Data Mesh: a Systematic Gray Literature Review

ABEL GOEDEGEBUURE, INDIKA KUMARA, STEFAN DRIESSEN, and WILLEM-JAN VAN DEN HEUVEL, Tilburg University, Netherlands
GEERT MONSIEUR and DAMIAN ANDREW TAMBURRI, Eindhoven University of Technology, Netherlands
DARIO DI NUCCI, University of Salerno, Italy

Data mesh is an emerging domain-driven decentralized data architecture that aims to minimize or avoid operational bottlenecks associated with centralized, monolithic data architectures in enterprises. The topic has picked the practitioners' interest, and there is considerable gray literature on it. At the same time, we observe a lack of academic attempts at defining and building upon the concept. Hence, in this article, we aim to start from the foundations and characterize the data mesh architecture regarding its design principles, architectural components, capabilities, and organizational roles. We systematically collected, analyzed, and synthesized 114 industrial gray literature articles. The review provides insights into practitioners' perspectives on the four key principles of data mesh: data as a product, domain ownership of data, self-serve data platform, and federated computational governance. Moreover, due to the comparability of data mesh and SOA (service-oriented architecture), we mapped the findings from the gray literature into the reference architectures from the SOA academic literature to create the reference architectures for describing three key dimensions of data mesh: organization of capabilities and roles, development, and runtime. Finally, we discuss open research issues in data mesh, partially based on the findings from the gray literature.

CCS Concepts: • **Information systems → Data management systems**;

Additional Key Words and Phrases: data mesh, decentralized data architectures, data governance, gray literature review

ACM Reference Format:
Abel Goedegebuure, Indika Kumara, Stefan Driessen, Willem-Jan van den Heuvel, Geert Monsieur, Damian Andrew Tamburri, and Dario Di Nucci. 2023. Data Mesh: a Systematic Gray Literature Review. *J. ACM* 37, 4, Article 111 (August 2023), 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The world is living in the golden age of data. IDC (International Data Corporation) predicts that by 2025 the amount of digital data generated by enterprises and individuals will grow 61% to 175 zettabytes [36]. While the advances in the analytic data landscape can enable organizations to turn their ever-growing raw data into value, the difficulties in the integration, management, and governance of data at scale hamper the success of data analytic projects of organizations [20, 35]. In the current

Authors' addresses: Abel Goedegebuure, a.a.goedegebuure@tilburguniversity.edu; Indika Kumara, i.p.k.weerasingh adewage@tilburguniversity.edu; Stefan Driessen, s.w.driessen@tilburguniversity.edu; Willem-Jan van den Heuvel, w.j.a.m.vdnHeuvel@tilburguniversity.edu, Tilburg University, Warandelaan 2, Tilburg, North Brabant, Netherlands, 5037 AB; Geert Monsieur, g.monsieur@tue.nl; Damian Andrew Tamburri, d.a.tamburri@tue.nl, Eindhoven University of Technology, Groene Loper 3, Eindhoven, North Brabant, Netherlands, 5612 AZ; Dario Di Nucci, ddinucci@unisa.it, University of Salerno, Via Giovanni Paolo II, 132, Fisciano SA, Salerno, Italy, 84084.

0004-5411/2023/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

practices of centralized data management, domain teams rely on a central data management team to collect, process, and manage domain data. This central team is becoming the bottleneck for unlocking the value from domain data. There is a need to shift the domain data's responsibility from the central team to domain teams [13].

A data mesh is emerging as a novel decentralized approach for managing data at scale by applying domain-oriented, self-serve design and product thinking [13]. Zhamak Dehghani first defined the term data mesh in 2019 [44]. Figure 1 shows the search index for "Data Mesh" on Google Trends for the past five years. A clear increasing trend line can be seen, indicating the increased interest in data mesh. Despite the popularity of data mesh, little academic literature exists on the topic. However, as organizations increasingly look into data mesh, we saw an opportunity to systematically review the gray literature to characterize the concept better and drive a research agenda.

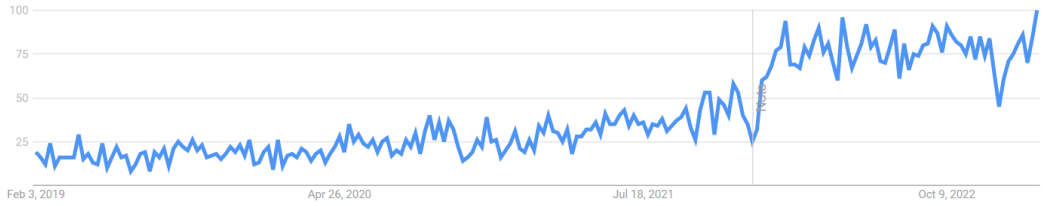

Fig. 1. Google trends for the word "Data Mesh".

Zhamak [13] defines the architecture of data mesh based on four principles; data as products, domain ownership of data, self-serve data platform, and federated computational governance. The data assets in an organization are offered as products that enable consumers to efficiently and autonomously discover and use domain data. Domain teams closest to the origin of data create and manage data products. A mesh of data products emerges when domains consume data products from each other. To ensure the data products are interoperable and compliant with internal and external regulations and policies, a federated governance team is responsible for defining and enforcing product interoperability standards and compliance policies across the data mesh. Finally, a self-serve data platform supports the domains by offering infrastructure and platform services necessary to build, maintain and manage data products.

To a certain extent, we observed a strong analogy between data mesh and service-oriented architecture (SOA), particularly with domain-driven microservices [7, 13, 44]. The success of the domain-oriented design and microservices architecture was an inspiration for inventing the data mesh architecture [13, 44]. Data products are often exposed as services and consumed through APIs. A data mesh relates and connects data products, which is analogous to a service mesh. The product owners can publish their data products, and consumers can discover and interconnect the desired products to perform cross-domain data analysis and build new value-added data products, which is analogous to service composition.

In this paper, we report a Systematic Gray Literature Review (SGLR) that aims to identify and review the gray literature on the data mesh to synthesize the practitioners' understanding of the data mesh and identify research challenges for academia. We selected and analyzed 114 gray literature articles. Our findings include the practitioners' characterization of four principles of a data mesh, common architectural components in a data mesh, and potential benefits and drawbacks of a data mesh. Moreover, the similarities between SOA and data mesh motivated us to adopt the well-established reference architectures from SOA to organize the findings from our SGLR into three reference architectures. Our mapping of data mesh to SOA can also help the research community

understand research challenges in data mesh and the potential use of SOA research results to address those challenges.

The remainder of this article is organized as follows. Section 2 describes our systematic literature review methodology. Based on the findings of the gray literature, Section 3 defines the data mesh and its four principles and presents its potential benefits and drawbacks. Section 5 describes our three reference architectures for the data mesh, while Section 6 lists potential research challenges. Section 7 summarizes the academic research studies on data mesh and the literature review studies relevant to our survey. Finally, we discuss the threats to the validity of our research in Section 8 and conclude the paper in Section 9.

2 Methodology

Due to the novelty of the data mesh paradigm, little academic literature is available on the topic. At the same time, since the data mesh concept has its roots in the industry as opposed to academics [44], gray literature about the topic, such as blog posts and white papers, is widely available. Since the quality of structured literature reviews (SLRs) depends greatly on the availability of enough source material, a gray literature review (GLR) was conducted to investigate the state-of-the-art for data mesh. In this work, we follow the guidelines provided by Garousi et. al. [19], who adapted the SLR guidelines of Kitchenham and Charters [22] specifically for GLRs. Following these methodologies, our GLR is organized into three stages: a planning and design phase, an execution phase, and a reporting phase. A high-level overview of these phases and their steps are depicted in Figure 2.

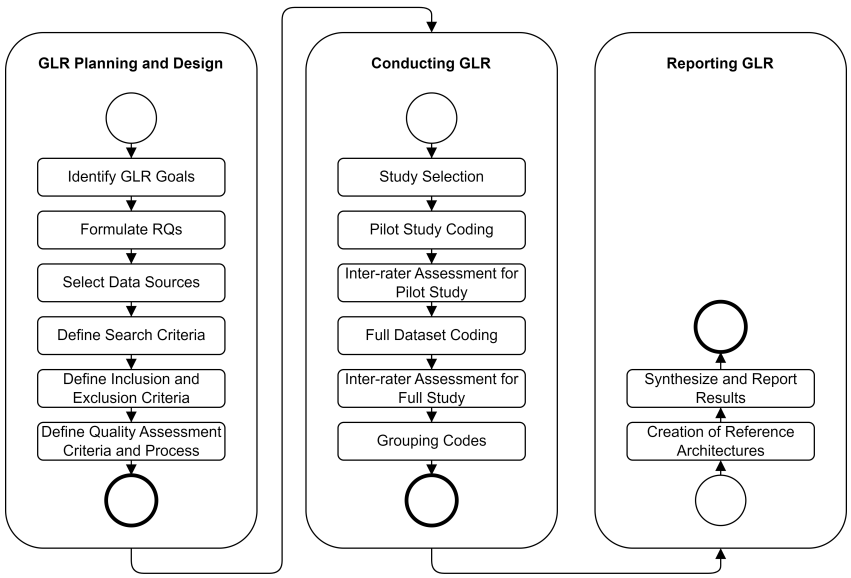

Fig. 2. Systematic gray literature review process.

2.1 Planning and Design of the Gray Literature Review

Our main goals were to provide a unified definition of the data mesh concept, create reference architectures for data mesh, and identify research challenges. We formulated four research questions to achieve these goals:

RQ1. What is a data mesh and why is it needed? A data mesh embodies four design principles [13, 44]: data as a product, domain-driven ownership of data, self-serve data platform, and federated computational governance. Practitioners may have different perspectives on these principles and experiences applying them. Hence, RQ1 aims to understand and characterize the four principles of data mesh from a practitioner's standpoint.

RQ2. What are the benefits and concerns of adopting data mesh? A data mesh has advantages and disadvantages and is not a one-size-fits-all approach for data management. Thus, RQ2 aims to understand data mesh's potential benefits, implementation challenges, and drawbacks.

RQ3. How should an organization build data mesh? Some organizations have started their data mesh journey and report their experience, including development and operation guidelines, options, and examples. Hence, RQ3 aims to analyze the relevant gray articles to create a set of reference architectures to guide organizations in making better design choices embarking on their data mesh journey. Furthermore, due to the novelty of the data mesh architectures, we aim to build reference architectures by mapping findings from the gray literature into the architectures used in comparable domains.

RQ4. What are the research challenges concerning data mesh? In the gray articles, practitioners report critical challenges; hence, RQ4 aims to identify research challenges concerning data mesh based on practitioners' challenges.

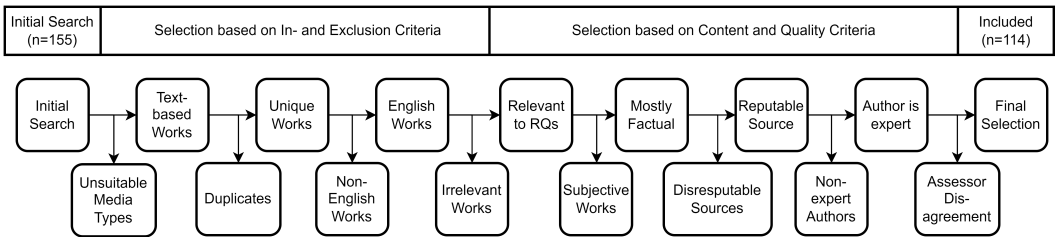

Fig. 3. The search and selection process. Steps are shown sequentially, and each step reduces the number of candidate sources until the final selection. After the final exclusion step, an inter-rater reliability test—featuring the well-established Cohen Kappa coefficient calculation—was performed with $\kappa = 0.79$ to ensure that reviewer bias was not inappropriately high.

Study Selection Strategy. Following the protocols by Kitchenham et. al. [22], we defined several search strings to identify relevant gray literature sources. Then, each string was applied to our selected search engine (Google), after which our pre-established inclusion and exclusion criteria were used to refine the results as shown in Figure 3.

Initial Search. As is common practice in GLRs, our search string selection process was based on trial queries with different search strings, which can indicate the usefulness of that search string [43]. After trying several combinations of search terms, we settled on two queries that appeared to yield promising results for answering our research questions:

Q1: Data Mesh (Requirements | Benefits | Drawbacks | Concerns | Advantages | Disadvantages | Challenges | Components | Architecture | Design)

Q2: Decentralized Data Architecture

These queries were then executed in the Google search engine[1] on September 1^{st} 2021, and all search results were checked against our inclusion and exclusion criteria. Due to the relative novelty of data mesh, information and sources at that time were mostly limited to the generic data mesh principles. Therefore, the GLR was extended to include data sources about the architectural design of data mesh until May 1^{st} 2022. This step resulted in 155 data sources from 2019 until 2022.

Inclusion and Exclusion Criteria. To effectively process our sources in line with grounded theory, we limited our gray literature source selection to text-based sources, such as reports, blog posts, white papers, official documentation from vendors, presentations, and transcriptions of keynotes and webinars. Additionally, duplicate and non-English sources were eliminated to remove bias and ensure the text was legible to the authors.

Quality- and Content-based Selection. To guarantee the relevance and quality of the sources used for this GLR, we applied several quality criteria inspired by previous gray literature reviews [5, 19, 23]. As a first step, we eliminated any work that does not explicitly contain either a theoretical discussion or a practical implementation of a data mesh that can aid us in answering our proposed research questions. Examples of the former include a formal definition, an example architecture, or an overview of the benefits and challenges of data mesh implementation. Examples of the latter include discussing tools or an instantiation of (components of) a data mesh.

Next, works that consist primarily of subjective statements that promote the author's opinion, as opposed to factual information, were excluded. Furthermore, we checked whether the publishing organization is respected in the field of information technology and whether the author is a reputable individual, i.e., they are associated with a reputable organization, they have reputable expertise, or have published other articles in the field of data management.

Finally, two authors checked each article against the content and quality criteria to eliminate potential selection bias. A mutual agreement was required for final inclusion in the data set, and disagreements were discussed between the authors. The resulting inter-rater reliability assessment [38] yielded a Cohen Kappa of 0.79, which indicates substantial agreement between the authors.

The final selection consists of 114. Figure 4 shows the distribution of the selected sources. A full overview of each source can be found in our online appendix (see Section 2.4).

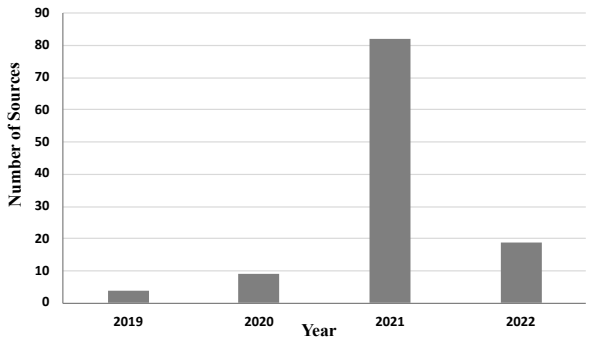

Fig. 4. The publication date of the selected sources (114). The significant upward trend in publications indicates the novelty of data mesh.

[1] google.com

2.2 Conducting the Gray Literature Review

The second set of steps is related to the execution of the review in line with a qualitative analysis methodology. Structural coding was used to get an initial overview of the topics, after which descriptive coding was used to summarize parts of the topics. Structural coding was used to identify larger text segments on a broad topic. Descriptive coding was then used to go more in-depth on these topics by summarizing the basic topic into a word, or short phrase [39]. Coding was performed using Atlas.ti too [2].

Like [23], an initial set of 20 sources was randomly selected and analyzed to establish an initial set of codes. These were discussed between the first three authors of the paper. Once consensus on the initial set of codes was reached, the entire dataset was coded. The grouping of codes into topics and the identification of the categories were established through discussion by the first three authors of this paper. An example of codes for a single topic of the “Domain Ownership" category can be seen in Table 1

We first used the data extracted from the gray literature using the established codes to answer RQ1 and RQ2. Next, to address RQ3, we created three reference architectures by mapping our findings to the reference architectures used in service-oriented computing [24, 26, 32, 33, 48]. The first three authors of the paper collaboratively created the reference architectures over a set of meetings. The other authors validated such architectures. We addressed any discrepancies through discussions. We followed a similar approach to answer RQ4. We identified the potential research challenges by mapping the difficulties mentioned by the practitioners in the gray articles to the relevant research issues from the service-oriented computing [33, 34, 46] and data management [2, 15, 20, 42].

2.3 Reporting the Results

The results from the systematic gray literature are presented in the rest of this paper. We defined four main categories as shown in Figure 5. For each category, a table shows the topics, the underlying atomic codes, and the sources that address these codes. Table 1 shows a snippet of the results table for the Domain Ownership category. Each topic has several underlying codes. For each topic and code, we provide a unique ID, the frequency of their occurrence in the sources, and the sources themselves. Note that a particular code can appear multiple times in a single source.

Table 1. A snippet of the results for the “Domain Ownership” category.

Sub Category			Atomic Codes			
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Unique Sources**
2.1	Domain Formation	42	2.1.1	Decentralized Domains	14	S1,S9,S19,S21,S22,S42,S45,S48,S52,S55,S78,S91,S110
			2.1.2	Domain-Driven Design Method	28	S2,S11,S36,S37,S39,S45,S46,S53,S55,S58,S71,S75,S78, S79,S87,S90,S91,S96,S98,S100,S103,S110,S111,S112

2.4 Replication Package

We made the related data available online[3] to enable further validation and replication of our study. It contains the full list of sources and the qualitative analysis (i.e., codes, groupings, and analysis) performed with the *Atlas.ti* tool[4].

[2]https://atlasti.com/
[3]https://github.com/IndikaKuma/DataMesh
[4]https://atlasti.com/

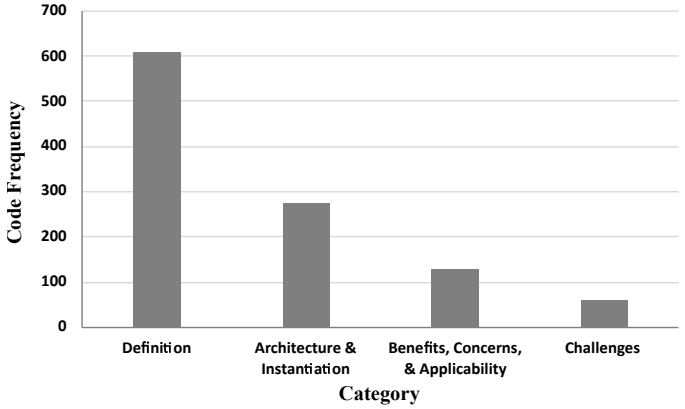

Fig. 5. Topics discussed in the analyzed studies.

3 RQ1. What is Data Mesh and Why is it Needed?

In short, a data mesh is a domain-oriented decentralized architecture for managing (analytical) data at scale. It enables the decomposition of an organization's monolithic analytical data space into data domains aligned with business domains. Such decomposition moves the responsibility of managing and providing high-quality data and valuable insights from the conventional central data team into domain teams that intimately know the data.

In this section, based on the results from our GRL, we first define a data mesh, explaining its four principles: (i) data product thinking (Section 3.1), (ii) domain ownership (Section 3.2), (iii) federated computational governance (Section 3.3), (iv) and self-serve data platform (Section 3.4). Next, we explain the expected benefits of a decentralized data mesh architecture, the concerns of a data mesh implementation, and to which organizations a data mesh architecture is applicable (Section 4).

Tables 2 to 6 overviews the sub-categories derived from the gray literature review. Each table also overviews the atomic codes, their frequency, and the unique studies that mentioned them. The results refer to the sub-categories or codes through their labels; for example, ID1.1.1 refers to the code "Data" belonging to the sub-category "Components" part of the category "Data As a Product".

3.1 Data as a Product

This principle applies product thinking to analytical data to offer the data as a valuable asset to potential data consumers.

ID1.1. Components. Similar to a service in SOA, a data product is an autonomous entry that receives or takes the input data from one or more sources, applies some computation/transformation on the received data, and produces and serves the results excepted by the consumers. We identified five common elements for a data product: data, metadata, code, interfaces, and infrastructure. However, some data products may include product-specific artifacts, e.g., a machine-learning model.

The metadata of a data product may describe its owners, schemas, quality metrics, and access policies. The product code is responsible for ingesting/consuming and transforming the data from source systems or upstream data products and serving the transformed data to end-users or downstream data products. The product can also include code to enforce various data governance policies programmatically (*policy as a code*), including data quality, access control, and privacy. A product also has various interfaces that define and enable interactions with other data products, data sources,

Table 2. Data as a product: codes and sources.

Sub Category			Atomic Codes			
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Unique Sources**
1.1	Components	67	1.1.1	Data	13	S1, S24, S37, S57, S82, S97-S99, S103, S104, S108, S110
			1.1.2	Metadata	13	S1, S10, S37, S45, S53, S57, S82, S97, S98, S110, S113
			1.1.3	Code	13	S1, S37, S86, S97, S98, S102, S103, S104, S108, S110
			1.1.4	Interfaces	19	S1,S24,S28,S29,S37,S53,S67,S82,S86,S88,S103,S108,S110,S113
			1.1.5	Infrastructure	9	S1,S37, S53, S57, S77, S98, S97
1.2	Data Product Types	32	1.2.1	Basic Data Products	12	S2,S6,S20,S86-S88,S96,S106,S109,S110,S112
			1.2.2	Composite Data Products	20	S2,S6,S86-S88,S95,S96,S98,S106,S109,S110,S112
1.3	Characteristics of Data Products	132	1.3.1	Discoverable	28	S1,S2,S3,S7,S9,S15,S16,S32,S33,S37,S38,S43,S53,S55, S57,S65,S73,S75,S77,S82,S88,S99,S100
			1.3.2	Interoperable	17	S2,S7,S15,S32,S37,S47,S53,S57, S63,S65,S77,S88,S99,S100,S113
			1.3.3	Natively Accessible	12	S15,S16,S32,S33,S37,S53,S66,S82,S88,S91,S97,S99
			1.3.4	Self-describing	12	S2,S3,S7,S28,S37,S38,S53,S57,S65,S77,S88,S100
			1.3.5	Understandable	9	S1,S9,S15,S16,S19,S55,S66
			1.3.6	Secure	21	S1,S7,S9,S15,S16,S18,S19,S24,S32,S37,S53,S57,S65,S77,S82
			1.3.7	Trustworthy	22	S1,S2,S7,S9,S15,S16,S32,S37,S38,S53,S57,S65,S77,S82, S88,S97,S99,S100,S113
			1.3.8	Valuable	11	S15,S16,S85,S91,S109-S112

data sinks, and management applications. Lastly, a product needs infrastructure resources and software platform services to ingest and store its data, deploy and execute its code, and make its data and capabilities accessible through its interfaces.

ID1.2. Types of Data Products. We identified two types of data products: atomic data products and composite data products. The first type uses data from data source systems (e.g., operational databases, external APIs, and sensors) as its source data and transforms it into the desired format. The processed data is then provisioned as a product to be consumed by the organization's end-users or downstream data products. Gray literature refers to atomic data products as source-aligned because they mostly ingest data from the operational source systems. Composite data products use other data products (both basic or composite data products) for their source data and integrate them to create new value-added products. Gray literature also categorizes composite data products as aggregates and consumer-aligned data products. Compared to the aggregate, customer-aligned data products are generally not sold as data products to other domains; instead, they serve data to implement the end-user use cases of the organization readily.

Figure 6 shows data products spread across domains. Some data products, e.g., product search, order, and invoice, are atomic products. They mostly ingest raw data from the underlying operational data sources (e.g., product and order databases), clean and transform the raw data, and store them locally. They also provide the interfaces to serve their local domain data to upstream data products. The composite data products, e.g., 360-degree customer views, product recommendations, and funnel analytic dashboards, aggregate data from atomic or other composite data products. Some data products, e.g., funnel analytics and CRM dashboards, primarily serve end-users.

ID1.3. Characteristics of Data Products. From the gray literature, we identified eight attributes of a high-quality data product.

ID1.3.1. Discoverable. Discoverability aims to overcome the high costs and friction of data discovery in centralized data architectures. Data consumers should be able to discover the data products in an organization easily. Gray literature recommends using a central data/product catalog. All products and their metadata should be published in the catalog so consumers can browse and search for data products easily.

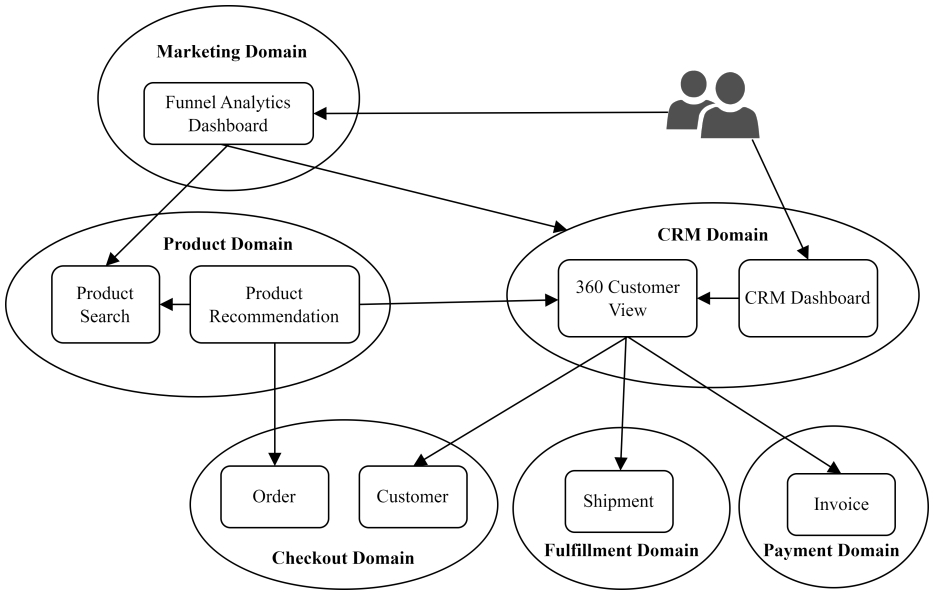

Fig. 6. An example of data products in different domains, adopted from [7].

ID1.3.2. Interoperable. It should be (relatively) easy to combine or integrate data from different data products into composite data products. The consistent use of open standards and conventions across data products can increase interoperability between data products. Gray literature lists several standards and practices a data mesh architecture should encompass. For example, the open standards for describing data products and their composition, including interfaces, metadata, and (access and management) policies, can ensure the interoperability of data products within and across organizations. In addition, standard metadata fields and business glossary models can guarantee that all data products are described unambiguously. Finally, a common method for data quality modeling can ensure all domains define their products' data quality concerning the same KPIs (key performance indicators).

ID1.3.3. Natively Accessible. All data products should provide a unique, programmatically accessible endpoint. For example, a data product can access its data as a web service or SQL endpoint. The endpoint description can be included in the metadata to enable consumers to connect to the data product automatically.

ID1.3.4. Self-describing. Data products should be documented so consumers can use them without further assistance from the product owners or data stewards. Self-descriptions allow the organization to effectively scale out its data sources and consumers as the human dependencies are reduced as much as possible. The gray literature mentions several metadata that make a data product self-describing. For example, interface descriptions can define all access endpoints and the policies that govern data access. The information about product owners and subject matter experts can ensure that the consumers know whom to contact when needing more information related to the data. The schemas for all the data views can be used to describe the data structure. Service level objectives (SLOs) can describe the product's quality and act as a data contract. Finally, the metadata of data lineage and provenance can show the origin of the data and gives consumers trust in the data product as it becomes clear how the data is created and processed.

ID1.3.5. Understandable. The models and the natural language documentation of data products should help potential consumers to understand the products' semantics, syntactic, and behaviors/usage. For example, a data product description can include the entities in the data the product consumes and produces, the relationship between those entities, and the relationships with the other data products in the mesh.

ID1.3.6. Secure. A data product needs to use appropriate measures to ensure data confidentiality, integrity, and availability. The measures include but are not limited to data access control, data encryption at rest and data in transit, and data access audit logs. In the data mesh architecture, while the security policies are primarily managed centrally, their enforcement happens in a decentralized fashion. For example, product owners can authorize consumers to access their products' interfaces.

ID1.3.7. Trustworthy. Data consumers should be able to trust the data coming from a data product. The gray literature provides several guidelines to enhance the trustworthiness of data products. First, product owners can use a methodology provided by the federated governance team to model data quality to define their own product's quality on several dimensions (e.g., completeness, accuracy, and timeliness). A set of service-level objectives (SLOs) can capture the target quality for each data product and act as a contract between the product owner and the product consumer. Finally, automation can support product developers in ensuring quality throughout the product's lifetime (e.g., automated monitoring of data quality issues and execution of corrective actions).

ID1.3.8. Valuable. A product needs to possess long-term value to the rest of the organization. The value should be continuously monitored and maintained. This can have implications for the deployment and operation of data products. For example, suppose the data product is an ML prediction service. In that case, the performance of the ML model needs to be continuously monitored and updated as necessary to keep its performance acceptable (i.e., MLOps (machine learning operations) [41] should be used).

Table 3. Domain ownership: codes and sources.

Sub Category			Atomic Codes			
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Unique Sources**
2.1	Domain Formation	42	2.1.1	Decentralized Domains	14	S1,S9,S19,S21,S22,S42,S45,S48,S52,S55,S78,S91,S110
			2.1.2	Domain-Driven Design Method	28	S2,S11,S36,S37,S39,S45,S46,S53,S55,S58,S71,S75,S78, S79,S87,S90,S91,S96,S98,S100,S103,S110,S111,S112
2.2	Domain Responsibilities	59	2.2.1	Data Product Development and Operation	49	S2,S4,S9,S13,S16,S20-22,S24,S25,S29,S33,S35,S41-43, S47,S49,S51,S53,S58,S64,S67,S71,S73,S74,S77,S87
			2.2.2	Domain Governance	10	S13,S35, S43, S63, S64,S71,S73,S76,S88

3.2 Domain Ownership of Data

Data mesh distributes the ownership of data to the teams or individuals closest to the origin of the data.

ID2.1. Domain Formation. Data mesh decomposes the analytical data in an organization into a set of disjoint domains. Autonomous domain teams are responsible for building and managing data products from domain data. In particular, data mesh advocates applying the Domain-Driven Design (DDD) approach to analytical data. DDD decomposes an organization's business functions and capabilities into domains, enabling the designing of software systems based on models of the underlying domains [17]. A domain model describes selected aspects of a domain using the vocabulary (i.e., abstractions) shared by domain experts, users, and developers. Organizations already

use separation of responsibilities by decomposing the organization into different business domains (e.g., products, sales, and marketing). Data products are built around operational and analytical data produced by these business domains. Hence, business domains form logical constituents for the distribution of data ownership and may be decomposed into subdomains to manage their complexity better (see Section 5.2).

ID2.2. Responsibilities. We identified two key responsibilities for domain teams: building and maintaining data products and carrying out governance activities at the domain level. Domain teams possess in-depth knowledge about domain data, including potential use cases and consumers of data, and the processes and techniques for maintaining the quality of data. Hence, they can determine which domain data assets should be provisioned as data products. They can use the tools and services provided by the self-serve data platform to create, manage and serve data products (see Section 3.4). Domain teams also carry out governance activities to ensure compliance and conformance of data products to relevant regulations, organizational policies, and standards (see Section 3.3).

Table 4. Federated computational governance: codes and sources.

Sub Category			Atomic Codes			
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Unique Sources**
3.1	Global Governace	74	3.1.1	Define Organization-wide Standards and Guidelines	30	S1,S3,S4,S9,S15,S18,S19,S21,S25,S33,S37,S38, S53,S58,S60,S62,S74,S77,S82,S95,S111
			3.1.2	Define and Enforce Global Governance Policies	15	S9,S21,S28,S36,S37,S53,S63,S67,S110,S111
			3.1.3	Define Data Quality Assessment Methodology	9	S1,S3, S25, S35, S45, S47, S58, S64, S111
			3.1.4	Business Glossary Modeling	6	S38,S90,S111,S114
			3.1.5	Data Mesh Monitoring	7	S1,S35,S57,S76,S81,S110
			3.1.6	Creating Incentive Models	7	S1,S25,S88,S98
3.2	Local Governance	34	3.2.1	Modeling Products' Data	8	S1,S2,S13,S16,S24,S37,S55
			3.2.2	Data Product Access Control	6	S13,S28,S29,S64,S86,S98
			3.2.3	Data Product Compliance and Conformance	6	S28,S64,S74,S86,S111
			3.2.4	Data Product Quality Assurance	9	S13,S20,S28,S47,S63,S64,S86,S88,S110
			3.2.5	Data Product Monitoring	5	S4,S48,S71,S88,S110
3.3	Automation in Governance	11	3.3.1	Objectives of Automation	5	S1,S26,S57,S98,S110
			3.3.2	Automatable Governance Action Types	6	S21,S57,S88,S98,S110

3.3 Federated Computational Governance

Data mesh uses a decentralized governance model in which sovereign domains govern their data products. Overarching governance on a global level across the mesh of the data products mainly aims to ensure interoperability between data products and to enforce standardizations and organization-wide policies. Balancing the levels of centralization and decentralization can be challenging in the federated governance model. On the one hand, centralization gives more control to ensure interoperability and compliance. On the other hand, the domains need sufficient autonomy to build data products in line with their business.

ID3.1. Global Governance. We refer to the governance activities conducted by the federated governance team on the data mesh level as global governance. The federated governance team comprises representatives from the domains, subject matter experts, governance experts, members from the legal department, the platform team, the management, and so on. The common global governance activities include:

ID3.1.1. Defining and Enforcing Organization-wide Standards. A key goal of standards is to increase the interoperability of data products. Examples are a shared API description language and a standard language for metadata modeling. Standards are also used to ensure data products adhere to security and privacy requirements modeled by the federated governance team.

ID3.1.2. Defining Governance Policies. Data governance policies define rules or guidelines that enforce consistent and correct collection, storage, access, usage, and management of data assets in an organization. For example, the governance policies at the data mesh level can enforce a minimum level of data security throughout the organization or enforce access control based on central identity management.

ID3.1.3. Developing a Methodology to Define and Assess Data Quality. The federated governance team is also responsible for creating a methodology to define data quality to be enforced by data product owners. The methodology consists of dimensions contributing to data quality (e.g., timeliness of data and completeness of data). A standardized methodology can ensure that all data products define quality on a common set of quality dimensions with shared definitions.

ID3.1.4. Maintaining a Common Business Glossary. Organizations use business glossaries of business terms and their definitions to ensure the same definitions are used throughout the organization when analyzing data. The data mesh principles discourage creating one canonical data model for the whole organization, which goes against decentralization. However, the gray literature recognizes the importance of having a shared definition of concepts used throughout the organization (e.g., a customer identification number) as this contributes to interoperability. The federated governance team is responsible for business glossary modeling.

ID3.1.5. Monitoring Data Mesh. The federated governance team should continuously observe and asses the overall health of the data mesh. For this purpose, It can collect the necessary data from the data mesh to calculate the KPIs (Key Performance Indicators), such as levels of interoperability, compliance, and usage of data products.

ID3.1.6. Creating Incentive Models. Provisioning data products will add to the domain teams' normal workload and will likely not fit their traditional job descriptions. For example, a supply chain department, whose main task is to optimize delivery routes, may not be keen on providing high-quality data to other domains. Without proper incentives, domains might not see a reason to build data products. Thus, organizations implementing or transforming to a data mesh architecture should develop an incentive model to motivate domains to provision their data as products and continuously govern the offered products.

ID3.2. Local Governance. We refer to the governance activities conducted by the product/domain teams on a product/domain level as local governance. The product owner, potentially in combination with other domain members, is responsible for conducting the local governance activities. The gray literature mentions several responsibilities of the local governance team:

ID3.2.1. Managing the Data Models of the Product. The product team uses its domain knowledge to define the data model of the product. The model schema can be changed and extended throughout the product lifetime to reflect business capabilities in the domain better.

ID3.2.2. Managing Data Access Control. Domain experts have a thorough understanding of the data and can, therefore, best judge who should and should not have access to (which parts of) the data.

ID3.2.3. Managing Compliance and Conformance. Like managing access controls, ensuring compliance and conformance throughout the product's lifetime requires a thorough understanding of the domain data.

ID3.2.4. Managing Data Quality. Using the quality assessment methodology defined by the federated governance team, the product team establishes the product's target quality based on a set of quality dimensions.

ID3.2.5. Monitoring Data Product Health. Throughout the product lifetime, the product owner is responsible for monitoring its operational health and ensuring its capacity is sufficient to meet the needs of consumers.

ID3.3. Automation of Governance Activities. Automation plays a significant role in the federated computational governance model to ensure it can effectively scale throughout an organization. Especially when data leaves the scope of a data product and thus the control of the product owner, automated decisions can be used to enforce policies in downstream consumers. The gray literature gives several examples of actions that can be automated in a data mesh architecture: data classification (e.g., sensitivity labels), data anonymization and encryption, data quality checks, data retention and deletion to ensure compliance, and automated monitoring, including alerts and automatic responses (e.g., scale a resource to meet demand).

Table 5. Self-Serve data platform: codes and sources.

Sub Category			Atomic Codes			
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Unique Sources**
4.1	Objectives	24	4.1.1	Reduce Required Specialization	8	S1,S4,S9,S25,S43,S72
			4.1.2	Increase Efficiency	9	S2,S7,S9,S43,S53,S55,S65,S71,S92
			4.1.3	Enable Uniformity and Interoperability	7	S1,S19,S33,S88,S93
4.2	Building Platform	19	4.2.1	Central Platform Team	7	S1,S2,S22,S58,S73,S85,S86
			4.2.2	IaC Blueprints	12	S7,S13,S53,S54,S93,S97,S98,S105,S108,S111
4.3	Platform Components	149	4.3.1	Compute	7	S2,S31,S37,S40,S53,S64
			4.3.2	Networking	8	S40,S50,S87,S102,S105
			4.3.3	Polyglot Storage	13	S2,S3,S7,S10,S31,S37,S53,S58,S80,S97,S98
			4.3.4	Services for Data Product Components	22	S2,S3,S7,S12,S31,S37,S42,S58,S64,S72,S73,S98,S108
			4.3.5	Metadata Repository	8	S31,S47,S64,S71,S83,S90,S113,S114
			4.3.6	Product/Data Catalog	15	S2,S4,S12,S31,S32, S35,S37,S64,S71,S80,S81,S88,S98
			4.3.7	Distributed Query Engine	6	S16,S31,S32,S37,S98,S110
			4.3.8	Monitoring	10	S2,S4,S13,S33,S35,S37,S42,S64,S71,S72
			4.3.9	Product Lifecycle Management	10	S2,S4,S21,S31,S53,S64,S111,S113
			4.3.10	Security and Privacy	22	S2,S4,S7,S12,S13,S28,S31,S33,S29,S37,S50,S71,S73,S80
			4.3.11	Policy Enforcement	14	S1,S2,S4,S21,S26,S32,S53,S57,S82,S88,S98,S110
			4.3.12	BI Tools	9	S10,S31,S47,S64,S66,S70,S98,S108

3.4 Self-Serve Data Platform

Data mesh advocates building domain-agnostic infrastructure and platform services and offering them in a self-service manner to empower different actors involved in creating, deploying, and maintaining data products. By infrastructure services, we meant those related to providing and managing resources such as VMs, networks, and disks. The platform services support design, development, deployment, and management of data products, governance applications, and their components (e.g., data pipelines, ML pipelines, and microservices).

ID4.1. Objectives. According to the gray literature, the self-serve data platform has several objectives. First, it aims to reduce product developers' required level of specialization as they do not need to possess in-depth knowledge of managing infrastructure and platform services. Second, it

can increase the efficiency of developing and managing products as there is no duplication of efforts from each domain to manage platform services. Last, the common platform services can improve the uniformity of data products, reduce friction between domains, and facilitate interoperability between data products.

ID4.2. Building Self-serve Data Platform. The platform is built and maintained by a centralized data platform team of highly specialized infrastructure and platform service developers. The central platform team has to be in close contact with all domains to identify and prioritize the requirements for platform services, their features, and their interfaces. The gray literature recommends using IaC (Infrastructure as Code) blueprints to simplify the provision of data products using the self-serve platform. The IaC templates encompass a predefined configuration for infrastructure resources and platform services and integrated security measures, policies, and standards in line with those set by the federated governance team. For example, the data platform team can develop IaC templates for different data product types (e.g., data pipelines, ML pipelines, and ML model services) and different product component types (e.g., a data ingestion component using the Kafka message broker, a data store component using Google object store, and a training pipeline executor component using the AWS SageMaker). The product developers can find the correct IaC templates, customize them as necessary, and use the updated templates to provision and manage their data products.

ID4.3. Components of Self-Serve Data Platform. From the gray literature, we identified the following key services that self-serve platforms should offer.

ID4.3.1. Computing. The platform should offer different computing resources, including hardware accelerators to efficiently execute complex data processing tasks, ML training jobs, and hosting product APIs.

ID4.3.2. Networking. The platform should offer various networking capabilities such as virtual private networks, domain name servers, network firewalls, and load balancers.

ID4.3.3. Polyglot Storage. Data products might contain different data types that require different data stores. For example, a data product might consist of highly structured, tabular data stored in a relational database. In contrast, other data products might consist of highly unstructured data that can best be stored in an object store (e.g., videos or images). Thus, the platform needs to support storing and serving polyglot data.

ID4.3.4. Services for Data Product Components. Data products generally consist of components such as data ingestion, data (ETL) pipeline, ML training pipeline, ML model storage, ML feature storage, message broker, and API. The product developers should be able to use platform services to implement and manage such product components. For example, the developers should be able to use the pipeline orchestration tools from the platform to develop, deploy, schedule, execute, and monitor data and ML pipelines in a self-service manner.

ID4.3.5. Metadata Repository. Products, their components, and their artifacts (e.g., ML models and database tables) have various metadata; thus, the platform needs a service to generate, store, and manage metadata. Moreover, company-wide metadata, such as standardized definitions of business terms, should also be available so product developers can incorporate them into their data products.

ID4.3.6. Product/Data Catalog. The platform can provide a product catalog service to publish and discover data products. The metadata repository platform service can be used to develop the product catalog.

ID4.3.7. Distributed Query Engine. The self-serve platform should offer a distributed query engine that can query data from different domains to enable product aggregators to create composite data products.

ID4.3.8. Monitoring. The platform should offer tools that enable domains to monitor their data products and the federated global governance team to monitor the data mesh as a whole. The tools should include but not be limited to functionality that enables monitoring of the operational health, lineage, costs, and performance concerning data quality metrics. They should also support logging, alerts, and automated interventions (e.g., scaling up resources). Finally, the monitoring tools must be integrated with the other relevant platform services, for example, adding monitoring capabilities to data pipelines and ML pipelines.

ID4.3.9. Product Lifecycle Management. The platform should offer capabilities (e.g., product versioning, source control, and testing) to allow product developers to manage the lifecycle of a data product. Developers should be able to apply DevOps [16] practices when deploying and operating data products. For example, they should be able to use CI-CD (Continuous Integration and Continuous Delivery or Continuous Deployment) platform services to atomically build and test the data pipelines and APIs used by the products, deploy them using the pipeline orchestration and container hosting services of the self-serve platform.

ID4.3.10. Security and Privacy. The platform services must enable realizing data products' security and privacy requirements. For example, the platform can include a data encryption service to secure data at rest and in motion. In addition, an identity and access management (IAM) service can be provided to manage user and product identities and enforce access control policies for data products and other platform services.

ID4.3.11. Policy Enforcement. The platform should offer tools that allow local and global federated governance teams to define, store, attach, observe, and enforce governance policies. The gray literature recommends using the *policy-as-code* approach, where policies are defined, evaluated, and managed programmatically. Moreover, the platform also needs to provide services to support policy implementation, e.g., data anonymization service and data quality metrics calculation service.

ID4.3.12. Business Intelligence (BI) Tools. The platform should enable business users to explore data, create visualizations, generate insights, and create reports in a self-service manner.

Table 6. Benefits, concerns, and applicability: codes and sources.

Sub Category			Atomic Codes			
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Unique Sources**
5.1	Benefits	48	5.1.1	Scalability	12	S3,S4,S12,S13,S18,S25,S28,S48,S60,S62,S69
			5.1.2	Increased Agility	7	S3,S25,S28,S40,S92
			5.1.3	Higher Data Quality	5	S3,S9,S10,S23,S25
			5.1.4	Better Data Discovery	8	S2,S28,S40,S50,S60,S87
			5.1.5	Better Governance	6	S3,S10,S12,S23,S28,S95
			5.1.6	Reduced Data Lead Time	10	S3,S5,S12,S25,S28,S53,S69,S77
5.2	Concerns	32	5.2.1	Change management	13	S3,S16,S21,S25,S28,S35,S58,S63,S77,S87
			5.2.2	Lack of Talent	5	S16,S41,S46,S48,S77
			5.2.3	Data Duplication	9	S3,S18,S24,S28,S35,S41,S48,S109
			5.2.4	Effort Duplication	5	S4,S9,S18,S35,S85
5.3	Applicability	37	5.3.1	Large and Diverse Data Landscape	20	S3-S5,S7,S9,S10,S12,S22,S25,S26,S58,S62
			5.3.2	Need for Agility	7	S3,S4,S5,S12,S22,S35
			5.3.3	Need for Better Data Governance	10	S3,S4,S7,S9,S12,S58,S98

4 RQ2. What are the Benefits and Concerns of adopting Data Mesh?

ID5.1. Benefits. From the gray literature, we identified the following potential benefits of data mesh compared to traditional data architectures.

ID5.1.1 Scalability. A data mesh architecture is much more scalable than a centralized data architecture since dependencies on a central management platform team are reduced, which allows the organization to scale horizontally (i.e., more data sources) and vertically (i.e., more consumers) without creating a bottleneck.

ID5.1.2. Increased Agility. The benefits mentioned above make organizations more agile as they can deliver new data applications at a faster rate and can thus respond quickly to changing needs and goals. Overall, organizations can create more data-driven applications that unlock the value of their data.

ID5.1.3. Higher Data Quality. By making those close to the origin of the data responsible for offering it as a data product, accountability for data quality is established. The federated governance team defines a set of global standards and policies that the domains must adhere to when creating data products. Furthermore, domain teams can use their domain-specific knowledge to build data products valuable to various customers.

ID5.1.4. Better Data Discovery In the data mesh, domain teams are incentivized to maximize the usage of domain data products and hence to make the products easily discoverable. Therefore, they create self-describing data products and publish product metadata to the central product registry/catalog for easy discoverability.

ID5.1.5. Better Governance. Data mesh also allows for better big data governance. The decentralized approach is more suitable for governing a high volume of data by distributing governance responsibilities to the domains close to the origin of the data. As the domains better understand their data, they can better judge who should and should not be able to use certain data in certain conditions and facilitate organizations to comply with external regulations.

ID5.1.6. Reduced Data Lead Time. Lastly, domains can independently provision and maintain data products, and consumers can independently use these products through the self-service model, dramatically reducing the data lead time throughout the organization.

ID5.2. Concerns. Data mesh comes with their own set of challenges. Below, we present an overview of the main concerns for the data mesh recognized by the gray literature.

ID5.2.1. Change management. Moving to a data mesh architecture will challenge established organizations from technical and organizational perspectives. Organizations already have established data management processes and a hierarchical organizational structure that will likely need to be restructured.

ID5.2.2. Lack of Talent. Data mesh requires domain teams to start operating independently, so the teams need members with the necessary technical skills (e.g., data engineering and machine learning). While a vital goal of the self-serve platform is to empower the developers of less technical expertise to develop data products, building self-serve platform services can be challenging and require time and effort for research and development.

ID5.2.3. Data Duplication. As domains repurpose data for new use cases in composite data products, data from source domains may be copied and duplicated. In particular, when implemented on a large scale, data mesh can potentially increase data management costs. Furthermore, as data is stored in different organizational locations, it will become more challenging to govern data.

ID5.2.4. Effort Duplication. Even though the self-serve platform tries to reduce the duplication of efforts concerning data platform management, there will still be a certain degree of duplication of efforts and skills for building and maintaining pipelines throughout the different domains compared to a team working on centralized data.

ID5.3. Applicability. Data mesh may not be the most practical architecture design for all organizations. The gray literature recognizes several conditions for which the data mesh is more suitable than a centralized data architecture.

ID5.3.1. Large and Diverse Data Landscape. Organizations with a large data landscape with many data providers and consumers might benefit from a decentralized architecture. Also, when organizations possess diverse data sets that frequently change over time, the data mesh might serve better than a centralized approach with many point-to-point connections.

ID5.3.2. Need for Agility. Organizations whose data platform and central data provisioning teams reduce their innovative capabilities and require a faster time to market for data applications can benefit from a data mesh architecture. Also, when there is a need to experiment with data frequently, data mesh is more beneficial than a centralized architecture as it removes the dependency on an oversized and overloaded data provisioning team.

ID5.3.3. Need for better Data Governance. Centralized data lakes can quickly become data swamps without data ownership and governance model. Data mesh can resolve these issues for organizations needing clear data ownership and highly governed data based on domain knowledge. In addition, organizations that face many external regulations can use data mesh to ensure compliance as the data mesh facilitates better data governance.

5 RQ3. How should an Organization Build a Data Mesh?

This section presents a set of reference architectures to describe the data mesh architecture in detail. We leveraged the relevant architectures used in the service-oriented computing domain and the insights from our gray literature review.

5.1 A Layered Architecture of Capabilities and Roles in Data Mesh

This section introduces a layered model to logically organize different constructs, roles, and responsibilities in a data mesh architecture. We mapped the relevant findings from our gray literature study into the xSOA model [33, 34], which aims to group and logically structure the capabilities of complex SOA applications. Figure 7 shows the layered model of the data mesh. The separation into layers highlights the different needs regarding i) providing the infrastructure and platform services necessary for realizing and consuming data products, ii) creating, consuming, and composing data products, and iii) managing data products throughout their lifecycle. It is important to highlight that although higher layers build on lower layers, they only represent a separation of concerns and do not imply a hierarchy.

The layered architecture shown in Figure 7 does not depict all responsibilities in a data mesh architecture. Instead, it provides a view of the responsibilities needed to create individually managed data products. For example, the federated governance team's responsibility to create a business glossary is not included in this architecture because this is not related to building a specific managed data product at runtime.

Roles. We identified six different roles for data mesh actors (see Table 7):

Data Platform Teams are responsible for building and maintaining the self-serve platform. They also create the IaC blueprints that product developers can use to provision and configure infrastructure resources and platform services for hosting and managing data products.

Product Owners are responsible for offering and governing the data products in their domains. They must ensure that the data products are interoperable and meet the requirements of the consumers.

Product Developers build and publish data products on the data mesh using the tools/services the self-serve platform provides.

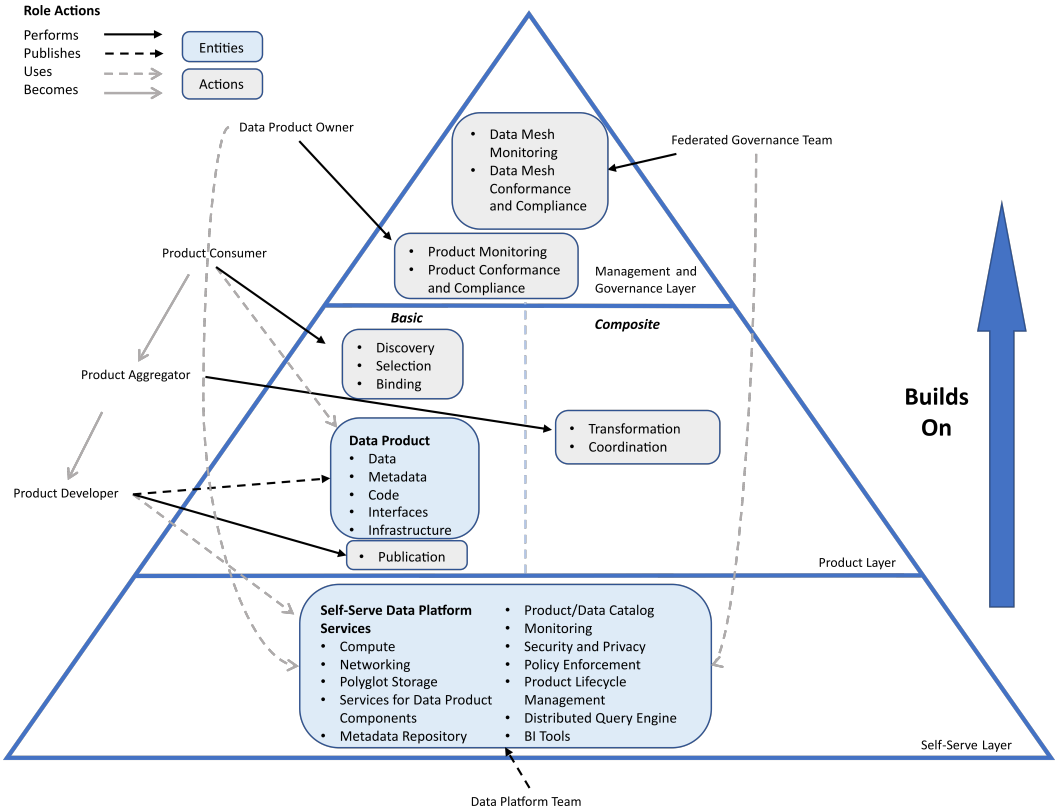

Fig. 7. The layered architecture of capabilities and roles in data mesh.

Product Aggregators build and provide composite data products. The gray literature does not explicitly mention the aggregator role. We introduced it to clearly distinguish between the responsibilities and capabilities of building source-aligned atomic data products from composite data products.

Product Consumers use data from the data products. Product aggregators are also product consumers since they consume multiple other data products and aggregate them into composite data products.

Federated Governance Teams comprise representatives from different domains across the organization and governs the mesh globally.

Table 7. Roles in data mesh: codes and sources.

Role	Frequency	Unique Sources
Data Platform Team	7	S2,S22,S42,S73,S85,S86
Product Owner	17	S1,S2,S3,S16,S18,S25,S33,S35,S43,S49,S67,S73,S82,S86-S88
Product Developer	10	S1,S2,S16,S18,S21,S35,S39
Product Consumer	9	S10,S13,S19,S54,S91
Federated Governance Team	9	S1,S16,S21,S25,S28,S110

Self-Serve Layer. Self-serve platforms offer a set of tools/services in a self-service fashion that the actors involved in a data mesh architecture can use to fulfill their responsibilities (see Section 3.4). For example, product developers can use the platform services to develop, test, deploy, and evolve their data products. In contrast, the product owners and the federated governance team can use platform services to monitor and govern individual data products and the overall data mesh.

Data Product Layer. The data product layer sits on top of the self-serve layer. As discussed in Section 3.1, data products consist of data, metadata, code, interfaces, and the infrastructure to deploy the product. Once the product developers define and implement all components, the data products can be published in a product catalog. Product consumers browse the catalog to discover candidate data products for their use cases and check the corresponding metadata to verify that they suit their needs. Product meta-data can also enable consumers to automatically connect (binding) to the data products to obtain their data.

Composite data products use data from other (basic or composite) data product(s), potentially apply complex transformations, and publish the results as new data products (see Section 3.1). Composite data products contain the same components as basic data products. However, product aggregators perform additional actions to build composite data products, similar to the additional responsibilities in xSOA [34]. To distinguish between these responsibilities, the layered architecture in Figure 7 differentiates between product developers, who build basic data products, and product aggregators, who create composite data products.

Composite data products need to provide new value to the organization. A simple integration of two data sources can easily be replicated and may not provide sufficient additional value. Therefore, a product aggregator may need to perform complex transformations on the data from upstream data products to create value-added data. Furthermore, composing products may involve coordinating a series of steps, e.g., ingesting data from multiple data products, merging the data from a subset set of products, cleaning and standardizing data, and storing intermediate and final results.

Composite data products must conform with the upstream data products throughout their lifetime. For example, upstream product developers might change their data products (e.g., schema and product interfaces), which can affect downstream consumers. Product aggregators must deal with such issues by adopting suitable strategies such as product versioning and using an anti-corruption layer (an adapter layer).

Management and Governance Layer. Data products must be managed and governed at runtime to ensure they provide value to the organization. As discussed in Section 3, there are two levels of governance: local and global. The local governance (per each product) is responsible for the quality and conformance of individual data products. For example, the local governance team must ensure that a data product operates within the specified SLOs throughout its lifetime and remains compliant with local and global policies and external regulations. The owners of the composite products also need to monitor the performance of upstream data products in terms of service level objectives (SLOs). The performance of the composite data products can be affected by that of upstream data products. Thus, a product aggregator needs to monitor the performance of upstream products to determine any SLO violations and ensure the composite products operate within its SLOs. The main goal of global governance at runtime is to ensure that the mesh architecture can scale throughout the organization effectively. The federated governance team monitors the mesh to assess the data mesh's overall health and enforce the global policies, interoperability standards, and guidelines.

5.2 A Layered Architecture for Data Mesh Development

The data mesh literature does not explicitly provide an end-to-end methodology for implementing data mesh in or across organizations. However, the literature advocates applying the principles of

domain-driven design (DDD), product thinking, and platform-based development. In particular, data products should be identified by applying DDD to the data produced by the business domains of the organization (see Table 8). Inspired by a layered model for SOA development [32], we introduce a layered model for data mesh development that integrates the previously mentioned principles. Figure 8 shows the proposed layered approach to data mesh development. The model comprises seven distinct layers: business domains, data domains, data-bounded contexts, data products, data product components, self-serve platform services, and operating infrastructure.

Table 8. Applying domain-driven design to develop data products: sources and code frequency.

Concern	Frequency	Unique Sources
Applying Domain-Driven Design	28	S1,S2,S11,S36,S37,S39,S45,S46,S53,S55,S58,S71,S75, S78,S79,S87,S90,S91,S96,S98,S103,S110,S111,S112
Business Domain and Data Domain Alignment	14	S36,S37,S39,S46,S53,S55,S56,S71,S75, S78, S87, S111, S112
Bounded Contexts for Data Products	14	S1,S35,S45,S79,S87,S90,S98,S112

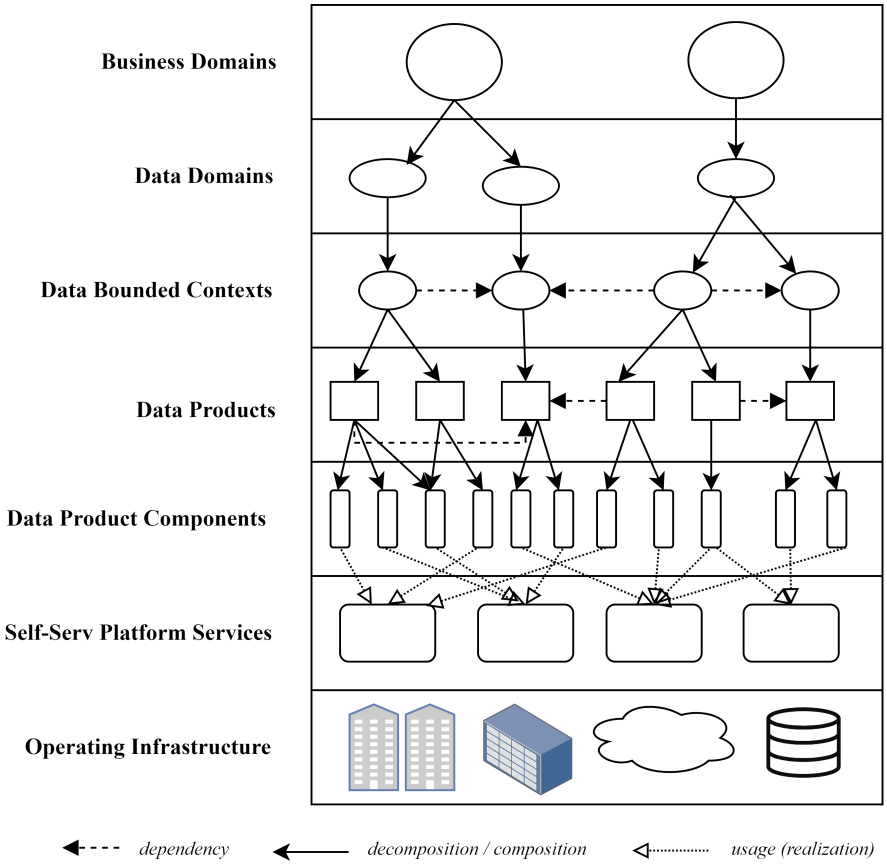

Fig. 8. A layered model for the data mesh development.

Business Domains. Following the DDD method, the first layer decomposes an organization's business into disjoint domains. A business domain consists of a set of business processes that shares a common set of business capabilities and collaborate to accomplish the business goals and objectives of the domain [32]. These processes produce valuable data for the domain and the rest of the organization and can thus be offered as data products. For example, a manufacturing organization may include domains such as distribution, manufacturing, finance, and human resources [32]. The distribution domain may comprise key business processes such as consumer purchasing, order management, and inventory. The purchasing process may produce data containing information about products or services, time of purchase, and the amount spent. This data may be exploited to build data products to answer queries about purchase history, customer buying patterns, and other relevant details, such as stock availability and product appearance.

Data Domains. The second layer partitions the analytical data of a business domain into one or more data domains. Data domains define boundaries around the analytical data. From the viewpoints of data mesh and data governance, data domains categorize and organize data to assign accountability and responsibility for the data. As discussed above, different business processes in a business domain may produce multiple data. Due to the diversity and complexity of processes, the responsibility for each process may be assigned to a separate team [9]. Similarly, the analytical data of processes may be owned by different teams, originating multiple data domains from the same business domain. The data mesh gray literature also recommends aligning business and data domains but creating new domains for shared data as appropriate.

Data-bounded Contexts. The third layer identifies the bounded contexts within each data domain. DDD designs and implements software systems based on domain models, which are the object models of the domains that incorporate both behavior and data in the domain [17, 18]. Creating and maintaining a unified model for larger and more complex domains, where multiple teams work on separate subsystems, may not be feasible or cost-effective [17, 18]. To tackle this issue, DDD partitions the solution space of a domain into bounded contexts, where each can have a separate domain model. A data domain can also be complex and include multiple logical groupings of data. Each group can have a separate data owner and model (i.e., bounded contexts for decomposing data domains). For example, there may be different categories of customers: online only, physical only, third party, prospect, individual, group, corporation, government, and charity. Managing data for each or subset of these categories might differ, and multiple teams or individuals might be responsible for it. Consequently, multiple bounded contexts emerge within a customer data domain.

Various dependencies (design time and runtime) may exist between data bounded contexts. The gray literature recommends using the strategic design patterns of DDD (e.g., customer-supplier and anti-corruption layer) to integrate bounded contexts (i.e., context mapping). For example, a bounded context of a composite product can use the anti-corruption layer to minimize the impacts of the changes to the data models of the data products from the upstream bounded contexts.

Data Products. The fourth layer identifies data products within each bounded context. For building applications for a bounded context, DDD recommends applying so-called tactical design patterns: entity, value object, aggregate, service, repository, factory, event, and module. These patterns can be used to enrich a domain model of a bounded context, and the enriched model can be used to derive the application architecture components. For example, in a microservice architecture, services, APIs, and events can be deduced from the domain model using the tactical design patterns [40]. The data mesh gray literature also recommends applying these patterns, for example, using domain main events for ingesting and serving data and implementing a log of analytical data changes, and using entities and aggregates for identifying products and creating database schemas and product APIs.

Data Products Components. The fifth layer consists of the components that implement functional and non-functional requirements of data products. There are different types of data products, for example, datasets (offered as database views or APIs), data pipelines, ML training pipelines, and ML prediction services. Different data products may use different architectures. For example, data pipeline products may use popular data architecture styles such as Lambda and Kappa [11], and ML prediction services may use architecture styles as model-as-a-service or model-as-a-dependency [45]. The resulting data products can have components like streaming data ingestion, ETL workflows, storage of raw, cleaned, and enriched data, model and feature storage, APIs, data quality assessors, and report generators. Data products can also share the implementation of some components, for example, a generic data validating service or model validation service.

Self-serve Platform Services. The sixth layer consists of the self-serve platform services that support the end-to-end lifecycle of data products and their components, from design, implementation, testing, deployment, execution, and monitoring, to evolution. For example, data product developers should be able to use an ETL platform service in a self-service manner to create data ELT pipelines for their data products, deploy and execute the created pipelines on-demand or according to a schedule, monitor the performance, resource usage, and errors of the executed pipelines, and make changes to the pipeline as necessary. Section 3.4 lists common platform services we found from the gray literature. The platform team is responsible for self-serve platform services. They may develop platform services from scratch, reuse and customize third-party software, use the platform services offered by cloud providers, and modernize existing platform services by incorporating self-service capabilities.

Operating Infrastructure. The seventh layer mainly includes the infrastructure resources used by the platform services and data products and the existing systems in the organization that act as data source systems.

In a data mesh, management and governance applications are used by the federated governance team, in addition to data products, to monitor and control data products. Figure 8 does not explicitly include those applications. However, since those applications may use the monitoring data of the data mesh, developers may develop and offer them as data products. Furthermore, the management applications should also be able to use self-serve platform services.

5.3 A Logical Runtime Structure of Data Mesh

This section presents a logical runtime view of a data mesh. The data mesh architectures presented in the gray literature source primarily focus on the runtime model of an inter-connected web of managed (governed) data products across data domains (see Table 9). We extracted the essential components from those architecture designs and created a consolidated model by consulting web service management architectures [24, 26, 32, 33, 48]. Figure 9 depicts the reference architecture at runtime for data mesh.

A product container is a deployment and runtime environment for a data product and its managing agents. We borrowed the concept of a container and a management agent from Web services containers and management agents [32]. As in the web service management architectures [26, 32, 33], we separate the communication between data products, source systems, and end-users (data channel and input/output interfaces) and the communication between the management agents and the management/governance plane (management channel and management interface). The input interface enables data products to receive data from source systems and other data products. The output interface allows data products to provide their data to consumers or composite data products. Finally, the management interfaces enable monitoring and controlling the behaviors of data products (e.g., monitor product performance, raising alerts, controlling data routing, and enforcing security

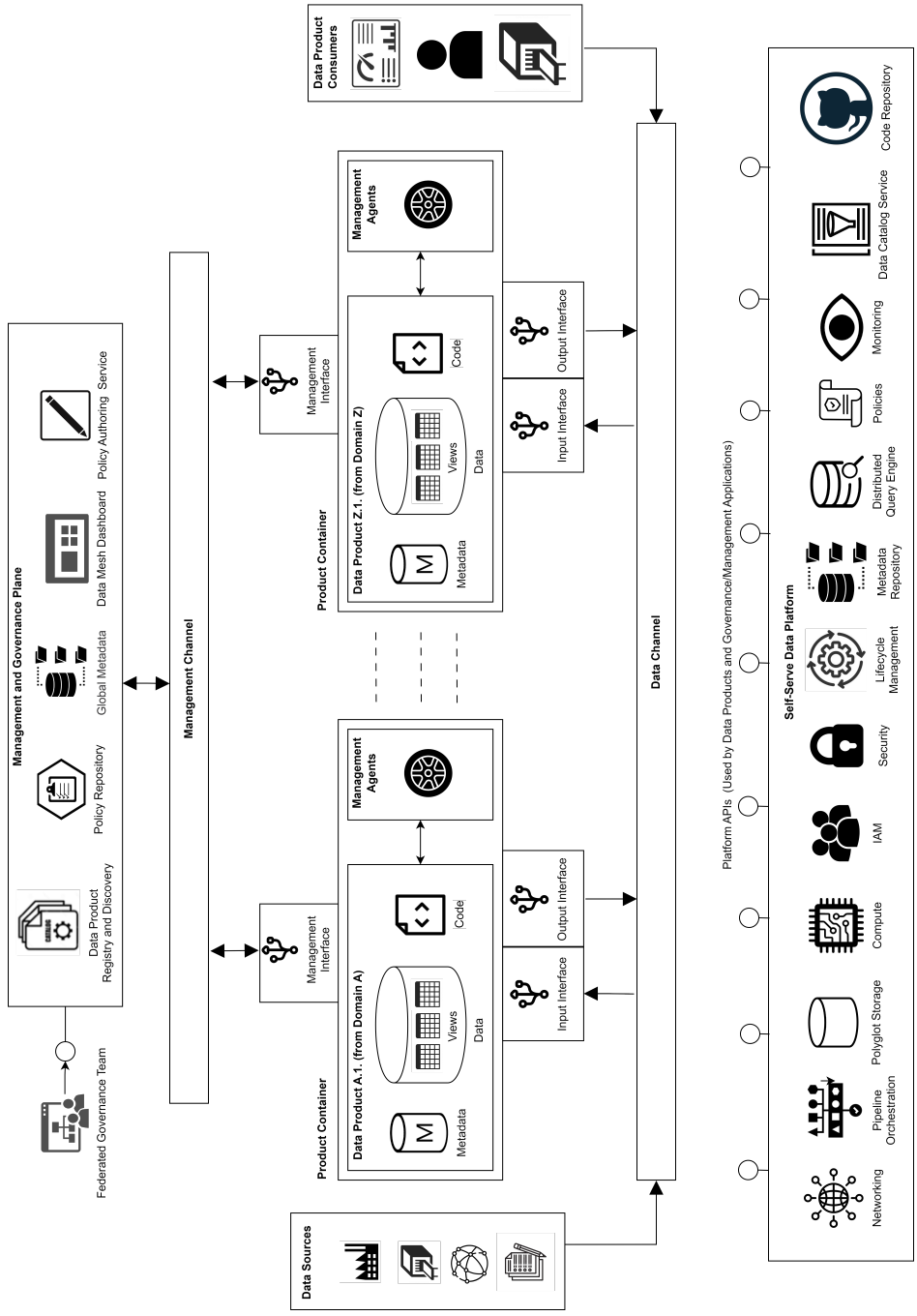

Fig. 9. Logical runtime structure of a data mesh.

policies and data contracts). The modes of communication can be pull-based (e.g., retrieving data

Table 9. Logical runtime architecture: codes and sources.

Sub Category			Atomic Codes			
ID	Name	Frequency	ID	Name	Frequency	Unique Sources
6.1	Overall Structure	21	6.1.1	Logical Structure with Three Layers (Self-serve Platform, Products, and Governance)	21	S1,S8-S10,S14,S29,S35,S42,S47,S53, S58,S70,S71,S78,S80,S86,S99,S110
6.2	Product Interfaces	25	6.2.1	Input	9	S51,S66,S72,S86,S97,S103,S108,S110
			6.2.2	Output	9	S51,S66,S72,S86,S97,S103,S108,S110
			6.2.3	Management	7	S86,S97,S103,S110
6.3	Communication Channels	40	6.3.1	API	12	S1,S64,S71,S75,S78,S82,S83,S90,S108,S111
			6.3.2	Message/Event Bus	18	S24,S29,S64,S78,S82,S83,S84,S86,S90,S108,S110,S111
			6.3.3	Shared Storage	10	S13,S24,S86,S87,S89,S97,S98,S100,S112,S108
6.4	Management Agents	13	6.4.1	Sidecar Proxy	9	S29,S36,S78,S83,S113
			6.4.2	Policy Agent	4	S2,S29,S78,S97

via a REST API or a database connector or polling data from a message broker) or push-based (e.g., pushing the data to the downstream consumers as the data become available).

The management agents of a product implement management tasks at the product level, such as marking and enforcing policy decisions, collecting performance metrics, and managing the product life cycle (i.e., creating, updating, and destroying products). The gray literature recommends using the sidecar design pattern and the policy-as-code approach to implement the monitoring and policy enforcement capabilities of the management agents. The sidecar pattern [4] decouples policy decision-making from the data product and can enable reuse and consistent implementation of monitoring and management logic across data products. The policy-as-code approach uses programming code to write policies, allowing applying software development best practices such as version control, automated testing, and automated deployment. A policy manifests as a set of rules that dictate the behaviors of the management agents, for example, how to enforce authentication and authorization policies for data product access, when to raise alerts, and where to route messages.

The components/applications in the management and governance plane can use the management interfaces of products to receive information (e.g., performance data, access logs, and alerts) from data products and to send configuration and policy updates to the data products. We identified several key components for this plane from the gray literature: data product registry and discovery service, policy authoring service, policy repository, global metadata repository, and data mesh dashboard. The product registry and discovery service is connected to the management channel. Product developers publish their products to it over the management channel. Consumers can discover data products by using the published metadata of data products. The federated governance team can create global policies (e.g., interoperability, security, and compliance) using the policy authoring services, store them in the policy repository for reuse and versioning, and enact them using the management interfaces provided by the individual products. The team can use the data mesh dashboard to monitor the operational health of the data mesh, including the cross-domain access and usage of data products and level of policy compliance.

The self-serve data platform plane provide infrastructure resources and platform services to develop, test, build, deploy, execute, monitor, and manage data products and governance components. For example, a data product can use the ML workflow platform service to create and run the training pipelines. The product registry and discovery service can use data/product catalog or metadata repository platform services. Section 3.4 discusses the key platform services of a self-serve data platform.

6 RQ4. What are the Research Challenges concerning Data Mesh?

In this section, we put forward a set of research challenges for data mesh that need to be addressed. We identified them by mapping the challenges mentioned by the practitioners in the gray literature to

the relevant research issues from the service-oriented computing [33, 34, 46] and data management [2, 15, 20, 42]. Table 10 shows the challenges proposed by the practitioners.

Table 10. Data mesh challenges: sources and code frequencies.

Challenge	Code Frequency	Unique Sources
Standardizing Data Mesh	6	S8,S35,S38,S61,S110,S111
Methodologies and Tools for Data Mesh Development and Operation	18	S1,S4,S28,S29,S35,S42,S98,S111
Data Product Life Cycle	14	S1,S35,S45,S79,S87,S90,S98,S112
Self-serve Platform Services	7	S2,S16,S33,S43,S53,S86,S108
Data Mesh Governance	7	S1,S28,S35,S86,S111
Organizational Change Management	8	S2,S16,S28,S33,S35,S61,S87

Standardizing Data Mesh. The data products in a data mesh should be interoperable to enable the seamless composition of products and foster cross-domain collaboration. The data products also need to be portable so that self-serve platform services and governance applications can be modified or replaced with little or no modification to data products. The standardization of the data mesh can enable its interoperability and portability. The standards are necessary for i) defining data products, their interfaces, and their composition; ii) data product publishing and discovery, quality assessment, deployment, and governance; iii) self-serve platform services and their interfaces; and iv) artifacts used/produced by data products (e.g., data models, metadata, and ML models). The standardization efforts in the SOA and cloud domains can potentially help address these challenges.

Methodologies and Tools for Data Mesh Development and Operation. The data mesh literature provides general principles for developing a data mesh architecture, e.g., domain-driven design, product thinking, platform thinking, and computational governance. However, no tools and detailed (empirically validated) methodologies exist for applying these principles to develop and operate a data mesh. Organizations would also need guidance and tools for systematically migrating their legacy data architectures into data mesh while assessing costs and benefits.

Data Product Life Cycle. Data products are at the heart of a data mesh. Each phase of a data product life cycle (e.g., identification, design, development, testing, deployment, discovery, composition, operation, monitoring, and evolution) can have open challenges. For example, the product identification phase decomposes the analytical data landscape of an organization into a set of data products. The partitioning process needs to support applying domain-driven design correctly, ensure the modularity and utility of data products, and consider reuse of the artifacts of the existing data architecture, such as data source systems, data pipelines, machine learning pipelines, documentation, and execution logs. Further exploratory research studies are needed to properly understand the research challenges in each phase of the product life cycle. Due to the comparability of a data product and a service, we believe the existing relevant research studies on SOA and microservices can help understand and address the above product challenges.

Self-serve Platform Services. The domain teams need to be able to use domain-agnostic platform services in a self-service manner to easily and independently build and manage their data products. While there exist research studies on self-serve systems, e.g., self-service BI (business intelligence) [25] and self-service portals for cloud applications [31], they do not cover each platform service type, e.g., data pipelines, ML pipelines, and IaC based data product provisioning. Moreover, the recommender systems can also be incorporated into platform services to assist users with building data products, e.g., providing recommendations interactively

when creating a data pipeline or IaC blueprint. Finally, since the organizations may already use data platforms, they would potentially need the guidance and techniques to add self-service capabilities to their legacy data platforms.

Data Mesh Governance. While data governance has been an active research topic for several decades, as noted in several recent reviews [1, 10, 21], there are still many open research issues. Data mesh, especially its principles of decentralized ownership of data and federated computational governance, may add new dimensions to data governance research issues. For example, the efficient composition of data products across domain boundaries (and potentially across organizational boundaries) depends on the ability of the federated governance mechanisms to ensure that data products are interoperable, compliance with QoS (Quality of Service) and policy requirements, and in line with the company-wide data strategy. Moreover, the end-to-end data governance automation using the policy-as-code approach (i.e., computational governance) is largely unexplored.

Organizational Change Management. As discussed in Section 5.1, data mesh introduces a set of new roles and responsibilities for teams and individuals in an organization. In particular, the responsibilities of domain data are shifted closer to decentralized autonomous domain teams. Hence, for organizations that use centralized operating models and organizational designs, embarking data mesh journey may require a significant organizational change (in addition to technological changes). Data mesh can also significantly impact the existing data management and governance processes and practices in an organization. Thus, more studies are necessary to understand the organizational impacts of data mesh, the barriers to the adoption of data mesh, and to create guidelines and frameworks to perform and manage required organizational changes.

7 Related Work

This section reports on the studies concerning data mesh and related surveys.

7.1 Studies on Data Mesh

Several studies on the application of data mesh architectures exist. Loukiala et. al. describe migrating from centralized, monolithic data platform architectures to a decentralized data-mesh-like architecture at a large Nordic manufacturing country [27]. Their work emphasizes concrete steps that can be taken during such a migration and provides high-level contrasts between traditional data warehouse architecture and data mesh. Machado et. al. present several works on data mesh anatomy and architecture. The first is an overview and explanation of data mesh concepts highlighted by two implementations in the industry (an online retailer, i.e., Zalando, and a streaming platform, i.e., Netflix) [3]. Then, in another paper, they introduce a "domain model" that contains the abstract classes of a data mesh and a high-level architecture [28]. Finally, based on these works, they investigate existing (commercial) tools that can be used to realize a data mesh implementation and validate their architectural model through a proof-of-concept implementation demo [29]. Similarly, Butte and Butte [6] present a more-complete reference architecture. However, it is unclear what their architecture is based on, as the paper provides no mThe clearinghouse logs data transactions we know, these are currently the only published academic works explicitly discussing data mesh architecture. However, their descriptions of the data mesh concepts lack a thorough exploration of the state-of-the-art in data mesh. Instead, they are almost exclusively based on the blog posts of Zhamak Dehghani [12, 44], as well as two corporate presentations: Zalando [30] and Netflix [8].

7.2 Related Surveys

Oliveira et. al. investigated the state-of-the-art literature on 'data ecosystems' in 2018 [37]. Data mesh falls under their intended definition of data ecosystems, and their work discusses many elements that are relevant to data mesh. However, none of the studies investigated mention data mesh or any closely related term. Another meta-study that investigates the activities and challenges in big data (eco)systems and their implications on architectures is presented by Davoudian and Liu [11]. Again, the activities and challenges identified in this paper relate directly to those of data mesh. However, the literature discussed herein focuses entirely on traditional data lakes and data warehouses as state-of-the-art. It is only recently that data mesh has started to appear in literature surveys. For example, Driessen et. al. [14] explicitly discuss data mesh, which they consider a special kind of (internal) data market. However, in this work, data mesh is considered only abstractly in the larger context of data-exchanging ecosystems, and no extensive discussion of data mesh architecture is provided. Based on the existing literature, therefore, we conclude that there exists a knowledge gap on data mesh architecture: On the one hand, papers that explicitly discuss a data mesh architecture ([27, 29]) do not consider a broad spectrum of sources on which to base their architecture definition. On the other hand, existing metastudies that look at the state-of-the-art in data ecosystems [11, 14, 37] hardly discuss data mesh sources nor provide a reference architecture.

Given the work above, there are no reviews of the data mesh literature. Moreover, only a little academic literature is available on the topic, primarily based on a few gray articles. Therefore, this review paper systematically analyzed, assessed, and summarized such literature sources.

8 Threats to Validity

This section discusses the potential threats to external, construct, and internal validity [47] that may apply to our study.

First, we note the risk of missing relevant studies for the systematic gray literature review. Since the field is developing rapidly, it can be interesting to repeat the gray literature review after some time to see how the topic matures as it is adopted more by the industry. For this reason, we extended the study to include sources from September 2021 to May 2022, which yielded 35 new sources.

Additionally, as part of the selection process of our methodology, we rely on the (subjective) expertise of the authors to determine the quality and relevance of a source. Such a process runs the risk of introducing bias to the final selection. Therefore, two authors assessed each source, and an inter-reliability kappa coefficient was calculated to ensure bias was not too high.

Finally, the reference architecture was created systematically by looking at various sources. However, as was noted above, few complete, holistic architecture instances exist to compare with, and many sources only consider partial architectures. These partial architecture instances might lead to a sub-optimal holistic reference architecture. At the same time, we note that this problem is somewhat cyclical: a reference architecture can contribute significantly to the number of complete architecture instantiations, which in turn can improve future reference architectures.

9 Conclusion and Future Work

Although enterprises are increasingly becoming data-driven, their centralized monolithic data architectures still prevent them from leveraging the full value of their analytical data. The data mesh is an emerging domain-driven decentralized architecture and sociotechnical paradigm that promises to address the limitations of centralized enterprise data architectures. Even though the data mesh is gaining significant attention in the industry, there is an evident scarcity of academic research on the topic. This paper systematically selects and reviews 114 industrial gray literature on data mesh to identify key design principles, architectural components, organizational roles, benefits, and

concerns. The findings from the review were further enriched with the academic literature from similar domains to create reference architectures for its key dimensions. Finally, we identified several research challenges that need to be addressed to realize the vision of the data mesh. We believe the identified open issues and reference architectures can provide a framework for future research in the data mesh domain.

We are currently working on domain-specific languages for defining data products and architecture decision-marking frameworks for three pillars of the data mesh: data as a product, self-serve platforms, and federated governance. Furthermore, we plan to develop a methodology for designing and instantiating a data mesh architecture by leveraging the reference architectures in the paper.

References

[1] Rene Abraham, Johannes Schneider, and Jan vom Brocke. 2019. Data governance: A conceptual framework, structured review, and research agenda. *International Journal of Information Management* 49 (2019), 424–438. https://doi.org/10.1016/j.ijinfomgt.2019.07.008

[2] Majid Al-Ruithe, Elhadj Benkhelifa, and Khawar Hameed. 2019. A systematic literature review of data governance and cloud data governance. *Personal and Ubiquitous Computing* 23 (2019), 839–859.

[3] Inês Araújo Machado, Carlos Costa, and Maribel Yasmina Santos. 2022. Advancing Data Architectures with Data Mesh Implementations. In *Intelligent Information Systems*, Jochen De Weerdt and Artem Polyvyanyy (Eds.). Springer International Publishing, Cham, 10–18.

[4] Brendan Burns and David Oppenheimer. 2016. Design Patterns for Container-Based Distributed Systems. In *Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing* (Denver, CO) *(HotCloud'16)*. USENIX Association, USA, 108–113.

[5] Bert-Jan Butijn, Damian A. Tamburri, and Willem-Jan Van Den Heuvel. 2020. Blockchains: a Systematic Multivocal Literature Review. *ACM Computing Surveys (CSUR)* 53 (2020), 1–37. Issue 3. http://arxiv.org/abs/1911.11770

[6] Vijay Kumar Butte and Sujata Butte. 2022. Enterprise Data Strategy: A Decentralized Data Mesh Approach. In *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*. 62–66. https://doi.org/10.1109/ICDABI56818.2022.10041672

[7] Jochen Christ, Larysa Visengeriyeva, and Simon Harrer. 2022. Data Mesh Architecture: Data Mesh From an Engineering Perspective. https://www.datamesh-architecture.com/

[8] Justin Cunningham. 2020. *Netflix Data Mesh: Composable Data Processing*. https://www.youtube.com/watch?v=TO_IiN06jJ4

[9] Kjersti Berg Danilova. 2019. Process owners in business process management: a systematic literature review. *Business Process Management Journal* 25, 6 (2019), 1377–1412.

[10] Elizabeth Davidson, Lauri Wessel, Jenifer Sunrise Winter, and Susan Winter. 2023. Future directions for scholarship on data governance, digital innovation, and grand challenges. *Information and Organization* 33, 1 (2023), 100454. https://doi.org/10.1016/j.infoandorg.2023.100454

[11] Ali Davoudian and Mengchi Liu. 2020. Big Data Systems: A Software Engineering Perspective. *ACM Comput. Surv.* 53, 5, Article 110 (sep 2020), 39 pages. https://doi.org/10.1145/3408314

[12] Zhamak Dehghani. 2020. Data Mesh Principles and Logical Architecture. https://martinfowler.com/articles/data-mesh-principles.html

[13] Z Dehghani. 2022. *Data Mesh: Delivering Data-Driven Value at Scale*. O'Reilly Media, Inc.

[14] S.W. Driessen, G. Monsieur, and W. Van Den Heuvel. 2022. Data Market Design: A Systematic Literature Review. *IEEE Access* 10 (2022), 1–1. https://doi.org/10.1109/access.2022.3161478

[15] Schahram Dustdar, Reinhard Pichler, Vadim Savenkov, and Hong-Linh Truong. 2012. Quality-aware service-oriented data integration: requirements, state of the art and open challenges. *ACM SIGMOD Record* 41, 1 (2012), 11–19.

[16] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. 2016. DevOps. *Ieee Software* 33, 3 (2016), 94–100.

[17] Eric Evans and Eric J Evans. 2004. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional.

[18] Martin Fowler. 2012. *Patterns of Enterprise Application Architecture: Pattern Enterpr Applica Arch*. Addison-Wesley.

[19] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology* 106 (2019), 101–121. Issue September 2018. https://doi.org/10.1016/j.infsof.2018.09.006

[20] Marijn Janssen, Paul Brous, Elsa Estevez, Luis S. Barbosa, and Tomasz Janowski. 2020. Data governance: Organizing data for trustworthy Artificial Intelligence. *Government Information Quarterly* 37, 3 (2020), 101493. https://doi.org/10

.1016/j.giq.2020.101493
[21] Marijn Janssen, Paul Brous, Elsa Estevez, Luis S. Barbosa, and Tomasz Janowski. 2020. Data governance: Organizing data for trustworthy Artificial Intelligence. *Government Information Quarterly* 37, 3 (2020), 101493. https://doi.org/10.1016/j.giq.2020.101493
[22] Staffs Keele et al. 2007. Guidelines for performing systematic literature reviews in software engineering. (2007).
[23] Indika Kumara, Martín Garriga, Angel Urbano Romeu, Dario Di Nucci, Fabio Palomba, Damian Andrew Tamburri, and Willem-Jan van den Heuvel. 2021. The do's and don'ts of infrastructure code: A systematic gray literature review. *Information and Software Technology* 137 (2021), 106593.
[24] Indika Kumara, Jun Han, Alan Colman, and Malinda Kapuruge. 2017. Software-Defined Service Networking: Performance Differentiation in Shared Multi-Tenant Cloud Applications. *IEEE Transactions on Services Computing* 10, 1 (2017), 9–22. https://doi.org/10.1109/TSC.2016.2594061
[25] Christian Lennerholt, Joeri Van Laere, and Eva Söderström. 2018. Implementation challenges of self service business intelligence: A literature review. (2018), 5055–5063.
[26] Wubin Li, Yves Lemieux, Jing Gao, Zhuofeng Zhao, and Yanbo Han. 2019. Service mesh: Challenges, state of the art, and future research opportunities. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 122–1225.
[27] Antti Loukiala, Juha-Pekka Joutsenlahti, Mikko Raatikainen, Tommi Mikkonen, and Timo Lehtonen. 2021. Migrating from a Centralized Data Warehouse to a Decentralized Data Platform Architecture. In *Product-Focused Software Process Improvement*, Luca Ardito, Andreas Jedlitschka, Maurizio Morisio, and Marco Torchiano (Eds.). Springer International Publishing, Cham, 36–48.
[28] Inês Machado, Carlos Costa, and Maribel Yasmina Santos. 2021. Data-driven information systems: the data mesh paradigm shift. (2021).
[29] Inês Araújo Machado, Carlos Costa, and Maribel Yasmina Santos. 2022. Data Mesh: Concepts and Principles of a Paradigm Shift in Data Architectures. *Procedia Computer Science* 196 (2022), 263–271. https://doi.org/10.1016/j.procs.2021.12.013 International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies 2021.
[30] Schultze Max and Wider Arif. 2020. *Data mesh in practice: how Europe's leading online platform for fashion goes beyond the data lake*. https://www.youtube.com/watch?v=eiUhV56uVUc
[31] Ralph Mietzner and Frank Leymann. 2010. A self-service portal for service-based applications. In *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. 1–8. https://doi.org/10.1109/SOCA.2010.5707165
[32] Michael Papazoglou. 2008. *Web services: principles and technology*. Pearson Education.
[33] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. 2007. Service-Oriented Computing: State of the Art and Research Challenges. *Computer* 40, 11 (2007), 38–45. https://doi.org/10.1109/MC.2007.400
[34] Mike P Papazoglou and Willem-Jan Van Den Heuvel. 2007. Service oriented architectures: approaches, technologies and research issues. *The VLDB journal* 16, 3 (2007), 389–415.
[35] Karthik Ramachandran. 2020. Data management barriers to AI success. https://www2.deloitte.com/xe/en/insights/industry/technology/ai-and-data-management.html
[36] David Reinsel-John Gantz-John Rydning, J Reinsel, and J Gantz. 2018. The digitization of the world from edge to core. *Framingham: International Data Corporation* 16 (2018).
[37] Marcelo Iury S. Oliveira, Glória de Fátima Barros Lima, and Bernadette Farias Lóscio. 2019. Investigations into data ecosystems: a systematic mapping study. *Knowledge and Information Systems* 61 (2019), 589–630.
[38] Frank E Saal, Ronald G Downey, and Mary A Lahey. 1980. Rating the ratings: Assessing the psychometric quality of rating data. *Psychological bulletin* 88, 2 (1980), 413.
[39] Johnny Saldaña. 2021. *The coding manual for qualitative researchers*. sage.
[40] Apitchaka Singjai, Uwe Zdun, and Olaf Zimmermann. 2021. Practitioner Views on the Interrelation of Microservice APIs and Domain-Driven Design: A Grey Literature Study Based on Grounded Theory. In *2021 IEEE 18th International Conference on Software Architecture (ICSA)*. 25–35. https://doi.org/10.1109/ICSA51549.2021.00011
[41] Monika Steidl, Michael Felderer, and Rudolf Ramler. 2023. The pipeline for the continuous development of artificial intelligence models—Current state of research and practice. *Journal of Systems and Software* (2023), 111615.
[42] Michael Stonebraker, Ihab F Ilyas, et al. 2018. Data Integration: The Current Status and the Way Forward. *IEEE Data Eng. Bull.* 41, 2 (2018), 3–9.
[43] Anselm Straus and Juliet Corbin. 2014. *Basics of qualitative research: Grounded theory procedures and techniques.*
[44] Zhamak Dehghani (Thoughtworks). 2019. *How to Move Beyond a Monothilitic Data Lake to a Distributed Data mesh*. https://martinfowler.com/articles/data-monolith-to-mesh.html

[45] Stephen John Warnett and Uwe Zdun. 2022. Architectural Design Decisions for Machine Learning Deployment. In *2022 IEEE 19th International Conference on Software Architecture (ICSA)*. 90–100. https://doi.org/10.1109/ICSA53651.2022.00017

[46] Yi Wei and M. Brian Blake. 2010. Service-Oriented Computing and Cloud Computing: Challenges and Opportunities. *IEEE Internet Computing* 14, 6 (2010), 72–75. https://doi.org/10.1109/MIC.2010.147

[47] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.

[48] Yu Zhou, Qian Zhao, and Mark Perry. 2002. Policy enforcement pattern. In *Proceedings of the Conference on Pattern Languages of Programs*. 1–14.