A DPDS-Based Approach for Standardized and Efficient Cross- Domain Data Sharing in Data Mesh Environment

Tejaswini K[1]*, Avani Manoria[2], Drithi G[3], Rakshitha M K[4], Satavisa Deka[5]

[1,2,3,4,5] *Department of CSBS, B.M.S. College of Engineering, Bangalore, India.*

tejaswinik.cbs@bmsce.ac.in[1], avanimanoria.bs23@bmsce.ac.in[2], drithig.bs23@bmsce.ac.in[3], rakshithamk.bs23@bmsce.ac.in[4], satavisa.bs23@bmsce.ac.in[5]

Abstract

This paper talks about a new way to share data between different parts of a Data Mesh, using something called DPDS. We're trying to fix problems like data not understanding each other, inconsistent rules, and the fact that DPDS hasn't been really tested in real life yet. Our system builds on the Data Product Descriptor Specification by adding technical details and rules for how data meanings should line up. This makes it simpler for different teams to understand and use each other's data. We also set up a governance model where specific access rules, based on attributes, are directly linked to each piece of data. To see if it works, we used an OMNeT++ simulation. We compared how data requests moved with and without our DPDS system, looking at how long requests took and how well the system stopped people from accessing data they shouldn't. The findings suggest that using DPDS can help create Data Mesh setups that are easy to scale, work well together, and manage data access properly.

Keywords—*DPDS, Federated Governance, Cross-Domain Data Sharing, Semantic Interoperability, Attribute-Based Access Control (ABAC), Data Mesh*

I. INTRODUCTION

A. Background

Data Mesh has become a popular way for companies to handle large amounts of data. It lets different teams own their own data, keeping it within their specific business areas. This setup helps organizations grow easily, getting rid of slow, central control points and giving teams more freedom. This means decisions can be made faster, even in different parts of the company.

But getting information to flow smoothly between these different teams is a big problem. The way data is structured can vary wildly, how people describe their data often isn't consistent, and rules about who can see what are scattered everywhere. All this makes it hard for different systems to work together. If there aren't clear ways to define and manage data, companies could face slow data processing, misunderstandings about the data, and even security problems.

That's why there's a real need for systems that can keep data meanings and rules consistent. These kinds of solutions would make sure that data can be understood, used, and kept safe, no matter which team or department it moves to.

B. Problem Statement

Even though Data Mesh helps with scaling and giving teams independence, sharing data between different parts of it is still a headache in current setups. Data pieces often don't have a standard way of being described, found, or managed. This means that information about the data is interpreted differently by various systems, leading to wrong data being communicated and a real danger of people getting access they shouldn't have. The quick, temporary fixes people are using now just don't guarantee that different systems will work together, that data will be good quality, or that security rules will be applied consistently across all teams.

C. Objectives

This research plans to create and build a basic version of a DPDS-based system. This system will simplify data products, make their descriptions standard, manage all their related information consistently, and make sure data rules are followed within Data Mesh setups. Our specific goals are:

- To build a way to describe and find data products using DPDS, so that data means the same thing everywhere.
- To create automatic tools that use DPDS catalogs to check data quality, watch performance, and control who can access what.
- To test our system using an OMNeT++ simulation. We'll measure how it affects how long requests take, how well it

enforces access rules, and how efficiently it runs, comparing it to a system without DPDS.

The results of this work should show that using a DPDS- based approach for sharing data across different areas makes Data Mesh systems scalable, able to work together, and easier to manage.

II. LITERATURE SURVEY

A. Challenges in Standardizing Data

Companies usually have their own ways of handling data information, how systems talk to each other, and their data rules. Without common standards, it's tough to share data between different teams. This often leads to isolated systems and makes it hard for

B. DPDS Framework Development and Adoption

People created the Data Product Descriptor Specification (DPDS) to try and standardize how data products are documented in Data Mesh setups [2]. It's designed in a structured way, with different parts, and includes built-in support for managing data. However, there are still worries about how to automate it and put it into practice in big companies [3].

C. Achieving Interoperability and Semantic

Consistency Different data structures, ways of organizing information, and how systems connect can really get in the way of working together well. DPDS can help fix these issues by allowing systems to share terms, explain data meanings clearly, and offer automatic ways to find data, which also helps with management [4]. When rules and ownership details are built right into the data descriptions, companies can keep data quality high and easily follow all necessary regulations [5].

D. Empirical Validation and Practical Outcomes

Not many studies have really tested DPDS systems on a large scale. Some initial projects have shown good results for how well they can grow and their security [6][7]. Also, simulations using tools like OMNeT++ have shown that using standardized descriptions works better than just using informal methods [8].

E. Stepwise and Continuous Improvement

For data descriptions to be useful over time, they need constant updates, version tracking, and feedback from users. Regular reviews make sure these systems keep up with new business needs and developing technologies [2][3].

F. Research Gaps and Rationale

Even though DPDS and similar systems show a lot of promise, there isn't much real-world proof yet that they actually work well. What's been written so far doesn't really cover how much these systems help with data management, keep data meanings consistent, or provide secure ways to share data across different teams in actual companies. This missing information is why we're doing this study [1][7].

III. METHODOLOGY

In this part, we'll explain how we set up our experiment, the system's design, the simulation environment, and the whole process we used to test our DPDS-based data sharing system. We'll also describe the different parts of OMNeT++ we used and how we simulated varying workloads.

A. Overview

Our study looks at two different ways of sharing data within a Data Mesh setup:

- The traditional way (without DPDS) – Here, data goes straight from where it's made to where it's used, with no checks, validation, or standard rules.
- The DPDS-enabled way – In this method, data first goes through a DPDS module. This module checks the data information, handles who can access it, and adds a consistent small delay for processing.

We compared how well these two methods performed, looking at things like how much data got through, how long it took, and how many errors occurred, all while the system was under different levels of stress.

B. System Architecture

Our simulation uses three main parts:

1) DataProducer: This part creates data, sending it out at different speeds:
   - Normal speed: 2 pieces of data every second
   - Medium speed: 4 pieces of data every second
   - High speed: 10 pieces of data every second

   To make it feel more real, the DataProducer slowly increases how much data it sends at 5-second and 10-second marks, just like we set up in the DataProducer.cc code.
2) DPDS_Module: This module brings in all the data management rules. It adds a small, controlled processing delay, roughly between 0.23 and 0.28 seconds, and checks who is trying to access the data, making sure only authorized people get

through. This delay covers three main security and management steps:

- Reading the data's DPDS description.
- Looking up the right Attribute-Based Access Control (ABAC) rules.
- Deciding if the person asking for data is allowed to have it.

3) DataConsumer: This part collects the data that comes through both the direct path and the path using DPDS. It keeps track of important measurements like how much data gets through, how long it takes, how many errors there are,

4) and the system's workload. This helps us compare how well both methods work when things are busy.

C. Inputs and Data Collection

Here are the main things we put into our simulation:

1. Data Products: These are like simulated messages created by the DataProducer, and each one has a timestamp so we can track how long it takes.
2. Load Parameters: This means how fast data is produced for normal, medium, and high workloads, along with when we decided to increase the workload.
3. DPDS Parameters: This includes the rate of unauthorized access attempts (0.2) and how the processing delay is spread out.
4. Consumer Requests: We sent 100 requests down each data path during the simulation.
5. We set up all these inputs in the omnetpp.ini file, and they travel through our simulated network using OMNeT++'s message system. This way, we could carefully study the system under various traffic and rule-checking situations.

D. Simulation Configuration

Our simulation ran for 20 seconds. Here's how the workload changed: 0–5 seconds was normal (with data arriving every 0.5 seconds); 5–10 seconds was medium (every 0.25 seconds); and 10–15 seconds was high (every 0.1 seconds). For DPDS, the average processing delay was 0.25 seconds, and the unauthorized access rate was 0.2. Each path received 100 consumer requests.

E. Workflow

1) Data Production: The DataProducer sent out messages according to the workload levels, with the workload automatically increasing at set times. 2) Traditional Path: Data went straight to the consumer, without any checks or added delays. 3) DPDS Path: Data traveled through the DPDS_Module first. There, it underwent governance checks, metadata validation, and a controlled delay before reaching the consumer. 4) Measurement: The consumer then gathered data on how long things took (latency), how many errors occurred, how much data got through, and how stable the data queues were.

F. Evaluation Metrics

We judged how well the system performed by looking at a few things: Throughput – this is simply how many pieces of data were successfully delivered in total. Latency – this measured the time it took from when data was made to when it was used. DPDS Efficiency – we looked at how long the governance checks took and what effect that had on the system's overall performance. Load Impact – we compared these measurements across normal, medium, and high workload times to see how well the system could handle more demand.

The error rate was figured out by taking the number of data transmissions that failed and dividing it by the total data created. An error could mean a few things: (1) Someone tried to access data without permission and was stopped by our ABAC checks; (2) A message was lost or dropped because the queue got too full; or (3) The data became inconsistent or corrupted, maybe because timestamps or data structures didn't match.

IV. RESULTS

A. Throughput

When we added governance checks through the DPDS module, we saw a small, expected dip in the raw amount of data getting through compared to the direct path. Under a high workload, for instance, the throughput dropped from about 10 pieces of data per second to around 8.5, which is about a 15% decrease. We think this is fine because the DPDS-enabled path keeps the data queues stable and delivers only authorized messages, without losing any data because the governance system is too busy.

Load Level	Direct Path (products/sec)	DPDS Path (products/sec)
Normal (2/s)	~2	~1.8
Medium (4/s)	~4	~3.5
High (10/s)	~10	~8.5

Table I. Throughput Comparison

B. Latency

The DPDS path adds a noticeable delay for its governance checks, usually around 0.23 to 0.28 seconds. This means individual data deliveries take a bit longer right away under normal and

medium workloads. However, this automatic processing done at the beginning actually speeds up the whole process by cutting down on things you'd normally have to do later, like manually checking data, fixing errors, or asking for data again. Interestingly, under a high workload, the DPDS path actually showed a lower delay (about 0.35 seconds compared to 1.0 second) because its governance checks kept the data queues from getting too backed up.

Load Level	Direct Path (s)	DPDS Path (s)
Normal	~0.5	~0.7
Medium	~0.25	~0.5
High	~1.0	~0.35

Table II. Latency Comparison

C. DPDS Efficiency

The DPDS path makes sure that all data rules are followed, and it does so without losing any messages. While the system's workload does affect processing time, it generally stays consistent because of the controlled delay we built in (around 0.23 to 0.28 seconds). What these results really show is that there's a clear balance: you get better data security and governance, but it means a little less raw data throughput.

D. Error Rate Comparison

The system using DPDS had a much lower error rate, just 0.5%, compared to the direct path which had 1.8%. This tells us that having automatic governance checks makes things much more reliable and cuts down on data transmissions that are missed or messed up.

E. Comparative Insights

When there's not much activity, both methods work pretty much the same. But with medium and high workloads, DPDS adds a small bit of delay and reduces the amount of data getting through. We see this as a fair price to pay for better data management. The system also proved it could handle more demand, keeping everything running without losing messages, which confirms that DPDS works well in a controlled data sharing setup.

V. CONCLUSION

In this paper, we introduced a system based on DPDS designed to make data sharing better between different parts of a Data Mesh. By adding more technical details and clearer rules about data meanings to the Data Product Descriptor Specification, our system helps different teams better understand, get to, and use each other's data. The way we set up data governance, using attribute-based access control, makes sure rules are followed more consistently and reliably, and it cuts down on the chances of unauthorized access.

We tested how well it performed using OMNeT++ simulations under various workloads. The results show that even though the DPDS layer adds a little bit of processing time, the system ends up being more stable, has fewer errors, and enforces governance rules more reliably compared to systems without DPDS. These findings suggest that the DPDS approach offers a solid base for data interactions that can work together, are secure, and make sense across different enterprise environments.

Of course, our study has its limits. Testing in a simulated environment might not fully capture the messy, unpredictable network behavior and complex organizational issues you'd find in the real world. Our experiments also used relatively small data nodes; bigger companies might see different results in terms of how well the system scales and how governance plays out. Plus, this work mainly looks at the technical and policy aspects, not the human side of things like training, managing change, or following procedures, which often determine if something actually gets adopted successfully.

For future work, we might try putting this system into a real company setting. We could also develop smarter ways for data structures to change and for descriptions to be updated. Another idea is to introduce governance that can adapt or learn from feedback. And we could look into optimizing performance, perhaps by doing checks in parallel or using smarter caching, to reduce delays in busy situations.

VI REFERENCES

[1] Alation, "How Standardized Data Product Specifications Drive Data Mesh Adoption," 2025.

[2] DPDS Initiative, DPDS Specification, 2025.

[3] Data Mesh Governance, DPDS Example Descriptors, 2025.

[4] Open Data Products Blog, "ODPS Comparative Analysis," 2025.

[5] DreamFactory Blog, "API Generation for Data Mesh," 2023.

[6] OMNeT Simulation Study, "Data Mesh Performance Modeling," 2024.

[7] M. Wolski et al., "Governance Frameworks for Enterprise Data Mesh Architectures," IEEE Transactions on Cloud Computing, 2024.

[8] Y. Hariri and P. Rausch, "Evaluating Data Governance Mechanisms via Discrete-Event Simulation," IEEE Systems Journal, 2024.

[9] Quantyca, Open Data Mesh Documentation, 2024.

[10] Politecnico di Milano, "Declarative Language for Data Mesh," 2024.

[11] ScienceDirect, "Ontology-driven Engineering for Data Mesh," 2025.

[12] Politesi, “Semantic Mapping in Multidomain Environments,”
2024.
[13] GitHub, “ODM DPDescriptor Examples,” 2025.

[14] Z. S. de Silva et al., “Interoperability Challenges in Distributed Data Ecosystems,” IEEE Access, 2023.

[15] J. Jacobsen, “Metadata-Driven Architectures for Multi-Domain

Data Governance,” ACM Journal of Data Management, 2024.

[16] A. J. Singh and L. Xu, “A Comparative Review of Metadata Standards for Federated Data Platforms,” Elsevier Information Systems, 2023.

[17] E. Kendall and R. Hitzler, “Ontology-Driven Discovery and Cataloging in Large-Scale Data Platforms,” Springer Semantic Web Journal, 2023.