Data Mesh: Systematic Gray Literature Study, Reference Architecture, and Cloud-based Instantiation at ASML

Abel Abraham Goedegebuure
STUDENT NUMBER: 2065510

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE AND ENTREPRENEURSHIP
SCHOOL OF ECONOMICS AND MANAGEMENT
TILBURG UNIVERSITY

Thesis committee:

Supervised by Dr. Indika P.K. Weerasingha Dewage
Second Reader: Prof. Willem-Jan van den Heuvel

Abstract

To no surprise, it recently has become the norm for organizations to become data-driven. As the world continues to digitize, new opportunities arise that organizations are trying to capitalize on. Becoming data-driven allows organizations to use novel business models, develop new digital products, and make better decisions. However, becoming data-driven is not as easy as it sounds. Although there have been incredible innovations concerning the storing and processing of large amounts of data, organizations struggle with unlocking the true potential of their data.

Traditional big data architectures, such as data warehouses or data lakes, consist of centralized technical solutions to store vast amounts of data. A centralized team provisions the data by building data pipelines that transform the data into the desired format and serve it to consumers all across the organization. This imposes several challenges to the organization. First of all, the capacity of this centralized team becomes the bottleneck of data sharing in the entire organization. Furthermore, since the centralized team is separated from the business domains providing and consuming the data, it becomes difficult to serve high-quality data in line with the consumer's needs. Moreover, this centralized team's many point-to-point data pipelines become challenging to oversee and manage as their number increases. This removes all agility from the organization resulting in a large and monolithic data landscape. Additionally, data becomes challenging to govern since there is no clear ownership of data.

Data mesh, a novel data architecture paradigm, attempts to address these challenges. Data Mesh is a decentralised data architecture in which business domains themselves become responsible for serving their data as a product to the organization. These domains are supported by a self-serve infrastructure platform that abstracts away the complexities of managing the infrastructure necessary to serve data as a product. Moreover, it uses a federated computational governance model in which governance responsibilities are distributed to the business domains. However, there is a gap between industry practice and academic research. No academic literature is available yet, while the industry is already adopting the data mesh paradigm. This thesis addresses this gap by conducting a systematic gray literature review that defines data mesh and its requirements. Moreover, the design science research methodology for information systems is used to incorporate these findings into a reference architecture for data mesh. Additionally, the thesis delivers an instantiation of the architecture on a public cloud provider, and the design artifacts are demonstrated in an illustrative use case at ASML. All results are validated through expert interviews with senior stakeholders from the industry.

Acknowledgements

First of all, I would like to thank my supervisor Dr. Indika P.K. Weerasingha for his continuous support throughout my thesis. You were always available to answer my questions and pushed me to deliver research I am content with. Also, your passion for research is very motivating and pushed me to deliver more than I thought would be possible within the given time.

Secondly, I would like to thank Stefan Driessen for his interesting discussions and valuable feedback. Your eye for detail significantly improved the quality of my thesis and made me a better researcher. Also, your enormous passion for data mesh is very inspiring.

Lastly, I would like to thank Diederick Edel for the opportunity to conduct this research at ASML. The industrial setting allowed me to understand better the challenges data mesh is trying to address. Moreover, I felt taken seriously and part of the team from day one. I'm curious to see how ASML will continue on its data mesh journey.

Table of Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
ADF	Azure Data Factory
AML	Azure Machine Learning
CDC	Change Data Capture
CLI	Command Line Interface
CSSC	Customer Sourcing and Supply Chain
DDD	Domain Driven Design
DP	Data Product
DSRM	Design Science Research Methodology
ESB	Enterprise Service Bus
ETL	Extract, Transform, Load
EU	European Union
GDPR	General Data Protection Regulation
GLR	Gray Literature Review
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
HR	Human Resources
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Management
IaC	Infrastructure as Code
IDSA	International Data Space Architecture
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MOM	Message Oriented Middleware
NoSQL	Not Only SQL
OLAP	Online Analytical Processing
OLTP	Online Transactional Processing
OPA	Open Policy Agent
PEP	Policy Enforcement Pattern
R&D	Research and Development
REST	Representational State Transfer
RDBMS	Relational Database Management Systems
SLA	Service Level Agreement
SLO	Service Level Objective
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQ	Sub-question
SQL	Structured Query Language
SSC	Sourcing and Supply Chain

VNet Virtual Network
WSDL Web Service Description Language
XML Extensible Markup Language
xSOA Extended Service Oriented Architecture
YAML YAML Ain't Markup Language

Chapter 1

Introduction

With upcoming trends like IoT, the amount of data available worldwide is growing at an exponential rate. Over the next four years, the available data worldwide is expected to triple (Holst, 2021). Organizations (e.g. governments, companies) are also widely adopting the use of data through all their organizational layers and are starting to see data as an asset that can contribute to the success of their company (Perrons & Jensen, 2015). By grouping, clustering and filtering the operational data, it is transformed into analytical data which is suitable for efficient data processing. The analytical data is used to support their decision-making process. The increase in data generation and data consumption calls for proper storage, processing and governance of data to ensure it can add value to the organization.

First-generation data architectures use databases to store their operational data. Extract, Transform and Load (ETL) processes are used to extract operational data from these databases to cleanse, customize and integrate the data and to load it into a data warehouse (El-Sappagh et al., 2011). Consumers can query the data warehouses to have them provision data for analytic use cases (e.g. reporting) (Cardon, 2014). However, databases require data to be highly structured and are therefore less flexible and not suitable to keep up with the growing amount of (unstructured) data. This increase in volume, variety and velocity of data called for a new term, big data, and requires its own set of technologies (Sagiroglu & Sinanc, 2013).

In second-generation data architectures, data lakes facilitate this large scale, cost-effective storage of structured, semi-structured and unstructured data. Large quantities of operational data can be ingested directly into the data lake at high speeds. Not having to structure data into a fixed data model allowed organizations to store many kinds of data with relatively low effort (Stein & Morrison, 2014). Data lakes also provide support for data processing and analysis in batch or near real-time (John & Misra, 2017). Organizations often use data warehouses and data lakes in a complementary way to serve their needs (Madera & Laurent, 2016). Both options provide a centralized method for storing data, whereby a single team of expert data engineers is responsible for the ingestion and consumption of the data.

Organizations have governance mechanisms in place to manage their data. These mechanisms are essential to enable effective data usage as it ensures high data quality and assigns decision rights about data assets to members of the organization (Khatri & Brown, 2010). Data governance consists of different mechanisms that contribute to designing and executing a strategy with regards to data management (Abraham et al., 2019). Structural mechanisms determine roles and responsibilities and allocate decision-making authority. Procedural mechanisms ensure accurate data collection, secure storage and effective data usage. Relational mechanisms provide a framework that facilitates collaboration between stakeholders concerning data governance (Borgman et al., 2016). These mechanisms are implemented throughout the organization on all levels.

Without proper data governance, organizations will fail in unlocking value from their data. There will be low-quality data, no secure storage and usage of data, no compliance with external regulations, and much more. Therefore, data architectures and data governance programs go hand in hand to enable organizations to use their data on a large scale effectively. However, current-generation data

architectures and governance models come with limitations that obstruct organizations from unlocking the value of their data.

1.1 Problem Indication

Although the introduction of the data lake enabled cost-effective storage and processing of Big data, organizations struggle to truly unlock the value of their data. The additional flexibility of data lakes led organizations to collect more and more raw data. Over time, organizations lost track of the overwhelming piled up in these data lakes, slowly turning them into data swamps (Stein & Morrison, 2014). This means that both data warehouses and data lakes, or a combination of the two, cannot fulfil the current needs of organizations. When looking closer at these phenomena, we find that they can be traced back to three problems that result from the centralized architecture.

First of all, centralized data architectures struggle with the increasing amount of data providers (Roima, 2021). Most data lakes contain large amounts of raw data without clear ownership (Keboola, 2021; Shankar et al., 2021). The lack of ownership makes it difficult to make data discoverable for consumers (Dehghani, 2020b). Although this is not the case for data warehouses, these are not suitable for high volume, large variety and high-velocity Big data. Therefore, centralized data architectures inhibit consumers from finding and using data as they are simply unaware of its existence.

On the other side, centralized data architectures also struggle with the increasing amount of data consumers (Lam, 2021). Although data does not have to be structured when ingesting it to the data lake, it does need to be heavily processed and cleaned when serving it to consumers (Brackenbury et al., 2018). Organizations often have a centralized team of hyper-specialized data engineers whose responsibility is to build point to point connections to serve the data (Gillin, 2021; Snowflake, 2021). Using a centralized team to serve the data imposes several challenges. The capacity of this centralized team becomes the bottleneck to the entire organization (Oostra, 2021; Roima, 2021). Especially as companies are trying to become data-driven and make use of more data, practice has shown that this team becomes overloaded and cannot keep up with demand. Moreover, as the number of point-to-point connections increases over time, they become hard to manage (Rajagopalan, 2021). Changes in the data source or data consumer require the point-to-point connection to be changed or rebuilt, significantly reducing the organization's agility. There is little room for experimentation. In short, a data lake's flexibility becomes its own Achilles heel with regard to agility (Analytics, 2021).

Thirdly, the separation of data producers, data consumers, and the serving team creates knowledge siloes (Roima, 2021; Shankar et al., 2021). Data providers, who thoroughly understand the data, are unable to ingest the data in the proper format and do not understand the needs of data consumers. Therefore, the organization uses a centralized team of data engineers to serve the data. However, these lack domain knowledge on the data itself and the requirements from consumers. These knowledge siloes inhibit consumers from receiving high-quality data (Serra, 2021).

The factors described above also have implications for data governance. Lack of ownership makes it more challenging to ensure that data is used in line with its intended usage, creating additional risks for organizations to comply with internal policies or external regulations (Shankar et al., 2021). Moreover, the lack of domain knowledge produced by these siloes makes it challenging to govern data as the data itself is not completely understood. Overall, centralized data architectures impose several challenges that make it difficult for organizations to truly unlock the value of their data. Organizations are unable to scale with the growing amount of data sources, consumers, and their variety in needs while ensuring effective data governance.

1.2 Research Indication

Dehghani (2020a) recognised the challenges of a centralized architecture described above. She came up with a paradigm shift named data mesh that is based on a decentralized decentralization of re-

sponsibilities supported by a federated computational governance model. Data mesh is supposed to address the challenges of the conventional data architecture and enable organizations to unlock the value of their data on a large scale.

Gray literature suggests that many companies all around the world are already working on implementing a decentralized data architecture based on the data mesh principles. However, little academic literature exists, indicating a gap between academic research and industry practice. This thesis aims to address this gap by conducting a systematic gray literature review to define data mesh. To be specific, the study aims to deliver an industry definition of data mesh, its expected benefits, drawbacks and applicability and to incorporate the definition in a supportive architecture that depicts the different roles, their responsibilities and entities in a data mesh architecture. Moreover, based on a set of requirements from gray literature, the thesis aims to deliver a reference architecture for data mesh at runtime. Lastly, the thesis aims to provide an instantiation of the data mesh architecture on a public cloud provider.

1.3 Research Questions

The following research question is proposed to address the problem indication:

What is data mesh, and how can a data mesh be designed and implemented systematically in an organization?

To answer the main research question, the study aims to answer two sets of sub-questions. The first set is related to the definition and applicability of data mesh and is answered by conducting a systematic gray literature review.

1.1 What is data mesh and how does it differ from traditional data architectures for sharing data in organizations?

1.2 Why is data mesh beneficial for data sharing compared to traditional data architectures?

1.3 What are the challenges in realizing a data mesh architecture?

1.4 For which organizations is data mesh most suited?

Before organizations can adopt the data mesh paradigm, they need to understand what data mesh is and how it differs from their traditional data architecture. Moreover, the benefits of data mesh compared to their traditional architecture need to be clear and the challenges of implementing a data mesh architecture. Lastly, it needs to be clear what organizations should and which should not consider data mesh as it is not suitable for all organizations. Industry has published a lot of gray literature on these matters. This study aims to analyze them to sythensize a comprehensive definition of data mesh and to describe its benefits, challenges and applicability.

The main objective is to provide one complete and comprehensive definition of data mesh. Answering SQ1.1 allows us to define data mesh and its principles in line with how they are used by industry. Furthermore, SQ2 describes the benefits of data mesh in comparison to traditional data architectures, making it clear why organizations should consider adopting the paradigm. Thirdly, SQ1.3 describes the challenges that may occur when migrating to a data mesh architecture. This informs organizations and enables them to make a well-considered decision about whether they should adopt the paradigm or not. Lastly, SQSQ1.4 describes which organizations should and which should not consider data mesh as an architecture for their organization as the paradigm is not the most suitable architecture style for all organizations.

Solely answering the sub-questions above does not enable organizations to implement a data mesh architecture. Therefore, the second set of sub-questions aims to provide a reference architecture for data mesh and to describe how this architecture can be instantiated. Lastly, the results are validated

with expert interviews. Together, both sets of sub-questions answer the main research question. The design science research methodology (DSRM) is used to answer the second set of sub-questions with the results of the systematic gray literature providing input for the DSRM.

2.1 What is a reference architecture for a data mesh?

2.2 What are common design patterns for realizing a data mesh?

2.3 What tools are available to instantiate a data mesh architecture?

2.4 How can the reference architecture be instantiated using a public cloud provider?

2.5 How do relevant stakeholders evaluate the reference architecture?

SQ2.1 synthesizes requirements from gray literature for a data mesh architecture and incorporates them into a reference architecture. Reference architectures provide a common language and industry-accepted solution based on best practices (Clements et al., 2003). In order to increase the reliability of the results, the architecture makes use of design patterns (SQ2.2). These commonly used, proven by industry solutions increase the validity of the architecture as they are already proven by industry. By answering SQ2.3, the thesis presents an overview of the tools that are frequently used by industry to instantiate (parts of) the data mesh architecture. This enables organizations to implement the reference architecture in practice. Furthermore, as organizations are moving more to cloud-based data architectures, SQ2.4 illustrates how a data mesh architecture can be instantiated on a public cloud provider.

Lastly, interviews are conducted with experts from industry to validate the results presented in the thesis (SQ2.5). Feedback provided by participants is reiterated back into the generated results, increasing the reliability of the results. This increases the likelihood of adoption by industry.

1.4 Research Scope

The scope of this thesis is 1) to define the data mesh principles, its benefits, concerns and applicability, 2) to incorporate the identified roles, responsibilities and constructs in a reference architecture at design-time, 3) to identify components of the data mesh through the systematic gray literature review, 4) to identify common design patterns for data mesh, 5) to incorporate the identified components into a reference architecture at runtime, 6) to identify tools to instantiate data mesh, 7) to provide an instantiation for the reference architecture on a public cloud provider and 8) to validate all findings with relevant stakeholders from industry and to reiterate their feedback into the result of this study.

1.5 Research Relevance

The thesis is relevant from a theoretical viewpoint and industry viewpoint, which are both explained below.

1.5.1 Theoretical Relevance

The theoretical contribution of this thesis is to address the gap between industry practice and academic literature. A comprehensive and consistent definition of data mesh creates a shared understanding of data mesh in the field, which enables communication. This also provides a framework within which future research can be placed. The same is valid for the reference architectures and instantiation produced by this research.

1.5.2 Practical Relevance

The thesis produces several practical contributions to industry. First of all, it creates a shared understanding of data mesh, its benefits compared to traditional data architectures and potential drawbacks and applicability. This enables organizations to make a well-considered judgement about the fitness of data mesh for their organization. Moreover, the reference architectures, tools and instantiation guide organizations in implementing the data mesh architecture.

1.6 Thesis Organization

The rest of the thesis is structured as follows. Chapter 2 provides a literature review on academic work related to data mesh. This consists of a review of big data architectures, data governance, domain-driven design, service-oriented architectures and related work. Chapter 3 describes the methodology used to conduct the systematic gray literature review and the design science research methodology for information systems that were used to design the artifacts. Chapter 4 defines data mesh based on its four principles, the benefits of data mesh compared to traditional architectures, challenges of implementing a data mesh architecture and also which organizations should consider a data mesh architecture. Chapter 5 presents a reference architecture at design-time depicting the different roles, responsibilities and constructs in a data mesh architecture. Chapter 6 presents the requirements for a data mesh architecture at runtime and the architecture itself. This chapter also provides an overview of common design patterns and tools to instantiate a data mesh architecture. Chapter 7 presents an instantiation of the data mesh architecture on a public cloud provider. Chapter 8 illustrates a use case of data mesh in an industrial setting at ASML. Chapter 9 explains how the reference architectures were validated and the outcomes of the interviews. Chapter 10 discusses the generated results, after which chapter 11 provides a conclusion and recommendations for future work.

Figure 1.1 provides a concise overview of the structure of this thesis.

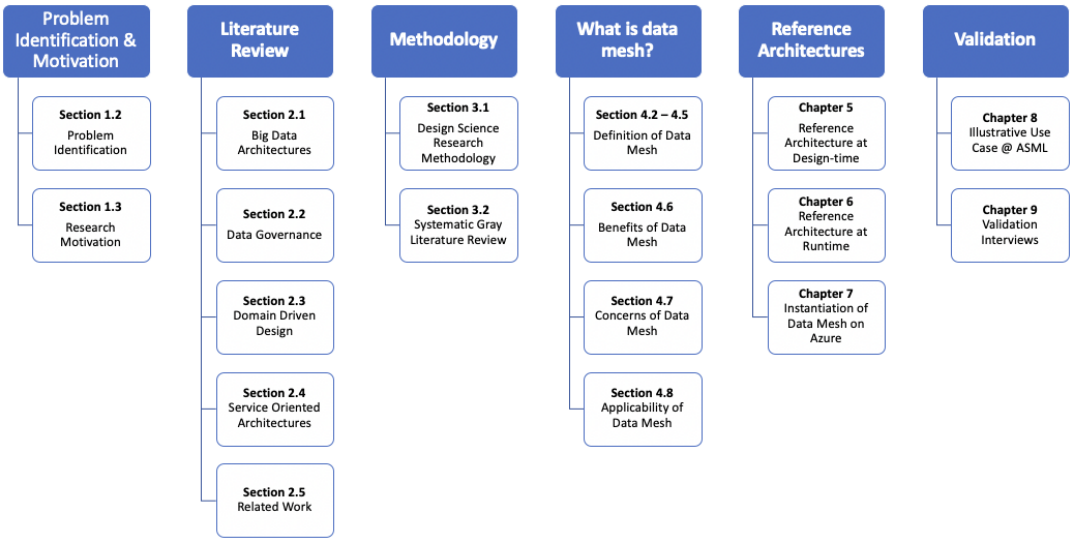

Figure 1.1: Structure of the Thesis

Chapter 2

Literature Review

This chapter introduces and explains different concepts from white literature related to the data mesh paradigm. Section 2.1 explains the history of data architectures and current generation data architectures. Section 2.2 describes how data is governed in these big data architectures. One of the data mesh principles is related to governance. Although data mesh preaches for a different governance model, the foundational elements will remain the same hence the inclusion of this chapter. Thirdly, section 2.3 explains the concept of domain-driven design in software engineering. One of the data mesh principles is based on the domain-driven design; hence it is introduced in this chapter. Section 2.4 describes service-oriented architectures (SOA). Data mesh resembles these architectures because both deliver services to users in a distributed environment. SOA is a heavily researched topic, and by making the analogy, this research can build on previous work. Lastly, section 2.5 describes work related to data mesh and why, although there is related work, this thesis remains relevant.

2.1 Big Data Architectures

As technology started to get a more prominent role in organizations, the collection of data began to explode, ultimately resulting in big data. This chapter describes how data architecture transformed over time to keep up with the changing needs.

2.1.1 First Generation Data Architectures - Databases & Data Warehouses

First-generation data architectures mainly consisted of SQL based data warehouses capable of storing highly-structured data. To store data in a data warehouse, it is extracted from the source, transformed and loaded into the data (better known as ETL). In the transformation step, the data is heavily cleaned and integrated into the schema of the warehouse (Bakshi, 2012). Data warehouses are efficient and allow for sophisticated online analytical support making them suitable for analytical use cases (H. Thomas & Datta, 2001). Data consumers can query the data warehouse using SQL to extract the data they need for their use case. Figure 2.1 shows the general flow of data from source to consumer when using a data warehouse.

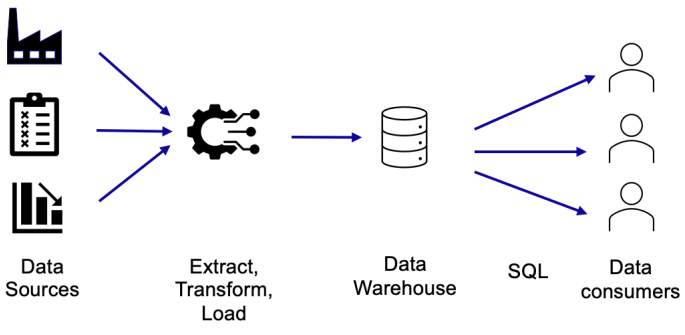

Figure 2.1: Data Flow in Data Warehouses

Data warehouses require highly structured data, making it unsuitable for all data types. NoSQL was invented to address the growing need for storing unstructured data. NoSQL separates data management from data storage through 'key-value stores', making it suitable for unstructured data (Bakshi, 2012; Nayak et al., 2013).

2.1.2 Second Generation Data Architectures - Data Lakes

Big data is characterized by three V's shown in fig. 2.2: a large variety in data types, increasing velocity of incoming data and an increase in the volume of data (Tanwar et al., 2015). The three V's are used to explain challenges in first-generation data architectures not suitable for big data.

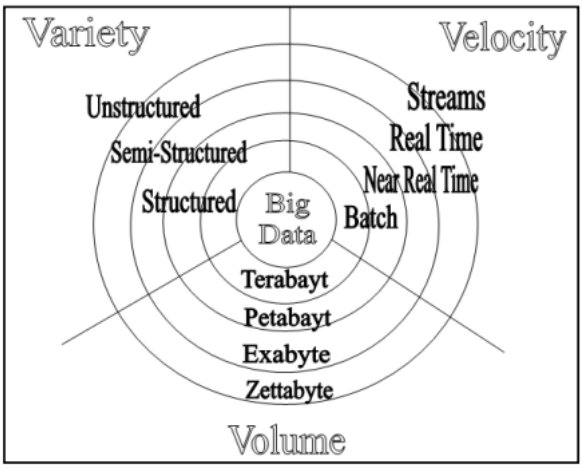

Figure 2.2: The Three V's of Big Data (Sagiroglu & Sinanc, 2013).

Over time, data generation evolved from having a transactional nature to also including interaction- and sensor data (Ramesh, 2015). During this transition, the variety of the collected data changed to include not only highly structured data but also semi-structured and unstructured data. The first has a fixed schema, while the latter has no fixed schema or is of a data type to which a schema is not applicable (e.g. photos) (Tanwar et al., 2015). At the same time, the increase in data sources led to a huge increase in the amount of data generated each day. Moreover, data becomes available more rapidly in real-time data streams. The recent adoption of IoT has led to an even larger increase in data volume and velocity of data (Marjani et al., 2017).

Data warehouses are not suitable to handle this explosion of data, advocating the development of new technologies. Although NoSQL provided the ability to store unstructured data, it was not suitable for Big data. Data lakes were developed to store and process big data through parallel compute configurations.

Data lakes provide a cost-effective data store for structured, semi-structured and unstructured big data. Organizations often use data lakes to store vast amounts of raw data. Data lakes originate from the Hadoop Distributed File System (HDFS), which provides a distributed file system and framework for the analysis and transformation of large datasets (Shvachko et al., 2010). Large scale data transformations and data ingestion are enabled by MapReduce, which provides automatic parallelization and distribution of large-scale computations (Dean & Ghemawat, 2004). Typically, data lakes are used to handle large volumes of unstructured data with a high-velocity (Miloslavskaya & Tolstoy, 2016). Data lakes enabled big data storage and processing.

When storing data in a data lake, raw data is extracted from the source and loaded into the data lake. Compared to data warehouses, this greatly reduces the amount of upfront complex transformations required for storing the data. Not until data consumers need data specific data stored in the lake, it is extracted, transformed and served to the consumer (Khine & Wang, 2018). The flow of data in a data lake architecture is depicted in fig. 2.3.

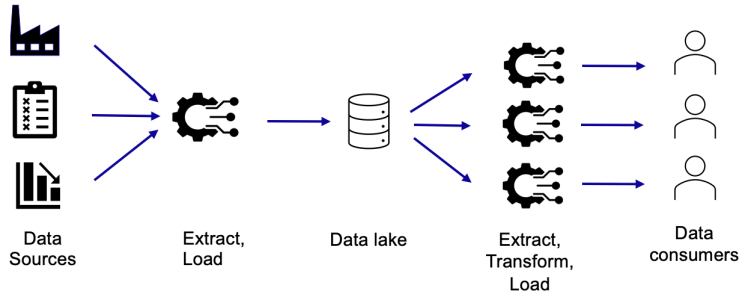

Figure 2.3: Data Flow in Data Lakes

The development of the data lake allowed organizations to effectively scale up their data capabilities and live up to the promise of big data. Big data analytics allowed for large scale processing of data to reveal insights and optimize decisions throughout the entire organization (Sagiroglu & Sinanc, 2013).

2.1.3 Third Generation Data Architectures - Live Data Processing

As the data landscape continued to develop, the need for real-time big data processing emerged. Where initially only batch processing was possible, the Kappa and Lambda architectures were developed to process big data in real-time.

Batch processing is used to process data that is already stored in non-volatile memory (Miloslavskaya & Tolstoy, 2016). The speed of processing depends on the computations to be performed by the batch algorithm. More intense computations require longer processing time which will thus delay the results. A downside of batch processing is that it can only approximate real-time processing to a certain degree but never be complete. A batch can potentially be processed every couple of minutes for simple computations. For some use cases, this is not real-time enough. Besides its slower processing capabilities, batch processing benefits from using high-quality data from the master dataset, leading to better performance in use cases that require high accuracy (e.g. training a model).

The Kappa architecture addresses the issue of batch processing by enabling real-time data processing. The architecture consists of a real-time layer and serving layer, as can be seen in fig. 2.4. The real-time layer processes only the most recent data to provide a real-time view of the data. The serving layer allows consumers to query these views. The Kappa architecture is relatively easy to maintain as it only requires one code base for the streaming algorithm. However, it is impossible to perform computationally intensive algorithms as this would delay the results. The results from the streaming layer are also less accurate in comparison to batch processing as it only uses limited data with lower quality as it comes directly from the source (Lin, 2017).

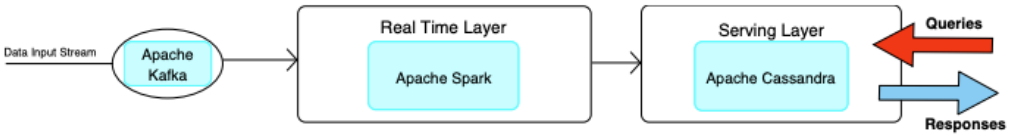

Figure 2.4: The Kappa Architecture (Feick et al., 2018).

The Lambda architecture uses a combination of batch processing and stream processing to provide the best of both worlds; see fig. 2.5. It uses a batch layer to perform batch processing on a master dataset as described above. On top of that, it uses a speed layer to apply real-time processing of only the most recent raw data (Warren & Marz, 2015). The serving layer merges the results of the batch and speed

layer so it can be queried by users (Feick et al., 2018). The Lambda architecture provides the best of both worlds as it can provide accurate results through batch processing while also providing a live view. This, however comes at the cost of additional complexity by having to maintain two separate code bases and an increase in the computational effort (John & Misra, 2017).

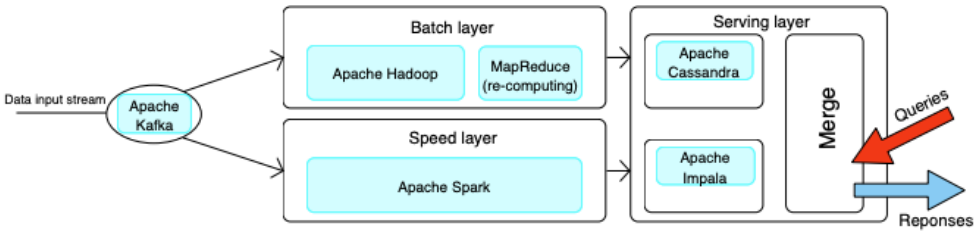

Figure 2.5: The Lambda Architecture (Feick et al., 2018).

2.1.4 Cloud Based Data Architectures

Where organizations initially only had their on-premise private clouds, the latest generation architecture started to incorporate cloud computing. For many organizations, cloud computing has become the way forward as it offers several benefits over hosting a private cloud. Cloud computing shifts the responsibility of building and maintaining physical infrastructure to the cloud provider. Clients can rent capacity and only pay for the resources that are used in a pay-as-you-go pricing model. Cloud providers offer Infrastructure as a Services, Platform as a Services and Software as a Service resources with increasing responsibilities for the cloud provider with regards to managing and maintaining the underlying infrastructure, operating system, etc. (Srivastava & Khan, 2018). Resources offered by cloud providers are accessed and managed over the internet.

Cloud computing has many benefits over using an on-premise, private cloud. First of all, the self-service model allows resources to be added or removed on demand, in an elastic manner (Dillon et al., 2010; Mohammed, Zeebaree, et al., 2021). Therefore, consumers can scale their resources in line with their organization's current needs without any upfront commitments and investments. Secondly, cloud providers pool resources for multiple tenants, lowering costs for all tenants. At the same time, tenants don't need to have highly specialized infrastructure knowledge in house since most of the technical complexity is taken care of by the cloud provider (Dillon et al., 2010). By masking the complexity of new services, organizations become more agile, which can lead to business model innovation (Berman et al., 2012). There are, of course, also drawbacks to cloud computing, such as reduced control over resources and security risks (Gai & Li, 2012).

Organizations often use a combination of the four architectures described above to serve their internal needs with regard to storing and processing big data.

2.2 Data Governance

A sound data architecture alone cannot guarantee effective data usage in organizations. Effective data governance and data management mechanisms need to be in place to ensure value can be derived from data. Governance refers to the decisions made to ensure effective use and management of data and define who makes these decisions (accountability). Data management is centred around making and implementing these governance decisions (Weill & Ross, 2004). Effective data governance and management enables organizations to derive business value from their data assets while reducing costs and risks (Abraham et al., 2019).

Abraham et al. (2019) designed a conceptual framework to develop a complete data governance program. The framework provides various governance mechanisms that together create an effective governance program. The framework distinguishes between three types of mechanisms: structural, procedural, and relational.

2.2.1 Structural Governance Mechanisms

Structural governance mechanisms determine the roles and responsibilities within the governance program and allocate decision making authority to said roles. Roles recognized by Abraham et al. (2019) are briefly explained below. Decision-making authority with respect to the overarching governance goals is assigned to these roles, depending on their hierarchical position, functional position and the degree of centralized decision making in the organization.

- The **Executive Sponsor** is a high level executive that provides the strategic direction and funding for the data governance program.
- The **Data Governance Leader** is responsible for the day to day implementation of the data governance program.
- **Data Owners** are business executives responsible for the data assets in their business domains.
- **Data Stewards** are subject matter experts from business domains that have a thorough understanding of the business- and data requirements. They translate these requirements into technical specifications for data quality (Wende, 2007).
- The **Data Governance Council** has the overarching, cross-functional role of providing strategic direction for the governance program and aligning the program with organizational goals. The council also monitors the governance program to see if it aligns with the strategic direction. The council is responsible for effective data governance by defining the top-down governance program (Alhassan et al., 2016; Traulsen et al., 2011).
- The **Data Governance Office** consists of supporting staff for data stewards and the data governance council.
- **Data Producers** create, aggregate or maintain data created by others.
- **Data Consumers** consume data to, for example, generate insights.

2.2.2 Procedural Governance Mechanisms

Procedural mechanisms ensure accurate data collection, secure data storage and effective data usage and sharing in the organization (Borgman et al., 2016). It consists of several sub-mechanisms that are outlined below (Abraham et al., 2019).

- **The Data Strategy** is the high-level course of action for the governance program. It consists of the strategic vision, business case, guiding principles, long-term and short-term objectives and roadmap for the program.
- **Policies** provide guidelines and rules regarding the creation, acquisition, storage, security, quality and permitted usage of data.
- **Standards** facilitate interoperability between data assets in an organization by ensuring that the data representation and execution of data-related activities are consistent throughout the organization. Standards can be related to the data (e.g. standard data definitions, taxonomies) but also to enforce technical implementations related to data (Panian, 2010).
- **Processes** are used to govern data in a standardized, documented and repeatable way.
- **Procedures** are the documented methods, techniques and steps to accomplish a specific task.

- **Contractual Agreements** act as a formalized agreement between participants in a governed data architecture. Examples are Service Level Objectives (SLOs) that state the data quality or data-sharing agreements that establish the data governance aspects before data sharing.
- **Performance Measurement** assesses the effectiveness of the governance program concerning the defined goals.
- **Compliance Monitoring** ensures that the use of data is compliant with external regulations and internal policies. Secondly, conformance to internal SLOs needs to be monitored.
- **Issue Management** provides standardized processes for the identification, management and resolution of issues related to data.

2.2.3 Relational Mechanisms

Lastly, the governance mechanisms consist of relational mechanisms that facilitate collaboration between stakeholders (Borgman et al., 2016). Their purpose is to communicate the governance program and its goals to all the relevant stakeholders in the organization. These mechanisms also provide proper training to all stakeholders to ensure they have the required knowledge to execute the governance program effectively. Lastly, the relational mechanisms give an approach to creating alignment across functions (e.g., top-down decision-making authority).

2.2.4 Implementing Governance Mechanisms

As shown in fig. 2.6, the exact implementation of the governance mechanisms is affected by the organizational scope, domain scope, data scope and any antecedents for a specific organization. These will be different for each organization. The organizational scope determines the breadth of the governance program. The data scope determines the type of data assets to be governed by the program. The domain scope translates the needs of the business domains into governance goals. Lastly, antecedents are any internal and external factors that affect the governance program (Abraham et al., 2019).

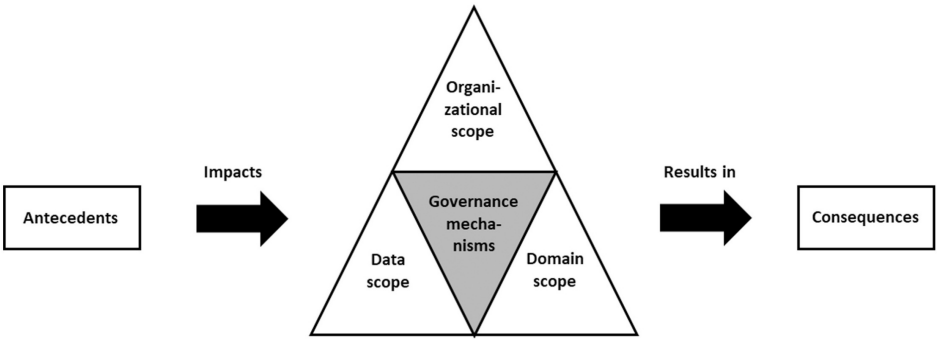

Figure 2.6: Conceptual framework for designing data governance mechanisms (Abraham et al., 2019).

Organizational Scope

The organizational scope determines the breadth of the governance program. An intra-organizational scope limits the breadth of the program to single projects in the organization or the entire organization. The inter-organizational scope opens the program to multiple organizations or complete ecosystems.

Domain Scope

The domain scope defines various goals centred around several decision domains (focus areas) (Abraham et al., 2019; Khatri & Brown, 2010). The first decision domain, data quality, states the required standards of the data to be in line with its intended usage (Khatri & Brown, 2010). Data quality can be defined on various metrics such as its accuracy, completeness, timeliness, consistency (Pipino et al., 2002). The second decision domain, data security, defines goals to ensure data is accessible, authentic, available, confidential and integer (Abraham et al., 2019). Data security can be enforced by encrypting data at rest and data in transit and having proper authentication and access policies to ensure confidentiality and integrity. Thirdly, the enterprise data model, on a conceptual-, logical- and physical level, needs to be defined as part of a complete governance program (e.g. define standards for data infrastructure) (Abraham et al., 2019). Moreover, data assets have a lifecycle that determines how long data should be retained and how data is managed throughout its lifetime concerning the needs of the business (Khatri & Brown, 2010). Furthermore, having a proper metadata strategy in place provides semantics to data and enables effective data governance (Abraham et al., 2019). Lastly, the data related infrastructure within an organization needs to be considered when drafting a governance program. The infrastructure frames the boundaries the program is to adhere to by, for example, any functional limitations from infrastructure (Tallon et al., 2013).

Data Scope

Furthermore, the governance program should apply to the data types in focus for an organization. A distinction is made between traditional data and big data (Abraham et al., 2019). Data governance concerning traditional data focuses on the consistent use of master data, transactional data and reference data in the organization (Dreibelbis et al., 2008). Master data describe the key business objects in an organization, which is important to define to create uniformity throughout the organization (Loshin, 2010). Secondly, transactional data represents the business transactions (e.g. purchase order). Lastly, reference data refers to agreed-upon values within the organization (e.g. customer ID).

Big data requires additional governance actions to prevent privacy infringements and data inconsistencies hence the distinction made by Abraham et al. (2019). Big data governance should reduce the risk of privacy infringement and data inconsistencies.

Antecedents

Besides data related factors, many other factors affect the exact implementation of the governance mechanisms. A distinction is made between internal factors and external factors. The first refers to facets like the organization's IT strategy, culture or organizational strategy. The latter refers to any external regulations or pressure from industry.

To summarize, an effective governance program uses the organizational scope, data scope, and domain scope to design structural-, procedural- and relational mechanisms that enable effective data governance. First, the program needs to be defined, after which it can be implemented. Then its performance needs to be monitored concerning the governance goals.

2.3 Domain Driven Design

Domain-Driven Design (DDD) is an approach to software development that incorporates domain knowledge into the design of software by basing software design on a domain model defined by domain experts (Vernon, 2013). Its main purpose is to generate higher quality software better aligned with the business. The design approach is not about technology but about understanding the business domain and basing software on this understanding. The domain is the area to which the developed software application is applied (Evans & Evans, 2004). Domain-driven design makes the development of complex software applications easier by dividing them into smaller pieces and aligning the

software with its application.

Domain experts draft a domain model using their domain knowledge. Software is developed using the language from the domain model, which causes a close link between the software implementation and the model (Evans & Evans, 2004). Because of the use of a common language, it also becomes easier for domain experts and technical developers to communicate with each other (Millett & Tune, 2015). Furthermore, basing software design on the domain model forces intense collaboration between the stakeholder to first define the domain model (Evans & Evans, 2004).

The domain model can be designed through intense collaboration between domain experts and software developers. This process consists of meetings, presentations, feedback sessions, brainstorms, etc. Throughout the process, it is important that the domain experts and software developers start talking in the same ubiquitous language (Evans & Evans, 2004). This means that software developers need to get an understanding of the concepts in the domain but also that the domain experts need to get some understanding of the technical jargon of software development. This allows both parties to give feedback on the domain model properly. The ubiquitous language should be used in all communication and also in the software's codebase (Evans & Evans, 2004). The language itself becomes a source of knowledge in software development (Vernon, 2013).

Typically, organizations are active in one core domain (e.g. retail). Drafting a domain model for the complete core domain would counter the DDD principles as this would result in a large and complex model. Therefore, it is important to recognize subdomains within the core domain. To illustrate, within the retail core domain, one can identify the subdomains shipping and invoicing. Furthermore, the bounded context acts as the conceptual boundary for a software application that requires its own domain model (Vernon, 2013). A bounded context can span multiple subdomains, and organizations can have various bounded contexts for different software applications. Each bounded context has its own ubiquitous language depending on the involved subdomains for said applications. Figure 2.7 depicts the relation between domains, sub-domains and bounded contexts.

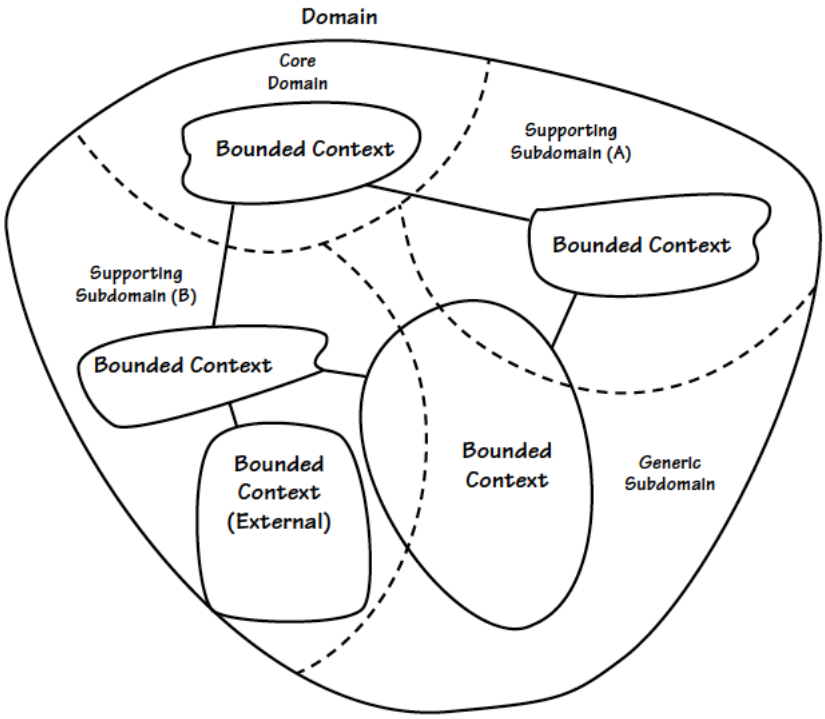

Figure 2.7: Abstract Business Domain Including Sub-domains and Bounded Contexts (Vernon, 2013).

There are numerous benefits to using domain-driven design for software development. First of all, having close alignment between the domain and software provides a better end-user experience. The software is more intuitive to use, and users need less training (Vernon, 2013). Moreover, from a technical perspective, software development becomes more agile and organized and reduces complexity for developers (Millett & Tune, 2015; Vernon, 2013).

2.4 Service Oriented Architectures

A Service-Oriented Architecture (SOA) is an approach to designing software systems that provide services to end-users or other services in a distributed environment (M. Papazoglou, 2008). It is a method to organize, use and manage distributed capabilities owned by different domains (MacKenzie et al., 2006). Web services lie at the core of a service-oriented architecture. Web services and a Service Oriented Architecture resemble data products and a data mesh architecture, hence their inclusion in the literature review.

A service-oriented architecture is a design philosophy that makes previously siloed software applications available and discoverable through standardized interfaces and messaging protocols (M. Papazoglou, 2008). The general goal is to make existing technologies available and interoperable throughout the architecture. It reduces complexity for large applications in a scalable, evolvable and manageable fashion and makes organizations more agile (MacKenzie et al., 2006). SOA is not a technology but an architectural style. Although it does not exclusively work with web services, this is the most prominent example.

2.4.1 Web Services

A web service is a self-contained module that encapsulates a business task, a complete business process application, or a service enabled resource that provides access to a back-end database (M. Papazoglou, 2008). Web services have an interface which can be accessed over networks (e.g. internet). An example web service performs an inventory check when a customer places a new order.

A distinction can be made between simple and composite web services, as illustrated in 2.8. The first only supports a simple request operation, after which it performs its encoded logic and returns a response to the service client. Composite web services coordinate several other services that execute their business logic, executing a complete business process. Typically, simple services are atomic and stateless. Composite services are more complex and often stateful (M. Papazoglou, 2008). Web services are loosely coupled, meaning that they do not depend on the state of other services to function properly. This allows for asynchronous, message-based communication (M. Papazoglou, 2008).

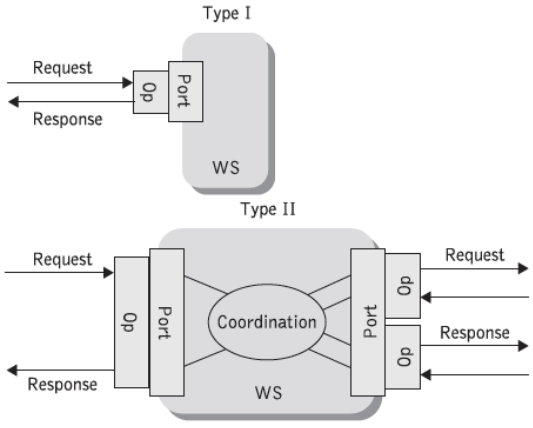

Figure 2.8: Simple and Composite Web Services (M. Papazoglou, 2008).

A service provider builds web services using their preferred programming language. Only the interface is exposed to clients who can request the service to execute its logic (Erl, 2007). Web services are described in a standardized service description language (e.g. WSDL). The service description provides all information necessary to interact with the service interface (e.g. message format, communication protocol) (Gottschalk et al., 2002). The standardized service description language provides a uniform representation of web services independent of their specific implementation. This supports heterogeneity on the interaction level (Curbera et al., 2001).

Web services make use of message-based communication. Messages use a standardized messaging protocol that structures the messages (e.g. SOAP). These messages are exchanged through transfer protocols over the internet. Message exchange is handled by the Message Oriented Middleware (MOM). Web services provide their messages to the MOM, which ensures it is delivered to the consumer (i.e. user or other web services), thereby abstracting away the complexity of message delivery. The MOM is often integrated by the message broker, which includes other aspects such as security or load balancing (M. Papazoglou, 2008). 2.9 shows the relation between the components described above.

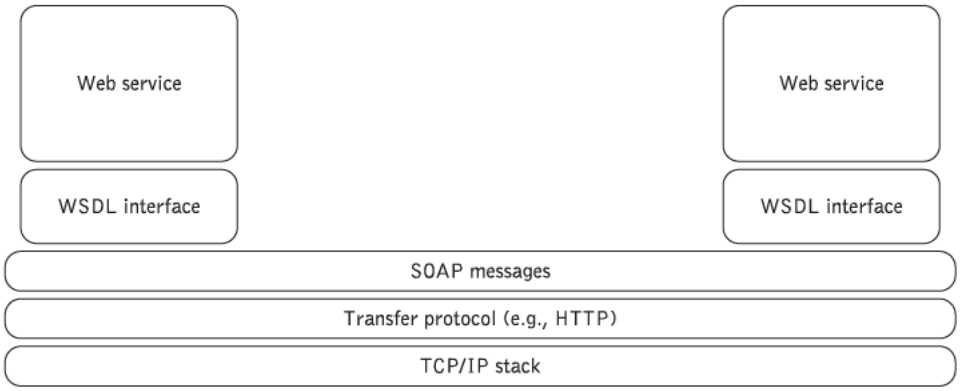

Figure 2.9: Web Service Communication (M. Papazoglou, 2008).

Generally, the MOM uses the publish / subscribe pattern, although other patterns are available. Web services publish their messages to the message broker, after which all subscribed clients receive said message (M. Papazoglou, 2008). Different variations of the publish / subscribe pattern exist that determine how the message broker distributes the messages. Messages can be distributed based on their type, content or the topic they are published to (Eugster et al., 2003).

2.4.2 Service Management

Proper functioning of a service-oriented architecture goes beyond simply offering and consuming web services. Web services need to be managed to guarantee a quality of service. Management solutions need to depict how web services perform in relation to their operational environment to enable management decisions and control actions to adjust the behaviour of web-service enabled applications (M. Papazoglou, 2008). This consists of the ability to monitor the user experience, the operational health of a web service, transaction monitoring, resource provisioning to properly manage the service load and SLA monitoring to see how the web service performs with regards to its Service Level Agreements (SLAs).

The actions above contribute to the operational and tactical management of web services. Operational management refers to discovering the existence, availability, performance, health, usage patterns, lifecycle support and maintenance of web services. Tactical management refers to management concerning the overarching business process (M. Papazoglou, 2008). To enable operational service management, SOA is extended with management infrastructure. Web services include a management interface to provide management-related capabilities. Management services bind to web services through the management channel to enable metering, metric and event mediation, monitoring, system

scanning, policy enforcement and management and model bridges (M. Papazoglou, 2008). The extension of SOA is shown in 2.10.

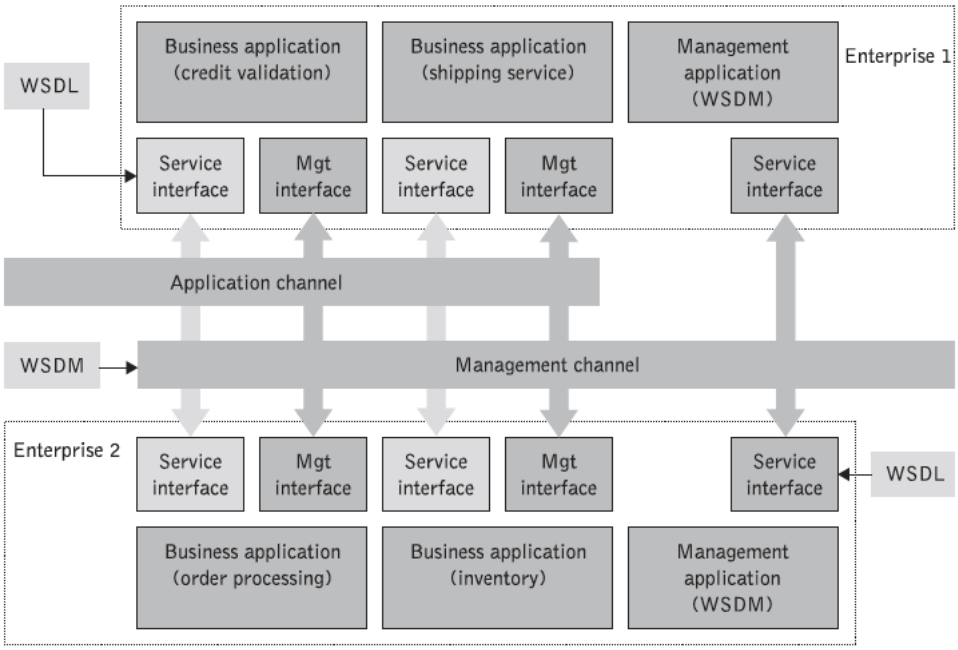

Figure 2.10: Managed Web Service Infrastructure (M. Papazoglou, 2008).

2.4.3 Development of Service Oriented Architectures

Typically, there are three roles in an SOA. The service provider is responsible for implementing the business logic into a service and publishing it to a service registry. A service can be published by describing the service from a utility and technical perspective. The service registry provides a searchable directory that makes the services discoverable. The registry also contains the service description to enable clients to bind to the interface. Lastly, the service clients invoke the web services developed by the service providers to utilize their functionality (Roy & Ramanujan, 2001). The interaction between the three roles is highlighted in 2.11.

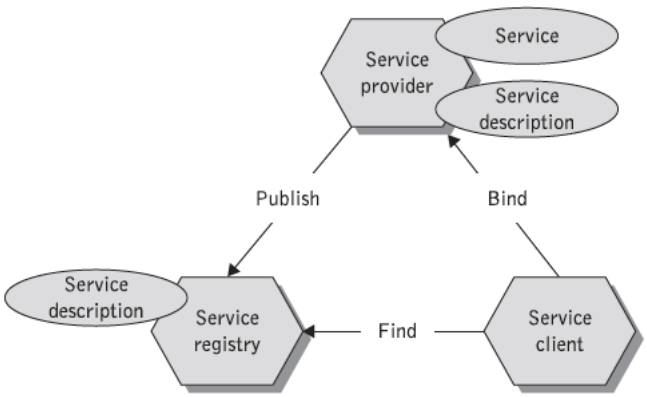

Figure 2.11: Web Service Roles and Operations (M. Papazoglou, 2008).

SOA development can be explained through a layered approach, as shown in 2.12. The top-down approach decomposes a business domain into a collection of services. The bottom-up approach composes operational systems into business services (M. Papazoglou, 2008). In the figure below, the top-down approach starts from the topmost layer that represents the domain to be decomposed into a set of services. Within said domain, a set of business processes occur that can be further decomposed into business services. These services automate atomic business tasks (e.g. creating an order). From a technical perspective, these business services are executed by infrastructure services. Infrastructure services are generic services that perform a single task and can be reused across different business processes or domains. Component-based services make existing operational systems available as a service (i.e. gives it an interface, etc.). Examples of operational systems are databases or monolithic applications (Arsanjani et al., 2007). The bottom-up approach starts the other way around by reasoning from the operational systems that are already available.

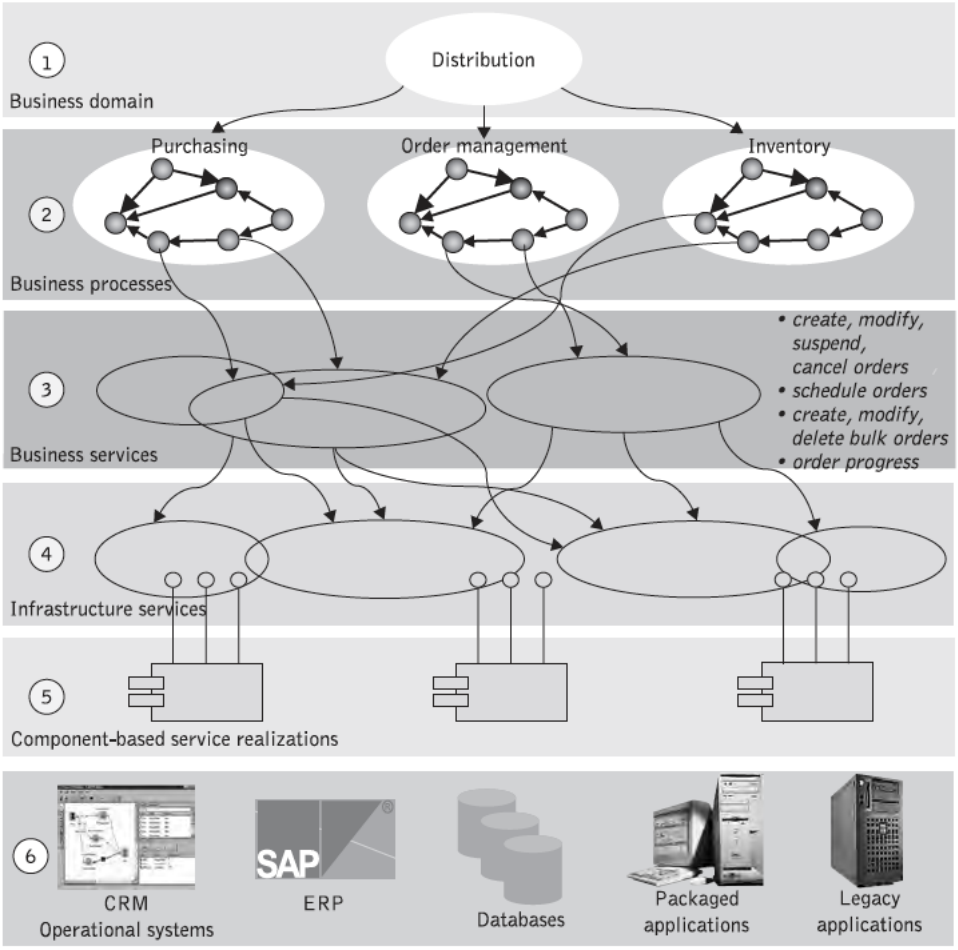

Figure 2.12: Layers in a Service Oriented Architecture (M. Papazoglou, 2008).

2.4.4 Extended Service Oriented Architecture

M. P. Papazoglou and Van Den Heuvel (2007) designed a layered architecture to streamline and structure the required functionality of applications making use of the service-oriented architecture paradigm. xSOA is split up into different layers with a set of constructs, roles and responsibilities, see 2.13. For a layer to be realized in practice, it builds on the preceding layers. Extended SOA differentiates between simple web services, composite web services and managed web services to depict the different actions required to go from simple to managed web services.

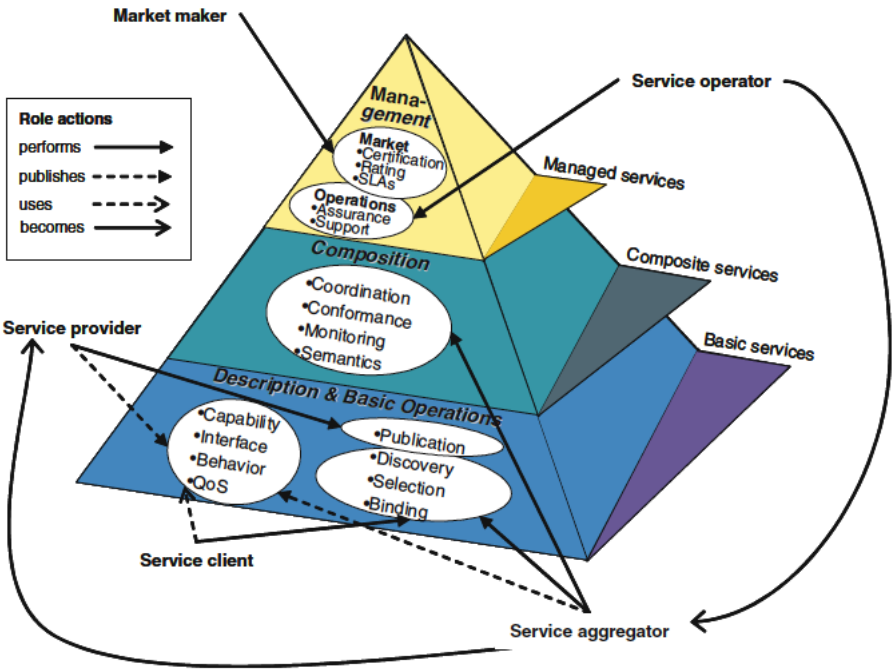

Figure 2.13: Extended Service Oriented Architecture (M. P. Papazoglou & Van Den Heuvel, 2007).

2.5 Related Work

GAIA-X is a European initiative to create the next generation of data infrastructure to be used by European companies and citizens (Braud et al., 2021). Its main purpose is to create an ecosystem for data collation and sharing in a trusted environment. Within this federated environment, data owners remain in full control of their data by setting constraints on the data's intended usage. GAIA-X is composed of an infrastructure ecosystem and a data ecosystem. The first focuses on delivering infrastructure services to store, transfer and process data provided by public cloud providers or network providers. The latter supports data spaces which allow for sovereign and secure exchange of data and the building of smart services across industries (Braud et al., 2021).

Cloud providers or network providers can provide the infrastructure ecosystem. Its main purpose is to provide an ecosystem of digital operators that create a federated cloud infrastructure in line with European values. Setting up such an ecosystem enables scalability, interoperability and freedom of choice between infrastructure providers (Autolitano & Pawlowska, 2021). In short, its purpose is to provide an EU sovereign cloud in line with European values and regulations such as the GDPR.

The data ecosystem is centred around using data. GAIA-X provides secure data spaces to participants who remain in complete control. It is intended that data stored in these data spaces are used for new data-driven services in line with the defined use cases for the originating data space (Otto et al., 2021). Data sovereignty has a central notion in GAIA-X. To enable sovereign data exchange, GAIA-X makes use of the International Data Spaces Reference Architecture Model (IDS-RAM) shown in 2.14.

Based on IDSA (2019), the data owner is the legal entity that creates data and/or executes control over it. The data provider makes the data available from a technical perspective on behalf of the data owner. Data spaces are made discoverable by registering their metadata in the Broker. Consumers can search the broker and request the data. If approved, data exchange occurs through the service provider, which is a secure and isolated container for individual data services. Data can also be processed by apps (i.e. from by cloud provider). Data transactions are logged by the clearinghouse. The vocabulary provider offers specific vocabularies to annotate datasets. Lastly, the identity provider provides a centralized authentication service for all IDS participants. The IDSA framework is implemented into the GAIA-X architecture.

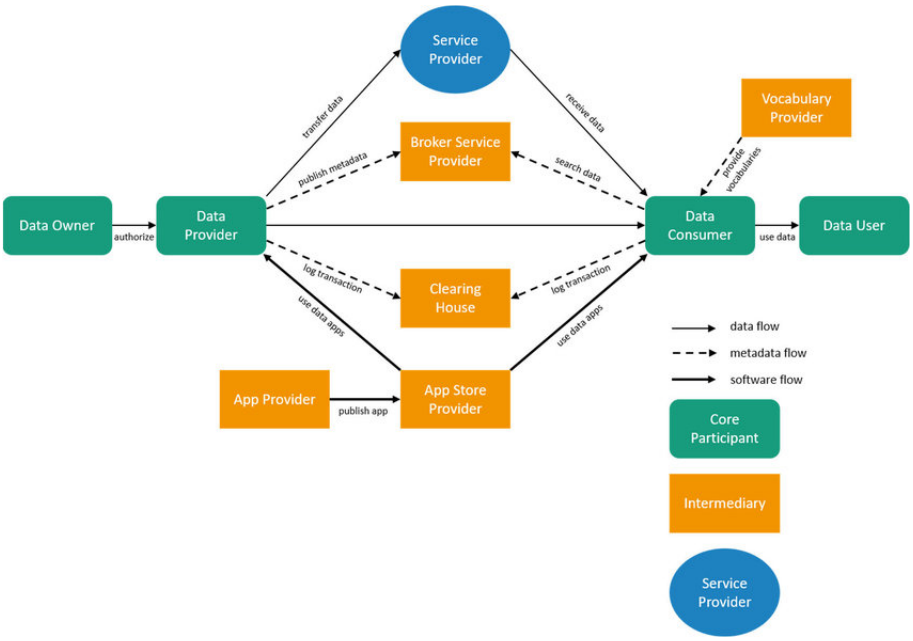

Figure 2.14: IDS Roles and Interactions (IDSA, 2019).

High Level GAIA-X Architecture

Braud et al. (2021) high level architecture for GAIA-X, shown in 2.15, depicts the integration of the infrastructure ecosystem and the data ecosystem. Nodes in the architecture represent infrastructure elements, services represent cloud offerings, service instances are imitations of services and data assets are data sets. Furthermore, participants of GAIA-X consist of organizations or legal persons providing or consuming data. Logically, some participants provide the nodes, services and service instances. Each participant is described using GAIA-X's mandatory self-description (i.e. data owner and usage policies for data assets) to establish trust and enable effective governance.

The federated catalogue enables the discovery of data assets by making them self-describing. Each data asset comes with monitoring capabilities and a metering interface. The architecture uses a federated identity management system that connects to national and international identity providers to establish trust among participants. Each participant, service or node receives an assurance level that indicates the level of conformity. Access to data assets is managed in a federated manner. Compliance between the relationship of a service provider and service consumer is established through definitions, onboarding and certification. The rights and obligations of both the provider and consumer are also defined. Lastly, sovereign data exchange is realized by defining data usage policies that describe how and by whom data can be used. Technical implementation of these policies ensures that the consumer adheres to the intended usage even when the data has left the provider's system boundaries.

All concepts described above are enforced throughout the architecture through policy rules, standards and interconnection.

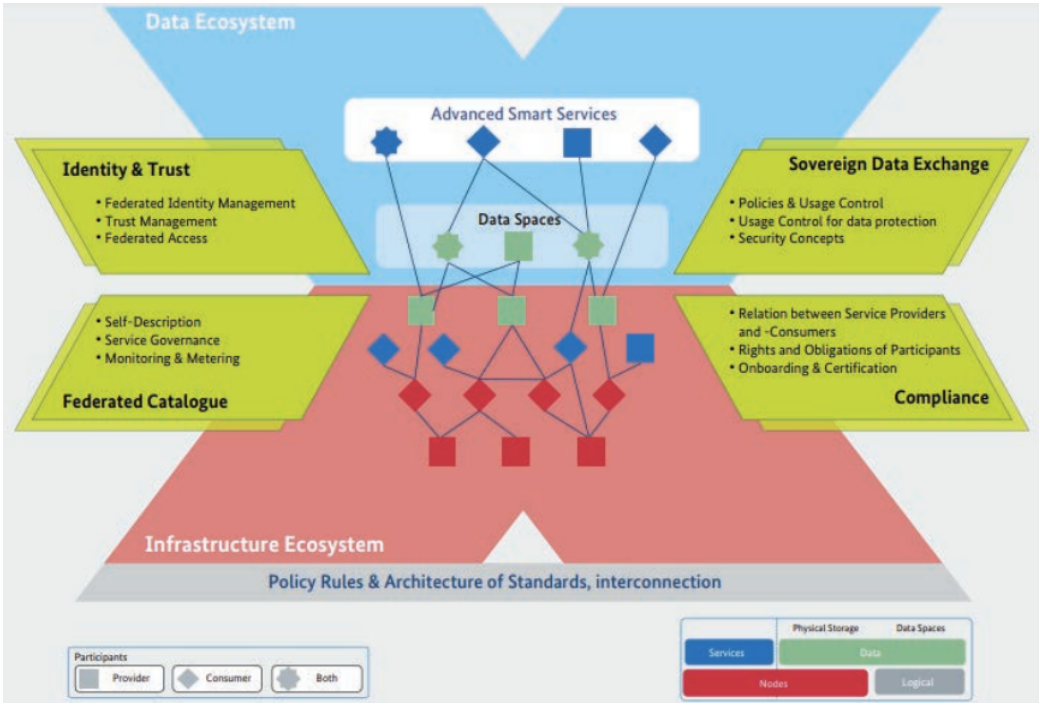

Figure 2.15: High Level View of the GAIA-X Architecture (Braud et al., 2021)

GAIA-X is still in an early stage, but the involved stakeholders hope it becomes successful in making the EU less dependent on large foreign companies and ensuring digital sovereignty (Autolitano & Pawlowska, 2021).

2.5.1 GAIA-X and Data Mesh

Although GAIA-X resembles data mesh, it is important to highlight the differences and why this study remains relevant.

As described above, GAIA-X is a European Initiative to facilitate secure data exchange between organizations and natural persons within the EU. Data mesh is an intra-organizational decentralized data architecture with an emphasis on data sharing between business domains. In principle, both are a form of decentralized data architecture.

However, the inter organizational scope imposes different challenges compared to the intra-organizational scope. In short, although there are numerous similarities, the difference in scope creates differences in the details.

Chapter 3

Research Design

This chapter describes the research design for this thesis. The thesis makes use of the design science research model for information systems research (DSRM) designed by Peffers et al. (2007). The model consists of different stages one can go through for design science in information systems. A systematic gray literature study was conducted to provide input for the first two stages of the DSRM.

Section 3.1 describes how the DSRM is applied in this study. Section 3.2 explains how the gray literature study was conducted and its relation to stages 1 and 2 of the DSRM.

3.1 The Design Science Research Method

Two paradigms lie at the roots of information system research: behavioral and design science. The first seeks to develop and verify theories that explain or predict human or organizational behavior. The latter is fundamentally a problem-solving paradigm that seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts (A. R. Hevner et al., 2004). Domain understanding is emphasized heavily in design science research (A. Hevner & Chatterjee, 2010). Behavioral and design science are heavily intertwined in information systems research because the information systems discipline is a junction of people, organizations, and technology (A. R. Hevner et al., 2004).

Peffers et al. (2007) designed a common framework to execute design science research on information systems in line with A. R. Hevner et al. (2004) guidelines on design science for information systems. The framework consists of a nominal process model to perform design science research for information systems and a mental model to structure and evaluate research outputs.

Peffers et al. (2007) design science research model consists of six steps as can be seen in fig. 3.1. Below, the DSRM is placed in the context of this thesis. The thesis starts at step one of the DSRM.

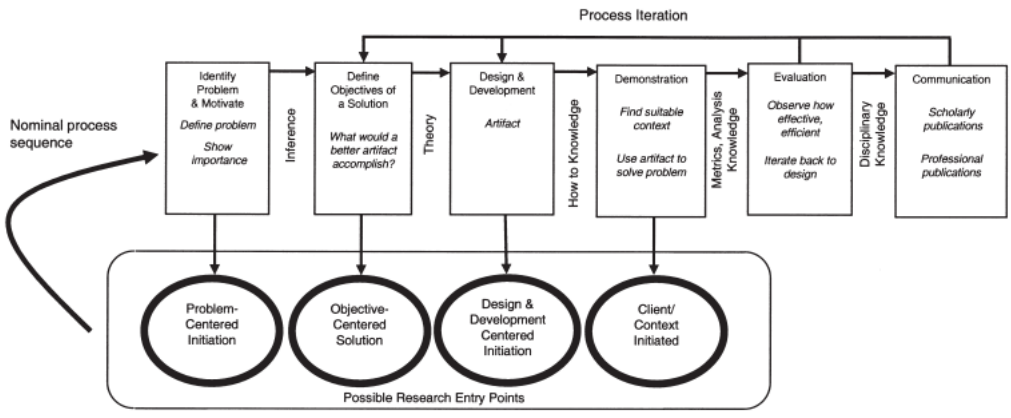

Figure 3.1: Peffers et al. (2007) Design Science Research Methodology.

(S1) **Problem identification & motivation**
This stage defines the research problem and a justification for a solution. Justification of the solution is important as it helps the audience understand the reasoning throughout the design process and motivates them to accept the proposed solution.

For this thesis, the gap between industry practice and academic research suggests the need for more formalized research into data mesh. Lack of a formalized definition of data mesh and lack of a reference architecture inhibit organizations from adopting a decentralized data architecture.

(S2) **Objectives of a solution**
Stage two defines the objectives of a solution artifact inferred by the problem definition in stage one. Objectives can be defined quantitatively or qualitatively, but the latter is more sensible for this problem definition.

This thesis has several objectives. The first is to define a comprehensive and consistent definition of data mesh based on how it is defined by industry. Moreover, the objective is to design a reference architecture for data mesh and to show how it can be instantiated. Lack of a consistent and complete definition of data mesh and reference architecture inhibits organizations from adopting the paradigm.

(S3) **Design and development**
This stage presents the artifacts (i.e. constructs, models or methods) with a research contribution embedded in their design. First, the desired functionality is defined based on the systematic gray literature review, after which they are incorporated into the artifacts.

The thesis provides four artifacts: first of all the comprehensive definition of data mesh based on a synthesis of gray literature. Secondly, the thesis presents a reference architecture at design-time. Thirdly, the thesis delivers a reference architecture at runtime which shows the logical dependencies of the constructs (i.e. data products, management services, etc.) at runtime. The lsat artifact is an instantiation of a data mesh architecture on Azure's Cloud platform based on instantiations presented in gray literature.

(S4) **Demonstration**
The efficacy of the designed artifacts are demonstrated in an illustrative use case at ASML. The demonstration aims to bring parts of the reference architecture at design-time and runtime into practice by provisioning a MVP self-service infrastructure platform that a business domain in ASML can use to provision data as a product. Due to the time constraint of the thesis, it was not possible to demonstrate the entire reference architectures.

(S5) **Evaluation**
The artifacts produced in stage three are evaluated concerning the objectives defined in stage two. The thesis uses qualitative semi-structured interviews with industry experts as a means for validation. The evaluation results are reiterated in the produced artifacts in stage three.

(S6) **Communication**
Lastly, this thesis communicates defined problem, solution objectives, and produced artifacts to a broader audience.

Figure 3.2 depicts the design science research methodology in the context of this thesis.

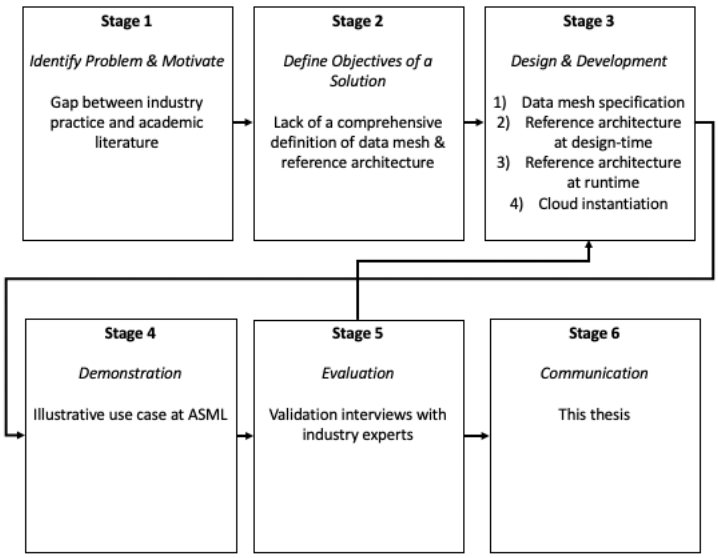

Figure 3.2: The DSRM applied in this thesis

3.2 Systematic Gray Literature Review

Due to the novelty of data mesh, no white literature is available on this topic yet. Therefore a systematic gray literature review (GLR) is conducted to provide input for the first two stages of the DSRM. Moreover, the gray literature is also used to extract the desired functionality for the reference architectures.

Garousi et al. (2019) framework for conducting systematic gray literature reviews was used to perform the gray literature study. The framework consists of two stages that contain different steps, see fig. 3.3. For each step in the framework, a brief explanation is given.

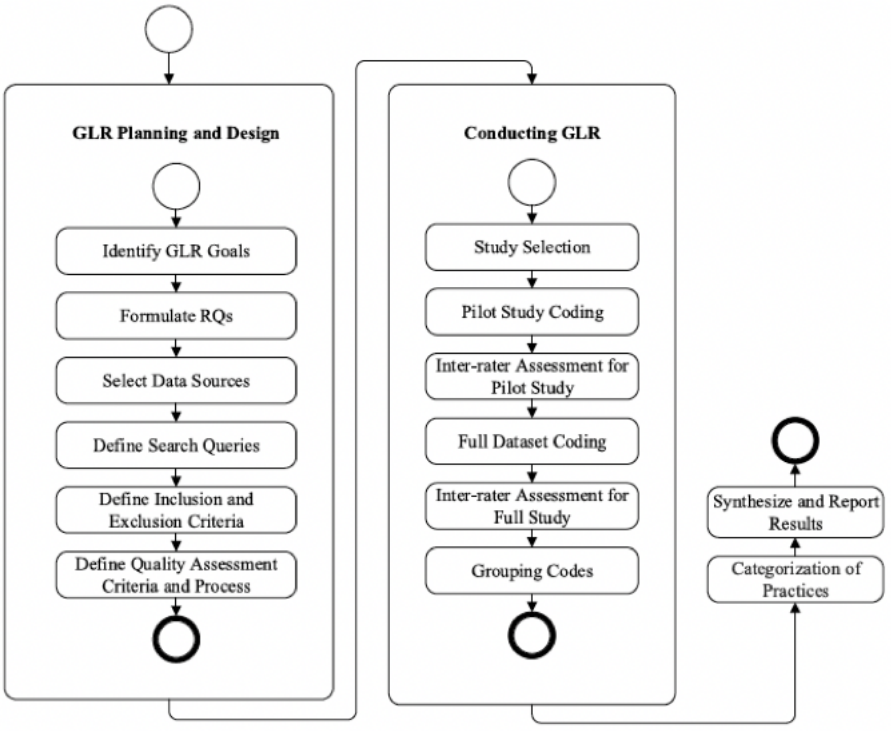

Figure 3.3: Framework for conducting systematic gray literature reviews (Kumara et al., 2021).

3.2.1 Stage 1: GLR Planning and Design

Due to the lack of white literature available on data mesh, the main goal of the GLR is to provide a definition of data mesh based on its four principles and to list the requirements for a data mesh-based architecture. The research questions are listed below:

- **R1:** What is the definition of data mesh based on its four principles?
- **R2:** What are the benefits and drawbacks of a data mesh architecture in comparison to a monolithic data architecture?
- **R3:** Which organizations consider using a data mesh based architecture?
- **R4:** What are the roles and their responsibilities in a data mesh architecture?
- **R5:** What are the requirements for a data mesh architecture?
- **R6:** What are the design patterns for realizing a data mesh?
- **R7:** What tools are available to instantiate a data mesh architecture?
- **R8:** How can a data mesh architecture be instantiated on a public cloud provider?

R1, R2 and R3 are used in the problem identification and motivation of stage one of the DSRM as it outlines the context of data mesh and the problem it is trying to solve. R4, R5 and R6 draft the objectives and requirements for the artifacts to be designed in stage three of the DSRM. R7 and R8 provide more context on how the designed artifacts can be implemented.

The selected gray literature is limited to textual sources such as reports, blog posts, white papers, official documentation from vendors, presentations and transcriptions of keynotes and webinars. Initially, the three search queries below were used to gather data sources from Google search engine. However, the last query obtained too many irrelevant results and was therefore excluded from the study. Reference lists and backlinks from this original set of articles were studied to find additional relevant gray literature as recommended by Garousi et al. (2019).

- **Q1:** Data Mesh (Requirements |Benefits |Challenges |Tools |Architecture)
- **Q2:** Decentralized Data Architecture
- **Q3:** Distributed Data Architecture

The first two queries were used in the Google search engine on September 1st 2021, until saturation. Due to the topic's novelty, information in sources up until that time was 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 April 1st 2022. In total, this resulted in 155 data sources.

Articles were filtered based on the defined inclusion and exclusion criteria listed below by reading through the main sources. Additionally, a quality assessment was performed based on the criteria listed below. Inclusion-, exclusion and quality criteria are partly based on Kumara et al. (2021). Articles not meeting one of the quality criteria were excluded from the GLR.

Inclusion

- Articles in English and full text is available
- Articles match the focus of the study
- Example use cases
- Instantiations of data mesh architectures

Exclusion

- Articles not matching the focus of the study
- Articles Restricted by paywall
- Duplicate articles from various sources
- Short elements that don't contain sufficient data
- Content with a very subjective nature

Quality Assessment Criteria

- Is the publishing organization reputable?
- Is an individual author associated with a reputable organization?
- Has the author published other work in the field?
- Does the author have expertise in the area?
- Does the source have a clearly stated purpose?

Validation of data sources concerning the quality assessment was conducted by two participants of the research team, in this case, supported by the supervisor of this study. A mutual agreement was required to include a data source in the GLR. Assessment of all data sources resulted in a final set of 114 selected sources, which can be found in appendix A. The Cohen Kappa can be used to assess the inter-rater reliability (Cohen, 1960). The inter-rater assessment had a coefficient of 0.79, indicating a substantial agreement between the two assessors.

Table 3.1 provides an overview of the selected studies. The content type, contribution type, year of publication, and publication venue are given for each study. Subsequent sections refer to individual studies using their identifier (e.g. [S1]). Moreover, the complete list of articles can be found in appendix A.

Table 3.1: Overview of the selected studies

Study	Type	Content	Year	Venue
[S1]	Article	Definition, Benefits, Architecture	2020	martinFowler
[S2]	Article	Definition, Benefits	2019	martinFowler
[S3]	Article	Definition, Benefits, Challenges,	-	Deloitte
[S4]	Article	Definition, Benefits, Applicability	2020	Towardsdatascience
[S5]	Article	Benefits	2021	Futurice
[S6]	Video transcript	Definition, Use Case	2020	Sparkt AI
[S7]	Article	Benefits	2021	Jamesserra
[S8]	Videos	Definition	2021	AgileLab
[S9]	Article	Definition, Benefits, Architecture	2021	CueLogic
[S10]	Blog	Use Case	2021	JPMorgan
[S11]	Blog	Definition	2021	LinkedIn
[S12]	Article	Definition, Benefits, Challenges, Applicability	2021	Immuta
[S13]	Blog	Benefits, Instantiation, Architecture	2021	Amazon Web Services
[S14]	Blog	Definition, Use Case, Architecture, Instantiation	2021	WikiBon
[S15]	Blog	Architecture, Tools	2021	Octo
[S16]	Blog	Definition,	2021	Silicon Angle
[S17]	Blog	Use Case	2021	DataVersity
[S18]	Blog	Definition, Challenges,	2021	Data Driven Investor
[S19]	Blog	Definition, Tools	2021	Capgemini
[S20]	Blog	Definition, Architecture	2021	Solace
[S21]	Blog	Definition, Tools, Architecture, Challenges	2021	CNR
[S22]	Blog	Definition, Architecture, Tools,	2021	Langaro Group
[S23]	Blog	Definition, Implementation, Tools,	2021	Scott Logic
[S24]	Blog	Definition, Architecture, Tools,	2021	Medium
[S25]	Blog	Definition, Challenges, Benefits, Applicability	2021	Data Integration

[S26]	Blog	Applicability	2021	Kainos
[S27]	Blog	Instantiation	2021	Helecloud
[S28]	Blog	Definition, Benefits, Challenges, Architecture	2021	K2View
[S29]	Slideset	Definition, Instantiation	2021	Oracle
[S30]	Blog	Tools	2021	Data Iku
[S31]	Blog	Definition, Use Case	2021	ITProportal
[S32]	Blog	Definition, Tools, Architecture	2021	MongoDB
[S33]	Blog	Definition	2021	Keboola
[S34]	Blog / Brochure	Benefits	-	IBM & KPMG
[S35]	Blog	Definition, Architecture, Challenges, Tools	2021	PeerIsland
[S36]	Podcast transcript	Definition	2021	ToroCloud
[S37]	Blog	Definition	2021	Advancing Analytics
[S38]	Blog + podcast	Definition	2019	Software Enginereing Daily
[S39]	Blog	Definition	2020	Zaloni
[S40]	Blog	Benefits, Tools	2021	Xenonstack
[S41]	Blog	Architecture	-	Nexla
[S42]	Blog	Definition, Architecture	2021	Medium
[S43]	Blog	Definition, Tools	2021	Snowflake
[S44]	Blog	Tools	2019	DreamFactory
[S45]	Blog	Instantiation	2021	DEV
[S46]	Blog	Definition	2021	Prog.World
[S47]	Blog	Definition, Architecture, Tools	2020	SAP
[S48]	Blog	Definition	2021	Sparke Quation
[S49]	Blog	Definition, Tools	2021	Juan Sequeda
[S50]	Blog	Definition	2021	Forbes
[S51]	Blog	Implementation	2020	Lakefs
[S52]	Blog	Definition	2021	Google
[S53]	Blog	Definition, Architecture	-	Dataception
[S54]	Blog	Definition, Instantiation	2021	Towardsdatascience
[S55]	Blog	Architecture, Tools	2021	Confluent
[S56]	Blog	Definition	2021	LinkedIn
[S57]	Blog	Definition, Architecture, Instantiation	2021	Francois Nguyen Blog
[S58]	Guide	Definition	2021	Montecarlodata
[S59]	Blog	Benefits	2021	Towardsdatascience
[S60]	Blog	Definition	2021	LinkedIn
[S61]	Blog	Benefits, Definition	2021	Vistaprint
[S62]	Blog	Benefits	2021	Towardsdatascience
[S63]	Blog	Definition, Benefits	2021	LinkedIn
[S64]	Blog	Definition, Benefits	2021	LinkedIn
[S65]	Blog	Definition	2021	FindHotel
[S66]	Blog	Architecture	2021	Medium
[S67]	Blog	Definition	2021	Medium
[S68]	Blog	Architecture	2021	Teradata
[S69]	Blog	Architecture	2021	Medium
[S70]	Slideset	Use Case, Tools	2021	Zenseact
[S71]	Blog	Instantiation	2021	Medium
[S72]	Blog	Benefits, Implementation	2021	Netflix
[S73]	Blog	Implementation	2021	AWS
[S74]	Blog	Definition, Benefits,	2021	OneModel
[S75]	Blog	Definition, Architecture, Benefits	2021	LinkedIn
[S76]	Blog	Architecture	2021	Privacera
[S77]	Whitepaper	Definition	2020	Thoughtworks
[S78]	Whitepaper	Implementation	2021	Oracle
[S79]	Video transcript	Implementation	2021	JP Morgan Chase & Company
[S80]	Blog	Definition	2021	Eckerson Group
[S81]	Blog	Architecture	2022	Towardsdatascience
[S82]	Blog	Architecture	2022	Towardsdatascience
[S83]	Blog	Architecture	2022	Towardsdatascience
[S84]	Blog	Definition	2022	Towardsdatascience
[S85]	Blog	Architecture	2022	Towardsdatascience
[S86]	Image of architecture	Architecture	2022	Deloite
[S87]	Blog	Implementation	2022	Microsoft
[S88]	Whitepaper	Definition, Instantiation	2021	Thoughtworks
[S89]	Blog	Instantiation	2021	DEV
[S90]	Blog	Instantiation	2020	ABNAmro
[S91]	Transcript of presentation	Use Case	2021	
[S92]	Blog	Architecture	2022	Towardsdatascience
[S93]	Blog	Definition	2021	Medium
[S94]	Blog	Architecture	2021	Medium
[S95]	Blog	Architecture	2021	Mulesoft
[S96]	Blog	Architecture	2021	Medium
[S97]	Blog	Definition	2021	InfoQ
[S98]	Blog	Architecture	2021	ITNext

[S99]	Blog	Definition	2021	Datanami
[S100]	Slideset	Use Case	2020	Zalando
[S101]	Blog	Architecture	2022	Towardsdatascience
[S102]	Blog	Instantiation	2022	Avenade
[S103]	Blog	Instantiation	2021	Medium
[S104]	Documentation	Instantiation	2021	Amazon
[S105]	Blog	Instantiation	2021	Medium
[S106]	Blog series	Instantiation	2022	Medium
[S107]	Blog	Use Case	2021	SiiconAngle
[S108]	Documentation	Instantiation	2022	Microsoft
[S109]	Blog	Definition	2022	Medium
[S110]	Blog	Architecture, definition	2022	INNOQ
[S111]	Blog	Architecture, definition	2021	Medium
[S112]	Blog	Architecture, definition	2021	Medium
[S113]	Blog	Architecture, definition	2022	Medium
[S114]	Blog	Definition	2022	Medium

3.2.2 Stage 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 (Saldaña, 2021). Coding was performed in Atlas.ti.

Similar to Kumara et al. (2021), an initial set of 20 sources was randomly selected and analyzed to establish an initial set of codes. These were discussed with the supervisor of the research team. Once a certain consensus on the initial set of codes was reached, the entire dataset was coded. Once the results were discussed within the research team, codes were grouped into overarching objectives and requirements. The subsequent chapters contain an overview of the labelled constructs further clarified in these chapters.

3.2.3 Descriptive Statistics

This section provides a brief overview of the descriptive statistics of the selected studies for the systematic gray literature review.

Figure 3.4 shows the search index for 'Data Mesh' on Google Trends for the past five years. A clear increasing trendline can be seen, indicating the increased interest in data mesh, hereby also substantiating the need for this research.

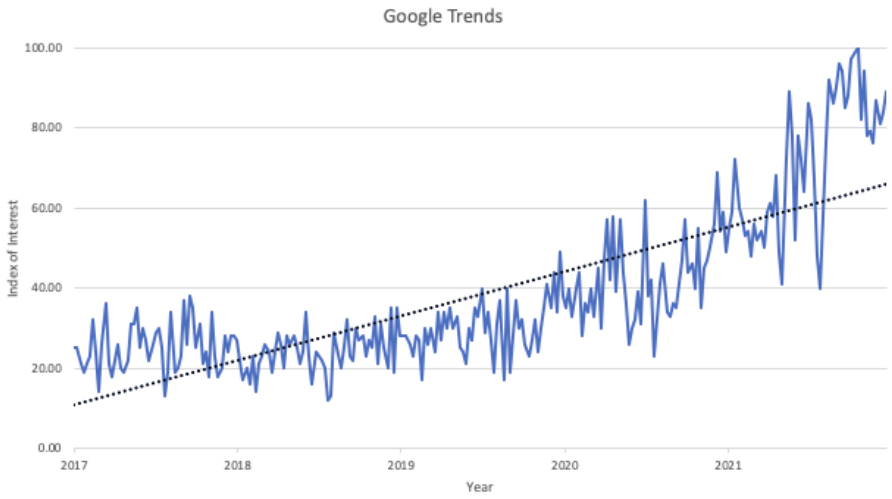

Figure 3.4: Google Trends for Data Mesh

Figure 3.5 shows the topics discussed in the selected studies. It is clear that most articles discussed the definition of data mesh, and fewer focused on its architecture. This imposed challenges for drafting the reference architecture, hence additional studies were included in the GLR that contained more information about the architecture.

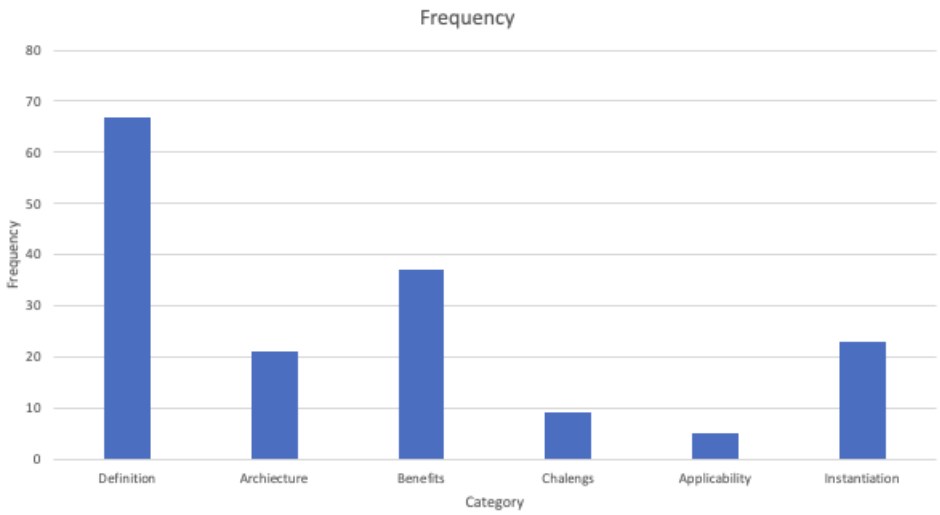

Figure 3.5: Topics in Studies

3.2.4 Reporting the Results

The results from the systematic gray literature are presented in tables throughout the thesis. Each category (e.g. the domain ownership principle) has its own table. Each category consists of several sub-categories. For each sub-category, the sources, atomic practices and frequency of codes are shown. The code frequency of a sub-category is constituted by adding the frequencies of its atomic codes. Table 3.2 below shows a sample table to illustrate how the results are presented. The thesis refers to sub-categories or atomic practices by referring to it's label. For example, **(D1.1.1)** refers to 'Sample atomic practice 1' and **(D1.1)** refers to 'Sample Sub-Category'.

Sub Category			**Atomic Codes**			
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Sources**
1.1	Sample Category	12	1.1.1	Sample atomic code 1	4	S1
			1.1.2	Sample atomic code 2	3	S2
			1.1.3	Sample atomic code 3	5	...

Table 3.2: Sample Table for Reporting the Results

Chapter 4

Data Mesh

This chapter presents the first artifact based on the systematic gray literature review by providing a general definition of data mesh (4.1), explaining its four principles: data product thinking (4.2), domain ownership (4.3), federated computational governance (4.4) and self-service infrastructure (4.5). Moreover it explains the expected benefits of a decentralized data mesh architecture (4.6), concerns of a data mesh implementation (4.7) and to which organizations a data mesh architecture is applicable (4.8).

All results are based on the systematic gray literature review or the feedback provided in the validation interviews conducted in stage five of the design science research methodology. Table 4.1, table 4.2, table 4.3, table 4.4, table 4.5 give an overview of the sub-categories derived from the gray literature review. Moreover it presents an overview of the atomic codes, their frequency and the studies that mentioned the codes. The results presented in the thesis refer to the sub-categories or codes through their label (e.g. **(D1.1.1)** refers to the code 'Data' belonging to the sub-category 'Components' part of the category 'Data Products'. In case results are derived from the validation interviews, the text explicitly mentions this.

4.1 What is Data Mesh?

Data Mesh is a decentralized big data architecture based on a conjunction of domain-driven design, self-service infrastructure platform, and data product thinking, all supported by a federated computational governance model. In short, data mesh aims to make business domains close to the origin of data, responsible for provisioning data as a product to the rest of the organization. Domains are supported by a self-service infrastructure platform that abstracts away some of the complexity of building and maintaining these data products. The architecture uses a federated computational governance model in which domains govern their data products while adhering to global standards. Moreover, automation plays a large role in this governance model.

A more extensive explanation of data mesh can be given based on its four principles - data product thinking, domain ownership, federated computational governance, and self-service infrastructure.

1 **Data as a Product thinking (Category 1)**
Data should no longer be seen as a byproduct coming from the organization but as an internal asset. Data assets are provisioned as data products that enable consumers to autonomously use the data. Data assets are provisioned as products to the rest of the organization

2 **Domain Ownership (Category 2)**
Teams closest to the origin of data become responsible for provisioning their data as products to the rest of the organization.

3 **Federated Computational Governance (Category 3)**
Governance responsibilities are delegated to the domains provisioning data products. These domains govern their data products while adhering to a set of global standards that aim to increase data products' interoperability. Moreover, the governance model is supported by automation.

4 **Self-Service Infrastructure (Category 4)**
A self-service infrastructure platform supports the domains by provisioning infrastructure necessary to build, maintain and manage data products.

Topologies

Gray literature describes several topologies for data mesh with varying degrees of decentralization **(D4.4)**. First of all, it recognizes the highly federated mesh, which is in line with Deghani's original work. In this topology, each data product is its architectural quantum. Data is distributed point-to-point. Secondly, gray literature defines a governed mesh topology that uses a centralized data distribution platform instead of point-to-point connections. In the last topology, the harmonized mesh topology, data is distributed point-to-point, but it is governed centrally at the same time.

The results presented in this thesis are for the highly federated mesh topology, extending from Deghani's original work.

Table 4.1: Category 1: Data as a Product.

Sub Category			**Atomic Codes**				
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Sources**	
						S1	S57
						S2	S67
						S10	S82
			1.1.1	Data	6	S13	S88
			1.1.2	Metadata	13	S24	S90
1.1	Components	40	1.1.3	Code	4	S28	S91
			1.1.4	Interfaces	12	S29	S97
			1.1.5	Infrastructure	5	S37	S98
						S42	S110
						S45	S113
						S53	
						S2	S95
						S15	S96
			1.2.1	Basic data products	7	S16	S98
1.2	Types of data products	34	1.2.2	Composite data products	16	S85	S109
			1.2.3	Requirement to add new value to the organization	11	S87	S110

Table 4.1: Category 1: Data as a Product (*continued*)

Sub Category			Atomic Codes				
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Sources**	
						S88	S111
						S91	S112
						S1	S55
						S2	S57
						S3	S63
						S7	S65
						S9	S66
			1.3.1	Discoverable	29	S15	S73
			1.3.2	Interoperable	16	S16	S74
			1.3.3	Natively Accessible	20	S18	S75
			1.3.4	Secure	20	S19	S77
			1.3.5	Self-describing	29	S21	S81
1.3	Characteristics of data Products	135	1.3.6	Trustworthy	21	S24	S82
						S28	S88
						S32	S91
						S33	S97
						S37	S100
						S38	S108
						S43	S113
						S47	
						S53	

4.2 Data as a Product

A data product can best be described concerning its components **(D1.1)**. For a data asset to become a data product, it needs to be extended with several things. First of all, besides the data itself, a data product includes metadata describing it. Secondly, the product incorporates the code responsible for consuming, transforming, and serving the source data transformed into the data asset (e.g., from source systems or upstream data products). Moreover, the product also includes code to enforce access control, policies, and other governance-related tasks. A product also possesses various interfaces that enable it to exchange data with other entities (i.e., incoming data or outgoing data) and management applications. Lastly, the product is incorporated in a product container that contains the necessary infrastructure to execute the code, store the data and make it accessible through the interfaces. In short, a data product provides both compute and data to the rest of the organization.

A distinction can be made between two types of data products: basic data products and composite data products. The first uses data from enterprise data sources as its source data and transforms it into the desired format. The processed data is then provisioned as a basic product to the organization.

Gray literature also refers to basic data products as source-aligned because the source systems mainly drive the product design **(D1.2.1)**. Composite data products use other data products (both basic or composite data products) for their source data and integrate them to apply complex transformations. The result is published as a data product to be used by consumers or downstream data products. The product's design is tailored towards the consumer's needs; hence gray literature also refers to them as consumer-aligned data products **(D1.2.2)**. A key requirement for data products is that they should provide value to the organization **(D1.2.3)**. A simple integration of two data products would not suffice to be published as a composite data product since this operation can easily be replicated. However, machine learning predictions resulting from integrating various upstream data products would suffice as this is a complex transformation that adds new value to the organization.

Characteristics of Data Products
Gray literature defines six qualities that data products should possess to provide value to consumers. Each quality is briefly described below **(D1.3)**.

Discoverable	*Secure*
Interoperable	*Self-Describing*
Natively Accessible	*Trustworthy*

*Discoverable **(D1.3.1)***
Data consumers should be able to discover the data products in an organization easily. All products and their metadata should be published in a central data catalog that enables consumers to browse and search data products easily. Discoverability aims to overcome the high costs and friction of data discovery in centralized data architecture.

*Interoperable **(D1.3.2)***
It should be (relatively) easy to combine or integrate data from different data products into composite data products. This can be realized by ensuring data products are interoperable. Ensuring interoperability of data products is one of the main concerns of a decentralized architecture with decentralized decision-making. However, interoperability is important because composite data products enable advanced use cases by using data from multiple domains.

Global standards can be used to increase interoperability between data products. Gray literature lists several global standards that a data mesh architecture should encompass.

- **Common metadata fields** ensure all data products contain a common set of metadata fields. This guarantees that all data products are described in a self-describing way.
- **Naming conventions** allow for uniformity throughout the architecture and ease of understanding.
- **Business glossary modeling** uses the same definition for common entities throughout the whole organization. This ensures all domains use the same definition for certain data and the same notation, which increases the interoperability of data products.
- A common method for **data quality modeling** ensures all domains define their product's data quality concerning the same KPIs.

Interoperability can also be achieved from a technical perspective by ensuring all data products share the same metadata description language (e.g. JSON) and have the same technical implementation for their interfaces by provisioning them through the self-service infrastructure platform.

*Natively accessible **(D1.3.3)***
All data products should provide a unique, programmatically accessible, multimodal access point depending on the data format. A description of the access point should be included in the metadata to enable consumers to bind to the data product.

*Secure **(D1.3.4)***
Access to data products should be limited to those authorized to use them. Identity Management (IAM) is managed centrally but should be executed in a decentralized fashion in which product developers authorize users to have access to the interfaces of their products.

Besides access control, data products should also be secure from a technical perspective. Data at rest and data in transit have to be encrypted. Overall, data should not be accessible by unauthorized users.

*Self-describing **(D1.3.5)***
Data products should be documented so that consumers can use them without any further assistance from the product owner or data steward. In short, they should be self-describing. This allows the organization to effectively scale out its number of data sources and number of consumers as the human dependencies are reduced as much as possible.

Although this list is non-exhaustive, gray literature mentions several aspects that make a data product self-describing. The facets below should be included in the product's metadata.

- **Product Owner Subject Matter Expert** so consumers know who to contact when needing more information related to the data.
- **Schema** for all the data views. Each view can have a different schema, for example, when certain features are masked due to sensitivity.
- **Service Level Objectives** that describe the product's quality and act as a data contract. More on this below.
- **Lineage Provenance** showing the origin of the data. This gives consumers trust in the data product as it becomes clear how the data is established.
- **Interface descriptions** that enable consumers to access the data. This explains how the consumer can bind to the interface.
- **Sample data set** that gives an impression of the data before access needs to be requested.

*Trustworthy **(D1.3.6)***
Data consumers should be able to trust the data coming from a data product. The exact definition of 'high quality' will differ for each data product. Product owners use a methodology provided by the federated governance team to model data quality to define their own product's quality on several dimensions (i.e. completeness, timeliness); more on this in section 4.4.

A set of Service Level Objectives (SLOs) captures the target quality for each data product and acts as a contract between the product owner and product consumer. This is meant to ensure high quality throughout the product's entire lifetime and to increase trust between participants in the architecture. Automation is used to support product developers in ensuring quality throughout the product's lifetime (e.g., automated monitoring, etc.).

Table 4.2: Category 2: Domain Ownership.

Sub Category			Atomic Codes				
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Sources**	
2.1	General Definition	28	2.1.1	Decentralized	12	S1	S52
						S9	S55
						S14	S66
						S19	S67
						S21	S78
						S22	S80
			2.1.2	Definition of domain	16	S24	S87
						S25	S103
						S37	S106
						S41	S109
						S48	
2.2	Responsibilities	40	2.2.1	Data Product	31	S4	S43
						S9	S47
						S13	S49
						S16	S53
						S20	S58
						S21	S63
						S22	S64
			2.2.2	Product Governance	9	S24	S67
						S25	S71
						S29	S73
						S35	S76
						S41	S77
						S42	
2.3	Roles	27	2.3.1	Domain Product Owner	16	S1	S39
						S2	S43
						S3	S49
						S16	S54
			2.3.2	Data Product Developer	9	S18	S67
			2.3.3	Consumers	2	S21	S73
						S25	S82
						S33	S87
						S35	S88

4.3 Domain Ownership

Responsibility for building data products is distributed to the business domains closest to the origin of the data. Data mesh makes use of the domain-driven design paradigm, so data products should be based on a domain model defined by domain experts. These domain experts have a thorough understanding of the data and thus know best what their data products should look like.

Organizations already use a separation of responsibilities by decomposing the organization into different business domains (e.g. HR, sales, R&D). These business domains form logical constituents for the distribution of ownership of data **(D2.1)**. However, especially for larger organizations, more granularity might be required. A further decomposition into subdomains is possible but not required. Bounded contexts are the conceptual boundary for a data product managed by a (sub-) domain. Each (sub-) domain can be responsible for building and maintaining one or more data products. Data products are built using a set of IT services provisioned by the self-service infrastructure platform, enabling a set of connectors to legacy data sources. The decomposition of domains into sub-domains and data products is shown in 4.1.

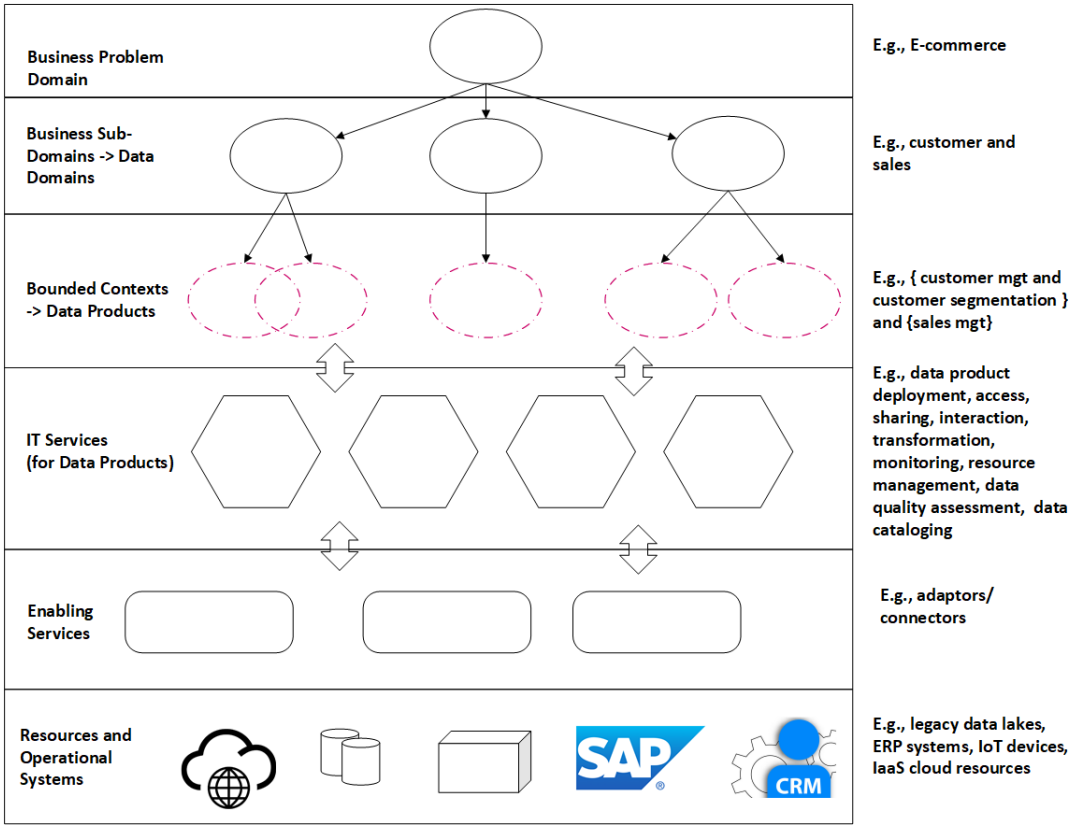

Figure 4.1: Decomposition of a business domain into data products.

Besides building and maintaining data products, the domains are also responsible for governing the product throughout its lifetime to ensure compliance and conformance to its SLOs **(D2.2)**. In short, each data product needs to be managed. More on this in the next section on federated computational governance.

Roles (D2.3)
Within each domain providing a data product, there are two distinct roles with their own set of responsibilities towards its data products.

- The **domain product owner** is the domain expert with in-depth knowledge about the origin of the data and is held accountable for the domain's products throughout their lifetime **D2.3.1)**. The product owner determines which data assets should be provisioned as a data product. To do so, the product owner needs a deep understanding of its consumers to make sure it can fulfill their needs. This includes the balancing and weighing of demands from different domains. Moreover, the product owner is responsible for managing the product throughout its lifetime.
- The **Data product developer** is responsible for the technical implementation of the domain's data products and their maintenance throughout their lifecycle **(D2.3.2)**. The developer uses the tools offered by the self-serve platform that abstracts away some of the technical implementations and thus reduces the required level of specialization for the product developer.

Table 4.3: Category 3: Federated Computational Governance.

Sub Category			**Atomic Codes**				
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Sources**	
						S1	S57
						S3	S58
						S4	S60
			3.1.1	General Governance Program	1	S9	S62
			3.1.2	Organization-wide standards	35	S15	S64
3.1	Global Governance	64	3.1.3	Business Glossary Modeling	6	S18	S74
			3.1.4	Global Policies	14	S19	S76
			3.1.5	Methodology for data quality	2	S21	S77
			3.1.6	Team composition	2	S25	S82
			3.1.7	Monitoring	4	S33	S90
						S35	S95
						S37	S103
						S38	S111
						S53	S114
						S2	S48
			3.2.1	Schema	5	S4	S63
			3.2.2	Access control at runtime	4	S13	S64
3.2	Local Governance	22	3.2.3	Compliance	2	S16	S71
			3.2.4	Data quality	6	S24	S88

Table 4.3: Category 3: Federated Computational Governance (*continued*).

Sub Category			**Atomic Codes**				
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Sources**	
			3.2.5	Operational Health	5	S28	S110
						S29	
3.3	Automation	20	3.3.1	Purpose	10	S1	S57
						S21	S82
			3.3.2	Specific needs to be automated	10	S26	S88
						S53	S110
3.4	Incentive Model	3	3.4.1	Incentives to create data products	1	S1	
						S88	
			3.4.2	Incentives to govern data products	2		
3.5	Balance between global and local governance	12			12	S1	S43
						S9	S63
						S25	S76
						S36	S82
						S37	S88

4.4 Federated Computational Governance

Data mesh uses a decentralized governance model in which domains are sovereign and govern their data products. Overarching governance on a global level across the complete architecture mainly aims to ensure interoperability between data products by setting global standards. To provide a more elaborate explanation of the federated computational governance model, a distinction is made between the activities conducted on a global level (by the federated governance team) and activities conducted on a local level (by domains).

Balancing the degree of centralization and decentralization can be challenging in the federated governance model **D3.5**. 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. Finding this equilibrium will be challenging.

4.4.1 Global Governance

The federated governance team has several responsibilities with regards to the global governance of the decentralized architecture **(D3.1)**.

First of all, the team is responsible for defining the general governance program in line with how it is described in section 2.2 **(D3.1.1)**. Logically, Abraham et al. (2019) conceptual framework is not fully applicable to data mesh due to the decentralization of responsibilities in data mesh. However, a large part of the framework is still relevant. I.e. the team will still have to formalize a set of procedures

that can be used to resolve issues, formalize communication methods, etc.

Moreover, the team is responsible for drafting organization-wide standards to increase the interoperability of data products **(D3.1.2)**. All data products are to adhere to this set of standards. Examples are shared API interface description language or a standard language for metadata modeling (e.g. JSON). Automation is used to scale these policies and standards throughout the organization effectively. Standards are also used to ensure data products adhere to security requirements modeled by the federated governance team.

Furthermore, what can perhaps be seen as a standard, is a common business glossary for the organization **(D3.1.3)**. The data mesh principles are against creating one canonical data model for the whole organization as this goes against decentralization. However, literature does recognize 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 drafting a list of these concepts and reaching a consensus on a common definition. This is also known as business glossary modeling.

Besides a set of standards, the federated governance team also publishes a set of policies **(D3.1.4)**. Policies can be seen as a set of rules that all participants in the architecture are to adhere to. An analogy can be made to governance policies and service policies in a service-oriented architecture. The first refers to high-level rules that define what to do and how. Service policies, or in the case of data mesh product policies, are rules that only apply to specific data products (Bernhardt & Seese, 2008). The latter are among the product owner's responsibilities, more on this below. For example, global policies can be used to enforce a minimum level of security throughout the organization or enforce access control based on central identity management.

Lastly, the federated governance team develops a methodology to define data quality to be used by data product owners **(D3.1.5)**. The methodology consists of dimensions that contribute to data quality data (e.g. timeliness of data, completeness of data). After all, the exact definition of high-quality data will differ for every data product. A standardized methodology ensures that all data products define quality on a fixed set of dimensions with shared definitions.

Team composition
The federated governance team consists of representatives from the domains, subject matter experts, governance experts, members from the legal department, platform team, management and more **(D3.1.6)**. The team composition should be so that there is a balanced representation of all interests in the organization.

4.4.2 Local Governance

Local governance is related to the governing of individual data products. The product owner, potentially in combination with other domain members, is responsible for conducting the local governance activities. The gray literature mentions several responsibilities concerning product governance outlined below **(D3.2)**:

- As described above, data mesh does not use one canonical data model for the whole organization. Using their domain knowledge, the product owners decide on the product's schema **(D3.2.1)**. The schema can be changed throughout the product's lifetime to reflect the business functionality better, or the schema can be extended.
- Managing access control over the lifetime of a data product **(D3.2.2)**. Domain experts have a thorough understanding of the data and can, therefore, best judge who should and who shouldn't have access to (which parts of) the data.
- Managing compliance **(D3.2.3)**. Similar to managing access controls, it is impossible to ensure compliance throughout the product's lifetime without thoroughly understanding the data.

- The local team is also responsible for managing data quality **(D3.2.4)**. Using the methodology defined by the federated governance team, product owners define the product's target quality on a fixed set of dimensions.
- Moreover, throughout the product's lifetime, the product owner is responsible for monitoring its operational health and ensuring its capacity is sufficient for the needs of consumers **(D3.2.5)**.

4.4.3 Automation

Automation plays a significant role in the federated computational governance model to ensure it can effectively scale throughout an organization **(D3.3)**. 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. Gray literature gives several examples of actions that can be automated in a data mesh architecture, some of which are listed below.

- Encryption
- Data classification (e.g. sensitivity labels)
- Data quality checks
- Lineage detection
- Data retention and deletion to ensure compliance
- Automated monitoring including alerts and automatic responses (e.g. upscale a resource to meet demand, measure quality with respect to SLOs)
- Policy enforcement

A subject that is relatively untouched in gray literature but was mentioned several times in the validation interviews is how multiple policies can simultaneously be implemented. For example, how can global and local policies be combined to ensure both requirements are met? If there are contradicting policies, how can they be integrated from a technical perspective, and which should be prioritized? These challenges are not yet addressed by gray literature but are essential to resolve to make the governance model work in practice.

4.4.4 Incentive Model

Organizations implementing or transforming to a data mesh architecture need to develop an incentive model to motivate domains to provision their data as a product and to govern them **(D3.4)**.

Provisioning data products will add to the domain's normal workload and will most likely not fit in their traditional job description (i.e. supply chain department focuses on optimizing delivery routes, not on provisioning data). Without proper incentives, domains might not see a reason to build data products **(D3.4.1)**. Especially to domains that only make limited use of data from other domains, the rewards of provisioning their data as products will most likely seem negligible compared to the additional load. Monetary rewards can be used to motivate domains to build provision data products. Money can be a good incentive when just starting to transform to a data mesh architecture, albeit it is more desirable to have the provisioning data products become part of each domain's job description as time passes.

Similarly, organizations will need to have an incentive model to motivate domains to govern their data products once deployed **(D3.4.2)**. Without such a model, there is the risk of getting a high number of low-quality data products that cannot provide any value to the organization. Moreover, organizations need to think about how to allocate costs related to data products. Should costs be allocated to the provisioning domains or the consuming domains? These questions are not easy to answer and will most likely require a lot of internal discussions.

Having proper incentive models was mentioned in gray literature. However, the validation interviews stressed their importance in successfully implementing data mesh.

Table 4.4: Category 4: Self-Service Infrastructure.

Sub Category			**Atomic Codes**				
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Sources**	
4.1	Objectives	31	4.1.1	Reduce required specialization	8	S1 S2 S4 S7 S9 S19 S25 S33	S43 S53 S58 S72 S85 S87 S92
			4.1.2	Increase efficiency	20		
			4.1.3	Stimulate uniformity	3		
4.2	Building self-service infrastructure	14	4.2.1	Central Platform Team	5	S2 S7 S13 S22 S42 S53 S54	S73 S85 S93 S97 S105 S111 S58
			4.2.2	Blueprints	9		
4.3	Components	81	4.3.1	Polyglot Storage	12	S2 S3 S4 S7 S10 S12 S13 S21 S28 S29 S33 S35 S37 S42 S50	S53 S58 S62 S64 S71 S72 S73 S80 S87 S97 S98 S102 S110 S111 S113
			4.3.2	Compute	2		
			4.3.3	Pipeline Orchestration	12		
			4.3.4	Security	10		
			4.3.5	Monitoring	6		
			4.3.6	Policy Enforcement	10		
			4.3.7	Networking	5		
			4.3.8	Metadata Repository	2		
			4.3.9	IAM Management	10		
			4.3.10	Lifecycle Management	9		
			4.3.11	Distributed Query Engine	3		

4.5 Self-Service Infrastructure

Domains are supported by the self-service infrastructure platform that provisions infrastructure necessary to build, maintain and manage data products. The general thought is that (relatively) generalist product developers interact with a user-friendly interface to request the infrastructure necessary to build and maintain their data products. Using the provisioned interface, they can build and implement their data pipelines to transform the data. The platform abstracts away the complexity of infrastructure management in a domain agnostic way.

Contrary to the situation in which domains would be responsible for building and maintaining their infrastructure, using a self-service infrastructure platform has several benefits **(D4.1)**. First of all, it reduces product developers' required level of specialization as they do not need to possess in-depth knowledge of infrastructure management **(D4.1.1)**. Secondly, it also increases efficiency as there is no duplication of efforts by having each domain reinvent the wheel **(D4.1.2)**. Lastly, the common infrastructure platform increases uniformity, reducing friction between domains and facilitating interoperability **(D4.1.3)**.

The platform is built and maintained by a centralized data platform team, which consists of highly specialized infrastructure developers **(D4.2.1)**. Based on the needs of the domains, the platform team decides which features to develop. The interviews stressed that it can be challenging to decide what should become part of the platform and especially in what order. If the platform offers too few capabilities, domains will not use the platform and build their tooling. This leads to shadow IT that is not compatible with the general platform and thus is also not integrated into the computational governance framework. On the other hand, offering too many services will lead to a complex and diverse platform that is difficult to maintain. Perhaps these tools are not domain agnostic and thus tailored too much to the needs of individual domains.

To overcome this challenge, the central platform team has to be in close contact with all domains to identify and prioritize their needs based on many discussions with representatives from all domains. There should be some form of consensus between the domains on what infrastructure should be part of the central platform. This is most likely a lengthy process involving many complex and political discussions.

Below is a general overview of the platform capabilities mentioned in gray literature **(D4.3)**. A more extensive explanation is given in chapter 5.

- Polyglot Storage
- Compute
- Pipeline Orchestration
- Security
- Monitoring
- Policy Enforcement
- Networking
- Metadata Repository
- IAM Management
- Lifecycle Management
- Distributed Query Engine
- Product Interfaces

Something not stressed in gray literature but mentioned in the validation interviews is the importance and complexity of lifecycle management for infrastructure in a distributed environment. Infrastructure will have to be maintained and updated to incorporate new features. However, it can be challenging to do this in a distributed environment where domains run their deployments. Also, there must be some degree of abstraction for infrastructure (i.e. through interfaces) that facilitates change to new technologies.

Blueprints
Infrastructure is provisioned through blueprints **(D4.2.2)**. These contain a predefined set of capabilities that domains can use to build a data product. Blueprints encompass a predefined configuration for the infrastructure itself and integrated security measures (e.g. private networks), policies, and standards in line with those set by the federated governance team. Multiple blueprints can be provisioned to fulfill all the needs of a domain. An example is a blueprint that deploys a data lake with access API or a blueprint that provisions a machine learning platform that domains can use to build models and bring them to production.

Table 4.5: Category 5: Benefits, concerns and applicability of data mesh.

Sub Category			**Atomic Codes**				
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Sources**	
5.1	Benefits	49				S3	S28
			5.1.1	Horizontal scaling	8	S4	S40
			5.1.2	Vertical scaling	12	S5	S48
			5.1.3	Agility	7	S9	S60
			5.1.4	Data quality	5	S10	S62
			5.1.5	Better governance	5	S12	S69
			5.1.6	Data discovery	2	S13	S75
			5.1.7	Reduced Data Lead Time	10	S18	S77
						S23	S92
5.2	Concerns	30				S3	S35
						S4	S41
			5.2.1	Change management	13	S9	S46
			5.2.2	Lack of talent	5	S16	S48
			5.2.3	Duplication of data	8	S18	S58
			5.2.4	Duplication of efforts	4	S21	S63
						S24	S77
						S28	S87
5.3	Applicability	45				S3	S25
			5.3.1	Large and diverse data landscape	27	S4	S26
			5.3.2	Need for agility	7	S5	S35
			5.3.3	Need for better governance	7	S7	S58
			5.3.4	When not to use data mesh	4	S9	S62
						S12	S98
						S22	S110

4.6 Benefits of Data Mesh

The data mesh principles aim to address the challenges discussed in the problem indication in section 1.1. Below a short overview is given of the benefits of data mesh compared to traditional data architectures.

1 **Scalability (D5.1.1 & D5.1.2)**
Data mesh is much more scalable than a centralized data architecture since dependencies on a central data provisioning team are reduced. This allows the organization to scale both horizontally (i.e. more data sources) but also vertically (i.e. more consumers) without creating a bottleneck.

2 **Reduced Data Lead Time (D5.1.7)**
Domains can independently provision and maintain data products, and consumers can independently use these products through the self-service model. This greatly reduces the data lead time throughout the organization.

3 **Increased Agility (D5.1.3)**
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 will be able to create more data applications that unlock the value of their data.

4 **Higher Data Quality (D5.1.4)**
By making those close to the origin of the data responsible for provisioning it into a data product, data quality will be higher. A set of global standards and policies supports the domains in creating high-quality data products while they can incorporate their domain-specific knowledge.

5 **Better Governance (D5.1.5)**
Lastly, data mesh allows for better governance of big data. 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. These domains have a better understanding of the data and can make better judgments about who should and who should not be able to use certain data in certain conditions. This will also facilitate organizations to be compliant with external regulations.

6 **Discoverable Data (D5.1.6)**
Clear ownership of data makes it easier to make data discoverable as there is a clear overview of the available data products.

4.7 Concerns of Data Mesh

Logically, data mesh comes with its own set of challenges, which is also recognized by gray literature. Below an overview of the main concerns for data mesh are given.

1 **Change Management (D5.2.1)**
For established organizations, it will be challenging to move to a data mesh architecture from a technical perspective and an organizational perspective. Organizations already have established processes and a hierarchical division that will likely have to be restructured. Moreover, a change in mindset or change in culture will be required, which is very challenging to achieve. Effective leadership and strong change management will be required. It is good to recognize that data mesh is not only a technological solution but also an organizational solution and that this comes with its

own set of challenges.

2 **Lack of Talent (D5.2.2)**
Data mesh requires domains to start operating independently, meaning that they all require the technical expertise to do so. One concern is the lack of available talent in the labor market. Experienced data engineers are in high demand all across the globe. Filling the required positions in the current environment with incredible shortages in the labor market will be challenging.

3 **Duplication of Data (D5.2.3)**
As domains repurpose data for new use cases in composite data products, data from source domains will be replicated. Especially when implemented on a large scale, data mesh will increase data management costs. Furthermore, as data is stored in different locations in an organization, it will become more difficult to govern.

4 **Duplication of Efforts (D5.2.4)**
Even though the self-service infrastructure tries to reduce the duplication of efforts concerning infrastructure management, there will still be a certain degree of duplication of efforts and skills for building and maintaining pipelines throughout the different domains.

4.8 When to Use Data Mesh

Data Mesh is not the most practical architecture design for all organizations. The decentralization of responsibilities and decentralized architecture creates additional complexity from both a technical and organizational perspective. Moreover, building and maintaining the infrastructure is more expensive. Gray literature advises against adopting data mesh by SMEs as a centralized architecture is still the better option for them **(D5.3.4)**. Gray literature does recognize several conditions for which data mesh is more suitable compared to a centralized data architecture **(D5.3)**.

1 **Large and Diverse data landscape (D5.3.1)**
Organizations with a large data landscape consisting of many data providers and many data consumers might benefit from a decentralized architecture. Also, when organizations possess diverse data sets that frequently change over time, data mesh might serve better than a centralized approach with many point-to-point connections.

2 **Need for agility (D5.3.2)**
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 a large and overloaded data provisioning team.

3 **Need for better data governance (D5.3.3)**
Centralized data lakes can easily turn into data swamps without an overview of the available and no data ownership. Data mesh can be a resolution for these issues for organizations in need of clear ownership of data and highly governed data based on domain knowledge. Organizations that face a lot of external regulations (i.e. banks) can make use of data mesh to ensure compliance as it facilitates better data governance.

Chapter 5

Data Mesh Reference Architecture : Design-time View

This chapter builds on the definition of data mesh provided in the previous chapter. The concepts defined above are placed in a reference architecture at design-time that structures the different constructs, roles, and responsibilities in a data mesh architecture regarding creating managed data products. The subsequent sections present the layered architecture and a description for each layer. The results refer to the sub-categories and atomic codes defined in the tables of the previous chapter by referring to them like this: **(DX.X)**.

5.1 The Layered Architecture

A layered architecture is used to depict the different constructs, roles, and their responsibilities in a data mesh architecture. The figure, presented in fig. 5.1, is inspired by M. P. Papazoglou and Van Den Heuvel (2007) xSOA and uses three layers to logically separate concerns. Each layer defines a set of constructs, roles, and responsibilities and builds on the constructs of the preceding layers. The separation into layers highlights the different needs regarding 1) providing the infrastructure and services necessary for realizing and consuming data products, 2) creating, consuming, and composing data products and 3) managing data products throughout their lifecycle. Important to highlight is that, although higher layers build on lower layers, the layers only represent a separation of concerns and do not imply hierarchy.

The layered architecture shown in fig. 5.1 does not depict all responsibilities in a data mesh architecture. Instead, the architecture provides a view for the different responsibilities to create *individual* managed data products at *runtime*. For example, the federated governance team's responsibility to create the business glossary is not included in this architecture because this is not related to creating a specific managed data product at runtime.

5.2 Roles

First, the various roles in a data mesh architecture and their relation to each other and the other constructs are defined.

- The **Data Platform Team** is responsible for building and maintaining the self-service infrastructure **(D4.2.1)**. They build the blueprints that product developers can use to provision the infrastructure necessary for building and managing data products.
- **Product Developers** build and publish data products on the data mesh using the tools provided by the self-service layer **(D2.3.2)**.

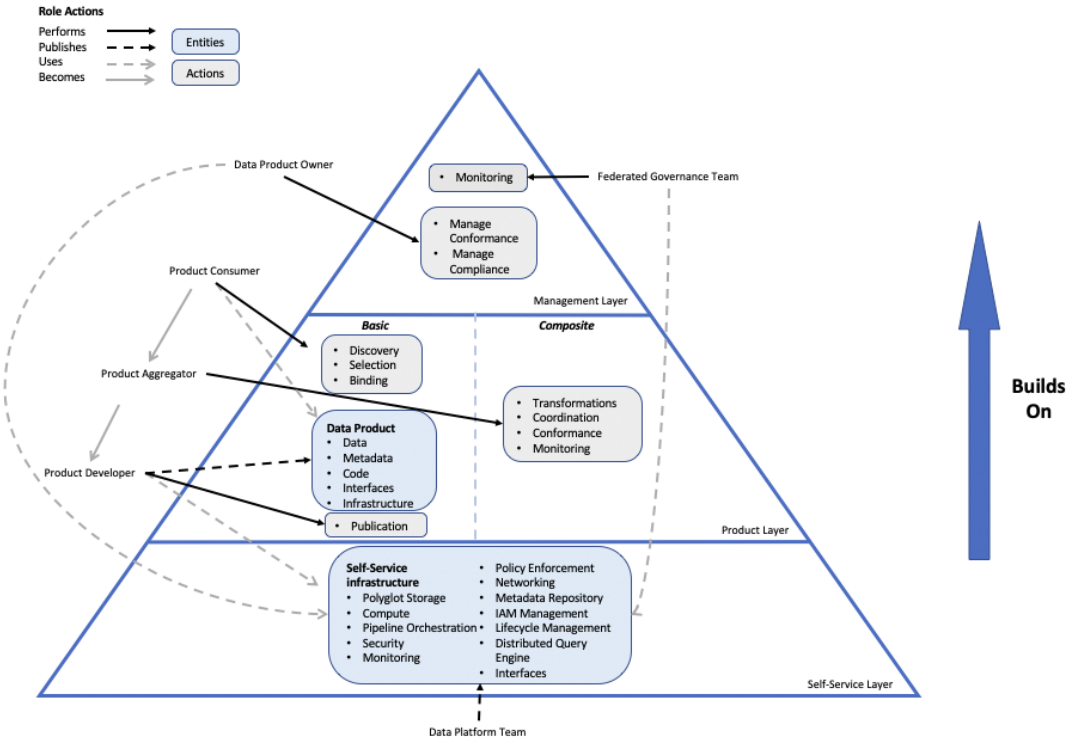

Figure 5.1: The Layered Architecture.

- **Product Aggregators** build and provide composite data products. From the outside, composite data products consist of the same components as basic data products. However, there is an additional set of responsibilities for the product aggregator to create a composite data product. More on this in section 5.4.
- **Product Consumers** use the data from data products **(D2.3.3)**. Product aggregators are also product consumers since they consume multiple other data products and aggregate them into a composite data product.
- The **Federated Governance Team** is composed of representatives from different domains across the organization and governs the mesh from a global perspective **(D3.1.6)**.
- **Product owners** are responsible for managing the data products in their domain throughout their lifecycle **(D2.3.1)**.

5.3 Self-Service Layer

The self-service infrastructure (SSI) forms the foundational layer of the mesh architecture. It offers a set of tools in a self-service model that the actors in a data mesh architecture can use to fulfill their responsibilities. Product developers can use it to deploy, manage and monitor their data products while abstracting away the complexity of the infrastructure enabling these tools. Product owners and the federated governance team can use it to monitor and manage data products.

The platform is developed and maintained by the central data platform team. Product developers interact with the platform through an interface that abstracts away the complexities of the infrastructure itself. The interface allows developers to request infrastructure, after which it is automatically deployed. Using the provisioned infrastructure, the developers build their data product. For example, any code to perform ETL will be implemented in orchestration tools to automate the ETL. Moreover, the self-service infrastructure should provide monitoring tools integrated with the other infrastructure, enabling product owners to monitor individual data products and the federated governance team to monitor the mesh from a global perspective.

Below an overview of all the services offered by the self-service infrastructure is given in line with the gray literature review **(D4.3)**.

- **Polyglot Storage (4.3.1)**
  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 database. In contrast, other data products might consist of highly unstructured data that can best be stored in a data lake (e.g. video). The SSI provides different technologies that support polyglot data.
- **Compute (4.3.2)**
  The platform should offer different types of compute for any data processing to make the data ready for consumption. Examples are virtual machines or compute clusters that perform compute for ETL.
- **Pipeline Orchestration (4.3.3)**
  To automate workflows, the SSI should offer tools to schedule, execute and coordinate pipelines. Pipelines can automate code execution to transform the data. For example, when new data becomes available, these pipelines can automatically preprocess, integrate and transform the data based on the code built by the product developer to then serve the latest data to consumers. Pipelines enable automation in the mesh architecture.
- **Security (4.3.4)**
  This is a broad overarching term indicating security should be embedded in all the tools offered by the SSI. Depending on the exact implementation, the platform needs to offer the ability to encrypt data at rest and data in motion. Moreover, data products need to guarantee confidentiality by only providing access to authorized users. Both encryption and authorization increase systems resilience (Stanton, 2004).

  Various encryption algorithms are available with differing levels of security and speed of the algorithm (Nadeem & Javed, 2005). Depending on the needs of an organization, they can choose themselves what encryption algorithms to use.
- **Monitoring (4.3.5)**
  The platform should offer a set of tools that enable domains to monitor their data products and the federated global governance team to monitor the architecture 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 allow for logging, alerts, and automated interventions (e.g. upscaling of a resource). To realize this, the monitoring tools need to be integrated with all the other tools offered by the SSI.
- **Policy Enforcement (4.3.6)**
  To implement policies and standards, the SSI should offer tools that allow both the local and global federated governance teams to enforce policies. Second, to this, the infrastructure should provide a sidecar container that executes policies locally at each data product, more on this in section 6.5. The sidecar prevents a centralized policy enforcement entity from becoming a bottleneck in the architecture. Policies should be written as code so they can be executed programmatically.
- **Networking (4.3.7)**
  The platform should offer various networking capabilities. Infrastructure should communicate through virtual networks to create a closed-off environment from the public internet. Additional resources such as domain name servers, load balancers, etc. need to be provisioned by the platform.
- **Metadata Repository (4.3.8)**
  Global metadata, such as standardized definitions, should be available in a self-service fashion so product developers can incorporate it into their data products. This includes but is not limited to the business model glossary, global policies, and global standards.
- **IAM Management**
  Identities in the architecture are managed centrally. However, product developers should have

the ability to control access to their data products on a local level. The platform should provision tools to do this.

- **Lifecycle Management (4.3.9)**
  The platform should offer capabilities for product versioning, source control, testing, etc. This allows product developers to manage the lifecycle of a data product. An analogy can be made with DevOps practices that provide a framework to integrate development and operations in software development (Ebert et al., 2016). A similar process can be applied to manage the lifecycle of data products.

  For example, if the interface of a data product is changed, the old version should not be decommissioned before all consumers have switched to the new interface. The platform should enable product developers to manage this workflow easily.

- **Distributed Query Engine (4.3.10)**
  To enable product aggregators to create composite data products, the self-service platform should offer a distributed query engine that can easily query data from different domains. The query engine is, of course, only relevant for structured data that can actually be queried (i.e. data warehouses / databases).

- **Interfaces (4.3.11)**
  Although not mentioned often in gray literature, the interviews stipulated that the self-service platform should provide data product interfaces. Product developers then only need to tailor them to their data product.

5.4 Data Product Layer

The data product layer sits on top of the self-service layer. Product developers build and publish data products using the infrastructure provisioned by the self-service platform. This section describes the actions performed to publish basic and composite data products, the data product as a construct, and the actions performed to consume data products.

Basic Data Products

The data product marked in blue in fig. 5.1 depicts the different components a product developer needs to build before publishing a data product. A product consists of data, metadata, code, interfaces, and the infrastructure to deploy the product **(D1.1)**.

Logically, a product's main component is its data. Basic data products serve data from source systems (e.g. ERP systems). Data mesh does not make use of a single canonical data model for the whole organization, so product developers are responsible for defining the schema for their data. The schema should be business-driven, meaning that it is not solely the output of the source system that determines the product's schema but mainly the business functionality that the product tries to serve. Therefore, in case of a change in the source system, there should be minimal schema changes as the business functionality does not change. Making sure the schema is business-driven, increases the interoperability of data products throughout their lifetime.

A data product can serve different views of the same data **(D1.1.1)**. Views can be made based on time-based groupings (e.g. daily or weekly) or segments of the data product (e.g. only non-sensitive data). The product developer is responsible for serving these different views, which likely entails building different ETL based on the same source data. Moreover, the product developer decides if the product should provide batch or streaming data. The first processes new data on a fixed interval (e.g. once a day) and then makes the new data available. The latter processes new data as it becomes available and immediately serves it to its consumers.

Secondly, data products possess metadata to make the product self-describing **(D1.1.2)**. The metadata consists of the facets described in section 4.2. The product owner is responsible for drafting usage policies for his data products. These policies define how downstream consumers can use the

product. For example, if certain data is sensitive, the policies should clarify that consumers can only use and provision the data in a way that respects this sensitivity. The included metadata aims to make a data product self-describing, discoverable, and interoperable, enabling consumers to use the product without any human intervention from subject matter experts. Each data view published by the product developer should have separate metadata because the schema, SLOs, or policies might be different for each view.

Thirdly, a data product contains code **(D1.1.3)**. Data products package data and compute as one entity. Code used for ETL to transform incoming data from operational systems is part of the product itself. Moreover, the policies described above are implemented as code (Policy as Code) to enable computational governance. This allows for a certain degree of automation in executing and managing these policies. Lastly, the product's interfaces are written in an interface description language. The code for these interfaces is also included in the data product.

Each data product also incorporates a set of interfaces that allow for data exchange or consumption and communication with management applications **(D1.1.4)**. Interfaces are provided by the self-service infrastructure but need to be implemented into the product developer's product.

The last component of a data product is the infrastructure itself **(D1.1.5)**. All components above are implemented on the infrastructure deployed from the SSI. Altogether, it provides a complete data product.

Once the product developer defines and implements all components, the data product can be published to the data catalog. Product consumers browse the catalog to discover candidate data products for their use case. The consumer checks the corresponding metadata to learn more about the data product and see if it suits their needs. The consumer also knows how to perform a binding to the data product to receive data based on the metadata.

Composite Data Products

A composite data product uses data from other data product(s), applies a complex transformation, and publishes the result as a new data product **(D1.2.2)**. Composite data products contain the same components as basic data products and deliver the same functionality to consumers. However, to build a composite data product, an additional set of actions must be performed, similar to the additional responsiblities in xSOA (M. P. Papazoglou & Van Den Heuvel, 2007). To distinguish between these responsibilities, the layered architecture in fig. 5.1 differentiates between product developers, who build basic data products, and product aggregators, who build 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 does not provide sufficient additional value. Therefore, the first action for a product aggregator is to perform a complex transformation on the data from upstream data products.

Secondly, when new data becomes available in one of the upstream data products, product aggregators need to perform some kind of coordination to dictate how and when this new data should be used. For example, the new data might need to be pulled from the data product, or it may be received through asynchronous message-based communication. Coordinating the flow of new data is among the responsibilities of a product aggregator.

An example is provided to illustrate what coordination entails; see fig. 5.2. Imagine composite data product 3 (DP3) uses two upstream data products that provide data about the sales history of customers and marketing campaigns. DP3 applies a machine learning application to predict which marketing campaigns are effective for certain customer groups. Suppose DP2 publishes new data to DP3 about a newly deployed marketing campaign, but the associated sales data coming from DP1 is not yet available. In that case, it might not make sense to retrain the ML algorithm in DP3. The product aggregator is responsible for creating the coordination between these data sources.

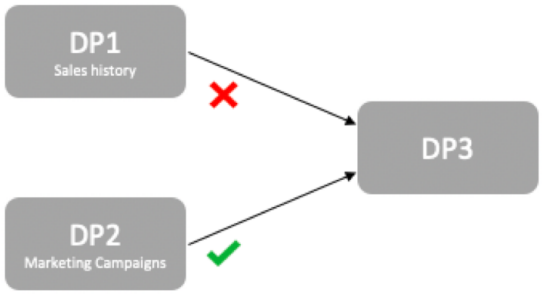

Figure 5.2: Data Products.

Conformance checking is the third additional responsibility for the product aggregator. Upstream product developers might change their data product (e.g. schema or interface) that affects downstream consumers. Product versioning is in place to prevent these changes from creating compatibility issues for downstream consumers. However, product aggregators will have to adjust their products accordingly to conform to new versions of upstream data products so that the old versions can be decommissioned. If products cannot be decommissioned over time, the number of data products will become unmanageable as there are too many products to maintain. Ensuring conformance throughout the composite product's lifetime is among the responsibilities of the product aggregator.

Lastly, product aggregators need to monitor the performance of upstream data products concerning the SLOs. Logically, the upstream product developer's responsibility is to ensure that their data products operate within the specified SLOs. However, the performance of the composite data product is affected by the performance of the upstream data products thus, a product aggregator needs to monitor performance to ensure his product can operate within its own SLOs. Monitoring is the final additional responsibility of the product aggregator.

5.5 Management Layer Layer

At runtime, data products need to be managed by both the product owner and the federated governance team to ensure they can provide value to the organization. This section describes how product owners manage their products from a local perspective and how the federated governance team manages products from a global perspective.

Local Governance
The local governance teams are responsible for governing individual data products within their business domain. Among their responsibilities are:

- **Managing Conformance (D3.2.4 & D3.2.5)**
  Product developers are responsible for ensuring that a data product operates within its specified SLOs throughout its lifetime. For example, the developer needs to ensure the timely availability of data that is in line with the specified quality metrics.
- **Managing Compliance (D3.2.3)**
  The product developer needs to ensure the product remains compliant with local and global policies and external regulations (e.g. privacy regulations). Data Mesh moves this responsibility to the local governance team as they fully understand the data and therefore know best who should have access to the data and what the data can be used for.

Global Governance
The main responsibility of the federated governance team is to monitor the mesh as a whole **(D3.1.7)**. Of course, the team is also responsible for drafting standards, guidelines, etc. but those actions do not directly relate to creating a managed data product at runtime.

The global federated governance team monitors the mesh with respect to various KPIs:

- **Conformance to global policies:** do the data products adhere to global policies (e.g. security)?
- **Usage:** Is the usage of data products scaling out throughout the organization?
- **Operational performance:** does the architecture perform well in relation to the operational demands?

The main goal of global governance at runtime is to see if the mesh architecture is able to scale throughout the organization effectively.

Chapter 6

Data Mesh Reference Architecture : Runtime View

This chapter presents the third artifact from the design science research methodology: the runtime view for the reference architecture of data mesh. Similarly to the previous chapters, the results are based on requirements derived from the systematic gray literature review. Table 6.1 gives an overview of the sub-categories derived from the gray literature review. Moreover it presents an overview of the atomic codes, their frequency and the studies that mentioned the codes. The results presented in the thesis refer to the sub-categories or codes through their label (e.g. **(D6.1.1)** refers to the code 'Input' belonging to the sub-category 'Product Interfaces' part of the category Architecture Components at Runtime.

6.1 Reference Architecture at Runtime

Figure 6.1 depicts the reference architecture at runtime for data mesh. This section summarizes the different components and how they relate to each other, on which the subsequent sections elaborate.

The product container acts as a logical container for data products and their sidecar (not to be confused with docker containers). Each product possesses three interfaces to allow data exchange and communication with management services. The input interface enables data products to receive new data **(D6.1.1)**. New data can be received by 1) connecting to enterprise data sources through connectors, 2) subscribing to other data products through the data channel, or 3) making ad hoc data requests to other data products through an API request. The output interface enables data products to provide their data to consumers or composite data products **(D6.1.2)**. Data can be supplied through an API (pull-based data exchange) or the publish / subscribe pattern (push-based data exchange) **(D6.3.1 & 6.3.2)**. The data channel represents the message broker that handles data routing in the publish / subscribe pattern **(D6.3.3)**.

The self-service infrastructure platform is the supportive layer that provisions the infrastructure necessary to run the components in the product container. It also provides a set of management applications for the product owner to govern their data products. The management applications use the publish / subscribe pattern to connect to the management interface of data products **(D6.1.3)**. The management channel acts as the message broker to handle the routing of management-related messages **(D6.3.4)**. The governance plane is also connected to the management channel to provide a global view of all data products.

The data catalog is connected to the management channel. Product developers publish their products to the catalog over the management channel. Consumers can discover data products in the catalog by reading their metadata.

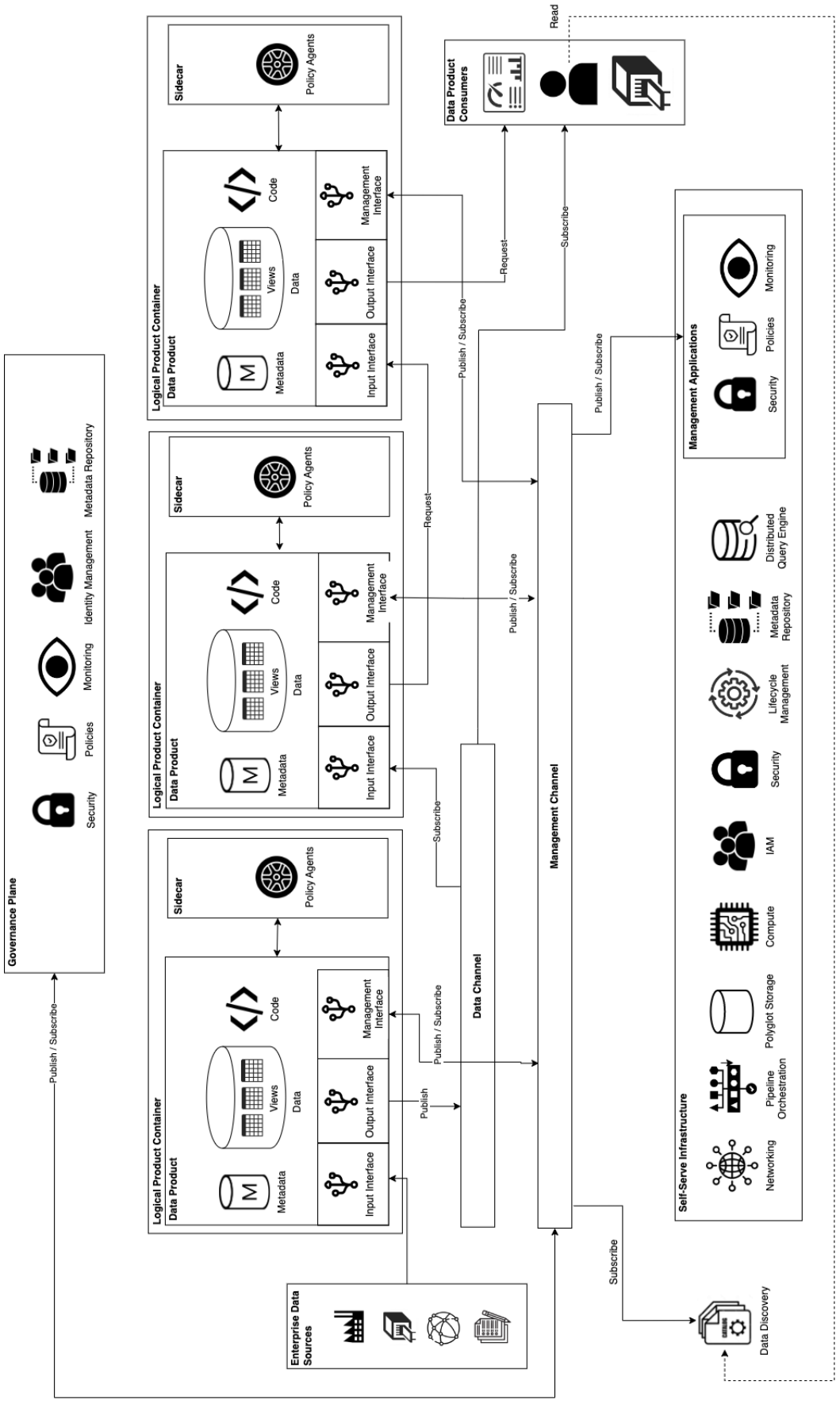

Figure 6.1: Reference Architecture at Runtime

Table 6.1: Category: Architecture Components at Runtime. .

Sub Category			Atomic Codes			
ID	**Name**	**Frequency**	**ID**	**Name**	**Frequency**	**Sources**
6.1	Product Interfaces	12	6.1.1	Input	4	S51 S72
			6.1.2	Output	3	S66 S97
			6.1.3	Management	5	
6.2	Policy enforcement	10	6.2.1	Sidecar	6	S2 S78
			6.2.2	Policy Agents	4	S29 S97
						S36 S113
6.3	Data Exchange	53	6.3.1	API	10	S15 S83
			6.3.2	Streaming	14	S24 S84
			6.3.3	Data Channel	15	S29 S90
			6.3.4	Management Channel	2	S64 S98
			6.3.5	Message Format	4	S71 S100
			6.3.6	Tools	8	S75 S101
						S78 S110
						S82 S111
6.4	Distribution of Data	32	6.4.1	Change Data Capture	9	S29 S90
			6.4.2	Benefits	16	S72 S101
			6.4.3	Tools	4	S82 S106
			6.4.4	Auditing	3	S84 S110
6.5	Governance Plane	6				S1 S76
						S35 S90
						S57

6.2 Data Products

As described in the preceding chapters, data products consist of five main components: data (views), metadata describing the product, code to execute any transformations and policies, a set of interfaces to facilitate communication with other data sources and management applications, and lastly the infrastructure to enable a data product **(D1.1)**. Preceding chapters have already elaborated on most of the components; however not so much on the product's interfaces. This section describes the product's interfaces and their purpose.

The product's interfaces enable it to communicate with other entities in the architecture. These entities can be data consumers, other data products, enterprise data sources, management applications, etc. Each product possesses three different interfaces: the output interface, input interface, and management interface **(R1.2)**.

Output Interface (D6.1.1)
The output interface enables consumers to access the contents of a data product. These consumers can be composite data products that use one or more other data products and perform a complex data transformation or consumers who directly use the data (e.g. in a reporting dashboard).

Input Interface (D6.1.2)
The input interface enables data products to receive new data. This is a connector to one of the enterprise data sources for basic data products. For composite data products, the input interface connects to the output interface of the other data product(s).

Management Interface (D6.1.3)
The last interface is used to manage the product throughout its lifecycle. It connects data products to management services offered by the self-service infrastructure. The product owner uses these services to monitor the product or enforce policies (e.g. authorize a consumer to access the data). Moreover, the management interface connects data products to the governance plane, which is used by the federated governance team to monitor the mesh from a global perspective. More on this in section 6.6.

Figure 6.2 depicts what the data product looks like in the reference architecture.

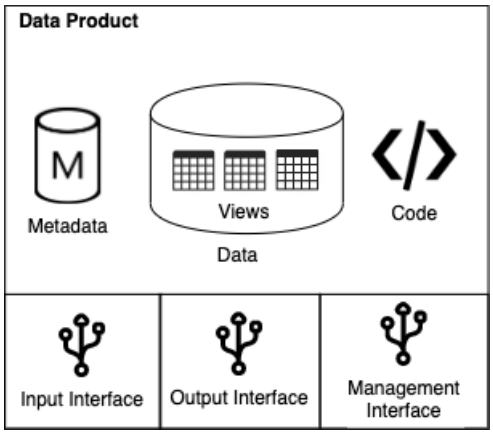

Figure 6.2: Data Products.

6.3 Interface Implementation

Various design patterns can be used to implement the interfaces. Gray literature differentiates between pull-based data exchange and push-based data exchange. The first refers to the consumer making ad hoc data requests (i.e. the consumer pulls the data). The latter refers to the situation when the data provider propagates data to consumers when it becomes available (i.e. the developer pushes data to consumers). Figure 6.3 depicts the two types of data exchange.

Pull based data exchange (D6.3.1)

There are several use cases for pull-based data exchange. First of all, for consumers that do not need updated data when it becomes available, pull-based data exchange offers the ability for ad hoc data requests. Moreover, static data products can only offer pull-based output interfaces as no new data becomes available to be pushed to consumers. Lastly, in case a consumer wants to experiment with some data, it might be more sensible for the consumer to pull the data once as it does not (yet) need regular updates. If the use case is industrialized, the consumer can switch to push-based data exchange to receive the new data when it becomes available. APIs can provide pull-based data exchange.

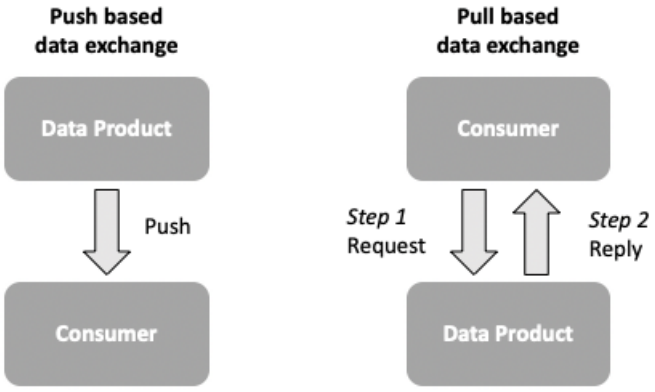

Figure 6.3: Push- and Pull Based Data Exchange.

Several technologies are available to implement APIs, such as SOAP and REST. Data Mesh is vendor and tool agnostic and thus has no specific requirements for the API implementation. Nonetheless, to increase interoperability between data products, all products should use the same API implementation for their interfaces. The federated global governance team decides on one technology that becomes a standard throughout the organization. The interfaces are then made available by the central platform team as part of the self-service infrastructure, so the product developer only needs to tailor them to his data products.

The REST implementation is mentioned most frequently in gray literature **(D6.3.6)**. RESTful APIs are widely used in practice and offer considerable benefits over the older SOAP technology used in service-oriented architectures. For example, REST requires less bandwidth and computing power for its interactions (Mumbaikar, Padiya, et al., 2013). It is also more flexible since RESTful APIs are not bound to a specific message format, contrary to SOAP which requires XML (Halili & Ramadani, 2018).

REST offers different ways to interact with a data product employing, but not limited to, GET, PUT, DELETE and POST commands (Richardson & Ruby, 2008). A data product's input interface can use the PUT and POST commands to receive new data from upstream data providers. An API part of the output interface can use the GET command to have consumers request the data. It is not sensible to implement the DELETE command as only a product developer should be able to delete data from a product.

APIs are described in a machine-readable description language that describes the API's characteristics and operations and reduces the required amount of custom programming. The description language also specifies the protocol used to transport its messages (M. Papazoglou, 2008). There are various description languages for REST APIs, but OpenAPI is used most frequently and mentioned multiple times in gray literature **(D6.3.6)**. It uses YAML or JSON to specify the interface (Cremaschi & Paoli, 2017). There are alternatives on the market that might be more suitable for some organizations, depending on their needs. The interface description is part of the product's metadata to inform consumers how they can bind to the interface.

Push based data exchange (D6.3.2)

Push-based data exchange is used for streaming products that provide data on time intervals (real-time or time-based groupings) or propagates changes made to data residing in a product to all consumers. The push-based method aims to automate data distribution through the architecture.

Push-based data exchange uses message-based communication, which is suitable for distributed environments like a data mesh architecture (Eugster et al., 2003). To be specific, data mesh makes use

of the topic-based publish / subscribe pattern **(R2.2)**. Messages containing new data are published to a topic on the message broker. Each view of a data product has its own topic on a message broker to which consumers can subscribe. The message containing the updated data is pushed to the consumer if subscribed. The publish / subscribe pattern allows for a highly scalable form of asynchronous communication in which data products and consumers can easily be added due to its loose coupling (Eugster et al., 2003).

Handling of messages is handled by the message-oriented middleware (MOM) and is bidirectional, meaning that the same channel can be used for sending and receiving messages. The MOM temporarily stores messages until their expiration time has passed or when all subscribed consumers have acknowledged having received the message (M. Papazoglou, 2008). This allows data products that are temporarily disconnected (e.g. due to maintenance or an error) to receive updated data when it comes back online. The MOM used for data exchange is depicted as the data channel in the reference architecture **(D6.3.3)**. Organizations might choose to implement the data channel as an enterprise service bus (ESB) which has additional features compared to the MOM. An ESB includes features such as logging, transformations, routing, and more (Menge, 2007).

Various message brokers are available 5o offer the technical implementation of the MOM. Kafka is mentioned frequently in literature, but alternatives such as RabbitMQ are available **(D6.3.6)**. Kafka and RabbitMQ have developed solutions that offer a highly scalable and efficient exchange of messages with features such as delivery guarantee (Kreps et al., 2011). Kafka is particularly suitable for capturing and distributing change data feeds, which makes it a good candidate for data mesh (Dobbelaere & Esmaili, 2017).

Messages can have different formats. As described before, there is no one size fits all approach for data mesh regarding the technologies used for the instantiation. XML and JSON are among the most common formats used in practice and gray literature frequently mentions JSON **(D6.3.5)**. However, for high scale data exchange and processing, JSON might be the best option available as it requires fewer resources and is faster in comparison to XML (Nurseitov et al., 2009). Especially for cloud implementations of data mesh, where you pay per use, JSON can prove to be cost-saving. Again, it is important that the federated global governance team sets a standard message format to be used throughout the organization to increase interoperability between data products.

Interfaces: Push and/or Pull

The design patterns described above need to be placed in the context of the three interfaces of a data product. First of all, a product's output interface can use both types of data exchange. The interface can have an API for ad hoc requests and the publish / subscribe pattern to push new data to data consumers automatically. Not all output interfaces need both; it is up to the product developer to decide if it needs one of the patterns or both depending on the data type and consumer needs. For example, if the developer builds a static data product, they decide to only implement an API for pull-based data exchange. The input interface can be either a connector to an enterprise data source, a subscription to another data product on a message broker (push-based data exchange), or the developer can make an API call to the output interface of another data product. The last method stipulates that the input interface has not always a continuous connection to the data provider but can also be an ad hoc connection.

The management interface makes use of push-based data exchange. Data products publish their operational data to the local and global management applications. The product owners use these applications to govern their products. For example, to see if the product's data quality adheres to its SLOs. Moreover, it creates a global view to be used by the federated governance team to manage the mesh as a whole. Moreover, each product is subscribed to the management applications. For example, if an update is made to a global policy, the updated policy is propagated to the management interface of all data products over the management channel.

The self-service infrastructure provides the interfaces. However, the developer has to make sure they integrate with their data products.

6.4 Change Data Capture

To automate the publishing of changes in data, data mesh makes use of the change data capture (CDC) pattern **(D6.4.1)**. The CDC, see fig. 6.4, tracks any changes made to data by having a transaction log that stores all operations performed on the data. New entries to the log are published to the message broker so all consumers can get the latest data changes. The bootstrap service allows new consumers of a data product to get the full set of transactions on the data. The CDC allows for near real-time distribution of data across the architecture. Moreover, the CDC pattern keeps track of an immutable change log that can be used for governing and auditing (Das et al., 2012). The immutable log tracks all changes made to the data over time, offering proper auditing capabilities for regulated environments **(D6.4.4)**.

Built on Kafka, Debezium is an open-source platform that offers change data capture for different data stores such as MySQL **(D6.4.3)**. The pattern also integrates well with the publish / subscribe pattern making it a suitable tool for data mesh.

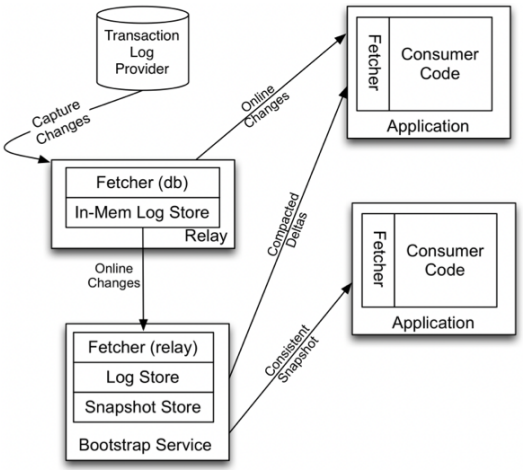

Figure 6.4: Change Data Capture Pattern (Das et al., 2012).

6.5 Policy Enforcement

As described in chapter 4, a federated governance model is one of the main spearheads for the data mesh architecture. Both the product developer and the global federated governance team publish policies applying to specific data products. All data products share a common execution method for said policies, provided by the self-service platform. The sidecar pattern is used to execute policies in a distributed manner **(D6.2.1)**.

The sidecar pattern extends the main product container with a so-called Sidecar container; see fig. 6.5. The Sidecar is responsible for enforcing policies in a decentralized manner, close to the data product itself. Decentralizing policy enforcement allows for scalability compared to a central policy enforcement engine that can become a bottleneck in the architecture. The sidecar can use different policy agents to execute any policy (e.g. access control policies or data quality policies) **(D6.2.2)**. Policy agents are specific patterns that determine how policies are enforced at runtime. For clarification, the sidecar pattern is about where policies are enforced, and the policy agent is about how policies are enforced.

Burns and Oppenheimer (2016) describe some of the benefits of using the sidecar pattern. First, using a separate container for policy enforcement can improve resource allocation. Secondly, having a separate container for policy execution makes it easy to separate the responsibility of maintaining the

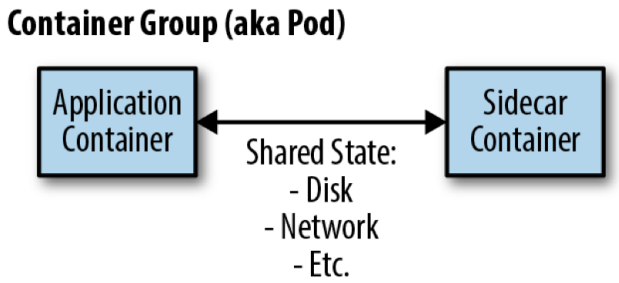

Figure 6.5: Sidecar Pattern (Burns, 2018).

policy agent from maintaining the data product. Thirdly, the sidecar container allows easy reuse for all data products across the mesh. Fourthly, having a separate container makes it easier to upgrade certain components of the policy engine without this affecting the main data product. These last three benefits make it possible for the self-serve infrastructure platform to offer a standardized policy execution for data products, as described in the gray literature. Lastly, having a separate container responsible for policy execution will not make the main data product fail in case of an error in the policy execution.

An example of a potential policy agent is an adaption of Zhou et al. (2002) Policy Enforcement Pattern (PEP) **(R4.2)**. The pattern, presented in fig. 6.6, makes use of a "When Who Can/Cannot Do What" model to specify which subjects can perform what actions on objects in specified conditions (Zhou et al., 2002). Policies are stored in the policy repository. The federated governance can push policies to this repository over the management interface. At the same time, the pattern offers the flexibility for domain owners to add their policies. The policy repository can also store elements of policies that support product developers in building their policies. PEP allows for a change in policies at runtime, meaning policies can be changed without having to redeploy the data product.

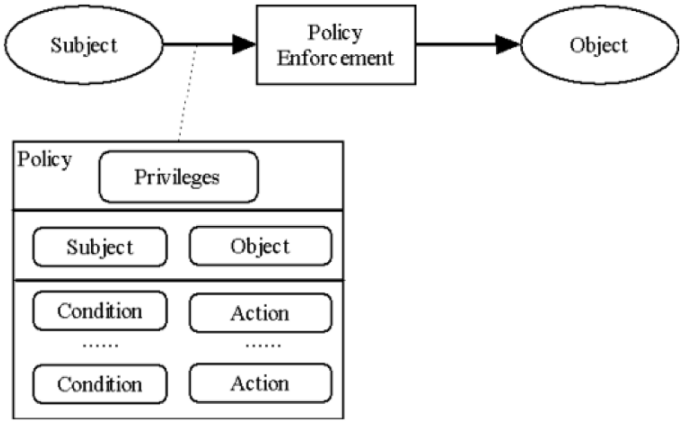

Figure 6.6: Policy Enforcement Pattern (Zhou et al., 2002).

The Open Policy Agent (OPA) is an open-source policy agent frequently mentioned in gray literature **(D6.2.2)**. It facilitates the decoupling of policy decision-making from the data product itself through the sidecar. Moreover, it uses declarative language to define context-aware policies.

6.6 Governance Plane

Lastly, data mesh has a global governance plane that enables the federated governance team to govern the mesh from a global level **(D6.5)**. The plane offers various services that subscribe to the management interfaces of all data products, creating a global view of the architecture. The applications enable monitoring of conformance to policies, usage of data products, etc. Moreover, the governance plane contains the global metadata repository containing standardized definitions, etc. If any changes are made to entries in this repository, the changes are propagated to all data products using their manage-

ment interfaces. The same applies to changes in global policies. Lastly, the governance plane contains the central repository of identities and their roles within the architecture.

The reference architecture depicts several icons in the governance plane that represent overarching services. Although gray literature mentions various specific monitoring use cases, these are combined into these overarching terms to simplify the architecture. For example, cost monitoring is part of the monitoring icon.

Chapter 7

Instantiation on Azure

The designed reference architecture is tool and vendor agnostic. Although some vendors market their tools as data mesh platforms, no vendor has created a full data mesh solution. Most likely, it is not possible to build a full data mesh platform as the exact implementation is heavily context-dependent. There are, however, many commercial and open-source tools available to implement a part of the reference architecture, of which some are already explained in chapter 6. This chapter presents an instantiation for data mesh on Microsoft Azure's cloud platform. It is based on a synthesis of instantiations on Microsoft's Azure, Amazon's Web Services, and Google's cloud platform presented in gray literature.

7.1 Data Mesh on Azure

First, the way Azure structures its resources is explained to help in understanding how the platform can be structured in a data mesh instantiation.

Azure allows users to create different resources such as virtual machines, databases, data lakes, etc. These resources are the cloud computing services used by end-users. Resources are deployed into resource groups, a logical container clustering a set of resources. One resource group can hold many different resources. Resource groups are part of a subscription, which clusters user accounts and resource groups created by those accounts. One subscription contains multiple resource groups. Organizations can use the structuring of resources and users into resource groups and subscriptions to manage their projects and overall workflow. Access controls, rights, and policies can be assigned on a subscription-, resource group-, resource, or user-level making it easier to manage the platform. Figure 7.1 shows how resources can be structured in Azure.

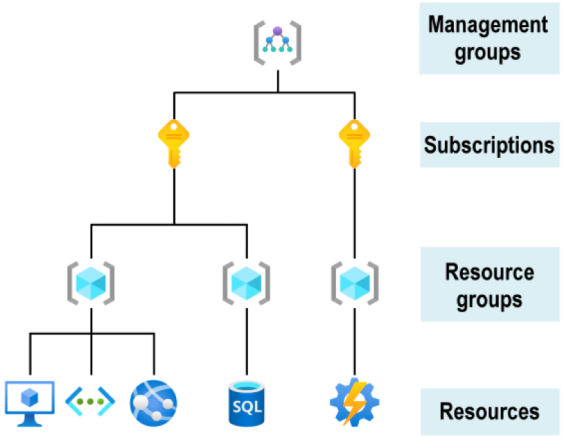

Figure 7.1: Structure of Azure (Microsoft, 2021)

Azure's methods to structure its users and resources can be used when applying the data mesh principles. Resource groups can be used as a bounded context to represent a data product. Each resource group contains all the resources necessary to store and process the data. Since users can be assigned rights for specific resource groups, it also offers a secure separation of concerns. The ability to assign policies on a resource or resource group level can give product owners access to the underlying infrastructure of a data product while offering other users only the ability to read data through, for example, an API. As described above, resource groups can be clustered together in a subscription. Therefore, it is sensible to use Azure subscriptions as a representation of domains. The subscriptions represent business domains, and data products represent bounded contexts in terms of domain-driven design. Figure 7.2 illustrates this concept that there might be subscriptions for the human resources department, marketing department etc.

Figure 7.2: Sample Domain Structuring

Central to the mesh is a subscription that is used for overarching management resources, shown in fig. 7.3. These resources are used throughout the platform and do thus not belong to a specific business domain. Some of these services are related to the core data mesh principles briefly explained below. This chapter does not go into detail on the advanced features offered by these resources as that is beyond the purpose of this illustration.

- **Purview** is Azure's data catalog that easily integrates with all resources offered by Azure. It can catalog data assets and display lineage and transformations performed on the data. It also has support for requesting and granting access to data products and policy enforcement.
- **KeyVault** is used to securely store any API keys, certificates, or secrets and can safely be exchanged between users and resources.
- **Policy** is Azure's central hub for generating and managing policies. These policies can be applied to resources, resource groups, subscriptions, and users. Execution of policies is taken care of by Azure. Policies are stored as code in a JSON format.
- **Security Center** is a central hub for security-related monitoring and management of assets throughout the platform.
- **Blueprints** allows the central platform team to build templates for data products that automatically deploy a set of resources as configured by the platform team. This stimulates uniformity and interoperability, as described in the earlier chapters.

- **DevOps** allows for code repositories, CI/CD pipelines, automated testing, etc., and can be used for product versioning.
- **Active Directory** is Azure's identity management portal.
- **Monitor** is a central hub that can be used to manage resources and activity on the Azure platform. is a central hub that can manage resources and activity on the Azure platform.

Besides their central hub, Azure Policy, Security Center, and Monitor also have a 'local' instance for almost every resource. For example, when navigating to a deployed virtual machine, there is a security center tab where you can manage anything related to security for only that resource. These local instances can apply policies, manage security or monitor specific resources. For data mesh, this means that product developers can use these local instances of the services to build and govern their data products.

Figure 7.3: Management Subscription

Azure offers many resources that can be used in the self-serve infrastructure. Some of the resources provided by Azure concerning the Pipeline Orchestration-, Polyglot Storage, and Compute services from the reference architecture are depicted in fig. 7.4. This list is non-exhaustive since Azure offers many more services but provides a good illustration of the possibilities. As can be seen, Azure offers many resources that can provide polyglot storage methods in data products such as data lakes or SQL databases. Azure also provides different platforms that can be used for machine learning, big data analytics, and more. Lastly, Azure also provides many services that can be used to perform ETL, orchestrate pipelines or execute code. A blueprint might make use of a different combination of these services based on the capabilities it is trying to implement, leading to many other possibilities.

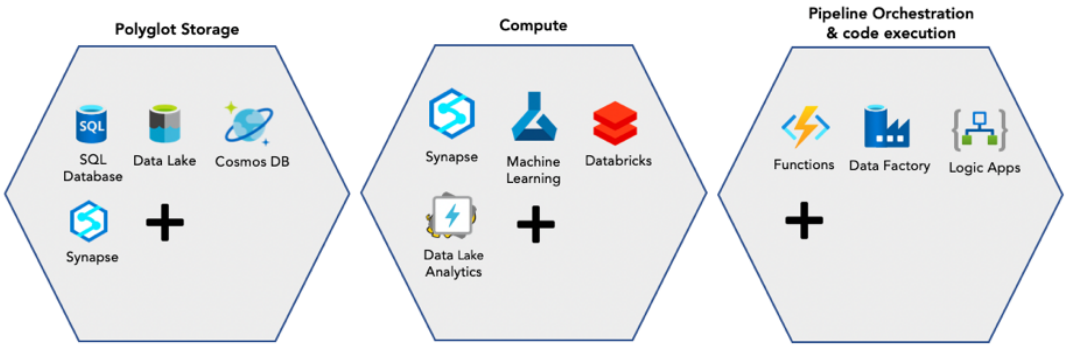

Figure 7.4: Azure Resources for Polyglot Storage, Compute and Pipeline Orchestration

Azure offers different services to implement the data exchange patterns from the reference architecture, of which some are shown in fig. 7.5. For example, Azure Event hub can distribute messages in a publish / subscribe format, and API Management can be used to implement RESTful APIs. These same tools can be used to implement the management channel to exchange management data, but this is mostly already abstracted away by the Azure platform. For example, Azure Monitor makes use of event-based communication under the hood (Ahamed, 2019). The same goes for Azure Policy, in which users can assign policies to resources in a GUI while Azure abstracts away the underlying technical implementation.

All resources in a resource group should be connected through a virtual network. This isolates the applications and data from the public internet and creates a secure environment. Resources within the virtual network can connect, but they are secured from outside connections. Azure offers different services that allow connections to virtual networks or specific resources in a virtual network, some of which are shown in fig. 7.5. There is not one implementation style for data mesh. Depending on the design choices of the organization, the network implementation of services will differ, which will likely also depend on the required level of security.

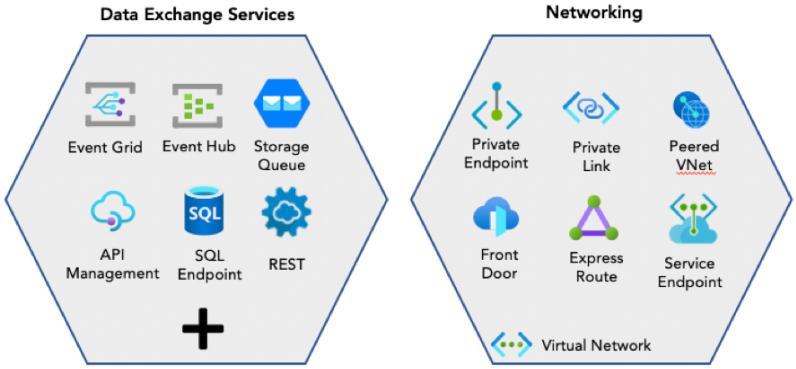

Figure 7.5: Azure Resources for Data Exchange and Networking

There are also different methods to connect the virtual networks; see fig. 7.6. In full VNet peering, all virtual networks are connected. The hub and spoke model connects the spokes through a central hub. Contrary to full VNet peering, the hub and spoke model allows two virtual networks to be connected without redeploying the networks. Having to redeploy data products each time a new connection needs to be established can potentially be undesirable as this might cause problems for downstream consumers depending on a live data feed. Therefore, the hub and spoke model is a good choice for a scalable mesh architecture. It does, however, come with the drawback of potentially higher costs due to extra data transfers for both the incoming and outgoing traffic of the hub.

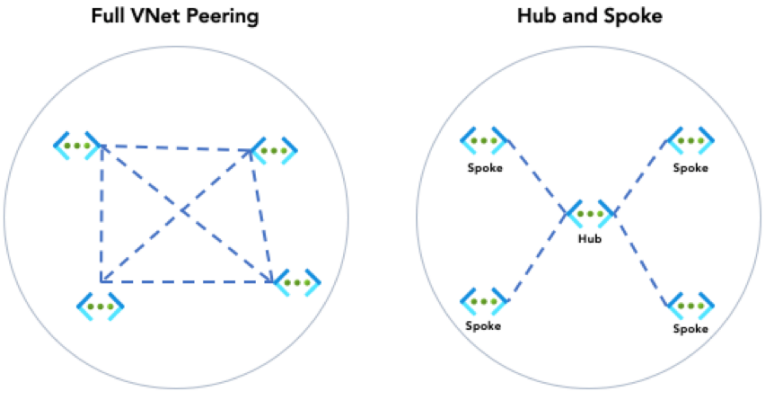

Figure 7.6: Azure Network Models

7.2 Illustrative Implementation

The section above outlines the different resources available on Azure to manage and create data products and govern the mesh. As described, these can be combined in many different configurations depending on an organization's specific needs. To make the instantiation more tangible, this section provides a more specific instantiation as an example; see fig. 7.7.

Central to the mesh is a management subscription with overarching services. A subscription is given to different data domains in which they can create data products through resource groups. Resource groups provide a logical separation of concerns. Data products are enclosed in a virtual network to create an isolated environment from the public internet. Full VNet peering is used to connect the virtual networks in the instantiation below.

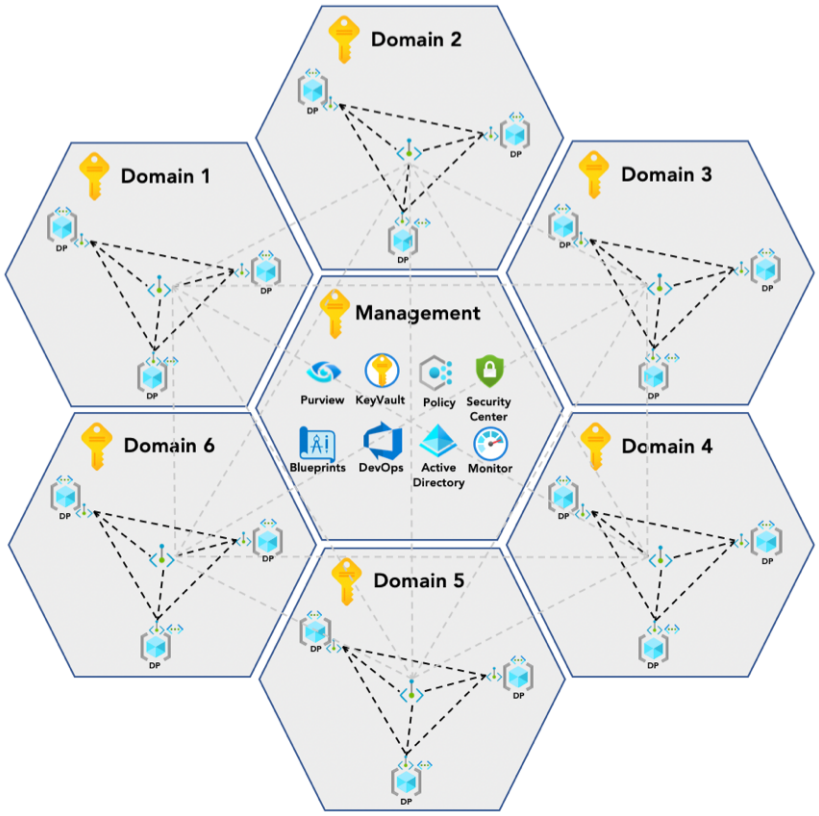

Figure 7.7: Sample Instantiation

Depending on their use case, data products will be implemented differently. The central platform team builds blueprints that deploy a set of resources in a specified configuration. This reduces the required specialization for product owners to develop their products since the basic infrastructure can be deployed from these templates. Figure 7.8 present two sample configurations of candidate data products.

In the first data product above (DP1), raw data from an SQL database and a data lake is integrated and preprocessed by some pipelines orchestrated in Azure Data Factory. A machine learning model is re-trained in Azure Machine Learning using the updated data every day. The predictions made by the model are stored in a data lake besides being propagated to downstream consumers using Azure Event Hub. API Management holds a RESTful API that can be called to make ad hoc requests for predictions from a specific date. Azure Monitor, Security Center, and Policy govern the data product. A private endpoint allows consumers to connect to the virtual network.

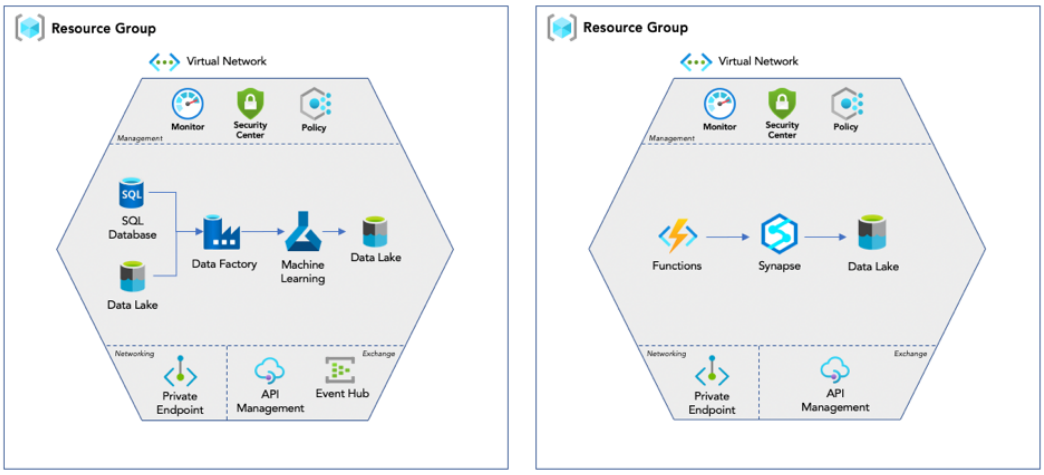

Figure 7.8: Sample Data Product 1 and 2

The second data product (DP2) is a downstream consumer from the predictions of DP1. The product is subscribed to the event stream of DP1. Each day when DP1 publishes the new predictions to its event stream, DP2 receives a message containing the latest predictions. This message triggers a serverless python function in Azure Function that filters out the predictions not necessary in DP2 and combines them with the predictions made in the last seven days. The outcome is used in Azure Synapse to generate meaningful insights. These insights are stored in a data lake with an API to access them. Similar to DP1, Azure Monitor, Security Center, and Policy are used to govern the data product.

7.3 Computational Federated Governance on Azure

The sections above briefly touch upon some of the governing functionalities in Azure. This section provides a more detailed description of how the Azure platform can be used to govern the mesh architecture.

First of all, Azure Purview is used as a central resource for metadata management and product discovery. Data assets, or in the case of data mesh, data products, can be registered in Azure Purview. Metadata can be added for each data asset to make a data product searchable and self-describing. Purview also has lineage detection, sensitivity labels, and policy integration functionality. Lastly, using Azure's Active Directory and role-based access controls, Purview can also provide a workflow to manage access to data products easily. Purview also provides functions for business glossary modeling and assignment of definitions to datasets, features, or specific data values.

Azure Purview can be integrated with Data Factory or Synapse Analytics to implement automated data quality checks and standards. Using Purview's REST API, metadata of data products can be extended to also include product-specific data quality factors. Purview's GUI does not yet provide support for this. Pipelines in Data Factory or Synapse Analytics can query the metadata in Purview to see what quality metrics are relevant for specific data products. Data Factory and Synapse Analytics can then use this to monitor how data products perform with respect to these quality metrics or perform operations on the data to make it adhere to the quality standards.

The federated governance team decides which quality metrics and standards should be verified by these pipelines, while the platform team builds templates for the pipelines. These can be version controlled using Git. Local domains can use these pipelines to implement their automated checks. These pipelines are not solely limited to checking for standards but can be extended to also include a product's SLA or data contract.

In the example of DP1, the Data Factory instance might have a pipeline running that checks the incoming data from the SQL Database and data lake on some quality metrics defined in Purview. If

the data quality does not adhere to a minimum standard, a notification is sent out using Azure Monitor to the product owner. The same concept can be used to read sensitivity labels from Purview to mask sensitive data to adhere to data protection law (e.g. GDPR).

Second to metadata management, Azure offers a wide range of functionalities related to security. Data at rest or in transit can be encrypted, and coarse-grained access controls can be used to manage access to resources and data assets, networks can be separated, etc. These functionalities can be used to create a safe environment from outside intruders and create a secure internal architecture. A combination of Azure Monitor and Azure Application Insights can be used to monitor the mesh architecture on both a local and global level. Among its functionalities is the ability to log, set alerts, and respond to issues (e.g. upscale certain resources).

Lastly, Azure Policy can manage and implement policies and standards to which entities on the platform are to adhere. Entities can be users, resources, sets of resources, or specific data assets. Consistency in the architecture can be accomplished by using Azure Policy to force entities to adhere to certain standards. Policies can be managed centrally by the federated governance team, but product owners can also add their local policies to their resources.

Conclusion

The description above shows how Azure can be used to instantiate a data mesh architecture. The wide range of resources offered by the platform allows for scalability and flexibility. It is important to note that the illustration above is just one of the different methods a mesh architecture can be instantiated on Azure. There is no one right solution, but the core components will most likely remain the same.

Chapter 8

Illustrative Use Case

This chapter describes what was brought into practice in the demonstration stage of the design science research methodology. An illustrative use case was conducted at ASML, a leader in the semiconductor industry. A blueprint to provision infrastructure was developed for a domain to build and provision a data product for a specific use case. Hereby, part of the runtime view of the reference architecture and the instantiation were applied in practice. This chapter describes the use case, what was done, and how it was implemented.

The use case was found through stakeholder interviews within ASML. The interviews were used to better understand ASML's current data landscape and see what, if any, initiatives around data mesh were already ongoing. We got in touch with a domain that experienced difficulties in sharing data. Further discussions identified a specific use case in which a data asset could be served as a product, which was realized in this use case. The subsequent sections provide context for the use case and explain what was done. Section 8.1 provides some context by placing the use case in the broader perspective of ASML. Section 8.2 introduces the specific use case, and section 8.3 explains which components of the reference architecture were implemented on Azure. Then, section 8.4 explains how a blueprint was built to provision the infrastructure for the use case. Lastly, section 8.5 describes how the solution was deployed, after which section 8.5 wraps up with a conclusion for the demonstration.

8.1 Context

ASML is one of the world's leading companies in the semiconductor industry. The company produces lithography machines that produce computer chips. Due to the incredible complexity of these machines, ASML has been able to have a monopoly in the field. In recent years, ASML has been under a lot of pressure from customers to increase its output due to the high demand for chips worldwide. ASML wants to use data to increase the number of machines they can produce every year by, for example, optimizing their supply chain. Moreover, they want to provide data-driven services to their customers that increase the output of their machines, with this fulfilling the needs of their customers.

Multiple factors make it challenging for ASML to achieve these goals. First of all, ASML possesses incredibly advanced technology. The company has been targeted multiple times by hackers to steal intellectual property. To reduce the risks of being hacked, ASML heavily governs its data and has strict security policies. Secondly, ASML uses a centralized team that provisions data to all consumers in the company. Due to the large growth of the company and the increase in demand for data, the team struggles to keep up with all requests. Moreover, due to the separation of knowledge between data providers, consumers, and the centralized provisioning team, the provisioned data is often not of the desired quality and is difficult to govern.

Although ASML wants to increase its manufacturing capacity and output capacity of its machines, the challenges described above inhibit the company from doing so. Teams need to be able to experiment with data to create innovative solutions, but their monolithic data architecture does not allow for this. Moreover, the lack of data ownership does not work well with their strict governance program.

Finding out if certain consumers should be authorized to use specific data costs a lot of time, especially due to the siloes created by the centralized architecture. Moreover, after a user is authorized to use data, there is no clear overview of how this data is used and if it is used in a secure environment. In short, the company experiences challenges in managing its data usage.

Data mesh can be a resolution to ASML. There is clear ownership of data by making domains responsible for provisioning their data as products. Moreover, there will be no dependencies on a centralized team to provision data to consumers, allowing ASML to scale out its data consumption better. The clear ownership of data makes it easier to govern the data properly, and since domains make use of the self-service infrastructure, a minimum security level can be guaranteed.

8.2 Use Case

Analysts from ASML's customer sourcing and supply chain (CSCM) department use predictions for stock destinations of specific materials to reduce the late arrival of materials. These predictions are made by a different department, the sourcing and supply chain (SSC) department, which has a model based on historical data on stock destinations. Besides the CSCM department, several other teams use these predictions throughout ASML. There are, however, several challenges for both the CSCM and SSC departments.

First, the SSC department finds it difficult to keep up with the ad hoc requests to retrain the model using the latest data. Since this is not part of their regular workload, they often have to find time between other tasks to generate new predictions and send them to the requestor over email. The CSCM often has to wait a few days before they receive the latest predictions, sometimes resulting in late decisions and late arrivals of materials. These issues lead to frustration for both teams.

To address these challenges, the SSC department would like to serve their data as a product. This removes the dependency on the SSC department, enabling consumers to request new data as it becomes available. Moreover, it reduces the unexpected ad hoc workload for the SSC department. To put this use case in the broader picture, domain-driven design is applied to decompose ASML into domains and the data product.

ASML is a large organization. For data mesh, domains are too large and thus further decomposed into sub-domains. Both the SSC and CSCM departments are considered sub-domains of the overarching Strategic Sourcing Procurement domain. The data product for stock destination predictions is the bounded context for this use case. Figure 8.1 depicts ASML's decomposition into sub-domains and the bounded context for this use case.

This demonstration aims to build a blueprint that provisions the infrastructure necessary for the SSC department to serve their predictions as a data product. The following section elaborates on what exactly is going to be implemented.

8.3 Blueprint Solution Design

To understand what infrastructure needs to be provisioned, a better understanding of the workflow is needed. The SSC department uses data from three different data sources: a file share, sharepoint, and an on-premise SAP HANA data lake to make the stock destination predictions. A set of python scripts clean and integrate the data sources and store part of the output in tabular format and the other part as unstructured data. The machine learning model reads these datasets to train a model and make predictions.

The blueprint should provision the infrastructure capable of executing all the steps mentioned above (e.g. ingestion, preprocessing, serving the data) and is implemented on ASML's cloud provider: Microsoft's Azure. As described in chapter 7, Azure offers many services that can be used to build a data product. Azure Data Factory (ADF) is used to ingest the data into the cloud environment. It stores the

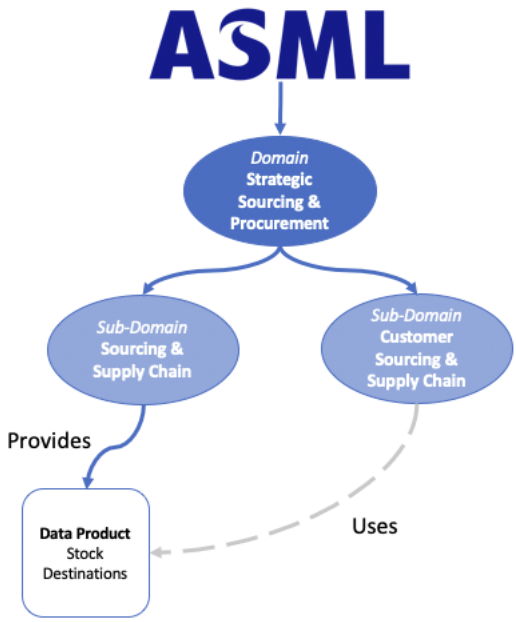

Figure 8.1: Decomposition of ASML into sub-domains

data in a data lake, after which it orchestrates pipelines to execute the python scripts to integrate and preprocess the ingested data sets. Part of the refined, preprocessed data is stored in an SQL database, and part is stored in a data lake. The predictions made using these data sets are stored in a data lake from which they can be served to consumers like the CSCM department. Azure Machine Learning (AML) is used to implement the industrialized machine learning model. Although the AML platform is not part of the blueprint, the blueprint must integrate with the ML platform. The flow of data is shown in fig. 8.2.

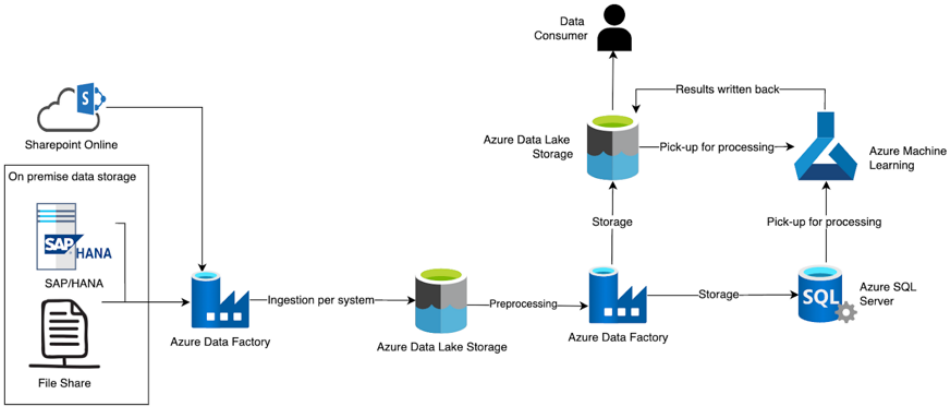

Figure 8.2: Flow of Data in Use Case

There is more to the blueprint than solely providing Azure Data Factory instance, data lake, SQL database, and AML platform. To comply with ASML's security requirements, the complete configuration needs to be placed into virtual networks, which creates a secure environment closed off from the public internet. Moreover, compute resources for ADF need to be provisioned to execute the pipelines. ADF is simply an orchestration tool and does not provide the compute itself. The complete solution architecture for the blueprint, including the integration with the AML platform, is shown in fig. 8.3.

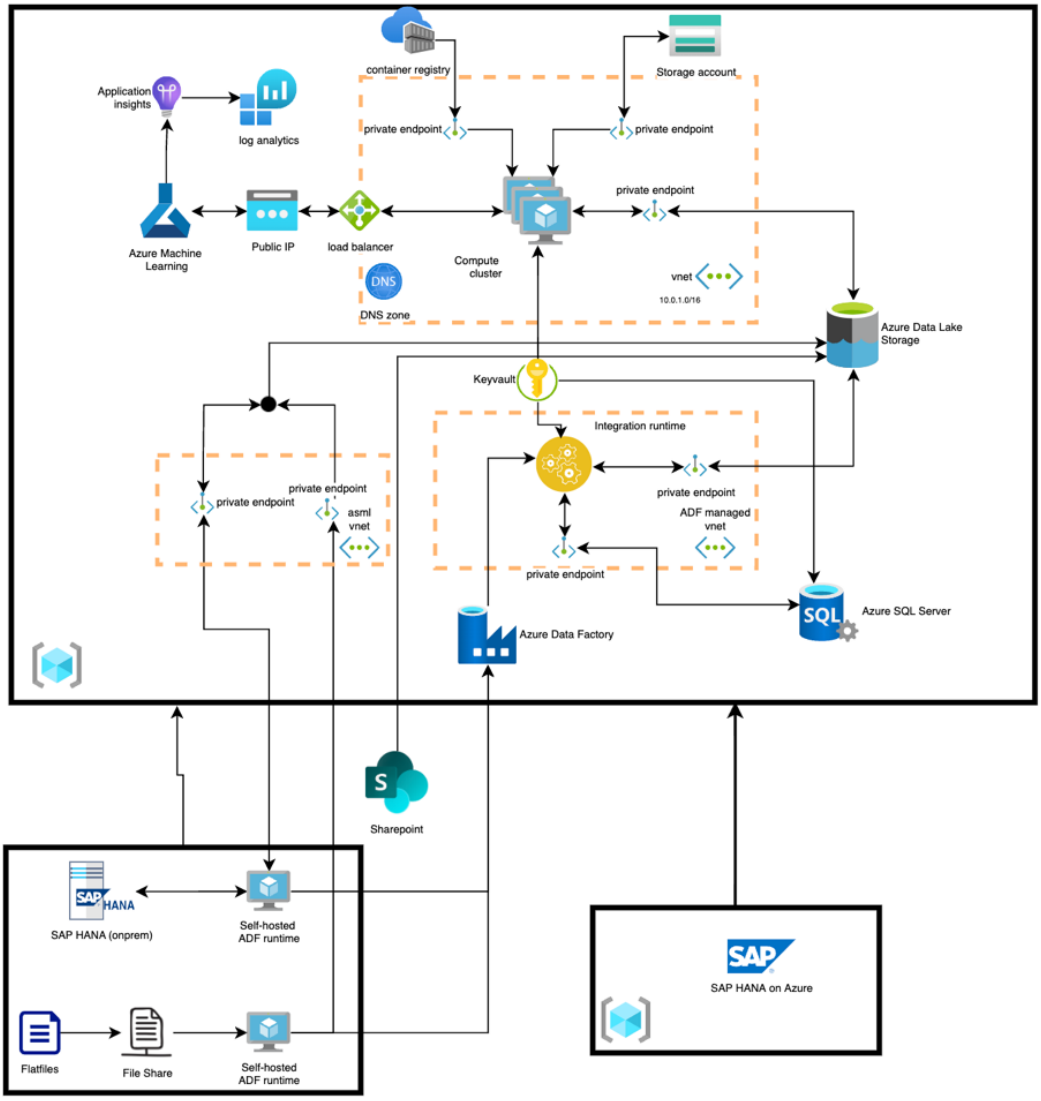

Figure 8.3: Complete Solution Architecture

ASML already has a blueprint for a complete MLOps platform in Azure. Therefore, this illustrative use case focuses on developing a blueprint that deploys the polyglot storage methods, pipeline orchestration, compute, and an API to access the data. All resources are placed into separated virtual networks that create a secure environment. Moreover, these virtual networks are then connected to allow data exchange between them. Figure 8.4 presents the solution architecture for the blueprint developed in this demonstration. A more elaborate explanation of all the resources is given below.

- **Virtual Network:** A VNet is deployed to create a network environment closed off from the public internet. The VNet contains the other resources provisioned by the template.
- **Subnet:** Within the VNet, three subnets are created for the storage account, private endpoint for the SQL Server, and API Management. Subnets contain a subset of the IP addresses of a VNet. Resources in different subnets but part of the same VNet can communicate with each other.
- **DNS Zone:** A DNS zone is used to link the deployed resources to IP addresses in the VNet.
- **Storage Account:** The storage account provides simple blob storage (data lake) that can hold both the raw data and preprocessed data.

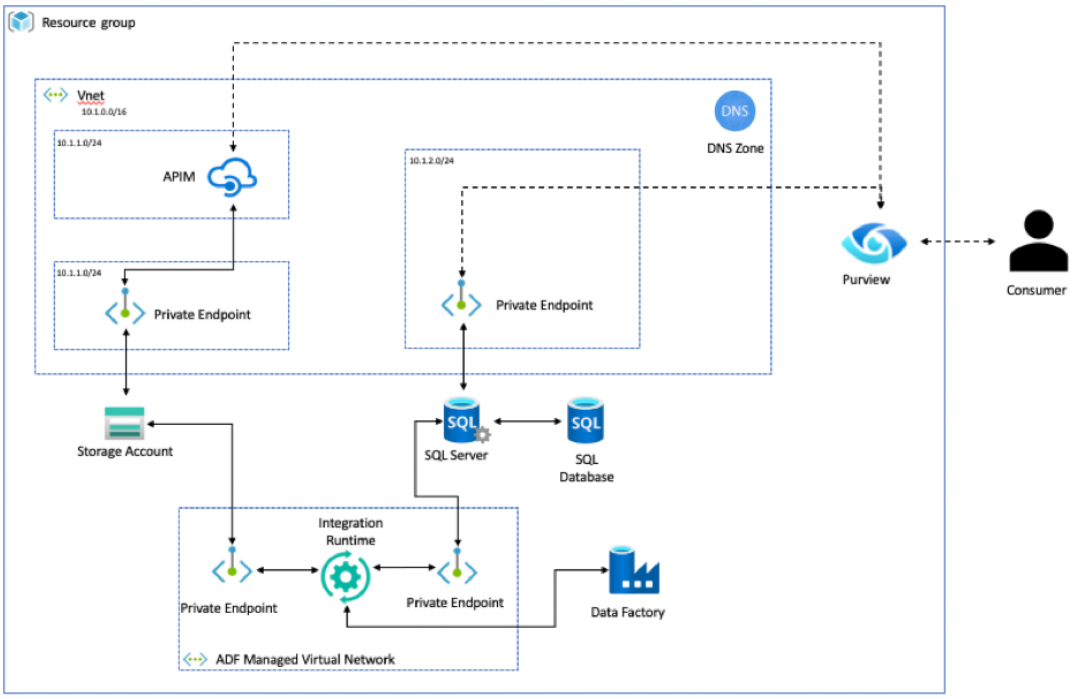

Figure 8.4: Solution Architecture of Developed Blueprint

- **SQL Server:** This is a dedicated virtual machine running Windows Server 2019.
- **SQL Database:** Within the SQL Server, a database is deployed that holds part of the preprocessed data.
- **Private Endpoints:** Private endpoints are used to create a secure connection between resources and their subnets. Resources in the private network can connect to each other while the resources remain protected from the public internet.
  The template also deploys two private endpoints that allow the integration runtime to access the storage account and SQL server so it can store the data.
- **API Management:** A resource used to build and manage APIs, including authorization and logging capabilities.
- **API:** Within the API Management resource, an API is deployed that provides access to the data in the blob storage.
- **API Policy:** A policy used by the API for authentication through Azure's Active Directory (IAM management) is deployed to allow users to make use of the API. This policy checks if the CSCM department is authorized to access the predictions when calling the API.
- **Data Factory:** This resource orchestrates pipelines to ingest the data but also to integrate and preprocess the raw data, which is then stored in the storage account and SQL server.
- **Integration Runtime:** Provides the compute necessary to execute pipelines in Data Factory.

By deploying the blueprint described above and the MLOps blueprint, the SSC department possesses all resources it needs to deploy its data product.

Blueprint in relation to the reference architecture

This subsection clarifies which components of the runtime view of the reference architecture are implemented in this demonstrative use case. Figure 8.5 presents the original reference architecture at runtime with the components implemented in this use case marked in green. The points below briefly describe the relation between these marked components and the resources explained above.

- **Input Interface**
  The product's input interface is implemented through connectors offered by Azure Data Factory. These connect to the file share, sharepoint, and on premise SAP HANA data lake to retrieve the source data.
- **Networking**
  All resources are incorporated into virtual networks. Related services such as subnets, DNS Zones, and Private Endpoints are also deployed to route traffic.
- **Pipeline Orchestration**
  Azure Data Factory is deployed to orchestrate the python scripts used for preprocessing and integrating the source data.
- **Polyglot Storage**
  The blueprint provisions two types of storage to handle the polyglot data. It provisions a data lake and SQL Server, and database.
- **Compute**
  The integration runtime is deployed to provide compute for Azure Data Factory to run the pipelines.
- **IAM**
  An XML policy is implemented that authorizes requests made to the product's output interface by validating the authorization and identity with Azure's centralized identity management service (Active Directory)
- **Output Interface**
  An API is implemented to realize pull-based data exchange. Consumers can request the API to retrieve the latest available predictions.

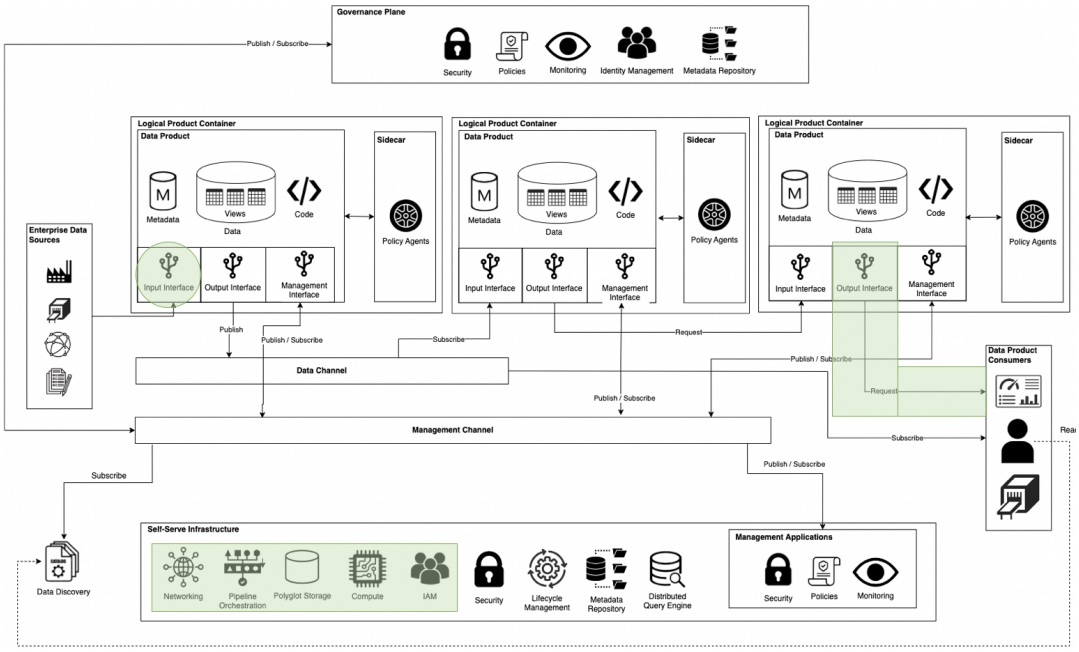

Figure 8.5: Components of the Runtime Reference Architecture Implemented in this Use Case

8.4 Building the Blueprint

The blueprint is developed in Terraform, an Infrastructure as Code (IaC) software tool from HashiCorp. Alternatives to Terraform are available, such as Ansible and Chef. Cloud providers also offer IaC software, such as Azure's Resource Manager. Terraform was chosen because it is already part of ASML's current toolset and its support for multi-cloud implementations.

Terraform is a declarative programming language, meaning that the code specifies the configuration of resources to be deployed in Azure (Terraform, 2022). Blocks are used to define the configuration of resources (e.g. the SQL server). Resources that are used together can be clustered together into a module. Modules allow for standardization, easy reuse of developed code, and reduce duplication of effort. For example, suppose developers always have to deploy a certain resource into a virtual network with a secure configuration. In that case, a module can be developed that deploys all these resources into the right configuration.

The code for the blueprint is structured into different files for ease of understanding. Terraform will read the code as one upon execution. The *terraform.tf* file tells Terraform the configuration for how to manage the infrastructure. It states which version of Terraform and Azure to use and some basic settings on how to deploy certain resources. fig. 8.6 shows a snippet of the configuration code.

```
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "=2.98.0"
    }
  }

  backend "azurerm" {
  }

  required_version = "= 1.1.7"
}
```

Figure 8.6: Snippet of the Terraform Configuration File

The *main.tf* file contains the configuration of the resources to be deployed by the template. It consists of a combination of modules and resources. Since the blueprint is intended for repeated use, some arguments are defined as variables. For example, the module below deploys a virtual network. The variable rg_name declares which resource group the virtual network should be deployed. The variable tells Terraform that this is a variable that needs to be declared upon deployment. If another domain than the SSC department wants to deploy the same infrastructure using the template, they do not have to go through the code to find all arguments that need to be changed for their deployment. Instead, they are declared as variables that can be found in the *variables.tf* file. This makes blueprints suitable for repeated use.

The code snippet presented in fig. 8.7 is used to deploy an API in the API Management instance. It first states in which API Management instance the API should be deployed. It does this by referring to the resource ID of the API Management resource declared elsewhere in the code. It then imports the API description that has to be published from an external file. The code snippet presented in fig. 8.8 makes use of a module to deploy a virtual network and subnets.

The API was implemented using the OpenAPI standard. Since consumers of the data product shouldn't be able to change or delete the data, only a GET command was implemented, as shown in fig. 8.9. Authentication was handled using Azure's Active Directory, for which a policy written in XML is included, see fig. 8.10.

Although only some of the resources are explained in this section, the complete blueprint can provision all resources described in section 8.3.

```
resource "azurerm_api_management_api" "dp1_api" {
  name                = "example-api"
  resource_group_name = var.rg_name
  api_management_name = module.api_management.name
  revision            = "1"
  display_name        = "Example API"
  path                = "example"
  protocols           = ["https"]

  import {
    content_format = "openapi+json"
    content_value  = file("dp1_api.json")
  }

  depends_on = [
    module.api_management
  ]
}
```

Figure 8.7: Code Snippet for API Management

```
module "data_product_1_vnet" {
  source          = "../../modules/vnet"
  rg_name         = var.rg_name
  location        = var.location
  name            = "vnet"
  prefix          = "data-product-1"
  address_space   = ["10.1.0.0/16"]
  subnet_prefixes = ["10.1.0.0/24", "10.1.1.0/24", "10.1.2.0/24"]
  subnet_names    = ["api_management", "dp1", "dp2"]

  subnet_enforce_private_link_endpoint_network_policies = {
    dp1 = true
    dp2 = true
  }
}
```

Figure 8.8: Code Snippet for VNet and Subnets

8.5 Solution Deployment

To deploy the resources, Azure CLI has to be installed in order to log in to Azure from your local terminal. Moreover, Terraform has to be installed to interpret the code. Then by calling the *plan* command, Terraform makes a plan for the resources that need to be deployed. By calling the *apply* command, Terraform will execute the deployment. The current configuration of the deployed resources is stored in a separate *state file* on your local machine. In case changes are made to the blueprint or new resources are added to the blueprint, Terraform makes sure only to update the resources for which the specified configuration has changed. It does this by comparing the configuration specified in the blueprint with the current configuration in the state file.

Once Terraform has finished deploying all resources specified in both the MLOps blueprint and the blueprint developed in this use case, the product developer from the SSC department can use the Azure portal to start building the data product. The team can use ADF to build pipelines that ingest and preprocess the data. The refined data can then be stored in both the data lake and SQL database. The model can be implemented in AML, which can store its predictions back in the data lake. The CSCM department can then call the API to access the predictions.

```
{
    "openapi": "3.0.0",
    "info": {
        "title": "get_files",
        "description": "",
        "version": "1.0"
    },
    "servers": [
        {
            "url": "https://innomesh-apim-2.azure-api.net"
        }
    ],
    "paths": {
        "/": {
            "get": {
                "summary": "Get File Overview",
                "operationId": "get-file-overview",
                "responses": {
                    "200": {
                        "description": null
                    }
                }
            }
        }
    },
    "components": {
        "securitySchemes": {
            "apiKeyHeader": {
                "type": "apiKey",
                "name": "Ocp-Apim-Subscription-Key",
                "in": "header"
            },
            "apiKeyQuery": {
                "type": "apiKey",
                "name": "subscription-key",
                "in": "query"
            }
        }
    },
    "security": [
        {
            "apiKeyHeader": []
        },
        {
            "apiKeyQuery": []
        }
    ]
}
```


Figure 8.9: Code Snippet for API

```
<policies>
    <inbound>
        <set-variable name="ContainerName" value="@(context.Request.Headers.GetValueOrDefault("Container"))" />
        <set-variable name="BlobName" value="@(context.Request.Headers.GetValueOrDefault("Blob"))" />
        <base />
        <set-header name="Blob" exists-action="delete" />
        <set-header name="Container" exists-action="delete" />
        <set-header name="Ocp-Apim-Subscription-Key" exists-action="delete" />
        <set-header name="Sec-Fetch-Site" exists-action="delete" />
        <set-header name="Sec-Fetch-Mode" exists-action="delete" />
        <set-header name="Sec-Fetch-Dest" exists-action="delete" />
        <set-header name="Accept" exists-action="delete" />
        <set-header name="Accept-Encoding" exists-action="delete" />
        <set-header name="Referer" exists-action="delete" />
        <set-header name="X-Forwarded-For" exists-action="delete" />
        <set-header name="x-ms-version" exists-action="override">
            <value>@{string version = "2017-11-09"; return version;}</value>
        </set-header>
        <set-backend-service base-url="@{
string containerName = context.Variables.GetValueOrDefault<string>("ContainerName");
string blobName = context.Variables.GetValueOrDefault<string>("BlobName");
return String.Format("https://agg3st2yi.blob.core.windows.net/{0}/{1}", containerName, blobName);
}" />
        <authentication-managed-identity resource="https://storage.azure.com/" />
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```

Figure 8.10: Code Snippet for API Access Policy

Conclusion

The illustrative use case demonstrated part of the reference architecture at runtime and the instantiation on Azure. Building the self-service platform to enable a domain to provision a data product showcased both how to address this problem and the challenges that might arise. Moreover, it also shows the complexity of building the platform. This was a small use case and only a limited set of components from the runtime architecture were implemented, yet it was quite challenging. However, it does show the challenges that organizations face day to day and how they can be addressed by data mesh.

It is good to recognize that the demonstration was limited to a part of the runtime architecture and the instantiation and that it thus does not demonstrate all of the designed artifacts. However, it does provide a good starting point for future work to further demonstrate data mesh in practice.

Chapter 9

Validation

This chapter describes how the validation phase of the design science research methodology was executed. Due to time constraints, validation was scoped to the two views of the reference architecture. Although the comprehensive definition of data mesh and the instantiation on Azure were also shared with interview participants, the interview questions were solely about the reference architecture. Five interviews were conducted with experts from industry. These were transcribed and systematically analyzed to extract their feedback. Based on the feedback, improvements were made to the design artifacts.

Section 9.1 describes how the semi-structured interviews were prepared, and section 9.2 describes how the conducted interviews were processed. Then, section 9.3 lists the specific points of improvements that were reiterated in the design artifacts based on the feedback provided by the participants, and section 9.4 provides a conclusion for the validation of the designed artifacts.

9.1 Interview Design

The designed artifacts are validated through semi-structured interviews with relevant stakeholders from industry. Semi-structured interviews follow a set of predefined questions but offer the flexibility for follow-up questions depending on the responses of the interviewee (Longhurst, 2003). There is a broad consensus that the way a qualitative study is set up greatly impacts the quality and reliability of the results (Kitto et al., 2008). Therefore, Kallio et al. (2016) framework for developing a qualitative semi-structured interview is used to prepare and structure the validation interviews. Their guide consists of five consecutive phases, which are explained below:

1 **Identify prerequisites for using semi-structured interviews**
This phase determines the appropriateness of using a semi-structured interview for the qualitative study. There are several research goals for which semi-structured interviews are effective, among which is the ability to explore participant thoughts and beliefs around a certain topic (DeJonckheere & Vaughn, 2019). Compared to structured interviews, semi-structured interviews offer more flexibility in exploring answers provided by the study's participants. Being able to ask follow-up questions when validating the reference architecture is valuable as this can lead to more insights into the participant's thought process. Semi-structured interviews allow this, making it the most suitable style for this study.

2 **Retrieve and use previous knowledge**
The second phase is all about gaining knowledge about the topic of the interviews. Since the interviews will be used to validate the design artifacts, no further preparation on that side is needed. However, it is important to consider the background of participants to tailor the questions to their expertise.

There are several relevant stakeholders when implementing a data mesh-based architecture, such as data consumers, producers, members of the governance team, etc. Table 9.1 provides an overview

Subject	Role	Experience
A	Data Architect	Worked on data mesh implementation at a large bank and large manufacturing company.
B	Senior Architect	Worked on data mesh implementation at a telecommunications company.
C	Data Product Owner	Product Owner of a team of data scientists in a large manufacturing company.
D	Data Governance Architect	Governance Architect in a large manufacturing company.
E	Senior Cloud Solution Architect	Senior architect at a large public cloud provider working on data mesh implementations.

Table 9.1: Interview Participants

of the included participants and their experience with data mesh is given.

3 **Formulate the preliminary structured interview guide**
This stage drafts an initial set of interview questions while considering the participant's background. Santos et al. (2013) methodology for designing a checklist for the evaluation of reference architectures for embedded systems was used as guidance. First, the stakeholders and their roles are taken into consideration. A different set of questions is relevant for each stakeholder, seeing their difference in expertise and relation to the architecture.

Secondly, the questions are structured into different stages, representing the different artifacts produced by this thesis. The first stage contains questions centered around design-time view of the architecture. These contain mostly questions about the different roles and their responsibilities. Stage two includes questions o the reference architecture at runtime. Depending on the interviewee, each stage holds a different set of questions.

Interview questions are centered around the Systems and software Quality Requirements and Evaluation (SQuaRE) model, specifically their quality-in-use model. Their model is composed of characteristics and sub-characteristics that can be used to evaluate computer systems and software in use. Below an overview is given of the sub-characteristics on which the interview questions are based (*Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*, 2011):

4 **Test the guide**
Internal testing was used to determine if the preliminary set of interview questions was clearly formulated and if they covered the purpose of the interview. In internal testing, the interview questions are evaluated by other participants in the research team, in this case, the supervisor of the study (Kallio et al., 2016).

5 **Present the guide**
The final phase presents the complete and revised set of interview questions. The interview questions for this thesis are presented in section 9.1.

Table 9.2: Interview Questions

Stage	Respondent	Question
1	Architect	1. Does the architecture help in understanding the different actors and their responsibilities? If so, how?
		2. Do the layers help in understanding the logical separation between simple products, composite products and managed products and their dependency on the self-service platform?
		3. Does the architecture provide a complete overview of the entities, actors and responsibilities?
	Data Product Owner	1. Are the responsibilities of the consumer-, aggregator- and developer role and their responsibilities clear and complete?
		2. Are the consumer-, aggregator- and developer role relevant to your role?
		3. Is it clear what support is available for you to create managed data products?
		4. What benefits do you see to using the data mesh-based architecture compared to a centralized architecture? Which disadvantages, if any, do you see?
		Clarification Questions
		How would you create a composite data product?
		Can you explain how you would use the self-service infrastructure to create and manage your data products?
	Governance Architect	1. Does the architecture help in understanding the different actors and their responsibilities with regards to governance in a data mesh architecture?
		2. Does the architecture clearly explain the difference between global and local governance?
		3. Are the governance goals clearly explained?
		4. Is it clear what support is available to govern the mesh architecture and is the support complete?
		5. Would you be willing to adopt a federated governance model?
2	Architect	1. Do the elements displayed in a data product provide a complete representation of the relevant components?
		2. Does the reference architecture clearly explain the different interfaces and their use cases and is it complete?
		3. Would you use this reference architecture to instantiate a decentralized data mesh architecture?
		4. Is it clear how the self-service infrastructure is related to the lifecycle of a data product?
		5. Is it clear how policies are propagated and executed in a federated manner?
	Data Product Owner	1. Does the reference architecture make it clear how you can build a data product and which components need to be included?
		2. Does the reference architecture clearly describe which components, as a product owner, are your responsibility to implement and which not?
		3. Is it clear how you can manage data products throughout their lifecycle?

Table 9.2: (*continued from previous page*)

Stage	Respondent	Question
		Clarification questions Can you explain how you would manage a data product once deployed?
	Governance Architect	1. From a governance perspective, does the reference architecture give you sufficient control? 2. Are there any redundant functionalities? 3. Is it clear how the architecture enables a federated governance model? 4. Is it clear how the architecture enables self-describing data products and easy data discovery?

9.2 Data Coding & Analysis

Each subject participated in a one-hour online interview. The design artifacts, including their descriptions, were shared with each participant one week before the interview to give them sufficient time to go through the material. The first fifteen minutes of the interview were used to answer any questions from the participant about the pre-shared material. Afterward, each participant was asked the interview questions related to their background.

Interviews were recorded and transcribed to allow inductive coding. Inductive coding is suitable to condense raw text data into a summarized format, which enabled us to easily extract the feedback from participants into actionable improvement points (D. R. Thomas, 2003). Anonymized versions of the transcriptions can be found in Appendix A.

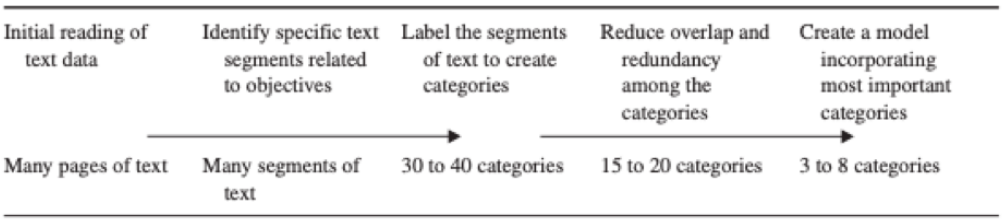

Figure 9.1: D. R. Thomas (2006) Coding Process for Inductive Analysis

The coding was executed in line with the steps mentioned by D. R. Thomas (2006), presented in fig. 9.1. After initially coding the interviews, a broad set of text segments was concentrated into a set of concrete responses for each question. Table 9.3 presents the summarized feedback from architects, section 9.2 presents the summarized feedback from data scientists and table 9.5 presents the summarized feedback from governance architects. In case anything interesting was mentioned, this was included as a quote.

Table 9.3: Summarized Answers from Architects

	Architect	
	Question	**Response**
Design-time View	Q1.1	The general concepts are there. The architecture clearly explains the relevant roles but lacks to explain this in the broader context of domains. Moreover, the architecture does not explain how global and local governance is combined into a model where local governance adheres to global governance. Lastly, the figure is quite complex due to the large number of different arrow types which inhibits easy interpretation of the results. *Quote Participant C: "But how do you just define local policies that adhere towards the more generic policies?"*
	Q1.2	The triangular shape implies that top layers are more important than bottom layers and that there is centralized control. It seems as a misalignment between the interpretation and intention of the figure. It should also be clear that simple data products and composite data products are managed and governed in the same way. The separation into different layers makes it seem that this isn't the case. Some of the participants indicate that this distinction does not make a lot of sense to them since they are both data products in the end. *Quote Participant E: "... it seems like there is some directionality and some hierarchy which might not be accurate... "*
	Q1.3	The main things are included in the diagram. What can be explained better is the complex process of deciding what becomes part of the self-serve infrastructure, what the timeline is of the platform and how to balance the degree of autonomy for domains. Also, how do you manage the lifecycle of infrastructure developed by the domains themselves that is not (yet) part of the self-serve infrastructure? This is a complex discussion between the governance team and the domains. Lastly, something that might be worth mentioning is a decentralized interorganizational architecture for which data mesh can be used. *Quote Participant C: "... there's always a battle and that's because of time and priorities and that's something you need to manage over time where some of the components eventually will end up in domains themselves ... "*
	Q2.1	Yes the architecture includes all components and provides a good explanation for the logical dependencies. However, something to consider might be the tighter integration of operational and analytical use cases. The connection of a data product to operational systems is not described well. *Quote Participant C: "Data is becoming more and more part of this operational processes ..."*
	Q2.2	The interfaces are clear and complete but should be standardized and provided by the self-serve infrastructure in a governed mesh topology. In less governed topologies, product developers have more freedom to build their own interfaces. Since less governed topologies provide more freedom to the domains, the reference architecture at runtime might look different in total.

Table 9.3: (*continued from previous page*)

Architect		
	Question	**Response**
Reference Architecture at Runtime		Additionally, the description can perhaps better highlight the difference between batch and stream in combination with the push and pull interfaces. *Quote Participant E: "You need to have push or pull and then you have batch versus events and any combination of that. So you basically have a matrix of different output interfaces."*
	Q2.3	The participants would use the general concepts from this reference architecture for their own implementation. However, the exact implementation is heavily context dependent. There is not one size fits all but the general concepts are good. Also, the different topologies for the data mesh architecture will result in a different architecture at runtime. This is not clearly explained in the current architecture. Moreover, it is important to highlight that there can be multiple implementations of the data channel within a single organization, depending on the needs of the domains. An interesting remark: domains can be defined from their business functionality (i.e. domain driven design) but it can also be interesting to look at the platform capabilities required by a domain and to base the boundary on this. Too many platform capabilities would suggest the data domain is too large.
	Q2.4	From a high level the architecture makes it clear that product life cycle is managed through the self-service infrastructure.
	Q2.5	Yes in general it is clear but the policy management application should be part of a governance plane and not the self-service infrastructure. Moreover, the complexity of declarative policy languages including examples of those should be added. Lastly, combining local and global policies is currently not explained. *Quote Participant C: "How do we manage our policies across all those big holders and it becomes much more, you know complex if the priorities are also involved."*

Table 9.4: Summarized Answers from Data Scientist / Product Owner

Product Owner / Data Scientist		
	Question	**Response**
	Q1.1	The roles were explained clearly and contain all responsibilities. Having clearly defined roles makes it easy to understand each other's responsibilities.
	Q1.2	All roles are deemed relevant. The participant identifies himself most with the product aggregator.
	Q1.3	Most aspects are clear, except for the tools available with regards to product management. It is assumed that these are made available through the self-serve infrastructure.

Table 9.4: (*continued from previous page*)

Product Owner / Data Scientist		
	Question	**Response**
Design-time View	Q1.4	Benefits: Separation of responsibility will lead to optimization. Experts close to the data will make that data available. Drawbacks: The architecture is complex and also complex to manage. Generating trust in the organization (i.e. that consumers trust the developer) has a lot in place to be realized. A strong vision is required in order to motivate all stakeholders and to implement the architecture. *Quotes Participant B: "How is the consumer going to trust the aggregated product developer? There needs to be a lot in place to make this work.* *"Basically, a decentralized approach to doing analytics and distribution of data and interfaces means complication, means expensive means having a very strong vision in order to make it happen."*
Reference Architecture at Runtime	Q2.1	It was unclear what the sidecar was used for and whether or not this was among the responsibilities of the product owner to implement.
	Q2.2	The participant clearly explained what his responsibilities as an aggregator are; from receiving, to transforming to serving the data.
	Q2.3	The participant explains that he remains responsible for monitoring the lifecycle, but the technical implementation of managing the lifecycle (e.g. decommissioning a data product) is provided by the self-serve infrastructure.

Table 9.5: Summarized Answers from Governance Architect

Governance Architect		
	Question	**Response**
	Q1.1	The general high-level picture with regards to a federated governance model is depicted well in the figure. However, the participant wonders if it is desirable to organize all data exchange in a decentralized manner. Especially for complex use cases that aggregate a lot of data from many different domains and which is consumed often and thus has a large dependency (e.g. financial reporting). *Quote Participant D: "... you have a lot of dependencies towards other sectors within the organization, I think it would make more sense to organize this information provisioning in a more centralized management manner ..."*

Table 9.5: (*continued from previous page*)

Governance Architect		
	Question	**Response**
Design-time View Architecture	Q1.2	The separation of local and global governance differs for each organization. Highly regulated companies (e.g. banks) will most likely govern more things from the global perspective. This could be highlighted better in the description of the architecture.
	Q1.3	In general it is good. However, the global governance goal are broader than solely monitoring. They include setting boundaries, creating guidelines and the guiding principles and ensuring that certain non-functional requirements from a governance perspective are met. *Quote Participant D: "I do like the term monitoring here as a as a verb, because that would imply that also the Federated governance team would have a job in actually monitoring the extent to which the rest of the organization adheres to the governance goals"*
	Q1.4	Expert guidance from architects to all stakeholders should be included in the design-time view. Furthermore, certain decision-making guidance is lacking, see the example below. *Quote Participant D: "... maybe there is already an existing data set and it's 80% compatible with the current state. OK, what are we then going to do? Are we going to create a completely new data set that's for the preferred approach? Or are we going to enhance the existing data set which is 80% compatible to become 100% compatible?"*
	Q1.5	The governance architect is already trying to implement a federated governance model in his organization. So yes, he would definitely implement the model. With regards to incentives, the participant notices that the power / influence that comes with owning a data asset is a significant motivating factor for others to want to become part of the federated governance model. *Quote Participant D: "Once people start to understand that ownership of data gives you a certain mandate within the organization, you will notice that from that moment on you will have plenty of volunteers to become the owner of certain data. However, this is an incentive that you wouldn't really, use in your marketing strategy"*
	Q2.1	In general the reference architecture provides a complete overview of all the components. However, an overarching governance plane is lacking that offers a number of services such as policies, IAM management, monitoring for the federated governance team, etc. Moreover, besides enterprise data sources, companies most likely also receive data from external partners. This interorganizational mesh approach is currently not depicted in the diagram.

Table 9.5: (*continued from previous page*)

Governance Architect		
	Question	**Response**
		Quote Participant D: "So we're actually also consuming a lot of data from our products in the field, right? But also, data from external partners, vendors, suppliers. I believe data mesh can be a solution for this"
	Q2.2	There is no redundant functionality in the architecture. However, at first glance the metadata repository in the self-serve infrastructure seems redundant with the metadata repository in the data products. The complementary role of the two can be explained better.
Reference Architecture at Runtime	Q2.3	Related to the question above, the architecture does not explain how facets such as company wide definitions from the central metadata repository are actually incorporated into individual data products. Also, the capabilities used in a centralized level and those on a local level can be highlighted better. *Quote Participant D: "Orchestration between the decentralized metadata repository and the centralized metadata repository can be added"*
	Q2.4	Yes it does but the realization of including companywide terms and definitions can be explained better. It is clear that the components are there but from a governance perspective, how are you going to ensure it is actually implemented by the domains?

9.3 Points of Improvement

Below an overview is given of specific improvements made to the artifacts based on the feedback of interview participants. Appendix A and appendix A provide the original versions of the artifacts into which these points were reiterated. The final versions of the artifacts are presented in the chapters above. The points of improvements are structured into generic points applicable to the definition of data mesh, improvements for the reference architecture's design-time view, and points of improvements for the runtime view.

General Improvements

- The illustrative use case was placed in the broader context of data mesh. Initially, it had a narrow focus, only on the self-service infrastructure. The use case failed to address how this relates to the bigger picture: how does it solve the problems ASML faces day to day? It was also restructured to address all of the data mesh principles.
- More emphasis was placed on the different topologies for data mesh by clearly stating for which topology the results presented in the thesis are.
- The importance of having incentive models was emphasized more.

Design-time View

- The figure was made more straightforward to ease comprehension and interpretation of the figure. For example, different colors were used to distinguish between the arrows instead of arrowheads.
- The explanation of the triangular shape was expanded, clearly stating that it does not imply importance.

- The initial figure used separate layers for basic and composite data products. Interview participants found this confusing, so they were included in the same layer, using a vertical line to indicate the separation.
- The explanation of the difference between basic and composite data products was improved.
- Interfaces were added as a tool offered by the self-service infrastructure platform.

Runtime View

- A separate governance plane was added to depict the tools used by the federated governance team to monitor the mesh from a global perspective. Initially, these were part of the self-service infrastructure platform, but interview participants found this unclear. Also, an explanation was added to highlight the difference in governance tools for the product owner and tools for the federated governance team.
- A line was added indicating data consumers can read the data from the data catalog.
- The explanation of the policy enforcement was expanded to include the Open Policy Agent. Also, the challenge of combining multiple policies at runtime was added.
- The explanation of pull- and push-based data exchange was improved.

9.4 Conclusion of Validation

In general, the feedback provided by the interview participants was very positive. They could see the utility of the design-time view of the reference architecture as it makes the separation of responsibilities and roles more straightforward. Although the participants indicated that the runtime view was a simplification of reality, they could see that the main components and dependencies were there and that it represents a good template for data mesh. Moreover, participants indicated a willingness to adopt the presented architecture.

Most of the improvements made based on the interviews were either to place more emphasis on something or to improve the concept, so it was easier to comprehend. No major components were missing.

The interviews provided valuable feedback from experts, with this increasing the validity of the results and the likelihood of adoption by industry. However, the validation can be subject to bias from the participants and remains theoretical as it does not implement the architectures. The validation is a good first step but should be elaborated on, perhaps in future research.

Chapter 10

Discussion

This thesis presents four main research contributions. Foremost, data mesh and its core principles are defined, including its benefits, concerns, and applicability. Secondly, the thesis presents a reference architecture at design-time that depicts the different roles and their responsibilities with regards to creating managed data products in a data mesh architecture. Likewise, the thesis presents a reference architecture at runtime that depicts the logical dependencies between data providers, data products, data consumers, the management plane and the self-service infrastructure. Lastly, the thesis describes how the data mesh architecture can be instantiated on the Azure public cloud provider. This section discusses the individual contributions and any threats to validity.

10.1 Comprehensive Definition of Data Mesh

As Deghani states in her original write-ups, her main intention was to provide a common language and mental model for data mesh which still had to be elaborated on. In the two years that followed, industry published many articles on data mesh. However, little academic literature exists, indicating a gap between industry practice and academic research. Moreover, a consistent and complete definition of data mesh was lacking due to the large variety of articles published by industry, each addressing either only a part of data mesh or addressing the concept slightly differently. This was problematic because it hinders communication and there is no framework within which academics and industry practitioners can position their work. Moreover, the absence of a complete and consistent definition of data mesh makes it difficult for organizations to adopt the paradigm. This thesis synthesizes the work published by industry by conducting a systematic gray literature review, resulting in a complete and consistent definition of data mesh as an extension of Deghani's original work.

Section 4.1 provides a concise definition of data mesh based on the findings of the systematic gray literature review. The subsequent four sections give a detailed and specific description of the four principles data mesh is based on **(SQ1.1)**. Moreover, chapter 4 explains the benefits and concerns of a data mesh architecture along with a description of the organizations that can and cannot benefit from a data mesh architecture **(SQ1.2, 1.3, 1.4)**. The first set of research questions provides a consistent and complete definition of data mesh, hereby creating a shared understanding of the paradigm which can help future research in the field. Moreover, it helps organizations make a well-considered judgment about the applicability of the data mesh architecture to their organization.

10.2 Reference Architecture at Design-time and Run-time

Reference architectures provide a template solution architecture for a specific domain. Reference architectures are beneficial to industry as it creates a common language and industry-accepted solution based on best practices (Clements et al., 2003). Deghani's original work lacked a reference architecture, inhibiting organizations from actually adopting the data mesh paradigm. This thesis presents two views of the data mesh reference architecture. The first view is at design-time, depicting the different roles, responsibilities and constructs in a data mesh architecture with regards to creating a managed

data product. The second view, a reference architecture at runtime, depicts the logical dependencies between data providers, data products, data consumers, the governance plane and the self-service infrastructure.

The layered architecture separates the responsibilities of the different actors in a data mesh architecture. It provides a common framework for organizations to organize themselves and distribute responsibilities throughout their organization. Moreover, the analogy made with service-oriented architectures, which are already heavily researched and used in practice, increases the validity of the results. This study acts as an extension of the status quo.

Although some of the gray literature describes some of the requirements for a reference architecture at runtime, a complete and consistent architecture was lacking. The gray literature review synthesizes the constraints into a concrete reference architecture **(SQ2.1)**. This results in a vendor and technology agnostic template for industry to implement a data mesh architecture at runtime. The architecture makes use of design patterns that provide a common language and are proven by industry **(SQ2.2)**. Moreover, the thesis also presents an overview of the available tools to instantiate components of the reference architecture to facilitate adoption of the architecture **(SQ2.3)**.

Lastly, the thesis presents an instantiation of the data mesh architecture on the Azure public cloud provider **(SQ2.4)**. The instantiation is based on a synthesis of various instantiations provided by gray literature on different cloud providers. As more and more organizations are slowly transitioning to cloud based data architectures, the instantiation shows how data mesh can be instantiated on Azure. Moreover, cloud providers commonly abstract away design patterns into specific services. The presented instantiation makes it more straightforward how these services relate to the data mesh principles.

10.3 Demonstration and Validation of Reference Architecture

The contributions produced in this research were validated through a demonstration in an industrial setting. A minimum viable product for the self-serve infrastructure platform was built that was used by a domain at ASML to create and provision a data product. It proves our idea by demonstrating the separation of responsibilities in a data mesh architecture in line with the reference architecture at design-time. Moreover, it instantiates the reference architecture at runtime on the Azure public cloud provider. Although it is not a complete implementation, we believe that future practitioners can take this implementation as a starting point when transitioning toward a data mesh.

Moreover, the results were validated by means of interviews with senior representatives from industry **(SQ2.5)**. By conducting interviews with the various stakeholders in the data mesh architecture, the thesis was able to produce a broad range of feedback. General opinion on the designed artifacts was positive and the provided feedback was reiterated in the designed artifacts, making them more reliable overall. This increases the likelihood of adoption by industry. Moreover, it acts as an additional indication of the reliability of the research contributions.

10.4 Threats to Validity

This section describes potential threats to validity of this thesis and how they were mitigated.

First of all, there is the risk of having missed relevant studies for the systematic gray literature review. This risk was mitigated by systematically collecting the initial list of studies. However, when starting the research in September 2021, little gray literature was available on the architectural design of data mesh. Therefore, the initial list of studies was extended with more studies on the architectural design of data mesh published after September 2021. The collection of additional studies was done in a less systematic way, although they were still assessed to the same criteria and systematically analyzed. Since the field is developing so rapidly, it can be interesting to repeat the gray literature review in some years to see how the topic matures as it is adopted more by industry.

Secondly, the thesis presents several tools that can be used to instantiate components of the data mesh architecture. It is important to note that this list is a good starting point but does not provide a complete overview of the available tools. Moreover, the tools are not assessed systematically regarding their appropriateness but rather selected based on the frequency of them being mentioned in gray literature. Especially as the field matures, future work can provide a more elaborate consideration of the tools available for data mesh.

Regarding the validation of the research contributions, three out of five interview participants were data architects. Therefore, feedback from their perspective might be overrepresented. More expert interviews can be conducted to validate the architecture further but we are confident that a more appropriate next step would be to more extensively test the designed artifacts in practice.

Chapter 11

Conclusion

This chapter concludes the conducted research by summarizing the main research contributions and their limitations. Moreover, it presents several suggestions for future work by academia or industry.

11.1 Conclusion

This section answers the sub-questions and main research question of this thesis. Moreover, it provides a concise overview of the research contributions.

Answers to Research Questions

The thesis presented two sets of sub-questions to answer the main research question. The first set of research questions relates to the definition of data mesh, its benefits, drawbacks and applicability. It is answered using the results of the systematic gray literature review and the validation interviews. Each question is answered below.

1. What is data mesh and how does it differ from traditional data architectures for sharing data in organizations?
Data Mesh is a socio-technical big data architecture in which business domains close to the origin of the data are responsible for serving their data assets as a product. The domains are supported by a self-serve infrastructure platform that abstracts away the complexity of building and managing the infrastructure necessary to provision data as a product. Moreover, data mesh uses a federated governance model in which domains are responsible for governing their data products. Global governance is mainly used to increase interoperability between data products. The governance model is supported by automation which enables it to scale through the organization effectively.

2. Why is data mesh beneficial for data sharing compared to traditional data architectures?
There are several benefits of data mesh in comparison to traditional data architectures. Data mesh is much more scalable, allowing organizations to more quickly onboard new data sources and data consumers while delivering high quality data at a rapid pace. This enables experimentation and increases the agility of the organization.

Moreover, the federated governance model is more effective in governing Big data because it makes business domains responsible for governing their own data. Using their domain knowledge, they can make better decisions that ensure compliance and high-quality data.

3. What are the challenges in realizing a data mesh architecture?
Gray literature recognizes several challenges when adopting the data mesh paradigm. First of all, data mesh is not only a technical solution but also an organizational solution. Therefore, to implement it, a significant amount of change management and strong leadership is necessary to move the organization in the right direction. Moreover, the lack of data engineering talent may make it challenging for organizations to provide all their business domains with the right capabilities to build and serve their

data as products.

Moreover, there is a concern of data being duplicated in the organization as consumers make use of data coming from other domains and copy it to their bounded context. Similarly, gray literature recognizes that there may be a duplication of efforts in the domains, although this is addressed as much as possible by the self-service infrastructure platform.

4. For which organizations is data mesh most suited?
Data Mesh is complex and requires significant investments into technology and people. It is, therefore not the most suitable architecture for all organizations. Gray literature recognizes that data mesh is especially suitable for large organizations with a diverse data landscape (i.e. many data producers and consumers). Moreover, organizations that need to be agile and experiment with data on a large scale can benefit from data mesh. Lastly, organizations that need better data governance (e.g. because of external regulations) should consider data mesh as it offers considerable benefits regarding governance compared to traditional data architectures.

The second set of research questions was tailored to designing a reference architecture for data mesh and instantiating it. The Design Science Research Methodology was used to create the artifacts, with the systematic gray literature review providing the input for the artifacts.

5. What is a reference architecture for a data mesh?
The thesis presents two views of the reference architecture. One of them is a reference architecture at design-time depicting the different roles, responsibilities and constructs in a data mesh architecture with regards to creating managed data products, the second is a reference architecture at runtime that depicts the logical dependencies at runtime between data providers, data consumers, data products, the self-service infrastructure and the governance plane. The architectures provide a common industry-accepted solution template that organizations can use to implement data mesh.

6. What are common design patterns for realizing a data mesh?
The reference architecture at runtime makes use of several design patterns. It uses the change data capture pattern to detect and distribute data changes to downstream consumers. The publish / subscribe pattern is used for real-time data products or the distribution of changes captured by the change data capture pattern. Moreover, the request / reply pattern is used for ad hoc data requests by consumers. Lastly, the Sidecar pattern is used to execute policies in a distributed fashion.

7. What tools are available to instantiate a data mesh architecture?
The thesis presents several tools that can be used to instantiate a data mesh architecture. REST can be used to implement the ad hoc, pull-based product interface. OpenAPI can be used to describe product interfaces. Kafka can be used as a message backbone for the data and management channel. The open-source Debezium platform can be used to instantiate the change data capture pattern, and the Open Policy Agent can be used to execute policies written as code. Although more tools are available to instantiate data mesh, this is a good starting point.

8. How can the reference architecture be instantiated using a public cloud provider?
Based on a synthesis of numerous instantiations presented in gray literature, the thesis presents an instantiation of data mesh on the Azure public cloud provider. The resources offered by Azure are related to the data mesh principles, and a potential illustrative instantiation is given. Moreover, the thesis explains how the computational federated governance model can be implemented on Azure.

9. How do relevant stakeholders evaluate the reference architecture?
All the results presented in the thesis were validated through interviews with senior stakeholders from industry. The general response was positive while some minor changes were made to the reference architectures to make them more straightforward. Also, some parts were elaborated on more in the description.

Based on the sub-questions, the main research question can be answered. The main RQ was:

What is data mesh, and how can a data mesh be designed and implemented systematically in an organization?

Data Mesh is a socio-technical big data architecture in which business domains close to the origin of the data are responsible for serving their data assets as a product. The domains are supported by a self-serve infrastructure platform that abstracts away the complexity of building and managing the infrastructure necessary to provision data as a product. Moreover, data mesh uses a federated governance model in which domains are responsible for governing their data products.

Organizations can use the reference architecture at design-time as a guideline to structure themselves. Moreover, the framework can help separate responsibilities and distribute them to the various roles in a data mesh architecture. Furthermore, the reference architecture at runtime can function as a template to implement the logical architecture. To instantiate the runtime architecture, organizations can use some of the described tools or the cloud instantiation.

Theoretical Contributions

A. R. Hevner et al. (2004) guidelines for effective design-science research require a clear and verifiable contribution in the area of the design artifact. The design artifacts resulting from this research are outlined below.

(C1) **Data Mesh Specification:** A definition of data mesh based on a synthesis of the findings in the systematic gray literature review. The definition is based on the four principles of data mesh - Data as a Product, Domain Ownership, Federated Computational Governance and Self-Service Infrastructure - and provides a shared understanding of the paradigm for academia and industry. Moreover, the expected benefits and drawbacks are defined, which helps organizations assess if the architecture suits their needs. Lastly, the thesis describes which organizations should and which should not consider adopting data mesh.

(C2) **Reference Architecture at Design-time:** A layered architecture that depicts the different roles and their responsibilities in a data mesh architecture with regards to creating managed data products. The architecture uses a separation of concerns to separate the capabilities provided by the self-serve platform from data products and managed data products.

(C3) **Reference Architecture at Runtime:** An architecture that depicts the logical dependencies between data providers, data products, data consumers, the governance plane and the self-service infrastructure at runtime. Moreover, the thesis presents an overview of design patterns used in the architecture and several tools that can be used to instantiate the architecture at runtime.

(C4) **Instantiation:** The thesis presents an instantiation of the data mesh architecture on a public cloud provider. Furthermore, an illustrative use case demonstrates some of the data mesh principles by building a data product blueprint. The blueprint is used in an industrial setting at ASML by having a business domain serve one of their data assets as a product by using the developed blueprint.

To conclude, this thesis provides an academic contribution by defining and designing a data mesh architecture based on a systematic gray literature review. Moreover, a practical contribution is made by providing an overview of the available tools available for a data mesh architecture and showcasing how a data product blueprint can be used to build a data product. All results presented in the thesis are validated through expert interviews to increase the legitimacy of the results.

11.2 Limitations

This section discusses the limitations of the results presented in this thesis.

First of all, it is important to view the results presented in this thesis in relation to the topology defined in chapter 4. As described in this chapter, there are several topologies for data mesh with varying degrees of centralization. Although this thesis presents a complete reference architecture for a governed

mesh topology, it is good to recognize that there might be changes to the given reference architecture for different topologies. Furthermore, while the results provide a proper definition and reference architecture for data mesh, it ignores the fact that organizations already have a data architecture in place. These organizations have established processes, governance procedures and people in specific positions. Changing these organizational aspects of a data architecture will require a significant change in mentality and perhaps restructuring part of the organization. There is a high probability of resistance to this change. A significant amount of change management will be required to successfully migrate to a decentralized data architecture. The thesis fails to address how organizations can successfully migrate from their traditional data architecture to the data mesh architecture.

Moreover, it will be challenging to move the entire organization in the (exact) same direction. There will be many different opinions on how the architecture should be implemented in a specific organization. Many complex and time-consuming discussions between all layers of the organization will have to take place to get everyone in the right direction. These challenges are mentioned in the thesis, but no guidance is given as to how to navigate these challenges. Data mesh is not only a technological solution but also an organizational solution.

The thesis presents an instantiation for data mesh on Microsoft Azure. In practice, organizations will most likely use multiple public cloud providers and their private cloud. This adds additional challenges, for example, the implementation of the computational governance model, that are not addressed in this thesis.

Lastly, it is good to recognize the limitations of the illustrative use case. Although the use case implements components of a data mesh architecture, it is not an instantiating of the complete data mesh architecture. However, it does provide a good starting point for a more elaborate demonstration.

11.3 Recommendations for Future Work

The recommendations for future work can be split up into academic and industrial recommendations. Both are presented below.

Academic Recommendations

Some of the limitations discussed above can be addressed by future research. First of all, future research can focus on how to migrate from a traditional data architecture to a data mesh. This can provide a framework, or set of steps, for organizations to follow to smoothen the migration to a decentralized data architecture.

To continue, research on multi-cloud implementations of data mesh can be conducted. The thesis presents an instantiation for data mesh on Azure but does not place the presented results in the context of multiple clouds. Additional research can focus on these multi-cloud implementations.

Besides addressing the limitations of this thesis, additional research can be done to extend the data mesh principles. Organizations are slowly integrating more and more with their suppliers and customers. This increases the need for data exchange between these entities, as mentioned in several validation interviews. The data mesh paradigm can be applied in this inter-organizational context as well. Future research can address the challenges of an inter-organizational data architecture (e.g. ownership of data). Inter-organizational data mesh also relates strongly to the European GAIA-X initiative, which is described in section 2.5.

Industrial Recommendations

Expert interviews validate the proposed reference architecture to a certain extent. It is recommended to further validate it by implementing it in an industrial setting. Additional validation can provide feedback for further improvements of the architecture and provide a set of best practices for data mesh in practice. Moreover, the list of available tools for data mesh is incomplete and should be

extended, especially due to the rapidly changing landscape. Best practices from industry can help identify the benefits and drawbacks of specific tools.

Bibliography

Abraham, R., Schneider, J., & Vom Brocke, J. (2019). Data governance: A conceptual framework, structured review, and research agenda. *International Journal of Information Management*, *49*, 424–438.

Ahamed, N. (2019). Azure event grid vs event hub.

Alhassan, I., Sammon, D., & Daly, M. (2016). Data governance activities: An analysis of the literature. *Journal of Decision Systems*, *25*(sup1), 64–75.

Analytics, A. (2021). Data mesh deep dive.

Arsanjani, A., Zhang, L.-J., Ellis, M., Allam, A., & Channabasavaiah, K. (2007). S3: A service-oriented reference architecture. *IT professional*, *9*(3), 10–17.

Autolitano, S., & Pawlowska, A. (2021). Europe's quest for digital sovereignty: Gaia-x as a case study. *IAI Papers*, (21), 14.

Bakshi, K. (2012). Considerations for big data: Architecture and approach. *2012 IEEE aerospace conference*, 1–7.

Berman, S. J., Kesterson-Townes, L., Marshall, A., & Srivathsa, R. (2012). How cloud computing enables process and business model innovation. *Strategy & Leadership*.

Bernhardt, J., & Seese, D. (2008). A conceptual framework for the governance of service-oriented architectures. *International Conference on Service-Oriented Computing*, 327–338.

Borgman, H., Heier, H., Bahli, B., & Boekamp, T. (2016). Dotting the i and crossing (out) the t in it governance: New challenges for information governance. *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 4901–4909.

Brackenbury, W., Liu, R., Mondal, M., Elmore, A. J., Ur, B., Chard, K., & Franklin, M. J. (2018). Draining the data swamp: A similarity-based approach. *Proceedings of the workshop on human-in-the-loop data analytics*, 1–7.

Braud, A., Fromentoux, G., Radier, B., & Le Grand, O. (2021). The road to european digital sovereignty with gaia-x and idsa. *IEEE Network*, *35*(2), 4–5.

Burns, B. (2018). *Designing distributed systems: Patterns and paradigms for scalable, reliable services*. " O'Reilly Media, Inc.".

Burns, B., & Oppenheimer, D. (2016). Design patterns for container-based distributed systems. *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*.

Cardon, D. (2014). Database vs data warehouse: A comparative review. *Health catalyst*.

Clements, P., Garlan, D., Little, R., Nord, R., & Stafford, J. (2003). Documenting software architectures: Views and beyond. *25th International Conference on Software Engineering, 2003. Proceedings.*, 740–741.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, *20*(1), 37–46.

Cremaschi, M., & Paoli, F. D. (2017). Toward automatic semantic api descriptions to support services composition. *European Conference on Service-Oriented and Cloud Computing*, 159–167.

Curbera, F., Nagy, W., & Weerawarana, S. (2001). Web services: Why and how. *Workshop on Object-Oriented Web Services-OOPSLA*, *2001*.

Das, S., Botev, C., Surlaker, K., Ghosh, B., Varadarajan, B., Nagaraj, S., Zhang, D., Gao, L., Westerman, J., Ganti, P., et al. (2012). All aboard the databus! linkedin's scalable consistent change data capture platform. *Proceedings of the third ACM symposium on cloud computing*, 1–14.

Dean, J., & Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters.

Dehghani, Z. (2020a). Data mesh principles and logical architecture.

Dehghani, Z. (2020b). Data mesh principles and logical architecture.

DeJonckheere, M., & Vaughn, L. M. (2019). Semistructured interviewing in primary care research: A balance of relationship and rigour. *Family medicine and community health*, *7*(2).

Dillon, T., Wu, C., & Chang, E. (2010). Cloud computing: Issues and challenges. *2010 24th IEEE international conference on advanced information networking and applications*, 27–33.

Dobbelaere, P., & Esmaili, K. S. (2017). Kafka versus rabbitmq: A comparative study of two industry reference publish/subscribe implementations: Industry paper. *Proceedings of the 11th ACM international conference on distributed and event-based systems*, 227–238.

Dreibelbis, A., Hechler, E., Milman, I., Oberhofer, M., van Run, P., & Wolfson, D. (2008). *Enterprise master data management (paperback): An soa approach to managing core information*. Pearson Education.

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). Devops. *Ieee Software*, *33*(3), 94–100.

El-Sappagh, S. H. A., Hendawi, A. M. A., & El Bastawissy, A. H. (2011). A proposed model for data warehouse etl processes. *Journal of King Saud University-Computer and Information Sciences*, *23*(2), 91–104.

Erl, T. (2007). *Soa principles of service design (the prentice hall service-oriented computing series from thomas erl)*. Prentice Hall PTR.

Eugster, P. T., Felber, P. A., Guerraoui, R., & Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, *35*(2), 114–131.

Evans, E., & Evans, E. J. (2004). *Domain-driven design: Tackling complexity in the heart of software*. Addison-Wesley Professional.

Feick, M., Kleer, N., & Kohn, M. (2018). Fundamentals of real-time data processing architectures lambda and kappa. *SKILL 2018-Studierendenkonferenz Informatik*.

Gai, K., & Li, S. (2012). Towards cloud computing: A literature review on cloud computing and its development trends. *2012 Fourth International Conference on Multimedia Information Networking and Security*, 142–146.

Garousi, V., Felderer, M., & Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology*, *106*, 101–121.

Gillin, P. (2021). Data warehousing has problems. a data mesh could be the solution.

Gottschalk, K., Graham, S., Kreger, H., & Snell, J. (2002). Introduction to web services architecture. *IBM systems Journal*, *41*(2), 170–177.

Halili, F., & Ramadani, E. (2018). Web services: A comparison of soap and rest services. *Modern Applied Science*, *12*(3), 175.

Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. *Design research in information systems* (pp. 9–22). Springer.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 75–105.

Holst, A. (2021). Volume of data / information created, captured, copied, and consumed worldwide from 2010 to 2025.

IDSA. (2019). Ids reference architecture model 3.0.

Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models (Standard). (2011). International Organization for Standardization. Geneva, CH.

John, T., & Misra, P. (2017). *Data lake for enterprises*. Packt Publishing Ltd.

Kallio, H., Pietilä, A.-M., Johnson, M., & Kangasniemi, M. (2016). Systematic methodological review: Developing a framework for a qualitative semi-structured interview guide. *Journal of advanced nursing*, *72*(12), 2954–2965.

Keboola. (2021). Data mesh - the answer to the failures of centralized data architectures.

Khatri, V., & Brown, C. V. (2010). Designing data governance. *Communications of the ACM*, *53*(1), 148–152.

Khine, P. P., & Wang, Z. S. (2018). Data lake: A new ideology in big data era. *ITM web of conferences*, *17*, 03025.

Kitto, S. C., Chesters, J., & Grbich, C. (2008). Quality in qualitative research. *Medical journal of Australia*, *188*(4), 243–246.

Kreps, J., Narkhede, N., Rao, J., et al. (2011). Kafka: A distributed messaging system for log processing. *Proceedings of the NetDB*, *11*, 1–7.

Kumara, I., Garriga, M., Romeu, A. U., Di Nucci, D., Palomba, F., Tamburri, D. A., & van den Heuvel, W.-J. (2021). The do's and don'ts of infrastructure code: A systematic gray literature review. *Information and Software Technology*, *137*, 106593.

Lam, M. (2021). Why snowflake is a good match for implementing data mesh.

Lin, J. (2017). The lambda and the kappa. *IEEE Internet Computing*, *21*(05), 60–66.

Longhurst, R. (2003). Semi-structured interviews and focus groups. *Key methods in geography*, *3*(2), 143–156.

Loshin, D. (2010). *Master data management*. Morgan Kaufmann.

MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R., & Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard*, *12*(S 18).

Madera, C., & Laurent, A. (2016). The next information architecture evolution: The data lake wave. *Proceedings of the 8th international conference on management of digital ecosystems*, 174–180.

Marjani, M., Nasaruddin, F., Gani, A., Karim, A., Hashem, I. A. T., Siddiqa, A., & Yaqoob, I. (2017). Big iot data analytics: Architecture, opportunities, and open research challenges. *ieee access*, *5*, 5247–5261.

Menge, F. (2007). Enterprise service bus. *Free and open source software conference*, *2*, 1–6.

Microsoft. (2021). Organize your azure resources effectively.

Millett, S., & Tune, N. (2015). *Patterns, principles, and practices of domain-driven design*. John Wiley & Sons.

Miloslavskaya, N., & Tolstoy, A. (2016). Big data, fast data and data lake concepts. *Procedia Computer Science*, *88*, 300–305.

Mohammed, C. M., Zeebaree, S. R. et al. (2021). Sufficient comparison among cloud computing services: Iaas, paas, and saas: A review. *International Journal of Science and Business*, *5*(2), 17–30.

Mumbaikar, S., Padiya, P. et al. (2013). Web services based on soap and rest principles. *International Journal of Scientific and Research Publications*, *3*(5), 1–4.

Nadeem, A., & Javed, M. Y. (2005). A performance comparison of data encryption algorithms. *2005 international Conference on information and communication technologies*, 84–89.

Nayak, A., Poriya, A., & Poojary, D. (2013). Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems*, *5*(4), 16–19.

Nurseitov, N., Paulson, M., Reynolds, R., & Izurieta, C. (2009). Comparison of json and xml data interchange formats: A case study. *Caine*, *9*, 157–162.

Oostra, G. (2021). Two steps towards a modern data platform.

Otto, B., Rubina, A., Eitel, A., Teuscher, A., Schleimer, A. M., Lange, C., Stingl, D., Loukipoudis, E., Brost, G., Böge, G., Pettenpohl, H., Langkau, J., Gelhaar, J., Mitani, K., Hupperz, M., Huber, M., Jahnke, N., Brandstädter, R., Wessel, S., & Bader, S. (2021). Gaia-x and ids.

Panian, Z. (2010). Some practical experiences in data governance. *World Academy of Science, Engineering and Technology*, *62*(1), 939–946.

Papazoglou, M. (2008). *Web services: Principles and technology*. Pearson Education.

Papazoglou, M. P., & Van Den Heuvel, W.-J. (2007). Service oriented architectures: Approaches, technologies and research issues. *The VLDB journal*, *16*(3), 389–415.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, *24*(3), 45–77.

Perrons, R. K., & Jensen, J. W. (2015). Data as an asset: What the oil and gas sector can learn from other industries about "big data". *Energy Policy*, *81*, 117–121.

Pipino, L. L., Lee, Y. W., & Wang, R. Y. (2002). Data quality assessment. *Communications of the ACM*, *45*(4), 211–218.

Rajagopalan, R. (2021). Demystifying data mesh.

Ramesh, B. (2015). Big data architecture. *Big data* (pp. 29–59). Springer.

Richardson, L., & Ruby, S. (2008). *Restful web services*. " O'Reilly Media, Inc.".

Roima, A. (2021). Data mesh - what is it, what are its benefits and when to consider it?

Roy, J., & Ramanujan, A. (2001). Understanding web services. *IT professional*, *3*(6), 69–73.

Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. *2013 international conference on collaboration technologies and systems (CTS)*, 42–47.

Saldaña, J. (2021). *The coding manual for qualitative researchers*. sage.

Santos, J. F. M., Guessi, M., Galster, M., Feitosa, D., & Nakagawa, E. Y. (2013). A checklist for evaluation of reference architectures of embedded systems (s). *SEKE*, *13*, 1–4.

Serra, J. (2021). Data mesh defined.

Shankar, N., Meyers, I., Mitchell, Z., & Hasson, R. (2021). Design a data mesh architecture using aws lake formation and aws glue.

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The hadoop distributed file system. *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, 1–10.

Snowflake. (2021). Empower data teams with a data mesh built on snowflake.

Srivastava, P., & Khan, R. (2018). A review paper on cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, *8*(6), 17–20.

Stanton, P. (2004). Securing data in storage: A review of current research. *arXiv preprint cs/0409034*.

Stein, B., & Morrison, A. (2014). The enterprise data lake: Better integration and deeper analytics. *PwC Technology Forecast: Rethinking integration*, *1*(1-9), 18.

Tallon, P. P., Ramirez, R. V., & Short, J. E. (2013). The information artifact in it governance: Toward a theory of information governance. *Journal of Management Information Systems*, *30*(3), 141–178.

Tanwar, M., Duggal, R., & Khatri, S. K. (2015). Unravelling unstructured data: A wealth of information in big data. *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*, 1–6.

Terraform. (2022). Terraform language documentation.

Thomas, D. R. (2003). A general inductive approach for qualitative data analysis.

Thomas, D. R. (2006). A general inductive approach for analyzing qualitative evaluation data. *American journal of evaluation*, *27*(2), 237–246.

Thomas, H., & Datta, A. (2001). A conceptual model and algebra for on-line analytical processing in decision support databases. *Information Systems Research*, *12*(1), 83–102.

Traulsen, S., Tröbs, M., AG, U., Tucherpark, A., & Ganghoferstrasse, C. (2011). Implementing data governance within a financial institution. *GI-Jahrestagung*, 195.

Vernon, V. (2013). *Implementing domain-driven design*. Addison-Wesley.

Warren, J., & Marz, N. (2015). *Big data: Principles and best practices of scalable realtime data systems*. Simon; Schuster.

Weill, P., & Ross, J. W. (2004). *It governance: How top performers manage it decision rights for superior results*. Harvard Business Press.

Wende, K. (2007). A model for data governance–organising accountabilities for data quality management.

Zhou, Y., Zhao, Q., & Perry, M. (2002). Policy enforcement pattern. *Proceedings of the Conference on Pattern Languages of Programs*, 1–14.

Appendix A

Gray Literature Review: Studies

This appendix provides an overview of the studies selected in the gray literature review.

S1 Zhamak Dehghani. Data Mesh Principles and Logical Architecture. martinFowler, 2020. https://martinfowler.com/articles/data-mesh-principles.html

S2 Zhamak Dehghani. How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh. martinFowler, 2020. https://martinfowler.com/articles/data-monolith-to-mesh.html

S3 Jarvin Mutatiina, Can Yurtseven, Ernst Blaauw. From data mess to a data mesh. Deloitte, 2020. https://towardsdatascience.com/what-is-a-data-mesh-and-how-not-to-mesh-it-up-210710bb41e0

S4 Barr Moses. What is a Data Mesh - and How Not to Mesh it Up. Towardsdatascience, 2020. https://towardsdatascience.com/what-is-a-data-mesh-and-how-not-to-mesh-it-up-210710bb41e0

S5 Aleksi Roima. When should organizatoins consider data mesh?. Futurice, 2021. https://futurice.com/blog/when-should-organizations-consider-data-mesh

S6 Max Schultze, Arif Wider. Data Mesh in Practice: How Europe's Leading Online Platform for Fashion Goes Beyond the Data Lake. Sparkt AI , 2020. https://databricks.com/session_na20/data-mesh-in-practice-how-europes-leading-online-platform-for-fashion-goes-beyond-the-data-lake

S7 James Serra. Data Mesh defined. martinFowler, 2021. https://www.jamesserra.com/archive/2021/02/data-mesh/

S8 Executives. 6 Keynotes: Data Mesh Explanation. AgileLab, 2021. https://www.agilelab.it/data-mesh-in-action/

S9 Bhavesh Furia. Data Mesh - Rethinking Enterprise Data Architecture. CueLogic, 2021. https://www.cuelogic.com/blog/data-mesh

S10 Paul Anu, N, Arup. Evolution of Data mesh Architecture Can Drive Signficiant Value in Modern Enterprise. JPMorgan, 2021. https://www.jpmorgan.com/technology/technology-blog/evolution-of-data-mesh-architecture

S11 Jeffrey Pollock. Data Mesh is not a Data Lake!. martinFowler, LinkedIn. https://www.linkedin.com/pulse/data-mesh-lake-jeffrey-t-pollock/

S12 Heather Devane. What is a Data Mesh?. Immuta, 2020. https://www.immuta.com/articles/what-is-a-data-mesh/

S13 Nivas Shankar, Ian Meyers, Zach Mitchell, Roy Hasson. Design a data mesh architecture using AWS Lake Formation and AWS Glue. Design a data mesh architecture using AWS Lake Formation and AWS Glue, 2021. https://aws.amazon.com/blogs/big-data/design-a-data-mesh-architecture-using-aws-lake-formation-and-aws-glue/

S14 David Vellante. Breaking Analysis: How JP Morgan is Implementing a Data Mesh on the AWS Cloud. WikiBon, 2021. https://wikibon.com/breaking-analysis-how-jp-morgan-is-implementing-a-data-mesh-on-the-aws-cloud/

S15 Olivier Wulveryck. POV: A streaming/communication platform for the data mesh. Octo, 2021. https://blog.octo.com/en/pov-a-streaming-communication-platform-for-the-data-mesh/

S16 Paul Gillin. Data Warehousing has problems. A data mesh could be the solution. Silicon Angle, 2021. https://siliconangle.com/2021/08/06/data-warehousing-problems-data-mesh-solution/

S17 Mathias golombek. Data Mesh in Practice: Learnings from a customer journey. DataVersity, 2021. https://www.dataversity.net/data-mesh-in-practice-learnings-from-a-customer-journey/#

S18 -. Data mesh: The Four Principles of a Distributed Architecture. Data Driven Investor, 2021. https://medium.datadriveninvestor.com/data-mesh-the-four-principles-of-a-distributed-architecture-59514eba1e52

S19 Martin Lam. Why Snowflake is a good match for implementing Data Mesh. Capgemini, 2021. https://www.capgemini.com/no-no/2021/05/why-snowflake-is-a-good-match-for-implementing-data-mesh/

S20 Jesse Menning. Use an Event-Driven Data Mesh to Avoid Drowning in the (Data) Lake. Solace, 2021. https://solace.com/blog/event-driven-data-mesh/

S21 Chris Riccomini. What the Heck is a Data Mesh?!. CNR, 2021. https://cnr.sh/essays/what-the-heck-data-mesh

S22 Pawel Mitrus. Data Mesh Explained. An End-to-End Guide to the Latest Data Architecture Trend. Langaro Group, 2021. https://lingarogroup.com/blog/data-mesh-explained-an-end-to-end-guide-to-the-latest-data-architecture-trend

S23 Andrew Carr. What Actually is a data mesh? And is it really a thing?. Scott Logic, 2021. https://blog.scottlogic.com/2021/05/28/what-actually-is-a-data-mesh-and-is-it-really-a-thing.html

S24 Trey Hicks. Catching Data in a Data Mesh: Principles (part I). Medium, 2021. https://medium.com/nerd-for-tech/catching-data-in-a-data-mesh-principles-part-i-2b2e11e9e33a

S25 Mullaned2002. What is Data Mesh (and who should be using it). Data Integration, 2021. https://dataintegration.info/what-is-data-mesh-and-who-should-be-using-it

S26 -. 6 Questions to ask before implementing data mesh. Kainos, 2021. https://www.kainos.com/insights/blogs/thinking-about-data-mesh

S27 Doichin Yordanov. DData Mesh, the new data paradigm set to rise. Helecloud, 2021. https://helecloud.com/blog/data-mesh-the-new-data-paradigm-set-to-rise/

S28 Yval Perlov. Data Mesh: Architecture, Use Cases, and Implementation via Data Fabric. K2View, 2021. https://www.k2view.com/blog/data-mesh/

S29 -. Enterprise Data Mesh. Oracle, 2021. https://www.oracle.com/a/ocom/docs/datamesh-ebook.pdf

S30 Adam Gugliciello. The Journey to an Enterprise Data Mesh. Data Iku, 2021. https://blog.dataiku.com/the-journey-to-an-enterprise-data-mesh

S31 Justin Borgman. Data Mesh: A paradigm shift in enterprise data management. ITProportal, 2021. https://www.itproportal.com/features/data-mesh-a-paradigm-shift-in-enterprise-data-management/

S32 Muthu Chinnasamy Vinod Menon. Modernize data between siloed data warehouses with Infosys Data Mesh and MongoDB. MongoDB, 2021. https://www.mongodb.com/blog/post/modernize-data-between-siloed-data-warehouses-infosys-data-mesh-monogdb

S33 -. Data mesh - the answer to the failures of centralized data architectures. Keboola, 2021. https://www.keboola.com/blog/data-mesh

S34 -. IBM — Data mesh for hybrid cloud. IBM KPMG, -. https://www.kpmg.us/alliances/kpmg-ibm/data-mesh-hybrid-cloud.html

S35 Rajesh Rajagopalan. Demystifying Data Mesh. PeerIsland, 2021. https://www.peerislands.io/demystifying-data-mesh/

S36 Zhamak Dehghani. Data Mesh: Defining a new data architecture paradigm with Zhamak Dehghani. ToroCloud, 2021. https://www.torocloud.com/podcast/defining-data-mesh-zhamak-dehghani

37 -. Data Mesh Deep Dive. Advancing Analytics, 2021. https://www.advancinganalytics.co.uk/blog/2021/8/5/data-mesh-deep-dive

S38 Zhamak Dehghani. Data Mesh with Zhamak Dehghani. Software Engineering Daily, 2019. https://softwareengineeringdaily.com/2019/07/29/data-mesh-with-zhamak-deghani/

S39 Devayon Das. Data Mesh: How to Overcome Data Lake Challenges. Zaloni, 2020. https://www.zaloni.com/resources/data-mesh-data-lake-challenges/

S40 Chandan Gaur. Complete overview of data mesh and its benefits. Xenonstack, 2021. https://www.xenonstack.com/blog/data-mesh

S41 Saket Saurabh. Data Mesh: Desing, Benefits, Hype and Reality. Nexla, -. https://www.nexla.com/data-mesh-design-benefits-hype-reality/

S42 Gerben Oostra. Two steps towards a modern data platform. Medium, 2021. https://medium.com/bigdatarepublic/two-steps-towards-a-modern-data-platform-37c74e7c104b

S43 -. Empower Data Teams with a Data Mesh Built on Snowflake. Snowflake, 2021. https://www.snowflake.com/blog/empower-data-teams-with-a-data-mesh-built-on-snowflake/

S44 Jason Gilmore. Easier Data Marts with DreamFactory Data Mesh. DreamFactory, 2019. https://blog.dreamfactory.com/easier-data-marts-with-dreamfactory-data-mesh/

S45 Patryk Orwat. Data Mesh on AWS. DEV, 2021. https://dev.to/aws-builders/data-mesh-on-aws-57ah

S46 -. Data mesh: how to work with data without a monolith. Prog.World, 2021. https://prog.world/data-mesh-how-to-work-with-data-without-a-monolith/

S47 Kenneth Dalvik. SAP Data Warehsouse + Data Mesh = True. SAP, 2020. https://blogs.sap.com/2020/10/22/sap-data-warehouse-cloud-data-mesh-true/

S48 -. What you need to know about data mesh. Sparke Quation, 2021. https://sparkequation.com/2021/02/24/data-mesh/

S49 Juan Sequeda. What is the deal with the Data Mesh?. Juan Sequeda, 2021. http://www.juansequeda.com/blog/2021/02/22/what-is-the-deal-with-the-data-mesh/

S50 Joe Gleinser. Should Your Application Consider Data Mesh Connectivity?. Forbes, 2021. https://www.forbes.com/sites/forbestechcouncil/2020/05/07/should-your-application-consider-data-mesh-connectivity/?sh=5ab427756d7f

S51 Einat Orr. Data Mesh Applied: How to move beyond the Data Lake with lakeFS. Lakefs, 2020. https://lakefs.io/data-mesh-applied-how-to-move-beyond-the-data-lake-with-lakefs/

S52 Firat Tekiner. Building a unified analytics data platform on Google Cloud. Google, 2021. https://cloud.google.com/blog/products/data-analytics/building-unified-analytics-data-platform-google-cloud

S53 Jon Cooke. Deploying Data Products at the speed of the business. Dataception, -. http://dataception.com/Data-Mesh-Deploying-Data-Products-at-the-speed-of-the-business.html

S54 Piethein Strengholt. Data Mesh topologies. Towardsdatascience, 2021. https://towardsdatascience.com/data-mesh-topologies-85f4cad14bf2

S55 Graham Stirling. Saxo Bank's Best Practices for a Distributed Domain-Driven Architecture Founded on the Data Mesh. Confluent, 2021. https://www.confluent.io/blog/distributed-domain-driven-architecture-data-mesh-best-practices/

S56 Azaz Rasool. Datamesh - a paradigm shift — are Data warehouses/ Data lakes dying or need a fresh perspective?. LinkedIn, 2021. https://www.linkedin.com/pulse/datamesh-paradigm-shift-data-warehouses-lakes-dying-azaz-rasool/

S57 Francois Nguyen. Towards a data mesh (part 2): Architecture Technologies. Francois Nguyen, 2021. https://francois-nguyen.blog/2021/03/22/toward-a-data-mesh-part-2-architecture-technologies/

S58 Data Mesh 101: how to get started. Montecarlodata, 2020.

S59 Rufus Lidman. The Contingency Model of Data Mesh. Towardsdatascience, 2021. https://towardsdatascience.com/the-contingency-model-of-data-mesh-c4bbe57577d6

S60 Seb B. Why Data Mesh 101: The Art and Science of Killer Data Discovery. LinkedIn, 2021. https://www.linkedin.com/pulse/why-data-mesh-101-art-science-killer-discovery-seb-bulpin/

S61 Oliver Bauer. Making a Mesh Start. Vistaprint, 2021. https://vistaprint.io/blog/making-a-mesh-start

S62 Abraham Enyo-one Musa. Data Management Architectures — Monolithic Data Architectures vs Distributed Data Mesh. Towardsdatascience, 2020. https://towardsdatascience.com/data-management-architectures-monolithic-data-architectures-and-distributed-data-mesh-63743794966c

S63 Nazia Shahrin. Building a successful Data Mesh – More than just a technology initiative. LinkedIn, 2021. https://www.linkedin.com/pulse/building-successful-data-mesh-more-than-just-nazia-shahrin/

S64 Anil Madan. Modern Data Platform - How to build one?. LinkedIn, 2021. https://www.linkedin.com/pulse/modern-data-platform-how-build-one-anil-madan/

S65 Fernando Raposo. The Evolution of FindHotel's Data Architecture - Part I. FindHotel, 2021. https://blog.findhotel.net/2021/07/the-evolution-of-findhotels-data-architecture-part-i/

S66 Adevinta. Building a data mesh to support an ecosystem of data products at Adevinta. Medium, 2021. https://medium.com/adevinta-tech-blog/building-a-data-mesh-to-support-an-ecosystem-of-data-products-at-adevinta-4c057d06824d

S67 Taavi Pungas. Dodging the data bottleneck — data mesh at Starship. Medium, 2021. https://medium.com/starshiptechnologies/dodging-the-data-bottleneck-data-mesh-at-starship-5925a2de45e6

S68 Kevin Lewis. Data Mesh and the Threads that Hold it together. Teradata, 2021. https://www.teradata.com/Blogs/Data-Mesh-and-the-Threads-that-Hold-it-Together

S69 Agile Lab. Data Mesh explanation. Medium, 2021. https://medium.com/bigdatarepublic/two-steps-towards-a-modern-data-platform-37c74e7c104b

S70 Vanessa Ericsson. Data Mesh Strategy and Architecture. Zenseact, 2021.

S71 Tristan baker. Intuit's Data Mesh Strategy. Medium, 2021. https://medium.com/intuit-engineering/intuits-data-mesh-strategy-778e3edaa017

S72 -. Data Movement in Netflix Studio via Data Mesh. Netflix, 2021. https://netflixtechblog.com/data-movement-in-netflix-studio-via-data-mesh-3fddcceb1059

S73 Ilan Raab Marco Chiapusso. How to create a modern CPG Data Architecture with Data Mesh. AWS, 2021. https://aws.amazon.com/blogs/industries/how-to-create-a-modern-cpg-data-architecture-with-data-mesh/

S74 Chris Butler. People Analytics needs to Support the Enterprise Data Mesh. OneModel, 2021. https://www.onemodel.co/blog/people-analytics-needs-to-support-the-enterprise-data-mesh

S75 Alexis McKenzie. A data mesh approach to data warehousing. LinkedIn, 2021. Adatameshapproachtodatawarehousing

S76 Chloe Feingold. Why Governance is the Critical Stitch in Data Mesh (and How to Avoid "Meshing" it Up). Privacera, 2021. https://blog.privacera.com/why-governance-is-the-critical-stitch-in-data-mesh-and-how-to-avoid-meshing-it-up-ade91a896ae5

S77 -. Data Mesh Whitepaper: The Data Mesh Shift. Thoughtworks, 2020.

S78 -. Technology Brief: Dynamic Data Fabric and Trusted Data Mesh using the oracle GoldenGate Platform. Oracle, 2021.

S79 James Reid. Implementing a Data Mesh Architecture at JPMC. JP Morgan Chase Company, 2021. https://www.dremio.com/subsurface/implementing-a-data-mesh-architecture-at-jpmc/

S80 Dave Wells. Data Architecture: Complex vs Complicated. Eckerson Group, 2021. https://www.eckerson.com/articles/data-architecture-complex-vs-complicated

S81 Eric Broda. Data Mesh Architecture Patterns. Towardsdatascience, 2022. https://towardsdatascience.com/data-mesh-architecture-patterns-98cc1014f251

S82 Eric Broda. An Architecture for the Data Mesh. Towardsdatascience, 2022. https://towardsdatascience.com/an-architecture-for-the-data-mesh-32ff4a15f16f

S83 Eric Broda. Data Mesh Architecture and the Role of APIs JSON Schemas. Towardsdatascience, 2022. https://towardsdatascience.com/data-mesh-architecture-and-the-role-of-apis-json-schemas-3dc616650960

S84 Eric Broda. Data Mesh Pattners: Change Data Capture. Towardsdatascience, 2022. https://towardsdatascience.com/data-mesh-pattern-deep-dive-change-data-capture-eb3090178c34

S85 Barr Moses. How to treat your data like a product. Towardsdatascience, 2022. https://towardsdatascience.com/how-to-treat-your-data-like-a-product-73731ec5f131

S86 -. Deloite Data Mesh. Deloite, 2022. https://www2.deloitte.com/content/dam/Deloitte/nl/Images/inline_images/deloitte-nl-sa-and-ma-data-mesh1-inline.jpg

S87 Piethein Strenholt Andrea Courtright. A financial institution scenario for data mesh. Microsoft, 2022. https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/data-management/architectures/reference-architecture-data-mesh

S88 -. Data Mesh for Trusted Public Sector Data Sharing in Singapore. Thoughtworks, 2021. https://www.thoughtworks.com/content/dam/thoughtworks/documents/whitepaper/whitepaper-tw-102021-ebook-data-mesh-singapore.pdf

S89 Adit Modi. Introduction to data mesh. DEV, 2021. https://dev.to/aws-builders/introduction-to-data-mesh-3f1b

S90 Piethein Strengholt. ABN Amro's data and integration mesh. ABNAmro, 2020. https://www.linkedin.com/pulse/abn-amros-data-integration-mesh-piethein-strengholt/

S91 Tareq Abedrabbo. Data Mesh in the real world: Lessons learnt from the financial markets. InfoQ, 2021. https://www.infoq.com/presentations/cmc-markets-challenges/

S92 Courtney Perio. Get More out of your data analytics microservices. Towardsdatascience, 2022. https://towardsdatascience.com/get-more-out-of-your-data-with-analytics-microservices-9a5a34a3ad2f

S93 P Platter. How and why data mesh is shapig the data maangements evolution. Medium, 2021. https://medium.com/agile-lab-engineering/yet-another-datamesh-article-83378b62f334

S94 Chris Dowsett. Using Microservices to Build and Scale Data Functions. Medium, 2021. https://towardsdatascience.com/using-microservices-to-build-and-scale-data-functions-28d47f400419

S95 Matt McLarty. How does API managmeent mesh with, um, data mesh. Mulesoft, 2021. https://blogs.mulesoft.com/api-integration/api-management-and-data-mesh/

S96 P Plattr. How to identify Data Products? Welcome "Data Product Flow". Medium, 2021. https://medium.com/agile-lab-engineering/how-to-identify-data-products-welcome-data-product-flow-76d7d85d23af

S97 Zhamak Dehghani. Data Mesh: An Architectural Deep Dive. InfoQ, 2021. https://www.infoq.com/presentations/data-mesh-concepts/

S98 Javier Ramos. Building a Data Mesh: A beginners Guide. ITNext, 2021. https://itnext.io/introduction-to-data-mesh-59e6f3a4c15e

S99 Alex Woodie. Data Mesh vs Data Fabric: Understanding the Differences. Datanami, 2021. https://www.datanami.com/2021/10/25/data-mesh-vs-data-fabric-understanding-the-differences/

S100 Max Schultze Arif Wider. Data Mesh in Practice. Zalando, 2020. https://www.iteblog.com/ppt/sparkaisummit-north-america-2020-iteblog/data-mesh-in-practice-how-europes-leading-online-platform-for-fashion-goes-beyond-the-data-lake-iteblog.com.pdf

S101 Eric Broda. Data Mesh Pattenrs: Event Streming Backbone. Towardsdatascience, 2022. https://towardsdatascience.com/data-mesh-pattern-deep-dive-event-streaming-backbone-99a5bb2a7cbf

S102 Paul Andrew. Building a Data Mesh Architecture in Azure series. Avenade, 2022. https://mrpaulandrew.com/2022/01/07/building-a-data-mesh-architecture-in-azure-part-3/

S103 Mansi Maharana. Implementing Data-as-a-Product(DaaP) using distributed data architecture and Smart Data Platform on GCP. Medium, 2021. https://medium.com/google-cloud/implementing-data-as-a-product-daap-using-distributed-data-architecture-and-smart-data-platform-on-c2fcd64c67d5

S104 Zhamak Dehghani. Reference Architecture . Amazon, 2021. https://docs.aws.amazon.com/wellarchitected/latest/analytics-lens/data-mesh-reference-architecture.html

S105 Valdas Maksimavicius. Launching Databricks at Lf. Medium, 2021. https://medium.com/if-tech/launching-databricks-at-if-819be388aa8a

S106 Piethein Strengholt. Implementing Data Mesh on Azure. Medium, 2022. https://towardsdatascience.com/implementing-data-mesh-on-azure-c01ee94306cd

S107 Dave Velante. A new era of data: a deep look at how JPMorgan Chas runs a data mesh on the AWS cloud. SiiconAngle, 2021. https://siliconangle.com/2021/07/10/new-era-data-deep-look-jpmorgan-chase-runs-data-mesh-aws-cloud/

S108 Piethein Strenholt Andrea Courtright. A financial institution scenario for data mesh. Microsoft, 2022. https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/cloud-scale-analytics/architectures/reference-architecture-data-mesh

S109 Piethein Strengholt. Data Mesh: The Balancing Act of Centralization and Decentralization. Medium, 2022. https://towardsdatascience.com/data-mesh-the-balancing-act-of-centralization-and-decentralization-f5dc0bb54bcf

S110 Jochen Christ, Larysa Vsiengeriyeva Simon Harrer. Data Mesh Architecture from an Engineering Perspective. INNOQ, 2022. https://www.datamesh-architecture.com/

S111 Piethein Strengholt. Data Domains and Data Products. Medium, 2021. https://towardsdatascience.com/data-domains-and-data-products-64cc9d28283e

S112 Piethein Strengholt. Data Domains - Where do I start?. Medium, 2021. https://towardsdatascience.com/data-domains-where-do-i-start-a6d52fef95d1

S113 Piethein Strengholt. Data Contracts - ensure robustness in your data mesh architecture. Medium, 2022. https://towardsdatascience.com/data-contracts-ensure-robustness-in-your-data-mesh-architecture-69a3c38f07db

S114 Piethein Strengholt. Master Data Management in Data Mesh. Medium, 2022. https://towardsdatascience.com/master-data-management-in-data-mesh-594d21f3ee10

Appendix A

Pre-Validation Layered Architecture

This appendix presents the original layered architecture presented to the interview participants. Their feedback was based on this version of the architecture.

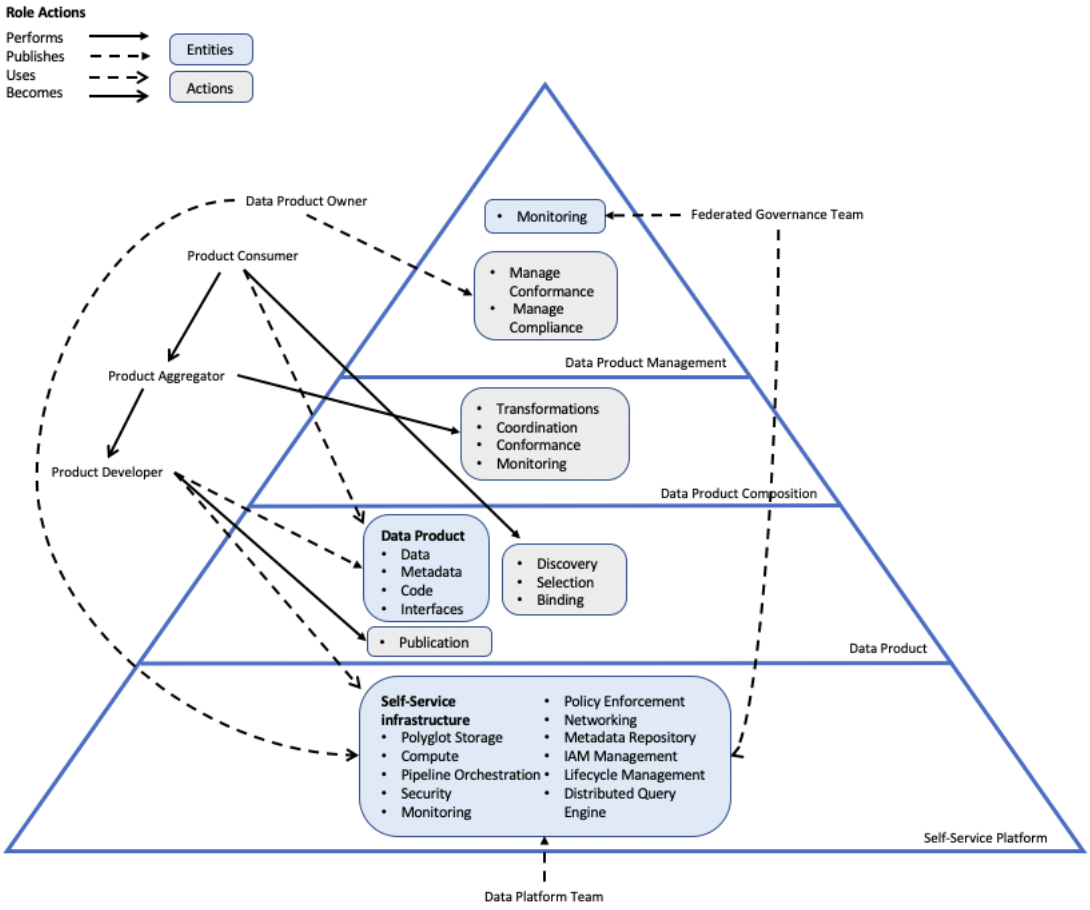

Figure A.1: The Pre-Validation Layered Architecture.

Appendix A

Pre-Validation Reference Architecture at Runtime

This appendix presents the original reference architecture at runtime presented to the interview participants. Their feedback was based on this version of the architecture.

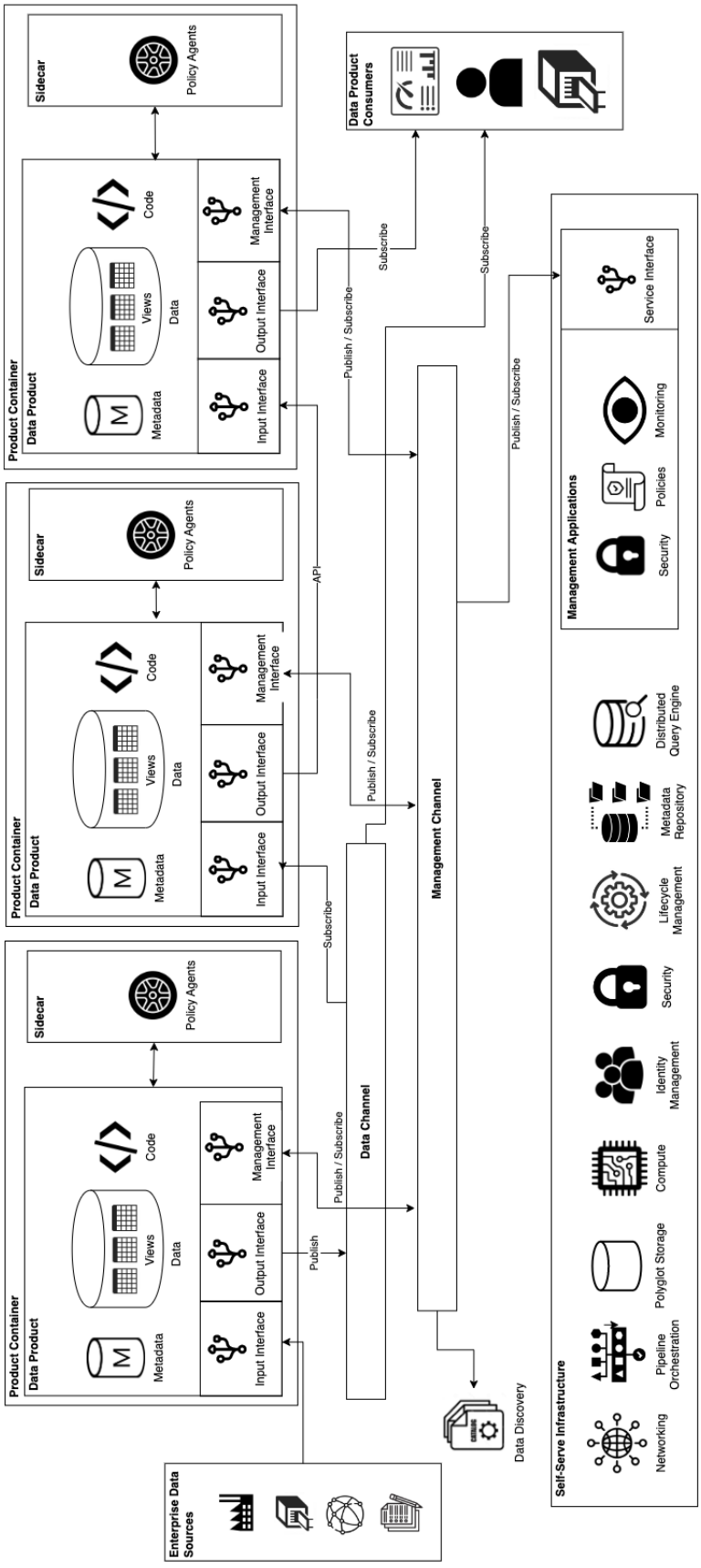

Figure A.1: The Pre-Validation Reference Architecture at Runtime.

Appendix A

Interview Transcriptions

A.1 Interview Subject A

Interviewer
So the interview is split up into two phases. The first stage is about the layered architecture that I shared with you that shows the different roles and their responsibilities and the second part is about the yeah architecture at runtime. So from your role as an architect at ¡disclosed company¿ I'm, I'm wondering if the architecture helps you in understanding the different actors and their responsibilities, and if so, how.

Interviewee
Yeah, about this layered architecture. I think the main entities are correct, but I do not really understand the role of product aggregator part because my feeling is it's feels like you are creating an intermediate Data lake with roles inside that doing the intermediate data transformation from consumer to producer, which means you are introducing extra layer of dependency. And just typical pain points that I have seen in the organizations that they try to leverage. Such an aggregator role to all, usually called a data engineer, or a data lake engineer to make sure the data can be served. Uh, from consumer from the provider to consumer, so I have doubts whether this role is fit into the data mesh principles. Because if you look at the original principles of the articles, one of the main purpose is avoiding intermediate aggregation and to try to serve or try to ingest from source rather than doing the intermediate transformations, so I'm not sure how did you come up with these roles.

Interviewer
Yeah, so we try to differentiate between what we call a basic data product that is based on, for example, raw source data and between a data product that integrates or combines data coming from other data products and we call that a composite data product. And what we recognize is that when you make a composite data product, there are some additional actions that you can perform with respect to creating a basic data product and those we try to illustrate in the in the architecture.

Interviewee
This aggregator role, because it can be the bottleneck of the of data pipeline failures, because usually there will be many consumers asking for the data from the providers and the work has to be done by this aggregator. In that case, then, which means it will have a big backlog from different teams. That requiring the domain knowledge on how the data should be mapped from source to the destination with the transformation in between. So those domain knowledge cannot be easily transferred from. Provider to consumer, uh, which means, uh, it's basically the same problem that data match has pain point out that those people has to be hyper specialized. In applying those domain knowledge into the aggregation part. So this is the potential risk that I see that can repeat the pattern of the data lake. Trying to create a central point of failure.

But I also understand your point because in real cases you need to have the aggregations in real organizations, so one of the not Outlook or one of the the potential mappings to the data match is usually. You treat those composition stage as a data product. It's equivalent equivalently to other data products, which means the data lake, like SAPHana business Enterprise Data Lake and via data products

machine Data Lake on Azure can be a data lake. Uh, there are also the data domains that basically are working on making those data available, such as as trusted datasets to be consumed by other people, but still I feel like if there is a business logics or domain knowledge involved in those, things should usually either happen on producer side or consumer side with as less as intermediate roles involved in.

Interviewer
Yeah, so the uh, the composite data product it it is published again like a data plot product on the mesh. So they take the incoming data and they perform some kind of transformation or aggregation or whatever, and and then the outcome of that is a stored in in its own data like for example and published as a data product.

Interviewee
So then the question is whether those products are consumer specific or their generic. I mean, are they required by specific consumers? How they can be consumed by multiple?

Interviewer
They can be consumed by multiple, uh? Consumers the composite data product should provide new additional value to the mesh.

Interviewee
Yeah, that's fine.

Interviewer
Uh, so let's say, uh, for my use case, I have built a, uh, predictions, for example, based on data coming from other data products. And besides my use case, these predictions can also be valuable to other consumers within the organization, and therefore I publish, my predictions as a new data product on the mesh. So it's it, the it is not that these extra steps are. Uh, performed on the request of a consumer, but it's really use case driven so to say.

Interviewee
Yeah, so my my direct feeling is this aggregator can already exist in a data product domain. Uh, it doesn't have to say it's in between data producer data consumer. It can be part of either.

Interviewer
Yeah, it it is not in in between. It is what you're saying is either so what we try to highlight with this part here is that in order to make a composite data product, you are also a product consumer. But you are also a product developer because you create your own data products. Maybe that clarifies it?

Interviewee
Yeah, so I mean, because I'm a viewer in my direct feeling is if you add it in between with the triangle layer, it gives the people the feeling that it has to be a extra layer from the top data product management towards the data products. Because if you use such model people will get the impression is it has to be a come from like composite site to the data products. So I think you can put it in the middle or in different stages.

Yeah, I'm not sure it's a triangle layer because triangle uh emphasize on the importance and priority of the sequence. Uh, because then it means that a team can has to go through the composite in stand off Going through the data products.

Interviewer
So the triangular shape is inspired by existing research on service oriented architectures and in their research they use the triangle to show the difference between a simple web service and composite web service.

We try to create an analogy between their study and our study, and I think what I can do better is explain the purpose of the triangle, because it's not per, say, about importance from top to bottom, but rather a separation of concerns.

Interviewee
Yeah, and another point is probably you can put actions in between the entities because now the actions are separate from entities because you usually use the action too. Uhm, so people take actions to do something. And here you point an action from some rules, but there is no result of the action. You know what I mean?

Interviewee
Like the actions should be from somewhere they will take actions and resulting in something that would help people to understand the sequence of the diagram as well.

Interviewer
Thank you.

Interviewer
Uhm the second question. And I think we briefly touched upon this already. Do the layers help in understanding the logical separation between simple products, composite products and managed products and their dependency on the self service platform?

Interviewee
Yeah, yeah. Let me see. For me, if you look at the original like pictures draw from data mesh, it doesn't emphasize on the importance between composite and data product. It's all free space in between. You either have the governance policies or Federated governance sitting on top supported by a self-service infra. And all the spaces in between consumers and producers of data products are sort of equal, like you do not create who has a higher importance on which they are all just intertwined with each other on the actions, so I'm not sure the triangle model would best describe the composite and product. Maybe you can put them in one box, but still can explain them inside of it.

Interviewer
Yeah, I don't use the triangular to emphasize that the top layers are more important than the bottom layers. But I tried to use this figure to uhm emphasize that every layer makes use of the preceding layers, so to say in order to realize its own layer. And it's not that the composite data product layer is more important than the basic product layer or than the self service platform, but try to show the the relation.

Interviewee
Ah okay, maybe you can highlight or explain this better in the description. The triangular shape suggests that it is about importance.

Interviewer
Then the final question about the layered architecture is does the architecture provides a complete overview of the entities, actors and their responsibilities.

Interviewee
I think the short answer is no. For me it's not clear what kind of roles is doing what kind of actions between those layers, I mean there are many arrows so I don't know what's the main purpose of showing the rows here. You should better explain the fundamental question you are trying ot answer by using the triangular shape. Explain what the shape entails. You want to show the because triangle model is upsizing on the foundation to the top. With efforts or with resources. Yeah, how the top maybe has least resources, but it has the most importance. The fundamental one probably has. It's not like it's not important, it's a foundation, but it's usually consume a lot of resources but they are on the same dimension.

But actually from my perception it's not like governance has to dictate the infra parts. Like they are supporting each other for the data products in the middle, they are both supporting role to some extent. And the main show our main actors are data products or composite products. I feel like this diagram will give a feeling in company context that the Chief data officer or data management people can dictate everything and that's usually not what data mesh is trying to say in its principles emphasizing on the both Federated governance and self serve infra.

They should support the data sharing or data exchange between different data domains, so maybe you can think about the main purpose of the triangle. Because maybe it's not the best model. Maybe you can come up with another model like similar structure but can still describe the purpose of these entities and actions in between.

Interviewer
Yeah, yeah no I understand what you are trying to say. We will take another look at the structuring of the model.

Then for the next question on the layered architecture. Does the architecture provide a complete overview of the entities, actors and responsibilities?

Interviewee
You won't be able to put everything here. Uh, just like the clear products, you have metadata, but probably you also have policies. As part of the product. It's it's a lot of things that's difficult for you to put in one page, but maybe you don't need to put everything here as that passes the main purpose of things you want to see here. So in general I think the main things are encompassed in the architecture.

Interviewer
Yeah, then we continue to the second part on the architecture at runtime. Uhm, let me see the first question is, do the elements displayed in a data product provide a complete representation of the relevant components? So then we are really talking about the product container.

Interviewee
Without business context is really difficult to say. Ah, I think as a meta structure or meta model it's good enough because you don't want to have a granularity in this type of architecture. That's restricting the agility in real business contexts, which means you do not want to specify all the rules or the details which box you should fill in when people are trying to use their reference architecture.

I think the the main purpose of the reference architecture is stating the logical dependencies. The concepts such as if you have a data product container you have. A data products and also inside are interfaces for all the interactions with different channels.

If I look back on the data mesh concepts, I think it's a good model. Uh, for sure, because you try to create some key concept in the data products such as metadata, data code as well as different interfaces. At least I can easily understand those are like fundamental elements that I can leverage to create data products. Uh, and also policy agents. They are also important.

Interviewer
So to rephrase, do you think a fundamental part is missing?

Interviewee
Yeah, so if you go back to the original like even the layered architecture you created, you have a management view. Our data governance, like centralized data governance view. And also a self serve infrastructure layer. So the governance part, I'm not sure if I can really get the sense of feeling here. How do you want to create the centralized governance capability to apply those? Data products for example.

Uh, if someone as a data scientist he wants to use the data set from somewhere, then probably he needs to go through the data discovery and then there is Federated data search across data products and then request the data set. Uh, so those kind of things are happening on top because it's not product yet, it's not requesting the data. So that part probably is not really clear from me.

But you do have the management channel yeah you yeah, the management channel could be that but you didn't put it explicitly for the top view. So probably you can enhance that with the data discovery box and also data channel box. To, uh, state out what would be the governance actions here. But and maybe they should sit on top. Uh, by leveraging your models, yeah.

Interviewer
OK. Anything else?

Interviewee
Umm so I'm impressed. I think the meta model is good enough to not put too much things here. Keep it as simple as you can, but this picture is already good enough. Just give I wouldn't call it reference architecture, I would probably the reference architecture design pattern for it match or something like that to indicate those are your research results.

Interviewer
Uh, you already briefly touched upon this, uh, but the next question is about the interfaces? Does the reference architecture clearly explain the different interfaces and their use cases and is it complete?

Interviewee
Yeah, so use those interface should come from our standardized design patterns which should be provided by the self serve infrastructure because usually you don't want those data products to develop their own interface. But the actual implementation should be customized but developed should be fulfilled by the data products.

For the patterns you I think you mentioned a few of them. Pull, push, subscribe, publish, and even soap but I was not sure soap will be still valid in the cloud context. I don't think many people will use soap anymore. Uh, they will just use pops up like Kafka.

But in the end, let's go back to our original question. I think the interface in your model is enough you don't need to create a collection of term exchange protocols or patterns, because that's not your main purpose. I think as long as you stated that they should follow the standard design supported by the self serve infra that should be sufficient.

Interviewer
Ok, if there are no more comments on to the next question. Would you use this architecture in order to instantiate a decentralized data architecture?

Interviewee
It is complicated. You have to analyze each data domains as well as business contexts, because usually the architecture is itself the drawing of the architecture is probably 5 percent of the real architecture work. The real architecture work always laying on analyzing the people, the process, and the technologies behind it and try to make sure all those things can happen in the same time and lead into that direction. The failures are usually happening on that side instant on the reference architecture side while you can have a perfect architecture. But I think the basic foundations are there to be used.

Interviewer
I get what you're saying. It's heavily context dependent. And then the second to last question. Is it clear how the self service infrastructure is related to the life cycle of a data product? So from a from starting to build a data product to publishing it to maintaining it over time and potentially decommissioning it over time as well.

Interviewee
I mean from this reference architecture, usually you don't go to that level of details on telling you the steps to create a data product and how you can decommission it. Uh if you ask people usually I don't have the direct impression on how it will be.

But since you have the service interface, the service interface should cover the lifecycle probabilities of things you are mentioning here, right? If you want to register data domain, whereas the metadata that a data domain requires. What are the service actions required? Uh, so if I'm going to use it, I will probably use the service interface that's saying these things should cover those lifecycle probabilities or request from a data domain.

Interviewer

Okay if there are no more comments, let's move on to the final question. Is it clear how policies are propagated and executed in a Federated manner?

Interviewee
Uhm, you have the policy agents as sidecar. Once again, so this is this high level reference architecture that does not go to the level of details on the policy execution and policy management. However, I'm not sure if the policies in the management application should sit into the self serve infrastructure because it depends on the policy types, whether they are infra policies or their governance policies or if they are like data confidentiality policies. It depends on the policy type.

If it's a infra policy like security policy that this data can never go to Google Cloud, I mean as a simply a simple example then this policy probably makes sense to sit in the self serve infra. But if this policy is about the data management like GDPR then probably this policy should stay somewhere in the management side. Then there is also a question of how do you collect those policies and combine them together with policy execution decision you want to execute the governance policy in parallel with infra policy or you want to do a different policy execution.

Once again, I'm not sure you want to expose such details into this picture or you want to create separate diagram just to emphasize on the policy part. Just, uh, one comment, just do not try to put everything in one picture.

Interviewer
Thank you, these were my questions. I don't know if you have any other remarks to share, and otherwise maybe we can switch to your questions.

Interviewee
Okay. I just have one main comment. Try to think about the storyline that you are trying to tell. The document you shared has three sections. First one reference architecture, second one is Azure implementations and one is used. Personally, I feel either one of them can be a standalone thesis projects already, so I'm not sure which part you are focusing on.

If you are actually trying to focus on your stages, I don't know if you already have the direction on it because those topics are very big, even in the company you know. You can really go down into details to either one of those three.

Personally, I feel like the first two parts are not really coupled with the third because with few capabilities you can solve your illustrative use case. You don't need to use all the elements in the data match to solve the use case problem because it is overkill for the use case because you the use case you presented is not, uh, enterprise use case level. You know what I mean?

So the problem is that you are doing much more work than the problem is itself, which means probably you will lose the point of focus on, uh, linking the solution back to the problem. You can mention probably 10 things that are useful and for your problem you will probably only use three or five of them. Yeah, so I don't know for you that would be valuable.

Interviewer
I understand what you are saying. Because what I try to do in the use case is to bring some of the data mesh concepts into practice, and I think I should illustrate more clearly what parts I'm trying to illustrate, in this case a simple blueprint. I will look at how I can better shape it and maybe also draft the context a little bit better off.

If there are no more remarks and questions I will end the interview. Thank you so much for your feedback.

A.2 Interview Subject B

Interviewer
So the interview is split up into 2 phases. Uhm, the first phase is about the layered architecture that I shared with you and 2nd about the architecture at runtime. I've prepared some questions and, uh please say anything that pops into your mind so, I can really get your feedback or anything else that relates to it, right? And also, try to answer the questions from your role as an architect.

So the first question is about the layered architecture, and I'm wondering if this architecture helps you in understanding the different actors and their responsibility in a data mesh architecture, and if so, how.

Interviewee
It does help. I mean obviously looking into the theory, but also into how things are executed in practice at ¡disclosed company¿. I think there is a mind shift needed in order to fully understand the role the roles described, this layout, the architecture and the actors, but so to say where there will be some discussion around the data product composition. Layer, because, in theory, you know it, it's considered to be a domain that consumes from other domains and aggregates data and produces again data that can be consumed by others. There will be discussion and I think that will lead to some discussions about additional definitions. Such as, what's a domain? And how should we identify the data product composition?

Looking at the reality, I think there are numerous examples where you can just depict a data product composition in your layered architecture. However, I think we should really be careful of what the purpose is of applying the data mesh paradignm in an organization. Also here within ¡disclosed company¿ because due to the two sided landscape we have at ¡disclosed company¿

an idea, product, product composition layer in your architecture. And however, I think we should really be careful about what's the whole purpose of applying a data matching paradigm within an organization there's like like multiple instantiations off of this layered architecture throughout the company. And the whole philosophy of the data meshes is how do we make analytics scale possible within an organization? So it's not only architectural technical view, but it's also how we organize things and. That's again, I think the, uh, the challenge I'll see within this layer architecture. How do you define your domains in a way that there is a clear understanding of the roles and responsibility within a domain, and how the domain interact with other domains in order to have a value add in the whole value chain of a company, because that's at the end, it all should all it boil down to our strategic objectives, right? So that's that's one angle.

Again, I think the architecture really helps frame the context, and it definitely helps in communicating with the multiple stakeholders and they're on different levels. And I'll tell you, the management level will understand the concept of domains and how to organize domains due to the fact that usually organize something around competencies for instance. So like there is a group responsible for alignment or overlay. So it usually boils down how you organize things around knowledge so those domains are really closely related to towards their knowledge domains. But there is no in, in practice, there's no no practice how to define, basically your total landscapes and domains so I think there a conceptual model would help you know to lay out what are the main objects that are defined within your company and how do things relate with each other? So the kind of subject area concept and that's also closely related towards the domain driven design

So again, I think you know it helps you structuring how you would like to set up your organization. So far at ¡disclosed company¿ what I see is to identify some use cases that are clearly defined around value. And to define these interfaces that need to be defined in order that somebody can consume the data in a way that's meaningful to them. So again, I think this architecture is a good example of how you can structurally approach the whole concept. Ideally, you would really build it in small steps and just make it grow over time once you have proof of value. Does that make sense?

Interviewer
Yes completely. Anything else about this or should I proceed to the next question?

Interviewee
No, I think you know the architecture is, it's a nice way of presenting it. If you look at that, for instance, the top layer at your, your data product management or also policy enforcement will take place on a more a global level. Uh, from the federation, you know, just depicted here with two double lines between the management, the top layer and the bottom layer. That I think is the hardest part of the whole data mesh how to implement this here is to be honest.

Interviewer
Can you clarify what you mean exactly?

Interviewee
So if you look at policy, for instance a data policy, every company has a more generic kind of data policy. The concept of a Federated computational governance within a data mesh, so one of the four key principles of the data mesh. Just measure the policies or the policy enforcements that are taking place on the domain level. But how do you just define local policies that adhere towards the more generic policies? How do we know that our data policies as a company adhere to those of our customers? That's basically by constantly and continuously monitoring the request towards objectives or resources. So basically, there is also a combination of a zero trust kind of architecture, various objects. There's a request that there is a policy enforcement point that you know, based on the policy definitions, uh, will execute and filter data regarding, you know the subjects that role or even on attribute based. How to get access to that? And that's really the nitty gritty details that will all need to be implemented in a domain level. On a self-service you know way because you don't want to have those domains manage their own policies by themselves, so somehow there has to be in their sidecar to what you described in your reference architecture. The way they sidecar could do for it. Uhm, it's more defining those policies on a domain level and measuring the policies according towards your global standards and policies on the top level.

Interviewer
I understand what you what you mean. OK, then the second question. Do the layers help in understanding the logical separation between basic data products, composite data products and managed data products and their dependency on the self-service platform?

Interviewee
Yes they do. And like I just said in the first you know question you asked the explaining the concept of the composite product, so that's, I think, the hardest thing to do regarding the implementation of a data mesh. Moreover, there's a lot of complexity involved in in developing a platform that you know supports the multiple domains, so basically the data product and the composite and the data product management layers, in a way where the architecture of this platform will evolve over time. And what I mean by that is that usually what you will see is that each and every domain has targets within the organization, there's always a full operational pressure on a domain in delivering stuff and doing stuff. The platform by itself is something you know that will evolve over time looking at requirements from multiple domains where the platform core team should, you know, make the right decisions. Like, what are we going to offer to what kind of you know layers on top of this, this self-service platform architecture? I think the complexity of this requirements management and how to choose the right components within your data platform, is not fully depicted over here so maybe you know the size from the layer from the individual perspective. You know just without this foundational layer, the layers on top will not function or there is a risk that they are doing platform development themselves. How to make the distinction between the core of what the whole actor on a layer like the data product layer should do versus what well the platform a self-service layer should offer a there's always a battle and that's because of time and priorities and that's something you need to manage over time and where some of the components eventually will end up in higher level layers which you should embed in the self-service architecture. But then how to decommission those kinds of functions over time and embed them within your self service platform? So there's a very interactive process going on between the platform team and all the layers.

Once the definitely the data product and data proper composition layers will be really depend on the self-service platform, how can you evolve overtime your platform capabilities. And the data product

and composition layer they don't want to wait, and that's the classical pattern you see with the these, the more centrally managed architecture like a data lake or data warehouse where there is always a dependency on a central team. And that's something, uh, well, again in this picture just doesn't really spot those so maybe that's an advice how to bring in the complexity of managing and balancing the role of a supporting a use case by the data products and composited layers? Uh, and you know, align platforms that there is a time evolution happening over there.

Interviewer
Yeah, I see what you're saying, and I I also think it's about how much freedom you as the Federated governance team want to give to your domains to build their own implementations, right?

Interviewee
Yeah, that's true. On the other hand, you know by containerizing solutions you can offer a lot of you know freedom towards your domains, but somehow the application needs to be deployed on your platform and taking into account principle we just discussed, for instance the policies, if you consider a platform that supports multiple use cases or multiple actors in your company, there has to be a multi tenant implementation. So how you make sure that for instance the domains are kind of independent, so they're autonomous, that they can just evolve over time. However, in a way, as soon as they just begin their own platform capabilities, this autonomy is a danger. It looks like on the short term. You know you're spot on and it just delivered hell, but their complexity will also grow over time and that's like I said, the balancing act between how you should, you know, I define the the priority of the components to make the self service, you know work at the at the large scale.

Interviewer
No very clear, I know what you are saying.

Interviewee
Also, when the platform is built, there's you know, uh, new innovations happening. There will be lifecycle management of all the components and that's a complexity. Also, how do you define abstractions within your architecture? That allows you to, for instance, interchange certain kind of technology components within the core platform. All the interfaces the API's, regardless or at synchronous or asynchronous, does you know abstract the implementation? It's always you know complex and it's not so easy. For instance, if you have KAFKA theory or backbone for exchanging messages across multiple domain teams. And if you do that in a multi-tenant fashion, it's not so easy to replace with another technology that's also, you know, applying towards the policies. And using for instance certificate based authentication, there's always implementation details. Even though you have to define that abstractions which are really necessary, but also are all dangerous in a way that they might impact your complexity, they might impact the latency end to end across you know the chain of services between the main components.

Interviewer
Anything else or should I continue?

Interviewee
Just continue please.

Interviewer
Yeah, OK, so the final question about the layered architecture is about completeness. So does this architecture provide a complete overview of the entities, actors and their responsibilities. And I think you already mentioned a couple things about that.

Interviewee
True and I both would also like to stipulate something new because. If you consider an enterprise like a monolith or an enterprise in organization, and the organization does what it needs to do and team that produces products and services. However, in reality organizations are networked with many other organizations. And you should also consider how to define an architecture and that also might support over organization legal entities to be part of this ecosystem. So this could be a mashup. An extreme can be that basically each and every organization has its own mesh. You can connect the meshes

to each other, but the self service platform, for instance for company A doesn't support services in company B. So that's more complex. And there I think there will be platform solutions out there.

The public cloud providers are basically an example of how to offer services that might be used by multiple companies, but using a multi-tenant implementation between legal entities, that's something different. Because even in a cloud service you have your own subscription or your own project or your own you know a virtual project clouds implementation. Regardless of what kind of definition you have from all those multiple cloud vendors, the pattern is basically the same exchange information across multiple companies. For instance, your suppliers or even customers. It takes some different approach, and even there, I think the data mesh might be a very suitable and scalable way in organizing data across multiple companies. And you haven't described this issue but I think it's very interesting to think about how a mesh might extend across multiple companies and haven't seen much in your documentation about this. It's mostly focused within a single company. But a company like ¡disclosed company¿ collaborates with undreds of partners in order to deliver services, and that's one of the things I'm working on. It's a platform strategy. So basically what's your platform and how that relates to your company strategy? And then to look how the paradigm matches across organizational boundaries.

Interviewer
OK, so then we switch over to Part 2 about the architecture at runtime. The first question is about the, the data product containers. Do the elements displayed in the data product provide a complete representation of the relevant components?

Interviewee
They do, but this architecture is really from a data product angle, right? Looking at a domain again and the role of a domain in the architecture. There's a dependency between the data product and the operational aspect or the operational components within a domain. But, looking at the metadata data, the codes and the interfaces. That's basically the frame of a of a data product, so I fully understand this here and also your idea of using sidecars for policy agents or policy enforcements. It's definitely, uh, something to consider. It's also something we are considering right now, uh, in how do you just use a kind of proxy to do your policy enforcements on the interfaces? So yes to me it's complete.

The problem however, is how would you describe the the concept of a domain, the first principle of a data mesh. Because it's a first class citizen and there is always been operational aspects within the domain. The boundary between operational and analytical is becoming more of a gray area because analytics is is used with any operation context. Machine learning models, feedback loops, feedforward loops that's all happening within the main context. And there will be a transformation processes of how do I transform the data, you know, within the operational context towards the, well data output port.

So basically your interfaces you describe are really focused on the data property part, but that is also the core of your thesis.

Interviewer
Can you give an example of how of your thought process of a data product in related in relation to the operational process?

Interviewee
So if you look for instance at ¡disclosed company¿ and where ¡components¿ are really important in the ¡product¿. There are a lot of sensors involved to make this ¡component¿ do what it needs to do and the ¡component¿ gets constant feedback from processes that measure and align. There is a feedback loop that's happening almost in real time. The software that's running that is usually embedded software, so within the system itself and data is still considered as a byproduct for reporting purpose. In my opinion. Data is becoming more and more part of this operational processes in order to base on models to steer the ¡component¿ in the right direction. So that's where you see the operational applications. Any analytical components are in a in cloud domain, so we in the domain itself, it's becoming more important that next to the use cases data is provided by an output interface. But then the question is, how will the data be represented? Because there are many different ways to do it. For

example, XML or JSON. But this might be different from the operational.

But coming back to your question, uh, yes, the components are really complete. The codes itself, however it's very abstract but you have described it.

Interviewer
Do you think that the way I described it and the text is sufficient or should I elaborate on that.

Interviewee
Maybe you should elaborate a little bit on that. I mean the patterns you described, just you know make sense and also what you defined in the basis of service oriented architectures. The patterns over there they're quite good and described in your thesis so that really makes sense.

I'm much more in favor of decoupling the different domains from each other in a asynchronous way to apply domain driven design for the autonomy of a domain. The life cycle and changes and requirements will evolve from domain to domain in different ways. So in order to support this pattern where you have different kind of patterns, I do think that a data channel is a very neat and nice way to to well to interact with different domains. However, in reality there's a lot of REST interfaces still involved to which you can implement stuff around it, but it's not in a way how really cloud native you know application can be scaled out in a way where the sole limitation is the amount of resources and money you have to spend.

Interviewer
Okay let's proceed to the next question. So would you use this reference architecture to instantiate a decentralized data mesh based architecture yourself?

Interviewee
Well, basically that's what I'm constantly doing, so I'm looking at what are the patterns for the components within the architecture. And my main interest is in providing this self service platform infrastructure. So the lowest layer. But also in talking towards the domain teams about how do you define a domain in the way that the granularity of the domain is the right granularity and what I mean by that is that you should avoid you know domains were too much complexity is hidden where the concept of microservices, for instance, is not followed up. So where the domain is becoming more a data platform on its own. And that's the kind of you know challenge you will have in any kind of domain area.

So your architecture definitely provides a mental framework, but it also provides a mean to communicate and to me this here is the most challenging thing because you have so many different kind of actors and stakeholders within a company that should understand this concept as through and through. And most of them are focused on the domain because hey, I'm autonomous, so that's fine, so I can do my stuff. I could bring in my own technologies in a container, so I'm I'm happy, but that's not how things scale and end up in a very complex thing, because they spent a lot of time in platform activities that they shouldn't do. They should focus on functionality and the core of what the domain should be. So again, yeah, yes, it definitely does help and I think you just really nailed it down.

And maybe if you look from the management channel. Maybe it's not such a management channel by itself, but that there might be multiple control planes or multiple planes within your architecture to support such patterns. So ideal channel by itsel could also be implemented with a series of technologies so that it really depends on your domain and the domains that you really support from the use case perspective. So Kafka isn't the solution for all the use cases, that's why.

You should you know, maybe consider the management channel also in a more detailed way, for instance, how do you just apply observability you know across the whole stack? Because this here is the foundation of all. How do you apply a concept like open telemetry tracing of messages and responsibility management. But I think from a reference architecture, this here is good.

I think it's more you know in describing what the components like the channel you know implies. So for me, that is also observability. I could plot in observability in this reference architecture so I

looked at this reference architecture.

Interviewer
Uhm, then the next question is about the policies. Is it clear how policies are propagated and executed in a Federated manner? You also briefly touched upon this.

Interviewee
Yes, they are. I mean, that's something we're currently implementing. By concept, they are what we call the first information broker. So the whole sidecar comes up your fly over here. It's basically you know something that you need in each and every all requests towards resources. However, you might also just think about what kind of policy languages are good examples to apply. Because a lot of the policies are seen so far, and so like exceptional like XACML, Caspin, OPA, et cetera work with allow or deny. But it's much more than allow and deny, much more. For example, complex situations that allow filtering selection based on certain kind of criteria. And then the question becomes very obvious. Then how do we manage our policies across all those big holders and it becomes much more, you know complex if the priorities are also involved. So consider intellectual property. How do you safeguard it properly?

You know it's the concept of expressing your policy in a in a declarative way like you explained for infrastructure through Terraform. But you know the fact that I'm an architect or you're an engineer or whatever role you played within organization doesn't mean that you know, once you have this role, you have access to any kind of a system. It really depends on context. It might be the location where I'm working at. Or I just might have a look at a certain kind of columns from a data set. More advanced functionality is required.

Interviewer
Ok thanks. Seeing the time, do you have any other remarks? If not we can close the interview.

Interviewee
No more remarks. In general, very good work, I'm impressed.

A.3 Interview Subject C

Interviewer
So the interview is split up into two phases. The first stage is about the layered architecture that I shared with you that shows the different roles and their responsibilities and the second part is about the yeah architecture at runtime. So from your role as a product owner at ¡disclosed company¿, as a leader of a team of data scientists, I'm wondering if the architecture helps you in understanding the different actors and their responsibilities, and if so, how.

So let's see. The first question is about the different roles and their responsibilities. Are the responsibilities of the consumer the product aggregator and the product developer and their responsibilities are they clear and complete?

Interviewee
Let me summarize them in a nutshell. Anyway, so let's start with the product consumer. I'm not exactly sure how you defined it but this is how I see it. The way I see the world is that data is being generated somewhere in some source systems The product developer in that case is the one responsible using the tooling provided by the platform team to create data products that may be used downstream by product aggregators and in the end consumers. Those definitions were quite clear and I think they also describe this as much in your paper.

Interviewer
Do you feel like there is there was something missing?

Interviewee

No, I don't think so. No, everything should fit in those three roles and responsibilities.

Interviewer
Then for the three roles for the consumer, the aggregator, and the developer role, are they relevant to your role within ¡disclosed company¿ as a product owner?

Interviewee
Definitely so these three roles are. Talking about these three roles, is something that is clearly not being done today at ¡disclosed company¿. So maybe in in my roles as a product owner, I'm responsible for delivering products right so in simplest way, it's a single product that can be anything ranging from a machine learning model which generates data or maybe a data set. If you look at these roles at ¡disclosed company¿, there is no such distinction but it can be really helpful for me as a product owner within IT. I'm, a little low in the triangle you showed me so having these clearly defined interfaces, then my responsibility is to deliver infrastructure to be used by somebody else to generate data used by someone else to make a better decision. Having the path clearly defined is something that is severely lacking in ¡disclosed company¿.

Having those three roles and especially the interfaces and responsibilities between those roles defined, would be very helpful because it also clearly marks the part that I own and influence.

Interviewer
OK, anything else about this?

Interviewee
No

Interviewer
Were already briefly talking about it, but is it clear when speaking from the data mesh perspective, what support is available to you to create managed data products?

Interviewee
If we were to have a full data mesh implemented?

Interviewer
Yes

Interviewee
In my current role, there is no support at all. So we don't embody this self service concept at all. We don't have some rules around data product creation, but that's not something which is really part of my team so. If you were to implement this data mesh architecture, I think it is clear that it's both from a technical perspective we should be supported by self serving infrastructure so we can we're not dependent on another team for the data for the creation. I think the policies and guidelines are clear, and especially who owns what. For example, schema creation. All that kind of stuff that is quite clear, quite defined, and it really helps in having it boundaries, this is what you do. This is what you deliver to other teams, so that helps.

Interviewer
Ok thanks. Then uhm, what benefits do you see to using data mesh based architecture compared to a centralized architecture and which disadvantages, if any, do you see?

Interviewee
Let me start first at the benefits. Well yeah, you have to split which means optimization. Every team does just one thing instead of trying to do everything everywhere. There are teams providing the tools, you have a team using the tools to create data products, you have other teams using combining or aggregating the data products and you have consumers which are just consuming.

The upside is everything does just one thing, or maybe a couple things, but it's very scoped and limited to doing one thing very well and you enable other teams to create the data products close to

where the data is. Because now it happens in a centralized fashion. Everything is created on one central platform, which means governance, control are slow and people don't have the freedom to create these data points closely with data is being generated, which means inefficiency.

The biggest upside I see is that by splitting these rules and giving and pushing the creation of these data points as close to where the data is being generated, you enable the experts close to where the data comes from to generate and create the products directly.

The biggest downside is on the screen. The model is a little bit more complicated than a centralized architecture because centralized architecture, especially centralized database, it's easier to govern and control. It's easier to implement policies. It's easier to onboard teams. It's easier to manage. Might be more cost efficient.

And the big problem is trust in this case. How is the consumer going to trust the aggregated product developer? There needs to be a lot in place to make this work. Basically, a decentralized approach to doing analytics and distribution of data. And interfaces means complication, means expensive means having a very strong vision in order to make it happen.

Interviewer
OK. Any anything else?

Interviewee
Let me just share my vision from my role as a product owner. What happens now at ¡disclosed company¿ is that we are very centralized. We have business and IT and if business needs something it goes to IT to have it built. I don't think with all the growth and need to be agile, especially in the domain of data and analytics, that a centralized model will hold because you cannot just add more and more people to the same platform and more and more people and more and more consumers because the model in itself does not scale. But the model in itself, having one centralized team responsible for infrastructure for all data is not going to scale with the size of ¡disclosed company¿, so you need to split it up.

So I do recognize all the benefits part of data mesh to make it faster, ownership close to the data and the dream for full self-service. I see the benefits but it is very complicated to get there. I would be very much interested to see and to know how this will work in practice because there is a lot of risks in all these new services, all these new concepts, all these new interfaces, all these new processes. It's 180 degrees different than what we used to do before and that's going to be very complicated and very complex to implement.

Interviewer
OK. If nothing else we go on to part 2 about the reference architecture at runtime. The first question is if the reference architecture makes it clear how you can build a data product and which components need to be included?

Interviewee
Yeah, so everything is clear except for the sidecar what it is used for.

Interviewer
Ok. The sidecar is used as a separate but connected container for policy execution. It is built by the platform team, shares resources with the data product and executes policies.

Interviewee
Okay then everything is clear.

Interviewer
Ok. Does the reference architecture describe which components are your responsibility to implement as a product owner and which are not?

Interviewee

But then I let's, let's assume that my team is we're neither consumer nor developer, so I think my team best fits with the aggregator role because we take existing datasets and transform them, for example using machine learning to create a new data set, so let's assume for now that that's our role and than it is clear.

Interviewer
Can you elaborate by explaining which are your responsibilities?

Interviewee
In order to create a new data product, we need to have the full self serve infrastructure which is the bottom layer here and also in the other diagram. We take that as a service. The management applications, of course are a different layer, which are also maybe less visible to us, but which should also definitely be there for us. So let's assume we work in one of these product containers, we are presumably subscribed to the data product of a different team which is created upstream which can be anything, dataset X, Y or Z or whatever. We are responsible for transforming, we fully own the container and all the logic which is happening there,

All the transformations the schema the tools which are which we use in order to create a new data set. So in this case it's a new machine learning model which is published, I believe, through the output interface. Yeah to the data channel. Again, as a aggregated data set so it's dependent on the upstream data set. It's entire flow from input to output is ours. And the channels are the responsibility of the platform team. We publish the information to the data channel.

Interviewer
OK. And then the final question. Is it clear how you can manage data products throughout their lifecycle?

Interviewee
From this picture. Or in general?

Interviewer
Yes, so how the tools and communication patterns enable you to manage your data products.

Interviewee
Yeah, so in this case, let's say a large part in it happens in one of these product containers, right? So we start off with a couple of experiments. Then at some point we push the machine learning model to production. And you generate a data which you deliver as a service to another team. Afterwards, it might decay overtime some point you want to pull it back. I would assume that this life cycle. And the tools which I need to manage this life cycle and to monitor this life cycle are delivered from the self serving infrastructure. Which means that we don't need the knowledge of the underlying complicated components needed to execute that lifecycle we are just responsible for the code.

Interviewer
OK, and maybe one follow up question if you can explain, let's say that you have a data product, how would you monitor that data product and what would you monitor?

Interviewee
I would monitor usage. But I'm not sure how I would do this. I presume I get monitoring as a service, so that there's a tool or platform which we can use. Again, that we don't have to create our own monitoring service. So the first assumption is that there is a service where we can write data to or whatever. But at least we we start from the code. And we monitor usage, who uses it? How often is used? What the load is on our on our endpoints. Whether it's being misused, whether correct identity and access management is implemented. Yeah, those kinds of things.

Interviewer
Yes very clear. Thank you. If there are no more remarks from you this marks the end of the interview.

A.4 Interview Subject D

Interviewer
So the interview is split up into 2 phases. The first phase is about the layered architecture that I shared with you and 2nd about the architecture at runtime. I've prepared some questions and please say anything that pops into your mind so, I can really get your feedback. Also, try to answer the questions from your role as a governance architect.

So the first question is about the layered architecture, and I'm wondering if this architecture helps you in understanding the different actors and their responsibility with regards to governance in a data mesh architecture, and if so, how.

Interviewee
Yeah, so for me it does. Obviously, it doesn't hold the all of the details but that's I, I think a on a high level perspective it does. For me, there are definitely, if I look at it from a data governance metadata management perspective, there are definitely a number of interesting challenges to address right? Because while I know from data mesh, this layered architecture and this layered governance, which is also somewhat well reflected in this in this diagram, it opens up a number of challenges here, so, uh, if you talk about the Federated Governance team. That would imply decentralized governance related teams and centralized governance team. You would have to balance out what are we going to organize on the central level and what are we going to organize on the decentral level. But in fact, that doesn't only apply to the governance of it all. It also applies to the actual data, right? Because there will still be data that you will probably want to have centrally available based on a central data pipeline. But the general idea is that a lot of data will be available in a more agile fashion in a more decentralized way.

Interviewer
Can you get can you give some examples of data that that you think should be available on the centralized?

Interviewer
Uh, yeah, through centralized platform.

Interviewee
Yeah, I think, Uh, for instance, state of the business, right? So financial reporting it is probably, or not, probably state of the business is something that senior management uses and the entire company is really interested in. So consumers will be senior managers whether they work within corporate real estate within DE or in CS. It doesn't really matter. Everyone wants to know about the state of the business. So to me it would make sense to have those let's say financial, periodic financial reports organized in a more central fashion. You could argue that this is something that we should be organized from the finance domain. On the one hand, that's true. On the other hand, as you have a lot of dependencies towards other sectors within the organization, I think it would make more sense to organize this information provisioning in a more centralized management manner. And then because you would are very likely consume data from a lot of different data source to come to such a state of the business. This would, in my opinion, still be something to be organized in a more central manner, but also governed in a more central manner because you have stakeholders throughout the entire organization.

Interviewer
So if I understand it correctly, it's due to the high complexity that creates a higher risk of failure as well, which of course cannot occur.

Interviewee
High complexity, really diverse in in terms of stakeholders, so lots of dependencies to move towards. Lots of stakeholders working in a lot of different departments and indeed also complexity in that sense that thedata will probably be derived from a number of different data sources, but you would want to govern this in a in a more central manner because you cannot just based on requirements from the one domain, a change a certain transformation without reaching consensus with all of the other domains

who are relying on these financial reporting's to actually enable them to do the right decision making within the company.

Interviewer
OK, then I think we should continue to question 2 because it's related to something you already mentioned about the distinction between the global governance and local governance. So does the architecture clearly explain the difference between global and local governance and the separation of responsibilities?

Interviewee
Well, you're stating here that the Federated governance team has a job in monitoring at all Ievels. I think that, uh, this is really dependent on the type of organization. I think for instance, in in banking, the Federated governance team probably holds a lot more responsibilities. And also within ¡disclosed company¿, I think the Federated governance team shouldn't only be monitoring, but they should also let's say, raise the bar in terms of certain guidelines that apply because those guidelines will apply to all of the data right, not restricted to a single domain. Anything related to data management that would apply to all of the data that we are using in the company I would say is something we should organize in a more central manner.

And and then you obviously need to create a clear distinction between what do we want to organize in a central manner and what do we allow within the organization to be organized in a more decentralized manner. And so that's I believe, a precondition. So you first need to determine those requirements and those boundaries, and then the central guidelines will be orchestrated from the central Federated governance team.

Interviewer
Okay. Then are the other governance goals clearly explained?

Interviewee
Uhm no. I think, uh, you should include more so it's not only monitoring, it's also setting the boundaries, creating the guidelines, the guiding principles and also ensuring that certain non-functional requirements from a governance perspective are met. So I think it's about monitoring. It's about constraints. It's about guiding principles. I think that's the main thing other than that, but that's more general remark.

A certain chief data officer in a certain organization because this Federated governance team, I would say also a certain link towards the Chief Data office right in terms of organizational structure. I think it's not a bad thing for a CDO office to be actually involved. Also, in the deployment of the of such guiding principles and that's why I do like the term monitoring here as a as a verb, because that would imply that also the Federated governance team would have a job in actually monitoring the extent to which the rest of the organization adheres to the governance goals, right? And that ensures this Federated governance team, whether it's the central team or the local governance team, still understands the actual data and holds a certain feeling with the actual data that we are that we are working with in this company. We don't want it to become an ivory tower, right?

Interviewer
Yeah, I understand. Anything else about the government goals?

Interviewee
Well, yeah, you're mentioning the data product data, metadata, code and interfaces. If you talk about code you are actually talking about software code, algorithms, et cetera, right?

Interviewer
In the broadest sense, so it is indeed code to do any data transformations, but also policies written as code, etcetera. Pretty much everything that needs to be managed in a computational manner that's incorporated as code.

Interviewee

Yeah, the question is whether you would, that's more on a detailed level, for instance a metadata repository. A metadata repository can be an access path to the data, right? It holds a data catalog. The data catalog itself is the access path to the data. So therefore I understand why you would include it in the self-service in infrastructure? You could also say, oh wait, you're also mentioning it in the data product. Indeed, you're mentioning metadata also in the data product so I think the metadata repository is somewhere in between the data product and the self-service platform, right? The data product would be maybe the data catalog and then the self service platform would be the metadata management solution as a whole with the business context added to it, et cetera, et cetera.

Interviewer
Yeah, exactly so maybe I should clarify that in the text a bit more?

Interviewee
Yes maybe that would be good.

Interviewer
OK then, for the next question, is it clear what support is available to govern the data mesh architecture and is the support complete?

Interviewee
I think from this this architectural diagram yes. Yes, the question would be and I think this is actually in practice also a struggle with in ¡disclosed company¿ you're mentioning different roles but I don't see any architectural role mentioned here in the entire architecture. So I believe that a data mesh approach still requires us to have mature data architecture expertise available in in the company. I think maybe even more than if you would organize the production and consumption of data in a more centralized manner. So you still need to also, in a decentralized or Federated concept still have architectural guidance in place.

Interviewer
Can you explain what you think should be their responsibilities?

Interviewee
That responsibility would be to, for instance, ensure that the most important thing is a long-term sustainability, right? And anything attached to it is related to long term sustainability. So certain guidelines from an architectural perspective that would or must apply for the technical implementation of data, right? So architectural guidelines on the technical implementation, but also a decision making tree. For instance, a really concrete example, and this is happening in in dozens of places, dozens of teams every day. A new request comes in. People want a certain data to be made available to them. The data already exists, but not exactly as that person at that moment is asking for it right? So it's maybe there is already an existing data set and it's 80 percent compatible with the current ask. OK, what are we then going to do? Are we going to create a completely new data set that's for the preferred approach, because he is not bothered by the existing data set at all, right? He can just create his own party, so to speak. Or are we going to enhance the existing data set which is 80 percent compatible to become 100 percent compatible? But that creates a lot of dependencies, because obviously other people are relying on the existing data set, right? So you need a certain handshake with data owners. You need to include additional transformations, those need to be aligned between previous requestor and current requested so a lot of governance and a lot of handshakes are required to in the end reach the goal of re-usability.

So are we going to really ensure re usability of our data. Or are we going to again create all of those silos? The truth lies somewhere in between, right? Because you want it to all be flexible and you want it to be really widely available. On the other hand, you want you also want to reuse the existing data as much as possible. For instance, to ensure that people all speak the same language within the company. And that's something that the architects should really help balance out and make the right decisions in, because for a certain use case it would be best to improve an existing data set but for another use case it probably would be best to just start all over again, right? And who's making those decisions? Who has the mandate to determine when to do what?

Interviewer
Good very interesting. Anything else about the available support?

Interviewee
Uh, no other than the role of the architect and the role of governance centralized decision making versus decentralized decision making. I think those are all important attention points to address in a data mesh type architecture concept, yeah?

Interviewer
Ok and would you be willing to adopt aFederated governance model?

Interviewee
That's actually what we as a CDO office already have as a vision. So what we actually foresee and in practice to an extent already have it, is indeed a Federated structure so we would have one central chief data office within ¡disclosed company¿ which we have established around ¡disclosed information¿. And then we would have, let's say, certain satellites within the organization, which to an extent we already have, for instance. Within ¡department¿ there is a team of people 10 to 15 people who are responsible for data governance around ¡specified¿ data. And they're actually sort of a satellite department of the Chief Data Office, but they have a specific focus being the ¡specified¿ data. ¡Disclosed information¿ and also within the corporate world, there are different decentralized teams responsible for data governance which have a link towards the central team and they are relying also on central guidance from the Chief Data officer. So actually this is already the vision of the CEO office. Our Federated governance nicely aligns with this data mesh concept.

Interviewer
Yeah, can you give some examples of the responsibilities within ¡disclosed company¿ that are for CDO and which are for these satellites?

Interviewee
And, for instance, determining the global roles for data management is a centralized exercise now, right? So the chief Data Office has formalized the global roles for data management we need within this this company with detailed descriptions, tasks, responsibilities, et cetera, et cetera. But, for instance, also a data principle. So the principles that data needs to adhere to are determined by the chief data Office. So the chief data officer sort of determines some of the fundamentals, right? They're setting the foundation and for instance, people who are working in the finance department hold responsibilities related to data governance activities. They're also asking chief data officer: explain us how to determine the ownership for data, right? So please establish a process so we understand how we can determine who is who, who you are advocating for to become the owner of data. Because for me, data governance, in short, is formalization of data management activities. How we use and how we produce and consume data within the company.

And this all starts with ownerships. It's all about ownership. That's really the first step, because once people feel responsible for the data then they also are open to understand what needs to happen next, right? And then the interesting part here is that once people really understand why it's important to organize, establish ownership on data, then people actually also are quite willing to do so. Once you're an owner of certain data it gives you power. And once people start to understand that ownership of data gives you a certain mandate within the organization then you will notice that from that moment on you will have plenty of volunteers when you're asking someone to, uh, to become the owner of certain data. But that's not something we are at. We haven't reached that maturity level yet.

Interviewer
OK, yeah because that's interesting. I think a large part of data mesh governance is also what are the incentives in the organization to govern your data? But also to make it available do you have incentives in place or think about that?

Interviewee
Yeah, this is an incentive that you wouldn't really, use in your marketing strategy, right? But in fact it is true. So another incentive would be cost reduction but cost reduction in itself is really interesting,

but people then immediately want to know how much cost and where are those reductions then coming from? And that's not that easy to then immediately just write down. If you do this, if you go left then you will save 50K next year, right? It isn't that straightforward. Because in my personal opinion, data governance largely is about hidden costs.

Of course we have even at this moment, someone giving a colleague a call, asking where can I find this data? This is what what's happening every minute, every hour in this company, every day. So we have a lot of people who are in need of data and we have a few people who know where to get the data. So there's a, uh, big this disconnect between or knowledge gap. I would actually say between the people that know where to get certain data and the people that need the data. The latter group is a lot bigger than the first one. Obviously that's also because of the fact that ¡disclosed company¿ has grown so fast. Right, so if you're a new, if you're a newbie in this company then you need information. You're in need of data and then you will soon find out that we have a huge gap in terms of information provisioning, information provisioning as a whole, but definitely also, specifically related to data. So what people do is they actually start to create their own network of people, of whom they know that are knowledgeable and that they can get the right information from. So we now have all kinds of networks of people connecting together asking each other about where to get certain data or where to get certain information. And those so all those network pipelines of people who are connected together tend to break when someone, for instance, leaves the company. Then the entire pipeline needs to be rebuilt again to fill in that that knowledge gap.

So we, we spend I don't know how many hours per week bridging those knowledge gaps due to a lack of data governance. But it's it's a lot about hidden costs. It's about re usability of the data which I mentioned before. It's about cost reduction. It's about efficiency and it's about just a general knowledge gap that we have within this company.

Interviewer
OK. Then I think we've answered all the questions about the layered architecture. Let's proceed through the architecture at runtime. So from a governance perspective, does the reference architecture give you sufficient control?

Interviewee
Yeah, the question would be how we would, for instance relate the self service infrastructure towards the product type containers right. So for instance we would have a centralized metadata repository and we would have a product or data product related metadata repository, how would those interact with each other? So this is an example of the metadata. Another example is indeed the input and the output interfaces in the product containers of the data product. How are those managed towards security and identity management from a self service perspective, right? So that's I think a layer on top of this architecture that you would need to orchestrate, right? So what do we do? What do we do in a more centralized manner and how do we, let's say, organize that all together? Because if you are not allowed to, from a self service perspective, access certain data then I would say you're also not allowed to access the input interface of that of that data product. Right, so how do we ensure that, or is the self service infrastructure by default the only access path to the data, and if so, how do we organize that then? How do we ensure that the end of input interface is not accessible by people who have the ability to access the self service infrastructure right. So those are certain dimensions that you, I believe, need to bear in mind.

Interviewer
Okay, anything else?

Interviewee
Yeah, in the particular example of ¡disclosed company¿, a reference architecture should probably on the left hand side not only go into the enterprise data sources but also into the external data sources, right? Uh, so we're actually also consuming a lot of data from our products in the field, right? But also data from from external partners, vendors, suppliers, etc. So, and I believe that this data mesh concept can also provide a solution here right?

Interviewer

Yeah, it's interesting that you mentioned that. Then for the next question, is there any redundant functionality with respect to governance?

Interviewee
The easy answer would be no. I don't believe there's any redundancy if we all implement it in the right way. OK, because you could say, but why do I have a metadata repository in my data product as well as a centralized metadata repository? At first glance, it looks like it's redundant, but it's not and if I am able to subscribe to an input interface, how does that relate to identity management and security in the self service infrastructure? Is there some redundancy there too? Because there are probably boundaries constraints to be being able to subscribe to such input interfaces right from a from a security perspective. So I think, there is a risk of redundancy, but it's up to the right authorities within the organization to ensure that that we don't have that redundancy, and that's all a matter of yeah, determining the right constraints, right?

Interviewer
OK. And is it clear how this architecture enables the Federated governance model?

Interviewee
No, not enough. Yeah, because I would expect an additional slide that, for instance, covers the orchestration between the decentralized metadata repository and the centralized metadata repository, and for instance, a clear distinction between which capabilities we use in the central manner and which capabilities we are using in a decentralized manner. Right so no, it's not fully detailed out, but I do understand the the let's say the conceptual diagram as you have written it down here. Because I think it really aligns with the way that we should look at data mesh within the company.

Interviewer
Ok, and then lastly. Is it clear how the architecture enables self describing data products and easy data discovery?

Interviewee
From this slide I would say yes. I think that especially the people that are interested in in datasets that are a rich combination of data coming from a lot of different sources they will probably want to know. How will this be organized in a data mesh concept? And that's actually also going into one of my previous comments right. Do we still need a central data lake in the data mesh world and if so what are we using it for? That's another angle to the same topic. How are we going to ensure that we still do our state of the business financial reporting, which is actually data coming from a lot of different data sources. How are we still going to organize this? Or how do we still keep this alive in a data mesh world and yeah, it turns out that for instance that ¡a data set¿ is used by ¡department¿.

Maybe the data owner of this data, but reporting and analysis on material master related data is being performed throughout the entire organization. So how are we going to still manage this? In the data match rules and how are we going to still ensure that the Finance Department interprets the data in the same manner as supply chain.

So yeah, if we are letting go of this centralized approach, that will be the effects of this. Because what I read from data mesh is that you would still have, let's say a loosely coupled architecture from within the different data products. Because if the one data product uses A and the other data product also uses A and you still want to ensure a common understanding between the first A and the other A. Should you need to still ensure within the organization that A remains A also within the other department, right? And not that the other department considers A to be a B.

Interviewer
Yeah, so the overarching terminology, right?

Interviewee
You mean terminology for data which is used, cross departmental, right? So still A and that would be in my opinion, this would be an example of what you would want to establish by the means of the the metadata repository, right? So you would use the centralized metadata repository to form a more

holistic approach, establish business context, a business glossary towards your metadata, and that you still need to ensure a handshake. In the example I just mentioned between finance and sourcing and supply chain that they at least both adhere to the same to the same business context, and that they accept the data ownership for instance.

Interviewer
No, I completely understand and agree. Anything else you would like to share because these were all the questions.

Interviewee
No nothing more.

Interviewer
Ok thanks for the interview.

A.5 Interview Subject E

Interviewer
OK, so the interview is split up into two phases. The first phase is about the layered architecture and the second phase about the architecture at runtime. Please say anything that pops into your mind, just throw it out there. And so, the first question is, does the layered architecture help you in understanding the different actors and their responsibilities, and if so, how or how not?

Interviewee
Yeah, when you first look at it and that's always the risk of using a picture like this, it seems like there's some directionality and some hierarchy which might not be accurate because for many actors in this field don't care about all of the other layers, they just want a product. So if you're a consumer, all you care about is a product we're going to get and how can I use for what is in it? How can I subscribe to it so from a consumer perspective?

But the figure is a bit much and so it could be simplified and made more role specific because it's trying to push a little bit too much into one picture with all of the four different arrows and the different colors but if you if you have some explanation with it starts to make more sense.

The other thing that I that I'm not a fan of is the data products composition layer. There's two reasons, one for simplicity, it's much easier to say any product that is derived from another product is just a product, so it doesn't matter if it comes from one source or ten sources or from other data products or directly from source systems. A data product is data products. And two more philosophically. The whole goal of the data product is that the understanding and the ownership of the data is in the same team as the team that is supplying the data part and a composition product as you as you call it? Kind of goes against that model because suddenly the guys who say, oh we have a nice data product. We combine five other data products they don't actually own that data. They don't know what changes are going to come, or they might have to do a lot of coordination with the underlying data product that it's. That's a potential bottleneck that starts to appear within this this picture.

And that's always hard, because one of the reasons you want data products is to be able to build on top of it and then to build more and more meaningful abstractions. But it's always good to keep that in mind that. How do you deal with those in the real world.

Interviewer
Do you have a resolution for that? I, I think it's also one of the main concerns right that once data leaves your own data product, unless you really have a sophisticated computational governance model, it's very challenging to make sure that the data is used in line with its intended purposes, et cetera.

Interviewee
Yeah, so for me that's the number one consideration when doing a data mesh is that there basically

is a scale in terms of governance that on the one side you don't have to aim for and you can go fully governed, which means that any data product is read only once by central team and they supply it out to other teams and they keep full 100 percent tracking of. This data product is used by these five other teams et cetera. Versus a much more loose mesh where one product and just for one team or one business actor or whoever could just pull from a data product and there's no centralized governance.

So who is using what for what purpose and things like that? And that's choice. It's really foundational and really specific to an organization because it needs to match the kind of risk appetite compliance environment. There's a lot of things that come into play into that decision, but that decision is key in how do we deal with composites? How do we deal with governance in general with monitoring, general and ultimately that has a big impact on how agile your mesh becomes with the more governance you do, the less agile it becomes at some point.

Interviewer
Good anything else about this question?

Interviewee
No I think the rest of the roles are clear. And then it's up to. Up to how you organize it, if you have a single team that does both governance and supply the platform, or if it's two separate teams, but that's all. Roles are a good way to define this.

Interviewer
OK then secondly, do the layers help in understanding the logical separation between simple products, composite products and managed products and their dependency on these self service platform?

Interviewee
Uh, yeah, like I said, for me it doesn't because for an end user, it doesn't matter if it's a composite or a simple product. So I don't care. Yeah, I need to know what it is? How I can access it? For this both, both from a technical perspective and from a business perspective. And then I just want to use it so you know if it's a composite or not, I don't care.

You know, I think in general. It would help to have kind of a consumption picture and a production picture, because then you end up with two much clearer pictures with the data product itself in the middle as the link between them.

Interviewer
And then, uh, does the architecture provide a complete overview of the entities, actors and their responsibilities, or is something missing?

Interviewee
Yeah, so what's missing for me is, and I don't know if you can ever really capture that in one picture, but that that choice between how much governance do we have? How do we allow teams to just directly go to the data, products partner teams? Or do we have a layer between that? That isn't captured here. And you can go both ways with this picture. Which isn't necessarily bad, but I think it's good to be aware.

Interviewer
Anything else that you feel like is missing. No, I think there was all of the content of the boxes make sense. So I think that's great.

Interviewer
No good. And then to the architecture at runtime. You already briefly touched on this, but do the elements displayed in a data product provide a complete representation of all the relevant components?

Interviewee
Yeah, I think in general. The important pieces are there and just looking for something right. Let's see. Yeah, if he if he goal here is to actually that actual data products are hosted in containers. I would have some doubts about that. I'd be good in terms of you know, viewing them in a way that makes

sense,you know, putting them on a picture in a way that makes them really self contained makes sense, but actually running in a container is going to lead some problems down the road. But that's more technology discussion.

Yeah, the way I look at it is that there's one thing really missing. You already have a bit of push and pull, but the other thing that's that comes into play for me is is basically event based versus batch. Where event based is potentially real time but at least near real time. That can be pushed with something like some kind of bus where messages are put on and then it's subscribe philosophy. I've seen that as well where other products actually just an API and whoever wants to use that product. And that's really you've had base, so they go changes almost at the role level versus batches that are made available and that can be queried which is a very different paradigm. And then you also have variations where a event is created when a new set was available for like the last day or so that so there you have a kind of hybrid between that or you have like a metadata event versus the actual data.

Interviewer
So I've a follow up question about this because in in the report actually I described this that the different views can be different time based groupings and one of the discussions that I had with my supervisor is that whether the batch size so to say, so the amount of time you group by, should that be part of the decision of the product owner or what my supervisor said is you can use an enterprise service bus that allows for that, is able to group the data so consumers can sort of request the service bus to make an aggregation. Based on their needs what are your thoughts on this?

Interviewee
To decide what is what, batch acceptance and what batch can our system handle because if you have a source system that is not very performant, but you still want to get the data out, it might be something where you just say OK, we make one dump per day. And that becomes our batch that we supply to the rest of the organization, while other systems might have like a change data capture feature where you can just have every transaction that runs through it, make that available directly to any user.

So that is for me really dependent on the source system and therefore the data product team needs to decide that. Of course the presumption they might say we will only read it once a day, even though updates come in every year, every year second. But then the source I think it's a good practice to always say. We're going to go as low level as possible, so as smaller batch as possible. But there's a limit and that becomes our lower limit and then consumers can choose whatever they want higher than that limit.

Interviewer
Yeah, yeah, it's also a trade off right between costs and if it's going to be used so to say.

Interviewee
Yeah, especially well like for minute batches or up, cost is not that big of a consideration, but if you really want to go to sub second then cost becomes a very big thing because then they need much more compute online to be able to run. That becomes expensive.

Interviewer
Yeah, OK. Anything else about the data products or?

Interviewee
Yeah, I think similar to the previous picture. The devil's in the in the details of how do you deal with a product that has a new schema and how to know what happens with that schema changed? Do they announce that in some way to all of their users or do they have to register that change to central governing body depending again on what kind of governance do we have in this service mesh. Uh, or do they just have a schema number or some kind of schema reference in their API's and then once they switch over to the new API that then it's just up to the consumers to start building from the new API and deal with the schema change themselves, or do they just change the schema and good luck?

There's obviously some big questions to answer there but not all of those are bad per say. It's a choice of how you want to do that and how and how closely do you want to coordinate between pro-

ducers and consumers? And that again comes into how agile do you want to be? Kind of preference, but there is also a ton of reasons to deviate from that. Is that you just say as part of the thing that we make available is the schema and whenever we switch to the new schema that gets updated and then all feed stops, the new feed starts and then it's up to the users to deal with that schema change in a way that makes sense for their business. So, that's the way to do it, I think. But from a governance perspective that has serious implications in how do you communicate that change? But for me the importance of keeping data producers agile is worth the effort of dealing with consumers having to make a change.

Interviewer
But are you not afraid that consumers are then reluctant to make use of data products they know that from one day to the other, suddenly the schema can change and their own application can fail. And especially in business critical applications they might be reluctant to make use of the data products.

Interviewee
This is exactly why the the communication between producers and consumers is so so key and how do you do that? And you can do that on kind of a business level? Because a schema change should be coming from the application behind a product being updated to support something new that the business wants. Otherwise, it's schema change shouldn't be visible in the daily output. And I mean, that's kind of the whole idea of making products that you have some stability in the way you decide that if you make back end changes, for example if you switch to a new database technology for the product or in the underlying application, then you should be able to supply the same data product without changes.

Interviewer
Yeah, the schema should be business driven, right?

Interviewee
Yes that's true. So how how they go about making a change to their data product? And then how do you communicate the change and how would the receiving side adopt those changes? That's really crucial thing in the data mesh and. But it's something that you need to be careful about, because what you want to avoid is that at some point, a data product can no longer make changes because there are 10 people behind it who say, yeah, this place is critical for us that we don't have time to make change and they all escalate to the CIO or to whoever. And then suddenly this data product can no longer innovate, and the underlying location can no longer change. I mean. That's the risk that you run if you don't allow schema changes to be pushed rather than to be agreed upon fully, but there are nuances there, and there are ways of dealing with that.

So for instance you could say we're going to have a preview version of the API which is going to have a schema change and will backfill the old schema in the old schema feed basically for the next five months. And in five months they will all those consumers need to switch out to the new one and if they don't, then they are responsible. So there are ways to deal with that, but not all schema changes can do that because if you delete columns in your source, then it might not make sense to backfill those columns with nonsense, things might still break downstream, and you know, as a data product team you don't want to know how they use that column, so you shouldn't care what's. Yeah, it's at some point you shouldn't have to worry about what is in there if you if you leave it there or how it's used if you delete it. Additions to the schema are usually a bit easier to do it.

Interviewer
Yeah, OK, seeing the time I will continue to the next question. Yeah, you already said something about the interface, so please touch upon that again, but does he reference architecture clearly explain the different interfaces they use cases and this is complete.

Interviewee
The input interface, for me, it's, uh, it's fully up to the data product team to decide how they ingest their data from their sources. So you can call it an input interface, but really it's whatever they use to to read that data from the source. And I don't really care. Only from an architecture perspective, if they want to use a query that reads it and stores it in a in a separate database, you know that's a perfectly valid interface for some products. If they want to read an API from the application, that's also fine. If

they want to completely sidestep the underlying database but talk directly to the application.

So obviously you need to have it, but it's really up to the individual data products to decide. What does that look like for them? And they need to be able to change that because their application might change So at some point they might completely throw out what they have and rebuild scratch. You can completely different technology and that doesn't change the data products. So that's only their input interfaces.

So for the output interfaces, yeah, we already talked a little bit about. You need to have push or pull and then you have batch versus events and any combination of that. So you basically have a matrix of different output interfaces. Uhm within the added pull variant of being able to query or just getting a complete set.

Interviewer
What does pull based event driven look like?

Interviewee
So for instance, and this is a bit of a technology, but in Azure we have a service called Event Hub and this is built just as an application, you push your messages, your events to that event, and anybody else can decide to subscribe to that. So it's a bit of a variation on who initiates meeting. So for pull, as you state it right, it is pushed to the bus and the events are retained for certain number of time and you as a consumer decide, ok I'm going to read at the moment that you want it. Which might be once a day or just in the moment.

Interviewer
OK. Would you use this reference architecture to instantiate a decentralized data mesh architecture?

Interviewee
Yeah, so the big question then is those product containers that are that I mentioned. I like them visually as kind of, you know a product is just this thing that has its own abstraction, but if that's actual docker containers like containers, there are some concerns.

Interviewer
No, it's it's a like you said to indicate that they are self contained.

Interviewee
Then that's fine. Some different decisions than, yeah, we come back to, and, well, it's really hard to capture these pictures so I don't blame you for that. How the governance is done and as you move to more government than you probably will also have more standardization of a lot of the components.

So for instance we have four different technologies that are being decided by the centralized team about your output interface or a specific schema for dimension interface. Well, if you move to a much more loose, then it might be up to each individual domain to decide this is the interface that will double supplier data, which again has profound implications on how agile it is. It's partly because one product you might say, oh, I would. I just want to make a sequel database endpoint available. Because then I don't have to do any work and that's it, while other more governed mesh you might say, everything is done as an API. That's a big big difference in how this picture is actually implemented.

Interviewer
Yeah, yeah, so I think. The literature, like you mentioned, recognizes different topologies, right? And this is really for the governed mesh, so quite strictly governed, I would say. But I completely agree.

Interviewee
Well, almost because you do have the data product consumer reading directly from an output interface. And if you have a fully governed mesh then that also wouldn't because then it would also go always go through the central governance layer

Interviewer

Is it clear how the self service infrastructure is related to the life cycle of a data product.

Interviewee
Yeah, that's pretty clear. And again here you have a big discussion on what pieces of the self service infrastructure are supplied and what aren't, because for instance distributed query engine in a governed mesh makes a lot of sense, it makes everything easier. But in a loose mesh, interfaces might move, might change, there is a lot less governance. For all these reasons, and so a distributed query engine might not be able to deal with all of the technologies that are being chosen. So how much you can actually supply as a self service starts depending on the way you go.

Interviewer
Great. Then, is it clear how policies are propagated and executed in a Federated manner?

Interviewee
Well, it's not immediately clear, but I can imagine that the policies are managed in the in the self service infrastructure, managed application management applications. And then I guess they're sent to the management channel. Yeah that is fine.

Interviewer
Uhm, those were the questions. Do you have anything else to share?

Interviewee
No, but I think it's a good picture. What you could do is kind of gray out certain pieces together to highlight a bit more of that difference between a fully government and a loose mesh that emphasize different things.

Also, one thing I would expect is an arrow between the data product consumers and the data discovery, because that's really what should happen is that the data product consumer has a question that it needs data for they go to the data discovery service, find out, oh it's there and then they subtracted that data and start using.

Laslty, maybe you can use different types of lines for management lines and data lines. And that is it.

Interviewer
Ok thanks for your participation.