dependencies, mode of use, etc. Since there are a lot of different product sources and configurable computing tasks that are produced by different teams, it is a good practice to register them in such a catalog.

2.3. Related Work

Since the introduction of the mesh concept, a number of industrial and academic implementations of this data platform have been proposed.

Machado et al. [19] discussed the first works from the industry regarding the transition to mesh architecture performed by the e-commerce platform Zalando and video-streaming service Netflix. The same authors also provided the first data mesh domain model [20] and presented a prototype system based on that model [9].

Wider et al. [10] extended this domain model with data product ports definition, provided an industrial perspective on enforcing the federated governance, particularly the access management to the sensitive data, and presented a cloud-based architecture using the Amazon Web Services (AWS).

Among other cloud-based works, we note Butte et Butte [12], AWS-based proposal with zonal data segmentation, and Goedegebuure et al. [11] who conducted the grey literature review on data mesh and also presented a Microsoft Azure-based architecture implementation for a large producer of lithography machines.

Hooshmand et al. [13] presented a novel view of an automobile production industry with the application of semantic web technologies (SWTs) for ensuring the integration and alignment of distributed domains, while Driessen et al. [21] described the usage of SWT specifically for data product metadata management.

Nonetheless, as we shall see in Section 6, the existing platforms for building the data mesh still do not satisfy all requirements. We provide these requirements in the next section.

3. Data Mesh Governance: Requirements and Challenges

This section outlines the requirements and challenges that we used to create a functionality maturity table of the related works that our system aims to fulfill.

3.1. Properties of Data Governance System

Dolhopolov et al. [16] consider properties required for the data mesh platform. Here, we provide detailed information on each and extend the list with two additional elements, specifically with *contract management* and *product compositionality*. These properties are based on the years-long research of data platforms, and data lakes in particular [22], principles of the domain-driven design, and data mesh concepts.

1. **Semantic Enrichment (SE)** is related to providing the descriptive information that defines the meaning of the underlying data and facilitates product comprehension and utilization [22]. On the basic level, the use of semantic tagging helps in relating different data sources, e.g., 'Financial' data, 'Medical' data, etc. On a more advanced level, the use of semantic web technologies and knowledge graphs can be implemented [21]. This can provide automatic product descriptions, data linking, relationship discovery, etc.
2. **Data Indexing (DI)** is a core element for efficient metadata querying and navigation. Almost all modern data storage providers have indexing functionality that enables fast data look-ups. In the data mesh case, the metadata system should provide the search capabilities to retrieve the data product information based on a diverse set of user requests: keywords, history, ownership, usage pattern, etc.
3. **Link Generation (LG)** helps to identify and integrate the related information [22]. For instance, the use of identical semantic tags helps to cluster the products into different groups. Data lineage establishes the tracing history of the upstream dependencies and enables the user alert or automatic product suspension in case of breaking changes. Product linking can also consist of similarity measurements, which is a further extension of product clustering with a relevance degree.
4. **Data Polymorphism (DP)** or polyglot data is related to the operation and management of multiple product facets [5]. Since in data mesh the data consumption is conducted via connecting ports, the same product can have multiple consumers as well as multiple forms. For instance, real-time analytics may require the streaming data interface while monthly reports do not need much time precision. Moreover, the data can have variable degrees of quality, representation (tabular data, vector), maturity, etc.
5. **Data Versioning (DV)** describes the management of metadata updates within the system while retaining the previous metadata states. It is very relevant since it ensures the system state recovery, reproducibility, and auditing. Moreover, versioning allows branching, enables backward compatibility, and parallel data product evolution [22].
6. **Usage Tracking (UT)** consists of providing and tracking access to the underlying data resources. It is a backbone for implementing the security and protection of data. Data security comprises multiple elements, such as information and communication encryption, firewall, etc [23]. In fact, in the aspect of metadata management, it is

important to keep records of product access, user identities, usage patterns, locations, time schedules, etc., in order to detect and prevent unauthorized activities.

7. **Computational Policies (CP)** play a vital role in automatic governance execution. Beyond the access control enforcement, it also enables data quality verification, consistency, uniqueness, lifecycle management, service contract tests, etc [5]. This reflects the need to define the rules on each level—global and local, which are then applied in the mesh. Such governance execution also requires an appropriate platform infrastructure as well [10].
8. In the context of micro-service architecture, **Independently Deployable (ID)** products provide the opportunity to update the performed services without the overall system interruption (e.g., canary deployment) [17]. In the context of data mesh, it means an option to deploy the new data products without affecting the established data consumption. This requirement is also applied to metadata registration and policy updates. Ideally, the new information and rules should not interrupt the existing data flows unless it is specifically intended by the domain owners.
9. **Automatically Testable (AT)** platform design ensures the correctness of the future system state upon the module's upgrade. For instance, when implementing new data resolution modules, e.g., transition from IPv4 to IPv6, the address space and links of the old resources should continue to work. To be sure that the introduction of a new module will not break the operations of the system, we are obligated to perform the automatic tests and verification of the system, assuming that the upgrades have taken their place, while in reality keeping the old system functioning.
10. **Contract Management (CM)** provides a way to negotiate, participate, and ensure the correctness of the delivered data. Usually, the contract includes the outlined service level objectives and agreements, including the quality of data, schema, update frequency, intended purposes of use, etc. [24]. As a part of the platform governance, it overlaps with the metadata management and computational execution modules.
11. **Product Compositionality (PC)** helps to speed up product development and to prevent dataflow interruption. Automatic contract composition verification enables advanced interoperability features and helps to prevent unauthorized behavior, recovery of PII, etc. In cases of schema composition, it automatically enriches the existing products or prevents the introduction of the breaking changes.

These properties represent the major functional modules of the data mesh. Despite the fact that it was outlined in 2019, there are still challenges in building the technology that can underpin its implementation.

3.2. Challenges of Federated Data Governance

Data governance refers to a comprehensive framework encompassing various processes, roles, policies, standards, and metrics, all aimed at improving the extraction of business value from data while staying compliant. This framework should provide a platform where technical engineers, business stakeholders, legal representatives, and other involved parties can easily define business processes and regulatory policies, assign data access roles, and shape the platform's operations.

Centralized governance lays down a unified set of rules and standards for all teams, ensuring that data systems can work together smoothly. However, this approach faces scalability challenges, as was shown by the limitations identified in data warehouse and data lake systems [19]. On the other hand, decentralized governance offers greater autonomy to individual domain teams, which can lead to the creation of incompatible data products, redundant development efforts, and overlooked compliance procedures.

Zhamak Deghani, the pioneer of the data mesh concept, underscores the necessity for federated computational data governance [5]. This approach seeks to balance domain autonomy with overall system interoperability, merging the strengths of both centralized and decentralized governance. Federated governance aims to facilitate collaboration and simplify the processes of defining and accessing data schemas, extending semantic

knowledge, and providing data lineage. It also strives to automate the enforcement of security, transparency, and legal standards. Nevertheless, the challenge lies in building such a system with the currently available tools and research.

In our introductory section, we noted that metadata management systems often serve the roles of data governance systems in research environments. Nonetheless, in the literature [25–29], and in the commercial products [30–32] we see the predominant adoption of a centralized governance model. This model involves gathering metadata in a central repository, where it is analyzed, interconnected retrospectively, and often made accessible to users through web portals.

Recent studies have explored the implementation of federated data governance using semantic web technologies [13,21], though these systems are still in their developing stages.

One of the approaches to making the federated governance, which is getting more attention in the data mesh community is "data contracts". Truong et al. [24] highlight the need for data contracts in managing the use, sharing, and transaction of data within cloud marketplaces. This provides a structured method to define the rights, obligations, and constraints associated with data usage, ensuring clarity and compliance across different parties. They address key challenges such as specifying permissible data use, establishing liability in case of data misuse or quality issues, and enabling automatic and efficient data exchange and utilization. Furthermore, data contracts contribute to the scalability of data marketplaces by automating compliance and governance processes. By formalizing these aspects, we facilitate a trustworthy and regulated environment for data sharing, promoting innovation and value creation within data ecosystems.

Consequently, there is a significant research gap in the design and development of federated governance systems, signaling a pressing need for advancement in this field. In the next section, we attempt to provide the formal model for building the federated metadata management system while using the data contract approach.

4. Introducing Blockchain-Powered Mesh Platform

In Section 3, we described the data mesh properties and the challenges associated with building efficient data governance. In this section, we make our first research contribution by extending three formal types of metadata systems that were previously defined in [14]. Afterward, we proceed with a description of how Hyperledger Fabric and Fybrik platforms can benefit from the implementation of an efficient federated metadata catalog.

4.1. Defining the Data Mesh Governance Types

On one side, centralized governance dictates how each domain should function, which defeats the goals of data mesh in the first place. On the other side, interoperability is a big challenge of complete decentralization. In Figure 2, we define the data mesh governance scale that helps to understand better how we can configure and build our metadata catalog. The horizontal axis shows the proposed deployment modes ranging from centralization (left) to decentralization (right).

4.1.1. Type I—Centralized Metadata Repository

The centralized repository implies that a set of all metadata records $m_1, \ldots m_n = M$ is stored in a single system $\mathcal{R}$. Upon the release of a data product p_x, the new metadata record m_x is pushed to the repository in order to make the product discoverable and accessible by other domains. Since data products can be consumed by multiple domains, we introduce the formal notion of a 'data contract' c that describes data product-to-product (DP2P) relations. To have control over the records, a repository operator can define the visibility map function $v : (u, m) \rightarrow \{1, 0\}$, and the access function $a : (u, m) \rightarrow \{Read, Write, Delete\}$ that are associated with user profiles U.

The metadata repository is defined as $\mathcal{R} = \{\mathcal{P}, \mathcal{M}, C, U, a_{u,m}, c_{p,p}, d_{p,m}, v_{u,m}\}$, where:

- $\mathcal{P} = \{P_1, P_2 \ldots\}$ is a set of data products and $P_i = \{p_1, p_2 \ldots\}$ is a set of product versions

- $\mathcal{M} = \{M_1, M_2 \ldots\}$ is a set of metadata records and $M_i = \{m_1, m_2 \ldots\}$ is a set of metadata versions
- $C = \{c_1, c_2 \ldots\}$ is a set of data contracts
- $U = \{u_1, u_2 \ldots\}$ is a set of system users
- $a_{u,m} : U \times \mathcal{M} \rightarrow \{Read, Write, Delete\}$ is a function returning a permissions map for any given pair (u, m)
- $c_{p,p} : \mathcal{P} \times \mathcal{P} \rightarrow C \cup \emptyset$ is a function returning the contract for any given pair (p_i, p_j) of data products or an empty set if it does not exist
- $d_{p,m} : \mathcal{P} \rightarrow \mathcal{M}$ is a function returning the metadata description of a data product
- $v_{u,m} : U \times \mathcal{M} \rightarrow \{1, 0\}$ is a function returning a visibility map for any given pair (u, m) of user and metadata

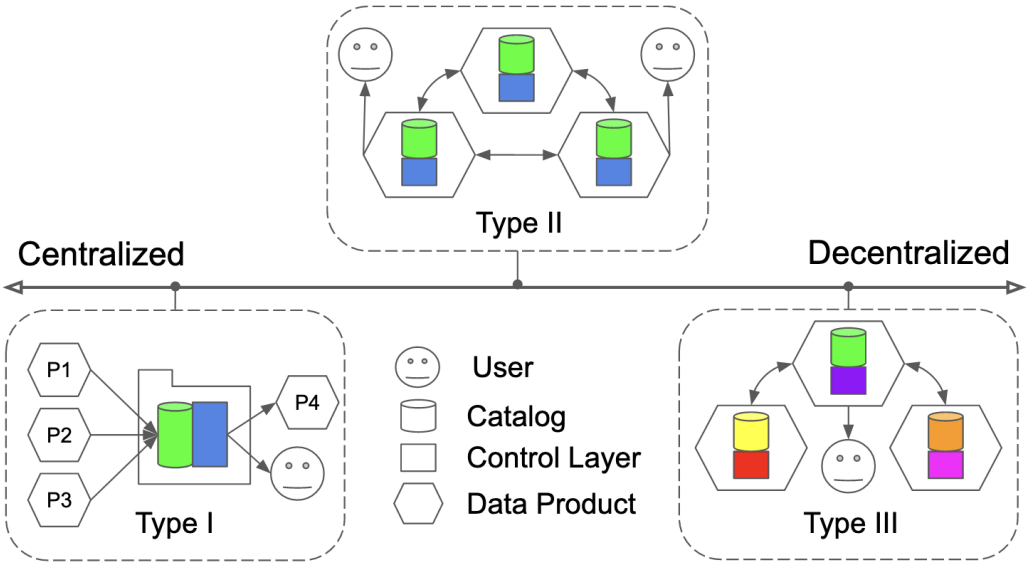

Figure 2. Data Mesh Governance Scale.

4.1.2. Type II—Federated Metadata Repository

The middle state between Type I and Type III is a federated system. There is no central repository and each domain team hosts a complete copy r of the metadata system on a dedicated metadata node n. Meanwhile, there must be a peer-to-peer (P2P) system that keeps the data in sync across all nodes. The system should also dispose of the methods or techniques T used for enforcing the established data contracts or the unified behavior B globally defined to be followed by all participants.

In total, we define the federated repository $\mathcal{R} = \{\mathcal{P}, \mathcal{M}, C, U, B, D, R, N, T, a_{u,m}, c_{p,p}, d_{p,m}, v_{u,m}, t_n, s_{r,r}\}$, where:

- $B = \{b_1, b_2 \ldots\}$ is a set of unified governing policies
- $N = \{n_1, n_2 \ldots\}$ is a set of metadata nodes that host the replicas
- $R = \{r_1, r_2 \ldots\}$ is a set of metadata repository replicas
- $T = \{t_1, t_2 \ldots\}$ is a set of methods that enforce the policies or contracts
- $t_n : N \times T \rightarrow B \cup C$ is a function returning a global policy b or a data contract c that is being enforce by the method t on a given node n
- $s_{r,r} : R \times R \rightarrow \{1, 0\}$ is a function returning the consistency or synchronization state for any two given replicas.

4.1.3. Type III—Decentralized Metadata Repository

The decentralized system is the case where each domain team (or a subset of domains) uses different technologies and policies for managing its metadata repositories. Each team independently owns and serves the (meta)data products, and in order to provide discoverability and querying one must cross-link the metadata with other domains. It

can be the case that n given domains can configure a shared federated metadata system, but also have their own private systems in parallel.

Therefore, the decentralized metadata repository Δ is defined as:

- $\Delta = \{\mathcal{R}_1, \mathcal{R}_2 \ldots\}$ with $\mathcal{R}_i$ being the repository associated with $D_i \subset \mathcal{D}$ and $D_i \neq \emptyset$
- $\mathcal{D} = \{d_1, d_2 \ldots\}$ is a set of all data domains
- $l_{i,j} : \mathcal{M}_i \times \mathcal{M}_j \rightarrow \{1, 0\}$ is a function establishing the link presence or absence between a pair of metadata records (m_i, m_j) that belong to different repositories $\mathcal{R}_i$ and $\mathcal{R}_j$, respectively.

4.2. Using Blockchain for Policy and Metadata Management

Blockchain technology offers multiple features that are well-suited for building a Type II metadata catalog. Its immutable, append-only ledger ensures that metadata records $\mathcal{M}$ are stored permanently, providing a version history for each record. The hashing function of blocks introduces a layer of trust crucial for verification and audit processes. In a blockchain network, each node maintains a copy of the ledger that contains all metadata records. This is analogous to nodes N deploying replicas R, ensuring redundancy and availability.

Furthermore, the EVM-like ledgers with smart contract support enable the automated enforcement of consistent, globally defined governance policies, represented by a set of governing methods T. In fact, a similar approach has already been proposed for building the data sharing systems [33].

We argue that it is also possible to adopt a smart contract as a policy approach for promoting data product-sharing practices based on executable data contracts. Upon the signature of the data contract between the data provider and data consumer, the data mesh governance system will automatically generate and register the agreement in the form of a new "smart policy" by using a contract factory pattern. Then, the processing system can pro-actively verify the state of this agreement and perform the necessary actions if the breach is detected [34].

The distributed consensus algorithm ensures the ledger remains synchronized across all replicas, acting as a consistency function s. This mechanism also facilitates the seamless adaptation and automatic testing of policy or contract updated in response to evolving business processes.

The distinction between public and private blockchains is critical, with public blockchains operating anonymously and treating all participants equally. This openness, however, makes them more susceptible to malicious acts. Private blockchains, like Fabric, offer a more controlled environment by pre-identifying participants, which enhances security and simplifies access management. Such a setup is inherently more secure and suitable for applications requiring strict access controls and data privacy.

The Hyperledger Fabric (HLF) platform is an open-source blockchain initiative under the Linux Foundation [35]. Its flexible architecture supports the integration of diverse computational elements tailored to specific system needs, including customizable consensus protocols, identity verification services, and transaction endorsement policies. To enhance security measures, it also facilitates network partitioning through the use of channels and algorithms for private data sharing.

We elaborate on building the data mesh governance with HLF thereafter.

4.3. Advantages of Hyperledger Fabric

In data mesh, the ledger of HLF can act as the federated repository for metadata management [36], benefiting from the blockchain's immutable, secure, and distributed characteristics. The search efficiency is enhanced through a state database that indexes the most recent asset transactions. To maintain ledger consistency and manage network updates, the platform offers various consensus mechanisms tailored to different trust levels within the network, from Byzantine Fault Tolerance for highly secure, cross-organizational exchanges to Crash Fault Tolerant algorithms for more trusted environments.

Platform governance is managed through detailed policy mechanisms that are related to the network itself, its channels, or smart contracts. It outlines how network changes are agreed upon and implemented. This structured approach facilitates a collaborative and secure environment for managing consensus and implementing updates across the blockchain network.

HLF uses Membership Service Providers to authenticate the identity of network components—ranging from organizations to nodes and applications [35]. This authentication framework supports secure, private communications across distinct channels and enables the private exchange of data, ensuring that sensitive information is shared only with authorized parties.

Within the Fabric, the use of smart contracts helps to automate the enforcement of agreed-upon rules that involve specific assets, e.g., data sharing operations [37]. Smart contracts are part of a broader construct known as chaincode, which includes both the contracts and the policies governing their execution. Deployed as compact Docker or Kubernetes containers, chaincode units are the foundational software modules for transaction endorsement and policy enforcement. These policies can, for example, designate which network nodes are required to endorse or order a transaction, ensuring that execution aligns with agreed-upon standards.

Chaincode serves a dual purpose: it not only governs transactions on the ledger transparently and interoperably but also standardizes data modifications while continuously verifying metadata integrity. This capability is especially relevant for Type II catalog requirements, where maintaining data accuracy and trustworthiness is crucial. Chaincode execution can also trigger event notifications, allowing network participants to stay informed about ledger updates and newly available information.

HLF stands out for its developer-friendly approach to smart contract development, supporting popular programming languages like Java, Go, and JavaScript/TypeScript [38]. This flexibility contrasts with platforms like Ethereum and lowers the barrier to adoption, making it easier for developers to build and deploy chaincode on the Fabric network.

On the other hand, it lacks the "contract factory" support which is present in the Ethereum ecosystem [34]. By introducing such functionality, we could significantly enhance the capabilities of the system. For instance, by utilizing such a factory we could use the smart contract template and automatically create the computable data contract instances.

4.4. The Fybrik Data Platform

Fybrik (https://github.com/fybrik/fybrik (accessed on 16 February 2024)) is an open-source project aimed at enabling a secure and efficient way of managing, accessing, and governing data across hybrid and multi-cloud environments. It focuses on data privacy, compliance, and optimized data usage across the enterprise, leveraging Kubernetes for workload orchestration.

Fybrik's architecture is modular and utilizes custom resources for defining data assets, applications, and governance policies. Its design allows for pluggable components, making it possible to integrate various data stores, governance tools, and compute platforms [39]. The extension mechanism includes two modes: the definition of new modules—operational blocks for integrating with different data processing platforms, and plugins—control blocks for facilitating the platform behavior. The latter includes connectors for custom and metadata and policies engines.

As a Kubernetes-based platform, Fybrik is designed to be interoperable with a wide range of cloud services and data sources. It supports integration with existing data management and analysis tools, making it easier for organizations to adopt and extend their current infrastructure.

Fybrik integrates policy-driven data governance and compliance to ensure that data usage adheres to organizational and regulatory standards [40]. It allows organizations to define policies that control how and where data can be accessed and used, ensuring compliance with GDPR, CCPA, and other regulations. It provides a mechanism for secure

and controlled access to data, regardless of where the data resides—on-premises, in the cloud, or across multiple clouds. It uses fine-grained access control and data masking techniques to protect sensitive information.

Fybrik facilitates the movement and virtualization of data across different environments without the need to duplicate data. This capability supports efficient data management and reduces latency by bringing the computer closer to the data, thereby optimizing performance for data-intensive applications.

5. Implementing Federated Data Mesh Governance

In this section, we describe our prototype system which is based on the Hyperledger Fabric and Fybrik platforms mentioned before.

As follows, we use the Fybrik as a backbone for data processing and storage, while Fabric performs the role of a metadata and policy management tool.

5.1. System Architecture

In Figure 3, we provide a simplified view of the Fybrik platform. The core elements include the *FybrikApplication* for registering and running the data workloads; a *Jupiter Notebook*, which is used by users manipulating the data assets; the data catalog for listing, linking, and search of the existing data products; policy manager and policy enforcement point (PEP) for providing security and compliance; and storage that can be represented as any supported platform—S3, PostgreSQL, etc.

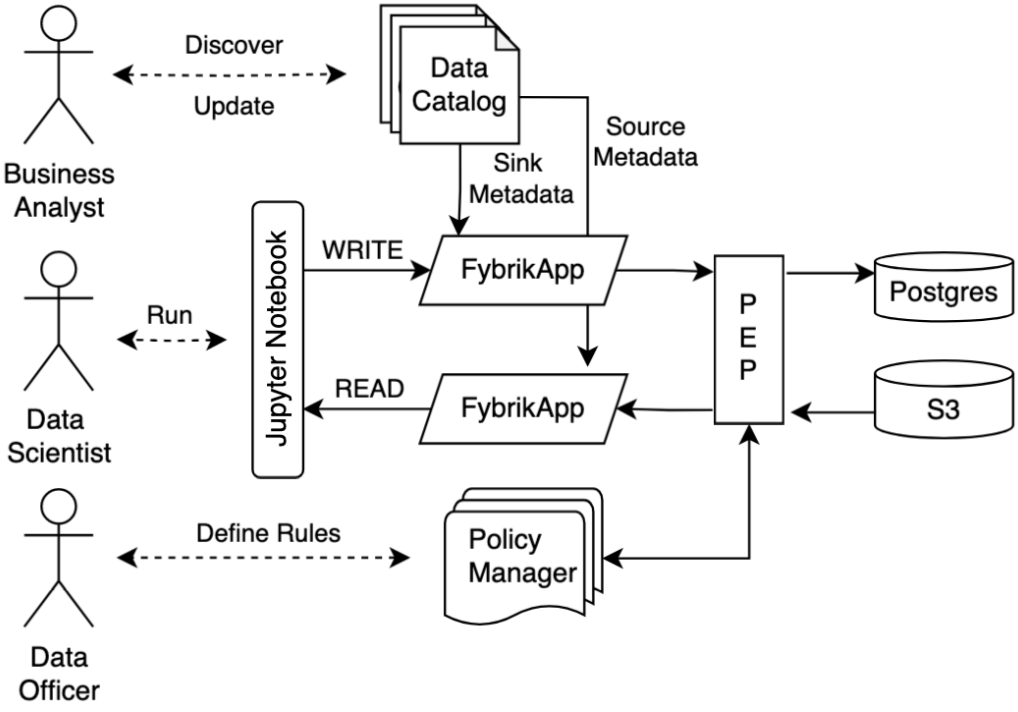

Figure 3. Fybrik architecture.

A FybrikApplication is a custom resource definition (CRD) submitted to the Kubernetes cluster where Fybrik is deployed. This CRD is required for defining how an application wants to access data, specifying the datasets it needs, the operations it intends to perform on the data, and any governance or compliance requirements that must be adhered to during data access and processing.

The Fybrik control plane interprets this CRD request and orchestrates the necessary governance mechanisms, data access paths, and computational workload placements to ensure that the application's data access is compliant with the specified policies. This process may involve interfacing with data catalogs for metadata management, deploying

policy enforcement points (such as data masking), and selecting the optimal data storage and computing locations to satisfy performance, cost, and compliance considerations.

Within Fybrik, the policy verification mechanism is based on the Policy Manager Connector. This manager ensures that data access and usage comply with the organization's policies and regulatory requirements. It acts as an intermediary between Fybrik and various policy management systems (e.g., chaincodes), enabling Fybrik to understand and enforce the governance rules applied to data operations. In order to use our *smart policies* we implement the required *getPoliciesDecisions* endpoint inside our connector.

When the user code (e.g., notebook) requests access to a dataset through a FybrikApplication resource, Fybrik uses the policy connector to query the relevant chaincode for any policy decisions that apply to the requested operation. During this request, it transmits the context information such as data identifiers and intended asset use. Then smart policies process the request by comparing it to the defined rules (global standards or data contract), write the decisions to the ledger, and return the actions that the Fybrik platform has to apply (e.g., data masking).

Within the Fybrik platform, the data catalogs are represented as plugin components. It uses the Data Catalog Connector to query these catalogs to retrieve metadata information about datasets. Therefore, it is possible to provide the custom catalog implementation. For being operable inside Fybrik, we define the connector app which supports four required operations over the metadata:

- *createAsset* is used for registering a new product, e.g when the user executes the notebook and the corresponding workload persists in the data;
- *getAssetInfo* returns the metadata information which is used for product discovery, workload processing, etc;
- *updateAsset* function updates any existing metadata records within the catalog;
- *deleteAsset* is used for deleting the metadata about the product.

Our metadata catalog and policy manager are based on the previously developed blockchain-powered metadata system [16]. The actual prototype was enhanced to also include policy management. The main components of our HLF-based governance system include the membership service provider with a certificate authority (CA) used for identifying and authorizing the parties, endorsement and ordering nodes for transaction processing (metadata, policy, or data contract processing), and channel configurations used for listing the participating domain and their rights.

In the following, we provide an example of using the Hyperledger Fabric system.

5.2. Hypeledger Fabric-Based Federated Governance

In Figure 4, we present an updated view of the running example from Section 2.2 where Sales and Supply domains own the dedicated data products and host the blockchain endorsement nodes. These nodes are used as a federated governance tool. In Section 4.2, we discussed that the ledger can be used as a metadata and data contract storage medium while chaincode can perform the policy verification.

In Figure 5, we show the pseudo code of the implemented smart contract used for persisting the data product information in the ledger. After deploying such a smart contract in the HLF network, we are able to manipulate the data product information (lines 6–9) according to the Data Catalog Connector requirements. We are also able to discover the existing products (line 4), and to sign the data contract related to the product consumption (lines 11–12). Currently, the system supports only basic operations within the data contract enforcement which includes allowing reading or denying access.

The smart contract also implements the decision validation function (line 14), which is necessary for operating *getPoliciesDecisions* endpoint in our custom Policy Manager Connector. During policy evaluation, the function receives the requested product asset information (e.g., id, column info) and the operation context. After performing the policy checks (e.g., according to established data contract conditions) it returns the decision results.

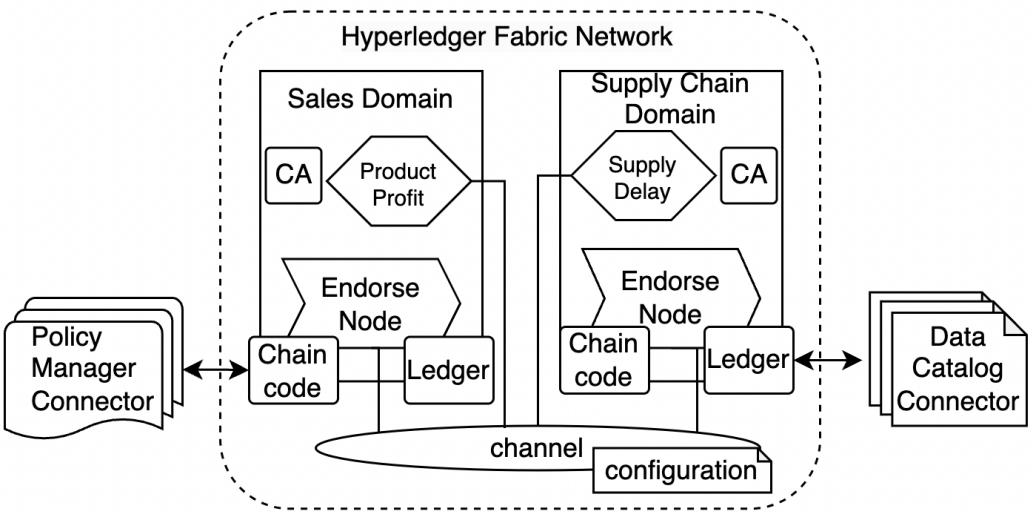

Figure 4. Federated governance system architecture based on HLF.

```
type CatalogContract struct {
    contractapi.Contract

    func GetAllProducts() ([]string, error)

    func GetDataProduct(id string) (*DataProductAsset, error)
    func RegisterDataProduct(product DataProductAsset, lineage []MetadataAsset) (*UUID, error)
    func UpdateDataProduct(id string, product DataProductAsset, lineage []MetadataAsset) (*UUID, error)
    func DeleteDataProduct(id string) (string, error)

    func RequestDataProduct(productId string, metadata DataContractAsset) (*UUID, error)
    func ResolveProductRequest(requestId string, metadata DataContractAsset) (string, error)

    func GetActionDecision(action streing, product DataProductAsset, context any) ([]string, error)
}
```

Figure 5. Pseudo code listing of the catalog smart contract.

5.3. The Data Product Quantum

According to the data mesh paradigm, the data product quantum refers to the smallest, self-sufficient unit of data that can independently deliver value as a product. It encapsulates all necessary aspects—data, metadata, quality, security, and access mechanisms—required to be useful and meaningful to its consumers, adhering to the principles of domain orientation and self-service infrastructure.

In Figure 6, we present the view of how the data assets deployed within the Fybrik platform constitute self-contained product quantums. Within our system, each data product includes four elements:

- The product Metadata—schema, semantic tags, lineage, etc;
- Notebook code for the data manipulation which is deployed to workload computing nodes (created automatically by Fybrik);
- FybrikApplication specification defining the source and destination identifiers, and intended operations provided in yaml format;
- Policies define the product usage rules, e.g., what is allowed to which consumers.

The product quantum may contain multiple policies. At the same time, each policy is applied to a single data output port that makes part of a data contract. This way the domain team can serve the same product to multiple consumers but on different conditions.

In Section 3.2, we discussed the purpose of data contracts. It provides a formal way to establish provider–consumer relationships and serves as a source of trust during the governance enforcement conducted by the platform.

For creating a new data product in our system, the domain team engineers have to follow the next procedure:

1. Request the data asset access, e.g., based on the protocol outlined in [16];
2. When the request is approved, the provider's metadata and policy are used to form a new data contract;
3. Submit the Notebook and FrybrikApplication documents for provisioning a new computing process;
4. Register a new data product in the catalog by providing the asset's metadata and policies.

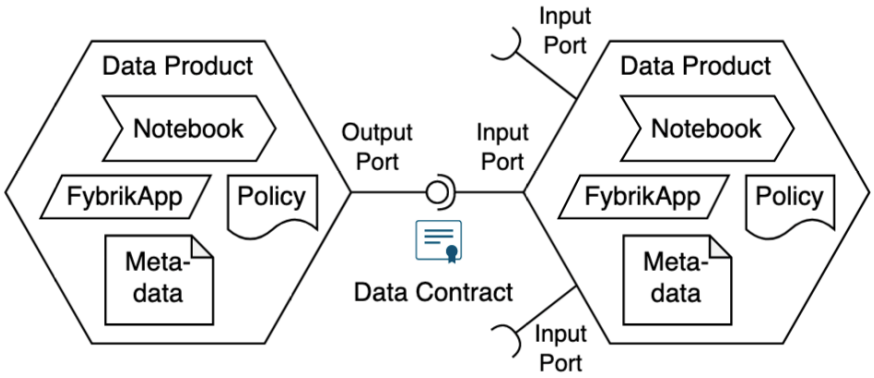

Figure 6. Data product quantum.

In this section, we described the architecture of our prototype system for building the federated data mesh governance. In the next section, we proceed by comparing our system and other related works based on the functional maturity requirements.

6. Contribution Discussion

Based on the provided requirements in Section 3, in Table 1 we have summarized the level of the functional maturity of related platforms from the governance perspective.

We see that no existing solution implements all the 11 requirements of the federated data mesh governance. As anticipated, almost all platforms have the data indexing and usage tracking modules, except the manual, web-form-based access control described in [9]. Indeed, these modules are built within any data platform architecture as they represent paramount importance in data management.

Semantic enrichment is a more difficult task than indexing or user authorization, thus leading to lower implementation rates. While our prototype and some platforms [9,10,19] (an extended description of the Netflix platform was accessed at: https://netflixtechblog.com/data-movement-in-netflix-studio-via-data-mesh-3fddcceb1059(accessed on 16 February 2024)) use the basic tag-based approach, the others use the advanced methods of knowledge graphs [13,21] or cloud-offered services for data similarity computation and enrichment [11,12]. This tendency is also reflected in the link generation aspect, where the most common technique is the lineage tracing of the data products, but some systems also provide the semantic relationship discovery and/or clustering options [11–13,21].

Unfortunately, very little information is provided on the versioning as part of the metadata management. Only our system enables metadata versioning based immutable ledger, while some research works [13,21] briefly mention this important feature.

It comes as no surprise that the polyglot data is represented only in the port-oriented data product models. This system design uses the input/output port connection abstraction in the first place and assumes the delivery of the identical product value by means of multiple channels (e.g., Message Queues, BLOBs).

Our model also uses an input/output port connection approach, but unfortunately, the current level of integration with other platforms in Fybrik is very limited. The Fybrik platform is quite recent, and today it is integrated with a few data storage providers (e.g., AWS S3, MySQL, Postgres) and processing engines (Apache Arrow Flight).

Even though modern companies adopt the responsibility of separations and containerization technology for ensuring the independent deployment of operational resources, it

is still an issue to bring this functionality to the governance tooling. Most systems either deploy it as a monolith module or it is under the management of a service provider. Since both parts of our system are based on Kubernetes, we are able to deploy any platform component independently—from computing nodes to smart policies.

The implementation of the automated policies is conducted as part of the access control modules (e.g., authorization enforcement) while the more general notion of the computational policies is not available. In some works, the process of instantiation of the composable products is still limited to the schema extensions [19,21].

When evaluating our system, we see that it satisfies 9 out of 11 requirements, which is greater than any other system. The distinct features of our prototype include the support of data contract management and enforcement based on the blockchain platform and partial support of the automatic governance system testing. This testing may be performed either during the gradual smart contract upgrades or in the mirrored testing networks so that any breaking changes would be refused.

Table 1. Overview of the supported federated governance requirements.

Property ⇓ / Source ⇒	**Our System**	**Zalando [19]**	**Netflix [19]**	**Machado et al. [9]**	**Wider et al. [10]**	**Butte et Butte [12]**	**Driessen et al. [21]**	**Hooshmand et al. [13]**	**Goedegebure et al. [11]**
Data Indexing	✓	✓	✓	✓	✓	✓	✓	✓	✓
Usage Tracking	✓	✓	✓	✧	✓	✓	✓	✓	✓
Semantic Enrichment	✧	✗	✧	✧	✧	✓	✓	✓	✓
Link Generation	✧	✗	✧	✧	✧	✓	✓	✓	✓
Data Polymorphism	✗	✗	✗	✗	✓	✓	✓	✗	✓
Data Versioning	✓	✗	✗	✗	✗	✗	✧	✧	✗
Independently Deployable	✓	✗	✗	✗	✗	-	✓	✓	-
Computational Policies	✧	✗	✗	✗	✧	✧	✗	✗	✧
Composable Products	✗	✗	✧	✗	✗	✗	✧	✗	✗
Automatically Testable	✧	✗	✗	✗	✗	-	✗	✗	-
Contract Management	✓	✗	✗	✗	✗	✗	✗	✗	✗
Total	(9)/11	2/11	(5)/11	(4)/11	(6)/11	(6)/11	(8)/11	(6)/11	(6)/11

✗—no information available or not supported ✧—partially supported ✓—supported.

7. Conclusions and Further Research

To conclude, in this article, we presented a further improvement of the blockchain-based federated governance system in data mesh.

First, we revisited and expanded the list of properties required for implementing efficient metadata and policy management. We provided an updated governance formal model that incorporates the utilization of data contracts, metadata, and product versioning.

Second, we described how the integration of two open-source platforms - Hyperledger Fabric and Fybrik - can benefit organizations that need to focus on data privacy, compliance, optimized data usage, and policy-driven decisions.

Third, we demonstrated a proof-of-concept system used to manage and govern data in cloud-native environments. It serves as a practical validation of the proposed multi-platform architecture for building the data mesh. We also presented a comparative study on other platforms aiming to build a data mesh governance and identified how our system satisfies the unaddressed issues.

We envision future research in multiple directions. The first observation is that the existing data mesh proposals are focused on a single company platform and our current prototype is also built on a single Kubernetes cluster. Therefore, we aim to explore the scalability and efficiency of blockchain-based governance in multi-cluster environments, e.g., cross data mesh scenarios or decentralized data sharing and management in general.

The second aspect is related to the limits of utilizing technology that affects the development of more sophisticated data contracts and policies, e.g., lack of support for

contract factory patterns. Hence, we want to investigate the impact of different blockchain platforms on performance and security and try to enhance data privacy and compliance in decentralized systems.

Author Contributions: Ideas structuration, formalization, and implementation, principal article writing—A.D.; Original idea, article review and editing—A.C. and A.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was conducted in the scope of a PhD thesis pursued by A.D., funded by University of Montpellier.

Data Availability Statement: The data presented in this study are available in this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Miloslavskaya, N.; Tolstoy, A. Big data, fast data and data lake concepts. In Proceedings of the 7th Annual International Conference on Biologically Inspired Cognitive Architectures (BICA 2016), Procedia Computer Science, New York, NY, USA, 16–19 July 2016.
2. Inmon, W.; Strauss, D.; Neushloss, G. *DW 2.0: The Architecture for the Next Generation of Data Warehousing*; Elsevier: Amsterdam, The Netherlands, 2010.
3. Madera, C.; Laurent, A. The next information architecture evolution: The data lake wave. In Proceedings of the 8th International Conference on Management of Digital Ecosystems, Hendaye, France, 2–4 November 2016; pp. 174–180.
4. Armbrust, M.; Ghodsi, A.; Xin, R.; Zaharia, M. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In Proceedings of the CIDR, Virtual Event, 11–15 January 2021; Volume 8.
5. Dehghani, Z. *Data Mesh: Delivering Data-Driven Value at Scale*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2022.
6. Evans, E.; Evans, E.J. *Domain-Driven Design: Tackling Complexity in the Heart of Software*; Addison-Wesley Professional: Boston, MA, USA, 2004.
7. Driessen, S.W.; Monsieur, G.; Van Den Heuvel, W.J. Data market design: A systematic literature review. *IEEE Access* **2022**, *10*, 33123–33153. [CrossRef]
8. DAMA-International. *DAMA-DMBOK: Data Management Body of Knowledge*; Technics Publications: Sedona, AZ, USA, 2017.
9. Araújo Machado, I.; Costa, C.; Santos, M.Y. Advancing Data Architectures with Data Mesh Implementations. In Proceedings of the International Conference on Advanced Information Systems Engineering, Leuven, Belgium, 6–10 June 2022; Springer: Cham, Switzerland, 2022; pp. 10–18.
10. Wider, A.; Verma, S.; Akhtar, A. Decentralized data governance as part of a data mesh platform: Concepts and approaches. In Proceedings of the 2023 IEEE International Conference on Web Services (ICWS), Chicago, IL, USA, 2–8 July 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 746–754.
11. Abel, G. Data Mesh: Systematic Gray Literature Study, Reference Architecture, and Cloud-Based Instantiation at ASML. Master's Thesis, School of Economics and Management, Tilburg University, Tilburg, The Netherlands, 2022.
12. Butte, V.K.; Butte, S. Enterprise Data Strategy: A Decentralized Data Mesh Approach. In Proceedings of the 2022 International Conference on Data Analytics for Business and Industry (ICDABI), Virtual, 25–26 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 62–66.
13. Hooshmand, Y.; Resch, J.; Wischnewski, P.; Patil, P. From a Monolithic PLM Landscape to a Federated Domain and Data Mesh. *Proc. Des. Soc.* **2022**, *2*, 713–722. [CrossRef]
14. Dolhopolov, A.; Castelltort, A.; Laurent, A. Exploring the Benefits of Blockchain-Powered Metadata Catalogs in Data Mesh Architecture. In Proceedings of the 15th International Conference on Management of Digital EcoSystems, Crete, Greece, 5–7 May 2023; Springer: Cham, Switzerland, 2023.
15. Dolhopolov, A.; Castelltort, A.; Laurent, A. Trick or Treat: Centralized Data Lake vs Decentralized Data Mesh. In Proceedings of the 15th International Conference on Management of Digital EcoSystems, Crete, Greece, 5–7 May 2023; Springer: Cham, Switzerland, 2023.
16. Dolhopolov, A.; Castelltort, A.; Laurent, A. Implementing a Blockchain-Powered Metadata Catalog in Data Mesh Architecture. In Proceedings of the International Congress on Blockchain and Applications, Guimarães, Portugal, 12–14 July 2023; Springer: Cham, Switzerland, 2023; pp. 348–360.
17. Newman, S. *Building Microservices*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2015.
18. Priebe, T.; Neumaier, S.; Markus, S. Finding your way through the jungle of big data architectures. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 5994–5996.
19. Machado, I.A.; Costa, C.; Santos, M.Y. Data mesh: Concepts and principles of a paradigm shift in data architectures. *Procedia Comput. Sci.* **2022**, *196*, 263–271. [CrossRef]

20. Machado, I.; Costa, C.; Santos, M.Y. Data-driven information systems: The data mesh paradigm shift. In *Information Systems Development: Crossing Boundaries between Development and Operations (DevOps) in Information Systems (ISD2021 Proceedings)*; Insfran, E., González, F., Abrahão, S., Fernández, M., Barry, C., Linger, H., Lang, M., Schneider, C., Eds.; Universitat Politècnica de València: Valencia, Spain, 2021.
21. Driessen, S.; Monsieur, G.; van den Heuvel, W.J. Data Product Metadata Management: An Industrial Perspective. In Proceedings of the Service-Oriented Computing–ICSOC 2022 Workshops: ASOCA, AI-PA, FMCIoT, WESOACS 2022, Sevilla, Spain, 29 November–2 December 2022; Springer: Cham, Switzerland, 2023; pp. 237–248.
22. Sawadogo, P.; Darmont, J. On data lake architectures and metadata management. *J. Intell. Inf. Syst.* **2021**, *56*, 97–120. [CrossRef]
23. Stafford, V. Zero trust architecture. *NIST Spec. Publ.* **2020**, *800*, 207.
24. Truong, H.L.; Comerio, M.; De Paoli, F.; Gangadharan, G.; Dustdar, S. Data contracts for cloud-based data marketplaces. *Int. J. Comput. Sci. Eng.* **2012**, *7*, 280–295. [CrossRef]
25. Hai, R.; Geisler, S.; Quix, C. Constance: An intelligent data lake system. In Proceedings of the International Conference on Management of Data, San Francisco, CA, USA, 26 June–1 July 2016; ACM Digital Library: New York, NY, USA, 2016.
26. Zhao, Y. Metadata Management for Data Lake Governance. Ph.D. Thesis, École Doctorale Mathématiques, Informatique et Télécommunications, Toulouse, France, 2021.
27. Sawadogo, P.N.; Darmont, J.; Noûs, C. Joint Management and Analysis of Textual Documents and Tabular Data within the AUDAL Data Lake. In Proceedings of the European Conference on Advances in Databases and Information Systems, Tartu, Estonia, 24–26 August 2021; Springer: Cham, Switzerland, 2021; pp. 88–101.
28. Eichler, R.; Giebler, C.; Gröger, C.; Schwarz, H.; Mitschang, B. Modeling metadata in data lakes—A generic model. *Data Knowl. Eng.* **2021**, *136*, 101931. [CrossRef]
29. Mehmood, H.; Gilman, E.; Cortes, M.; Kostakos, P.; Byrne, A.; Valta, K.; Tekes, S.; Riekki, J. Implementing big data lake for heterogeneous data sources. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW), Macao, China, 8–12 April 2019; IEEE: Piscataway, NJ, USA, 2019.
30. Halevy, A.Y.; Korn, F.; Noy, N.F.; Olston, C.; Polyzotis, N.; Roy, S.; Whang, S.E. Managing Google's data lake: An overview of the Goods system. *IEEE Data Eng. Bull.* **2016**, *39*, 5–14.
31. Apache Software Foundation. Apache Atlas—Data Governance and Metadata Framework for Hadoop. Available online: https://atlas.apache.org (accessed on 14 August 2023).
32. DataHub Project. The Metadata Platform for the Modern Data Stack. Available online: https://datahubproject.io/ (accessed on 14 August 2023).
33. Abbas, A.E.; Agahari, W.; Van de Ven, M.; Zuiderwijk, A.; De Reuver, M. Business data sharing through data marketplaces: A systematic literature review. *J. Theor. Appl. Electron. Commer. Res.* **2021**, *16*, 3321–3339. [CrossRef]
34. Desai, H.; Liu, K.; Kantarcioglu, M.; Kagal, L. Adjudicating violations in data sharing agreements using smart contracts. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1553–1560.
35. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018; pp. 1–15.
36. Demichev, A.; Kryukov, A.; Prikhodko, N. The approach to managing provenance metadata and data access rights in distributed storage using the hyperledger blockchain platform. In Proceedings of the Ivannikov Ispras Open Conference, Moscow, Russia, 22–23 November 2018; IEEE: Piscataway, NJ, USA, 2018.
37. Koscina, M.; Manset, D.; Negri-Ribalta, C.; Perez, O. Enabling trust in healthcare data exchange with a federated blockchain-based architecture. In Proceedings of the International Conference on Web Intelligence—Companion Volume, Thessaloniki, Greece, 14–17 October 2019.
38. Valenta, M.; Sandner, P. Comparison of ethereum, hyperledger fabric and corda. *Frankf. Sch. Blockchain Cent.* **2017**, *8*, 1–8.
39. Ayed, D.; Dragan, P.A.; Félix, E.; Mann, Z.A.; Salant, E.; Seidl, R.; Sidiropoulos, A.; Taylor, S.; Vitorino, R. Protecting sensitive data in the cloud-to-edge continuum: The FogProtect approach. In Proceedings of the 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Taormina, Italy, 16–19 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 279–288.
40. Dittmann, G.; Giblin, C.; Osborne, M. Automating privacy compliance in the decentralized enterprise. In Proceedings of the IEEE International Conference on Big Data (Big Data), Osaka, Japan, 17–20 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 2218–2223.