technologies

MDPI

Article

Discovering Data Domains and Products in Data Meshes Using Semantic Blueprints

Michalis Pingos * and Andreas S. Andreou

Faculty of Engineering and Technology, Department of Electrical Engineering, Computer Engineering and Informatics, Cyprus University of Technology, Limassol 3036, Cyprus; andreas.andreou@cut.ac.cy
* Correspondence: michalis.pingos@cut.ac.cy

Abstract: Nowadays, one of the greatest challenges in data meshes revolves around detecting and creating data domains and data products for providing the ability to adapt easily and quickly to changing business needs. This requires a disciplined approach to identify, differentiate and prioritize distinct data sources according to their content and diversity. The current paper tackles this highly complicated issue and suggests a standardized approach that integrates the concept of data blueprints with data meshes. In essence, a novel standardization framework is proposed that creates data products using a metadata semantic enrichment mechanism, the latter also offering data domain readiness and alignment. The approach is demonstrated using real-world data produced by multiple sources in a poultry meat production factory. A set of functional attributes is used to qualitatively compare the proposed approach to existing data structures utilized in storage architectures, with quite promising results. Finally, experimentation with different scenarios varying in data product complexity and granularity suggests a successful performance.

Keywords: big data; data lakes; data meshes; data products; data blueprints; metadata semantic enrichment

check for updates

Citation: Pingos, M.; Andreou, A.S. Discovering Data Domains and Products in Data Meshes Using Semantic Blueprints. *Technologies* **2024**, *12*, 105. https://doi.org/10.3390/technologies12070105

Academic Editor: Mohammed Mahmoud

Received: 12 April 2024
Revised: 8 June 2024
Accepted: 23 June 2024
Published: 7 July 2024

1. Introduction

Nowadays, big data is ubiquitous [1] and, by definition, the term refers to the enormous amounts of data that is digitally generated by the global population through tools and machines [2]. According to [3], at the beginning of 2023, the entire digital universe contained over 44 zettabytes of data, while approximately 2.5 quintillion bytes of data is generated each day. For a decade, Doug Laney's 3Vs model defined big data, considering the volume, variety and velocity characteristics as the three main challenges when dealing with big data [4]. It became obvious that the 3Vs model was incomplete after continuous work on big data [5], and three more Vs were added to it. IBM introduced veracity, which represents the unreliability of some data sources [6]. Oracle introduced value in 2012 as a defining characteristic of big data [7]. Finally, SAS coined variability, which refers to the variation of data rates as an additional dimension to big data characteristics [8]. Additional characteristics, such as viscosity, virality and ambiguity, were later proposed by other authors [9].

A vast amount of big data originates from heterogeneous sources with atypical patterns, which produce various kinds of structured, semi-structured and unstructured data in high frequencies [10]. This heterogeneous data needs to be treated differently than normal production speed data and to be stored in more flexible and/or higher servicing speed data storage architectures or structures compared to classic relational databases and data warehouses, such as big data warehouses, data lakes and data meshes. The current literature also shows a trend towards more decentralized data exchange architectures/structures, such as data markets and data meshes [11]. The latter two were key targets for many (large) companies and organizations to achieve, which adopted initiatives to facilitate transition from their existing, monolithic data platforms. One of the main challenges for this transition, in addition to the novelty of the concepts, is how to divide up the data landscape

into domains and identify data assets that should be turned into data products [12]. These organizational challenges are in fact often perceived to be more daunting than the technical challenges associated with data mesh design [13].

The main research contribution of this paper lies with the utilization of semantic data blueprints (SDBs) for discovering data products and domains in data meshes. Additionally, this work offers a standardized way to transform a data lake into a data mesh. The set of SDB essentially describes properties of data via stable attributes, such as variety, value, velocity and veracity, and attributes that are not stable over time, such as volume, last source update and keywords. The proposed approach builds upon previous work on the topic that introduced a semantic metadata enrichment mechanism for data lakes [14], which allows for the efficient storing and retrieval of data belonging to dispersed and heterogenous data sources. The same concepts are extended, modified and adapted in this work to match the characteristics of data meshes. A data mesh is conceived here as the evolution of a data lake in terms of organizing huge volumes of information (i.e., big data) expressed in multiple data forms (structured, unstructured and semi-structured), but, most importantly, for tracing this information easily, quickly and efficiently. Although both data lakes and data meshes can offer the backbone for software analytics with useful insights, a data mesh provides a more narrowly focused and domain-centric approach. Users can have more control over their data and improve analytics skills, as well as provide more precise insights for software products and procedures by utilizing the data mesh principles [15]. In this context, we propose a new set of semantic blueprints to facilitate the creation of data products through a domain-driven approach that allows us to retrieve information directly from its stored location. The proposed approach is demonstrated using real-world manufacturing data collected from a major local industrial player in Cyprus, namely, Paradisiotis Group (PARG). Performance is then assessed via the construction of data meshes based on various data products and the execution of SPARQL queries that vary in complexity, that is, regarding the granularity of information sought and the number of data sources.

The remainder of the paper is structured as follows. Sections 2 and 3 discuss the technical background and the related work respectively in the areas of data lakes and data meshes. Section 4 presents the extended data meshes framework and discusses its main components. This is followed by a qualitative evaluation between data lakes and data meshed in Section 5 based on a set of qualitative criteria. Section 6 demonstrates the applicability and assesses the performance of the proposed framework through a series of experiments conducted using real-world data collected at PARG. Finally, Section 6 concludes the paper and highlights future research directions.

2. Technical Background and Literature Overview

2.1. Technical Background

Data lakes (DLs) were proposed in 2010 as architectures suitable for dealing with big data and assisting organizations towards adopting data-driven approaches [16]. Storing structured, semi-structured and unstructured data in a DL at any scale was made feasible, as was selecting and organizing the data in a central repository. The data (relational and non-relational) could be stored directly in a DL in their current form, without the need to convert them to a structured one. Moreover, there was also no need to move to another system for performing a wide range of analytical methods, including dashboards and visualizations, big data processing, real-time analytics and machine learning to support decision making through predictive and prescriptive analytics [17]. DLs provide a cost effective, flexible and scalable way to host data in their raw format, enabling organizations to store large amounts of data without the need to conform to a specific schema beforehand. At the same time, a DL is one of the debatable ideas that emerged during the big data era. Since DLs are a relatively recently developed concept with revolutionary ideas, they present numerous adoption hurdles.

Khine and Wang [18] outline the main DL challenges: (i) The need for decentralization (physically & virtually); (ii) the need for data product discovery and a domain-driven approach; (iii) the absorption of all types of data without monitoring or governance; and (iv) the lack of a descriptive metadata mechanism to prevent a data swamp. In [19], a comprehensive state of the art of different approaches to a DL's design is provided, focusing on architectures and metadata management, which are key issues in successfully utilizing DLs. Enhancements to DLs provide additional features and benefits. A data pond (DP) constitutes a fraction of a DL (see Figure 1); it is typically smaller in scale, and it is used for specific purposes, such as testing or implementation of dedicated functionality. In addition, a DP is used for a specific application or use-case, and its design and architecture are optimized for that specific purpose. Data puddles, on the other hand, are smaller, pre-built datasets (see Figure 1) that are created to fit a special purpose (e.g., providing information on a portion of the data).

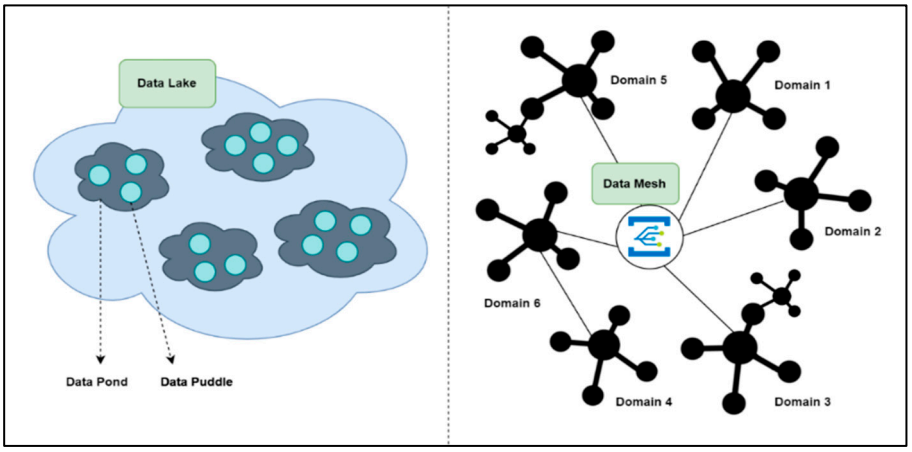

Figure 1. Core architectures for data lakes and data meshes.

The literature nowadays shows a tendency towards systems for decentralized data interchange, like data meshes (DMs). The term DM was first defined by Dehghani in 2019, who then provided greater detail on its principles and logical architecture [20]. According to [20], a DM is the next-generation data architecture that adopts a decentralized method of data management by treating data as products. It defines ownership and accountability for data products and emphasizes data governance, quality and operational excellence. In a DM, data are managed as products with clear product owners, and data consumers have self-service access to the data they need. It provides a more flexible and scalable solution than a traditional DL as it enables multiple sources of the truth, promotes data agility and encourages data literacy across the organization.

A DM is based on four core principles [21]: (a) *Decentralized data ownership* states that each team or microservice in an organization should own and be responsible for the data they produce; (b) *product-centric data* emphasizes that data should be treated as products rather than just as by-products of software development; (c) *automated data governance* advocates for the use of automation to enforce data quality, security and privacy policies; and (d) *shared data services* involve the creation of shared data services and APIs to enable teams to access and use data in a consistent and reliable manner. In addition, a DM may be seen as a data architectural pattern that provides a way to manage and share data between microservices in a scalable and decentralized manner. It is based on the idea of creating a "mesh" of data services that work together to provide consistent and reliable data access to all services that need it. The goal is to reduce the complexity and dependence on central databases, making it easier to manage data at scale in a microservice-based architecture. Furthermore, DMs attempt to address some of the shortcomings of monolithic data platforms such as DLs by creating data products and domains [12]. A data product is a tangible and valuable output of data that serves a specific business need. It is created

and managed within a DM architecture. Data products are created with a product mindset, which means that they have clear goals, user personas and metrics for success. Creating proper data products puts requirements on metadata templates that are not yet addressed by existing approaches.

2.2. Related Work

The emergence of big data, fueled by diverse software applications, has led to the establishment of big data warehouses and DLs as fundamental components for organizational decision making. However, the limitations of these monolithic architectures have highlighted the necessity for a paradigm shift towards data-oriented organizations. Enter DM, a novel architectural concept that prioritizes data as the central organizational concern. In a DM architecture, data is intentionally distributed across multiple nodes, mitigating chaos and data silos through centralized governance strategies and shared core principles. The work in [11] elucidates the motivation behind the DM paradigm, its key features, and approaches for its practical implementation. Furthermore, the authors in [22] discuss the prevalent trend of enterprises investing in next generation DLs to democratize data access and drive business insights through automated decision making. However, traditional DL architectures often encounter failure modes that hinder scalability and fail to deliver on their promises. To overcome these challenges, the paradigm needs to shift away from the centralized model of a DL or data warehouse towards a distributed architecture. This paradigm shift involves prioritizing domains as the primary concern, implementing platform thinking to establish self-serve data infrastructure, and treating data as products. The reference to Dehghani's article highlights the importance of transitioning from a monolithic DL to a distributed DM to address these issues effectively.

In order to correlate and systematize enormous amounts of dispersed manufacturing data, associate the "normalized" data with operations and orchestrate processes in a more closed-loop performance system that delivers continuous innovation and insight, the term "manufacturing blueprints" was coined to create a basic knowledge environment that gives manufacturers more granular, fine-grained and composable knowledge structures and approaches [23].

A metadata mechanism serves as the foundational architecture DL, orchestrating the organization and categorization of the multiple heterogeneous datasets hosted within. Without such a mechanism, a DL risks devolving into "data swamp", where the large amount of data lacks structure or coherence, impeding effective utilization. By carefully organizing and recording important information about where the data comes from, how they are structured, and how they are used, the metadata system gives data owners valuable insights. This helps them navigate and understand the data better, making tasks like searching for, retrieving and managing data much easier within the DL. Ultimately, this improves how efficiently operations are carried out and ensures better control over the data.

Previous research on DLs utilized the fundamental ideas behind manufacturing blueprints and extended them for describing and defining data sources in DLs by introducing the SDB and a standardized description of the data it produces. In particular, a novel standardization framework was proposed in [14] that combines the 5Vs big data characteristics mentioned earlier, blueprint ontologies and a DL architecture and is based on SDB, a metadata semantic enrichment mechanism. In this context, the data lake blueprint (DLB) metadata history was formed, which enables quick storage to and efficient retrieval from a DL via visual querying (VQ), something that contributes to addressing the extremely complex problem of dealing with heterogeneous data sources. The proposed DL architecture was made up of several data ponds, each of which hosted or referred to a specific type of data according to the pond design. Each pond had a unique data processing and storage system depending on the sort of data it contained. The suggested mechanism was compared to existing metadata systems using a set of functional qualities or features and the findings proved it a promising strategy [14].

Further to the above, DLMetaChain was introduced, which is an extended DL metadata system that connects diverse data sources with IoT data through blockchain and uses the aforementioned pond architecture. The blockchain and NFT technologies are considered as a viable option for tackling security and privacy concerns, as well as for developing trust between entities, where trust has either been under-developed or nonexistent. This enhanced approach focused on creating an architecture that guarantees the DL data will not be changed or altered [24].

Finally, a novel approach for storing and retrieving large amounts of data was proposed in [25] that aims to best serve the data-mining processes that use it. More specifically, a unique metadata mechanism based on SDB was proposed that enables blueprints to characterize and describe data sources, data items and process-related information, all being kept in a DL. Using the notions of a DL's blueprints, that approach extended prior work on the topic by adding a unique class of blueprints to keep track of data pertaining to process mining activities in a smart manufacturing environment (i.e., processes, events and machines). It also offered an extension of the DL architecture, introducing the concept of data puddles for facilitating the storing of high-frequency data. The aforementioned work used the resource description framework (RDF), designed for representing information about resources on the Web, to describe a data source's blueprints with the combination of the SPO triple model (subject–predicate–object).

In the realm of metadata enrichment, as presented in [14,24–26], the SDB emerges as a pivotal mechanism in preemptively identifying and characterizing potential data sources prior to their assimilation into the DL framework. Integrated with blueprint ontologies, the SDB framework lends robust support to data processing within DLs structured around a pond and puddle architecture. The concept of ponds and puddles in DL architecture as presented in the aforementioned framework serves as a forerunner to DMs by addressing the need for structured organization and specialized handling of diverse data types within a large, unified repository. In this framework, each pond is dedicated to a specific type of data, such as structured, unstructured or semi-structured, with specialized storage systems and processing methods tailored to the data type it hosts. Detailed blueprints, and, specifically, SDBs, which include static attributes (name, type, value, velocity, variety and veracity) and dynamic attributes (volume, last update and keywords), are used to enhance metadata, making data filtering and retrieval more efficient. Using RDF, the SDB provides a short but complete description for all data sources in the DL, making it easy to combine and use them, thus producing insights within the DL.

Building upon these notions of data semantic annotation, this paper extends the aforementioned approaches to align with the data mesh environment, providing further insights that will be detailed later. Transitioning from a pond and puddle architecture to a DM structure involves adopting principles of domain-oriented decentralization, self-serve data infrastructure, and treating data as products. Treating data as products, with clear ownership, quality standards and discoverability, aligns with the pond architecture's tailored handling and storage of each data type. The pond and puddle architecture provides the opportunity to have two levels of data products. However, the paradigm of DMs presents an expansive landscape where the potential for data product creation is boundless and unrestricted.

The limitations of monolithic data architectures (i.e., DLs) were investigated in [15]; these struggle with scalability and cost inefficiencies over time. Inspired by software engineering's move away from monolithic systems, the paper advocated for the adoption of decentralized data architectures, and specifically, of DMs. A DM, according to the authors, distributes data across multiple nodes while maintaining centralized governance and core principles to avoid chaos and data silos. That work proposed a domain model and a conceptual architecture to achieve decentralized data management. DM represents a paradigm shift towards a more data-oriented approach, with data organized into domains managed by agile, domain-focused teams. This shift is both structural and organizational, fostering improved data product cooperation.

The work in [27] explored the concept of transitioning from "data mess" to "data mesh", a decentralized data architecture framework that addresses scalability and governance issues within organizations. This approach organized data by business domains, such as marketing, sales and customer support, allowing domain-specific data producers to assume ownership and establish governance policies tailored to their expertise. As the paper presents, by decentralizing data management, data mesh promotes self-service data usage across the organization, avoiding inefficiencies of centralized systems while still utilizing traditional storage solutions like data lakes and warehouses.

Coined by Zhamak Dehghani in 2019 [22], DM is underpinned by four key principles: domain-oriented decentralized data ownership, self-service data infrastructure as a platform, data as products, and federated computational governance. These principles encourage domain teams to own and manage their data, treat data as products for internal and external users and maintain high-quality data governance. The authors in [28] investigated the operational mechanics, benefits, architectural elements, limitations and prospects of DMs, offering a comprehensive guide for organizations seeking to enhance their data management strategies.

Transforming DLs into DMs leads to a significant development concerning how organizations manage their data. By decentralizing data ownership and management, DM architecture empowers domain-oriented teams to take ownership of their data domains, creating a culture collaboration in terms of data [22]. This not only promotes agility and scalability but also equalizes access to data, enabling teams to make data-driven decisions independently. However, this transition requires significant organizational change, including restructuring teams and redefining roles and responsibilities [29]. It also introduces technical challenges, such as managing distributed systems and ensuring interoperability across different domains. Despite these difficulties, the potential benefits of improved agility, scalability and decentralization make the transformation to a DM architecture an important and vital transformation in the big data era.

3. Methodology

A novel approach for storing and retrieving large amounts of data is proposed here that aims to best serve efficient storing to and retrieval from DLs, while at the same time offering the means to transform a DL into a DM when needed. More precisely, a unique metadata mechanism based on SDB is established that enables blueprinting to characterize and describe the data sources and data items that are kept in a DL. A novel standardization framework based on this mechanism is also introduced to convert a DL into a DM by discovering data products and domains using semantic data blueprints (see Figure 2). The framework utilizes standardized descriptions in the form of blueprints to create data products using a domain driven approach. A real-world case-study from the domain of manufacturing is formed to demonstrate the proposed approach.

The data utilized are accessible via this link https://github.com/mfpingos/TechnologiesMDPI (accessed on 12 April 2024) and were collected within the PARG factory (https://paradisiotis.com/). PARG is one of the most significant local companies and experts in the field of poultry farming and trading of poultry meat in Cyprus. It provides a large assortment of items that are delivered to local supermarkets. The operational procedures and production data of the factory are confidential. Consequently, this paper discloses only a portion of the processes, providing limited details, and utilizes a masked and downgraded version of the data. Nevertheless, the case study sufficiently illustrates the fundamental principles of the proposed framework, validating its applicability and effectiveness.

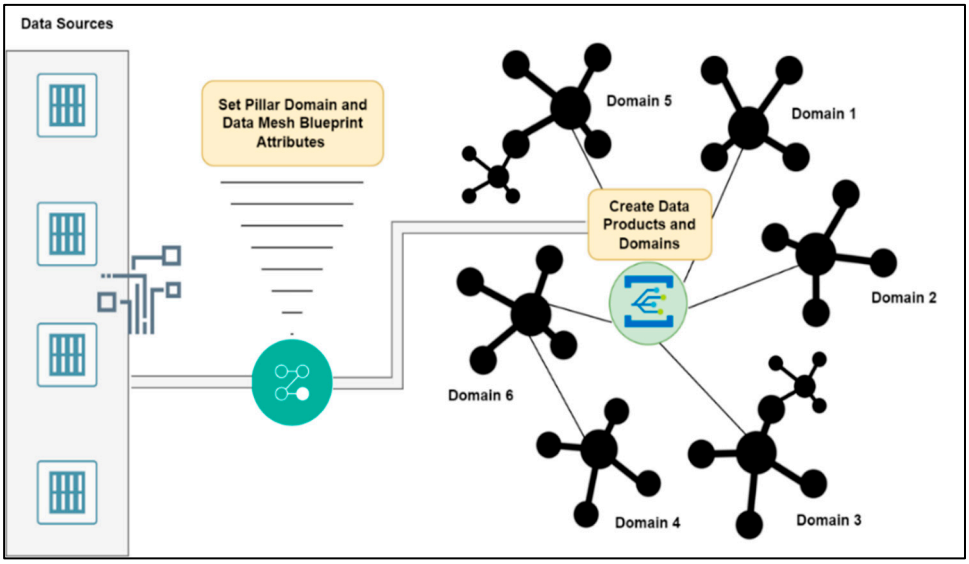

Figure 2. Summary of the proposed data mesh architecture.

The ability to discover data products and domains while creating DMs is based on a dedicated form of blueprint depicted in Figure 3. Actually, this may be regarded as a global blueprint that can be applied to any application domain and type of data, not only in the manufacturing area. Specifically, the blueprint provides a standardized form of describing data constituents and contains as a starting point the pillar domain, followed by subdomains. The pillar domain is the key operational attribute or category within the DM structure. It is the highest level of organization and acts as the starting point for structuring the data in the DM. Subdomains are more granular categories or attributes that refine and further detail the data organized under the pillar domain. They are secondary and tertiary levels of categorization within the DM architecture and are considered the more granular parts of the DM. A Terse RDF Triple Language (TTL) file is created for each level, which is written in XML format and describes the DM blueprint (see sample code provided in GitHub link https://github.com/mfpingos/TechnologiesMDPI). Using the manufacturing data as example, it will be demonstrated how the DM is constructed by creating appropriate data products and domains using the DM blueprint in Figure 3. A dedicated Python script (*finalpyoptfinal.py* file in GitHub) is developed which utilizes the DL metadata enrichment mechanism to create semantic annotation and enrichment and produce data products according to owner/user needs. The sample data originates from PARG's systems, which operate in different locations in the factory and monitor or facilitate chicken farming. These systems can be considered as data sources, collecting data and managing measurements from various sensors within the facilities of the factory. For example, the Flock Daily files contain daily measurements of a specific poultry farming unit's cycle. A typical farming cycle usually spans from 1 to 60 days. These files include daily battery temperatures, minimum/maximum/required temperatures and humidity measurements, with specific timestamps indicating when the sensor readings were captured/sent. Moving to the Flock Hourly files, these consist of hourly measurements for a particular day of the facilities and provide data on the hourly required temperature, temperatures of specific sensors and temperatures outside the facility, as well as measurements of humidity and carbon dioxide levels, all provided with corresponding timestamps for sensor data transmission. Examples of these data are also uploaded on GitHub (https://github.com/mfpingos/TechnologiesMDPI).

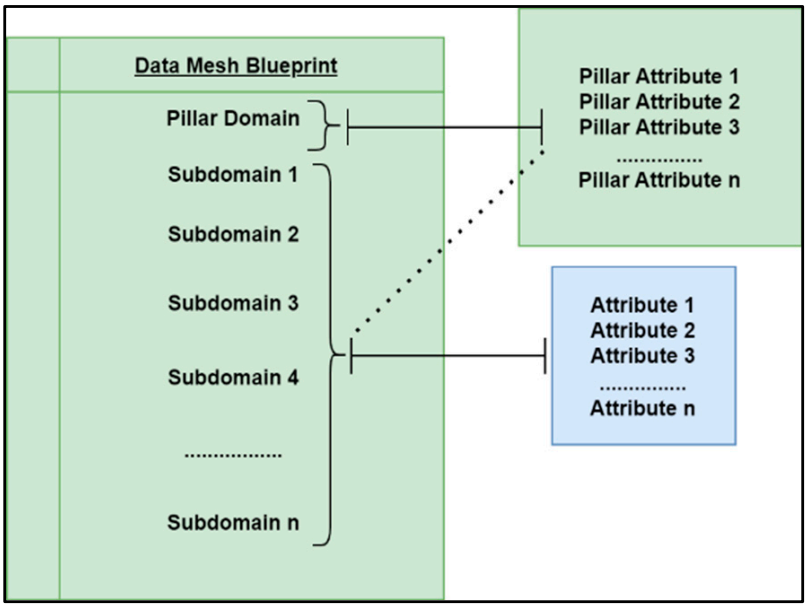

Figure 3. Data mesh blueprint.

Each data source in the PARG environment is described via an RDF using TTL format. In order to demonstrate the proposed DM framework, we have selected the following metadata characteristics to describe a source: (i) source name; (ii) location; (iii) feed cycle start; (iv) feed cycle end; (v) keywords; (vi) variety; (vii) velocity; (viii) volume; and (ix) source path. The corresponding description may be found at https://github.com/mfpingos/TechnologiesMDPI.

While DM is a decentralized data architecture that treats data as products according to business needs, it promotes domain-oriented ownership, with products and sub-products owners being responsible for the quality, discoverability and usability of the data. Figure 4 shows an example of how the DM Blueprint and DM architecture are constructed taking into consideration the metadata characteristics listed above: The pillar domain attribute (*Location* as Level 1), as selected by the DM owner, constitutes the main part of the DM structure, while selected subdomains (*velocity* as Level 2 and *variety* as Level 3) define the second and third level of refinement in the creation of the data products (structure also presented in Figure 2). The latter are treated as the next components of the DM architecture, providing the ability to create domains according to selected attributes expressed via the blueprint mechanism introduced in Figure 3. Each level of the DM consists of a TTL file that includes all the descriptions of the sources, which are filtered according to the level. A sample TTL description for Source 1 is presented in Figure 5. Let us now assume that we want to retrieve all the sources in the DM for the Data Product <<`Limassol|Daily|Structured` >>. The semantic Web framework Apache Jena is fed with the preferred characteristics of the attributes and executes the following SPARQL query:

```
SELECT ?flockid ?source_name ?source_path
            WHERE {?source rdf:type ex:Description;
          ex:flockid ?flockid;
          ex:source_name ?source_name;
          ex:source_path ?source_path;
          ex:location "Limassol";
          ex:variety "Structured";
          ex:velocity "Daily".  }
```

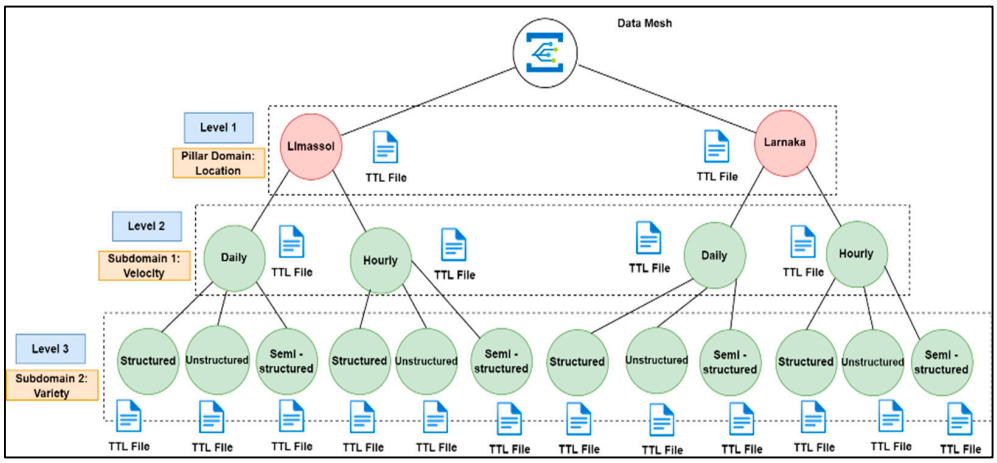

Figure 4. Creation of data mesh domains with PARG data.

```
ex:source1
  rdf:type ex:Description ;
  ex:source_name "1_FLOCK_Daily_EXPORT-07-05-2022" ;
  ex:flockid "1" ;
  ex:location "Limassol" ;
  ex:feedcycle_start "2022-05-07" ;
  ex:feedcycle_end "2022-05-20" ;
  ex:keywords "growDay, hour, requiredTemperature, coldTemperatureAlarm, hotTemperatureAlarm,
               sensor1, sensor2, sensor3, sensor4, sensor5, outsideTemp, currentAverageTemp,
               humidity, staticPressure, currentCO2, CO2HourMax, CO2HourMin" ;
  ex:variety "structured" ;
  ex:velocity "Daily" ;
  ex:source_path "hdfs://your-hadoop-namenode:9000/user/sources/daily_flock_1_data" ;
  ex:volume "76 KB" ;.
```

Part of TTL File

Figure 5. Creation of data mesh domains with PARG data—Source 1 TTL description.

The result of the above query execution consists of the metadata (*flock-ID*, *source name*, *source path*), which satisfies the query parameters (*Location*: `Limassol`, *Variety*: `Structured`, *Velocity*: `Daily`). According to the level at which the SPARQL query is executed, the execution time differs as demonstrated in the experiments section. As we move to including more data products (levels) more fine-grained information is produced and the execution time of the query becomes shorter. Therefore, the proposed DM architecture offers the ability to treat data as a list of data products according to specific business needs, while the pillar domains and subdomains are defined to reflect these needs via the DM blueprint presented in Figure 3.

To sum up, Figure 6 presents how our framework identifies data products and domains using SDBs and a series of steps for transforming a DL into a DM. The workflow begins by defining the metadata characteristics to describe each data source. In Step 2, the SDB are developed, serving as a standardized form of describing data constituents and defining the pillar domain and subdomains. In Step 3, the semantic annotation is created by using SDB to tag and enrich data sources with semantic metadata. This step involves creating RDF descriptions for each data source using a dedicated format. In Step 4, the SDBs are utilized to create data products based on the owner's needs. The latter are defined by selecting the attributes that will represent the pillar domain and subdomains. Therefore, these needs are matched with metadata characteristics to return the desired pieces of information stored in the DL in the form of data products. Finally, the DM architecture is constructed by organizing data products into domains and subdomains. The pillar domain acts as the highest level of information organization, while the subdomains provide secondary and tertiary levels of refinement.

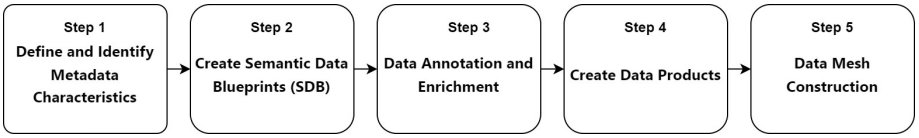

Figure 6. Detailed workflow to identify data products and domains using SDB.

The next section demonstrates the effectiveness of the proposed framework through its evaluation and comparison with other forms of a DL architecture, which are considered rivals or predecessors of DMs.

4. Qualitative Evaluation

This section aims to investigate, in general, the ability of creating data products via comparison between the proposed DM architecture and the following data structures of storage architectures:

- Traditional DL without the proposed metadata enrichment mechanism
- DL with a semantic metadata enrichment mechanism [14]

The selection of DLs as the counterapproach serves two purposes. The first is to show the differences between the widely known and used architecture of DLs and that of DMs. This will provide some indications about whether DMs can be regarded as the natural evolution of DLs in big data management. The second, since there is limited work on the topic, is to provide a comparison with the closest approaches, that is, with similar studies that have introduced the same concept of semantic enrichment and blueprints. This comparison will enable identifying the potential pros and cons of the two approaches.

The following characteristics/metrics were selected to facilitate comparison between the alternative architectures: (i) data domain readiness and alignment; (ii) granularity; (iii) decentralization; (iv) ease of storing and retrieval; and (v) agility.

Data domain readiness and alignment refers to the level of preparation of a particular data domain or set of data for analysis or processing. It involves ensuring that the data are accurate, complete, consistent, properly formatted and related to a specific domain. Once the data domain is deemed ready, a data product may be created and used for various purposes, such as building models, making predictions, generating reports or creating visualizations. Overall, ensuring data domain readiness is crucial for achieving accurate and meaningful results from business data analysis or processing tasks. Without proper preparation, the data could lead to incorrect or misleading insights and decisions.

Granularity refers to the level of detail at which data are collected, processed and analyzed. Granularity can be defined at different levels depending on the specific use-case, business requirements and data sources. To support different levels of granularity in a DL or DM, the data must be structured in a way that allows for easy querying, aggregation and analysis. This can be achieved through techniques such as data modeling, normalization and partitioning. By supporting different levels of granularity in a data storage architecture, organizations can ensure that each domain has access to the specific data they need to drive business outcomes. This can help to improve data quality, reduce data redundancy and promote collaboration across different teams and domains.

Decentralization in data architectures refers to the distribution of data across multiple nodes or storage systems instead of the reliance on a central data repository. This approach offers several advantages, including increased fault tolerance, improved scalability and greater flexibility in data management. In a decentralized storage architecture, data are distributed across multiple nodes or storage systems. Each node may contain a subset of the data or a complete copy. Nodes are connected to a network and can communicate with each other to exchange data and perform computations. This architecture can be organized in a variety of ways, such as peer-to-peer networks, distributed file systems or blockchain-based systems. Decentralization can improve fault tolerance by reducing the risk of a single point of failure. In a centralized architecture, if the central repository goes down, all access to the data is lost. In a decentralized architecture, the data are distributed

across multiple nodes, so if one node goes down, the others can continue to operate and serve data. Decentralization can also improve scalability by allowing data to be stored and processed in parallel across multiple nodes. This can improve the performance of data-intensive applications and enable them to handle larger volumes of data. Finally, decentralization can offer greater flexibility in data management by allowing data to be stored and processed closer to where they is being generated or used. This can reduce the latency and costs associated with transferring data to a central repository.

Agility in data storage architectures refers to the ability of an organization to quickly and easily adapt its infrastructure to meet changing business needs. This includes the ability to scale up or down, change data formats or structures, and integrate with new data sources or systems. Agility is important because it allows organizations to respond quickly to changes in their business environment, such as new regulations, new markets, or new opportunities. To achieve agility in data storage architectures, organizations must adopt flexible and scalable storage technologies and data management structures and practices that can be tailored to meet new business needs.

The characteristics described above are evaluated using a Likert linguistic scale including the values of low, medium and high. Table 1 provides a definition of these linguistic values for each characteristic introduced. In the case of the *Data domain readiness and alignment*, the levels are defined by the number of actions required to prepare the data for analysis or processing. A low level indicates that more than five actions are needed, a medium level requires two to three actions, and a high level necessitates only one action. These actions include tasks such as data cleaning and formatting, and aligning data with specific domains.

Table 1. Definition of low, medium, and high values of each characteristic.

Characteristic	Low	Medium	High
Data domain readiness and alignment	4–5 actions	2–3 actions	1 action maximum
Granularity	1 level	2 levels	3 or more levels
Decentralization	none or limited	normal	unlimited
Agility	none or limited	normal	unlimited

Granularity bears levels that are determined based on the number of detail levels supported by the architecture: one level for low, two levels for medium and three or more levels for high. This granularity ensures that data can be queried, aggregated and analyzed at various levels of detail according to business needs.

Decentralization is categorized based on the extent to which data are distributed across multiple nodes or storage systems. A low level indicates none or limited decentralization, with data being largely centralized. A medium level represents normal decentralization, with some distribution of data across nodes. A high level of decentralization means data are distributed in an unlimited manner, promoting fault tolerance and scalability.

Agility is assessed by evaluating the architecture's flexibility and ability to adapt to changing business needs. A low level signifies none or limited agility, where the system is rigid and slow to adapt. A medium level represents normal agility, with some capacity for adaptation. A high level indicates unlimited agility, where the architecture can rapidly scale, integrate new data sources and adjust data formats or structures as needed.

A traditional DL without semantic metadata enrichment can be characterized as having low *Data domain readiness and alignment*, as more than 5 actions are needed to prepare the data to create data domains and data products through the existing data residing in the DL. Naturally, this characteristic depends on whether semantic annotation is used in the DL. If not, then the DL is highly likely to become a data swamp where data domains are not distinct. A scheme with metadata enrichment, on the other hand, greatly benefits data domain readiness as it efficiently guides the retrieval process. *Granularity* also ranges according to the metadata semantic enrichment of the DL. When a DL does not

follow any semantic enrichment policy, it may be characterized as having low *Granularity*. *Decentralization* in DLs can be provided somehow only through data ponds and data puddles [26]. If a DL follows a flat architecture, then it can be characterized as having low *Decentralization* and low *Agility*, as it is quite difficult to adjust quickly to changes of business needs. The traditional DL without a metadata architecture was deliberately selected as an alternative approach for comparison purposes in order to demonstrate that without a metadata mechanism, a DL can indeed end up being a data swamp. Similarly, we argue here that a DM may suffer from a similar weakness, which may lead to becoming what we call here a data knot, that is, a route to a data product that is obstructed at some point before the full utilization of the relevant information is concluded due to the inability to combine semantics that lead to the product.

A DL with semantic enrichment, such as the one relying on blueprint metadata proposed in [15], can be characterized as having medium *Data domain readiness and alignment*, as three actions are needed to prepare the data in the DL to create data domains as follows: (1) Set pillar domains and subdomains according to business needs; (2) utilize ponds' and puddles' TTL metadata descriptions; and (3) create the DM with a pillar domains matching ponds metadata attributes and subdomains according to puddle attributes. These actions are basically creating data ponds and data puddles inside the DL using a domain driven approach with a maximum of two levels. The metadata mechanism in [14] also presents high *Granularity* because of the metadata enrichment included in the DL, and specifically, in the blueprint metadata history. High levels of *Granularity* are also achieved by using the data puddles, which are smaller portions of organized data.

Decentralization, as described above, can somehow be provided in DLs only through data ponds and data puddles as the framework in [26] suggests, and, of course, if distributed across multiple nodes or storage systems instead of relying on a central data repository as the original DL concept dictates. Finally, a DL enhanced with the blueprint semantic mechanism may be characterized as having high *Agility* due to the fact that it can quickly adopt changes in business needs by utilizing the keywords attribute in the relevant blueprint mechanism. On the contrary, a flat DL architecture does not offer such a flexibility and, thus, it is characterized as having low *Agility*.

The proposed DM architecture presented here achieves high *Data domain readiness and alignment*, *Granularity* and *Agility* due to the proposed DM Blueprint presented in Figure 3 and applied as demonstrated in Figure 4, which drives the creation of data domains and data products. *Decentralization* is one of the main characteristics of a DM architecture as presented in Section 2, while the proposed mechanism can be characterized as high for this feature.

Table 2 summarizes the points of the short comparison presented above between the DL and DM architectures and the utilization of the metadata enrichment mechanism proposed in this paper. It is evident that the use of the mechanism offers significant benefits to the underlying data structures used in storage architectures which outperform their rivals (i.e., without the mechanism) in all characteristics used. What is most important, though, is that DMs enhanced with the data blueprint mechanism improve their performance even further in terms of the *Data domain of readiness and alignment* and the *Decentralization* characteristics compared to the counter approach of a DL with the same mechanism.

Table 2. Evaluation and comparison of the mechanism and data structures of storage architectures.

Approach	Data Domain Readiness and Alignment	Granularity	Decentralization	Agility
Traditional DL without the proposed metadata enrichment mechanism	Low	Low	Low	Low
DL with the proposed metadata enrichment mechanism [15]	Medium	High	Medium	High
DM proposed architecture	High	High	High	High

5. Experimental Assessment

This section provides a short and concise description of the experiments conducted, starting with the design of the experiments and ending with discussing the results obtained.

5.1. Design of Experiments

The experimentation here aims to investigate, on one hand, the ability of the proposed approach to create refined data products, and on the other, to assess its performance and effectiveness with the execution of queries. In this context, a series of experiments were designed and executed to support the above targets. This sub-section describes the rationale behind their design.

Two alternative data structures of storage architectures were constructed to compare with the DM. The first one is a basic DL enhanced with a semantic enrichment mechanism based on blueprints, similar to the one reported in [14]. The second one is an upgraded version of the first, that is, a DL using the semantic enrichment mechanism but also structured with ponds and puddles, as presented in [26] and depicted in Figure 1. The selection of DLs as the counterapproach serves two purposes. The first is to show the differences between the widely known and used architecture of DLs and that of DMs. This will provide some indications about whether DMs can be regarded as the natural evolution of DLs in big data management. Since there is limited work on the topic, the second purpose is to provide a comparison with the closest approaches, that is, with similar studies that introduced the same concept of semantic enrichment and blueprints. This comparison will enable identifying the potential pros and cons of the two approaches.

Performance was assessed by varying the complexity of the experiments in terms of two factors: the number of sources producing data and the number of data products required. The former was set equal to three distinct levels, 100, 10,000 and 100,000, while the latter used five different values, that is, 2, 3, 4, 5 and 7. The value ranges of both factors were selected so that scaling up serves as a complexity rising factor, but at the same time, the lower and upper boundaries are reasonable for addressing real-world needs, and even exceed reality expectations (i.e., above 100 data sources), just to measure or compare performance. Data products for the PARG datasets were constructed at each level using the following characteristics: Level 2—`Location and Variety`; Level 3—`Location`, `Variety` and `Velocity`; Level 4—`Location`, `Variety`, `Velocity` and `Feed-cycle Start`; Level 5—`Location`, `Variety`, `Velocity`, `Feed-cycle Start` and `Feed-cycle End`; Level 7—`Location`, `Variety`, `Velocity`, `Feed-cycle Start`, `Feed-cycle End`, `Volume` and `Flock ID`. The varying complexity was targeted at investigating performance and efficiency of the proposed approach in terms of the time required for constructing the mesh (data products), as well as the ability and time for locating the appropriate sources to retrieve data from.

Description of the sources and their characteristics was performed using TTL files (uploaded on GitHub) and reflected the data characteristics provided by the PARG factory. The TTL files were created automatically by Python scripts that also masked the confidential data. Increasing the number of sources directly affects (increases proportionally) the size of the corresponding TTL file, which is the main element parsed to return sources matching a query. Indicatively, 100 sources described in a TTL file resulted in a size of 62 KB, with 10,000 sources of 6.1 MB and 100,000 sources of 61.2 MB. Additionally, the DL architecture with ponds and puddles was created for the same datasets to facilitate direct comparison with the DMs at the same level (Level 2 of data products). All DL and DM constructs were implemented by splitting information in different layers of granularity using the characteristics of the TTL files described in the SDB.

Experiments were executed on a server computer with three virtual machines, a CPU with 4 $\times$ dedicated cores (the base server hosting the machines had 48 cores), memory size of 8192 MB and hard disk capacity of 80 GB. The software stack included Hadoop (version 3.3.6) for distributed computing, Python (version 2.7.5) for scripting, the generation of data based on PARG's raw real-world data and the creation of the data products (DM level), and Apache Jena for SPARQL query processing.

Various queries were constructed and executed. (i) One reference query (Query#1) requires all description data to be returned for each relevant source and its purpose is to measure response time (i.e., the time to locate the relevant sources). (ii) Three performance assessment queries included Query#2, which retrieves `source names`, `velocity`, `feed-cycle start` and `feed-cycle end` for all descriptions; Query#3, which adds a filter to Query#2 to select only descriptions with a specific `velocity (monthly)`; and Query#4, which retrieves the `source names` and `velocity (monthly)` and calculates the duration of each feed cycle in days for descriptions with a specific velocity.

5.2. Experimental Results

Table 3 presents the execution time required to construct a DL with a metadata enrichment mechanism and ponds and puddles structure, and various forms of DMs in terms of data products (granularity levels), while, at the same time, varying the number of data sources. The simple DL structure (i.e., without ponds/puddles) was not included in Table 3 as the comparison with the other two alternatives would not be "fair" since it cannot semantically categorize information upfront and, hence, by default, it would fall short. As can be observed in Table 3, creation time increases according to the number of sources and granularity: DL and DM with two data products require the least time to create, with time steadily increasing as more data products are created, something which is expected.

Table 3. Creation time for each structure architecture used for experimentation with varying number of sources and data refinement levels.

Structure Architecture with Levels	Number of Sources	Creation Time (s)
DL with Ponds & Puddles/ Data Mesh Level 2	100	0.0229
	10,000	3.9879
	100,000	29.9931
Data Mesh Level 3	100	0.0673
	10,000	7.0856
	100,000	72.3088
Data Mesh Level 4	100	0.1075
	10,000	6.6337
	100,000	70.0903
Data Mesh Level 5	100	0.0906
	10,000	9.7921
	100,000	93.5849
Data Mesh Level 7	100	0.2379
	10,000	15.8697
	100,000	166.9752

The construction time for DMs with the maximum level used (seven data products) is substantially higher than it is for DMs with lower levels, with an increase of 10 to 15 times more than the previous value of the number of sources for the same level. It is worth noting that the maximum DM construction time is less than 3 min, which may be considered a quite satisfactory performance, taking into account the extreme test conditions with values equal to 100,000 for the sources and 7 for granularity level, which, in practice, are very rarely met.

The reference SPARQL query (Query#1) was then executed using the various DM structures for comparison purposes. The execution times of the query are listed in Figure 7, along with the number of sources returned. As may be observed, the query execution time

is dependent on the overall number of sources used and is analogous to the number of sources returned when there exist various sources satisfying the query (levels three and four). When granularity increases above level four, only a limited number of resources is returned (one, in this case), which leads to executing the query rapidly and stably, irrespective of the number of underlying data sources (see Figure 7). This is actually the most significant benefit of using the proposed DM structure, that is, to restrain the range of information categorized in the data product levels and retrieve data in an immediate and direct way.

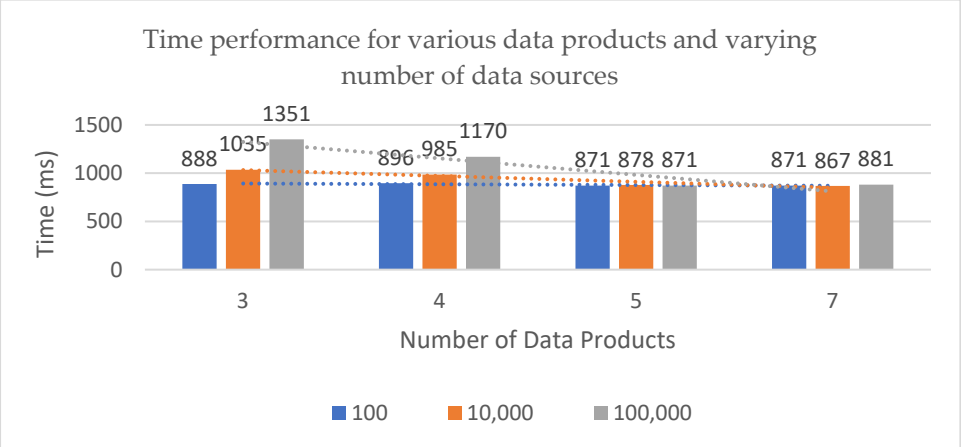

Figure 7. Execution of reference query on various DM architectures and varying data sources (10, 10,000, 100,000).

Finally, the same DM structures and data sources as above were utilized to execute the last experiments that used 3 SPARQL queries with varying complexity as previously described (uploaded also on GitHub). Figure 8 graphically depicts the results, which indicate consistent behavior across the queries. The average execution time after 100 iterations is quite low even with the maximum number of data sources tested; it increases proportionally to the number of available data sources, and it stabilizes as the number of sources returned saturates to one (data products equal to 5 and 7).

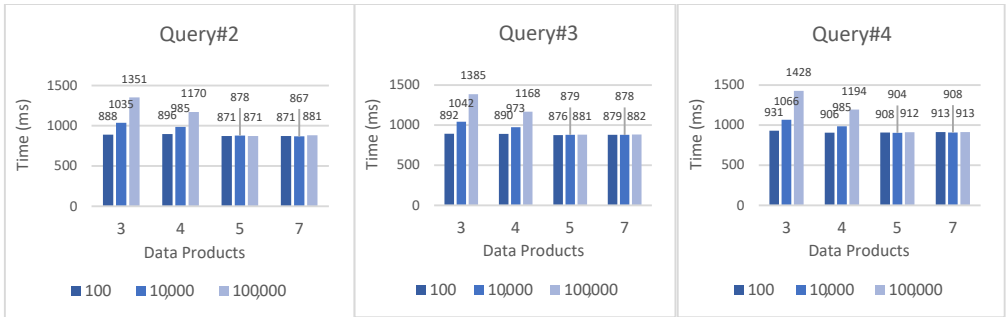

Figure 8. Execution of queries on various DM architectures with increasing complexity and varying the number of data products and number of sources (10, 10,000, and 100,000).

6. Discussion and Conclusions

This paper investigated the transformation of DLs into DMs by proposing a standardized approach to easily discover and construct data products. This approach modified and extended earlier work on DLs and their metadata enrichment achieved through SDBs [14,25,26]. This was performed by following a domain-driven approach and providing a new set of blueprints able to identify and describe data products. The proposed approach was demonstrated and validated in two ways: The first involved comparison

with alternative DL-based structures which indicated the superiority of DMs over a set of qualitative features. The second involved using real-world data collected within the environment of a poultry meat factory. A set of experiments was designed and executed which revealed a successful performance both when compared to DLs with similar semantic enrichment mechanisms and when varying complexity in terms of the available data sources, the number of data products created and the type of queries run.

One may argue that since a DM requires some time to create, which depends on the type and number of data products required, as well as the number of sources producing the data utilized, time performance may constitute a drawback hindering its wider adoption as the data products must be available before the execution of any queries. This could lead to characterizing the nature of DMs as rather static in the sense that if data products need to change according to new business needs, then the DM must be recreated to accommodate changes. Nevertheless, as shown in the experiments conducted, the time it takes to create the mesh and the corresponding data products is very short. In addition, and more importantly, the execution of the queries once the data products are in place is far quicker than it is for other similar storage architecture structures. Therefore, even with very large volumes of data, DMs prove adequate to handle efficiently data retrieval. This advocates in favor of using DMs as the underlying data management structure for practically any real-world application domain.

Combining a DM with software analytics may offer useful information on software processes and products, such as:

- Granular Insights: A DM enables individual teams to own their data, allowing them to use software analytics methods unique to their software systems. This strategy offers fine-grained insights into usage patterns, team-specific utilization performance, and other pertinent indicators;
- Contextual Awareness: By utilizing DM principles, teams increase their awareness of the context of the data they produce and how it relates to their software processes and products. This context gives a greater understanding of the variables affecting software performance and behavior, which improves the usefulness of software analytics;
- Rapid Iteration and Improvement: A DM provides teams with control and autonomy over their data, allowing them to iterate and enhance software products and procedures quickly using the knowledge gleaned from software analytics. Continuous improvement and agility are fostered by this iterative methodology.

Future research steps will include the following: The DM architecture with the semantic metadata enrichment mechanism presented in this paper was built by primarily suing independent software modules built in Python as needed to support experimentation. Therefore, future work will aim at implementing an integrated software environment to facilitate the creation of data products and defining the level of granularity in a user-friendly and uniform manner. This will enable us to assess our framework more thoroughly and, in particular, to compare it more closely to other current DM systems using specific performance measures, although this data structure storage architecture is quite new. Additionally, we plan to utilize blockchain technology and smart contracts to enhance privacy, security and data governance in DMs. Finally, we will investigate how machine learning models may be applied to enhance the efficiency of the proposed framework, and more specifically, how such models may be trained via queries created by users to suggest better DM structures to DM owners. Along these lines, we will also investigate the potential of integrating DM blueprints with recommendation engines so that historical data describing user preferences when interacting with the mesh in the past dictate the upfront creation of new data products foreseen to be useful for serving future needs.

Author Contributions: Conceptualization, M.P. and A.S.A.; methodology, M.P. and A.S.A.; software, M.P; validation, M.P. and A.S.A.; formal analysis, M.P.; investigation, M.P.; resources, M.P. and A.S.A.; data curation, M.P.; writing—original draft preparation, M.P.; writing—review and editing, M.P. and A.S.A.; visualization, M.P.; supervision, A.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

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

References

1. Rodríguez-Mazahua, L.; Rodriguez-Enriquez, C.A.; Sánchez-Cervantes, J.L.; Cervantes, J.; García-Alcaraz, J.L.; Alor-Hernández, G. A general perspective of Big Data: Applications, tools, challenges, and trends. *J. Supercomput.* **2016**, *72*, 3073–3113. [CrossRef]
2. Chen, M.; Mao, S.; Liu, Y. Big data: A survey. *Mob. Netw. Appl.* **2014**, *19*, 171–209. [CrossRef]
3. Howarth, J. 30+ Incredible Big Data Statistics. 2023. Available online: https://explodingtopics.com/blog/big-data-stats (accessed on 10 February 2024).
4. Kościelniak, H.; Puto, A. Big data in decision making processes of enterprises. *Procedia Comput. Sci.* **2015**, *65*, 1052–1058. [CrossRef]
5. Santos, M.Y.; Costa, C. *Big Data Concepts, Techniques, and Technologies*; River Publishers: Aalborg, Denmark, 2020; pp. 9–36.
6. Luckow, A.; Kennedy, K.; Manhardt, F.; Djerekarov, E.; Vorster, B.; Apon, A. Automotive Big Data: Applications, workloads and infrastructures. In Proceedings of the IEEE International Conference on Big Data (2015), Santa Clara, CA, USA, 29 October–1 November 2015. [CrossRef]
7. Kim, Y.; You, E.; Kang, M.; Choi, J. Does big data matter to value creation? Based on oracle solution case. *J. Korea Soc. IT Serv.* **2012**, *11*, 39–48. [CrossRef]
8. Herschel, R.; Miori, V.M. Ethics & Big Data. *Technol. Soc.* **2017**, *49*, 31–36. [CrossRef]
9. Gandomi, A.; Haider, M. Beyond the hype: Big Data Concepts, methods, and analytics. *Int. J. Inf. Manag.* **2015**, *35*, 137–144. [CrossRef]
10. Blazquez, D.; Domenech, J. Big data sources and methods for social and economic analyses. *Technol. Forecast. Soc. Change* **2018**, *130*, 99–113. [CrossRef]
11. Machado, I.A.; Costa, C.; Santos, M.Y. Data Mesh: Concepts and principles of a paradigm shift in data architectures. *Procedia Comput. Sci.* **2022**, *196*, 263–271. [CrossRef]
12. Eichler, R.; Giebler, C.; Gröger, C.; Hoos, E.; Schwarz, H.; Mitschang, B. Enterprise-wide metadata management. In Proceedings of the 24th International Conference Business Information Systems, Hannover, Germany, 15–17 June 2021; pp. 269–279. [CrossRef]
13. Dehghani, Z. Data Mesh: Delivering Data-Driven Value at Scale. Available online: https://www.thoughtworks.com/insights/books/data-mesh (accessed on 12 February 2024).
14. Pingos, M.; Andreou, A. A Data Lake Metadata Enrichment Mechanism via Semantic Blueprints. In Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering, Virtual, 25–26 April 2022; pp. 186–196. [CrossRef]
15. Jung, C. From Data Lakes to Data Mesh: A Guide to the Latest Enterprise Data Architecture. Available online: https://towardsdatascience.com/from-data-lakes-to-data-mesh-a-guide-to-the-latest-enterprise-data-architecture-d7a266a3bc33 (accessed on 28 March 2024).
16. Xin, R.S.; Rosen, J.; Zaharia, M.; Franklin, M.J.; Shenker, S.; Stoica, I. Shark: SQL and rich analytics at scale. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 22–27 June 2013; pp. 13–24. [CrossRef]
17. Leclerc, B. What Is a Data Lake? Available online: https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/ (accessed on 12 March 2024).
18. Khine, P.P.; Wang, Z.S. Data Lake: A new ideology in Big Data Era. In *ITM Web of Conferences*; EDP Sciences: Les Ulis, France, 2018; Volume 17, p. 03025.
19. Sawadogo, P.; Darmont, J. On Data Lake Architectures and metadata management. *J. Intell. Inf. Syst.* **2020**, *56*, 97–120. [CrossRef]
20. Dehghani, Z. Data Mesh Principles and Logical Architecture. Available online: https://martinfowler.com/articles/data-mesh-principles.html (accessed on 15 March 2024).
21. Dehghani, Z. Data Mesh. Available online: https://www.oreilly.com/library/view/datmesh/9781492092384 (accessed on 23 March 2024).
22. Dehghani, Z. How to Move beyond a Monolithic Data Lake to a Distributed Data Mesh. Available online: https://martinfowler.com/articles/data-monolith-to-mesh.html (accessed on 28 March 2024).

23. Papazoglou, M.P.; Elgammal, A. The Manufacturing Blueprint Environment: Bringing Intelligence into manufacturing. In Proceedings of the International Conference on Engineering, Technology and Innovation 2017 (ICE/ITMC), Madeira Island, Portugal, 27–29 June 2017. [CrossRef]
24. Pingos, M.; Christodoulou, P.; Andreou, A. DLMetaChain: An IoT Data Lake Architecture Based on the Blockchain. In Proceedings of the 13th International Conference on Information, Intelligence, Systems & Applications (IISA), Corfu, Greece, 18–20 July 2022; pp. 1–8. [CrossRef]
25. Pingos, M.; Andreou, A.S. A smart manufacturing data lake metadata framework for process mining. In Proceedings of the Seventeenth International Conference on Software Engineering Advances ICSEA 2022, Lisbon, Portugal, 16–20 October 2022; Volume 11. [CrossRef]
26. Pingos, M.; Andreou, A.S. Exploiting Metadata Semantics in Data Lakes Using Blueprints. In *Proceedings of the International Conference on Evaluation of Novel Approaches to Software Engineering*; Springer Nature Switzerland: Cham, Switzerland, 2022; pp. 220–242. [CrossRef]
27. Panigrahy, S.; Dash, B.; Thatikonda, R. From data mess to data mesh: Solution for futuristic self-serve platforms. *Int. J. Adv. Res. Comput. Commun. Eng.* **2023**, *12*, 677–683. [CrossRef]
28. Wrembel, R. Data integration revitalized: From data warehouse through data lake to data mesh. In Proceedings of the International Conference on Database and Expert Systems Applications 2023; Springer Nature Switzerland: Cham, Switzerland, 2023; pp. 3–18.
29. Machado, I.; Costa, C.; Santos, M.Y. Data-driven information systems: The data mesh paradigm shift. In *Information Systems Development: Crossing Boundaries between Development and Operations (DevOps) in Information Systems (ISD2021 Proceedings)*; Insfran, E., González, F., Abrahão, S., Fernández, M., Barry, C., Linger, H., Lang, M., Schneider, C., Eds.; Universitat Politècnica de València: Valencia, Spain, 2021.