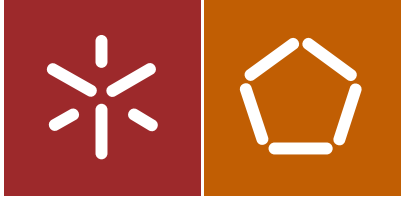
Universidade do Minho
Escola de Engenharia

Inês Araújo Machado

Proposal of an Approach for the Design and Implementation of a Data Mesh

December 2021

Proposal of an Approach for the Design and Implementation of a Data Mesh
Inês Araújo Machado
UMinho | 2021



Universidade do Minho
Escola de Engenharia

Inês Araújo Machado

Proposal of an Approach for the Design and Implementation of a Data Mesh

Master's Thesis
Integrated Masters in Engineering and Management of Information Systems

Work done under the supervision of
Professor Maribel Yasmina Santos (PhD)
Professor Carlos Santos (PhD)

December 2021

DIREITOS DE AUTOR E LICENÇA DE USO

AGRADECIMENTOS

Fui muito feliz. Ser feliz deve ser o início e o fim de tudo no decorrer da nossa vida. Estes anos são mesmo uma viagem, e foi sem dúvida uma viagem alucinante. MIEGSI (porque por muita reformulação que este curso leve, será eternamente MIEGSI para mim), deu-me mais de que algum dia poderei retribuir e por isso sou eternamente grata. Professor Carlos, obrigada por acreditar em mim e nas minhas capacidades, mesmo quando nem eu acreditei. Por estar sempre disponível a ajudar e levar-me mais longe. Nunca teria chegado até aqui "se não estivesse nos ombros de um gigante". Professora Maribel, obrigada por ser uma inspiração como mulher, professora e pessoa. Obrigada por estar sempre disponível, por puxar por mim e me impulsionar a sonhar e alcançar sempre mais. Espero um dia ser metade daquilo que a professora é (em todas vertentes). Professor João Galvão e equipa do projeto P56 Bosch, obrigada por sempre me ajudarem e me fazerem acreditar que tudo é possível! Obrigada pelos conselhos, ensinamentos e a cima de tudo: a amizade. Aos meus pais, que são o meu alicerce de todos os dias. Que fizeram e fazem muitos sacríficos, para me poderem ajudar a crescer, tirar o meu curso e caminhar sozinha nesta viagem bonita que é a vida. Obrigada, pai, por me puxares as orelhas em pequenita e me fazeres ver que podemos sempre ser melhores. Obrigada, mãe, por nunca me deixares desistir de nada e por me ensinares, a cima de tudo, a ser um bom ser humano e tratar todos com o respeito e bondade que merecem. À minha irmã e cunhado, que me acolheram na sua casa durante grande parte destes cincos anos, que me ajudaram nas mudanças para a minha casa, que me fizeram madrinha da sua filha. Obrigada, Beatriz, por ensinares à madrinha que os problemas do dia a dia não são nada comparados à vitalidade do teu sorriso e ao teu abraço. Ao meu Rui, que nestes cinco anos passou de namorado a noivo! Que me deu a mão nos momentos difíceis, que me reergueu quando eu não consegui, que me ajudou sempre (muitas vezes sem saber). Que me apoiou para viajar e apresentar o meu trabalho noutro país, que quer que vá sempre mais longe e fica orgulhoso a cada conquista. Obrigada por ser o "testo do meu tacho". Sem ti, nunca seria possível e sem dúvida que estes cincos anos seriam muito mais enfadonhos. Que continuemos a escrever memórias juntos. Sempre.À minha família e amigos, muito obrigada. Tudo o que sou se deve a vocês. Em especial, àqueles que durante cinco anos cantaram de tricórnio ao peito comigo para os quais serei sempre a Bufas. Obrigada ao staff do Bar de Engenharia e ao eterno Sr. Araújo da EC. É um fechar de um capítulo, mas o abrir de um novo: e essa é a magia da vida.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducentes à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Atualmente existe uma tendência, cada vez mais acentuada, para a utilização de *software* por parte da esmagadora maioria da população (aplicações de caráter social, *software* de gestão, plataformas *e-commerce*, entre outros), identificando-se a criação e armazenamento de dados que, devido às suas características (volume, variedade e velocidade), fazem emergir o conceito de *Big Data*. Nesta área, e para suportar o armazenamento dos dados, *Big Data Warehouses* e *Data Lakes* são conceitos cimentados e implementados por várias organizações, de forma a servirem a sua necessidade de tomada de decisão. No entanto, apesar de serem conceitos estabelecidos e aceites pela maioria da comunidade científica e por diversas organizações a nível mundial, tal não elimina a necessidade de melhoria e inovação. É, este contexto, que origina o surgimento do conceito de *Data Mesh*, propondo arquiteturas de dados decentralizadas. Após a análise das limitações demonstrados pelas arquiteturas monolíticas (e.g., dificuldade em mudar as tecnologias de armazenamento usadas para implementar o sistema de dados), é possível concluir sobre a necessidade de uma mudança de paradigma que tornará as organizações verdadeiramente orientadas aos dados. A *Data Mesh* consiste, na implementação de uma arquitetura onde os dados se encontram intencionalmente distribuídos por vários nós da *Data Mesh* e onde não existe caos, uma vez que existem estratégias centralizadas de governança de dados e a garantia de que os princípios fundamentais dos domínios são partilhados por toda a arquitetura. A presente dissertação propõe uma abordagem para a implementação de uma *Data Mesh*, procurando definir o modelo de domínios do conceito. Após esta definição é proposta de uma arquitetura concetual e tecnológica, que visam a auxiliar a materialização dos conceitos apresentados no modelo de domínios e assim auxiliar na conceção e implementação de uma *Data Mesh*. Posteriormente é realizada uma prova de conceito, de forma a validar os supracitados modelos, contribuindo com conhecimento técnico e científico relacionado com este conceito emergente.

PALAVRAS CHAVE

Big Data, Data Mesh, Arquiteturas de Dados

ABSTRACT

Currently there is an increasingly accentuated trend towards the use of software by most of the population (social applications, management software, e-commerce platforms, among others), identifying the creation and storage of data that, due to its characteristics (volume, variety, and speed), make the concept of *Big Data* emerge. In this area, and to support data storage, *Big Data Warehouses* and *Data Lakes* are solid concept and implemented by various organizations to serve their decision-making needs. However, despite being established and accepted concepts by most of the scientific community and by several organizations worldwide, this does not eliminate the need for improvement and innovation in the field. It is this context that gives rise to the emergence of the *Data Mesh* concept, proposing decentralized data architectures. After analyzing the limitations demonstrated by monolithic architectures (e.g., difficulty in changing the storage technologies used to implement the data system), it is possible to conclude on the need for a paradigm shift that will make organizations truly data driven. *Data Mesh* consists, in the implementation of an architecture where data is intentionally distributed over several nodes of the *Data Mesh*, and where there is no chaos, since there are centralized data governance strategies and the assurance that the fundamental principles of the domains are shared throughout the architecture. This master thesis proposes an approach for the implementation of a *Data Mesh*, seeking to define the domain model of the concept. After this definition, a conceptual and technological architecture is proposed, which aim to help materialize the concepts presented in the domain model and thus assist in the design and implementation of a *Data Mesh*. Afterwards a proof-of-concept is carried out, to validate the aforementioned models, contributing with technical and scientific knowledge related to this emerging concept.

KEY WORDS

Big Data, Data Mesh, Data Architectures

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations and Acronyms

API - Application Programming Interface

BI - Business Intelligence

B2C - Business to Costumer

ETL - Extract, Transform and Load

ELT - Extract, Load and Transform

GB - Giga Bytes

HTTP - Hypertext Transfer Protocol

IOT - Internet of Things

JSON - JavaScript Object Notation

KPI - Key Performance Indicators

OLAP - Online Analytical Processing

REST - Representational State Transfer

SMART - Specific, Measurable, Achievable, Relevant and Time-Bound

UML - Unified Modelling Language

1. INTRODUCTION

The purpose of this chapter is to present the scope and motivation for the development of this master's thesis. It also presents the main research objectives, the research methodology and the literature review process used in this work, and the structure of the document.

1.1. SCOPE AND MOTIVATION

Big Data is an emerging concept and it is related with the ability of providing access to vast amounts of data that may be converted into significant value for organizations (Krishnan, 2013). Therefore, Big Data is inevitable for modern organizations (Santos & Costa, 2020), in order for them to achieve competitive advantages and to improve the interaction and service level provided to the customers (Manyika et al., 2011). Nowadays, there are several companies that have Big Data systems to support their daily business and decision-making processes (Barr, 2020). However, not all organizations treat their data architecture with the proper scalability and democratization that it needs (Barr, 2020), which leads to problems emerging from the monolithic data architectures currently implemented (Dehghani, 2019).

The Data Mesh concept emerges as a necessary paradigm shift that will enable companies to become truly data-oriented, implementing an architecture that brings the opposite of the current models for efficient data product cooperation (Dehghani, 2019). This paradigm shift manifests itself at different levels. From a more structural perspective, data is organized into domains and data teams manage themselves and carry out their own work in an agile and product-oriented way. However, this paradigm shift does not occur only on a structural level, but also on an organizational level - as the way data teams organize and work will become decentralized and mainly focused on a domain (Dehghani, 2020a). The Data Mesh allows for the provision of complex management, access, and support components through the connectivity layer it implements - data from different locations will now be connected in the Mesh (Lance Johnson, 2020).

Consequently, an architecture will be instituted where the data is intentionally distributed through several nodes of the Mesh, also known as domains (e.g., sales, purchases, customer management). However, this should not imply chaos, because the Data Mesh concept ensures six core and shared principles (Discoverable, Addressable, Trustworthy,

Self-describing, Interoperable, Secure), and centralized governance strategies guarantee its homeostatic functioning accompanied by a high interoperability character - an infrastructure of shared self-service data (Dehghani, 2019).

Recently, Zhamak Dehghani began taking the first steps in consolidating what might be the core principles and logical architecture of a Data Mesh (Dehghani, 2020b). However, these specifications are significantly high level, and there is still a lack of empiric, consolidated and validated scientific knowledge on the subject. Although the concept is being disseminated by the author (Zhamak Dehghani, 2020a) and by some companies that already have implemented their own Data Mesh (Justin Cunningham, 2020), due to its emerging characteristic, this concept still lacks constructs, models (e.g., architectures), methods, and instantiations proposed through a research process, being this the main motivation of this master's thesis.

1.2. Research Goal and Objectives

This work aims to propose an approach that guarantees that organizations can focus on building systems that promote data democratization, leaving the Data Lake (or other data systems) and pipelining tools as a secondary concern (Dehghani, 2019). To make this possible, the research goal for this master's thesis is to propose an approach to design and implement a Data Mesh, including a domain model that represents the Data Mesh's constructs and their relationships, and an architecture detailed at the conceptual and technological levels, accompanied by a method for building a Data Mesh, which encompasses a set of best practices and rigorous steps that practitioners may follow. Taking this into consideration, the following research objectives are defined:

1. Propose a domain model that fully describes a Data Mesh, including all its constructs and how they relate to each other through the use of a formal modelling language (e.g., Unified Modeling Language - UML).

2. Propose a conceptual and technological architecture for a Data Mesh that is scalable (e.g., can accommodate an infinite set of domains and datasets), user friendly (e.g., can be easily implemented in any organization or context), and efficient (e.g., can provide fast implementation of Mesh nodes and fast data consumption). Moreover, the architecture should be fully compliant with the six core principles of a Data Mesh (Dehghani, 2019).

3. Propose a method that encompasses a set of best practices and rigorous steps that are fully compatible with the Data Mesh architecture discussed above, aiming to help practitioners implement, as quickly as possible, their Data Mesh.

4. Implement a proof-of-concept to validate the proposed domain model and both conceptual and technological architectures.

1.3. RESEARCH METHODOLOGY

To sustain the rigor of the results produced in this research process, the Design Science Research Methodology for Information Systems is followed. This methodology includes six activities that aim to solve problems, in an efficient and effective way, in the field of Information Systems research (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007).

Although it is expected that the process occurs sequentially, from activity one to activity six, the methodology here depicted does not make this mandatory. Therefore, it is possible that to initiate the process in an activity other than activity one, and we can also return to the previous activities whenever necessary (Peffers et al., 2007). Figure 1 shows the methodology used in the present master thesis.

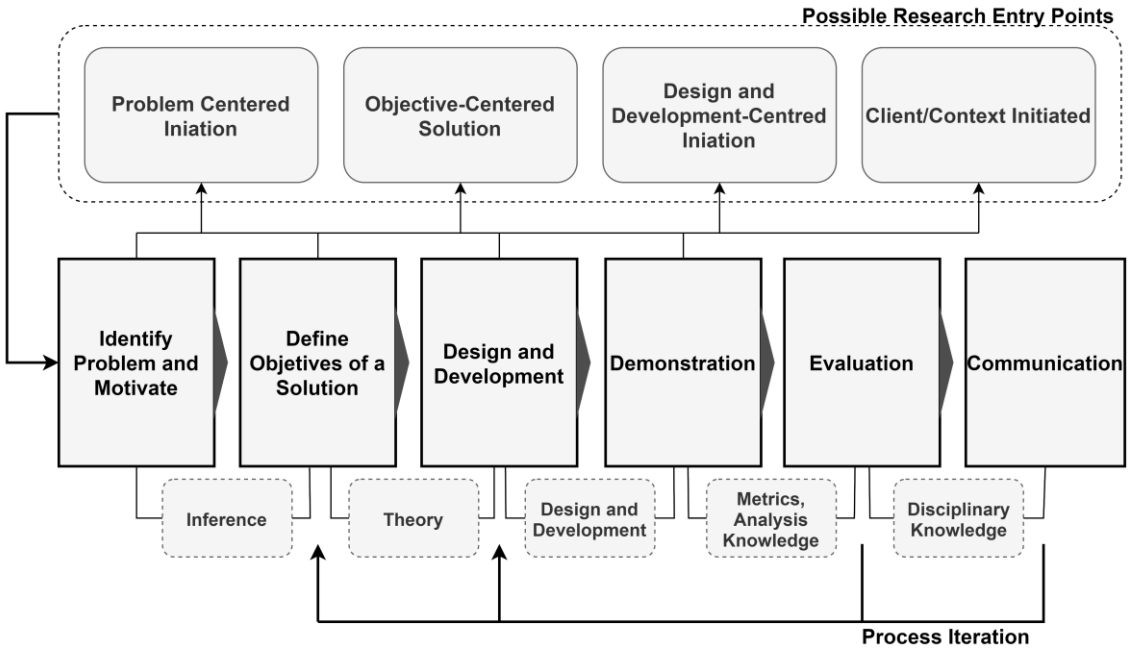


Figure 1. Design Science Research Methodology for Information Systems. Adapted from (Peffers et al., 2007)

- Activity 1: identify problem and motivate. Deals with the definition of the research problem as well as presents a justification for the value of the

solution. The main purpose of this activity is the construction of an artifact that holds the definition of the problem. In this master's thesis, this activity corresponds to the elaboration of the literature review related to the Data Mesh concept.

- Activity 2: define objectives of a solution. Once the problem has been defined, it is relevant to start defining the objectives of the solution. These objectives should be defined based on what is known to be possible to achieve and the same should be aligned with the specification of the problem. In this master's thesis, this activity corresponds to the identification or refinement of the research goal and objectives, making them SMART (Specific, Measurable, Achievable, Relevant and Time-Bound).

- Activity 3: design and development. In this activity, the artefacts to be proposed are conceived and created, which may include any models (e.g., architecture), methods or instantiations, for example. In this master's thesis, this activity will occur in two stages. The first stage concerns the definition of the artefacts (models and methods) before their demonstration and evaluation, and a second stage in which the artefacts will be refined considering the conclusions of those two activities.

- Activity 4: demonstration. In this activity, the demonstration of the produced artefacts is performed, making it possible to verify their usefulness, efficacy and efficiency, for example. To accomplish this, it is necessary to create instances of the problem or to resort to simulations or case studies. As a demonstration case for the resulting artefacts of this master's thesis, will be applied to implement a prototype of a Data Mesh in a specific organizational or societal context (e.g.: online seller company).

- Activity 5: evaluation. this this activity, the proposed artefacts are evaluated. In this sense, it is necessary to understand how the obtained solution satisfies the research goal and objectives established before. In this master's thesis, the prototype implemented in a demonstration case

will be evaluated and discussed in terms of usefulness, scalability, efficiency and user friendliness.

- Activity 6: communication. Lastly, the communication of the problem and the resulting artefacts, as well as their relevance and usefulness. This communication is carried out for the scientific and technical community (e.g., conferences or journal papers). The elaboration of the manuscript associated with this master's thesis is included in this communication activity, as well as the final presentation focusing on the results of this research process.

1.4. LITERATURE REVIEW PROCESS

For the literature review process to be as efficient as possible, and to obtain a consistent basis as a starting point for the development of the present master's thesis, it is necessary to establish a method for it.

In this sense, at the beginning of the literature review process some guidelines were defined to be followed to filter relevant content that can sustain the state of the art of the concept. It should be noted that the literature review is based essentially on posts on the subject published on valid websites, and on informational videos on the subject published by companies, web talks, conferences, and other scientific or technical contents (e.g.: Big Data concept). This fact is justified by the lack of scientific contributions on the topic, due to its emergence and temporal youth (about one year).

Since this is an emerging theme, and therefore temporally very young, there are scientific articles, papers, or other similar documents published to date. Therefore, one of the first points of the methodology followed is related only to the inclusion of reliable sources, which do not lack a subjective basis, influenced by some company/organization. At the time level, it was established that priority would be given to documents published from 2015 onwards - the exception to this rule is associated to documents focused on other existing concepts (Data Warehouse, Big Data Warehouse, Data Lake, among others) - so that current content is presented in the area.

Some keywords used in the search engine of some reference databases such as "Scopus", "Research Gate", "Mendeley", "Google Scholar", among others, were therefore defined.

Some of these keywords are: "Data Lake", "Data Warehouse", "Big Data Warehouse", "Big Data", “Data Mesh”, “Microservices”, “Domain Driven Approach”, among others. YouTube videos and blogs were also used to collect information about the Data Mesh concept. When facing any scientific contribution or publication on the subject, these were selected through their abstract or a quick reading/viewing (procedure more adopted under Data Mesh) to infer the contribution/relevance of it. Finally, the most relevant contents were selected within each contribution, carefully analyzed to finally formulate the literature review. Online alerts have also been created about new publications on Data Mesh, so that they can be included.

1.5. DOCUMENT STRUCTURE

This document is divided into five chapters. The first chapter concerns the introductory part of the document. It starts by presenting the scope and motivation that originated the development of this master’s thesis. Then the research goal and objectives are presented, as well as the followed research methodology (in this case, the Design Science Research Methodology for Information Systems) and the literature review process.

In the second chapter, the literature review is presented, focusing on the concepts and works related with this master’s thesis’s topic. Initially, and to provide the necessary context on the subject, the main concepts (e.g., Big Data, Data Lake and Data Warehouse) are presented. In the second section of this chapter, the motivation for the appearance of the Data Mesh is presented, and in the third section, the main characteristics of this emerging concept are listed. The fourth section of this chapter aims to describe the existing approaches for the design and implementation of a Data Mesh.

The third chapter is for the presentation of the domain model and architectures developed within this master's thesis. First, the domain model is presented, which synthesizes the main conceptual classes about the Data Mesh concept. Subsequently, and based on the materialization of the domain model, a conceptual and technological architecture is formulated.

Chapter four is for the presentation of the proof-of-concept developed around the models and architectures presented in chapter three. This chapter presents the scope of the proof of concept, the organization of the Data Mesh nodes, the organization of folders in HDFS, the organization of tables and databases, and data quality script and report. Further, the

code storage, consumption list, and finally, the communication channel of the Data Mesh is explained. The purpose of this chapter is to detail the implementation of the Data Mesh that was carried out as part of the proof-of-concept.

Finally in chapter five the conclusions of this master's thesis are presented. First the conclusions concerning the literature review process, then the conclusions regarding the proposed approach. Finally, the scientific publications made during this master thesis are presented and detailed and future work is indicated is the end section of this chapter. This master thesis also includes an appendix where some complementary material is presented.

2. BACKGROUND KNOWLEDGE AND RELATED WORK

As mentioned in section 1.5, the present chapter is divided into four sections that have as their purpose the exploration of the related work related to the five main concepts of the present thesis: Big Data, Data Warehouse, Big Data Warehouse, Data Lake, and Data Mesh (the latter encompassing the issue of appearance motivation, features, and approach to design). Although the main theme of the master's thesis is the Data Mesh, it is important to understand the evolution from the appearance of Big Data to the need for a paradigm that we currently face (the Data Mesh). The chapter thus begins with an explanation of the main concepts inherent to the topic in question.

2.1. MAIN CONCEPTS

As already mentioned, this work's core concept is the Data Mesh. However, several other concepts existing in the current literature need to be taking into consideration, as some of them describe concepts and paradigms that originated the need for the concept of Data Mesh. Consequently, this section aims to present and describe those concepts so that they are properly interpreted whenever they are mentioned in this document.

2.1.1. Big Data

The production and consumption of data is a constant in today's world (Santos & Costa, 2020). Clearly, the way data is produced and consumed nowadays is nowhere near the way this phenomenon occurred a few decades ago. Some state that a decade ago what was considered "a great dataset", would nowadays be probably considered absurd (Diebold, 2013).

Explaining the concept of Big Data implies considering this phenomenon of constant data production and consumption in which society presents itself (whether at a social, organizational, industrial level) (Santos & Costa, 2020) but also return to the origin of this concept. It is not clear where the term originates from, with references pointing to the fact that the term was used for the first time in the paper "Big Data' Dynamic Factor Models for Macroeconomic Measurement and Forecasting" (presented in 2000 at the "Eighth World Congress of the Econometric Society in Seattle"), according to (Diebold, 2013). However, it is known that its origin comes from the conjuncture of distinct

contexts and areas such as industry, academia, computer science and statistics and that it will have appeared in the context of an informal conversation in the 90's (Diebold, 2013).

The concept has evolved over time, particularly from 2012 onwards, and in recent years, there has been a growing interest in the area (Santos & Costa, 2020). The rapid evolution of the concept has led to some confusion on how to explain it, thus diverging between "what Big Data is" and "what Big Data does" (Gandomi & Haider, 2015). Some authors try to define Big Data as the access to a vast set of data that allows organizations to create value from it (Krishnan, 2013). Other authors recognize that there may still be some ambiguity in determining at what point we start talking about "Big Data" instead of just data (Adam Barker & Jonathan Stuart Ward, 2013), and they choose to define Big Data as everything that is too big, too fast, and too difficult to be processed by the tools currently in use within a specific context (Costa & Santos, 2017), citing (Sam Madden, 2012). More authors follow a similar line of thought, defining Big Data as data so large (in the order of terabytes to exabytes) and complex (from Internet of Things (IOT) to social networks) that it needs innovative technologies for data storage, transformation and analysis (Chen, H.L.Chiang, & C. Storey, 2018).

With the observation of the different definitions, we can conclude that to define Big Data, it is necessary to go into detail about its characteristics. Doug Laney formulated the 3Vs model (Volume, Variety and Velocity) in 2001, and this model served as the basis for defining Big Data for a decade (Santos & Costa, 2020), because it is believed that these would be the three main challenges when dealing with Big Data, also forming its main characteristics.

According to some works, the volume in Big Data cannot be based on the size of the data (e.g., terabytes or exabytes). Just as 200GB of data were considered "big" a decade ago (Diebold, 2013), it is possible that the same will happen with the measures that today are considered "big", due to the continuous increase in data production and storage capacity (Gandomi & Haider, 2015). Therefore, the volume in Big Data is related to the amount of data that is continuously generated (Krishnan, 2013), so it is not possible to set a threshold for Big Data (Gandomi & Haider, 2015), since different types of data require different technologies capable of processing them (Costa & Santos, 2017). One of the main causes related to this incremental amount of data being generated, is the fact that

we are continuously storing information about our interactions with the services that we use daily (Santos & Costa, 2020).

The variety in Big Data concerns the structural heterogeneity present in the data (Gandomi & Haider, 2015). The data can thus be classified into three distinct categories: structured, semi-structured and unstructured (Santos & Costa, 2020). Examples of structured data is transactional data and relational databases. Unstructured data follow a distinct logic, being examples of the same audio, video, or social network posts. The semi-structured data is in the middle of the other two categories, being web server logs and JavaScript Object Notation (JSON) files examples of semi-structured data (Santos & Costa, 2020). Big Data systems allow the processing of this data with different structures, which was already partly the case in the past by various organizations, now becoming a more efficient process when it comes to leveraging data in business processes (Gandomi & Haider, 2015).

The third "V" in Big Data concerns the inherent velocity of data (Gandomi & Haider, 2015). This data velocity can be seen from two different perspectives: either the velocity in which the data is produced, or the velocity needed to meet the associated decision-making needs (Santos & Costa, 2020). The increasing usage of data-producing components (e.g., sensors, applications, and cell phones) leads to more and more data being produced. This data production leads to the need for subsequent processing and storage, in order to meet the needs imposed by the stakeholders (Gandomi & Haider, 2015). This phenomenon leads to the installation of a continuous stream of data, which with the proper transformation, guarantees an added value to the organizations' decision-making process (Andrade, Costa, Correia, & Santos, 2019).

With the continuous work in Big Data, it was possible to see that the 3V's model was incomplete, and that two more could be added to it: veracity and value (Santos & Costa, 2020).

The veracity, the fourth “V” of Big Data, is related to the inevitable imprecision present in the data. Thus, there may be data analyses that present different degrees of accuracy, reliability, or quality (Santos & Costa, 2020). On the other hand, there is another dimension of imprecision related to data "subject to strict interpretation" (e.g., people's feelings). However, these data can be quite useful and can bring value to the analyses

when handled and treated with the appropriate techniques and technologies (Gandomi & Haider, 2015). The value, the last "V" of Big Data, is related to the analysis and processing of Big Data (Santos & Costa, 2020). Normally this value tends to be lower in raw data, but in the end, after an adequate data processing and analysis, it becomes much higher (Gandomi & Haider, 2015).

Although these five main characteristics have been defined, other authors identify even more characteristics such as variability, complexity, ambiguity, viscosity and virality, for example (Santos & Costa, 2020), citing (Gandomi & Haider, 2015). Figure 2 summarizes these characteristics.

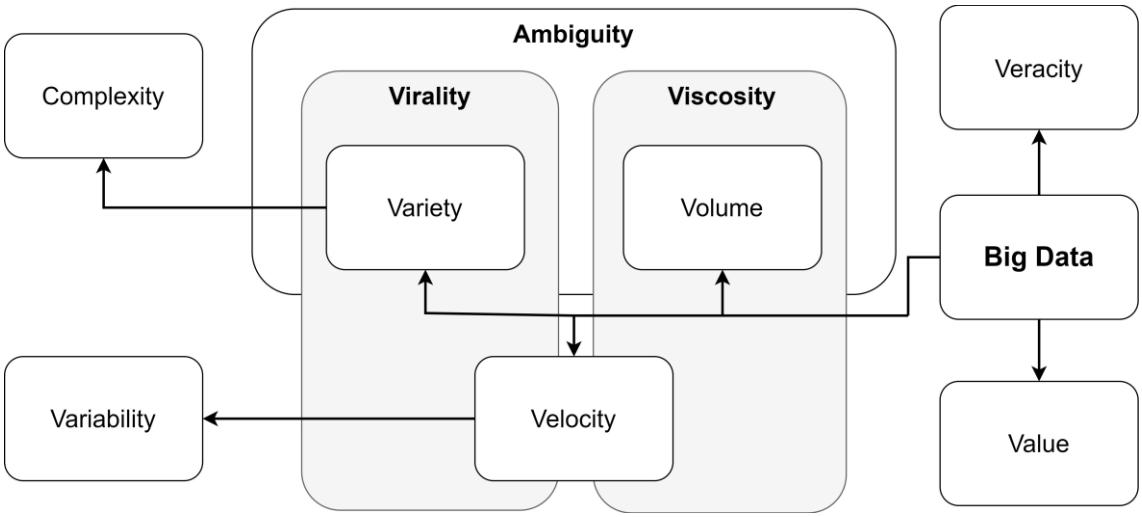


Figure 2. Main Characteristics Identified in the literature. Adapted from (Santos & Costa, 2019)

The complexity in Big Data arises from the heterogenous ambit that is often associated to it, since there are different data producing sources that require greater effort in their integration and treatment of your data (Gandomi & Haider, 2015). On the other hand, this constant flow of data, nourished with high volume and speed, can cause friction, which explains the viscosity being pointed out as a characteristic. Variability is already based on the different velocities that are verified in the various data flows (so there may be higher rates than others in certain sources). The ambiguity, in Big Data, appears as the gap associated with the lack of metadata to accompany the data - mainly coming from the junction of volume and variety in this area. Finally, virality is related to the speed of data propagation (Krishnan, 2013). Figure 2 represents all the characteristics of Big Data identified in the reviewed literature, as well as the relationship found between them.

Big Data due to its characteristics remains an abstract concept for which there is no consolidated definition accepted by all. However, its characteristics are already well defined among the scientific community. With the explanation of the Big Data concept, it is possible to infer that one of the challenges felt in the area is the issue related to the paradigms and technologies for storing Big Data. To understand the path taken by the scientific and technical community until the appearance of the Data Mesh, it is necessary to explore what precedent paradigms and technologies exist. In this sense, the concepts of Data Warehouse, Big Data Warehouse, Data Lake and, finally, Data Mesh will be presented in the sections below.

2.1.2. Data Warehouse

The technological advances over decades have led to an exponential increase in the amount of operational data that organizations produce (Golfarelli, M., & Rizzi, 2009). However, this operational data alone does not allow to support decision-making processes due to its nature - for this to happen, there must be mechanisms that extract perceptible analytical value from this operational data, so that it can reach the stakeholders interested in that value (Kimball & Ross, 2013).

As a solution to that problem, in the seventies, the phenomenon of Data Warehousing emerged as a response to this imminent need to use the data produced by organizations, in order to generate value - which goes beyond the routine tasks of an organization (Golfarelli, M., & Rizzi, 2009). Until then, organizations were only concerned with keeping the operational data resulting from business processes, leaving aside the ability to access information needed for the decision-making process (Golfarelli, M., & Rizzi, 2009).

According to Kimball, Data Warehouses can be summarized as systems that ingest operational data (over which they have no quality control) and hold as output the analytical value for decision-making (Kimball & Ross, 2013). Golfarelli & Rizzi, on the other hand, define a Data Warehouse as a collection of techniques, methods and tools that support managers, directors, and analysts. The purpose of this collection is to conduct data analysis to support decision-making. They also admit that this definition is intentionally vague, so that it is inferred on the conceptual aspect but not on its structure (Golfarelli, M., & Rizzi, 2009).

In 1996, Kimball made a survey of the main complaints at the organizational level, which made it possible to infer some relevant characteristics to be held by the Data Warehouse. In this regard, the identified characteristics are accessibility, integration, query flexibility, information consistency, multidimensional representation, correctness and completeness (Golfarelli, M., & Rizzi, 2009). Later, it also points out that there is a special relevance to the fact that Data Warehouses need to be accepted by the organization and need to present themselves as authoritative and trustworthy systems to improve decision-making. Only then they will be truly successful (Kimball & Ross, 2013).

Inmon makes the definition more concrete, presenting a Data Warehouse as also being a data repository that supports decision-making. This is characterized by being subject-oriented, presenting an integrated and consistent, non-volatile character capable of presenting evolution over time (Inmon, 2005).

Kimball & Ross clearly define the structure of a Data Warehouse as the integration of four distinct components: source transactions, extract transform load (ETL) System, presentation area and Business Intelligence (BI) application (Kimball & Ross, 2013). In Figure 3, it is possible to visualize how these components are organized.

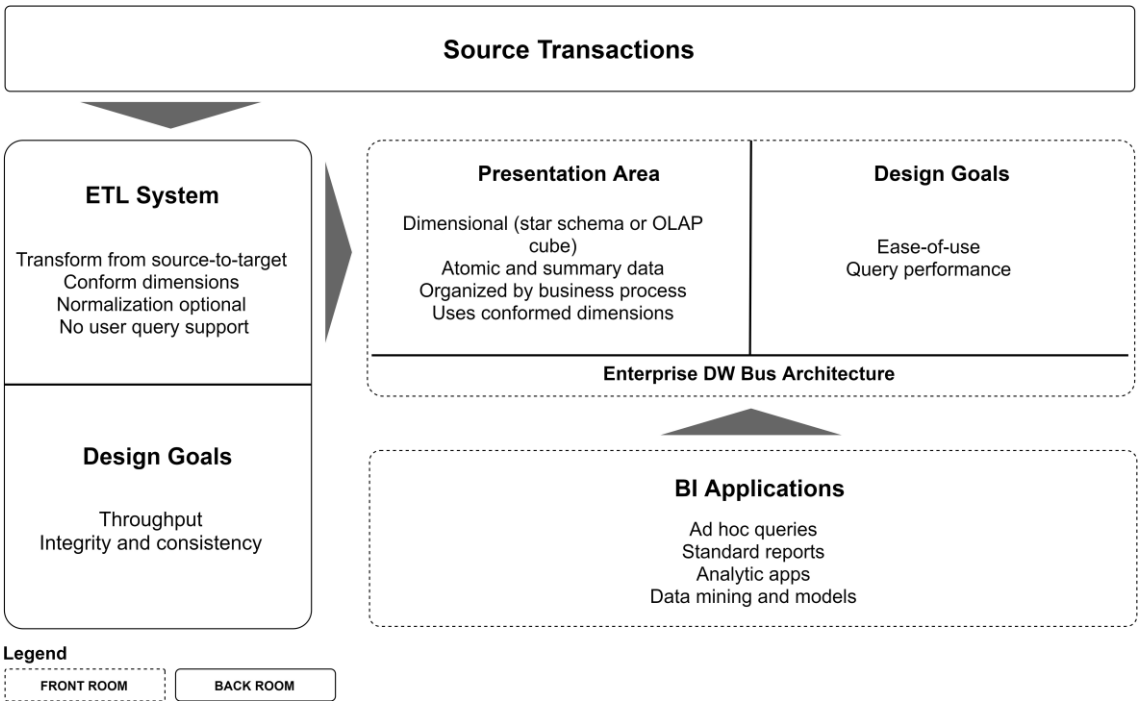


Figure 3. Core elements of Data Warehouse architecture. Adapted from (Kimball & Ross, 2013)

The source transactions component concerns the sources that store the operational data (business transactions), being these data the fuel for the Data Warehouse. The ETL system is the component responsible for the extraction, transformation and loading of the data from the source transactions component. Thus, the data undergoes structural and corrective changes, so that its analytical value is enhanced. The presentation area component concerns the availability of data for access, through its organization and storage. Finally, the BI Application concerns dashboards, ad-hoc queries and data mining components linked to the Data Warehouse, i.e., a combination of analytical capabilities (Kimball & Ross, 2013).

There are several Data Warehouse modeling strategies, one of the most emblematic being the dimensional modelling strategy (Nenad Jukic, 2006). According to the dimensional modelling strategy (Kimball & Ross, 2013), two distinct constructs should be taken into consideration when modelling Data Warehouses: the fact tables and the dimension tables. In the fact tables, we store the events that take place in the organization and that affect the decision-making processes. Dimensions are prisms on which the fact data is analyzed, bringing different perspectives to the analysis (Golfarelli, M., & Rizzi, 2009).

In short, the Data Warehouse is a repository that stores the organization's information, and that enhances its analysis by a wide range of users (Santos & Costa, 2020), being the same structured and modelled according to the constructs and components presented in this section. It should be highlighted that there are several other ways to build Data Warehouses, but the works discussed in this section only aim to represent the most commonly used constructs and strategies so that we understand one of the core paradigms used for analytical data storage, the Data Warehouse.

2.1.3. Big Data Warehouse

Although the Data Warehouse as described in section 2.1.2, is widely accepted and implemented, the way we design and implement a Data Warehouse to support Big Data contexts has been the focus of some research contributions during the last few years. These changes mainly focused on the evolution that was needed due to the Big Data characteristics and to the need for advanced analytics, which began to make the existing Data Warehouse architectures precarious to support these contexts (Santos & Costa, 2020). Therefore, the scientific community began to study the modernization of the Data Warehouse, in order to accommodate these several changes that were needed (Russom,

2016). During the modernization process, some difficulties arose, such as the cost of implementing new technologies on a use case basis (without adequate models and methods) and the lack of data governance (Santos & Costa, 2020), which naturally led to the more research on the Big Data Warehouse concept. Research related to the Big Data Warehouse concept can be divided into five different topics, being them respectively (Santos & Costa, 2020): the characteristics and design changes of the Data Warehouse for Big Data environments, Data Warehouses using NoSQL Databases, benchmarking of storage technologies for Big Data Warehouses, improvements for query engine and evolution in OLAP systems, and Big Data Warehouse implementations in specific contexts.

(Costa, Andrade, & Santos, 2019) define a Big Data Warehouse as a scalable, high-performance, and highly flexible processing system - capable of handling ever increasing volumes of data, accompanied by significant variety and speed. They emerge as well as the way to overcome the difficulties experienced by the Data Warehouse when processing Big Data (Krishnan, 2013). We can thus define a Big Data Warehouse as a system presenting flexible storage, accompanied by adequate scalability and performance. These systems also focus on low latency when it comes to data ingestion and analytical workloads of complex nature (Costa et al., 2019). Big Data Warehouses also have real-time capabilities (linked to low latency and streaming processes), significant interoperability and fault tolerance, making use of commodity hardware to reduce their inherent costs (Santos & Costa, 2020).

The evolutionary process from a Data Warehouse to a Big Data Warehouse can happen according to two different strategies: "lift and shift" or "rip and replace". The first strategy is related to the augmentation of the implemented Data Warehouse capacities (e.g.: introducing a new technology); the second strategy is based on a more extremist perspective, in which one adopts a totally new strategy to build a new Big Data Warehouse. Hadoop and NoSQL databases are present either to increase the capabilities of a Data Warehouse, or as a new technological layer to be used in a completely different architecture (Santos & Costa, 2020).

When designing a Big Data Warehouse, the focus must be divided equally by two layers: the physical layer and the logical layer. However, there is still a significant gap between what a Big Data Warehouse should be, and how to build one (Santos & Costa, 2020). The

authors (Costa et al., 2019), have scientifically contributed in this area, formulating an approach that aims to ensure the characteristics of a Big Data Warehouse, focusing on both the logical and physical layers (Santos & Costa, 2020). This approach presents three logical components of a Big Data Warehouse, being them respectively: i) data collection, preparation, and enrichment; ii) platforms - data organization and distribution; and iii) analytics, visualization, and access.

The first logical component, “data CPE”, is related to the arrival of data (from the data provider) and its collection. The data can be collected either via batch or streaming mechanisms. After this collection, the data advances to the second logical component (“platforms: data organization and distribution”). This component is composed of three different storage areas, divided into two types: file system and indexed storage. These three storage areas make it possible to deal with different Big Data characteristics and workloads efficiently. Finally, the logical component of “analytics, visualization, and access” holds as core component a distributed query engine, which allows the combination of batch and streaming data in a single query. This component can thus be used to exploit data from the different storage areas of the Big Data Warehouse, which can then be used for visualization and other types of data access (Costa et al., 2019).

Therefore, it is possible to conclude that considering the challenging Big Data characteristics, there has been the need for an evolution from a Data Warehouse to a Big Data Warehouse. It can also be concluded that, in certain justifiable contexts, implementing a Big Data Warehouse may only mean augmenting the capabilities of an existing Data Warehouse to partly (or less efficiently) deal with Big Data.

2.1.4. Data Lake

The concept of Data Lake dates back to a decade ago, through James Dixon, and was partially devalued at the time because it was believed to be a Hadoop marketing label (Miloslavskaya & Tolstoy, 2016). However, the concept has remained and has grown over time (Khine & Wang, 2018).

Laskowski describes Data Lake as a largely scalable repository denoted as a significative mass, where data is stored in its "As-Is" form, remaining in this state until the need for the data to be processed arises. This addition of raw data does not interfere with the data structures already present in the lake, which allows to continuously inject data in the lake

Data Lake, without the concern about the above-mentioned data structure (Nicole Laskowski, 2016). Miloslavskaya & Tolstoy present their definition of the concept as an immense pool of data, in which it is continuously store new data of three types (structured, semi-structured and unstructured), being it accumulated with historical data. The authors also complement with the note that the data schemes and requirements are not defined upfront, i.e., until the data needs to be processed (Miloslavskaya & Tolstoy, 2016). Terri McClure (Terri McClure, 2014) translates the conceptual change in the definition itself, stating that "Yesterday's unified storage is today's enterprise Data Lake", which reinforces the fact that this concept appears intimately to the entrepreneurial part at the expense of the academic one (Miloslavskaya & Tolstoy, 2016).

Data Lake and Data Warehouses are both data repositories. However, they differ from each other in several aspects, and even in structure and implementation (Khine & Wang, 2018). Data Warehouses follow a "Schema-on-Write" approach since they make use of the traditional ETL logic. The data is thus extracted from the sources, then processed and finally loaded into the repository (having its schema defined before loading). Therefore, it is assumed that Data Warehouses are prepared to handle read-heavy workloads.

On the other hand, in the Data Lake there is a different order regarding data processing, which is justified by its "Schema-on-Read" approach. The preprocessing of the data does not happen until the data is needed by an application or consumption query. Consequently, there is a change in the ETL tractional order, being the process now defined as Extract, Load, Transform (ELT). Note that when the data is extracted from the source, the necessary metadata is thus added to it. The Data Lake is thus prepared to handle write-heavy and read-heavy workloads (Khine & Wang, 2018).

Summarizing the comparison between the Data Lake and the Data Warehouse, it is possible to conclude that the former deals with three types of raw data (unstructured, semi-structured and unstructured), while the latter mainly deals with processed and structured data. Storage costs are much higher in Data Warehouse environments, and this type of repository is less agile compared to the Data Lake. There is also a differentiation in the level of the users, as the Data Warehouse mainly targets professional business users (e.g.: managers, directors, among others), while the Data Lake mainly targets data scientists (Khine & Wang, 2018).

The main characteristics of a Data Lake are data storage without processing and theoretically infinite storage capacity (Khine & Wang, 2018). For a Data Lake to be successful, it must also ensure scalability of the architecture with great availability, data governance, centralized cataloging and indexing, shared-access model, and agile analytics (Miloslavskaya & Tolstoy, 2016).

From a technological point of view, several Data Lake implementations are based on Apache Hadoop. The various datasets will be extracted and stored in a Hadoop Cluster. In the case of real-time data involving streaming, Apache frameworks such as *Spark* and *Flink* are commonly used. There is thus the ability to handle data at different velocities and to store heterogenous data in a structured way. The Data Lake also includes a semantic database, a model and the addition of a layer that allows the relationships between data (Khine & Wang, 2018).

It is possible to decompose a Data Lake into three layers. The first one is related to the raw data (transactional data), the second one is related to the data that increase daily and finally, external information. This division can also be performed using a timeline. The first layer corresponds to the data of the last six months, the second layer to the data older than six months but still used by the organization and the third layer to the stored data (data that isn't used frequently) (Miloslavskaya & Tolstoy, 2016).

In recent years, some works have been focused on Data Lakes at the level of services and optimization (Beheshti et al., 2017) (Hai, Geisler, & Quix, 2016). *Constance* is an intelligence data lake system that can discover, extract and summarize structural metadata from data sources, and it appears as a way to avoid the lack of metadata management in place - which can turn a Data Lake into a Data Swamp (Hai et al., 2016). *CoreDB* is also an example of a work developed in this area, being an open-source service that offers a Representational State Transfer (REST) Application Programming Interface (API) for indexing and organizing all the metadata of a Data Lake - thus helping to suppress some difficulties arising from the heterogeneity of the data sources (Beheshti et al., 2017).

In short, the Data Lake corresponds to a pool that accepts structured, semi-structured and unstructured data, and theoretically scales to an infinite amount of raw data. The Data Lake can be considered more effective and efficient to deal with heavy workloads, compared to the already established Data Warehouses. It should be highlighted that in a

Data Lake governance must always exist, so that it does not become a Data Swamp (Khine & Wang, 2018).

2.1.5. Data Mesh

The Data Mesh emerges as a disruptive and innovative concept, which proposes a paradigm shift to make organizations truly data-oriented (Dehghani, 2019). Dehghani is one of the pioneering authors on the subject, which serves as a basis for the other concepts and definitions formulated so far. The author defines Data Mesh as an intentionally distributed data architecture. In a Data Mesh, it must exist governance and standardization efforts that allow for interoperability among the Mesh nodes. For this to be possible there is a homeostatic self-service data infrastructure (Dehghani, 2019). Moses extends the previous definition by stating that the Data Mesh is an architecture that embraces ubiquity, with a design oriented to data organized by domains (Barr, 2020). Johnson argues that the Data Mesh enables connectivity between the various "silos" of distributed data, preventing the Mesh from becoming inefficient due to its distributed nature. It also abolishes the complexities of managing and connecting distributed data (Lance Johnson, 2020).

In a ThoughtWorks post, the basic concepts that support the Data Mesh are presented. The first concept is related to the orientation by domains, as well as the decentralization of data ownership. The second concept highlights the concept of data as a product. The third concept is related to the self-service nature of the platform and, finally, governance issues are highlighted to allow homeostasis of the Mesh (ThoughtWorks, 2020). In this same contribution, two core issues are highlighted in the implementation and use of the Data Mesh, i.e., the technological and organizational adaptation that is needed to make the Data Mesh possible (Dehghani, 2019; ThoughtWorks, 2020). The explanation of these concepts can be found in detail in section 2.3 and 2.4 of the document.

Currently, there are some companies around the world that have a Data Mesh implemented and supporting their data analytics processes. One example is brought by *Justin Cunningham*, from Netflix, which currently uses this type of distributed data architecture. At the technological level, for its implementation, it uses a group of technologies developed internally at Netflix (part in open source) and others like *Apache Kafka*, *Apache Flink*, among others (Justin Cunningham, 2020). Another company betting on this data architecture is Zalando (Max Schultze & Arif Wider, 2020). They adopt a

more concise position, advocating that the Data Mesh is more related to the ownership of data, rather than focusing on the technological perspective. In their view, which also comes from their practical implementation, they argue that Data Mesh is based on three concepts: product thinking, domain driven distributed architecture and infrastructure as a platform (Max Schultze & Arif Wider, 2020).

To conclude this section, is highlighted that the Data Mesh arises from the frustrations felt with the current monolithic data architectures, accompanied by a logical path for the application of the principles of microservices architectures to the data (Dehghani, 2019). In the following sections, it is explained the motivation for the appearance of the Data Mesh, as well as the current body of knowledge that form the state of the art related to the concept of Data Mesh.

2.2. MOTIVATION FOR THE APPEARANCE OF THE DATA MESH

Thomas Kuhn, a physicist and philosopher wrote *"The structure of Scientific Revolutions"* (Kuhn, 1970). In it, the author emphasizes the way science evolves, being divided into four phases: normal science, detection of anomalies, crisis and change of paradigm. In a simple way, the change of paradigm occurs when, in the face of scientific progress, it is realized that there is no way around the anomalies, entering a crisis, which is overcome through a change of paradigm (Kuhn, 1970). This cycle postulated by Kuhn, justifies the appearance of the concept of the Data Mesh, considering the current monolithic data architectures and their respective limitations (such as scalability, for example) (Dehghani, 2020a). *Dehghani* argues that the data architectures are currently in a state of crisis, and therefore the word "paradigm" establishes a symbiotic relationship with the concept of Data Mesh (Dehghani, 2020a).

In the last two years there has been a significant increase of interest and investment by companies in areas such as Big Data and Artificial Intelligence. Between 2018 and 2019 there was an increase in this interest and a consequent investment of 66% (compared to the previous year). However, contrary to the expectations, the satisfaction of the companies has decreased (recognition of the importance and belief in these technologies to increase competitiveness and value). Only between 2018 and 2019, there was a decrease of 19%. According to Dehghani, this fact shows how, although there was an

evolution since the 80s (from Data Warehouses) to the present day (Data Lakes in the cloud), there are still serious gaps in the adopted architectures (Dehghani, 2020a).

There is currently an overload in the data teams when they try to respond to the growing needs of the organization, ranging from ad-hoc exploration to central ETL data pipeline management. There is an unsatisfactory alignment between the organizational needs and the architectures instituted in the organizations (Barr, 2020). The two facts above-mentioned, lead to this overload felt by the data teams and the dissatisfaction of various investors. Although in software engineering there has been a notable evolution from monolithic architectures to microservice architectures, the same did not occur in the data engineering space (Dehghani, 2019).

This stagnation implies the segmentation of the data architecture into three components: sources, big data platform and consumers. By sources it is defined the data producing sources (data resulting from the operational nature of organizations, as well the external sources influencing the organizational performance), and by consumers, this means, those who use the analytical outputs (Dehghani, 2020a). Therefore, there is a dimension that is not considered here - the nature of the data itself and the way it is organized in an organizational environment. Currently, there is no concern about domain organization (Dehghani, 2020b). More than that, this is an architecture similar to those that were abandoned in the past: data (ingest), business (serve), user interface (consume). Layer defined by their technological capabilities (Dehghani, 2020a). The problem of this decomposition into layers, is the change that happens in systems: the change is not often restricted to a layer. For example, if a new source is added, there must be a change in the three components - and this obligation to change the process in the layers, causes significant friction in the process. (Dehghani, 2020b).

On the other hand, the investment problem arises, since the management of monolithic architecture (e.g.: Data Lake) requires extremely specialized professionals, which with the increase in the complexity of the system, requires an increase in the composition of these teams. Currently the teams dealing with monolithic data architectures (e.g.: data platform engineers, data scientists, domain's operational systems teams, among others), are working on the same subject, however there is a “space” between them. As an example, the domain's operational systems teams have as main concern the execution of their systems, leaving behind the availability of analytical data in a friendly way at the

level of analysis and consumption by the rest of the organization. At the other end of the spectrum, data scientists are concerned that the data they need exists, in a consumable way, so that they can train their Machine Learning models and thus fulfill their part in the system. However, to have this data, data scientists depend on data engineers, who act as a bridge between the parties, who are not aware of the date itself (has no notion of their domains). Due to these needs (from other teams), and the lack of knowledge of the nature of the data they handle, they are under high pressure in the organization (Dehghani, 2020a). Therefore, it is possible to infer that the teams are organized in silos highly specialized in data tools, which when they look at the system as a whole, explains the friction that exists in them (Dehghani, 2019).

The Data Mesh arises as a paradigm shift that occurs both at the technological and organizational levels. This change has the purpose of solving the problems enumerated in this section, to make organizations completely data-oriented, thus being able to gain competitive advantages and withdraw organizational profits from this fact.

2.3. FEATURES OF A DATA MESH

The Data Mesh has as main purpose the creation of an architecture that enhances the extraction of value from historical facts and analytical data at scale. Scale is understood here as being the adaptation to constant change and proliferation of data production sources, in order to satisfy consumer needs (Dehghani, 2020b).

For the Data Mesh to achieve its purpose, it must be based on four core concepts. The first one is related to the way data is organized according to the nature of the organization - **domain-oriented decentralized data ownership and architecture**. The second concept implies changing the way data is viewed within the organization - the concept of **data as a product**. The third concept is related to the transformation of the service infrastructure into a **self-serve data infrastructure** paradigm. Finally, the concept that avoids chaos in the Mesh - **federated computational governance** (Barr, 2020; Dehghani, 2019, 2020b).

When looking at the organizational structure, it is realized that, naturally, divisions are defined by areas of operation (e.g., logistics, customer support, and sales), also known as business domains (Dehghani, 2020a). The Data Mesh postulates the existence of a distributed responsibility, by the teams of the organization, that can better understand and

produce the data of their specific business domain (Barr, 2020). In this sense, ownership should be taken into consideration, and not forgetting the data domains from the moment the data is ingested. Therefore, serving the analytical data must always be aligned with the established domains (Dehghani, 2019). To distribute responsibility and to decentralize the already known monolithic architectures, it is necessary to model the current data architecture based on the organization of analytical data by domains (Dehghani, 2020b). Will thus be facing a situation where the domains store their datasets and serve the respective data in a simple and friction-free way (Dehghani, 2019). The physical storage of data can be kept centralized. However, the datasets and the ownership/responsibility are kept close to the respective domains. The domains establish, logically, relationship among themselves. However, these relationships do not cause friction, and a new logic of serving and pulling is established, from domain to domain (Dehghani, 2020b).

The cost of discovering quality data, in line with the high friction, is pointed out as one of the greatest difficulties experienced in current monolithic data architectures (Dehghani, 2020b). Part of the problem comes from the fact that even organizations that consider themselves data-oriented do not treat the data with the proper democratization (Barr, 2020). In this sense, the second concept of the Data Mesh emerges, i.e., data as a product (Dehghani, 2019). The above mentioned concept has as objective the resolution of the problems related to data silos, as well as the problems of the data with poor quality and freshness (Dehghani, 2020b). The Mesh applies the already known concept of *"Product Thinking"* to the data, so that it becomes the organization's top priority, and leaves data pipelining and storage concerns in the background (Dehghani, 2019). A simple concept is applied here: analytical data is now seen as a product (and therefore there is an underlying quality dimension), and consumers of this data are now seen as customers, and their needs must be met (Dehghani, 2019).

Observing the two concepts highlighted here, it is possible to infer that they imply the existence of an infrastructure that allows the teams to produce and maintain their data products. For this, it is necessary that the teams have access to a high-level infrastructure, capable of encapsulating all the complexity that usually comes with it. Thus, are dealing here with the third concept - self-serve data infrastructure - that empowers teams with the autonomy needed to manage their domains (Dehghani, 2019).

From domain to domain, one will experience a choice of disparate technologies, which at the limit, reach the desired goals in each data product (Dehghani, 2019). For the Data Mesh to work as expected, the notion of interoperability and connectivity must be present and well-defined (Dehghani, 2020b). Thus, the self-serve platform must be able to provide the tools and interfaces necessary for the creation and maintenance of data products, without the need for highly specialized knowledge, such as the one that is currently see in Data Lakes. In short, this platform must be as multilingual as possible, from the data storage to the data pipeline declaration (Dehghani, 2020b).

Finally, there must be a mechanism that allows interoperability between different domains - the governance model (fourth concept). This governance model must be able to carry out an automated execution of decisions, as well as accompany the decentralization and independence of each domain in the Mesh. For this, global normalization is necessary, as previously mentioned, which Dehghani denominates as "federated computational governance" (Dehghani, 2020b). This model embraces the globality and complexity of the Mesh as a whole, hence creating global rules for it, but leaving room for local rules in each domain. This concept aims to apply a set of rules to all interfaces of the various data products, as well as to the data products themselves, in order to guarantee the homeostasis of the Mesh. However, and due to the architectural complexity of the Data Mesh, the definition of this governance model is something particularly difficult to present. Nevertheless, the global rules defined by this model must allow interoperability, as well as the adequate functioning of the Mesh. Federated computational governance is a complex model, which does not reject change and has several contexts (Dehghani, 2020b).

In general, and taking into account the above-mentioned concepts and key points, the main features that the Data Mesh provides are (Dehghani, 2020b):

1. Decentralized team constituted by domain representatives, and a clear ownership and responsibility for each data product.

2. Use of a self-serve platform to support the development of data products in the Mesh.

3. Definition of how to model the quality, requirements, and security of data.

4. Dealing with the various languages/technologies used in the data products, to ensure interoperability.

Figure 4 illustrates a model of a Data Mesh, with all its implicit components. Through the analysis of the figure, it is possible to infer about the interconnection of the various components of the architecture. Analyzing the figure from top to bottom, it is first illustrated the federated computational governance, where are present the policies that allow the interoperability in the Mesh. These are applied to the various data products present in the analytical data plan and operational services data plan. In this part of the figure, the architecture quantum, and domain-oriented data (example) is also highlighted. Analyzing the highlight of architecture quantum, it is possible to perceive that it includes interactions between the two planes (analytical and operational services) at the level of data product and micro-services. It is also possible to infer that the computational policies are not only applied to the data products present in the analytical data plan, but also in the platform itself for this purpose - thus serving the figure as a concretization of the concepts presented above.

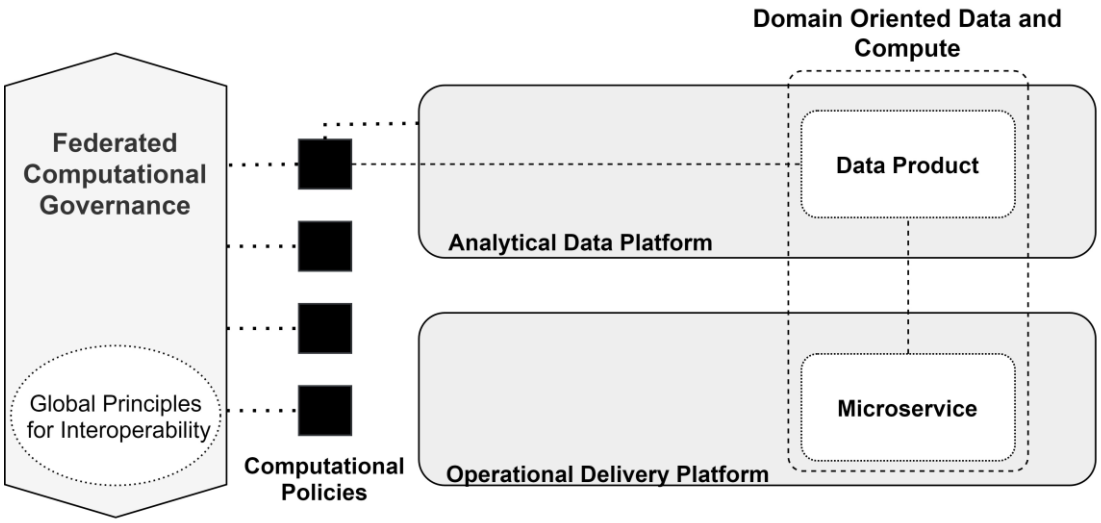


Figure 4. Data Mesh Architecture. Adapted from (Dehghani, 2020b)

2.4. Examples of Data Mesh Proposals and Implementations

2.4.1. Approach Followed by Dehghani

Due to the logic of operation and organization imposed by the Data Mesh and the concepts defended by this paradigm, several concepts must be taken into account when designing

the Mesh (Dehghani, 2019). This section provides an overview of the contributions that focus on proposing an approach for designing and implementing a Data Mesh. Some contributions focus more on the conceptual layer of the Mesh, lacking technological details that are relevant for the implementation of a Data Mesh, while others solely describe real-world implementations that may lack sufficient conceptual or technological details and diversity to be generalized for the design and implementation of a Data Mesh in any organizational or societal context.

Regarding the concept of domains, it is important to understand how these domains arise and are organized. There are two types of domains: source and consumer (shared) domains (Dehghani, 2020a). Source domains consist of the data in their raw state at the point of creation, not modeled for any consumer. They represent the reality of the business (with the data being mapped very close to its origin) and therefore change less frequently (regarding its structure) - since the business facts do not present a very volatile nature. These domains are characterized by their functional character and the permanent need for data collection (Dehghani, 2019). Consumer (shared) domains are data domains that may or may not be aligned with source domains. They are different in nature from source domain data, as they undergo significant structural changes. In these domains, the transformed data is often presented in aggregated views (originating from the source domains). These domains also include models that allow access to them (Dehghani, 2019). The interaction between the two types of domains can be seen in Figure 5.

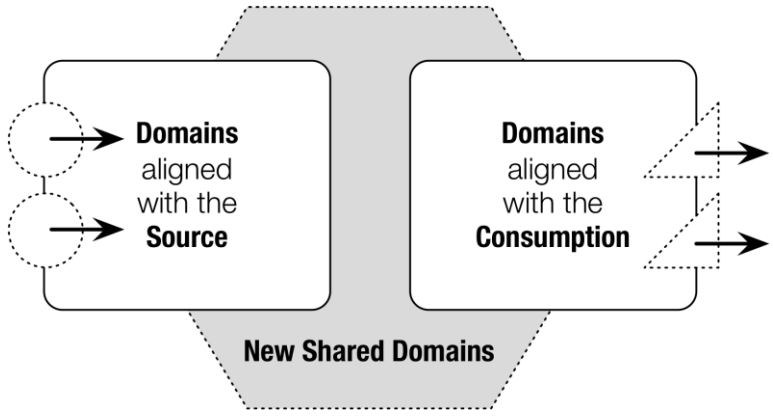


Figure 5. Structure and interaction of domains at Data Mesh. Adapted from (Dehghani, 2019)

Considering this division of domains, the data pipelines are made internally within each domain (Dehghani, 2020a). Therefore, there will be a distribution of the data pipeline steps within each domain. At this point in the process, one of the Mesh crucial points

arises: the quality of the data. It is up to each domain to establish its service quality level, making available to its consumers the quality that its data holds, as well as its timeliness, error rate, specification, among others (Dehghani, 2019). In short, the aggregation phases of a centralized data pipeline are here migrated to the implementation details of each (shared) domain.

The fulfillment of the second concept of the Data Mesh, "Data as a Product", implies that there is a set of characteristics that are held by the data (Barr, 2020), since this concept intends to maximize the quality of the data (and as a consequence of this, increase the satisfaction of the consumers) (Dehghani, 2020b). *Dehghani* defends that there are six principles that must be fulfilled to maintain the data quality and the effectiveness and efficiency of the Mesh. These principles (can be abbreviated as DATSIS principles) are the following (Dehghani, 2019):

1. Discoverable - all the data present in Mesh nodes/domains, must be present in a centralized data catalog. This registry must also contain the respective source, metadata, lineage, and a small sample. This way, there will be a centralized discovery service, transversal to all the Data Mesh (Dehghani, 2019).

2. Addressable – each data product must have a unique address that allows the access to it. The addresses of each data product must be unique and known in the Mesh (discoverable). To facilitate access to the various data products, common conventions should be created throughout the Mesh. (Dehghani, 2019).

3. Trustworthy - each owner of a data domain must provide the level of service quality. Moreover, they must also ensure that this level corresponds to the reality, the data is clean, and the metadata and lineage have been provided. In this way, there will be an increase in the consumers' confidence and an easier understanding of the node/domain and its assets (Dehghani, 2019).

4. Self-describing - the data must have an intuitive syntax and semantics and it must be aligned with the provided data sample. One way to achieve this is to use data schemas (e.g.: parquet) (Dehghani, 2019).

5. Interoperable - the data must be governed by global rules within the Mesh. To aggregate data from different domains, for example, it is necessary to establish a

correlation between the different data domains. In this sense, proper standardization of the data must be established - that is present in the global governance of the system – in order to avoid polysemy (Dehghani, 2019).

6. Secure - to avoid chaos and prevent Mesh failures resulting from misuse, it is necessary to establish rules regarding domain access. In the case of decentralized data domains, this control although it obeys a set of centralized standards, it can be different in each data product (presenting specific granularities) (Dehghani, 2019).

This orientation towards data as a product implies that there are new roles and management strategies in the teams that are responsible for data processing (Dehghani, 2019). Consequently, there are two new roles: domain data product owner and data product developer. The domain data product owner is responsible for the decision-making that focuses on the vision around their data product, i.e., its direction and possible capabilities within the organization. This role is concerned with the satisfaction of the consumers of their data products and it takes care of the lifecycle of these data products. The domain data product owner must also make their work measurable, making use of KPI such as lead time for data availability, data quality, among others (Dehghani, 2019).

Each domain will also include data product developers, being them responsible for building, maintaining, and serving the data product domains (Dehghani, 2020b). A more concrete example of their tasks is the construction and maintenance of internal data pipelines for each domain (Dehghani, 2019). Data product developers within a specific domain will work together with developers in another domain, and it is possible that the same domain team works for different data products. When compared to current and past paradigms, the responsibility model is thus reversed, since the responsibility over the data is now close to the source (Dehghani, 2020b).

Considering this concept, it is possible to define the ***Architectural Quantum***. This concept, by definition, consists of the smallest unit of architecture that can be deployed with high cohesion and includes all the structural components involved (e.g.: Code, Metadata, Infrastructure) (Dehghani, 2020b). Considering the definition of this concept, it is possible to define the architectural quantum as a data product in the Mesh – this architectural quantum can be visualized in Figure 6.

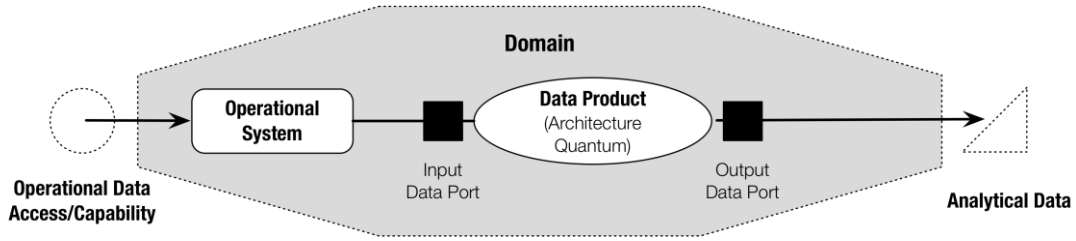


Figure 6. Domain: data product and operational system. Adapted from (Dehghani, 2020b)

A Mesh node consists of a data product, which includes three main components: code, data and metadata, and infrastructure. The code component encompasses three distinct segments. The first one refers to the data pipeline that transforms data, receiving it either from source domains or from other data products. The second one refers to applications that allow access to the data and metadata. Finally, the third one refers to the code used to reinforce access policies, among other related concerns. The data and metadata component are linked to the core of the analytical and historical data. The data within a data product can be of different natures (e.g., batch files, events, and graph), but for it to be used, there must be an association between the data and the respective metadata. This way, it is the metadata and semantics of the data that is used to maintain the governance of the Mesh, since they enable the correct interpretation of data and are also incorporated into data access policies. The infrastructure component, also encapsulated within a data product, allows access to the data and metadata, as well as running the code related to the data product in question (referred to as the "code" component in this paragraph) (Dehghani, 2020b).

In short, the common data engineering concepts, such as data pipelines and storage structures, are now combined to the data they handle, being an incorporated part of the data product.

One of the major concerns with self-serve data infrastructure is the duplication of efforts in the setup of the data pipeline engine (among others) by the domain teams (Dehghani, 2020a). To avoid this inefficient duplication, it is necessary that, when building the platform, business domain concepts are not considered, so that there is an abstraction of complexity. In this sense, the author points out some capabilities that this type of platform must provide, such as scalable polyglot big data storage, unified data access control and logging, and data governance and standardization (Dehghani, 2019)

This architecture can thus be divided into planes, being them remade into levels , which serve different user profiles and not architectural layers (Dehghani, 2020b). Analyzing the data infrastructure provisioning plan, this includes access control management provisioning, orchestration for the internal code of data products, query engine, among others. In a data product developer experience plan, there is a naturally different provisioning, as this plan is characterized by a high level of abstraction, related to the user function. In a Data Mesh supervision plan, there is naturally a series of capabilities that make sense to be made available on a global level, such as, for example, the ability to discover data products for a specific use case (Dehghani, 2020b). This division of planes can be better understood in Figure 7**.**

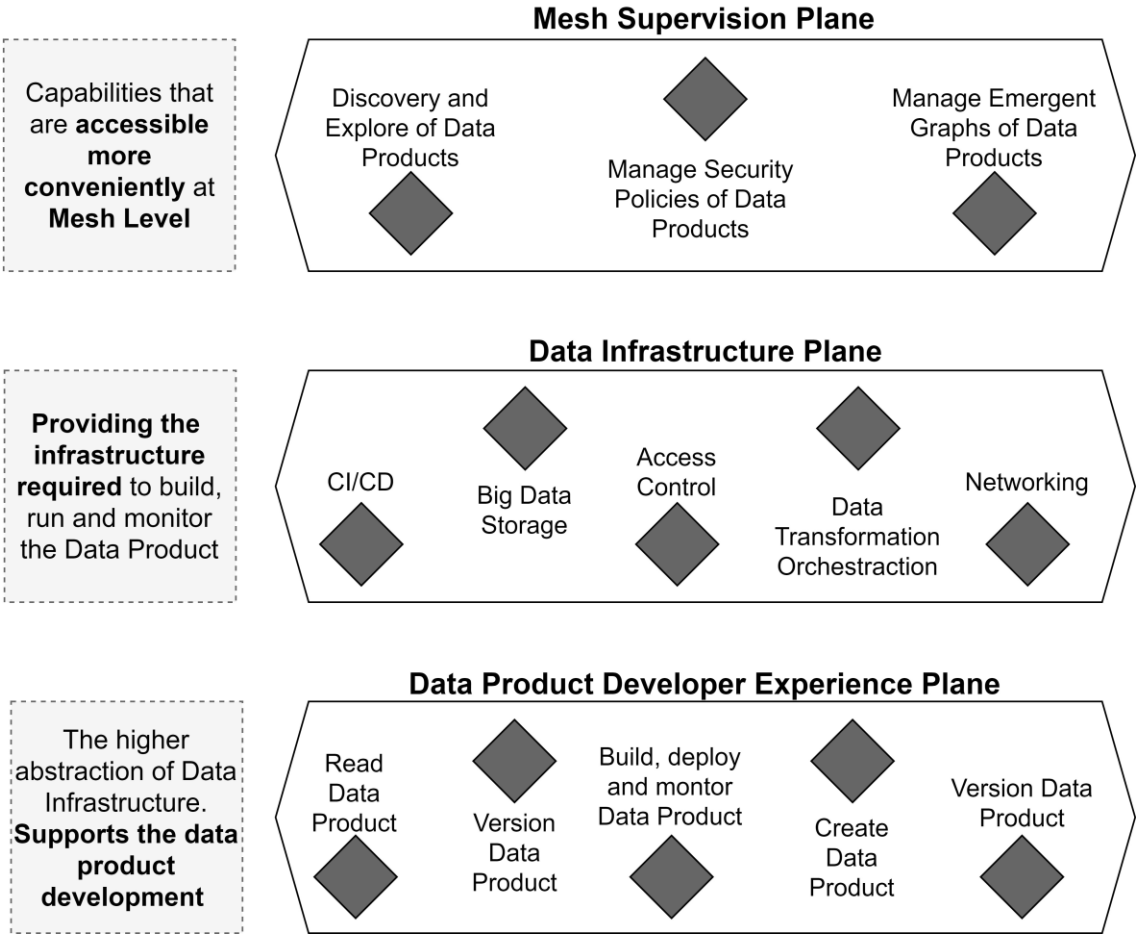


Figure 7. Planes differentiation in self-serve architecture. Adapted from (Dehghani, 2020b)

Dehghani also highlights that, although cloud usage effectively decreases operating costs and the effort required, there is no removal of the higher abstractions that need to be placed in business context. Thus, abstractions must be created that fit the context of the

business, and these establish the communication with the cloud services. (Dehghani, 2019). One criteria that can be used to measure the success of this type of infrastructure is the time required to create a new data product (Dehghani, 2019).

The governance model applied in the Data Mesh must balance two relevant dimensions: achievement of the measures imposed at a global level and respect for the autonomy of the various data domains that compose the Mesh. Therefore, when defining this governance model, there is the need to reflect on what must be defined on a global level (Mesh level) and what each domain should have freedom and responsibility to define. Figure 8, shows an example of these several elements that compose the federated computational governance model, highlighting global decisions and domain decisions.

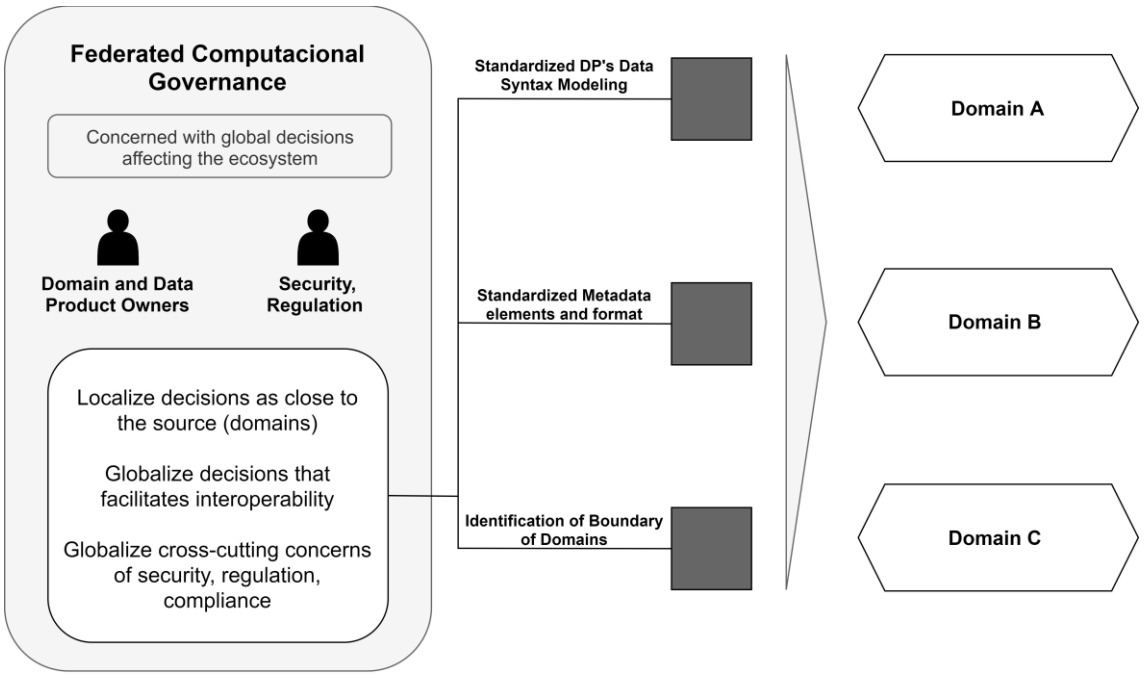


Figure 8. Example of distribution a Federated Computational Governance. Adapted from (Dehghani, 2020b)

Taking this into consideration, a part of the knowledge that is applicable to the current paradigms lose its meaning in the Data Mesh paradigm. For example, dataset only become data product, when within the domain itself it is subject to the necessary processes to obtain their quality and respect the rules of global standardization. This fact highlights the relevance of bringing, for the definition of the data product model, the domain data product owners, since they are the ones who know the domains most closely (Dehghani, 2020b).

2.4.2. APPROACH FOLLOWED BY ZALANDO

Zalando was founded thirteen years ago as a startup linked to online shoe sales and is now the leading fashion platform in Europe (Max Schultze & Arif Wider, 2020). Naturally, and given the nature of the E-Commerce business, there is a need to store, process and use petabytes of data per day. In this sense, the company first established a Data Warehouse capable of storing this data - however, limitations in terms of scheduling and a desire for greater flexibility in terms of infrastructure, led them to begin the migration to the Cloud (Max Schultze & Arif Wider, 2020). Later, they established a messaging bus that acted as a bridge to services that were no longer accessed directly (microservices). Through the analysis of this messaging bus, they realized that the data they needed and used was present in it, that is, they realized that this information was there centralized. These conclusions led them to opt for the establishment of a Data Lake (Max Schultze & Arif Wider, 2020).

Later, with the use of this Data Lake, some challenges began to arise: the lack of ownership over the data, the poor quality of the data already after its processing and the problem of organizational escalation (since by constantly increasing the sources of data production and the final consumers of the same, a bottleneck appears regarding the team itself) (Max Schultze & Arif Wider, 2020). Faced with these difficulties, and as a way of trying to overcome them, Zalando decided to build his own Data Mesh. To do so, they based themselves on Dehghani's work, and equated three concepts: product thinking, domain driven distributed architecture, infrastructure as a platform. Thus, they instituted a paradigm shift in their own organizational environment. Therefore, there were some core changes: i) evolution towards decentralized data ownership, ii) prioritization of Data Domains, in detriment of data pipelines, iii) vision of data as a product and not by-product, iv) institution of teams organized by domain-data with a spectrum of diversified functionalities, v) abandoning a centralized data environment (e.g.: Zalando) to an ecosystem of data products. As a (desired) consequence of these core changes, it was possible to overcome the bottleneck situation at the data team level (decentralizing this infrastructure responsibility, to a data infrastructure as a platform) and migrate from a monolithic data architecture (e.g.: Data Lakes), to an interoperable services environment (Max Schultze & Arif Wider, 2020). Following the above concepts, Zalando implemented the Data Mesh architecture present in the Figure 9.

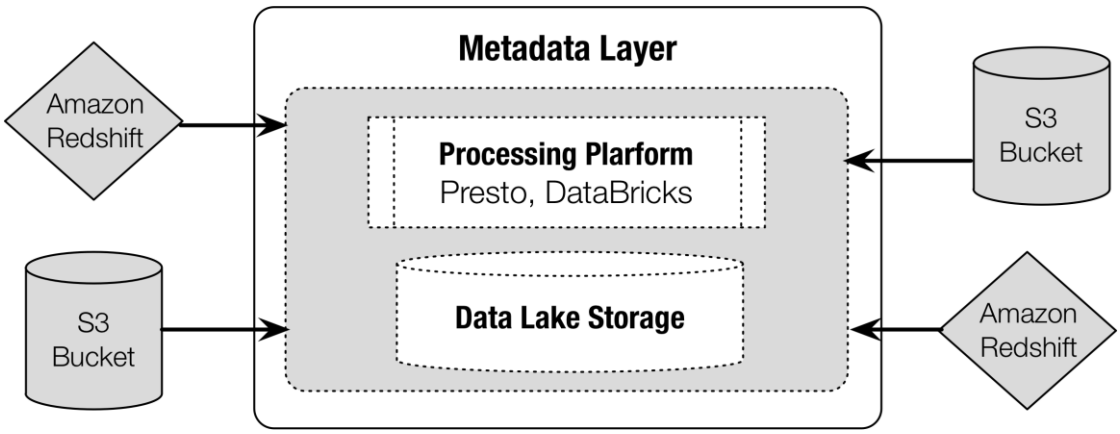


Figure 9. Zalando Data Mesh Architecture. Adapted from (Max Schultze & Arif Wider, 2020).

In a synthesized way, the initial central services (Data Lake Storage) were maintained and the metadata layer and governance that holds information about them were implemented, as well as enabling standardized processes about them (e.g.: data access). Zalando then created a concept of "Bring your own Bucket" (Max Schultze & Arif Wider, 2020). This concept allows users to integrate their S3 buckets with the data from the datasets working on their domains (teams) into the 'core' part of the infrastructure - the authors point out that AWS technology is very useful in this process for integration. Zalando has retained the central processing platform which uses technologies such as *DataBricks* and *Presto*. At this point the authors consider that they have achieved success in using self-serve infrastructure in an agnostic manner (Max Schultze & Arif Wider, 2020). Clusters (in this case Spark) are made available on this processing platform and the various users make use of these technologies, without the team responsible for the infrastructure having to know what the users do and without these users having to configure the clusters or know how complex they are. It is also possible to add more technologies to this architecture if there is a need for it at the processing level. The main goal was to achieve data sharing among the organization, something that was achieved with the use of this architecture, according to the authors (Max Schultze & Arif Wider, 2020).

Therefore, Zalando has an architecture where there is a decentralized warehouse, which uses a central infrastructure of data processing accessed by all. There is also a data ownership that is decentralized but makes use of a central governance, which allows the homeostasis of the system. All these components make use of the interoperability concept, which enables the creation of a self-serve platform (Max Schultze & Arif Wider,

2020). In this sense there are two key behavioral changes: treating the data as a primary concern and devoting resources to data quality assurance and understanding of its use (Max Schultze & Arif Wider, 2020).

2.4.3. Approach Followed by Netflix

Netflix is a company that provides a streaming service (e.g.: films, series), which currently serves about 150 million global users and is available on several different devices. Each time one of these users interacts with the application (e.g.: search for content to view) events are generated - events that need to be treated and stored to generate useful analytical value for the business and operation of the application. Due to the number of users worldwide, Netflix generates about trillions of events and with this, trillions of petabytes of data per day (Justin Cunningham, 2020). For this it has a data platform that is divided into three components: big data platform, cloud database engineering and real time data infrastructure. The first of these components deals with data storage, the second with cloud databases (as in *Cassandra*) and finally, the third component deals with real time data infrastructure (as in *Apache Flink*). Thus, the Data Mesh, in the case of Netflix, is described as a data processing system based on Apache *Flink* (Justin Cunningham, 2020).

The main objective linked to this topic (Data Mesh) is to make all the studios that work with Netflix (and its productions) into one system, capable of dealing with this large volume of data in an integrated and sustainable way - this process is currently being developed by the Netflix team (Justin Cunningham, 2020). Therefore, at the beginning of the process Netflix made a survey of the problems of data transport that they feel in their scope. They found five major problems, being: i) duplication of effort regarding the data pipelines and the teams; ii) unnecessary overload in the maintenance of the data pipelines (because they are poorly managed); iii) the lack of implementation of good practices throughout the various processes; iv) the need for lower latency; v) problems in the correction of errors (due to poor construction and lack of knowledge by users) (Justin Cunningham, 2020).

Currently, in its Data Mesh (still in a pre-alpha state as described by Cunningham), the team provides an infrastructure for the various users to develop data pipelines (Justin Cunningham, 2020). In this infrastructure they abstract the user from the complexities of the configurations, being the main concern to understand what the users want to do with

their data pipelines. In this infrastructure the user can use technologies such as *GraphQL* and *Apache Iceberg* (open-source project developed by Netflix, thought for huge table datasets that feeds your data warehouse). The user also has access to a metadata catalog, which he sees as a list of sources, from which he can choose to build his data pipelines. The user also has access to a list of process standards (which he can use as plugins to his data pipelines), already defined and established. This leads to the fact that few changes must be made to use this process, which once again abstracts the user from the complexity of the technology with which it interacts. Netflix is prioritizing the decrease of operational complexity over the associated cost and performance (Justin Cunningham, 2020). The Figure 10, summarizes the architecture currently used by Netflix.

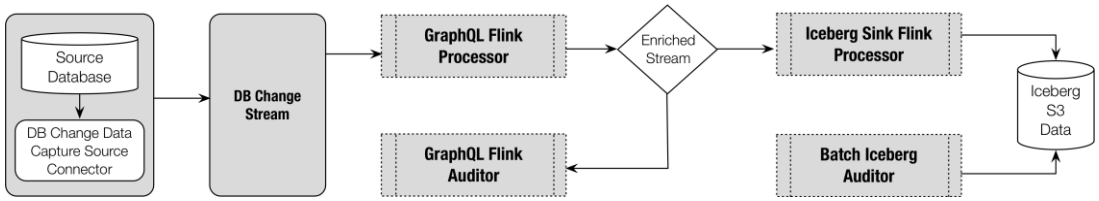


Figure 10. Netflix's implementation typology. Adapted from (Justin Cunningham, 2020)

In a simplified way, in the Netflix implementation typology, changes in the database are used to trigger the process. Once these changes are detected, the processor is given trigger (Graphic QL) that writes the entities of this stream in a table in the Iceberg Apache (to do so, it fetches these same entities from another Graphic QL, as illustrated in the figure). Note that Netflix implements this topology without the user having this notion, and implements an audit mechanism with the data source to verify its accuracy, at the processor's output, as well as in the batches of data stored in the Iceberg Apache (Justin Cunningham, 2020).

It is possible to conclude that, although Netflix does not present in its work the concepts implied to Data Mesh (e.g.: Data as Products), these are implanted in its Data Mesh architecture - a fact evidenced, for example, by the presentation of a metadata catalog, standard processes, quality assurance of the various data products, availability of a self-serve infrastructure, among others.

2.4.4. OPEN CHALLENGES

Although during the previous subsections, a literature review is presented on the topic of Data Mesh, this work lacks materialization and contextualization. Thus, there are no

models or methods that provide a practical implementation path to follow, by applying models that translate the Data Mesh concept. Regarding the implementations presented (Netflix and Zalando), these are use-case driven, proposing no methodological approach. Thus, throughout chapter three an approach for the design and implementation of a Data Mesh is presented.

3. Proposed Approach for the Design and Implementation of a Data Mesh

The third chapter of this master's thesis aims to present an approach for the design and implementation of a Data Mesh, including several guidelines that can be followed. Thus, throughout this chapter, three distinct models will be presented, which are respectively the domain model, conceptual architecture, and technological architecture. The purpose of these models is to specify the various conceptual and technological components that compose the Data Mesh, thus establishing relevant knowledge that can assist in its design and implementation, in the most varied contexts. First, the proposed domain model will be presented and detailed. It contains all the constructs that compose the Data Mesh and that should conceptually be present in it. Next, the conceptual architecture that identifies the generic components that should be included in the Data Mesh is presented, so that it obeys the four core concepts presented in the section 2.3. Finally, to assist the translation of the conceptual architecture into a practical implementation in real environments, the technological architecture is presented, composed of a diverse set of technologies that meet the needs of each conceptual component. This architecture intends to enable and facilitate the choice of technologies that may support the implementation of the Data Mesh.

3.1. Data Mesh Domain Model

According to Larman, the domain model is a fundamental part of the investigation of a problem, representing the conceptual classes that compose the same (Larman, 2004). Despite being widely used as a source of inspiration when it comes to software development, this type of model represents real-world conceptual classes and not software components. Thus, their representation is based on the presentation of domain objects or conceptual classes and their associations, and may also include attributes of the conceptual classes (Larman, 2004).

In this sense, the definition of a Data Mesh design and implementation approach begins with the definition of the Data Mesh domain model. The construction of this model is based not only on the knowledge acquired through the literature review on the subject, but also on the vision that the authors shared for the Data Mesh. Briefly, the construction of the domain model took into consideration the four main concepts of the Data Mesh

(domain-oriented decentralized data ownership and architecture, data as a product, self-serve data infrastructure and federated computational governance) and the fulfillment of the DATSIS principles. For this set of concepts and principles, conceptual classes were defined to correspond to the materialization of these concepts and principles. It is intended that, in this model, all the conceptual classes (or constructs) that should be considered in the design and implementation of the Data Mesh are present. Figure 11 presents the domain model built within the scope of this master's thesis, summarizing all the constructs that compose the Data Mesh.

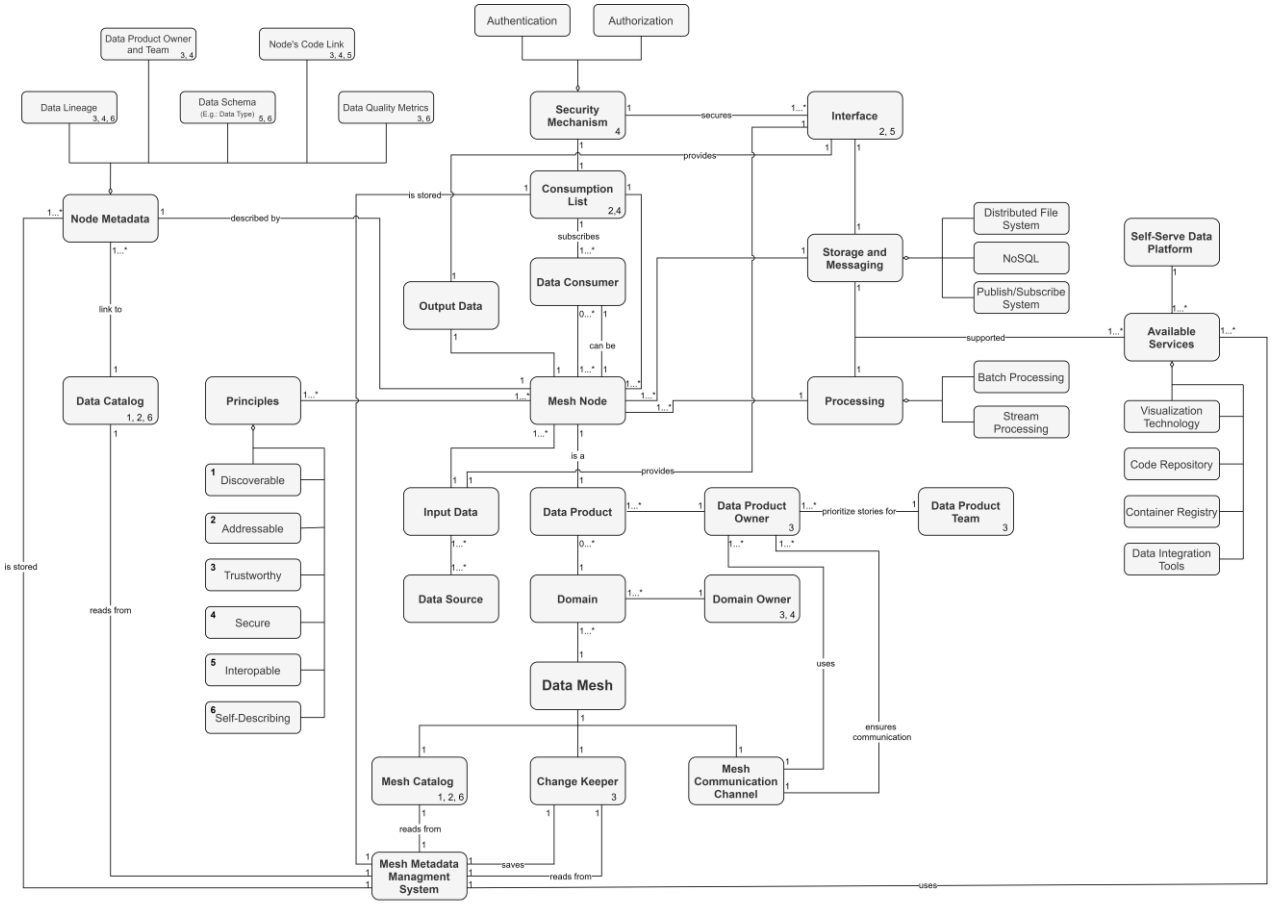


Figure 11. Data Mesh Domain Model

When looking at the Data Mesh from a high-level perspective, it is quickly possible to infer that it heavily relies in the connection and constant communication between domains in an organization. In this architectural context, the **Domains** are the agglomeration of various **Data Products** - each of these data products being a **Node** in the Data Mesh. These data products are in constant communication, being able to access

other domains' data (when necessary), also storing all their metadata and changes in the corresponding storage component (e.g.: Mesh Catalog and Data Catalog).

Although the domain model represents the conceptual classes that compose the Data Mesh, the **Data Mesh** itself is a constituent part of it, since associated with this central concept come several important related constructs. The same logic was applied to the principles that form the Data Mesh: discoverable, addressable, trustworthy, secure, interoperable, and self-describing data. Thus, a numerical notation was used in this model that correlates these principles with the various Mesh constructs (e.g.: in the data catalog construct it is possible to verify the number 1 and 2 that represent the "discoverable" and "addressable" principles respectively, since the use of a data catalog enhances them). In this sense, analyzing the proposed domain model, it is possible to infer that a Data Mesh comprises four components: one or more data **Domains**, a **Mesh Catalog** (catalog of the Data Mesh itself), a **Mesh Communication Channel,** and a **Change Keeper** component.

The DATSIS principles and the concept of governance are significantly relevant to detail how a Data Mesh should work. In this sense, there is the need to have a strong policy regarding the quality, reliability, and interoperability of the Data Mesh. Thus, it is essential that in a data architecture like this, there are components that guarantee this policy to its users. For this end, constructs such as the **Data Mesh Catalog** and **Change Keeper** are included. The **Data Mesh Catalog** is the component that allows to quickly discover which nodes (data products within a domain) exist in the Data Mesh, and what their general characteristics are - reducing data discovery time - pointed out as one of the main problems in current data architectures (Dehghani, 2020a). More than the speed of discovery, this provides a synchronous view of the Data Mesh at the level of its constitution, as well as information about the people responsible for each domain or data product. The Data Mesh Catalog is connected to a **Mesh Metadata Management System**, from which it reads the information about the nodes present in it. This repository not only stores information for the Mesh Catalog, but also combines information related to the changes that the Mesh is undergoing, from each node's Data and Metadata Catalog, and from each node's Consumption List.

Another proposed construct for the Data Mesh is a component that registers the various changes that occur in it (**Change Keeper**). Naturally, a business has a dynamic nature which is reflected in the data it daily generates, manipulates, and accesses. In this sense,

it is proposed to keep track of these changes, to inform all Data Mesh users about, for example, the creation of a new domain, or data product, or even the deletion of an existing one. This component should also register the changes that take place in the metadata of the various data products. The main idea and function intended is to minimize the impacts that arise in the data pipelines caused by the phenomenon of change. This construct also interacts with the Mesh Metadata Management System, storing and reading the various changes in it (as mentioned above). However, it is not enough that there is only one way for all the users to be informed about changes that occur in the Data Mesh.

In an organization, there is a large flow of emails and messages exchanged between co-workers. Naturally, these messages are of the most varied nature (e.g.: human resources information, corporate campaigns, scheduling meetings, among others) which leads to the fact that the answer is often delayed, and in the limit, many emails remain unanswered. Thus, when proposing the **Data Mesh Communication Channel**, it is intended that there will be a unique efficient communication channel, in real time, so that the various Data Mesh users (and experts in each domain) can quickly exchange impressions and doubts with other experts, potentiating an organized environment in which the response time is expected to decrease. This communication channel can be created using what is already a company practice, or if necessary, a new communication tool can be adopted. The main idea is that the adoption of this channel facilitates the process of communication and knowledge sharing in an organized and fast way.

In the Data Mesh, there can be several **Domains**, and it does not make sense that there isn't at least one. A Domain can be defined as being a distinct area of operation (e.g.: human resources and sales, when we talk about a business context) or subjects of a distinct nature (e.g.: fruit trees or deciduous trees, when in a non-business context). To have control over the operations developed in the various Domains, it is proposed that, for each Domain, there should be a person who understands it deeply in terms of scope, operationalization, and data. This Domain expert, who also manages the various data products that are part of it, is called the **Domain Owner**.

Within the same Domain, there can be several **Data Products** or at the limit even none – if a given Domain in an organization does not produce data, but only consumes it from other domains (for example). A Data Product can be understood as a set of data with analytical value, that is generated within a domain. Although in the literature review it is

suggested that a data product corresponds to a domain (Dehghani, 2019), in the present work it is clarified that, according to our perspective, several data products can exist in one domain. This proposal is made based on the analysis of the disparity in complexity that arises from organization to organization. For example, probably when facing relatively small organizations or problematics, a domain may effectively have only one data product. However, if we consider, for example, the domain sales of a large company like Amazon, it is possible to quickly infer that there will be several different data products (e.g.: related to different views/analysis on the sales process). In this sense, it is proposed that there can be some flexibility when it comes to establishing the number of data products per domain, and that each context may apply the rule that makes the most sense given the circumstances. So, considering all these aspects, it is possible to conclude that a data product is a Data Mesh node and vice-versa.

Just as a Domain Owner is established for each domain, it is necessary to establish an expert person for a given data product. This person is called **Data Product Owner** and, together with the **Data Product Team**, they build, support, and maintain a given data product. It is not mandatory that a data product team and a data product owner can only assume these roles for a specific data product - given the limited resources of an organization, it is natural that there is the need to allocate the same resources to different data products.

Each **Node** of the Data Mesh will contain a set of components from the overall architecture. That is, a node will gather all the components that are part of the Data Mesh architecture, and this composition will be replicated by each node throughout the Data Mesh - note that when the term composition is mentioned it is related to the conceptual architecture (e.g., processing service component) and not to the technological architecture, because from node to node there may be disparities in technological choices. A node in the Data Mesh (data product) will have four base constructs: Input Data, Output Data, Node Metadata, and Data Consumer (this last one been facultative). Any data product, even if aligned with the source or aligned with the consumption, will always have in its composition the **Input Data** that composes it. This Input Data can be data that originates from the operational and transactional nature of the business (and therefore aligned with the source), or data from nodes already created in the Data Mesh that are consumed by this new node (thus being a node aligned with the consumption). Naturally, a **Data Source** is always associated with this input, despite of its nature. Following the

same logic, each Data Product will also contain the **Output Data** that will be associated with the flow of the Input Data through the data pipelines developed in each case. This data composes the Data Product itself, that will be available for consumption by other Nodes of the Mesh.

To be able to develop data pipelines and store data in each Node (Data Product), it is necessary that each Node has access to a technological layer that allows the development of these processes. To accomplish this goal, each Mesh Node makes use of **Processing and Storage Components**. The processing component consists in the access and use of batch and stream processing tools, to meet the needs of each Data Product. In the Storage and Messaging Component, NoSQL databases, publish/subscribe systems and distributed file systems are used so that it is possible to store and make available data in the most adequate way according to their nature. Naturally, in both the processing and storage components, not all the components need to be used in the same Mesh Node. However, the Data Mesh should have a wide set of available technological options that allow it to cover the various scenarios relevant for the organization. This vast option of distinct technologies also makes it possible to adapt to the needs of each Data Product Team, matching their know-how with the available standard technologies within the organization, to make the process of building and developing a Data Product faster and more agile. Processing and Storage Components are supported by the **Available Services** in the Data Mesh, which compose the **Self-Serve Data Platform**. The purpose of this platform is to aggregate several services of different natures, to enable the construction and maintenance of the various nodes in the Data Mesh. Briefly, it includes the data Visualization Technology (so that each team can create the analytical dashboards that help in the decision-making process), Code Repository (so that all code produced is stored in a secure environment, prepared for team collaboration), Containers Registry (to be able to manage the container images required for each data product), and Data Integration Tools (to include several data sources in the same data pipeline environment). It is expected that this platform will not have a static character in terms of the set of technologies and functions that it includes, and teams may, depending on their needs, suggest for the Data Mesh initiative in the organization to approve and add new technologies and functions, that can be applied to the several Data Products.

As mentioned earlier, each Data Product may be consumed by other Nodes, so there is an association between Data Product and Data Consumer. When a **Data Consumer** node

needs to consume data from another node, it needs to have a permission to do so. The issue of having/granting permission to access data, related to the reading and usage of a specific Data Product, implies that there is a component that allows the permission request and its authorization, thus being proposed in this work a **Consumption List** construct. This list consists, briefly, of a mechanism that maintains the various accesses that the Nodes have to each other. It is proposed that a given Data Product Team, when faced with the need to consume data from another Node, request this permission through the Consumption List. Once the request is made, it can culminate in the acceptance or rejection of this request. There is also a **Security Mechanism** that secures the data **Interface** itself, which provides the input and output data for each node. It is intended with this proposal that there is a security concern regarding the data that is used in each Node.

In the Data Mesh paradigm, it is relevant to have higher quality assurance in the data flowing between the nodes of the Data Mesh. To accomplish this, it is important that the **DATSIS principles** are guaranteed for each component and flow (as illustrated in the domain model, Figure 11). Thus, each Node will be associated with the DATSIS principles, ensuring that, in the Data Mesh as a whole, data is discoverable, addressable, trustworthy, secure, interoperable, and self-describing.

For data to be mainly self-describing, it is necessary that each Node of the Data Mesh is associated with its own metadata. The **Node Metadata** consists of the agglomeration of several characteristics of the data, such as Data Lineage, Data Schema, and Data Quality Metrics (that enhance trust in data). In addition to this Metadata, Information about the teams that create and maintain each Data Product is also relevant to be included, as well as Links to the Code used in the construction and data pipelining of a node (allowing the evaluation of the degree of trustworthiness related to a Data Product). All this information is thus stored in the Mesh Metadata Management System. Naturally, these characteristics are present in a Data Catalog that reports the specifications of each Node, being fed by the information in the above-mentioned management system. The **Data Catalog** is extended with new information parameters (e.g.: domain to which the data product belongs), and the same can be used by everyone in the organization to discover data products (due to the availability of metadata). More than just accessing this information, with the proposed domain model, distinct teams in disparate domains of an organization will be able to consult this data, make use of it, and build new Data Products (Mesh

Nodes), each choosing the available standard technologies that best suit their case, without jeopardizing the interoperability of the system itself or harming the quality of the data that is handled and produced.

In short, no longer there will be a centralized data team, overloaded with requests from an entire organization, transitioning to a decentralized reality of these teams across various organizational domains. The data will then be handled closer to its creation point, by teams that know their organizational domains intimately, including the nature of the data when it is ingested.

3.2. DATA MESH ARCHITECTURE

More than defining the domain model, synthesizing the Data Mesh constructs/conceptual classes, it is necessary to build an artifact that makes the concept more tangible at the implementation level. In this sense, a conceptual architecture and a technological architecture are proposed for the development and implementation of a Data Mesh.

3.2.1. Conceptual Architecture

The main purpose of the conceptual architecture is to present the standard architectural components that compose the Data Mesh. In this way, in a more advanced phase of the Data Mesh implementation, these components will be implemented through specific technologies that allow the proper functioning of the Mesh.

The conceptual architecture preserves the four core concepts and DATSIS principles on which the Data Mesh is based, having been at the base of its creation, the domain model presented in 3.1. In this sense, the conceptual architecture here presented can be divided into four parts being them respectively: i) the organization between nodes, catalogs, and repository of the Mesh; ii) self-serve data platform; iii) infrastructure; and iv) security mechanism. Figure 12 presents the proposed conceptual architecture.

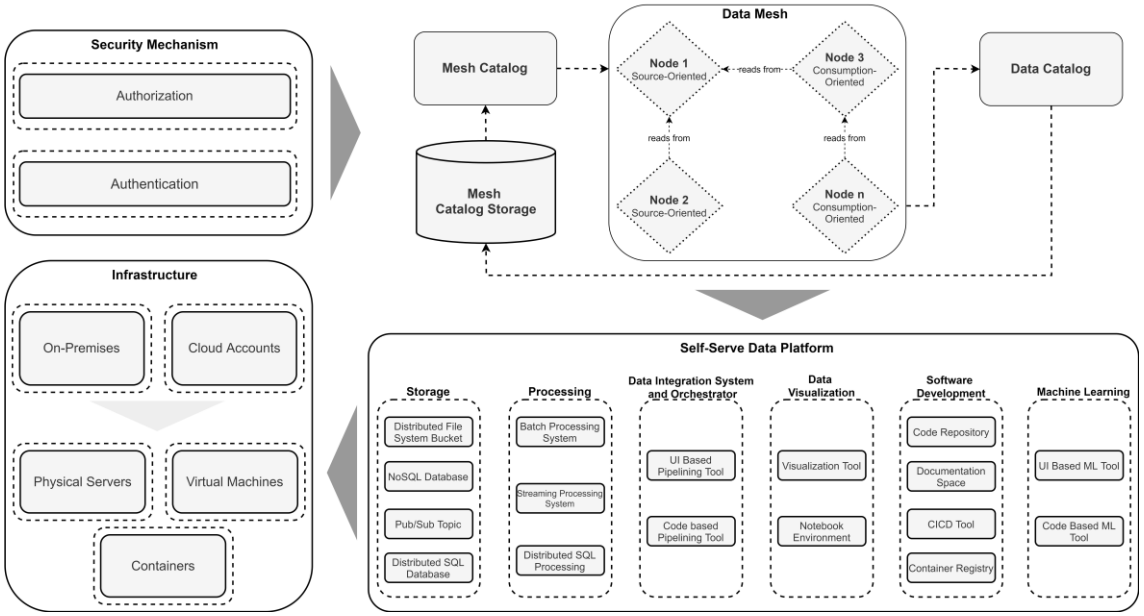


Figure 12. Data Mesh Conceptual Architecture

In the proposed architecture the articulation between the various Data Mesh nodes is presented in a way that is possible to read data from each other (thus highlighting the interoperability aspect in this architecture). As illustrated in Figure 12, these nodes can be of different natures (depending on their alignment with source or the consumption), working in a network to meet the data needs of the organization.

As also illustrated in the figure, the entire Data Mesh is connected to a component called the **Mesh Catalog**. This catalog has the purpose of presenting the Data Mesh's metadata, allowing users to quickly and effectively discover the nodes (data products) that compose it. Ideally, this catalog component should also contain other information such as, for example, the identification of those responsible for each node. This catalog stores and reads all this information from the **Mesh Catalog Storage**, which works as a central storage component for the Data Mesh - allowing for its adequate management. At a lower level, each node that composes the Data Mesh is present in the **Data Catalog**. This catalog differs from the one previously presented, because it details the characteristics of the data itself present in each node (data schema, data product owner and team, data lineage, data product's code links, and data quality metrics), allowing a new user of the Data Mesh or, even other data product teams, to discover the data quickly. Like the Mesh Catalog, the Data Catalog stores and reads its data from the Mesh Catalog Storage component. So, in this first architectural part, the issues regarding the articulation between nodes and their

catalogs are unblocked. However, there is still the need to establish the following: i) how the Data Mesh articulates itself in terms of security (for example, whenever a data team needs to consume data from a node already built); ii) what infrastructure supports all the above-mentioned relationships between nodes (data products); and iii) how to access the technologies that allow building and maintaining these nodes.

To answer the first question raised in the previous paragraph, an architectural component concerning the Data Mesh **Security Mechanism** is proposed in the conceptual architecture. To avoid chaos, it is necessary to have a governance layer in Data Mesh so that, for example, when a given data product team needs to consume data from a data product that already exists, it can consume it efficiently. For this end, as previously seen, the domain model proposes a consumption list that unfolds, in the conceptual architecture, into a security mechanism implemented in two phases: authentication and authorization (this mechanism can be implemented on-premises or in the cloud). Data Mesh users request access to a Data Mesh node, authenticate themselves and finally they are granted authorization to access it (if applicable). The way this authorization is deliberated must be based on the need for a given team to consume/access a specific data product (using, for example, permission management technologies like Apache Ranger). Once this authorization is conceived, the user (e.g., a member of a data product team) can access the output data from a given Mesh node and can consume it in another node (build a consumption-oriented data product) to satisfy the analytical needs within a given domain.

However, there must be the tools that enable the creation and maintenance of the various data products (nodes) in the Data Mesh. One of Data Mesh's core concepts is related to the possibility to adopt different technologies to suit the needs of each team, allowing the abstraction of the complexity of the infrastructure that provides those technologies. Thus, a **Self-Serve Data Platform** is proposed in the conceptual architecture. This platform aims to provide all Data Mesh users with a set of technological components that allow them to build and maintain their data products in an interoperable way. At the limit, if none of the technologies made available on the platform meets the team's needs, more technologies can be adopted, after an approval process, to allow each team to adjust their development and implementation expertise to the needs of their data products. This approval process, according to (Dehghani, 2020b), must be done by the team that administrates the platform at the infrastructure level. This platform can be functionally divided into six distinct parts, which are: storage, processing, data integration system and

orchestrator, data visualization, software development, and machine learning. In essence, this represents the aggregation of the functionality needed to create, test, and maintain the various data products and their visualizations (e.g.: data analytical dashboards).

The group of storage functionalities, divided into four distinct parts, is intended to satisfy the most diverse needs of data **Storage**, considering their nature. It is proposed that the Data Mesh Self-Serve Data Platform includes: i) buckets (or folders) in a distributed file system; ii) NoSQL databases (for non-relational data); iii) publish/subscribe topics (for data flows using brokers and likely streaming data); and iv) distributed SQL databases (for providing distributed storage and processing capabilities for relational data). It should be noted that it is not implied that a Mesh must mandatorily include in its platform all these components, as each Data Mesh can and should be designed to cover the features required for the data needs of the specific organization or context.

The data **Processing** component should include technologies that respond to the various types of data to be handled, such as large volumes of historical data or real-time data. Thus, it is ideal that the processing component provides batch processing, streaming processing, and distributed SQL processing tools. Again, this diversification of technologies across functionalities allows one source-oriented data product team, for example, to process in real-time the events associated with customers on an e-commerce platform, and another consumption-oriented data product team to consume this data and merge it with data from other nodes, using batch processing.

The **Data Integration System and Orchestrator** component is essential for building the data pipelines, allowing the flow of data from its source to each node's storage. To do this, the Self-Serve Data Platform must provide this component, allowing it to be subdivided into two distinct segments: UI based or code based. In this case, the division of technologies is done according to how the users prefer to develop their data products: either by code development or through UI elements. In certain teams of the organization, there may be a more robust level of expertise for code-based data pipelines, but the opposite can also happen. In this sense, and always with the view of bringing the teams closer to the solutions that make them more productive and effective in the construction and maintenance of data products, the UI-based component is also included. This type of technology abstracts the complexity behind the various components from the end user,

avoiding the complete development of the solution through code - which can make the process easier for the various data product teams.

Of course, transforming the data and making it more reliable is important (so that it responds positively to a range of data quality metrics), but for decision-making and for retaining the analytical value of the data, the data needs to be analyzed. Thus, the **Data Visualization** capabilities are significantly relevant, reason why the Data Mesh Self-Serve Data Platform needs to enable such capabilities. Once again, there is a division of the technological components, based on the experience that best suits each team, and this visualization can be achieved through a dashboarding tool or a notebook environment. Ideally, a Data Mesh will contain both development options, allowing teams to use these tools to meet broader needs when it comes to visualizing the data they produce and consume. In this way, a business user can use the self-serve data platform to consume data from a node into a dashboarding technology (present in the platform) and develop reports to assist the decision-making process, or explore the data in an *ad hoc* notebook environment.

Clearly, the development and maintenance of the various nodes of the Data Mesh is closely related to some **Software Development** components and therefore, there is a range of features that must be gathered in the Self-Serve Platform for this process to be possible. The first component has as its purpose the organized storage and versioning of all the code produced to create and maintain a node, hence the inclusion of a code repository is suggested. Next, within an organization, it is necessary that the various processes are properly documented and that there is knowledge sharing between teams. In this sense, a documentation space is proposed. Still on the software development component, it is necessary that the code is properly tested and delivered, before a solution is made available to the entire organization (in this case, for example, before a node is published in the Mesh), reason why Continuous Integration and Continuous Deliver (CICD) Tools component is also included. Finally, a container registry component is suggested, to store and make available the various images used by the container-based solutions in the Data Mesh.

Machine Learning is a method that enables the creation of predictive models through data analysis (Janiesch, Zschech, & Heinrich, 2021). Over the time, machine learning has evolved and proven to be significantly valuable to complement data analysis and elevate

it to the level of predicting future events. In this sense, there is a machine learning component in the platform, divided into code or UI-based tools.

The combination of the six components explained above (storage, processing, data integration system and orchestrator, data visualization, software development, and machine learning) make up what is proposed as the Data Mesh Self-Serve Data Platform: a platform where the various data product teams can access and use the tools that best suit their needs to obtain well-designed and implemented data products.

For the Self-Serve Data Platform to exist and function as proposed, an **Infrastructure** to support it needs to be in place. The implementation of this infrastructure can follow two distinct paths considering the organization's resources (both hardware and software, and the expertise of its teams): on-premises or in the cloud. Today, the cloud provides a wide range of services that allow organizations to implement the most varied functionalities in terms of data storage, processing, visualization, among other tasks (AWS, 2021; Google, 2021; Microsoft, 2021). However, tools that are not covered by these services but are needed can and should be included in the Data Mesh. On the other hand, the Data Mesh itself does not need these cloud accounts to exist, and its infrastructure can be solely based on on-premises infrastructure (physical servers, virtual machines, and containers). These should allow the configuration of clusters capable of containing the technological components, and thus create the necessary conditions for teams to use the tools required to build the Data Mesh. The infrastructure implementation option (on-premises or cloud accounts) should be a singular decision in each organization since both bring advantages and disadvantages. In this work, we do not aim to detail those advantages and disadvantages, but, for example, although the cloud attracts users due to its scalability, the ability to develop complete solutions with cloud accounts requires some level of expertise from the self-serve data platform team and the data products teams, which may not exist inside the organization.

Thus, this decision should be well considered in each organization, always seeking to take advantage of the existing infrastructure and skills, or seeking to adopt a radically different infrastructure that will raise the potential value that can be brought to the company. Nevertheless, what’s relevant is for the Self-Serve Data Platform to meet the expectations of the Data Mesh initiative and the data product teams.

On an on-premises perspective it is important to consider that it is not mandatory that there is only one cluster that hosts the Data Mesh, quite the contrary. The relevant aspect to consider is that the interoperability between nodes is always guaranteed, so that the "micro-architectures" that support each data product's design do not make it impossible for other nodes, based on other Data Mesh-compatible technologies, to access their data, making possible to accomplish the vision of the Data Mesh. The same holds true when multiple data products are supported by different cloud accounts belonging to the same organization.

3.2.2. Technological Architecture

Naturally, the conceptual architecture needs to be unfolded into a technological architecture, aiming to present a wide range of technologies suitable to implement the components of the conceptual architecture. Figure 14 correspond to the bridge established between the proposed conceptual architecture and the technologies that are feasible to implement it.

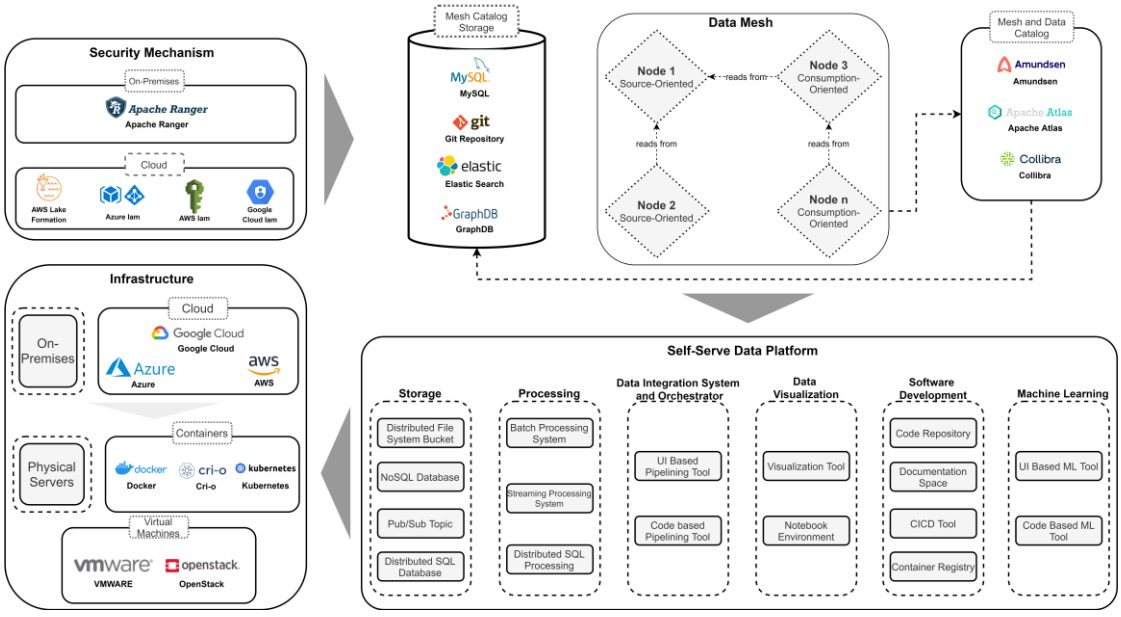


Figure 13. Data Mesh Technology Architecture

Due to the broad scope of the architecture and the wide set of available technologies, the technological architecture has been divided into these two figures (which can be seen as two parts of the same figure). The first part, figure 13, presents the technological architecture of Data Mesh as a whole, leaving the Self-Serve Data Platform component unspecified. The second part, Figure 14 presents in more detail all the technologies that

make up the Self-Serve Data Platform. It is intended that, with the presentation of the technological architecture, practitioners have available an starting point for the implementation of a Data Mesh.

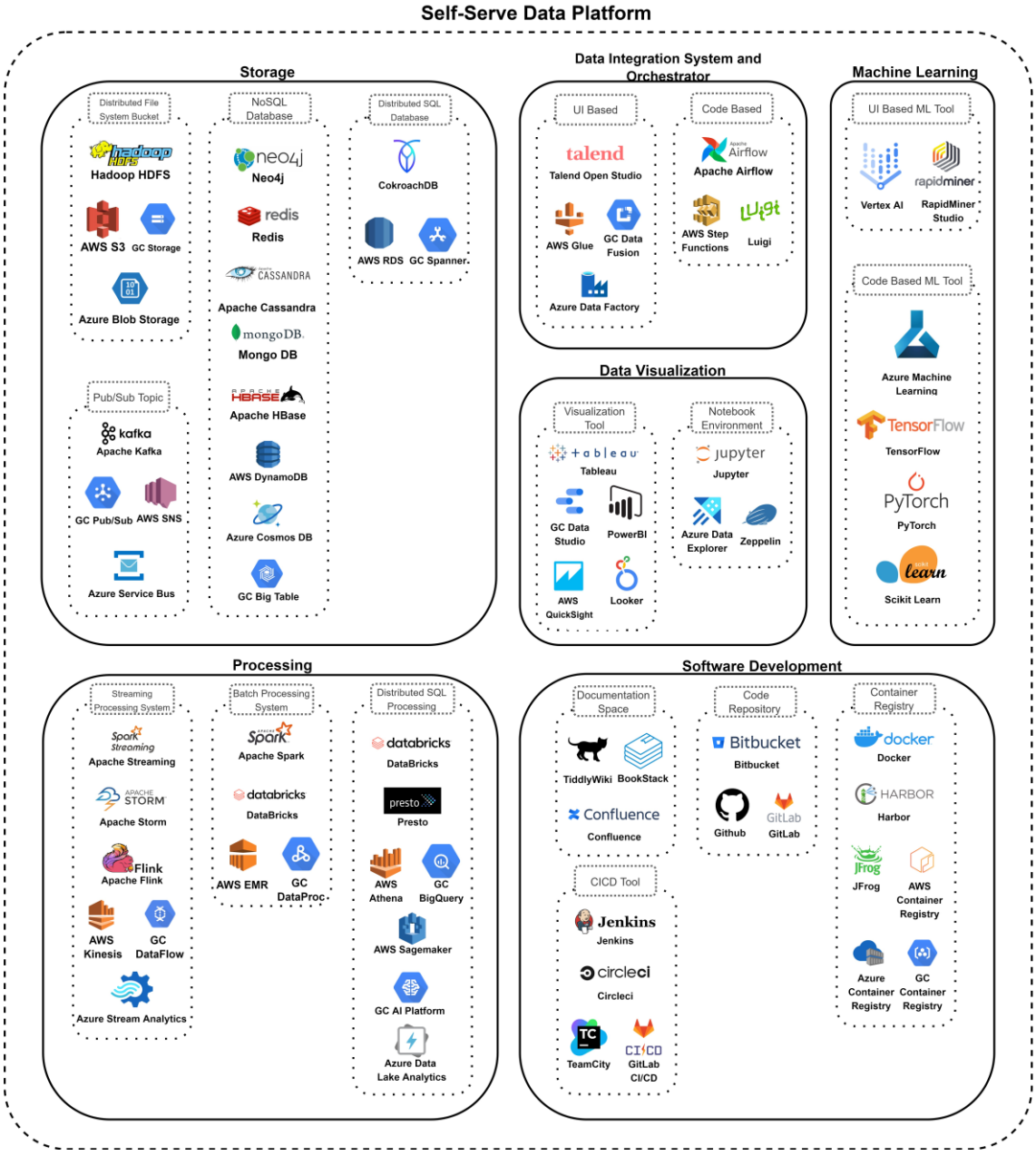


Figure 14. Self-Serve Data Platform in Detail

It should be noted that there are currently several other technologies on the market that can respond positively to the needs of the Data Mesh implementation, several of them listed in the figure, but many of them left out. The choice of technologies integrated in the figure was based on two distinct criteria: 1) being open-source technologies (which implies a significant impact in reducing costs in the implementation of a Data Mesh) or

2) being services available in the Cloud (in AWS, Azure, and Google Cloud). In this sense, several Data Mesh implementations can be outlined, using combinations of the technologies presented in Figure 14. Each organization should reflect on the solutions presented and depending on the balance of their needs and resources, choose, amongst the various options, the one that will make more sense to use (in terms of cost, performance, and implementation difficulty). In the next chapter of this master's thesis, the architecture of the proof-of-concept will be presented, which, similarly to the decision process mentioned above, will contain the most suitable technologies given the resources available for this research work. Analyzing Figure 13, we can see that several technological solutions are proposed to implement the conceptual components shown in Figure 12. Due to the vast amount of technologies, the description and explanation of the figures provided throughout the rest of this section will only highlight some of the technologies in the figure. However, as they are competitors to implement the same conceptual component, the understanding should remain consistent no matter the technology serving as example to detail a given technological architecture component.

The security mechanism contains solutions divided between on-premises (Apache Ranger) and the cloud (e.g., Azure, AWS and Google Cloud IAM). These solutions will make sure that the various nodes of the Mesh can securely interoperate with each other. These Mesh Nodes can be developed with the resources from the technological solutions presented in Figure 14. For example, it is possible to construct a Data Mesh node as being a folder in Hadoop Distributed File System (HDFS) or a bucket in S3. To feed this node, there are data pipelines using Apache Spark or AWS EMR. Each node can also use HBase or AWS DynamoDB to store the output data of a node in a NoSQL database if the Mesh Node wants to expose data to online applications, for example.

Compared to the conceptual architecture, a difference is highlighted: the merging of the Mesh Catalog and Data Catalog. This agglutination is due to the fact that the proposed technologies are the same, so this agglutination is used to facilitate the interpretation of the figure and not having repeated components. Thus, technologies such as Amundsen, Apache Atlas and Collibra can be used either as Mesh, or Data Catalog. The Mesh Catalog Storage can use various technologies such as MySQL, Git Repository, ElasticSearch or GraphDB, depending on the nature of the data. However, it is also possible to use as a storage system the databases that are already integrated with the solutions (e.g.: default Apache Atlas database).

At the infrastructure level, technologies are proposed in two distinct ways: on-premises or through cloud accounts. The choice of implementation path must be carefully weighed in considering the complexity of handling the infrastructure itself. Although the cloud is significantly attractive due to its centralization of services, with high usability, scalability, and interoperability, it is necessary to consider the complexity of handling these systems securely. However, given the technology offerings in terms of infrastructure, security engine, and self-serve data platform, it is possible to implement an entirely cloud-based Data Mesh. Solutions are also available to create an on-premises infrastructure if that is the path practitioners want to follow. In this case, this implementation involves three components: physical servers, virtual machines (where VMWare and OpenStack can be highlighted for their capabilities), and container-based solutions (such as Docker, Cri-o and Kubernetes). This infrastructure implementation can also be combined with self-serve data platform technologies that share this on-premises compatibility (e.g., use Apache Ranger for the security mechanism, with data pipelines in Apache Spark, writing files in HDFS, feeding a Hive database for each Nesh Node, and being the metadata readable through Apache Atlas).

The self-serve data platform is, as explained in the conceptual architecture, a set of six base components, which agglomerates a wide range of technologies. The main purpose of presenting such a wide range of solutions is to provide a range of technologies capable of meeting the needs of its users when faced with a certain design and implementation context. Thus, with data product teams at different levels of specialization, all of them will be able to design, create and maintain their data products in the Mesh. Moreover, in case there is no technology on this architecture that meets the specific needs of a given team, new technological components or technologies can be easily added to it.

Although all the technological components of Data Mesh are essential for its proper functioning, there are two factors of major relevance in this architecture: data discovery and data access. These two factors are fundamental for the Data Mesh to work according to its four core concepts and DATSIS principles (section 2.4.1). Therefore, it is recommended for special attention to be given to these components. The choice of the appropriate data catalog is a key part of the Mesh, possibly one that allows its model to be extended to meet all the constructs of the domain model. Without being able to obtain a rich data catalog that allows a new user to know the domains, data products and how they are built, the Data Mesh can be seriously jeopardized.

4. DATA MESH PROOF-OF-CONCEPT

Although the domain model and architecture (both conceptual and technological) are the result of several insights emerged from the literature review and also from technical experience within the area, it is necessary to have a proof-of-concept of the proposed models to demonstrate their suitability for the implementation of a Data Mesh. Taking this into account, the proof-of-concept that was developed, will be detailed in this chapter. To accomplish this, it will be explained its scope, the used infrastructure, the technologies that support the creation of data products, the extensions made to the tools to meet the needs of a Data Mesh, as well as an example of the development of a data product throughout the entire process. It must be taken into consideration that given the resources available for the completion of this master's thesis, this proof-of-concept focuses on aspects considered as fundamental for the Data Mesh (e.g.: the part related to data catalogs and data products), leaving some components with their fully implementation planned for future work (e.g.: security mechanism within the Data Mesh).

4.1. SCOPE AND ORGANIZATION OF MESH NODES

Let us start by defining the scope in which this proof-of-concept takes place, namely the retail area. To illustrate a case that comes close to the requirements of a real market, the scope of the proof-of-concept will cover three domains of what is assumed to be a furniture commercialization company, in the business to costumer (B2C) segment.

Thus, three domains will be created and published in the Data Mesh, concerning three domains of the business in question, being respectively: Sales, Production, and Financial Management. In the case of these three domains, it is considered that each one has one data product that originates one mesh node, being them: Online Sales, Product Cost, and Profits. Considering the above-mentioned domains, there are two source domain (Sales and Production) and one consumption domain (Financial Management), since the join of the Online Sales data and Product Cost data, will originate the calculation of the Profits generated in each sale.

Note that, in this proof of concept, the aspect to emphasize is the flow of data throughout the architecture, and not the complexity inherent to its transformations - since this can be as complex as the user wishes (making use of the scalability and response to complexity

that the self-serve data platform presents). Figure 15 summarizes the organization of the domains and data products.

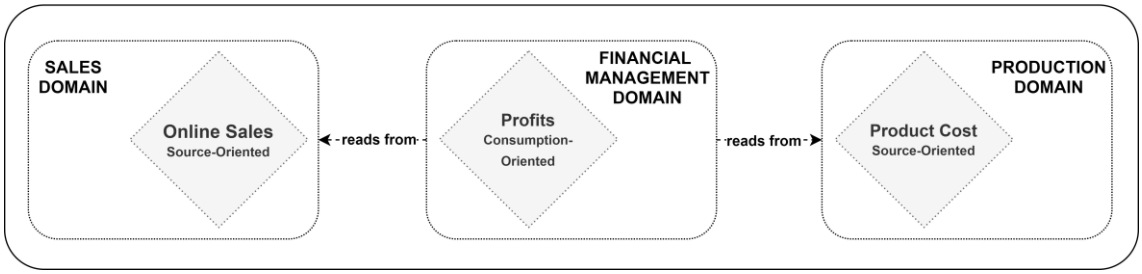


Figure 15. Data Products and Domains Organization

Looking in more detail at the data and metadata level for each data product, they have the schema illustrated below in table 1, 2 and 3.

Table 1. Schema of Data Product Online Sales Data

Source-Oriented Data Product: **Online Sales**	
id_sale	Integer
id_costumer	Integer
id_product	Integer
amount	Integer
sales_price	Double
payment_type	Integer
date_time	Timestamp

Table 2. Schema of Product Cost Data Product

Source-Oriented Data Product: **Product Cost**	
id_product	Integer
cost	Double
production_factory	String
warehouse	String

Table 3. Schema of Data Products Profits Data

Consumption-Oriented Data Product: **Profits**	
id_sale	Integer
Price	Double
cost	Double
profit	Double
date_time	Timestamp

The data sources used in this work are in CSV format, stored in a distributed file system (in this case HDFS), which include the record of sales made online and the listing of production costs for the various products that were sold. Note that there is no data source for the profits data product, since it is a data product based in a consumption-oriented domain and, in this case, it will be built based on the consumption of data from the two data products mentioned above (which does not mean that a consumption-oriented domain must be strictly built based on another data product's data and therefore cannot include its own data).

4.2. PROOF-OF-CONCEPT INFRASTRUCTURE

In a real context, a Data Mesh may be implemented in more than one cluster (in the case of this proof of concept, and due to the available resources, it is implemented within only one Hadoop cluster). In this sense, most of the technologies used in this proof-of-concept are present in a Hadoop cluster, which does not invalidate the usage of technologies outside this scope.

An on-premises approach was followed, using the Lynx Lab cluster from the University of Minho. In this cluster, a Docker container was created, and, inside this container, an image of the Cloudera sandbox was instantiated, containing the Hadoop technology. Later, it was necessary to configure the various needed services (e.g.: Apache Atlas). Each Data Mesh node corresponds to a folder in HDFS, and each one contains tables stored in an isolated Hive database. To develop the data pipelines, we simulated a context in each data product team makes use of Zeppelin Notebooks, using the programming language that better fits the team needs. Once the output data for each data product is ready, Apache Atlas (extended in this work to accommodate the Data Mesh requirements)

is used to allow all the cataloging of the Data Mesh's data. In this case, the Apache Atlas repository serves as the Mesh Catalog Storage itself (due to the optimization of resource usage). For data visualization, besides being possible to use Zeppelin Notebooks, Power BI Desktop and Server were used to create and publish the visualizations. To meet the premises of the domain model, regarding the assurance and monitoring of data quality metrics, a script of automated analysis of data quality was created and indexed to each notebook, and its information transformed into a dashboard in Power BI (so that its interpretation is more interactive).

All the code created by each data product team, must be documented, and stored. Therefore, GitHub was used as a tool for storing and making code available. In order to have a direct communication channel in this proof-of-concept Data Mesh, Slack was used, and direct channels were created between users and data teams/owners of the various data products. Regarding access (consumption list), a form was created, through Google Forms, so that this access can be requested, and this information is forwarded to those entitled to it (e.g.: data product owners who can grant authorization to access their data product's data). Ideally, this process should go through Apache Ranger, however, given the available resources for this work, it was not possible to perform this integration, being however the logical part of the process (the data consumption request) present in this proof of concept. The following sections will present the implementation details of the various components of this proof of concept.

4.3. HDFS Folders Organization for each Mesh Node

As illustrated in the domain model (section 3.1), it is possible to infer that a domain can have more than one data product. Revisiting the concept of domain, it is possible to understand domains as being the structured division of the organization by different areas of operation (such as sales, customer service, logistics, etc.). However, depending on the size and complexity of the organization in question, domains may have more than one data product (and in the limit, one domain of the organization may not generate any data product that is relevant to be shared with the entire organization). Thus, the same domain may have zero or more data products within its scope.

Therefore, with the adoption and implementation of the Data Mesh, there must be an imminent concern to ensure that the various domains, and the teams that build and maintain these domains, can communicate with each other efficiently. For this reason,

and as explained in previous sections, there is the need to ensure a governance approach that extends to the entire Data Mesh and respects the DATSIS principles. To keep the implementation aligned with the previous principles, a standard organization of the Data Mesh Nodes is proposed, which should be followed by all the teams, to implement these best practices in their domains and data products.

In the case of this proof of concept, it is defined that the whole company shares a cluster and therefore the file system is also shared amongst all - in this case HDFS. In this sense, it is important to build from the beginning a structure that obeys a certain set of parameters and to establish rules for the creation of folders and for their designation. Figure 16 illustrates the structure proposed for the HDFS folders organization into domains and data products.

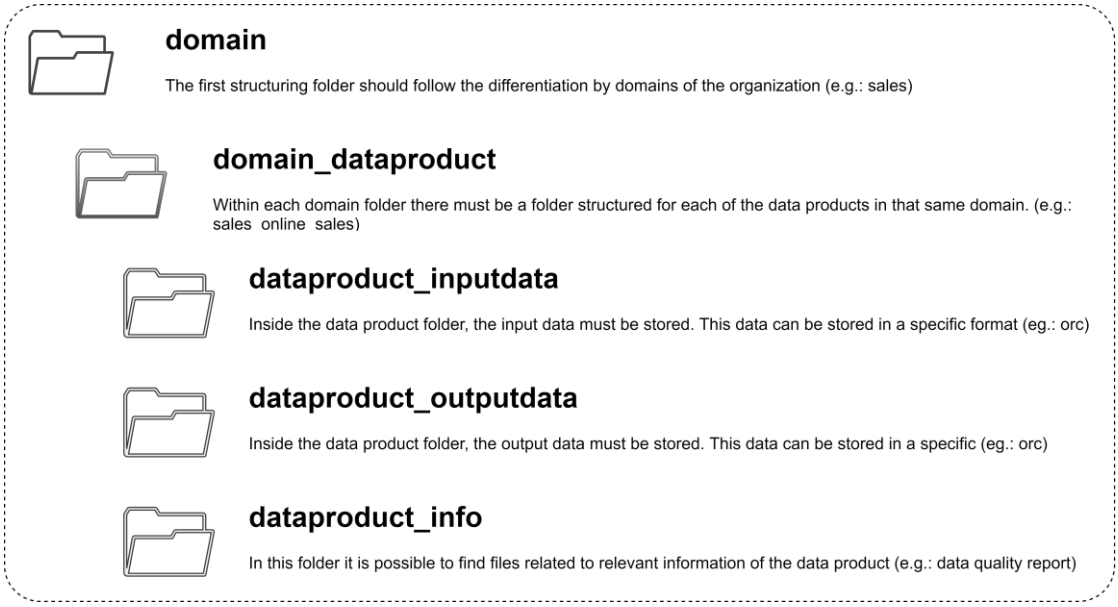


Figure 16. Folder Organization in HDFS

Considering the organization present in Figure 16, there is a brief set of rules to be followed, to guarantee the homeostatic organization of the Data Mesh. Firstly, it is necessary that the current data teams perform the exercise of dividing the organization by domains. Advisably, and given that the main purpose is that the nature of the data is not lost upon ingestion, it is important that this division into domains is aligned with the business itself and the people that work daily with this data. There may be as many domains as necessary, as long as they are always aligned with the operational nature of the organization.

Once the domains have been defined, it is important to identify the data products that compose each one. An example of this identification might be the sales domain. In an organization where there are two distinct sales segments (e.g.: physical stores and online stores), it will make sense for this domain to hold at least two distinct data products (one related to the operational data of online sales and the other related to the operational data of physical sales). Naturally, even though both are part of the sales domain, both are composed of different data, so it will make sense to analyze them differently. The initial data product definition will not be static, and more data products may be added along the way as the business develops. Thus, and considering the nature of this proof of concept, it is proposed that the domains are organized in distinct folders at the root of the file system, shared by the entire organization. This way, they will be quickly found and will follow an equitable distribution. This is also recommended to avoid semantic problems when handling data in the Data Mesh. To avoid these problems, it is proposed that all domains follow a similar naming pattern as illustrated in the Figure 16 - lower case letters, with the word "domain" being replaced by the real name of each domain. It is significantly relevant that this naming pattern is extended to all domains in the Data Mesh, thus minimizing semantic problems.

Within each domain, there are folders for each of the data products that compose it. The naming of these folders must also follow a consistent pattern, and it is proposed that they are identified in lowercase letters as shown in the pattern illustrated in Figure 16 ("domain"_"dataproduct"). The folder for each data product will include three distinct components: **input data** (corresponding to raw data), **output data** (corresponding to data resulting from the pipeline process), and **data product info.** The presence of the last component, data product info, is in line with the domain model explained in section 3.1. This folder should include relevant information such as the data quality report of the output data (In order to facilitate the reading of this report, it will be also available in Apache Atlas, with a direct link to the Power BI Server). The remaining information about the output data (such as Data Schema or Lineage) will also be present in Apache Atlas.

This proposed folder organization is not limited to this proof-of-concept but can be followed in other Data Mesh implementations. However, if the technology used does not support this type of structure, this folder organization should be adapted, ensuring that the intuitive and clear structure of the domains and their data products is maintained. Otherwise, the proper functioning of the Data Mesh can be jeopardized.

4.4. Organization of Databases and Tables for Domains and Data Products

In the same way a structure was defined and followed for the HDFS folders that represent the various nodes of the Data Mesh, it is also necessary to think about how the domains and data products will be organized in the Hive Metastore. Figure 17. Hive Structure for Domains and Data Products shows how the organization of domains and data products was defined in the scope of the proof of concept.

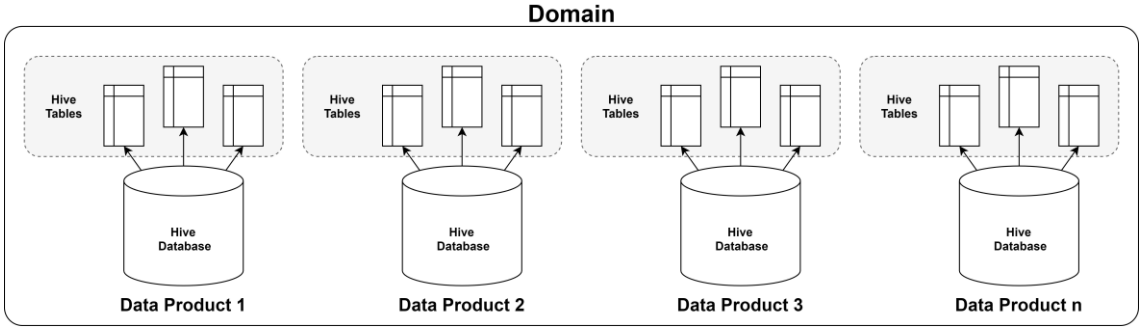


Figure 17. Hive Structure for Domains and Data Products

When considered the usage of a Data Mesh in complex organizational contexts, it is logically implied that the domains will be of significantly complex dimensions, and naturally the data products may follow the same pattern. Thus, in this proof of concept, it was defined that a data product corresponds to a Hive database (corresponding to the collection of several tables). On the other hand, domains are defined as being a collection of databases (of the various data products that compose them). It is believed that, although the dataset of this proof-of-concept is not very complex, this will be the most correct way to organize the Hive Metastore, anticipating that the data products will evolve over time, given the dynamic nature of organizations. Note that it is recommended to follow the data modelling approach (e.g.: flat tables, star schemas, among others) that is more suitable for each data product (here depicted as a Hive database) to more adequately manage storage space and querying performance.

4.5. Mesh and Data Catalog (Apache Atlas)

One of the most important principles of the Data Mesh is the complete discovery of the data that it holds by all its users. So, it is necessary that there is a component in the proof-of-concept where information about the data products is centralized, such as, for example,

data lineage, data schema, and data quality information. As explained in section 3.3.2, Apache Atlas was chosen as the Mesh and Data Catalog. By default, Apache Atlas does not provide all the information to portray Hive tables and databases. Therefore, it is necessary to extend it. Figure 18 presents an organization of the parameters per data product (hive database) and the respective tables (hive tables).

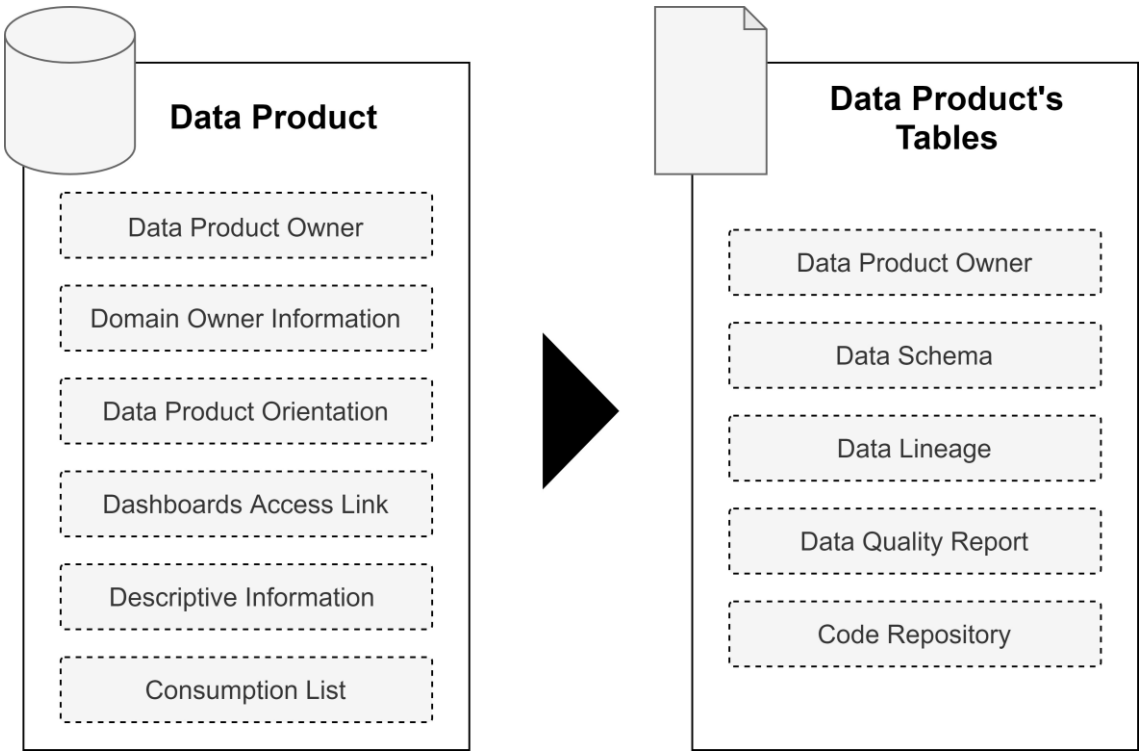


Figure 18. Attributes per entity in Apache Atlas

To better understand the extensions made to Apache Atlas, it is important to note how Atlas organizes metadata. In Apache Atlas, there is a type system to organize metadata, that is subdivided into three concepts: types, entities, and attributes. Types in the Apache Atlas can be understood as a collection of properties that characterize a metadata object. On the other hand, entities correspond to instances of types, and attributes to properties that characterize the types. Attributes are defined by a set of parameters that define the characteristics of their properties (e.g.: *name, isCompositive,* and *isUnique*) (Atlas, 2018). As previously explained, the data products correspond to Hive databases and the tables that compose them to Hive tables. Thus, an implementation of the Atlas extension was defined, which comprises two steps: i) create a new type in the Atlas type system called *Data Product*, which inherits the characteristics of hive databases with the addition of a set of complementary attributes (illustrated in Figure 18); and ii) extend the hive table to

include the attributes explained in Figure 18 (e.g.: Data Quality Report). Each of the extensions will be explained next.

Regarding the creation of the Data Product's metadata, it follows the structure shown in Figure 18 including the following attributes: *Data Product Owner*, *Domain Owner Information, Data Product Orientation, Dashboards Access Link, Descriptive Information,* and *Consumption List*. All these attributes are entered manually in Apache Atlas UI. Ideally, these attributes could be obtained automatically, but given the resources available, the implementation followed the manual collection of the metadata, and this automation may be subject of future work. The collection of this metadata requires the Apache Atlas REST API and the development of a JSON file to create or edit the existing metadata. In this JSON file, the structure of the Data Product type is indicated, and its attributes are divided into two groups: string type (when they are manual fields, such as the *Data Product Owner* attribute) or hive_db type (when the Data Product is associated with the hive database to which it corresponds). The following figure shows the structure of the data product metadata created.

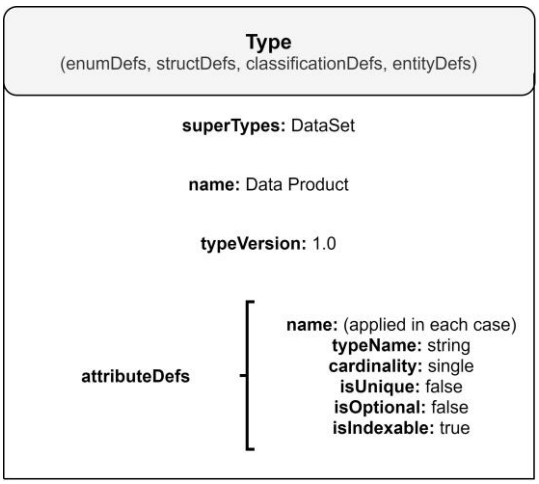


Figure 19. Data Product Metadata Structure

As explained earlier, the Data Product corresponds to type that has a hive database, extended to accommodate more parameters. For this reason, this type inherits the supertype DataSet. The attributes are defined using *typeName*, *cardinality*, *isUnique*, *isOptional*, and *isIndexable* (this option is set to true so that the attributes can be used/search by their value). The only case where this attribute definition changes is in the database association that corresponds to the Data Product. In this case, the *typeName* is

set as hive_db, to obtain the characteristics of a hive data base already defined in Apache Atlas. Figure 20 shows a part of the JSON file created for the Data Product definition.

Figure 20. JSON File from Data Procut type creation

Once the JSON file with the Data Product type creation code was defined in the Apache Atlas type system, it was necessary to use the HTTP (Hypertext Transfer Protocol) POST method for the REST service to create the metadata type. To do this, the following command was executed inside the container that holds Apache Atlas:

```
curl -u admin:hortonworks1 -ik -H "Content-Type: application/json" -X POST
http://localhost:21000/api/atlas/v2/types/typedefs -d
```

Once successful, the Data Product type is created in the type system. However, it is necessary that the tables stored within the Data Product database also contain additional information, as shown in Figure 18. To do this, the above process was repeated, and a JSON file was created, where the attributes Data Product Owner (dataProductOwner), Data Quality (dataQuality) and Code Access (codeAccess) are added to the existing hive table type. The added metadata definition for hive tables is presented in Figure 21 .

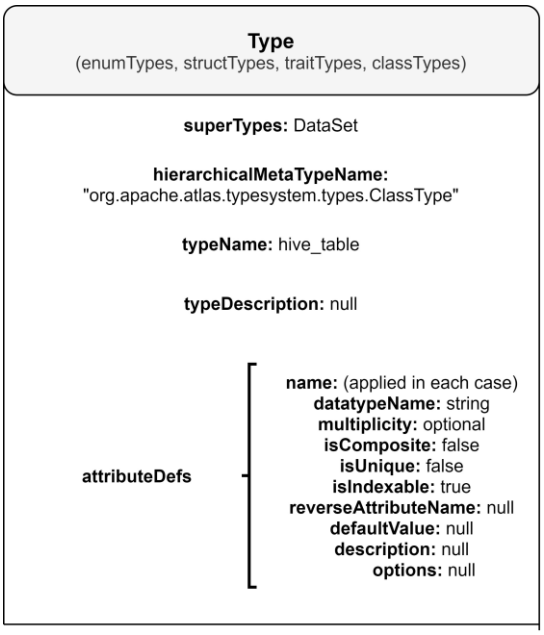


Figure 21. Hive Table Metadata Structure

In this case, and because it is an edition of an existing type, and not a creation of a new metadata type (as in the case of Data Product), the attributes must be defined according to the already existing set of properties. Thus, each new attribute added is defined by: *name, datatypeName, multiplicity, isComposite, isUnique, isIndexable, reverseAttributeName, defaultValue, description*, and *options* (some of these attributes like *description* are set to null, because in terms of implementation and considering the propose of this extension they are not required to have other definition). There is also a hierarchical association that must be made explicit (*hierarchicalMetaTypeName*), since there is already a hierarchy defined for the hive table type (which, for example, composes the hive databases) and without this definition, the code would go into error and the extension for the new attributes wouldn't be successful. Like the creation of the Data Product type, all the attributes are manually collected and are of string type. Through them, we can access the Data Quality report of each table, for example. Figure 22 shows a part of the JSON file created for the extension of the hive table already defined in Apache Atlas.

```
"enumTypes": [],
"structTypes": [],
"traitTypes": [],
"classTypes": [{
   "superTypes": ["DataSet"],
   "hierarchicalMetaTypeName": "org.apache.atlas.typesystem.types.ClassType",
   "type_name": "hive_table",
   "type_description": null,
    "attributeDefinitions": [
      {
      "name": "Data Product Owner",
      "datatypeName": "string",
      "multiplicity": "optional",
      "isComposite": false,
      "isUnique": false,
      "isIndexable": false,
      "reverseAttributeName": null,
      "defaultValue": null,
      "description":null,
      "options": null,
      },
```


Figure 22. JSON File for the extension of Hive Table Type

To execute this JSON file, the PUT method was used in order to edit the existing type, having executed the following command:

```
curl -u admin:hortonworks1 -ik -H "Content-Type: application/json" -X PUT
http://localhost:21000/api/atlas/types -d
```

In short, once all the extensions are completed, it is possible to use Apache Atlas to create and catalog the various nodes of the Data Mesh. A Data Product corresponds, in this proof of concept, to a hive database. In turn, the various tables that make up the Data Product are of the hive table type, and the domains can be found by clustering the various databases (you can also find them by searching for their name in Apache Atlas). The next figures show the extensions made in Apache Atlas. Figure 23. Search by Domain Designation shows, it is possible to use Apache Atlas, so that by searching for a domain name, all the data products that compose it are displayed.

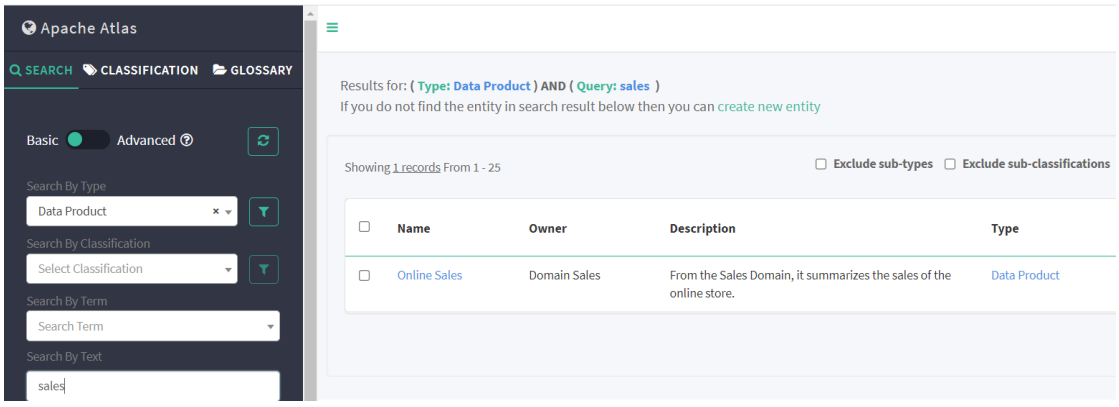


Figure 23. Search by Domain Designation

On the other hand, by using the side bar to search by data product, it is also possible to get a list of the full set of existing data products in the Data Mesh, as shown in Figure 24.

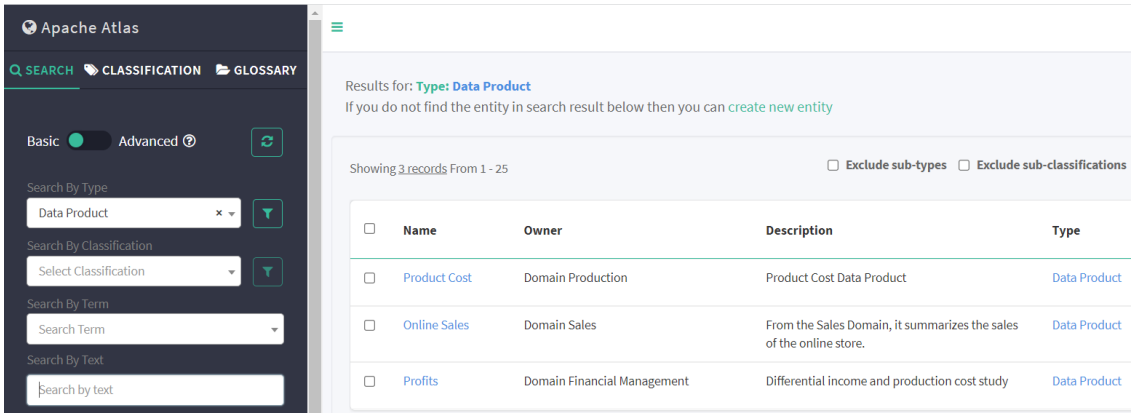


Figure 24. List of existing Data Products in the Data Mesh

By selecting a given data product, and following the logic presented before in this section on the Apache Atlas extension, we can retrieve information such as Data Product Owner, Dashboards Access, and Consumption List, as shown in Figure 25.

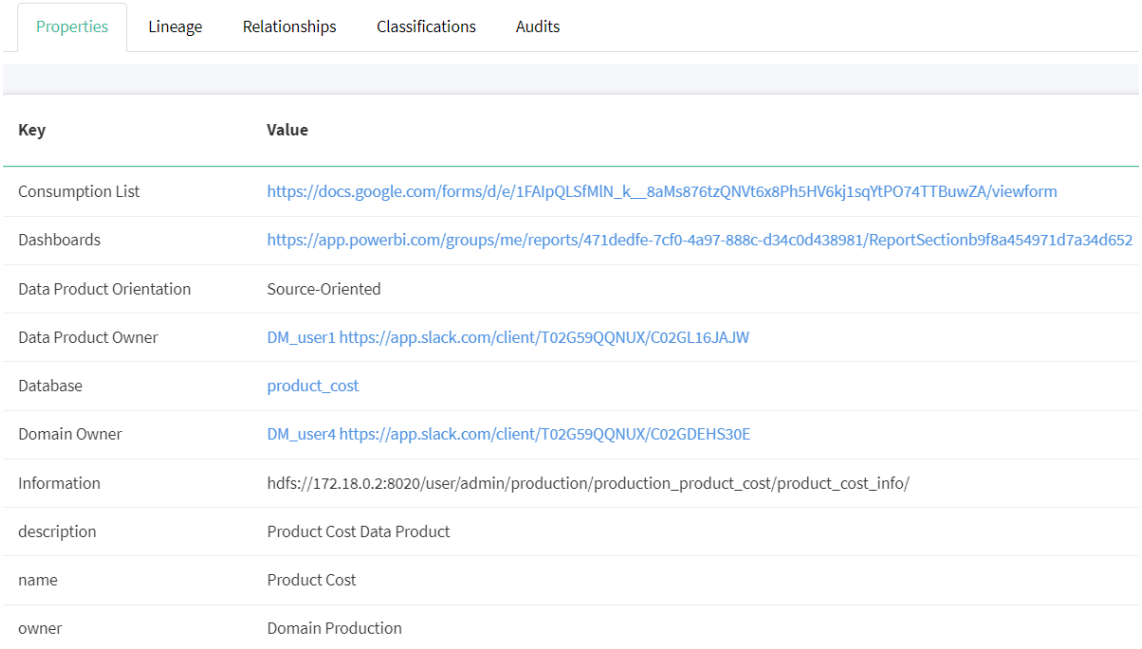


Figure 25. Data Product Details

Once inside the universe of a data product, it is possible to navigate and retrieve more details about the tables that are present in each data product's database. In the case of this

proof of concept, only one database was considered. The Figure 26 and Figure 27 show a possible navigation path from the database of the data product to its tables.

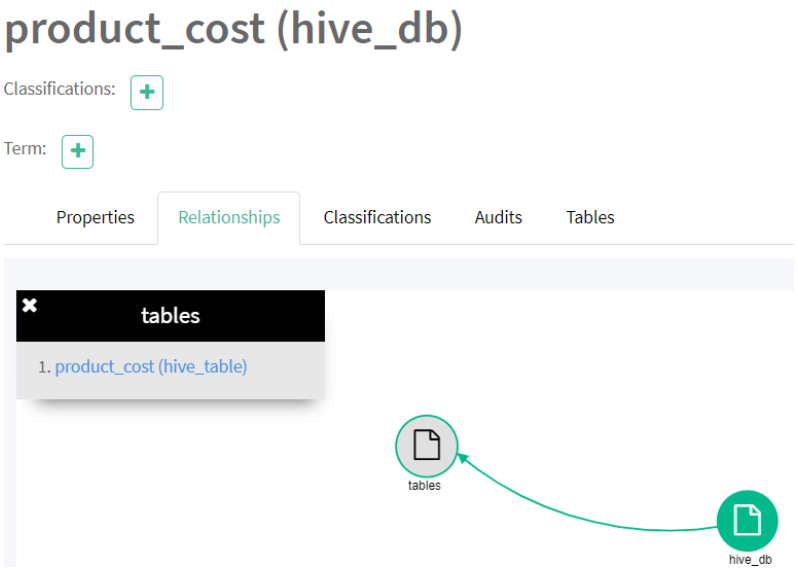


Figure 26. Data Product's Database View

Figure 27. Data Product's Tables View

Already within each table that Is held in each data product, it is possible to access a new set of information that is considered pertinent and more specific to each table, such as the data quality report, and the code repository that fuels them. Figure 28 shows the extensions made to Apache Atlas to provide these features for each table. Furthermore, it is possible to see the data lienage in each case, as shown in Figure 29.

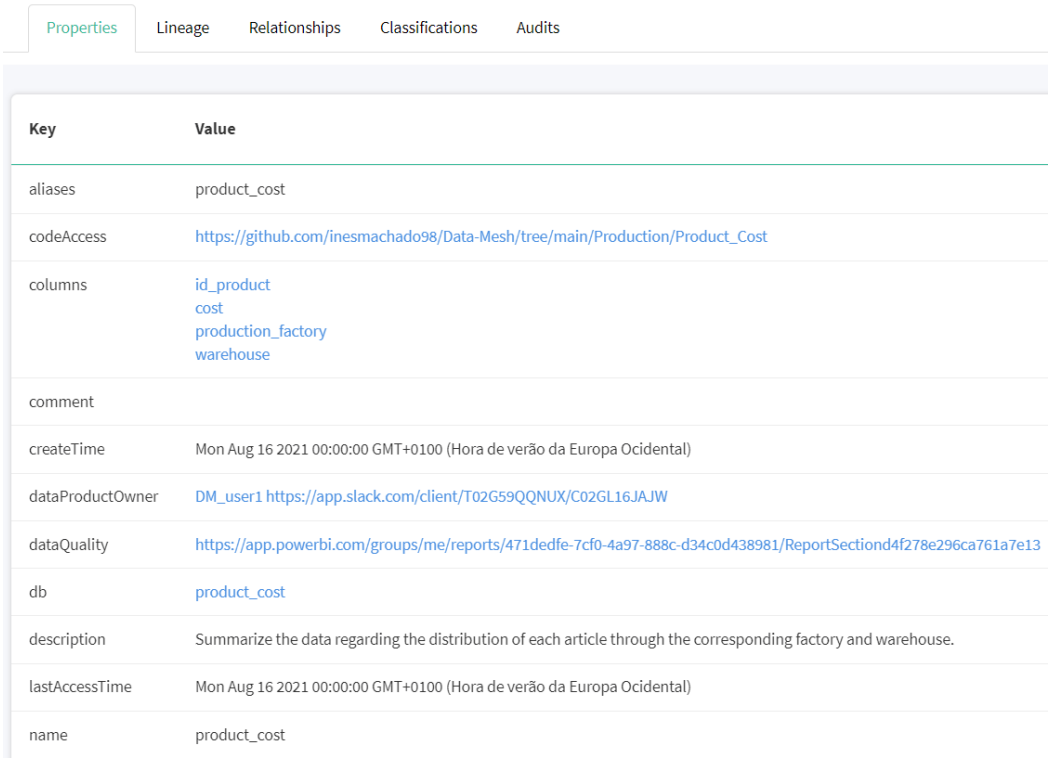


Figure 28. Data Product's Tables Details

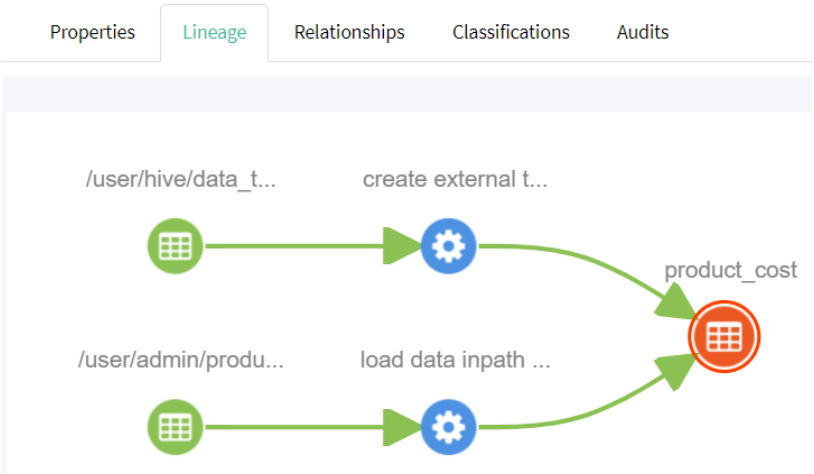


Figure 29. Data Lineage in each Table

The navigation flow shown in figure 29 represents an example of the flow that a Data Analyst undergoes when discovering and investigating a new dataset. Thus, with the previously presented Apache Atlas extensions, it is possible to have a decentralized architecture with functional centralization when it comes to cataloging its various nodes and data.

4.6. DATA QUALITY SCRIPT AND REPORT

Data quality analysis is a process that accompanies the flow of data in several phases: either in the analysis of data quality at the time of ingestion (to understand its context and characteristics), after the creation of data pipelines (to infer on the consistency of this data with that already present in the data storage systems), and as reports and indicators that allow monitoring the data. Data quality is also a conceptual class presented in the domain model (section 3.1) and it is seen as crucial when dealing with decentralized data architectures. Thus, an automated Data Quality script was developed and integrated into this proof of concept. This script allows Data Mesh users to know the data they want to use, in terms of quality assessment. For example, a Data Analyst who needs to gather data and report based on it, can access, through Apache Atlas, the output of this script and quickly infer on the nature of the data that will be used, knowing its schema, maximum and minimum values, among other characteristics. A part of this script is shown in Figure 30.

```
val rowsCount = df.count()
//Nulls
val df_null = df.select(df.columns.map(colName => {sum(when(col(colName).isNull, 1).otherwise(0))as s"${colName}"}): _*)
val df_output = df_null.withColumn("Data Quality Report",lit("Nulls"))

//Not Nulls
val df_not_null = df.select(df.columns.map(colName => {sum(when(col(colName).isNotNull, 1).otherwise(0))as s"${colName}"}): _*)
val df_out_not_nulls = df_not_null.withColumn("Data Quality Report",lit("Not Nulls"))

//Blank Spaces
val df_blank = df_not_null.select(df1.columns.map(colName => {sum(when(col(colName) === "", 1).otherwise(0))as s"${colName}"}): _*)
val df_out_blank = df_blank.withColumn("Data Quality Report",lit("Blank Spaces"))
```

Figure 30. Data Quality Script

For this analysis to become more intuitive, the script output is transformed into a dashboard, using the Microsoft PowerBI tool, as illustrated in Figure 31. Through this dashboard, the user can analyze the percentage of rows with information (not null), the number of attributes that contain the hive table in question, the distribution of not nulls, nulls and blank spaces in each attribute, the distribution of the lengths of the various columns, and even know the maximum and minimum values for each attribute. This script was developed in *Scala* and it was implemented in Jupyter Notebooks that encode the various data pipelines of each Data Product. As mentioned earlier in this section, users can access this dashboard through Apache Atlas, clicking on each Data Product and navigating to the table environment.

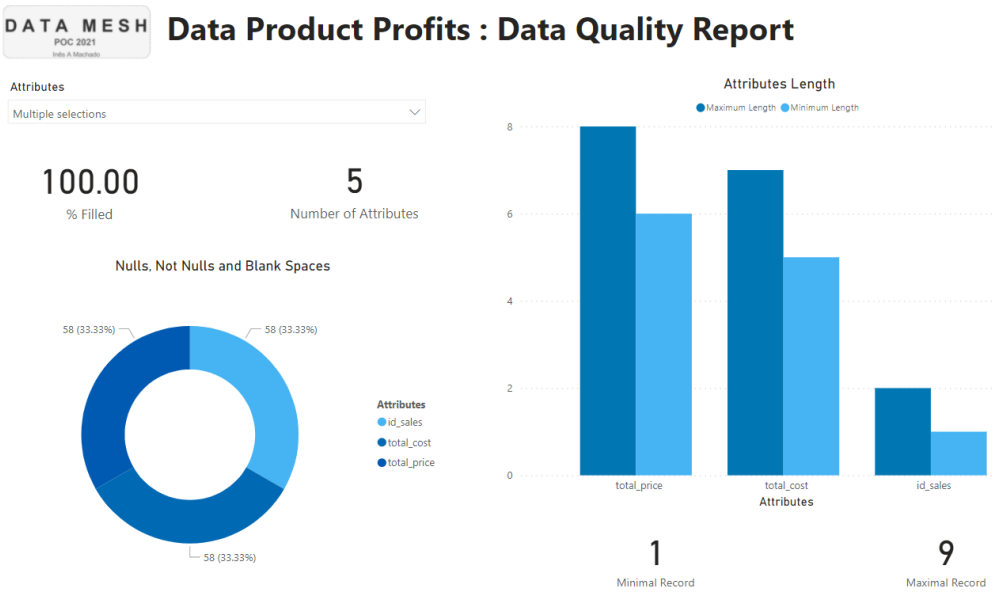


Figure 31. Data Quality Dashboard

Data products (e.g. product cost) can also have their own visualizations (not a mandatory condition for all data products) and, therefore, it is also relevant to allow various users of the Data Mesh to have access to these analytical dashboards. Thus, in the details of each Data Product in Apache Atlas, it is possible to find the analytical dashboards that correspond to them, using the Microsoft PowerBI tool. The dashboards developed within this proof-of-concept are for illustrative purposes only and hold no particular meaning neither they intend to display a properly defined and planned data visualization. Figure 32 show an example of a dashboard developed in this work.

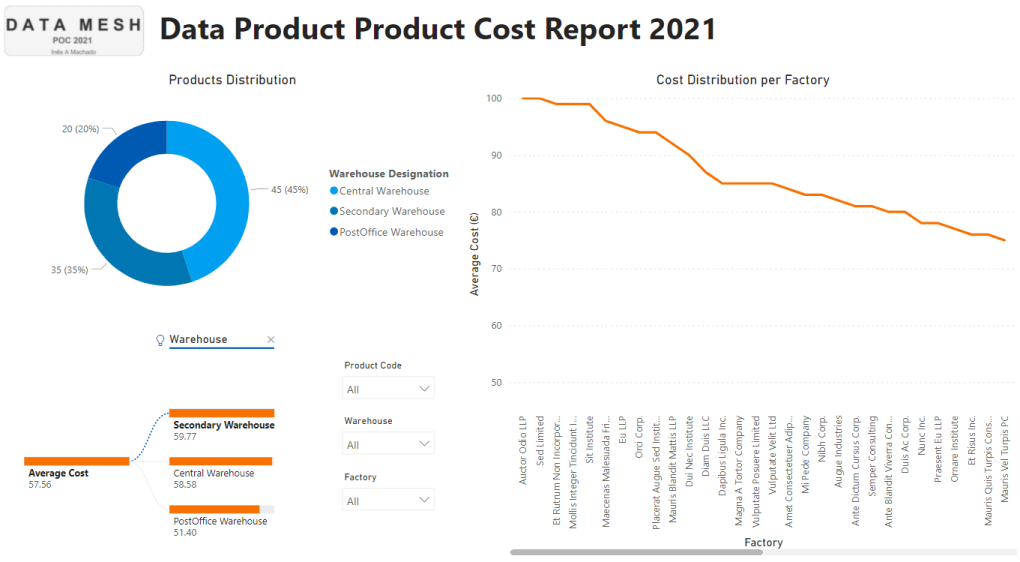


Figure 32. Product Cost's Analytical Dashboard

In this proof of concept, analytical dashboards and data quality dashboards were developed for each data product, being the remaining dashboard in the appendix.

4.7. Code Repository

As illustrated in the domain model (section 3.1), it is important to have a code repository to store the code from the various notebooks (or other scripts and files) that are being developed and that maintain the data pipelines. More than the storage component, a Data Mesh user can find a data product in Apache Atlas, and by requesting access to it, they can analyze the code that builds and maintains the data pipeline, allowing the users to analyze the code and frameworks used for building the data product's pipelines. Thus, a repository on GitHub was used as a code storage component. It is important to note that although in this proof-of-concept a centralized type of code repository has been used, it is possible that a more decentralized perspective is adopted (in this way, each domain or even data product, may have its own repositories). In the repository of the present proof-of-concept, the folder structure is very similar to the structure presented previously in Figure 16, being divided by the various domains and data products. However, there are folders that contain information that targets all domains and, therefore, these folders are also located in the root of the data repository (e.g.: Data Quality Script folder). The explained structure can be seen in Figure 33 Figure 34.

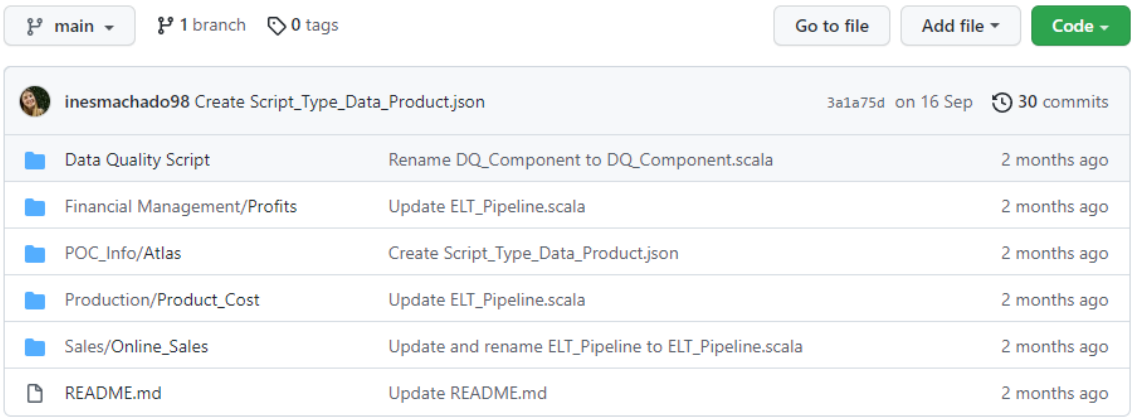


Figure 33. Folder Organization in Code Repository

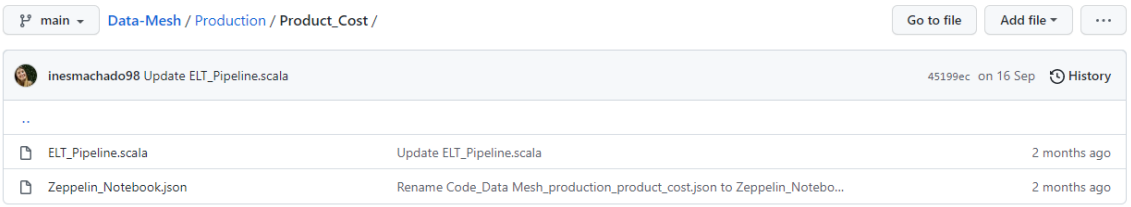


Figure 34. Detailed view of a folder content

However, although in the context of this proof of concept, there is centralization regarding the code repository, this is not mandatory and therefore must be adjusted to the actual context of each organization. Moreover, each data product team should choose, among the options available on the self-serve data platform, the tool that best fits their needs. Assuming that the eventual disparity of choices between the teams does not compromise the availability of the data repositories, each team must submit in Apache Atlas the address where this code can be accessed and analyzed, if necessary.

4.8. Consumption List

The consumption list, as explained before in section 3.1, is the component that allows the various users to request access to a particular data product. This list is connected to an authentication and authorization mechanism, which allows processing the validity of the requests made by the Data Mesh users. It is thus possible to find in this list the set of users that subscribe to each data product, and through the security mechanism associated with this list, maintain the homeostasis of the Data Mesh, as far as accesses by users is concerned. Ideally, this consumption list would be developed using a data security monitoring and management technology on a cluster, such as Apache Ranger. However, although the setup of this service was done on the cluster of this proof of concept, it was not possible to implement this Data Mesh feature, given the time available to complete this research process. Therefore, to demonstrate the applicability of the conceptual classes presented in the domain model (section 3.1), a Google Form was created. This form serves as an example of the consumption list component.

This access form requires the identification of the cluster, data product team, domain and data products to which the access is requested, as well as the reason for this request, and the same is later sent to the respective domain owners. The consumption list is available in Apache Atlas (thanks to the extension made in this work) and it can be accessed directly

by each user. However, these rules (e.g.: sending the access requested to the domain owner) and parameters (e.g.: filling in the data product team) were defined in the scope of the proof-of-concept and should be extended to the necessary parameters in a real context. It should be added that this consumption list can be managed by each Domain Owner, as well as by a team assigned to the management and monitoring of the Data Mesh data product accesses - each organization should therefore think and build its teams in a way that meets its needs and does not compromise the proper functioning of the Data Mesh. Figure 35 shows a possible way of filling in the consumption list form, in order to request access to a specific data product.

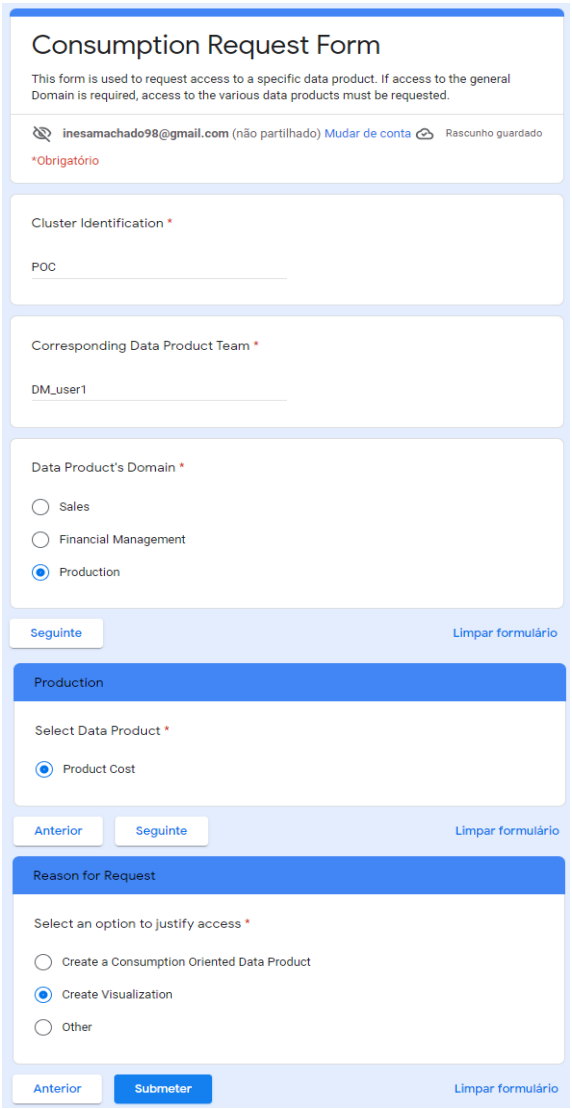


Figure 35. Consumption List Form

4.9. DATA MESH COMMUNICATION CHANNEL

One of the conceptual classes proposed in the domain model (section 3.1), which goes beyond the conceptual classes identified during the literature review process, is the Data Mesh Communication Channel. This component was created to make the communication more direct when it comes to topics related to the Data Mesh. In organizations, there is a significant flow of data between teams that often translates into emails, scheduling meetings, etc. Thus, the main intention of the communication channel is to reduce the entropy associated with communications regarding Data Mesh issues, by creating a specific channel for this purpose.

In the case of this proof of concept, the Slack tool was chosen. In this tool, as many channels as necessary can be created and directed to the various domains and data products. The connection to these channels is also available in the Mesh and Data Catalog (Apache Atlas), so that the various users can directly access these channels and communicate more directly and quickly to those responsible for each topic (e.g.: data product team).

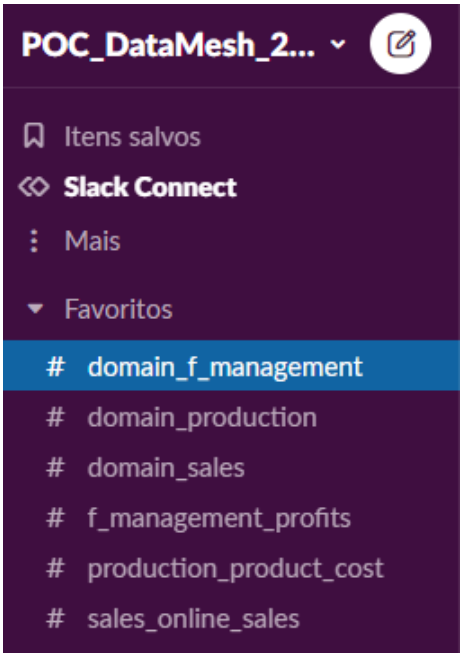


Figure 36. Data Mesh Communication Channel

5. CONCLUSION

The purpose of this chapter is to present the conclusions drawn during the elaboration of this master's thesis, as mentioned in section 1.5. This chapter is divided into three sections, the conclusions concerning the literature review and state-of-art, the conclusions of the proof-of-concept, and the future work respectively. The first section summarizes the conclusions regarding the concepts explained, the analysis of the motivation for the appearance and the various approaches presented in Data Mesh. The second section is related to the conclusions retrieved after the realization of the present proof-of-concept and, in the future work section, the work to which this master's thesis is proposed is presented.

5.1. CONCLUSIONS ABOUT THE LITERATURE REVIEW

The main objective of the literature review process is to enable a better understanding of where the present master's thesis topic comes from. For this, it was important not only to understand the whole related work (although scarce) on Data Mesh, but also concepts such as Big Data, Data Warehouse, Big Data Warehouse and Data Lake. It was quickly possible to infer that there is a huge "unknown" as far as Data Mesh is concerned, from its conceptual component to the practical implementation component. When analyzing the three approaches exposed in section 2.4 , it is possible to see that although they share some bases among themselves (e.g.: Data as a Product concept, guarantee and use of DATSIS principles), there is no consistency among them, because as it is an emergent architecture there are still no well-established guidelines on Data Mesh design and implementation. This last fact justifies the relevance and potential benefit of this master's thesis.

Throughout chapter 2, the evolution of data architectures over time - from Data Warehouses to Data Lakes - was explored. Naturally, and as explained in this chapter of the document, each of these data architectures has been demonstrating its limitations (e.g.: scalability), and it is within these limitations that the various evolutions have emerged until today. As far as Data Lakes are concerned, being these currently the most common type of data repository adopted by companies (Dehghani, 2020a), it was possible to infer that despite all the advantages associated with them (e.g.: the possibility of a large data

injection without worrying about rigid data schemas), they have not come to satisfy efficiently the needs of organizations. And so, there is a problem that must be addressed.

The sections 2.2, 2.3, 2.4 allowed to conclude that Data Mesh is not just about inserting new technologies in existing architectures, or adding new capabilities, or reorganizing only their components. Data Mesh brings with it the need to change the current paradigm, encompassing this change several aspects ranging from the platform's infrastructure itself to the reorganization of the teams that work with this platform. The fact that organizations make data "agnostic of its nature", when ingested, raises the problem of lack of ownership. Now, in an organization where data "belongs to everyone, but in reality, to no one" there are, as stated in the related work, problems of data quality - which ends up affecting the potential of their analytical value, and so on. So, Data Mesh tries to solve this problem making data the true organization’s concern.

Therefore, it is possible to synthesize, that Data Mesh ensures compliance with DATSIS principles, turns data into a product, organizes the various teams and data products according to the organizational domains (from which they emerge) and decentralizes the whole process (relieving current teams of the pressure of requests they feel), providing a self-serve infrastructure, so that the various teams can create and process their data products. These are made available in a Mesh's own node, known by all users, who can access it according to the federated governance policy present in that node (e.g.: can only read the node and not change its content). This way, it is possible to overcome the bottleneck felt in the data platform teams and ensure the quality of the data, within the organization, as demonstrate by Zalando work in Data Mesh (Max Schultze & Arif Wider, 2020). Regarding the evolutionary process of science, and according to Kuhn, there is a change of the paradigm (Kuhn, 1970).

Analyzing Netflix and Zalando work, it is possible to conclude that the migration process from a Data Lake architecture to a Data Mesh architecture is possible. Furthermore, it is possible to realize that the architecture previously instituted (the Data Lake) is reused and is part of the Mesh (working, for example, as a node of it). However, when analyzed the practical work of both it is possible to infer that there is no similarity in the way Data Mesh is implemented and designed - as much as they respect most of its core concepts (domains driven approach, data as products, self-serve data infrastructure and federated

computational governance). The present master's thesis topic is therefore justified in this "gap".

There are still many open challenges that need to be studied and developed such as: "What are the rigorous and concrete steps that can be followed to implement a Data Mesh?"; "What are the conceptual and technological components for a Data Mesh architecture that can be easily translated into a real solution?"; or "Will the convergence of data processing technologies be beneficial in terms of data quality? Even if this convergence is beneficial for the better management of the self-serve infrastructure?". In short, there is the need to stablish the design and implementation of the Data Mesh - this being the main objective of this master's thesis.

5.2. Conclusions about the Proposed Approach

Throughout chapter 3, the produced artifacts are presented: the domain model, the conceptual architecture, and the technological architecture. The domain model synthesizes the conceptual classes that must be present when building and using the Data Mesh. Intentionally, this model mixes more technical components (e.g.: batch processing), as well as topics related to the structuring and organization of the data teams themselves (e.g.: establishment of a Data Product Team) - since the Data Mesh is exactly this paradigm shift at all levels, from those related to infrastructure and technologies, as well as the organizational level of the companies, of the people that make up the teams. The domain model serves as the ignition for the whole of chapter three, and for the artifacts that follow: the conceptual architecture and the technological architecture. The conceptual architecture intends, at a high level, to define and make explicit the various components (even at the infrastructure level) that are necessary to support the Data Mesh, naturally ending up in the technological architecture. This second architecture arises from the need to provide companies with tangible solutions of what may be the tools to be adopted to build and maintain Data Mesh - and may even, by reviewing these presented technologies, infer that they only need to reorganize their mindset, teams, and way of working, to build their Data Mesh from within and reuse what they already have. These technologies are, however, only examples and suggestions, leaving it up to each company to adopt those that best meet its needs.

Once these models (artifacts of the master's thesis) were consolidated, the proof-of-concept was carried out to validate the postulated assumptions. It is concluded, with the realization of the proof of concept, that the model of established domains is possible to be implemented, corresponding to a functional decentralized architecture. Despite not having been implemented in its entirety (which would not be feasible within the scope of a master's thesis), its key elements are implemented and there is, above all, a catalog of data and nodes, which encompasses various aspects - the point that centralizes the decentralization of the architecture. Phase to the known commonly instituted data systems (where there is no such concern in treating data as a product and fulfilling a series of metrics to enhance usability), the implemented architecture allows the various users to discover in a complete way (and with complete means the team, people, quality, lineage, etc. behind it) the data they want to handle. Moreover, all data products share the same "space" (since the existing resources allow only one cluster to be used), but are available separately, and their access can be granted depending on the authorization given, via the consumption list. The consumption list also allows the users that consume each data product, to be known. Let's consider that the available services are the services made available on the Hadoop cluster, combined with those that go beyond this environment but are necessary for the Data Mesh architecture (such as GitHub). Thus, each team can develop its data products using the technologies that are most favorable to it, and consume the data from each product via the Hive table. It is thus considered that this master's thesis fulfills its main goal and the defined objectives.

5.3. SCIENTIFIC PUBLICATIONS

The present master's thesis made it possible to publish two scientific papers on the topic of Data Mesh, both of which were accepted and presented at the respective conferences. The first paper: "Data-Driven Information Systems: The Data Mesh Paradigm Shift", was the first Data Mesh scientific paper worldwide (at the time of publication), and it was published at 29th International Conference of Information System Development 2021, as a vision paper, in collaboration with Professor Carlos Costa and Professor Maribel Yasmina Santos. This paper presents the domain model and conceptual architecture.

The second scientific publication, entitled "Data Mesh: Concepts and Principles of a Paradigm Shift in Data Architectures", is the first scientific article that portrays the state-of-the-art of the Data Mesh topic, and was published in International Confonference

Enterprise Information Systems 2021 (Centeris 2021), as a full paper, also by the same authors.

Finally, an empirical paper entitled "Advancing Data Architectures with Data Mesh Implementations" has been submitted for evaluation at the 34th International Conference on Advanced Information Systems Engineering (CAiSE 2022). This paper was written in partnership with the same authors of the previously mentioned papers and aims to present the proof-of-concept that aims to validate the models proposed in this master thesis.

5.4. FUTURE WORK

After analyzing the existing literature about the concept of Data Mesh, it can be quickly concluded that there is still a long path to explore to establish a clear basis for the constructs, models, methods, and instantiations of the Data Mesh. Although the concept is already being embraced by some organizations, when analyzed the work developed by Netflix or Zalando (Justin Cunningham, 2020; Max Schultze & Arif Wider, 2020), for example, there are significantly different approaches to accomplish a Data Mesh architecture (Justin Cunningham, 2020; Max Schultze & Arif Wider, 2020), but at the same time, there are no general models and methods that can be followed, without the same being too conceptual to be translated into real-world instantiations. On the other hand, there is a relevant core aspect related to the success of the Data Mesh implementation: the organizational reorganization - from the rearrangement and management of the teams to the company's vision towards data as a product. In this sense, not only is necessary to take into consideration the conceptual and technological aspects of the design and implementation of a Data Mesh, but also there is the need to consider the relationship between this concept and the organizational needs, structure, and processes. Finally, it is also necessary to find the contexts and scenarios in which the Data Mesh may fail or show significant disadvantages, highlighting opportunities for refining and evolving the concept.

Thus, this master's thesis, tried to tackle some of these challenges related to the Data Mesh concept, which have been discussed throughout this section. This work focused on, not only on proposing and demonstrating models and methods at the conceptual level, but also at the technological level (major gap identified in the literature), while discussing organizational aspects whenever applicable and relevant.

However, there are some key points that can still be pointed out as future work on this topic: the first and most related to the presented proof-of-concept presented, will be the full functional development of the consumption list, using technologies suitable for this purpose, such as the Apache Ranger. Next there are the challenges of mapping the data within the various data products so that, node to node, there is not too much replication of data and effort (the development of a framework that enables this mapping and helps contextualize the topic). Finally, there is also the challenge of “change”: whether in terms of the impact that this paradigm shift has on the people in the organizations (in terms of reorganizing teams, for example), or in how the change of data within each data product will be handled in the Data Mesh as a whole. Data Mesh is progressing every day, as new contributions come in from all over the world, which does not mean that there is not still a long way to go when it comes to consolidating the subject.

REFERENCES

Adam Barker, & Jonathan Stuart Ward. (2013). Undefined By Data: A Survey of Big Data Definitions.

Andrade, C., Costa, C., Correia, J., & Santos, M. Y. (2019). Intelligent event broker: A complex event processing system in big data contexts. *25th Americas Conference on Information Systems, AMCIS 2019*, 1–10.

Atlas, A. (2018). Atlas Technical User Guide. Retrieved 20 October 2021, from https://atlas.apache.org/#/

AWS. (2021). AWS. Retrieved 26 May 2021, from https://aws.amazon.com/pt/free/?trk=ps_a134p000003yhdnAAA&trkCampaign=acq_paid_search_brand&sc_channel=ps&sc_campaign=acquisition_IBERIA&sc_publisher=google&sc_category=core&sc_country=IBERIA&sc_geo=EMEA&sc_outcome=Acquisition&sc_detail=aws services&sc_co

Barr, M. (2020). What is a Data Mesh — and How Not to Mesh it Up. Retrieved 29 October 2020, from https://towardsdatascience.com/what-is-a-data-mesh-and-how-not-to-mesh-it-up-210710bb41e0

Beheshti, A., Benatallah, B., Nouri, R., Chhieng, V. M., Xiong, H., & Zhao, X. (2017). CoreDB, (i), 2451–2454. Retrieved from https://doi.org/10.1145/3132847.3133171

Chen, H., H.L.Chiang, R., & C. Storey, V. (2018). Business Intelligence and Analytics: From Big Data To Big Impact. *MIS Quarterly*, 36(4), 1165–1188. Retrieved from http://www.jstor.org/stable/41703503

Costa, C., Andrade, C., & Santos, M. Y. (2019). Big Data Warehouses for Smart Industries. *Encyclopedia of Big Data Technologies*, 341–351. Retrieved from https://doi.org/10.1007/978-3-319-77525-8_204

Costa, C., & Santos, M. Y. (2017). Big Data: State-of-the-art concepts, techniques, technologies, modeling approaches and research challenges. *IAENG International Journal of Computer Science*, 44(3), 285–301.

Dehghani, Z. (2019). How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh, 1–20. Retrieved from https://martinfowler.com/articles/data-monolith-to-mesh.html

Dehghani, Z. (2020a). Data Mesh Paradigm Shift in Data Platform Architecture. San Francisco, USA: InfoQ. Retrieved from https://www.youtube.com/watch?v=52MCFe4v0UU

Dehghani, Z. (2020b). Data Mesh Principles and Logical Architecture. Retrieved 7 December 2020, from https://martinfowler.com/articles/data-mesh-principles.html

Diebold, F. X. (2013). A Personal Perspective on the Origin(s) and Development of 'Big Data': The Phenomenon, the Term, and the Discipline, Second Version. *SSRN Electronic Journal*. Retrieved from https://doi.org/10.2139/ssrn.2202843

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2), 137–144. Retrieved from https://doi.org/10.1016/j.ijinfomgt.2014.10.007

Golfarelli, M., & Rizzi, S. (2009). *Data Warehouse Design: Modern Principles and Methodologies*. McGraw-Hill, Inc.

Google. (2021). Google Cloud. Retrieved 26 May 2021, from https://cloud.google.com/products/?utm_source=google&utm_medium=cpc&utm_campaign=emea-pt-all-pt-dr-bkws-all-all-trial-e-gcp-1010042&utm_content=text-ad-none-any-DEV_c-CRE_282851264863-ADGP_Hybrid %7C BKWS - EXA %7C Txt ~ GCP ~ General%23v3-KWID_4370005328

Hai, R., Geisler, S., & Quix, C. (2016). Constance: An intelligent data lake system. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 26-June-20, 2097–2100. Retrieved from https://doi.org/10.1145/2882903.2899389

Inmon, W. H. (2005). *Building the Data Warehouse*. (John wiley & sons,Ed.) (Third Edit).

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*. Retrieved from https://doi.org/10.1007/s12525-021-00475-2

Justin Cunningham. (2020). Netflix Data Mesh: Composable Data Processing - Justin Cunningham. Retrieved 25 September 2020, from https://www.youtube.com/watch?v=TO_IiN06jJ4

Khine, P. P., & Wang, Z. S. (2018). Data lake: a new ideology in big data era. *ITM Web of Conferences*, 17, 03025. Retrieved from https://doi.org/10.1051/itmconf/20181703025

Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit, The Definitive Guide to Dimensional Modeling*. *Wiley*.

Krishnan, K. (2013). *Data Warehousing in the Age of Big Data*. *Data Warehousing in the Age of Big Data*. Elsevier. Retrieved from https://doi.org/10.1016/C2012-0-02737-8

Kuhn, T. S. (1970). *The Structure of Scientific Revolutions Second Edition, Enlarged*. *International Encyclopedia of Unified Science*.

Lance Johnson. (2020). What is a Data Mesh? Retrieved 30 September 2020, from https://trustgrid.io/what-is-a-data-mesh/

Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. *Analysis*.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute*.

Max Schultze & Arif Wider. (2020). Data Mesh in Practice: How Europe's Leading Online Platform for Fashion Goes Beyond the Data Lake. Retrieved 15 December 2020, from https://www.youtube.com/watch?v=eiUhV56uVUc

Microsoft. (2021). Azure. Retrieved 26 May 2021, from https://azure.microsoft.com/pt-pt/services/

Miloslavskaya, N., & Tolstoy, A. (2016). Big Data, Fast Data and Data Lake Concepts. *Procedia Computer Science*, 88, 300–305. Retrieved from https://doi.org/10.1016/j.procs.2016.07.439

Nenad Jukic. (2006). Modeling strategies and alternatives for Data Warehousing projects. Retrieved from https://doi.org/10.1145/1121949.1121952

Nicole Laskowski. (2016). Data lake governance: A big data do or die. Retrieved 15 December 2020, from https://searchcio.techtarget.com/feature/Data-lake-governance-A-big-data-do-or-die

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. Retrieved from https://doi.org/10.2753/MIS0742-1222240302

Russom, P. (2016). Data Warehouse Modernization. *TDWI Best Practices Report*.

Sam Madden. (2012). From databases to big data. *IEEE Internet Computing*, 16(3), 4–6. Retrieved from https://doi.org/10.1109/MIC.2012.50

Santos, M. Y., & Costa, C. (2020). *Big Data concepts, warehousing, and analytics*. River Publishing.

Terri McClure. (2014). Yesterday's unified storage is today's enterprise data lake. Retrieved 15 December 2020, from https://searchstorage.techtarget.com/opinion/Yesterdays-unified-storage-is-todays-enterprise-data-lake

ThoughtWorks. (2020). Data Mesh. Retrieved 15 December 2020, from https://www.thoughtworks.com/radar/techniques/data-mesh

APPENDIX – DATA PRODUCT'S DASHBOARDS

This appendix contains all the dashboards developed in this work, both for data quality and analytics of each data product. To organize the presentation of this information, the data quality dashboards will be presented first, followed by the analytical dashboards for each data product.

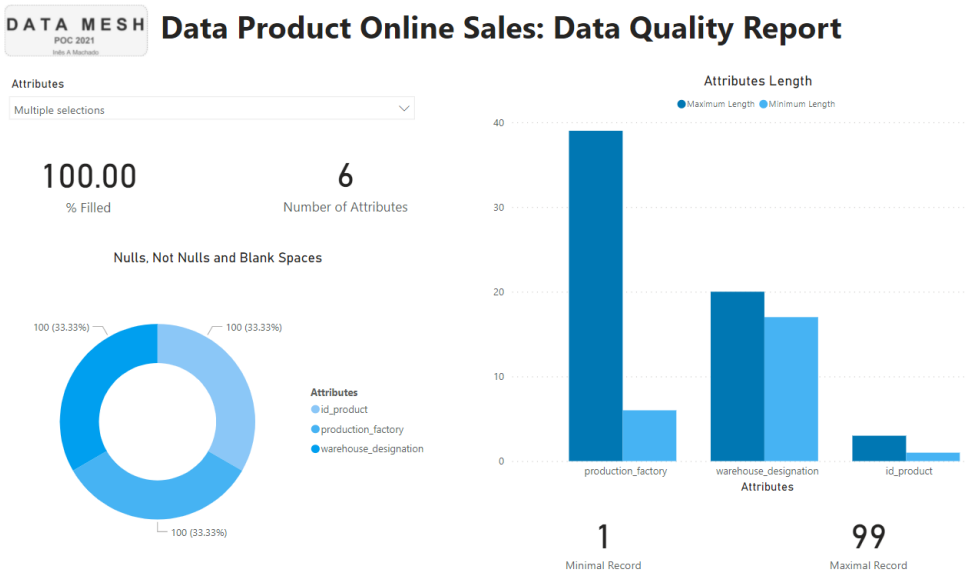


Figure 37. Data Quality Online Sales

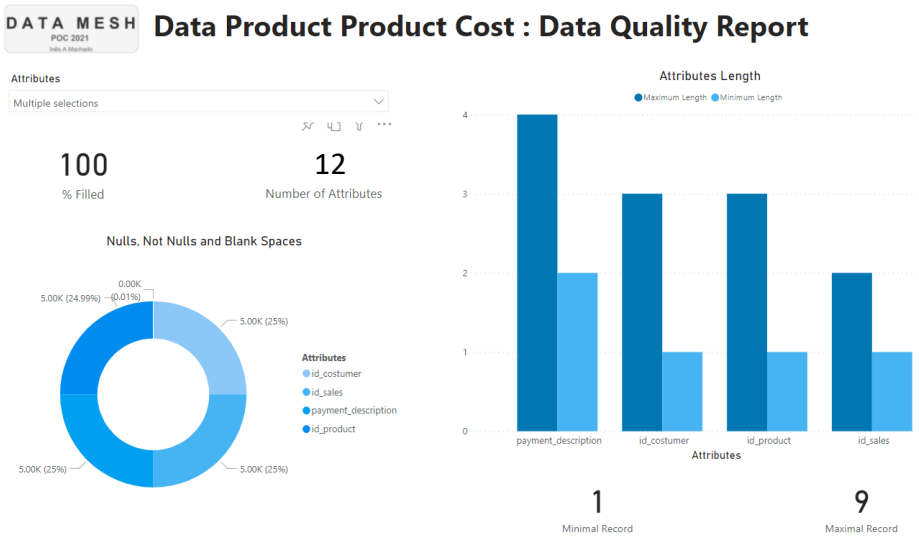


Figure 38. Data Quality Product Cost

Figure 39. Analytical Dashboard Profits (1)

Figure 40. Analytical Dashboard Profits (2)

Figure 41. Analytical Dashboard Online Sales