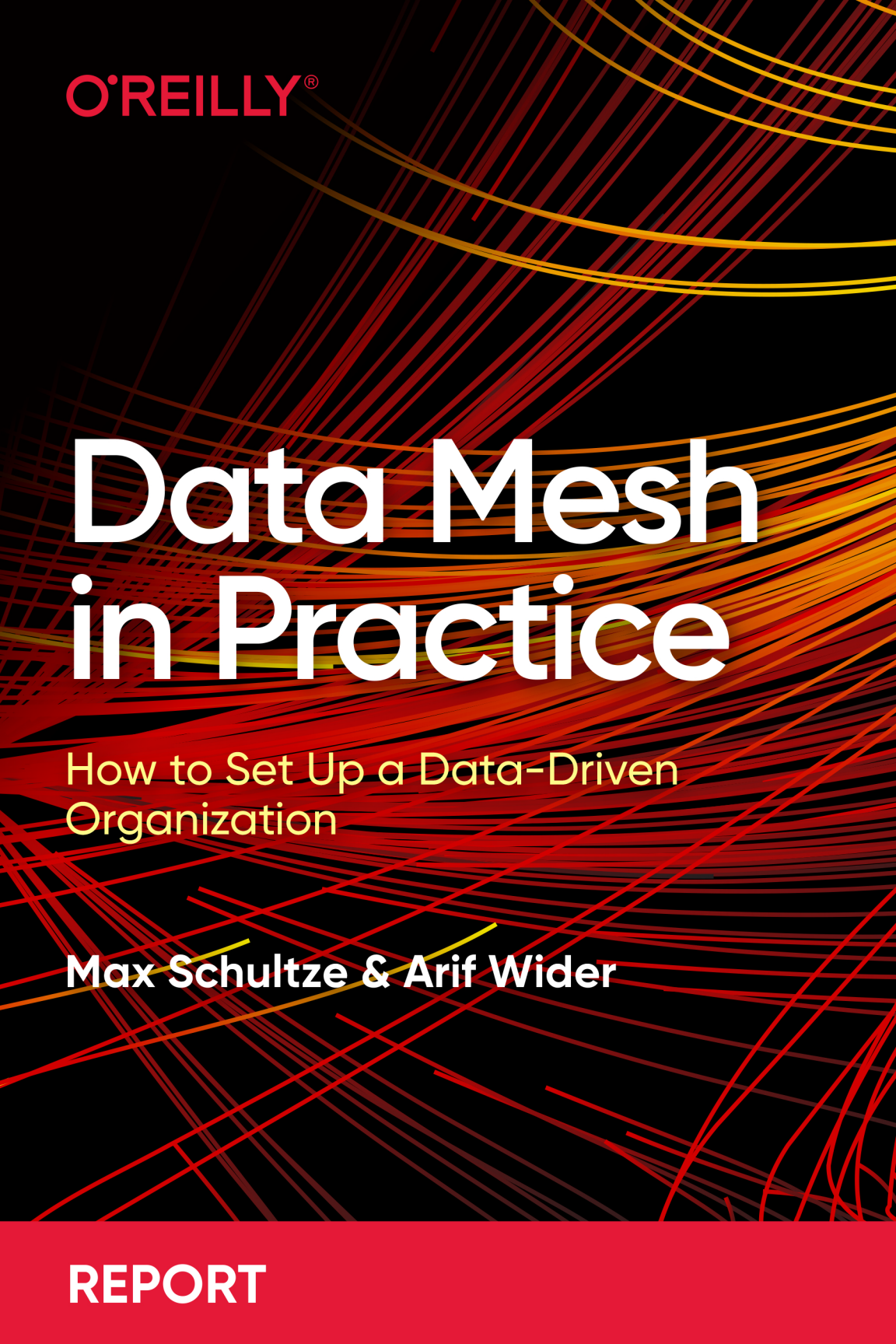
O'REILLY®
Data Mesh in Practice
How to Set Up a Data-Driven Organization
Max Schultze & Arif Wider
REPORT

Data Mesh in Practice

How to Set Up a Data-Driven Organization

Max Schultze and Arif Wider

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Data Mesh in Practice
by Max Schultze and Arif Wider

Copyright © 2022 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Andy Kwan
Development Editor: Melissa Potter
Production Editor: Gregory Hyman
Copyeditor: nSight, Inc.
Proofreader: Amnet Systems, LLC
Interior Designer: David Futato
Cover Designer: Randy Comer
Illustrator: Kate Dullea

December 2021: First Edition

Revision History for the First Edition
2021-12-16: First Release

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Data Mesh in Practice*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

The views expressed in this work are those of the authors and do not represent the publisher's views. While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

This work is part of a collaboration between O'Reilly and Starburst Data, Inc. See our statement of editorial independence.

978-1-098-10849-6

[LSI]

Table of Contents

Introduction v

Part I. What Is Data Mesh and Why Do We Need It?

1. Pain Points of Centralized Data Responsibility 3
The Data Warehouse Approach 4
The Data Lake Approach 4
Centralized Data Responsibility 6

2. The Pillars of Data Mesh 9
Decentralized Domain Ownership of Data 10
Data as a Product 10
Self-Serve Data Infrastructure as a Platform 13
Federated Computational Data Governance 14

Part II. The Data Mesh Journey

3. Getting Started: A Data Product–Centered Mindset Shift 19
The Rise of Big Data Technologies 19
Case Study: Pain Points of Unclear Data Ownership 21
Moving Toward Decentralized Data Products 22
Where and How to Get Started 25

4. Scaling the Mesh: Self-Serve Data Infrastructure. 29
Pain Points of Central Data Responsibility 30
Case Study: Centralized Compute Capabilities 31
Data Infrastructure Capabilities 35

5. Sustaining the Mesh: Federated Computational Data Governance 37
Ensure Interoperability Through Semantic Cross-Domain Modeling 37
Use Automation to Enforce Global Rules Without Centralization 39

6. Industry Practices. 43
Common Pitfalls 43
Best Practices 45

Closing Remarks. 49

Acknowledgments. 51

Introduction

When you, the reader, are curious enough to jump into a lengthy in-practice report on *data mesh*, chances are you have already heard of the term or maybe have already read some introductory content on the topic. You might be looking for in-depth technical advice on how to implement a data mesh. Or you might be looking for real-world examples that will help you move from a general understanding to the concrete ability to assess whether data mesh is the right tool for you. In the latter case, you are certainly at the right place. In the former case—detailed technical instructions—it's a bit more complicated: we, the authors, are engineers at heart, so we love talking about technology and concrete practical solutions. However, data mesh is to a large degree an organizational topic that is independent of specific technologies, which is why it has to be interpreted by every company, depending on its specific situation. There simply is no 10-step program to get to the correct data mesh, because the right mesh will look a bit different for each company. Nonetheless, this report is intended to give you a clear picture of what capabilities you want to look out for, as well as the "why?" behind those capabilities. It should equip you much better to choose the right tools for the job, once you understand what exactly you want to accomplish. Also, moving to a successful data mesh setup is a long organizational process, which means most organizations will never be done with it. They will, rather, get better at practicing a data mesh mindset in their pursuit of generating actual business value from analytical data.

We do not expect you to know what data mesh is, so we will introduce you to all its important concepts in the following sections. We do expect, though, that you have had some exposure to the

challenges of managing analytical data, be it from a managing perspective or an on-the-ground perspective (e.g., as a data engineer, developer, or data scientist; or as a data analyst or product manager working with data). Also note that data mesh is not the right approach for every company. Data mesh provides solutions for problems with scaling. To be specific, it is about organizational scaling in terms of proliferation of use cases and diversity of data sources and not so much about technical scaling in terms of sheer size or velocity of data—but more on that later. Not every company is large (or growing quickly) and is using data to improve its business model. If your company does not have more than two engineering teams or your business model is extremely stable, you might simply not experience the issues that data mesh is addressing. Nevertheless, having a good knowledge about data mesh could prove useful in case you run into those situations at a later point.

This report is structured into two major parts. In the first part, we introduce the data mesh concept. First, we reflect on the pain points of working with analytical data that many companies are facing today and that lead to the need for a change. After that, we introduce the core pillars of data mesh. In the second part of this report, we will take you on a practical journey of how to get started, how to scale the mesh, and how to achieve a sustainable impact across your organization. This part contains case studies and practical examples to visualize the path that many of you are just stepping onto. To close out the report, we have collected a series of common pitfalls and best practices to ensure you have an additional selection of references to learn from.

PART I

What Is Data Mesh and Why Do We Need It?

For at least three decades now, companies have been trying to get to a place where they can easily use analytical data to improve their business. For example, they try to utilize data about the behavior of their customers and about the usage of their products to generate clear and actionable insights that help them to run a better business and build better products. Although there are a few success stories in this regard, when talking to leaders and practitioners within the industry, we are faced with countless struggles that seem to go nowhere and create a lot of frustration. The now overused analogy of oil and data has triggered a rush of investment in big data technologies and business intelligence capabilities that often do not meet the expectations that were placed on them.[1]

Data mesh is a fairly new approach to data architecture that decentralizes ownership of domain data while applying product thinking to analytics data. It is mainly a solution to problems with organizational scaling. The reason why data mesh is not focusing on

1 NewVantage Partners, *Big Data and AI Execution Survey 2021: The Journey to Becoming Data-Driven—A Progress Report on the State of Corporate Data Initiatives*, 2021, https://oreil.ly/Go2tX.

addressing problems with technical scaling is that many of those challenges have already been addressed by the technological innovations of the past years, most notably by public cloud platforms and by parallel computation frameworks.

The following two chapters sketch the status quo of data architecture and introduce the main concepts of the new data mesh paradigm.

CHAPTER 1
Pain Points of Centralized Data Responsibility

In the first decade of this century, there was a strong trend toward centralization of enterprise data: all the transactional data of a company needed for its IT operations was consolidated in one central monolithic database in order to have a *single source of truth*. In the operational data field, architects started to move away from the paradigm of having one central place of truth for transactional data. Since the rise of NoSQL and microservices, it has been a standard approach for individual services to maintain their own data store.

In the analytics data field, however, the centralization paradigm is still prevalent, in terms of both using centralized storage systems and maintaining central *data teams* that centralize data expertise. Analytical data is often used by overarching functions such as finance, management, or marketing, who need a holistic view of the entire business. The industry went through several iterations of analytical data architectures that innovated on how the data is physically stored and how it is processed. Yet, seemingly because of this need for a holistic view of data, no one dared to touch the general paradigm of central analytics data ownership as the gold standard to gain reliable insights about the business. In the next sections, we look at the currently most used analytics data approaches, their differences, and their commonalities.

The Data Warehouse Approach

The data warehouse approach is the oldest of these iterations and still successfully applies in many places. The approach, in its simplest form, is to collect data from different sources to then transform it so it can be combined in a common format before it is loaded into the warehouse. Furthermore, before making the data available to consumers, it is usually cleansed and quality-checked to a certain extent. The idea here is that consumers can rely on the data in the warehouse as the single source of truth about what happened at what time, with appropriate levels of data quality to engender trust.

One of the issues with this approach is that it does not scale well to a growing number of diverse data sources: the more data sources need to be integrated, the higher the likelihood of contradictions between the different sources. Therefore, it becomes harder and harder for the people maintaining the warehouse to curate the data to a consistent state. Also, the effort for central cleansing and quality control increases with a growing amount of data. The latter is particularly hard for a central team to do efficiently because they often lack the domain knowledge that the people who maintain the data-generating systems have. This leads to the curation becoming slower and slower until consumers get frustrated because the data they get served is not up to date. It is basically impossible to fulfill the promise of one perfect-quality, contradiction-free source of truth when dealing with a dynamic system (e.g., when you want to allow new data to be loaded into the warehouse and need to integrate new data sources frequently). This is a major fallacy often seen with large data warehouse projects.

The Data Lake Approach

The data lake approach, which appeared about a decade ago,[1] addresses the curation bottleneck the data warehouse approach commonly exhibits with a growing number of diverse data sources. A data lake in its simplest form is a central data store with general accessibility into which data in almost any form or quantity can be put more or less as is. The curation, transformation, and schematization happens, if needed, at the time of consumption. The term

1 Martin Fowler, "DataLake," MartinFowler.com, February 5, 2015, https://oreil.ly/xlhau.

schema-on-read, which is often used in this context, illustrates that data in the data lake has no fixed schema but is often stored in its raw form and is only interpreted or transformed into a schema when reading the data. This process usually has to be done by the people or the applications that consume data from the lake. Figure 1-1 shows how the ingestion from different sources into the lake happens as is, whereas the processing (e.g., cleansing and schematization) happens when users read data from the lake.

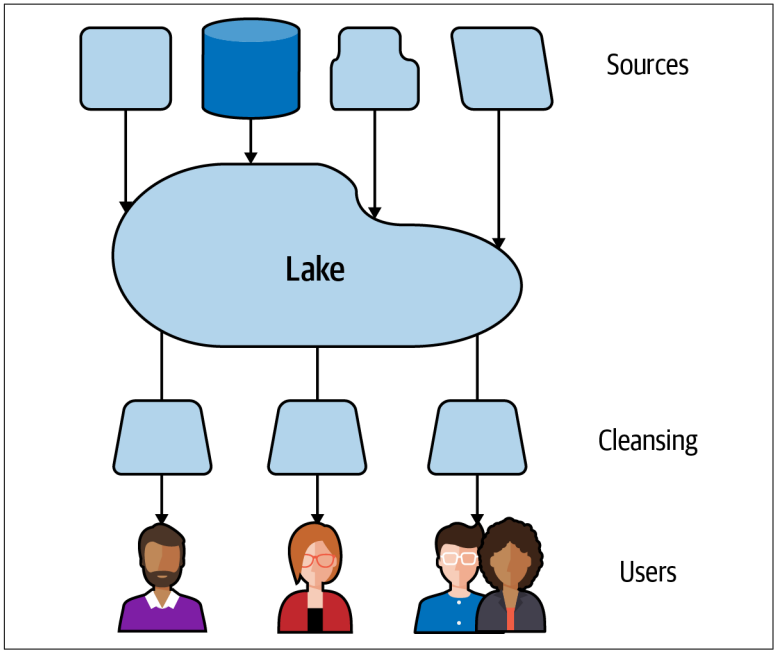

Figure 1-1. The data lake approach

This removes the bottleneck at ingestion time and therefore, at least initially, scales much better for a large number of diverse data sources. The work of curation and especially of cleansing is still needed, though, in order to allow for meaningful data applications. In general, this approach can also scale better on the consumption side, because individual data consumers or stakeholders can invest exactly the amount of effort for cleansing and curation of a particular dataset that they need for their data consumption scenario, and multiple data consumers can do so simultaneously. Of course, this sometimes leads to duplication of data and redundant data-processing efforts.

However, quality control in particular tends to get less efficient the further you go from the source where the data is generated for mainly two reasons. First, there is less detailed knowledge about the specifics of the data-generating systems to understand the issue at hand. Second, there are more intermediate transformation steps between the dataset to be corrected and the original source, so that issues need to be tracked and corrected through various layers of transformation. In the worst case, the data lake creates distance and anonymity between data producers and data consumers.

Therefore, the pain points with a data lake setup look different from those with a data warehouse setup. With a data lake, poorly documented data, data quality issues, and unclear data ownership are more commonly observed issues than with slow provisioning of data. It is also often hard to correlate different datasets in a data lake correctly. Data quality issues are then often attributed to the lack of central quality control and are therefore addressed with central rules and checks about what can go into the lake and what quality standards need to be met.

Centralized Data Responsibility

The fact that the data lake approach distributes certain tasks, such as cleansing, to consumers does not mean that it is a departure from the general paradigm of centralized data responsibility, which the data warehouse approach established. For example, there are data lake setups that make extensive use of distributed storage solutions and provide use case–specific *lakeshore marts*,[2] which are maintained by their stakeholders. Nevertheless, there is usually still one or multiple central teams that own the data lake and make the connection between data producers and data consumers. That means that it remains the central team's responsibility to make sure that data, for example, has a certain quality, is delivered in a timely fashion, and is continuously available. But central infrastructure teams are usually detached from both the specific use cases and the details of data generation in the source systems. Even if the members of such a central data team have the motivation to fulfill this responsibility, they usually lack the domain knowledge and the ability to fix issues directly. Instead, they need to urge data-producing teams to

2 See Fowler, "DataLake."

perform this task. You basically separate data producers and data consumers from each other, which usually leads to unnecessary friction, misunderstandings, and often a bad experience.

This way, central data teams—no matter whether they maintain a data warehouse, a data lake, or both—often get into a middleman position and quickly become a bottleneck when the number of data sources proliferates and when, simultaneously, data consumption use cases grow in number and in complexity. In the next chapter, we will reflect on what we can do about the pain points we have discussed so far and, in doing so, will introduce the pillars of the data mesh concept.

CHAPTER 2
The Pillars of Data Mesh

Data mesh is more than another iteration of an analytics data architecture that sticks with the centralization paradigm. Instead, it presents entirely different ways of interaction between data producers and data consumers and therefore should rather be considered a paradigm shift.

Data mesh builds mostly upon concepts and practices that have already successfully been applied in general operational systems architecture but have yet to be widely applied in the analytics data space. Figure 2-1 shows the four main pillars of data mesh, as presented by Zhamak Dehghani,[1] each pillar representing an application of an already established architectural principle to the analytics data domain. The following sections will introduce them one by one.

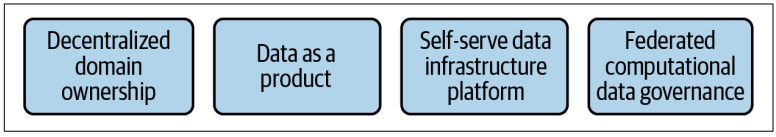

Figure 2-1. The pillars of the data mesh paradigm

1 Zhamak Dehghani, "Data Mesh Principles and Logical Architecture," Martin Fowler.com, December 3, 2020, https://oreil.ly/Dh3g2.

Decentralized Domain Ownership of Data

One key departure from the established data architectures is that data mesh promotes decentralized domain ownership of data. The keyword is "ownership" here. With ownership, we mean full end-to-end responsibility. In order for this decentralized data ownership to succeed, data mesh applies domain-driven design.[2] A *domain* in this context can comprise different things. Typically, there are business entity domains such as customer, sales, and the like, but there can also be more technical domains (or subdomains) such as click data generated from a web page analytics tool. Basically, every topic that is complex and important enough to justify building up expertise around can be a domain. From an architectural perspective, this means that instead of using systems, technologies, or process stages as the guiding criteria for structuring ownership, business domains or their subdomains should be used to define boundaries of ownership.

To generate the most value from a domain's data, we want to build up domain expertise and then give domain experts both the authority to make the important decisions and the capabilities (i.e., skills and resources) to implement these decisions. Importantly, though, they also need to have the responsibility and capabilities to deal with the consequences of their decisions. This culmination of responsibility and implementation into one domain instead of outsourcing parts of it to centralized entities results in much-improved efficiency because problems are solved at the place where they originate.

The main motivation for this decentralization is to scale better with a growing number and diversity of both data sources and data-consuming applications.

Data as a Product

Decentralization of data ownership has been attempted before, especially in large, distributed data lake setups. Often, however, responsibility is pushed on data-generating teams without much incentive to take over this responsibility. Also, in many cases, those teams are not aware of the impact of, for instance, improving on

2 Eric Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software* (Upper Saddle River, NJ: Addison-Wesley, 2003).

data consistency. To help with that, data mesh applies established principles of *product thinking* to analytical data. That is, data is not treated as a by-product of another product but as a self-contained product of its own.

Let's take sales data as an example. Building a successful sales data product means first of all—as with any other digital or physical product—that you need to understand who your *customers* are. You need to find out who is dependent on sales data, what they do with the data, and what kind of use cases need to be supported. Most of the time this means that you need to talk directly to those people in order to understand their challenges, aspirations, and desires. For instance, marketing might have different requirements than finance when it comes to sales data. Some users might be mostly interested in the timeliness of the data in order to react quickly, whereas others might need the highest possible accuracy but are OK with data that is a few days old. Some use cases will prefer that the data is provided as a continuous stream, whereas others might be focused on batch-processing. The sales data product is then designed from the ground up to cater to the needs of your most valuable users in the best way possible. Furthermore, applying product thinking means that you want to attract more users over time, for example, by marketing your product, and to keep the existing ones happy by responding to their feedback. It is that customer-centricity that characterizes the creation of successful data products.

Of course, you cannot start with a full-featured data product. Similar to building any other digital product, you want to start with a small core feature set to ship a *minimum viable product* (MVP) to your users quickly, so that you can get feedback on whether your product actually meets their needs. From there onward, you want to expand and maintain the data product, which means you have to prioritize which features to build next. Building such a data product calls for a cross-functional team that has all the skills and expertise needed to maintain the product long term and to take full end-to-end responsibility. Furthermore, the team needs to have a product management that defines a road map for that data product, manages requested features, and understands the requirements of the users.

How a data product works internally is then entirely up to the team that develops it. They should have full autonomy to decide whether it is most pragmatic to work with a relational database or whether an intricate machine learning model is needed to provide the

requested features. The same goes for the data product's output ports: data is provided in a way that is most suitable for the users, independent of the data product's underlying data storage technology, and data can be made available to different user groups using different interfaces.

A data product usually provides programmatic interfaces, such as streams, APIs, SQL endpoints, or storage access. It is usually not an application for end users. The reason is that within a data mesh, data products are meant to act as building blocks from which more complex data products can be composed. You could say that in the data mesh approach, a data product is both the manifestation of applying product thinking to data and a data product being the *architectural quantum*,[3] that is, the fundamental building block for structuring your system. Furthermore, a data product is self-contained, that is, it comprises everything that is necessary to enable its users' use cases (e.g., documentation, metadata, interface definitions, and access management).

Generally, we distinguish between two kinds of data products: *consumer-aligned* data products and *source-aligned* data products. The previous sales data product example is a typical consumer-aligned data product. It provides a lot of its value by aggregating data from different sources, dealing with potential contradictions between the sources, and finally providing the data in the form and shape that is most useful for its users. A consumer-aligned data product usually focuses on fulfilling the needs of a specific user group. An example of a source-aligned data product is a *checkout data product* that accompanies a respective checkout service and that provides parts of the original data that sales data is ultimately computed from: in this case, checkout events. Such source-aligned data products should be developed by a team that is as close as possible to where the data is originally generated. Their job is to provide checkout data in such a way that users—and other data products—can consume and make sense of the data without needing intricate knowledge about how this data is generated.

Applying product thinking to both source- and consumer-aligned data products also means that a team can obtain resources and

3 Neal Ford, Rebecca Parsons, and Patrick Kua, *Building Evolutionary Architectures* (O'Reilly Media, 2017).

management support based on the success of their data product. For instance, a data product's usage metrics and its business impact can be used to drive resources, budget, and product focus, in the same way the success of any digital product for external customers would be measured.

Self-Serve Data Infrastructure as a Platform

Enabling decentralized domain teams to take over full ownership of data products is not an easy task. At least at the time of this writing, capable data engineers are highly sought out in the job market and therefore are difficult to hire. In most cases, it is not feasible to have a dedicated data engineer in every domain team. And even if this was possible, having to deal with intricate data infrastructure details will create too much cognitive load for a team that should focus on building up domain expertise instead. This is one reason why data mesh applies the idea of infrastructure-as-a-platform toward data to create a self-serve data infrastructure.

Providing IT infrastructure as a self-serve platform is something that cloud providers have been doing successfully for several years now. Applied to data infrastructure, a platform needs to provide self-serve tools that allow domain teams without dedicated data engineering capabilities to autonomously create, develop, and maintain a data product. This also includes tools to manage tasks such as compliance, documentation, and encryption. Please note that such requirements are independent of the infrastructure platform being cloud-based or running on premises.

Furthermore, there are a couple of infrastructure pieces that need to be provided centrally for data products to interoperate or even to be discovered in the first place. A typical example of a centrally provided piece of self-serve infrastructure is a data catalog that data product teams can use to register themselves, provide metadata and documentation, or discover other data products that they can build upon. In general, the data infrastructure platform has to provide self-serve tools for two audiences: tools for data producers (e.g., for documentation or access control); and tools for data consumers (e.g., for data discovery or programmatic consumption). We will provide more examples of infrastructure capabilities in the section "Data Infrastructure Capabilities" on page 35.

It is important here to understand that decentralized data ownership does not necessarily imply decentralization of data infrastructure. Central data infrastructure is often a useful and cost-efficient option as long as it provides logical separation between data products. This does not take away any autonomy from data product teams as long as they can independently create, manage, develop, and destroy their data products using central infrastructure.

While a data infrastructure platform should provide the best possible support to data product teams, it is crucial that the platform stays entirely domain-agnostic. One of the main motivations for data mesh is to provide a scalable model that does not exhibit the central bottleneck that data warehouse and data lake setups show on a regular basis. As soon as a central data infrastructure platform provides domain-specific tool support, it ties itself to that domain and therefore potentially becomes a bottleneck for the independent development of the respective data products.

Federated Computational Data Governance

The fourth pillar of a successful data mesh is to establish a federated computational data governance. Data governance is often seen as a function that needs to be handled centrally—and often goes very much wrong when handled with a centralized mindset. By centralized data governance, we mean the kind of governance where rules are created by a centralized team that is detached from these rules being applied on the ground. Such a centralized governance team then has to enforce and regularly check on the adherence to their rules. This often leads to them feeling like—and being perceived as—a *data governance police*. Often, for a lack of positive incentivization, this approach does not work in the long run. As with centralized data ownership, this approach also does not scale well. In the face of a growing number of diverse data sources and data use cases, centralized data governance becomes a bottleneck that slows things down.

Data mesh therefore approaches data governance differently. We acknowledge that there is a need for some *global* data governance rules, for example, because of the severe legal implications when not meeting data protection regulations. However, there are three adaptations to the rigid police-like governance described before. First, the group defining global governance rules should be *federated*. This

means it should consist of representatives from the different domains plus a few experts who, for example, inform the group about legal issues. Federation also makes sure that the governance group focuses on topics that the teams in the domains have an interest in being decided on globally. Second, global governance should be as *thin* as possible and focus on enabling interoperability so that a data mesh can exhibit its desired network effect. Governance issues that can be decided and handled within a domain should stay within this domain and therefore be dealt with in a decentralized fashion. The need for global governance often shows at the intersections between domains, where it should facilitate the necessary alignment. Third, federated data governance should be *computational*. This means that checking for the adherence to global rules should be automated in as many cases as possible. Ideally, global governance should only decide *what* needs to be taken care of. Decentralized data product teams should then decide locally *how* something is taken care of. Finally, the self-serve data infrastructure platform should provide the tools to make sure *that* it is taken care of in a reliable, auditable, and automated way. Another way a self-serve data infrastructure platform can foster adherence to global governance rules is by providing tools that, when used, automatically implement adherence to those rules. If those tools are attractive and easy to use, they can make it easier to adhere to global rules than not to. This is why we sometimes refer to this as "compliance by convenience," which we will provide concrete examples of in Chapter 5. Overall, the federated governance group will be a decision-making body and not a central function that is kept busy with checking adherence to global rules.

This rounds out our introduction of the pillars of data mesh and, with that, the first part of our report. Building on the data mesh principles outlined so far, the second part of this report focuses on how to apply those principles and illustrates the different stages of data mesh adoption, from the very first data product to scaling and sustaining the mesh.

PART II

The Data Mesh Journey

Understanding ideas of decentralized data ownership sets the foundation for an organization to become truly data-driven. It is as important, however, to understand that performing a change for a bigger, brighter future does not happen in one day. Adapting the ideas of the data mesh paradigm at company scale is a journey that can take months, even years, and can look overwhelming at first. In this part of the report, we want to give you a set of tools to get started and guide you along the journey. As shown in Figure II-1, the data mesh journey is structured in three parts, each of which focuses on what we think is a particularly important aspect of the journey.

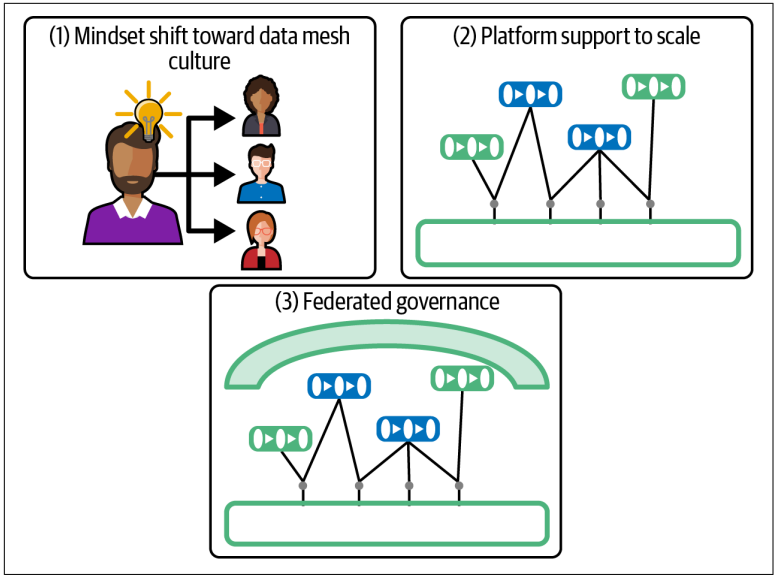

Figure II-1. A data mesh journey in three steps

First, we look at how creating an initial mindset shift is key for getting started with the transformation toward a data mesh culture. Second, we explore how powerful self-serve data infrastructure will help you scale the mesh from a few initial teams to large parts of your company. Third, we illustrate how federated governance is applied to sustain the mesh and ensure successful long-term value creation. Once we have concluded our journey, we will take a look at additional examples of common pitfalls and best practices that start to establish across the industry.

CHAPTER 3

Getting Started: A Data Product–Centered Mindset Shift

One of the biggest misconceptions when working toward a data-driven organization is that change can be forced by technology. When people hear about a new concept that is supposed to change all their data-related problems for the better, they look for the shortcut, the easy way out, the tool to apply that solves all their challenges. By now, most data practitioners understand that even a change of technology is often a mid- to long-term project, especially when it requires migration of hundreds to thousands of internal data use cases. Therefore, a fundamental change of direction cannot succeed with hurried short-term plans but needs a bigger shift of mind instead.

In Chapter 2, we introduced the idea of data products and outlined the advantages of having domain experts taking the ownership of data they are offering to others. To apply these ideas in practice, we have to go one step back and start with what is often the status quo. Let's reflect on how data is commonly shared and processed across many organizations.

The Rise of Big Data Technologies

We often divide data between two different purposes, transactional and analytical. Transactional data is anything that supports the day-to-day business of your company. It can range from comparatively raw data—like an order being placed in a shop—to complex

messages that are exchanged for communication between two services. Analytical data is changing the focus toward decision making. It is extending the scope and the time horizon of relevant data. The main purpose is no longer day-to-day business transactions but moving toward understanding the bigger scheme of things. Different approaches for supporting such decision-making processes can range from classical business intelligence reporting for understanding historical events to machine learning models for predicting future behaviors.

Analytical data processing has changed significantly over the last decade, both in terms of data-processing paradigms and technology to support those changes. Only a decade ago, the state-of-the-art approach was to operate a central data warehouse that directly integrates with various backend services to fetch, process, and integrate data for analytical purposes through classic Extract, Transform, Load (ETL) processes.[1] The introduction of cloud computing—and, with that, large-scale cloud object stores—introduced a comparably cheap option to store amounts of data orders of magnitude higher than the capacity of a traditional data warehouse. At the same time, the enhancement of parallel processing engines allowed us to scale the necessary processing power to still be able to handle these large amounts of data. Many companies jumped onto the hype train and started setting up data lakes that arguably were the big successor of the previous data warehousing paradigm. In many cases, those were built on top of or in addition to existing on-premises systems.

Unfortunately, we forgot one tiny detail. Nobody knew how to work with all the newly emerging technology choices right away. Over several years, companies were driven by the question "What *can* we store?" instead of "What *should* we store?" Default archiving of data became the de facto standard compared to conscious decision making about what data to store for analytical purposes. The result of this is what we observe in many organizations today: large amounts of poorly documented data with unclear data ownership. We will now take a look at some of the pain points resulting from this setup and how they can lead to detrimental impact at organizational scale.

1 Panos Vassiliadis and Alkis Simitsis, "Extraction, Transformation, and Loading," The Department of Computer Engineering and Informatics, Ioannina, Greece, 2009, https://oreil.ly/8EoyN.

Case Study: Pain Points of Unclear Data Ownership

> I started in a new team as an analyst responsible for expanding on the service-level key performance indicators (KPIs) for the services of my team. Previously, all KPIs were derived from the data the team produced, but for the first time, we needed to get additional information from a different service. I got pointed toward a team supposedly responsible for this data and reached out to them. It took one week to get a reply from the team that they were not responsible for this service and that they did not know whom to forward me to. Talking to the BI team of our area, I learned about a central data lake team that was responsible for archiving all the data of the company. In a discussion with them, we found the dataset I was looking for and, after checking for the source of that data, we identified that the owning team was in fact the first one I had been talking to. When I reached out to them again, they apologized for the previous misinformation and told me they indeed still had an old service running that was producing this data, but the person working on that service had left the company and, as a result, the knowledge got lost. After talking again to the central data lake team, I was finally able to get access to the data I needed. In total, I spent 4 weeks of manual effort to organize a task that eventually took 10 minutes to be resolved.

The previous example is not a real individual story but a combination of several real stories from different companies. It outlines one of the core problems of unclear data ownership: missing awareness and lacking responsibility. Every data journey starts with discovery, and every so often it is this entry point where data practitioners already get stuck. Not knowing whom to approach, relying on personal networks instead of company-wide capabilities, and, in cases like the one just cited, getting rejected even when finding the right contact are unpleasantly close to reality. It is apparent how much time is being wasted in such situations and how much potential for improvement there is. However, lacking data ownership can become even more devastating further down the road, as shown in the following continuation of our example:

> After integrating the required data I now had access to, I was able to set up our extended KPI reporting. One day, after two months of analyzing the new KPIs, I discovered a drop in numbers. I reached out to the central data lake team, who confirmed after an initial investigation that their pipeline did not have any disruptions and that, for the dataset of interest, new data was still available. Digging

> deeper, they found out that the shape of the data changed and that one of the fields I was relying on did not have data any longer. With that knowledge, I reached out to the owning team again, who told me that, after my discovery of two months earlier, they decided to clean up the service and rediscover the lost knowledge for its operation. The updated version of the service was put into production just the day before and included some improvements, like a replacement for the field I was relying on with a different version. The team was not aware of our new dependency on the service and did not inform us up front about that change.

In the preceding example, the team in question wanted to improve the operational situation and took back ownership for their service and for the direct dependencies on it. However, the owning team did not gain awareness about the dependencies on the dataset they were producing or take ownership of the data that another team was relying on. This is only one of many examples of breaking changes in production datasets that were discovered by its consumers after the fact. Data is often produced as a by-product of service operations and is rarely the focus point of conscious decision making in data-producing teams. Changes for data frequently keep transactional use cases and dependencies in mind, yet rarely consider the consequences for analytical users that rely on the same data that is archived for historical availability. But how can we change this status quo? How can we move toward data sharing of well-defined datasets with an owner that has a business interest in the well-being of its dependencies and stakeholders?

Moving Toward Decentralized Data Products

In an ideal world, we would like to change our organization such that data that is shared for analytical purposes has clear ownership within its domain. Getting there might be a long journey, however, especially when reality is much closer to the examples we introduced before. How can we start to incrementally improve on that reality? What is the first step to get going?

Understanding the Different Perspectives

Trying to improve a situation where multiple parties are involved always starts with understanding all the different perspectives. In the earlier scenario, where the analyst tried to set up KPIs for their team, a fully understandable human reaction would be anger at the

data-producing team that was indirectly responsible for all the extra work they had to go through. How often have we observed others around us, or even caught ourselves, steaming off about "that other team that is always doing everything wrong"? Thinking about this situation from some distance, however, we get the chance to ask ourselves, What do we actually accomplish by thinking and acting that way? By talking bad about the team whose data we are using, we will only alienate and potentially even offend them. There will be a continuous emotional tension between our teams that will make any future collaboration incredibly difficult. This becomes even more important considering that we depend on the data produced by "that other team."

If we try to understand the perspective of that team, we might learn an important lesson. We will understand that this team also has a packed road map they are trying to deliver on and that what happened was not out of malintent toward us but because they were as overloaded as we were. In fact, the second incident that broke the production data shows that they were more than willing to improve the situation. They might simply not have known all the points to consider while doing so.

Turning your perspective toward the data producer, it is important to understand the situation of those who depend on the services and data you are providing. Having awareness about your downstream dependencies allows you to broaden the perspective on how your services and data can be used. At the same time, it allows you to prevent incidents caused by breaking changes through early communication. It can even help you to prioritize and decide on new features by understanding the value those would add for your users.

Creating Incentives

Awareness for your dependencies and empathy with the people behind those are the first steps for creating an environment that allows for fruitful collaborations and cross-team success stories. But how can you create the right incentives for the people you depend on to guarantee your paths are aligned? Creating incentives can be grouped into two main categories, as displayed in Table 3-1: social incentivization through appreciation and awareness, and material incentivization.

Table 3-1. Types of incentivization

Social incentivization	Material incentivization
Recognition and appreciation after a job well done	Team-level budget grants (e.g., for new roles)
Awareness of value created by one's contribution	Ambition for success and career progression
Satisfaction of helping others solve their challenges	Direct bonuses for individuals

Social incentivization builds on the human desire to be recognized for the efforts we have invested. When we receive appreciation after a job well done, we get a strong feeling that the time spent on the task was worth it. This effect amplifies when such expression of gratitude is happening in an environment where more people can see what happened: the next time you announce a new feature to your group of stakeholders, consider mentioning that upstream dependency. Even if there were challenges on the way, like in our cross-team KPI example from earlier, you would not have been able to complete your work if the team you depend on had not provided the data in the first place. The awareness that there is another team that is creating value based on their contributions can be a strong driver in itself and provide an intrinsic satisfaction of helping others to solve their challenges. Mentioning them in your announcement will show that the effort they are putting in to improve the situation is perceived very positively and incentivizes them to double down on it.

An example of material incentivization is the increase of a team's budget based on their impact. Let's assume our team runs a machine learning model that allows us to improve the sales predictions of our company. This model depends on the availability of yesterday's sales data. Based on a data latency analysis, we identified that having the source data available two hours earlier every day would gain us a profit of $1 million per year because of increased sales due to better predictions. It would require a significant engineering effort by the source team to implement this improvement, and currently they do not have the resources to take care of that. Now that we understand the value this improvement would bring to the company, it would be easy to pitch for an additional engineer for that source team so they can take the responsibility for this data product.

Where and How to Get Started

Now that we have explained the foundations for the necessary mindset shift of moving to a data mesh architecture, we want to tackle the question that we get most often: "Where do I start with all of this, and how do I do it?"

At this point, it is necessary to repeat that moving toward a data mesh is not so much a question of technology that can be solved by centrally rolling out a new set of tools for your organization but is first and foremost a change of mindset on how to work with data on an organizational scale. To change the way people think about working with data takes time. The best advice to begin an organizational change in the data space is to start small.

Trying out new approaches and winning over people to do the same is easiest done in a setup with limited complexity and only a few people involved that can work in an iterative "fail fast"[2] approach. Often in an organization, there are already teams that are more progressive in their ways of working. These are the perfect candidates to start a seed of change by building a textbook example of a first data product that can later be referenced across the company. Find a responsible person in that team to play the role of the data product manager. Maybe the team already has a product manager who is data-savvy. They will be responsible for finding and communicating with the stakeholders, understanding their needs, and defining exactly what the desired data product is supposed to look like. What guarantees should be given, for example, for data latency and quality control?

This initial team building a first data product should ideally be paired with a specific team of data consumers as well (e.g., a team starting to build a new data-consuming application based on the first data product). This way, both the first data product itself and the new kind of interaction between data producers and data consumers can be showcased. This helps to build up the product mindset that is so important for the data mesh paradigm shift. Both teams should focus on building MVPs (e.g., within three months) and working tightly integrated during that time. It is important to be

2 Jim Shore, "Fail Fast," *IEEE Software*, September/October 2004, https://oreil.ly/N0wHe.

very clear which minimum requirements have to be fulfilled to talk about a "data product." This is something each company has to define for itself.

Creating New Infrastructure Alongside the First Data Product

As the first of two cases, we will assume that the very first data product we are introducing is built as a greenfield project and cannot rely on any preexisting data infrastructure platform. We are well aware that in most companies there is already some sort of data infrastructure even before you start moving toward a product-focused approach of working with data. In the next section, we will explore the case of how to evolve an already existing platform to serve such cases.

In the case of a fresh start in isolation, the first data product team will also have to understand what kind of data infrastructure needs they have and start by building out and operating the infrastructure needed for their specific use case. We do not advise building the platform for data infrastructure before the first data product, because this comes with a high risk of overengineering the platform or of building it in a way that is not the best fit for the requirements of the first data products. Instead, we suggest building the initial data infrastructure alongside the first data product and iterating and improving over both the data product and the infrastructure in an MVP together. This will allow you to harden the understanding of the basic concepts, but it will also allow you to understand what should be tweaked to the needs of your specific setup in your specific company.

As you can see in Figure 3-1, you will gradually build out more data products based on the learnings of this initial project. At the same time, the infrastructure components that are built up as part of the first data product can then be extracted to become the first capabilities of the self-serve data infrastructure platform once you realize that further data products start to have repetitive needs. We will talk about this in more detail in the next chapter.

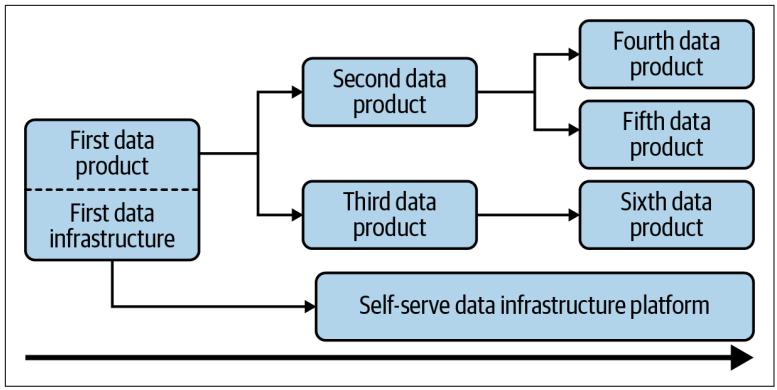

Figure 3-1. Starting small and scaling out

The strategy outlined so far will allow you to create a positive example of a value-generating production setup—not only a proof of concept—within your organization. Being able to show a positive example will make it much easier to convince additional teams to follow similar approaches and will also give your senior management the confidence to continue their support.

Adapting Existing Infrastructure

In contrast to starting your first data product as a complete greenfield project, most companies will need to work with existing infrastructure. This usually involves data warehouses or data lakes that already exist, including central infrastructure teams that take care of them. In the upcoming chapter, we will be discussing the troubles of those infrastructure teams in more detail and how to address them, but for now we would like to gain an understanding of how the first data product teams could benefit from the already existing infrastructure.

The unsatisfying but honest answer is it highly depends. Very similar to our greenfield example, our first data product team needs to understand what exactly they need in terms of infrastructure capabilities to support their use case. Here, it is important to not be blinded by the already existing capabilities alone but to allow for a fair amount of creative thinking and discovery of new possibilities. Of course, being able to reuse existing infrastructure capabilities can highly benefit the speed of our project, yet it should not be the reason to commit to suboptimal solutions. Ultimately, we want to set the foundation for an innovative and sustainable future of our

company, so we should not discourage innovation from the very start. Reality, of course, is not always black and white, and there are also effective solutions for integrating existing infrastructure into a new setup. An example is the introduction of a virtualization layer that allows for accessibility of data across various source technologies.

Once we understand the infrastructure capabilities that we need, we should check again for the existing capabilities that are already provided. If we find a match between supply and demand—amazing—let's partner up with the infrastructure team and establish a strong relationship between them and our newly introduced data product team. In a case where the infrastructure needs for our project are not yet covered by existing capabilities, we are in a very similar situation to our previous greenfield example. The very first data product team will have to take care of the needed infrastructure components by themselves but also has to keep in mind that at a later stage such needs will likely be extracted into platform features. The biggest difference between our two cases is that once we reach the point that we build out our self-serve data infrastructure platform based on the common patterns across the first couple data products, it is likely that the already existing infrastructure team will get involved in the platform development, so it will again be helpful to have a frequent exchange with them early on.

At this point in your journey, you will have arrived at having a low number of additional cross-functional domain teams working toward a data product approach, and slowly it will become apparent what infrastructure needs are specific for each team versus what patterns are repeating across multiple teams. That is when you invest in a platform for data infrastructure. That is when you start scaling the mesh.

CHAPTER 4

Scaling the Mesh: Self-Serve Data Infrastructure

Platform capabilities—provided through a central infrastructure team—enable many organizations to become more efficient and more scalable as a whole. Mature organizations often start their journey already having some form of platform infrastructure teams in place, whether this be teams responsible for existing data infrastructure around data warehouses or data lakes, or more generic infrastructure teams around cloud resource management. Often those teams already have a strong understanding about the pain points around central responsibility and can be strong partners for collaboration. It is important to understand, however, that changing toward a style of data-agnostic, self-serve data infrastructure usually requires bigger changes that again take time and resources. Do not expect your existing central infrastructure team to be the cure for all evil and enable the data mesh all by themselves while still taking care of their current responsibilities. In order to understand how infrastructure teams overloaded with central responsibility can escape their vicious circle, we first need to visualize the pain points of central infrastructure responsibility.

Pain Points of Central Data Responsibility

To fully grasp how demanding it can be to take on central responsibility for data and processes that require distributed domain knowledge, let us look at an example that is well known across many organizations at scale: granting access to data in a central data lake:

> When building up a data lake in our company, we started by providing a central data pipeline for data ingestion, a distributed storage layer, and a distributed processing engine on top of it. After an initial testing phase, we started having real production data flowing in and we quickly realized the need to restrict the access for the sake of not sharing confidential data all across the company. We set up a process where anybody that needed access to data would file a request stating what data they needed access to, who it should be granted to, and what the reason was for the request. We would then check those requests for approval and execute the technical granting process if the request was legit. For the first couple months, this process was working quite fine. We had a good grasp of the data we were storing, and we understood the use case descriptions that were provided to request access. Yet slowly but surely the situation started to get out of hand. The number of requests started to increase, and it became more and more tedious to manually follow up on all of them. On top of this, the number of different datasets we stored was exploding, making it incredibly hard for us to understand all of them and judge if a certain request even made sense to begin with. It took until a year after the process was introduced for us to realize that our central infrastructure team became completely overwhelmed and paralyzed by a small process we introduced early as an easy fix to a problem at hand.

As expressed in the preceding example, it is common for teams responsible for central data-processing systems to be involved in the access-granting process as the last entity that executes the technical granting of access to requestors. In our case, however, the responsibility for the process as a whole was taken on by the central team. This included receiving all the requests, with all its details, from any aspiring data user in the company: what data the requester needs access to, who is part of the request, and what the use case is. At first glance, this sounds like a reasonable process, but it has some ingrained flaws:

Decision making requires domain knowledge.
: Judging if a provided use case description can suffice for an access approval requires an understanding of the use case itself,

the requested data, and the security requirements around data sensitivity. This might work fine in a smaller company where a single team can still have an overview of all existing data use cases. With a growing number of datasets and use cases, this ultimately leads to one of two outcomes. Either the process becomes a bottleneck for the whole company, or the checking becomes superficial and no longer fulfills its original purpose.

A manual centralized process does not scale.
: Consider an ever-growing number of datasets, as well as an ever-growing number of users that want to access those. You quickly reach the point where parts of the central team will be doing nothing but answering access requests. Unless the company is willing to invest serious resources in maintaining a team with the sole responsibility to take care of such manual labor, one needs to rethink how to change the process itself to better scale for a data-driven organization.

This is only one of many examples that spark from central data teams taking on central responsibility that either requires distributed domain knowledge or heavy manual effort. The pattern across such cases is often very similar. It starts with a simple process that is set up quickly in small-scale environments but is eventually outgrown by an increasing number of data-driven use cases. Rather sooner than later you should take a step back and check if your setup still makes sense. What are the cases that still work perfectly fine? What are the cases that run the risk of being outscaled? What are the cases that already long breached that point? And, lastly, what can you do about it?

To understand how we can address these types of challenges, we will take a look at another example: the provisioning of compute resources for data processing.

Case Study: Centralized Compute Capabilities

Across many companies, the provisioning of compute resources for data processing is among the first capabilities to be put into the hands of central data infrastructure teams. Frequently, companies come from a setup where across multiple data teams a common story repeats over and over again until consciously disrupted. Let's take a look at the following example:

> I was hired as a data scientist to work on a new machine learning product for my new team. Upon joining the team, I was quickly introduced to our use case and to the data source we wanted to use to train a new model on. I worked with the data on my local machine and developed an algorithm that was fitting the given criteria quite well. I quickly realized, though, that productionizing my model was much more complicated than I originally thought. The full amount of data that needed to be processed to train our model was much bigger than what would fit on my local machine. While the team had some experience in working with the cloud, nobody ever had worked with a distributed data-processing engine before. It took us six weeks to set up a computation cluster and train the first model on the full production data. It took several hours to complete the run, but at least we made it work. Two weeks later, I was introduced to two fellow data scientists from different teams in our company and painfully realized that they went through the exact same journey when they joined.

When joining the team, instead of being able to fully focus on the problem domain, the person in the example had to spend the majority of their time setting up a computation cluster to be able to process the data they needed. Not only did this distract them from the job they were actually hired for, but it was also fully outside their area of expertise. It required a big learning curve to complete the task and, considering their summarizing statement, they completed the effort with mediocre results at best. Even worse, across their company, which seemed to be continuously looking for new use cases, multiple colleagues were following the same pattern and were reinventing the wheel.

Now that we discovered this repetitive scenario, how could we address the situation? The obvious choice is to instead put the compute infrastructure responsibility into the hands of a central data platform team. But there are many options on how to do so, and some options can lead you directly back into the traps of central infrastructure responsibility that were described in the previous section. To illustrate this point further, we will follow two scenarios and highlight the most important differences between the two.

In the first scenario, a central team is taking full responsibility for operating and maintaining the compute infrastructure, which is now offered to all decentralized data teams. They offer APIs for submitting jobs and allow teams to specify input and output locations for the data they are working with. This abstraction not only allows the decentralized teams to not care about the infrastructure itself

anymore and leave it to the experts but even allows for resource optimization by running multitenant clusters that operate multiple jobs in parallel.

While the intentions behind the implemented model are pretty clear, it has some inherent flaws that lay underneath the surface. First, you need to develop a sophisticated data access model that understands who is allowed to see which data. Maintaining multitenant environments, even more so when abstracted away from the user, requires clearly defined rules and enforcement for data access. How do you guarantee that a team trying to access data that they should not have access to will get denied that access, while another team running on the same physical infrastructure is allowed to have that access? Second, while everything might work fine in the happy path, what happens if a job fails? How are job logs shared with the user—again, in a way that only they can see the logs for themselves? Are they even trained in how to read such logs? Does it require a central infrastructure engineer to babysit every team for every use case that is in development and again become the bottleneck? And for the worst-case scenario, what if a job failure is in fact caused by a cluster multitenancy issue, where one job influenced the execution of another?

The line of abstraction in the second scenario is drawn a bit lower than in the first. Again, you have clusters centrally operated and maintained by a data platform infrastructure team. However, instead of offering an API for job submission, you offer a template or an API for requesting a cluster. It contains all details necessary to be specified: What data do I need access to? Who is allowed to use the cluster? What should be the scale of my cluster? Do I need additional libraries to get the job done? Once the request is issued, the cluster creation is fully automated, and the requester will be provided the necessary information on how to use it. From that point on, they again have to use API calls to submit jobs to that specific cluster, but some key usability points change:

- The cluster is running in isolation: no "noisy neighbor" can interrupt your work.
- I am provided full access to the logs of my cluster—I am fully enabled to analyze any job execution errors by myself.
- Data access is clearly defined on a cluster level and can be controlled and monitored on that entity.

As expressed in Figure 4-1, moving away from a fully decentralized responsibility of everyone needing to operate their own infrastructure does not necessarily mean full centralization. Giving all domain teams strong dependencies on the central infrastructure team would immediately make them the bottleneck. Rather, you need to strike a balance where the central team is offering a capability that takes away the biggest pain point (going from operating my own infrastructure to filling a template) yet still enabling the domain teams to take full ownership for the use cases they are working on. Yes, it requires more knowledge and responsibility on their side compared to the "all-central service" scenario, but in the end it is important to hit the trade-off between offloading decentralized teams and overloading central teams.

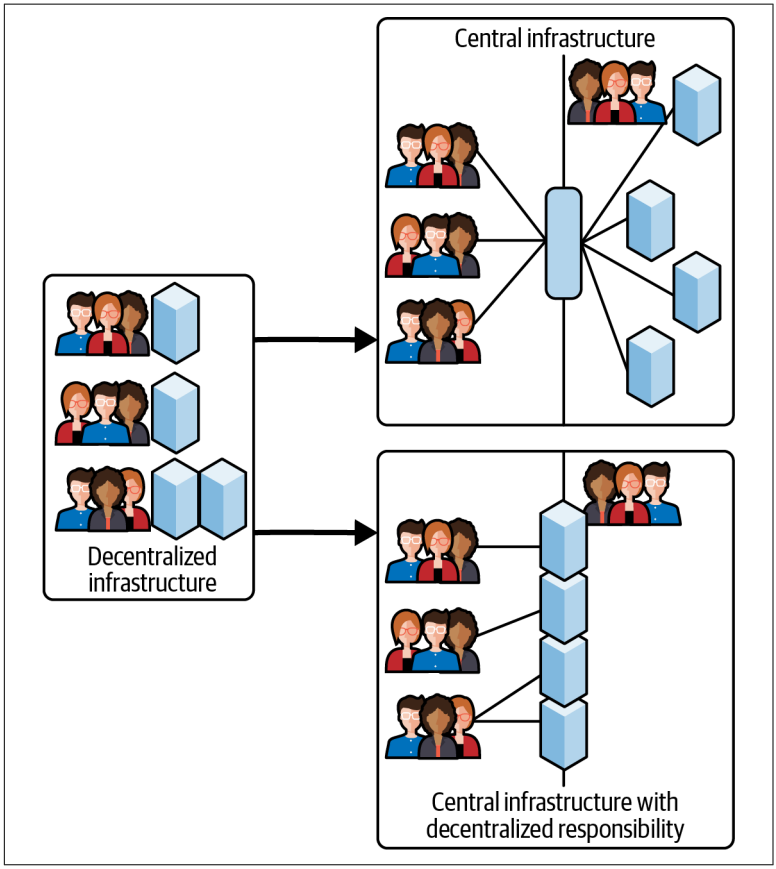

Figure 4-1. Decentralized versus central infrastructure

Data Infrastructure Capabilities

The introduced scenario of data compute infrastructure is only one example of a data platform capability. In a real company setup, such capabilities can evolve in many different directions, as expressed in Figure 4-2.

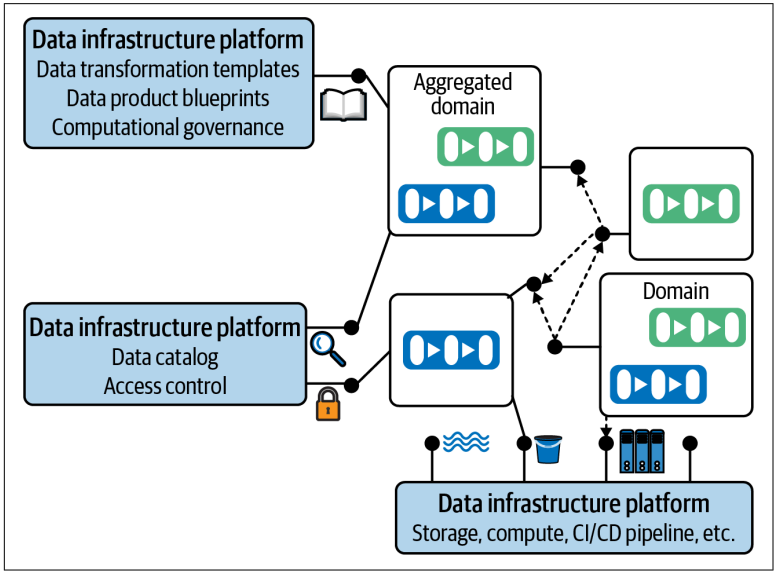

Figure 4-2. Data infrastructure platform capabilities

The foundation of the platform focuses on raw infrastructure topics like offering a company-wide event bus for data communication, a general data storage layer, a CI/CD pipeline, or the aforementioned compute infrastructure use case. It expands toward capabilities that allow defining processes around data, like data documentation and discoverability through a data catalog, or data access management. Finally, we are targeting advanced cases focusing on data practitioners' productivity, like providing templates for the execution of data preparation use cases (e.g., format changes, cleaning of data), entire data product blueprints, or platform support for computational governance. Overall, the data infrastructure platform should drive—and should itself be driven by—open standards that allow for interoperability between data products and infrastructure tooling.

By now we should have many teams that have built successful and well-maintained data products. In addition to that, we have increased the efficiency and productivity of our data teams across the organization by providing them with a self-serve data infrastructure platform. We are in a strong position for working with data in our organization, and there is but one question that we still need to answer: how can we make our efforts last and set ourselves up for a successful path forward? How can we sustain the mesh?

CHAPTER 5

Sustaining the Mesh: Federated Computational Data Governance

Building a data mesh at company scale addresses several different angles of working with data. We have already covered what it takes locally to start building a data product. We also introduced how it is possible to support those local data product builders through infrastructure platform capabilities to ease their journey toward high-quality data products. What we have not addressed yet is how can we make sure various data products are not starting to drift apart? How can we prevent different domains from becoming isolated silos of information?

Ensure Interoperability Through Semantic Cross-Domain Modeling

> We wanted to build an integrated viewpoint between the sales data of our company and the behavioral data of our customers that was collected along their journey on our platform. Both areas had high-quality data products that were well described, easy to find, provided strong guarantees, and had contact product people to work with and discuss those products' usages. Unfortunately, both data products were residing in different source systems, and we heavily underestimated the integration effort between them. After one month of integration effort for each of those two systems into our usual analytics platform, we had to realize that those well-defined data products were not compatible at all. The identifiers used in each product not only had different data formats but also followed

> different semantics. While in the sales data a user was identified through a user identification number, the behavioral data was bound to session UUIDs that were not strictly correlated to the user itself. It took another three months of effort in collaboration with both teams to build the required mapping to support our use case.

The preceding example shows how much of an impact misalignment of data products can have. While both described data products were built with the best intention to serve high-quality data to its respective user base, they were still built in complete isolation without awareness of each other.

To some extent, this behavior was even fostered by the first layers of the data mesh concept, as it is all about giving ownership and responsibility and, with that, also local decision power on how to shape the respective domain's products. With decentralization, you increase speed, but you always risk building up silos that only focus on their given context without a viewpoint beyond these limitations. To counter this natural drifting effect and strive toward organizational alignment, we need to introduce the last layer of the data mesh concept. We need to think about how we achieve interoperability through federated data governance.

As a first step, as soon as a second domain is building a data product, we should look at the semantics needed to join data from multiple domains together. In many cases, this means looking at cross-domain terms such as *user*. As described in the previous example, it is not guaranteed that the term means the same in all domains or that its semantics such as identifier uniqueness are shared. And in fact this is perfectly fine. One of the ideas of data mesh is that there is not one central source of truth but multiple contextualized versions of the truth. The key here is to be aware of this and not be caught by surprise because of the false assumption that a term with the same name must surely have the same meaning and the same semantics everywhere.

Coming from this awareness, an early task of the federated governance group is to bring together representatives from different domains to identify *polysemes*. Polysemy is a concept from language theory describing words or phrases that can have different but related meanings, depending on the context. Once those polysemes have been identified, the relations between their different meanings need to be modeled. Ideally, automatable mappings between the domains can be defined. Having those mappings in place will ease

cross-domain joins tremendously. Figure 5-1 illustrates how concepts such as "user," "session," and "profile" can be interpreted differently in different domains but can be connected by a cross-domain mapping that translates between the different interpretations.

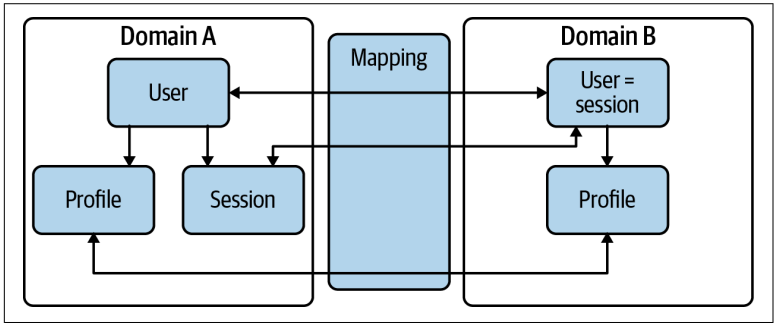

Figure 5-1. Mapping polysemes between domains

There are a couple of pitfalls with this cross-domain modeling. First, not all polysemes and their relations need to be modeled but only those that are relevant for joins that are necessary for foreseeable data applications. Otherwise, it is very easy to get caught up in the attempt to come up with a comprehensive global data model—a task that has led to endless governance processes in the past and to global models that are not being touched anymore once completed because it was so hard to come up with them and they are therefore quickly rendered useless. Instead, a focus on join-relevant polysemes is crucial. A good approach here is to prototype actual cross-domain data applications early in the data mesh journey to quickly identify issues. Second, the work of defining and implementing the concrete mappings should not happen in the federated governance group. This group needs to focus on decisions and triggering discussions, not on implementation. The implementation of mappings and the modeling of relations between polysemes should be delegated to individual domains or short-lived cross-domain working groups.

Use Automation to Enforce Global Rules Without Centralization

One of the main aspects of traditional data governance is the company-wide enforcement of global rules for legal and compliance reasons. An example that comes to mind quickly and that we have

seen as a common challenge is the implementation of Article 17 of the European General Data Protection Regulation (GDPR), which constitutes the right to erasure (often called "right to be forgotten"). Once a request has been made, a company bound to this regulation is obliged to delete all information about the respective individual from their systems. This is one of those situations where you cannot simply let everyone deal with their data as they see fit but where a global data governance is necessary to prevent serious consequences.

The traditional centralized data governance approach to such a task is as follows. All data producers store their data in a central storage system, and there is a central data team that takes care of data governance and therefore has destructive access to all data. They are the ones who fulfill the request by deleting all data associated with the requesting individual. Because they control all the data, they can ensure that no data is left unchecked. However, the central team needs to cooperate with data-producing teams in two ways. First, they need to figure out how data producers store data that is related to individuals and how this data can be queried using the identifier provided in the request (e.g., full name and birthdate). Second, they need to notify data-consuming teams before the deletion in order to make sure that nothing breaks when the data is being deleted. In the long run, this will either result in the central team being very slow with fulfilling such requests or will push them to enforce very rigid rules with data producers on how they are allowed to store their data. The latter will increase the effectiveness of the central team but will slow down the rest of the company while providing little incentive for data producers to adhere to those rules.

A naive approach to data governance decentralization would be to push all the responsibility of data deletion to data product teams, for example, by sending out a global notification to all data product teams and letting them take care of the deletion using the provided identifier. However, this basically means that there is no data governance at all because there is no control over whether or not data product teams ignore the request or how diligent they take care of it.

The goal of federated computational governance, on the other hand, is to give autonomy to data product teams while achieving reliable data governance and efficiency through platform automation. To achieve this, data product teams need to be provided with tooling that allows them to describe their data in a way that has been agreed

on globally. For example, they can describe how all the data corresponding to an individual can be queried and how it is deleted if needed. This description needs to be automatable, that is, it is not enough to just document this relation. For instance, each team can be delegated to use the deletion tools provided by the self-serve data infrastructure platform to carry out the request. As long as the centrally provided tools are used and a confirmation about the completed deletion is sent back to the team that received the initial request, there should be enough confidence that the deletion has been done in a sufficiently governed way while providing full autonomy to data product teams. In Figure 5-2, we illustrate another example of an automatable governance task that we have seen in practice: the enforcement of global encryption policies for personal identifiable information (PII).

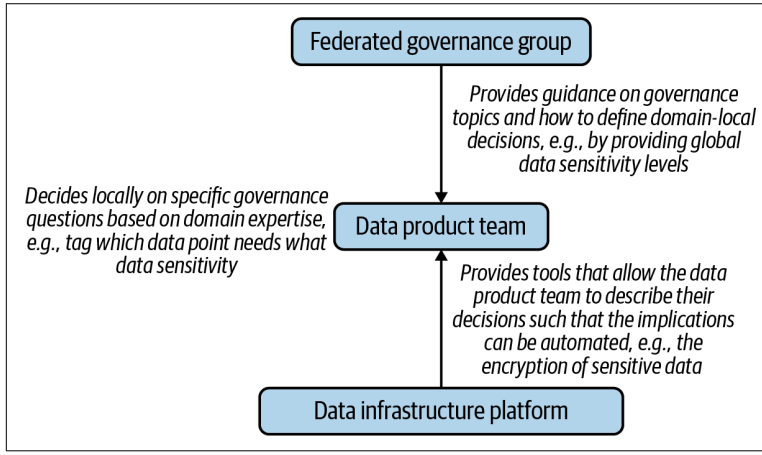

Figure 5-2. Federated governance group, data product team, and data infrastructure platform working together

Similar to the GDPR example, the best approach here is to give autonomy to data product teams by letting them describe what data is PII-relevant and then letting the central data infrastructure take care of the actual implementation of globally agreed-upon encryption policies. To accomplish this, three things have to be in place. First, you need to have globally defined data sensitivity levels and respective encryption policies for each level. To come up with those sensitivity levels is a task for the federated governance group. Second, you need to have centrally provided tooling to tag data accordingly. Third, all data needs to be stored on centrally managed data

infrastructure—not to be confused with centrally owned data—so that the right encryption policies, or changes thereof, can be applied to PII-relevant data automatically without the need for the data-owning teams to do something about it.

Now that we have seen examples of how a federated data governance group, a decentralized data product team, and a data infrastructure platform can efficiently work together to implement data governance in a scalable fashion, the next chapter will provide you with further examples of best practices that we have seen in the industry as well as some common pitfalls.

CHAPTER 6
Industry Practices

You have seen throughout this report that many other data practitioners have been in a similar position as you are right now. We conclude this report by discussing common pitfalls and suggesting best practices that different companies have come up with.

Common Pitfalls

In this section, we will introduce you to a set of traps that are noteworthy to avoid, as they can significantly harm your data mesh journey.

Overloading Your People

When talking to different companies that are just getting started, we're often asked if additional resources are required to take the first steps, or if one can start with the teams and people that are already there. Especially for product managers, the answer is often the latter, since when starting to work with data products, a lot of methodology is indeed very similar to working with other technical products like microservices and applications. However, one important consideration frequently overlooked in these situations is whether the people in question have the capacity to take on such additional responsibilities. You have to keep in mind that in most cases we are considering pilot projects that can have a significant impact on the data-related future of our organization. Don't make the mistake of pushing such important responsibility on teams and people that are

already on low capacity. Even if you already have the required skill sets in your company, involving them in such a project still means they will need to step back on existing responsibilities.

Creating a Platform with Central Data Responsibility

When starting a data platform in your organization or deciding to extend an existing one with additional capabilities, it is important to abstract the new requirements that you collected in a domain-agnostic way. It is easy to fall into the trap (described multiple times in the previous chapters) of taking on central responsibility for domain data. When collecting specific requirements, we tend to design solutions that are tailored to solve exactly the problem at hand. Yet we need to pay attention that this does not involve shifting parts of the data responsibility again toward the platform team. Take, for example, the requirement to build a mechanism that executes GDPR right to be forgotten. Make sure you do so in a data-agnostic way and do not take ownership about the detection and configuration of which datasets to be executed on, as that would again require knowledge about the content of the stored data. At any time, keep in mind that the platform exists to support as many users across your organization as possible, not only a few selected ones.

Building the Perfect Platform Up Front

Building a data platform is striking a balance between short-term needs to support data product teams and a long-term vision to drive sustainable innovation for your company. Driving the platform development has to follow the same product management principles that we ask our data product teams to follow. Selection and prioritization of the features and capabilities to be added to the platform need to be driven by the needs of your stakeholders and the value generated for the company. The desire to anticipate all required platform capabilities up front—even though driven by good intention—has but two flaws. On the one hand, today's assumptions about the needs of tomorrow might already be overthrown before you even finish your design. On the other hand, writing a strategy paper about the desired state of your platform in many cases has to consider a large number of challenges, many parties to be involved, and even more solution options. The sheer complexity of such an effort leads to a process that could require months, if not years, to

complete the design. Before realizing it, your data platform becomes a big waterfall project, which is eating up a lot of resources without netting any benefits until much later in time.

Misunderstanding the Data Mesh Concept

The data mesh concept, while comparatively new, has created a lot of buzz in social networks and social media. Everybody seemingly wants to board the hype train and has a strong opinion on the dos and don'ts. Unfortunately, not everyone with an opinion has fully grasped the core of the concept or has thoroughly informed themselves before jumping into the discussions. If you have read up to this point, you should by now have a good understanding of the core of the data mesh concept and the ideas and intentions behind them. If you now want to put what you learned into practice, you need to keep two points in mind. First, this is the start of a journey during which you still need to identify what exactly data mesh means for your company. Every organizational setup is different and might make the different parts of the concept more or less applicable. Second, the people around you might not have the same level of knowledge that you have (by now). Take the time to share your knowledge and provide references for them to catch up, so that you can discuss the future of your company on equal footing.

Best Practices

In contrast to the previous section, we are now reinforcing some positive behaviors that will help you advance your data mesh journey.

Start Small, but with Commitment

When attempting to move toward a data mesh architecture, do not plan a company-wide program to introduce a data mesh and do not try to secure lots of resources for a big data infrastructure project. At the same time, do not decide on a whim to try out a little data mesh experiment in some lab. Instead, the most successful approach is to carefully select a meaningful use case with a limited but valuable impact. "Meaningful" here means that the use case is solving an actual problem or meets an actual need and is not just a proof-of-concept or toy example. At the same time, the selected use case should not be business critical as this would put too much pressure

on the team implementing it. Select early adopters who are already motivated to drive this initial product development. Provide them all the support you have, including securing sufficient management backing, to make this first data product a success that can be demonstrated and learned from.

Define Your Domains Following Your Business Capabilities

Many data projects fail because they are not driven by a clear business need. The same is true for the definition of domains. Following the ideas of domain-driven design, business terminology and processes are more long-lived than technical terminology and processes. Domains are the main means to structure responsibilities and organizational collaboration in a data mesh setup. For those structures to be rather stable, they should largely follow the structure and processes of your business. Having a high amount of cross-cutting data domains can largely complicate the development of their data products by increasing the number of additional dependencies and by reducing the independence and flexibility of the data products.

Evangelize Data Mesh

As much as data mesh is about cultural and organizational changes, it is also about expanding the perspective of the individual. To do so not only takes time and effort but also requires opportunities for learning and knowledge sharing. Once your company has a decent amount of data mesh practitioners, we highly suggest taking a community-driven approach and fostering continuous exchange and collaboration between the members of the community. Starting off by creating communication channels for early data product teams allows them to discuss when there is still a high amount of uncertainty on how to adapt parts of the data mesh concept to your particular company. Later on it can become an introduction channel for a broader audience within which you discuss and communicate recommendations and best practices, as well as learning materials to get people started. Some of the early adopters might even become interested in giving internal talks or trainings about their initial experiences and what they have learned on the way.

Apply Product Thinking to Platform Development

A common mistake we often see is that platform development is approached entirely differently than product development. This is usually not a conscious decision, but nevertheless it is common at workplaces to see a mindset of "this rule does not apply here because this is platform work." Instead, most successful platform development is fundamentally approached the same way as developing any digital product should be approached: there needs to be a clear MVP state, a product manager, a road map, and a prioritized backlog of planned features that can be killed off in case there is not sufficient demand by the targeted user groups.

Closing Remarks

Let's reflect on the topics we have introduced and how you can apply those ideas moving forward. As a first step, we described how the prevalent analytical data setups with their centralized nature have ultimately led to the development of the data mesh concept. After introducing the main pillars of data mesh, we took you on a journey on how you get started, how you scale, and how you sustain your data mesh. Finally, we have shared with you some common pitfalls and best practices that we have observed across the industry.

We hope we have provided you with lots of food for thought, but likely the most pressing question for you now is how you can turn this into something practical. As a starting point, it is important that you reflect on the different parts of the data mesh concept and understand which parts are most relevant for your organization. As a next step, we can strongly advise starting conversations on the topic with the people around you in your organization. Do not shy away from sharing your thoughts and considerations, but also make sure that others are provided with the material to get up to speed themselves. Training and education are key to a successful data mesh journey. Lastly, join the conversation! Data mesh—at the time of this writing—is a strongly debated topic across the industry. Contribute to the discussions and help us draw out this journey together.

Acknowledgments

We would like to thank our reviewers of this report, Martin Chesbrough, Patrick Klingler, Haris Syed, Scott Hirleman, and Andrew Mott. Your input allowed us to escape our thought bubble and widen our perspective to make this report more approachable. Special thanks go to Zhamak Dehghani, not only for the introduction of the data mesh paradigm and the continuous effort she is putting into driving the topic forward, but also for her direct support and for always having an open ear to discuss new perspectives and ideas. Arif is grateful for the ongoing support he receives from Thoughtworks, both as a company and as a community of thought-leading individuals. We want to personally thank O'Reilly Media for making this report possible and for the great support we received along the way. Lastly, we want to thank Starburst for making this report accessible to a broader audience.

About the Authors

Max Schultze is a data engineering manager working on building a data infrastructure platform at Zalando, Europe's biggest online platform for fashion. His focus is on building data pipelines at petabytes scale, productionizing distributed processing engines like Spark and Trino, and providing services and tooling for data management to hundreds of teams across the company. As an early adopter of the data mesh paradigm, he is frequently advocating its usage through conference appearances and online trainings. Max graduated from Humboldt University of Berlin, actively taking part in the university's initial development of Apache Flink.

Arif Wider is a professor of software engineering at HTW Berlin and a principal technology consultant with Thoughtworks Germany, where he served as head of Data and AI before moving back to academia. As a vital part of his research, teaching, and consulting, he is passionate about distilling and distributing great ideas and concepts that emerge in the software engineering community. He therefore frequently speaks and writes about technology but enjoys nothing more than bringing together people with diverse backgrounds and areas of expertise.

O'REILLY®

There's much more where this came from.

Experience books, videos, live online training courses, and more from O'Reilly and our 200+ partners—all in one place.

Learn more at oreilly.com/online-learning

©2019 O'Reilly Media, Inc. O'Reilly is a registered trademark of O'Reilly Media, Inc. | 175