THE PRACTICAL GUIDE TO DATA MESH

BY GUILLAUME BODET

zeenea

THE PRACTICAL GUIDE TO DATA MESH

SETTING UP AND SUPERVISING AN ENTERPRISE-WIDE DATA MESH

TABLE OF CONTENTS

ABOUT THE AUTHOR

Guillaume Bodet

Co-founder & CPTO

zeenea

Guillaume Bodet is a visionary product leader with an entrepreneurial spirit, combining serious technical abilities and a passion for growing businesses. With a career that began as a software engineer, he has held significant roles including Principal Consultant on Distributed Architectures at Borland, CTO and Associate Partner at Xebia, and CTO - Head of Product & Engineering at Finance Active, culminating in his current position as co-founder and CPTO at Zeenea. Guillaume is also the author of "Scrum en Action" (PEARSON), showcasing his expertise in agile methodologies.

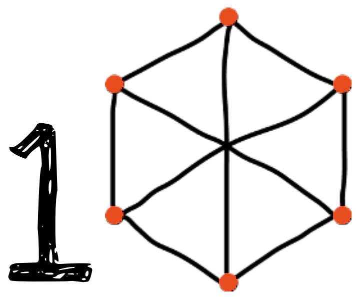
INTRODUCTION

Data Mesh, a Gentle Revolution in Data Management

Nearly 20 years ago, Werner Vogels stepped into his role as CTO at Amazon, setting off a true technological and organizational revolution at his new workplace.

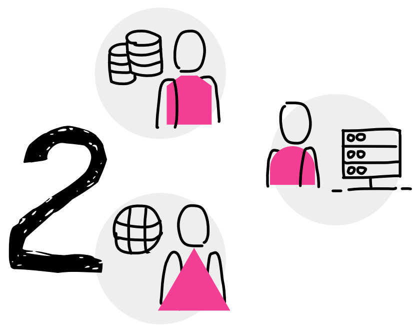
At the time, *amazon.com* was already one of the most visited sites globally, running on a massive technical infrastructure. However, it relied on a monolithic codebase with over 500 developers, following a highly centralized development structure. This setup posed a significant risk for the company: subject to demanding constraints (load, performance, availability, security), the platform could only progress through very strict change controls, moving at a pace incompatible with the ambitious goals of its founder.

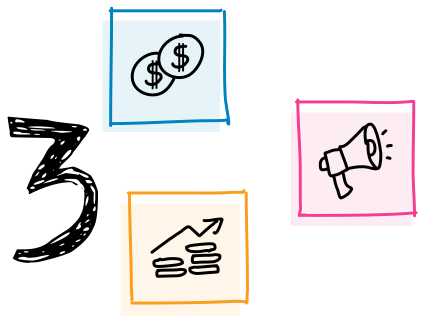
To tackle this challenge, Vogels didn't just redefine the technical architecture of the platform; he embraced a systemic approach, covering architecture, organization, and culture. His approach rested on several key pillars:



Platform modularization - Following the principles of hexagonal architectures, this approach meant establishing a set of distributed components and services, developed and operated autonomously by small agile teams (the famous "2 pizza teams"). It provided assurances in terms of interoperability, backward compatibility, security, and performance.

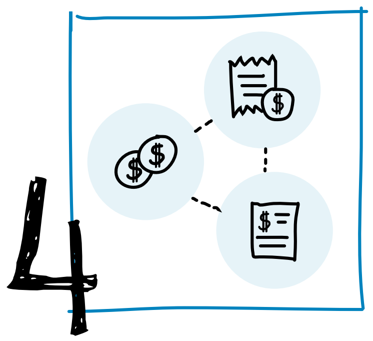
Infrastructure design and management by dedicated, specialized teams - Infrastructure components were provided to development teams as services (for storage, processing, databases, networking, analytics, etc.).

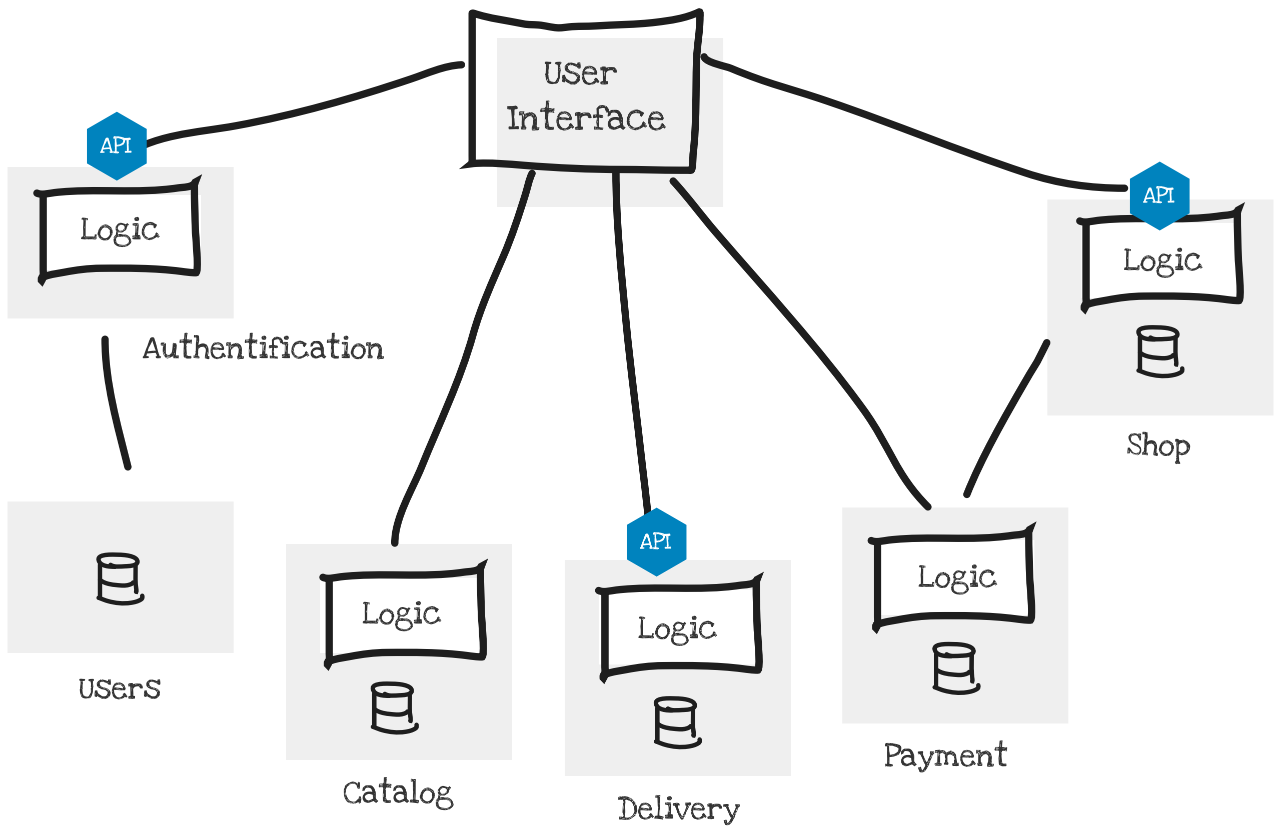




Organization by domains - These domains were structured according to the major operational capabilities of the company (sellers, catalog, customers, search, recommendations, reviews, cart, payment, delivery, etc.), with each domain responsible for the development of its components and services.

Internal product culture - This culture was instilled at all levels to foster innovation and enhance value creation. Services and components developed by the domains were treated and managed as digital products, with their performance continuously measured and optimized by dedicated managers. Some internal products were even commercialized at Amazon - particularly infrastructure, which led to AWS.





Example of a service architecture

Successfully executed, this comprehensive transformation has yielded remarkable results, especially considering Amazon's growth, diversification, and innovation over the past two decades.

Birth of a new paradigm in data management

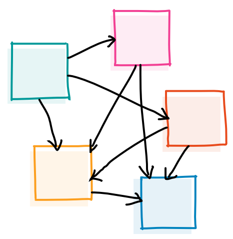
Why discuss the prehistoric times of the web in a paper dedicated to data management? Simply because the world of data management is undergoing a very similar gentle revolution to what Amazon experienced, and subsequently, many other organizations, in the early 2000s.

The paradigm driving this transformation observed over the past twenty years at Amazon now has a name: the data mesh, introduced in 2019 by Zhamak Dehghani.

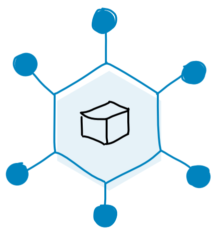
Dehghani's foundational article on the data mesh was published on Martin Fowler's website, one of the leaders in distributed architectures, agile development, and domain-driven design, among others. She fully embraces this legacy and diagnoses a situation very similar to what Vogels encountered when he joined Amazon:

Centralized and monolithic data management, focused around a data lake or data warehouse, creates a massive bottleneck that stifles innovation and severely limits or even eradicates the ability of data teams to meet the increasingly urgent demands of the business.

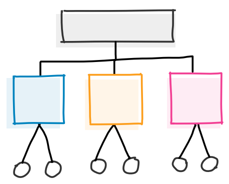
To eliminate this bottleneck and, above all, restore organizations' ability to innovate rapidly, it was necessary to rethink data management practices based on four fundamental principles:



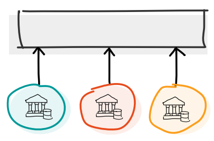
Domain-oriented decentralized data ownership and architecture



Data as a product



Self-serve data infrastructure as a platform



Federated computational governance

According to a study by research and consulting firm Eckerson Group from early 2023, the adoption of data mesh principles is widespread - nearly 70% of the surveyed companies have begun implementing them or plan to do so. Other studies corroborate these figures, and my personal experience confirms it: I've had the opportunity to converse with many data leaders from large companies, and the majority have already incorporated decentralization of data management into their strategic roadmap.

Key factors driving the rise of data mesh

I see two types of converging factors explaining the enthusiasm for data mesh and decentralization of data management: factors related to economic pressure and competitiveness weighing on organizations, and factors related to the very birth of data mesh itself.

Firstly, there's the growing frustration among executive teams struggling to discern the returns on often substantial investments made in data infrastructure and data management over the past decade from their economic results.

This frustration is compounded by **the fear of losing competitiveness due to the inability to take advantage of the rapidly democratizing opportunities offered by artificial intelligence.** Until recently, developing AI models was a long and risky process with uncertain outcomes. The rapid development of highly performant, inexpensive, and easy-to-integrate off-the-shelf models has changed the game entirely. It's now possible to prototype an AI application in a few days by adjusting and combining shelf models. However, scaling requires feeding these models with data that is of quality, traceable, secure, compliant, etc. In short, well-managed data adds additional pressure on centralized data management teams.

These initial factors aren't directly related to the data mesh, but they outline a context in which data organizations are pressured to reform to improve their performance and address current strategic challenges.

Other factors are more directly related to the data mesh itself. The first is the very nature of data mesh. Data mesh is not an architecture, a language, a method, or even a technology - all of which are often complex, controversial, and divisive subjects.

Data mesh simply lays out a few easy-to-understand principles, and these principles aren't prescriptive - they can be implemented in a thousand different ways.

These principles are also not purely academic: they transpose to the world of analytical data the practices that allowed large software organizations to master the complexity of their systems while continuing to innovate at a rapid pace. Data mesh is based on strong theoretical and empirical foundations - it's very hard to resist the argumentation developed by Dehghani. And I suspect that many data leaders, upon reading it, had a genuine epiphany - I am one of them.

Data mesh has the rare quality of easily gaining the support, even enthusiasm, of data teams, including at the decision-making level. This unanimity limits resistance to change, ensures strong sponsorship, and partly explains the speed of its adoption worldwide.

The last factor, probably the most important one:

The principles of data mesh are easy to implement, without significant investments, simply by reallocating existing resources.

When transforming a monolithic software platform into a plethora of loosely coupled and tightly integrated distributed services, one knows that the operation will be lengthy, costly, and risky. For data, the situation is very different. Data is already, by nature, distributed. And all organizations already have the necessary technologies to extract, process, store, and consume their data in higher-level applications.

Implementing the basics of data mesh primarily involves transforming an organization and practices, not making new massive technological investments.

In their eagerness to reform their practices, data leaders have found in data mesh a convincing and accessible framework and have massively included it in their strategic roadmap. It goes without saying, however, that the transition from a centralized data management model to an operational data mesh can only be done gradually - there is no magic wand. And each organization begins this transition in its own context - its strategic challenges, personnel, organization, processes, culture, or even its technological stack.

From literature to practice: the approach at the heart of this ebook

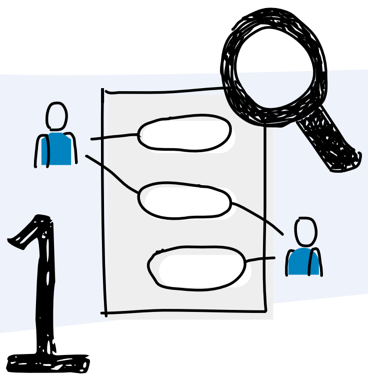
While the literature on data mesh is extensive, it often describes a final state, rarely how to achieve it in practice. The question then arises:

What approach should be adopted to transform data management and implement a data mesh?

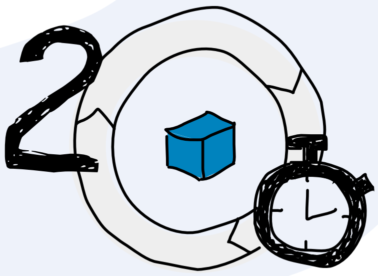
In the following sections, I propose an iterative approach, structured around the four principles of data mesh and informed by certain principles of Lean Manufacturing - particularly the elimination of waste in production chains.

This approach is based on the idea that building a data mesh is a learning process and that this learning can start very quickly by leveraging existing human and technological resources.

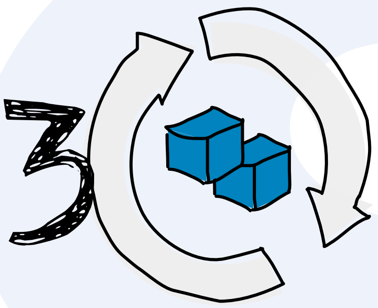
The overall approach is based on the following steps:



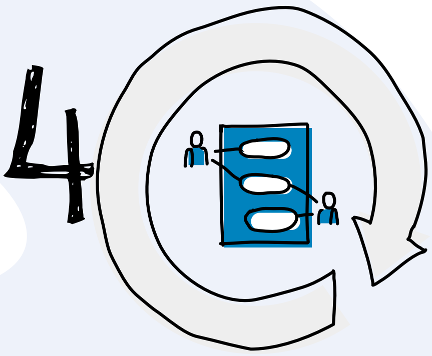
Identify an initial use case and develop it by implementing the 4 principles of data mesh with existing resources.



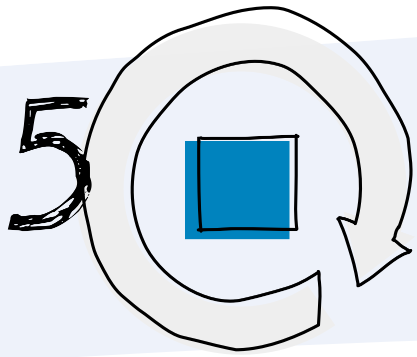
Measure the production and consumption cycle times of data products.



Iterate the production and consumption processes of data products to reduce production cycle times.



Iterate on use cases to increase the reuse of data products.



Iterate on Domains to expand and then generalize the data mesh.

PREMIUM OFFICES EXAMPLE

Throughout this document, and in order to illustrate this iterative approach for implementing a data mesh, we will rely on a concrete example: that of the fictional company Premium Offices - a commercial real estate company whose business involves acquiring properties to lease to businesses.

CHAPTER 1

Building a Pilot Project: The Embryo of Data Mesh

The initial step in the approach outlined in this document to transform data management and implement a data mesh within your organization involves, as often is the case, selecting a pilot use case. This will be developed based on the 4 principles of data mesh, using existing resources, meaning without impacting the organization.

Establishing the pillars of the pilot project

Before embarking on the development of this initial use case, it's vital to focus on several essential prerequisites to ensure a successful start to your data management decentralization initiative.

Identification and selection of domains

The primary prerequisite for launching the pilot project is, of course, the identification of domains - the federation of autonomous domains being at the core of the data mesh.

This step generally poses no difficulty. Indeed, **the concept of domains is already widely understood, and the division into domains is often stable - whether structured according to value chains, major business processes, or organizational operational capabilities.**

Domains sometimes have their own technical teams and operational systems that generate the majority of the data. The transition often involves reallocating data ownership according to an existing structure.

Characteristics and selection of the first use case

The choice of a use case for the pilot project is relatively arbitrary - it could involve revamping an existing dashboard, creating a new dashboard, adding AI capabilities to an application, or even commercializing certain data.

However, this first use case must possess specific characteristics to facilitate optimal learning conditions:

- ☒ **It must focus on usage, not just one or more data products -** the intrinsic value of a data product is null, and its value is realized through its uses, which we will revisit in Chapter II.
- ☒ **It should not be overly cross-cutting and should consume data from one or two domains at most -** ideally, just one.
- ☒ **It should not be overly simplistic and should consume more than one data product; two or three are sufficient -** combining data products is indeed a fundamental learning process.
- ☒ **It should not be overly experimental -** the goal is to achieve concrete results quickly.

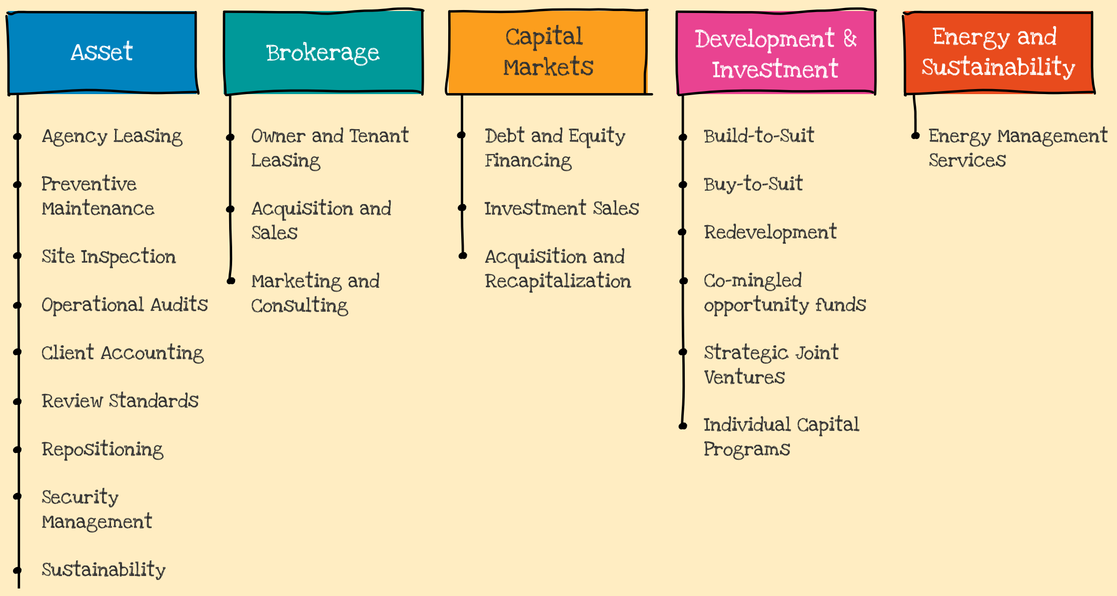
PREMIUM OFFICES EXAMPLE

Premium Offices is already structured around domains that reflect its major capabilities. Here are three examples of domains:

🟦 **Asset -** a domain responsible for acquiring and managing real estate assets. It primarily relies on asset management software.

🟩 **Brokerage -** a domain that manages the commercialization of properties for rent and tenant management. It utilizes Tenant Management software and is responsible for the commercial website and posting offers on specialized marketplaces.

🟧 **Capital Markets -** a domain responsible for loans to finance purchases and optimize the loan portfolio. It uses another specialized software.



Organization by domains - From the main capacities of Premium Offices

Premium Offices already has a modern data platform, based on DBT, Google BigQuery, and Tableau. It is managed by a centralized team supported by a centralized Data Office.

For the pilot project, Premium Offices has chosen to build a credit risk dashboard for its tenants to better anticipate and prevent potential defaults.

This dashboard must combine tenant data from its software and credit data acquired from a specialized provider. These data are already used operationally in the process of evaluating a new tenant.

Building the pilot project development team

As mentioned, the first step in our approach is to identify an initial use case and, more importantly, to develop it by implementing the 4 principles of data mesh with existing resources.

Forming the team responsible for developing the pilot will help implement the first principle of data mesh, *domain-oriented decentralized data ownership.*

PREMIUM OFFICES EXAMPLE

The data required for the pilot belongs to the 🟦 Brokerage domain, where the team responsible for developing the pilot will be created. This multidisciplinary team includes:

A Data Product Owner: should have both a good understanding of the business and a strong data culture to fulfill the following responsibilities: designing data products and managing their lifecycle, defining and enforcing usage policies, ensuring compliance with internal standards and regulations, and measuring and overseeing the economic performance and compliance of their product portfolio.

Two engineers: one from the 🟦 Brokerage domain teams - bringing knowledge of operational systems and domain software engineering practices, and the other from the data team - familiar with DBT, GCP, and BigQuery.

A Tableau developer: who can design and build the dashboard.

The composition of this development team varies depending on the context, but it should meet two requirements: include a Data Product Owner and integrate all the necessary skills to develop and manage its products.

Developing the pilot project with an initial use case

Now that we have identified the domains, defined an initial use case, and assembled the team responsible for its development, it is time to kick off the pilot project and build what we can call the embryo of data mesh.

Defining the first data products

We have already discussed the concept of data products several times. This is the second principle of data mesh, *data as a product*.

Over the past decade, domains have often developed a product culture around their operational capabilities. They offer their products to the rest of the organization as APIs that can be consumed and composed to develop new services and applications. In some organizations, teams strive to provide the best possible experience to developers using their domain APIs: search in a global catalog, comprehensive documentation, code examples, sandbox environments, guaranteed and monitored service levels, etc.

These APIs are then managed as products that are born, evolve over time (without compatibility breaks), enriched, and are eventually deprecated, usually replaced by a newer, more modern, more performant version.

The data mesh proposes to apply this same product-thinking approach to the data shared by the domains.

In some organizations, this product-oriented culture is already well established. In others, it will need to be developed or introduced. But let's not be mistaken:

A data product is not a new digital artifact requiring new technical capabilities (like an API Product). It is simply the result of a particular data management approach exposed by a domain to the rest of the organization.

Managing APIs as a product did not require a technological breakthrough: existing middleware did the job just fine. Similarly, data products can be deployed on existing data infrastructures, whatever they may be.

Technically, a data product can be a simple file in a data lake with an SQL interface; a small star schema, complemented by a few views facilitating querying, instantiated in a relational database; or even an API, a Kafka stream, an Excel file, etc.

A data product is not defined by how it is materialized but by how it is designed, managed, and governed; and by a set of characteristics allowing its large-scale exploitation within the organization. These characteristics are often condensed into the acronym DATSIS (Discoverable, Addressable, Trustworthy, Self-describing, Interoperable, Secure).

In addition, obtaining a DATSIS data product does not require significant investments. It involves defining a set of global conventions that domains must follow (naming, supported protocols, access and permission management, quality controls, metadata, etc.). The operational implementation of these conventions usually does not require new technological capabilities - existing solutions are generally sufficient to get started.

An exception, however, is the catalog. It plays a central role in the deployment of the data mesh by allowing domains to publish information about their data products, and consumers to explore, search, understand, and exploit these data products.

In the data mesh, the data catalog plays a somewhat different marketplace role from its classical usage in large organizations, and it is uncertain whether the existing solutions can properly fulfill this role - we will revisit this in Chapter II.

PREMIUM OFFICES EXAMPLE

To establish an initial framework for the governance of its data mesh, Premium Offices has set the following rules:

✓ **A data product materializes as a dedicated project in BigQuery -** this allows setting access rules at the project level, or more finely if necessary. These projects will be placed in a "data products" directory and a sub-directory bearing the name of the domain to which they belong (in our example, "Brokerage").

✓ **Data products must offer views to access data -** these views provide a stable consumption interface and potentially allow evolving the internal model of the product without impacting its consumers.

✓ **All data products must identify data using common references for common data (Clients, Products, Suppliers, Employees, etc.) -** this simplifies cross-referencing data from different data products (LEI, product code, UPC, EAN, email address, etc.).

✓ **Access to data products requires strong authentication based on GCP's IAM capabilities -** using a service account is possible, but each user of a data product must then have a dedicated service account. When access policies depend on users, the end user's identity must be used via OAuth2 authentication.

✓ **The norm is to grant access only to views -** and not to the internal model.

✓ **Access requests are processed by the Data Product Owner through workflows established in ServiceNow.**

✓ **DBT is the preferred ETL for implementing pipelines -** each data product has a dedicated repository for its pipeline.

- ✓ **A data product can be consumed either via the JDBC protocol or via BigQuery APIs (read-only).**
- ✓ **A data product must define its contract -** data update frequency, quality levels, information classification, access policies, and usage restrictions.
- ✓ **The data product must publish its metadata and documentation in a marketplace -** in the absence of an existing system, Premium Offices decides to document its first data products in a dedicated space on its company's wiki.

This initial set of rules will of course evolve, but it sets a pragmatic framework to ensure the DATSIS characteristics of data products by exclusively leveraging existing technologies and skills.

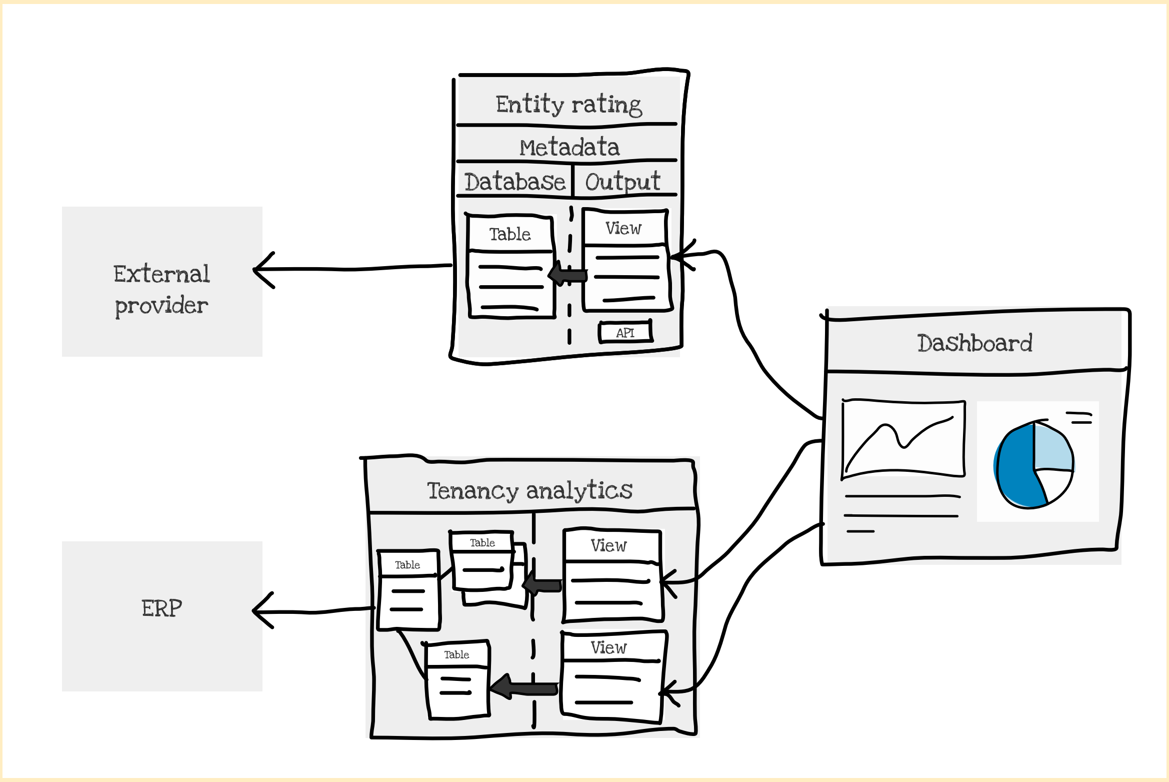
For its pilot, Premium Offices has chosen to decompose the architecture into two data products:

- ☐ **Tenancy analytics -** this first data product offers analytical capabilities on lease contracts - entity, parent company, property location, lease start date, lease end date, lease type, rent amount, etc. It is modeled in the form of a small star schema allowing analysis along 2 dimensions: *time and tenant* - these are the analysis dimensions needed to build the first version of the dashboard. It also includes one or two views that leverage the star schema to provide pre-aggregated data - these views constitute the public interface of the data product. Finally, it includes a view to obtain the most recent list of tenants.

- ☐ **Entity ratings** - this second data product provides historical ratings of entities in the form of a simple dataset and a mirror view to serve as an interface, in agreement with common rules. The rating is obtained from a specialized provider, which distributes them in the form of APIs. To invoke this API, a list of entities must be provided, obtained by consuming the appropriate interface of the ☐ **Tenancy analytics** product.

To identify entities and allow data cross-referencing, Premium Offices uses the LEI (Legal Entity Identifier) and already knows that this data has quality issues - the LEI is not systematically filled in its ERP, and when it is filled in, it is sometimes incorrect. This quality issue does not need to be resolved immediately - but it must be documented and will be a matter for the product owner.

The general schema of this embryo of data mesh at Premium Offices is as follows:



Data mesh embryo at Premium Offices

Designing a data product is certainly not an exact science - for our example, we could have also made only one product, or three or four. To guide this choice, it is once again useful to leverage some best practices from distributed architectures - a data product must:

- Have a single and well-defined responsibility.
- Be usable in several different contexts and therefore support polyglotism.
- Have stable interfaces and ensure backward compatibility.

As our data products begin to take shape, let's move on to the third principle of the data mesh, *self-serve data platform.*

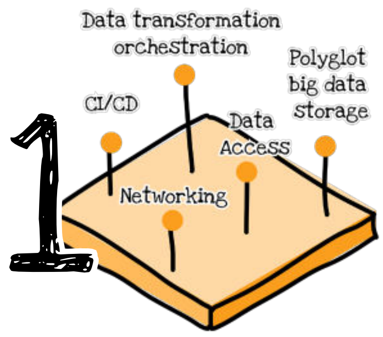
Domain tooling: the data platform of the data mesh

One of the main barriers to decentralization is the risk of multiplying the efforts and skills required to operate pipelines and infrastructures in each domain. But in this regard, there is also a solid state-of-the-art inherited from distributed architectures.

The solution is to structure a team responsible for providing domains with the technological primitives and tools needed to extract, process, store, and serve data from their domain.

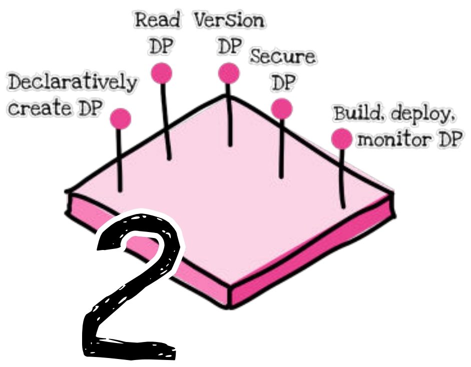
This model has existed for several years for application infrastructures and has gradually become generalized and automated through virtualization, containerization, DevOps tools, and cloud platforms. Although data infrastructure tooling is not as mature as software infrastructure, especially in terms of automation, most solutions are transferable, and capabilities are already present in organizations as a result of past investments. Therefore, there is nothing preventing the establishment of a data infrastructure team, setting its roadmap, and gradually improving its service offering: simplification and automation being the main axes of this progression.

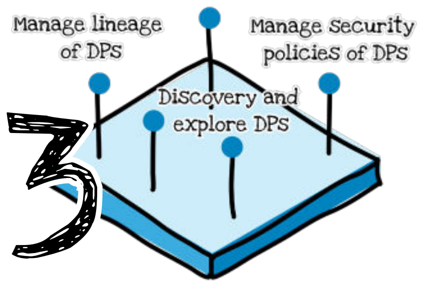
The data platform for data mesh covers a wide range of capabilities, broader than infrastructure services. This platform is divided into three planes:



The Data infrastructure provisioning plane - provides low-level services to allocate the physical resources needed for big data extraction, processing, storage, real-time or non-distributed distribution, encryption, caching, access control, network, co-location, etc.

The Data product developer experience plane - provides the tools needed to develop data products: declaration of data products, continuous build and deployment, testing, quality controls, monitoring, securing, etc. The idea is to provide abstractions above the infrastructure to hide its complexity and automate the conventions adopted on the mesh scale.





The Data mesh supervision plane - provides a set of global capabilities for discovering data products, lineage, governance, compliance, global reporting, policy control, etc. We will revisit this in Chapter II.

On the infrastructure side, the data mesh does not require new capabilities - the vast majority of organizations already have a data platform. The implementation of the data mesh also does not require a centralized platform. Some companies have already invested in a common platform, and it seems logical to leverage the capabilities of this platform to develop the mesh. But others have several platforms, some entities, or certain domains having their infrastructure. It is entirely possible to deploy the data mesh on these hybrid infrastructures: as long as the data products respect common standards for addressability, interoperability, and access control, the technical modalities of their execution are of little importance.

PREMIUM OFFICES EXAMPLE

Premium Offices has invested in a shared cloud platform - specifically, GCP (Google Cloud Platform).

The platform includes experts in a central team who understand its intricacies. For its pilot project, Premium Offices simply chose to integrate one of these experts into the project team. This individual will be responsible for finding solutions to automate the deployment of data products as much as possible and identifying manual steps that can be automated later, as well as any missing tools.

Development of the first data products

Developer experience is a fundamental aspect of the data mesh, with the ambition to converge the development of data products and the development of services or software components. It's not just about being friendly to engineers but also about responding to a certain economic rationality:

The decentralization of data management implies that domains have their own resources to develop data products. In many organizations, the centralized data team is not large enough to support distributed teams. To ensure the success of the data mesh, it is essential to be able to draw from the pool of software engineers, which is often larger.

The state of the art in software development relies on a high level of automation: declarative allocation of infrastructure resources, automated unit and integration testing, orchestrated build and deployment via CI/CD tools, Git workflows for source and version management, automatic documentation publishing, etc.

The development of data products should converge toward this state of the art - and depending on the organization's maturity, its teams, and its technological stack, this convergence will take more or less time. The right approach is to automate as much as possible using existing and mastered tools, then identify operations that are not automated to gradually integrate additional tooling - this is typically the responsibility of the "Infra & Tooling" team (we will come back to this approach later).

In practice, here is what constitutes a data product:

Code first - for pipelines that feed the data product with data from different sources or other data products; for any consumption APIs of the data product; for testing pipelines and controlling data quality; etc.

Data, of course - but most often, the data exists in systems and is simply extracted and transformed by pipelines. Therefore, it is not present in the source code (excluding exceptions).

Metadata - some of which document the data product: schema, semantics, syntax, quality, lineage, etc. Others are intended to ensure product governance at the mesh scale - contracts, responsibilities, access policies, usage restrictions, etc.

Infrastructure - or more precisely, the declaration of the physical resources necessary to instantiate the data product: deployment and execution of code, deployment of metadata, resource allocation for storage, etc.

PREMIUM OFFICES EXAMPLE

For its pilot project, Premium Offices must develop two data products, ☐ **Entity ratings** and ☐ **Tenancy analytics**.

The company's development teams use GitHub to manage the development process and automate the integration and deployment of software components (CI/CD). The development of data products will therefore rely on the same tools: the source code of each data product will be managed in a dedicated repository on GitHub, and deployment on GCP will be automated via CI/CD.

However, using GCP for the software part is not familiar, and the central data team has not invested in DevOps tools to automatically create resources. The pilot team adopts this state-of-the-art and manually creates the necessary resources using the GCP admin console - setting up Terraform is simply on the roadmap of the "Infra & Tooling" team.

For the rest, things are relatively simple: the ☐ **Entity ratings** product is mainly composed of a dbt job that queries the list of tenants via the view provided by the ☐ **Tenancy analytics** product, invokes the data provider's API to obtain the ratings, feeds a dataset with timestamped ratings, and creates a view to expose them. The rating data is subject to a license that states that the information can only be used for internal purposes and not distributed to third parties. This usage restriction will be clearly documented and governed by the Product Owner.

The ☐ **Tenancy analytics** product is also composed of a dbt job that reads raw data from the ERP (this raw data has already been extracted and integrated into GCP) and builds the small star schema and associated views.

Deployment via CI/CD, scheduling, and execution of dbt jobs on GCP are well documented and pose no particular problem for the pilot team. It also uses DBT features to automatically test jobs and document views and tables. Some steps remain manual: creating resources in GCP, documenting the data product in the company wiki, defining policies and access controls, and data quality (which is not assessed or governed at this stage).

Transition to a federated governance model

The final fundamental principle of data mesh is *federated computational governance.* In the data mesh, governance has a federated structure. This governance model is well-known and already operational in certain highly decentralized groups.

In a federated structure, the central body defines the rules and standards that domains must adhere to. Local leaders are responsible for implementing these rules in their domain and providing the central body with evidence of their compliance - usually in the form of reporting.

Although the model is theoretically simple, its implementation often faces internal cultural challenges. This is particularly the case in heavily regulated sectors, where centralized governance teams are reluctant to delegate all or part of the controls they historically had responsibility for.

Federated governance also faces a rarely favorable ground reality: data governance is closely linked to risk management and compliance, two areas that rarely excite operational teams. Consequently, it becomes difficult to identify local responsible parties or to transfer certain aspects of governance to data product owners - who, for the most part, must already learn a new profession. Therefore, in most large organizations, the federated structure will likely be emulated by the central body and then gradually implemented in the domains as their maturity progresses.

To avoid an explosion of governance costs or its fragmentation, Dehghani envisions that the data platform could eventually automatically support entire aspects of governance. It's a nice dream but seems very distant for most.

So, what aspects of governance are likely to be automated?

Quality controls - many solutions already exist.

Fine-grained access policy management - there are already solutions, all of which rely at least on tagging information.

Traceability - development teams can already automatically extract complete lineage information from their data products and document transformations.

With a little imagination, one could even imagine generative AI analyzing transformation SQL queries and translating them into natural language (solutions exist).

The road is long, of course, but decentralization allows for very iterative progress, domain by domain, product by product. And let's also remember that any progress in automating governance, in whatever aspect, relies on the production and processing of metadata.

PREMIUM OFFICES EXAMPLE

At Premium Offices, the Data Office has a very defensive governance culture - as the company operates in the capital market, it is subject to very strict regulatory constraints.

As part of the pilot, it was decided not to impact the governance framework. Quality and traceability remain the responsibility of the Data Office and will be addressed retroactively with their tools and methods. Access control will also be its responsibility - a process is already in place, in the form of a ServiceNow workflow (setting permissions on BigQuery requires several manual operations and reviews). The only concession is that the workflow will be modified so that access requests are verified by the Data Product Owner before being approved and processed by the Data Office. In other words, a small step toward federated governance.

Regarding metadata, the new tables and views in BigQuery must be documented, at both the conceptual and physical levels, in the central data catalog (which is unaware of the concept of data product). It is a declarative process that the pilot team already knows. Any column tagging will be done by the Data Office after evaluation.

For the rest, user documentation for data products will be disseminated in a dedicated space on the internal wiki, organized by domain, which allows for very rich and structured documentation and has a decent search engine.

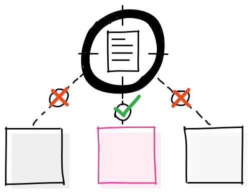
CHAPTER 2

Scaling Up the Data Mesh

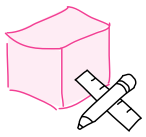
Premium Offices has successfully completed its pilot project and laid the groundwork for a future (and still hypothetical) data mesh. It is now time to capitalize on this success to industrialize it at the organization's scale, and to include it in the company's strategic roadmap.

This strategic objective of decentralizing data management complements two other operational objectives set by the Data Office, itself pressured by regulatory authority recommendations: automating quality controls on critical data and extracting technical and functional lineage from the same critical data.

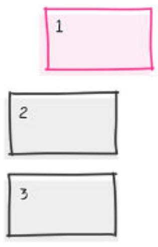
First, at the strategic level:



The pilot project generally concerns only one domain - usually the most mature one. It seems wise to first extend the data mesh in this domain to capitalize well on the practices before extending it to other domains.



The design and combination of data products require architectural work that should not be underestimated.



Prioritization should be done based on usage, prioritizing new ones over old ones - in practice, this means that existing uses (dashboards, reports, applications, etc.) should only be connected to the data mesh as part of a major evolution of these uses. The data mesh is not an end in itself. In short, we prioritize by value.

Then, on the operational level:

In her articles, Dehghani describes a data mesh in which the platform handles everything - resource allocation, build, deployment, and monitoring of data products, access management, quality controls, lineage, compliance, performance analysis, etc. This level of automation is for many a distant ideal from their operational reality. And it suggests new monumental investments with uncertain returns.

Once again, we can take reassurance by looking at recent history. The first large distributed architectures worked by manual effort, and gradually, solutions appeared to automate entire aspects. These solutions have often been the product of web giants, who have open-sourced some of the solutions they developed internally.

No data platform on the market covers all of these capabilities, and I doubt there will ever be one. However, many free or low-cost solutions exist, and they are generally easy to integrate into various existing tools. Many of them come from the world of distributed architectures and are already very robust.

The right approach, once again, is to move forward in small steps, automating one aspect of one data product, and then generalizing the practice - this is the role of the "Infra & Tooling" team.

The question remains of how to prioritize these automation projects.

Optimizing the production and consumption of data products

The introduction of data products into data management has an interesting side effect. It allows us to consider the development of data products as an activity of producing a digital object - just like a service, a component, or an application.

Reducing production cycles based on Lean Manufacturing principles

Therefore, **it is possible to exploit certain tools of Lean Manufacturing to gradually improve the production and consumption of data products** and increase the overall throughput of the system - and, in this case, the multiplicity of uses.

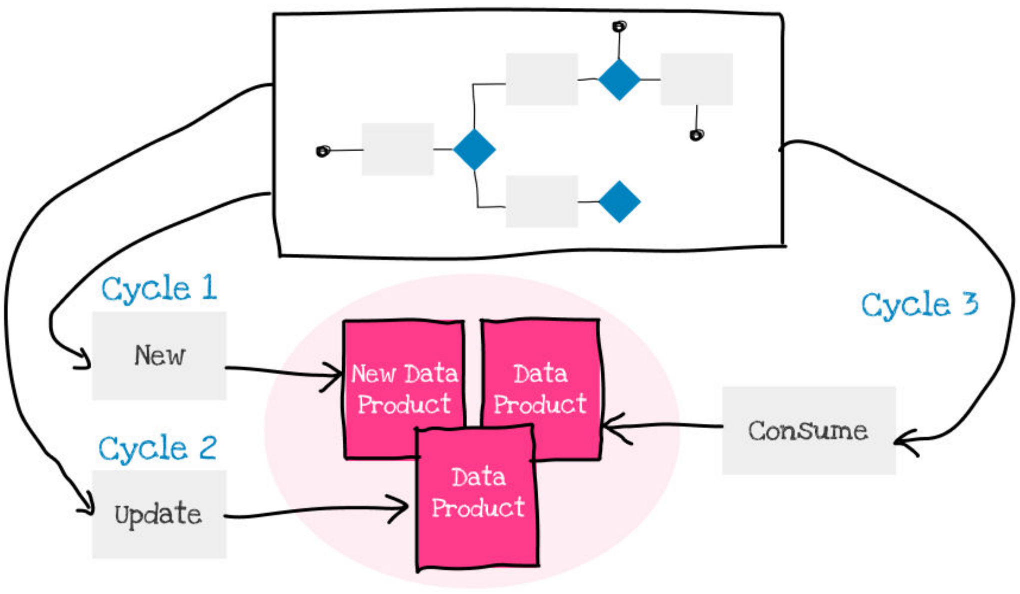
One of the principles of Lean Manufacturing is to look at a system through its value chains, measure the cycle times, and then reduce them. The principle may seem a bit abstract, but it is quite simple in practice.

The production of data products includes 3 value chains (or cycles):

[Cycle 1] Creation of a new data product - its lead time can be defined as the time elapsed between the moment the development of the data product is validated and the moment the first version of the data product is completed.

[Cycle 2] Update of a data product - its lead time is the time elapsed between the validation of the evolution and its deployment.

[Cycle 3] Consumption of a data product - its lead time separates the moment when a consumer defines their need and the moment when they can use the data from a data product that meets their needs.



Production cycles of data products

To define *Cycle 1*, it is necessary to agree on what a completed data product is - otherwise, steps may be missed when trying to reduce the cycle time. The definition may vary from one environment to another.

PREMIUM OFFICES EXAMPLE

In the case of Premium Offices, the following definition has been adopted - it can of course be reviewed later:

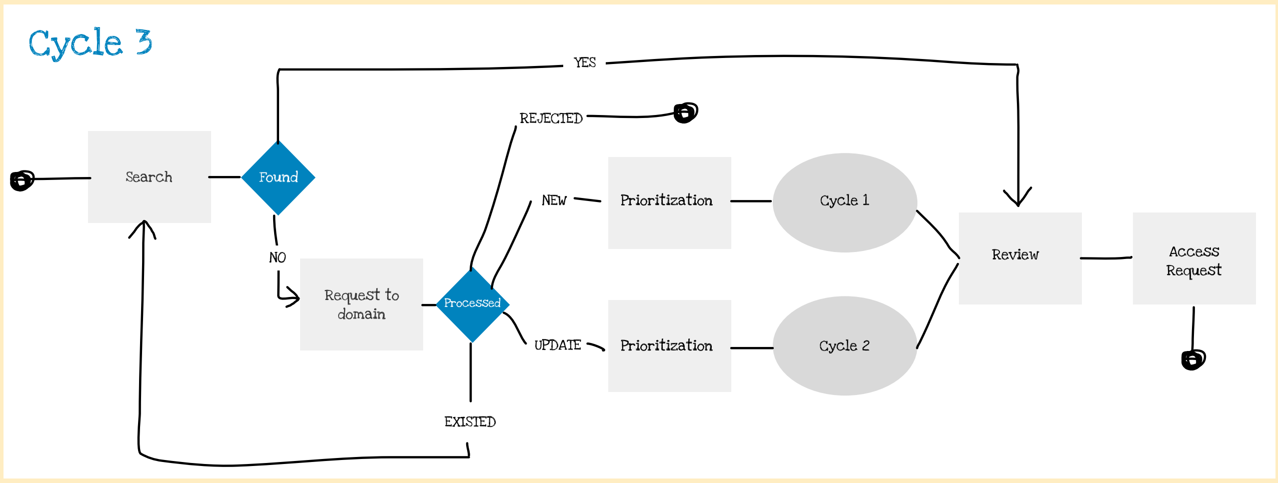
- ✓ The data product is deployed in the production environment and complies with the technical and security standards of the Mesh.
- ✓ If the data product involves critical data or is involved in the data production chain of critical data, the concerned data must undergo quality controls as defined by the Data Office.
- ✓ Under these same conditions, the technical lineage of the data product must be transmitted to the Data Office in an agreed format.
- ✓ The tables of the data product are documented in the catalog following global conventions.
- ✓ The functional documentation of the data product is available on the wiki.

Once the starting and ending points of each cycle are well defined, it is possible to break them down into stages - this breakdown is called a **value stream map.** This concept can be illustrated by decomposing *Cycle 3*.

PREMIUM OFFICES EXAMPLE

Here is the breakdown of the steps of *Cycle 3* at Premium Offices:

- The consumer searches for the data product in the wiki.
- If found, the consumer consults its documentation to understand how to use it.
- Otherwise, the consumer submits a request to the domain owner - or the Data Office - if unsure of whom to address the request to.
- The domain owner reviews the request, with four possible outcomes:
  - → They inform the consumer that a data product meeting their needs already exists and provide the URL in the wiki for easy access to documentation.
  - → They consider it a new data product, which will be prioritized - once planned, it enters *Cycle 1*. Upon completion of this cycle, they provide the URL of the data product documentation to the client.
  - → They believe the need can be met by evolving an existing data product - they provide its URL to the client and submit the evolution request. Once planned, the evolution enters *Cycle 2*, and the client regularly checks for updates to the data product in the wiki.
  - → They simply refuse the request, which interrupts the cycle.
- Once the data product is identified and understood, the client triggers an access request. It is processed by a ServiceNow workflow. Once the request is processed and the appropriate permissions are set up on BigQuery, the client can consume the data - marking the end of the cycle.



Breakdown of *Cycle 3* for data product production

The objective of this stage in setting up a data mesh is to reduce the production cycle times of data products.

For each of the steps, it is possible to measure (or estimate) the time needed to complete it. It is also possible to estimate the delays between steps, which lengthen the cycle time. The frequency of certain loops or paths in this map can be measured. Additionally, the frequency of cycle failure, which can reveal other sources of waste, can be measured.

The final idea is to identify the most significant time losses and act on them - sometimes by improving certain processes, often by introducing new automation tools.

Not all cycles are equivalent. Some are triggered more often than others - it is natural to prioritize optimizing these cycles first.

In the initial phase of implementing the data mesh, it is often useful to optimize *Cycle 1*. Indeed, in a developing mesh, creating a new data product is common. However, quite quickly, as with all digital products, update operations will multiply - in the long run, maintaining and evolving a digital product is notoriously much more costly than its initial development. A good approach is therefore to prioritize optimizations that concern steps common to *Cycles 1 and 2.*

PREMIUM OFFICES EXAMPLE

After this production and consumption process analysis, Premium Offices makes two decisions:

- **Automation of physical resource allocation in GCP using Terraform is not a priority -** manual creation of resources is certainly tedious but infrequent and low-risk as it is supported by already established processes.

- **Production of lineage information on data products is, however, identified as a significant time loss -** manually performed, it is a lengthy operation with a high risk of error. Moreover, it is necessary to review this lineage with each product evolution: therefore, an initiative is launched to propose tools to automate the extraction of the technical lineage of a data product and integrate this tooling into CI/CD. The project is entrusted to the "Infra & Tooling" team.

Analyzing the cycle times of value chains is a valuable tool for supporting the progressive deployment of the data mesh and guiding the roadmap of the "Infra & Tooling" team - especially for *Cycles 1 and 2*. This analysis must be continuous because each optimization shifts the constraint and may challenge the previous prioritization.

Improving the processes of consumption and reuse of data products

The *Cycle 3* presented earlier is somewhat unique. Indeed, even if this cycle is infrequent initially - the data product offering must first expand for clients to multiply - it becomes critical for scaling.

After all, a data product has no inherent value until it contributes, directly or indirectly, to a valid use. The reuse rate is also a good indicator of the value of a data product.

Initially, due to an insufficient offering, the performance of *Cycle 3* is highly dependent on the other two - this simply means that existing data products rarely satisfy new uses as they are. New ones need to be created or existing ones adapted, and this step constitutes the biggest part of the cycle time. But quite rapidly, the cycle will reduce to the following steps:

1. Searching for data products.
2. Consulting the data product documentation to understand its content, structure, and how to use it.
3. Requesting access to the data product.

This workflow constitutes the customer experience of the data mesh - an experience, on paper, very similar to that of a marketplace or an e-commerce system: I search, I consult, I choose, I order, I am delivered. We will revisit this in Chapter III.

Drawing the data mesh supervision plane

There are pragmatic approaches that allow laying the foundations of a data mesh and guiding its deployment in a large organization, gradually improving the platform's performance and usability for both data product producers (their developers) and consumers.

I do not underestimate the difficulty of introducing a product culture above a data culture that is often still in its infancy, or convincing governance leaders to delegate some of their responsibilities to the domains.

Decentralizing and automating data management are not projects but long-term programs whose foundations can be quickly laid, but whose maturity will take years, probably less than for software architectures, which have paved the way.

Innovation around data platforms is supported, and new tools are likely to gradually complement existing ones. Through the example of Premium Offices, it is evident that some capabilities of the data platform are challenging to implement with existing tools, mainly those belonging to the data mesh supervision plane.

Recall that the data mesh supervision plane, which only makes sense on a global scale, provides a set of capabilities for exploiting and governing the mesh as a whole. These capabilities can be grouped according to the three main categories of mesh users:

Data consumers - who need a simple system to search, understand, and order the data products they want to use.

Domain data producers - who need a system to publish information about their data products, announce new products or new versions, manage access requests, manage evolution or new product requests, provide evidence of regulatory compliance, and monitor their product's performance.

Federal governance leaders - who want to control the compliance of data products with common or regulatory rules and supervise the overall performance of the data mesh to guide its development.

PREMIUM OFFICES EXAMPLE

At Premium Offices, these capabilities were set aside for pragmatism or implemented in a rudimentary way.

Data product documentation was placed partly in the group catalog, essentially to support the Data Office's governance processes, and partly in a wiki - whose ergonomics and search capabilities will likely not withstand the multiplication of data products.

For data producers, creating and maintaining this documentation is manual, lengthening the cycle times.

Ultimately, metadata is scattered across different systems, in various more or less structured formats, making it difficult to access and impossible to activate for automating certain processes - especially access control.

Deploying a marketplace to facilitate data product consumption

Let's quickly go back to the notion of a data product, which we believe is the cornerstone of the data mesh and the first step in transforming data management. This notion is also at the heart of the *Data Fabric approach*.

Sharing and exploiting data products through metadata

A data product, as we've seen, is a governed, reusable, scalable dataset offering data quality and compliance guarantees to various regulations and internal rules.

Note that this definition is quite restrictive - it excludes other types of products such as machine learning algorithms, models, or dashboards. While it's desirable for these artifacts to be managed as products, they are not data products - They are other types of products, which could be very generally termed "Analytics Products", and of which data products are one subset.

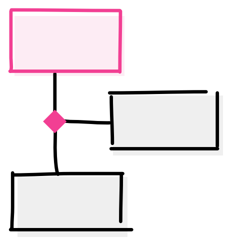
In practice, an operational data product consists of two things:

Data - materialized on a centralized or decentralized data platform, guaranteeing data addressing, interoperability, and access security.

Metadata - providing all the necessary information for sharing and using the data.

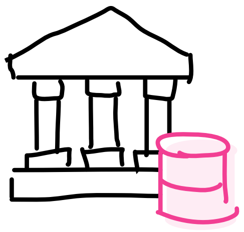
Metadata ensures that consumers have all the information they need to use the product.

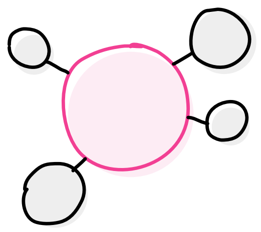
It typically covers the following aspects:



Schema - providing the technical structure of the data product, data classification, samples, and their origin (lineage).

Governance - identifying the product owner(s), its successive versions, its possible deprecation, etc.





Semantics - providing a clear definition of the exposed information, ideally linked to the organization's business glossary, and comprehensive documentation of the data product.

Contract - defining quality guarantees, consumption modalities (protocols and security), potential usage restrictions, redistribution rules, etc.

In the data mesh logic, these metadata are managed by the product team and are deployed according to the same lifecycle as data and pipelines. There remains a fundamental question: where can metadata be deployed?

Using a data marketplace to deploy metadata

Most organizations already have a metadata management system, usually in the form of a Data Catalog.

But data catalogs, in their current form, have major drawbacks:

They don't always support the notion of a data product - it must be more or less emulated with other concepts.

They are complex to use - designed to catalog a large number of assets with sometimes very fine granularity, they often suffer from a lack of adoption beyond centralized data management teams.

They mostly impose a rigid and unique organization of data, decided and designed centrally - which fails to reflect the variety of different domains or the organization's evolution as the data mesh expands.

Their search capabilities are often limited, particularly for exploratory aspects - it's often necessary to know what you're looking for to be able to find it.

The experience they offer sometimes lacks the simplicity users aspire to - search with a few keywords, identify the appropriate data product, and then trigger the operational process of an access request or data delivery.

Given these shortcomings, a new concept is gaining popularity, the internal marketplace, or *Enterprise Data Marketplace (EDM)*. Like a general-purpose marketplace, the EDM aims to provide a shopping experience for data consumers.

Setting up an internal data marketplace

The EDM is thus a simple system in which consumers can search among the data product offerings for one or more eligible to perform a specific use case, become aware of the information related to these products, and then order them. The order materializes as access opening, physical data delivery, or even a request for data product evolution to cover the new use case.

There are three main options for setting up an internal data marketplace:

→ **Develop it -** this is a lengthy and costly option, but holds the promise of a user experience optimized for the organization.

→ **Integrate a solution from the market -** there are several available, initially designed to ensure data commercialization or exchange outside the organization.

→ **Use existing systems -** like Premium Offices, which chose a combination of its Data Catalog and its corporate wiki.

We will not delve further into the "in-house" development option - much has already been said about the advantages and disadvantages of this type of solution versus a commercial one.

Although often offering a satisfying user experience and native support for the data product concept, commercial marketplaces often have significant drawbacks: highly focused on transactional aspects (distribution, licensing, contracting, purchase or subscription, payment, etc.), they are often poorly integrated with internal data platforms and access control tools.

They generally require data to also be distributed by the marketplace - meaning they constitute a new infrastructure component onto which data must be transferred to be shared (such a system is sometimes called a *Data Sharing Platform*).

In a pragmatic approach, we do not believe that, in most cases, it is desirable to introduce a new infrastructure component to deploy a data mesh - as already mentioned, it seems highly preferable to leverage existing capabilities as much as possible.

A data marketplace is thus an essential component to ensure the exploitation of the data mesh on a larger scale - it allows data consumers to have a simple and effective system to search for and access data products from various domains.

Feeding the marketplace via domain-specific data catalogs

Structuring data management around domains and data products is an organizational transformation that does not change the operational reality of most organizations: data is available in large quantities, from numerous sources, evolves rapidly, and its control is complex. Data catalogs traditionally serve to inventory all available data and manage a set of metadata to ensure control and establish governance practices.

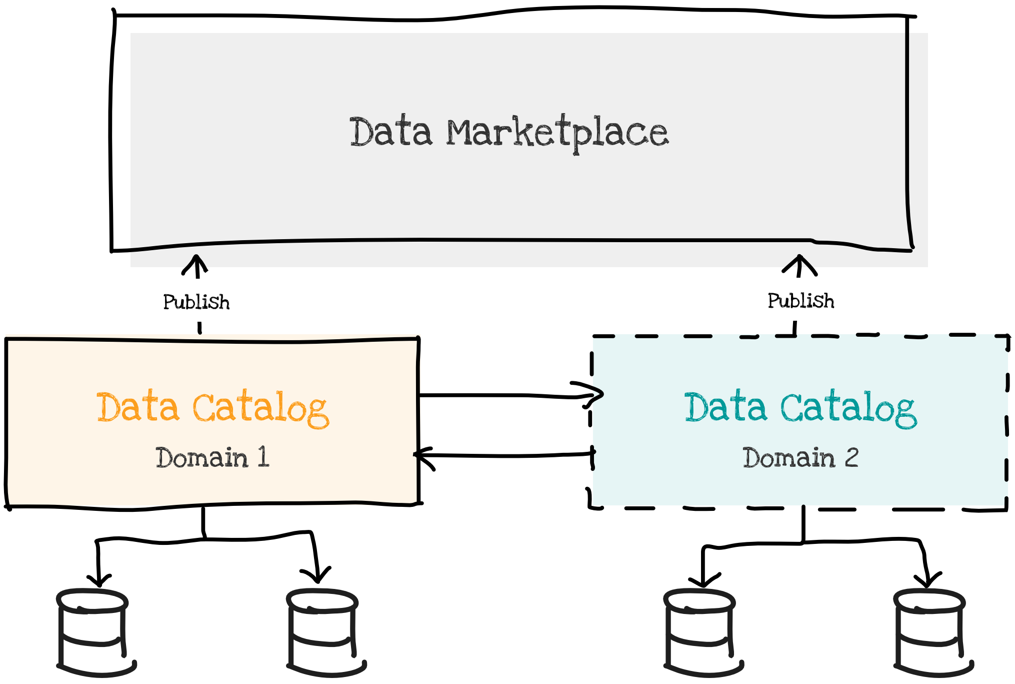
Data mesh does not eliminate this complexity: it allows certain data, managed as data products, to be distinguished and intended for sharing and use beyond the domain to which they belong. But each domain is also responsible for managing its internal data, those that will be used to develop robust and high-value data products - its proprietary data, in other words.

In the data mesh, the need for a Data Catalog does not disappear, quite the contrary: each domain should have a catalog allowing it to efficiently manage its proprietary data, support domain governance, and accelerate the development of robust and high-value data products. Metadata management is thus done at two levels:

At the domain level - in the form of a catalog allowing the documentation and organization of the domain's data universe. Since the Data Catalog is a proprietary component, it is not necessary for all domains to use the same solution.

At the mesh level - in the form of a marketplace in which the data products shared by all domains are registered; the marketplace is naturally common to all domains.

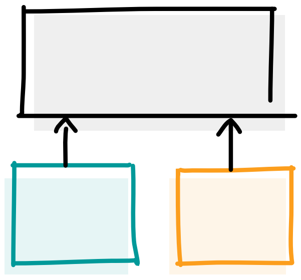
With a dedicated marketplace component, the general architecture for metadata management is as follows:



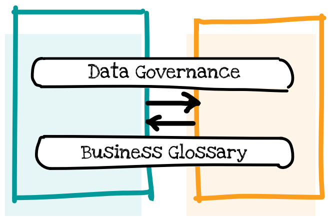
General architecture for metadata management

In this architecture, each domain has its own catalog - which may rely on a single solution or not - but should be instantiated for each domain to allow it to organize its data most effectively and avoid the pitfalls of a universal metadata organization.

The marketplace is a dedicated component, offering simplified ergonomics, and in which each domain deploys metadata (or even data) for its data products. This approach requires close integration of the different modules:

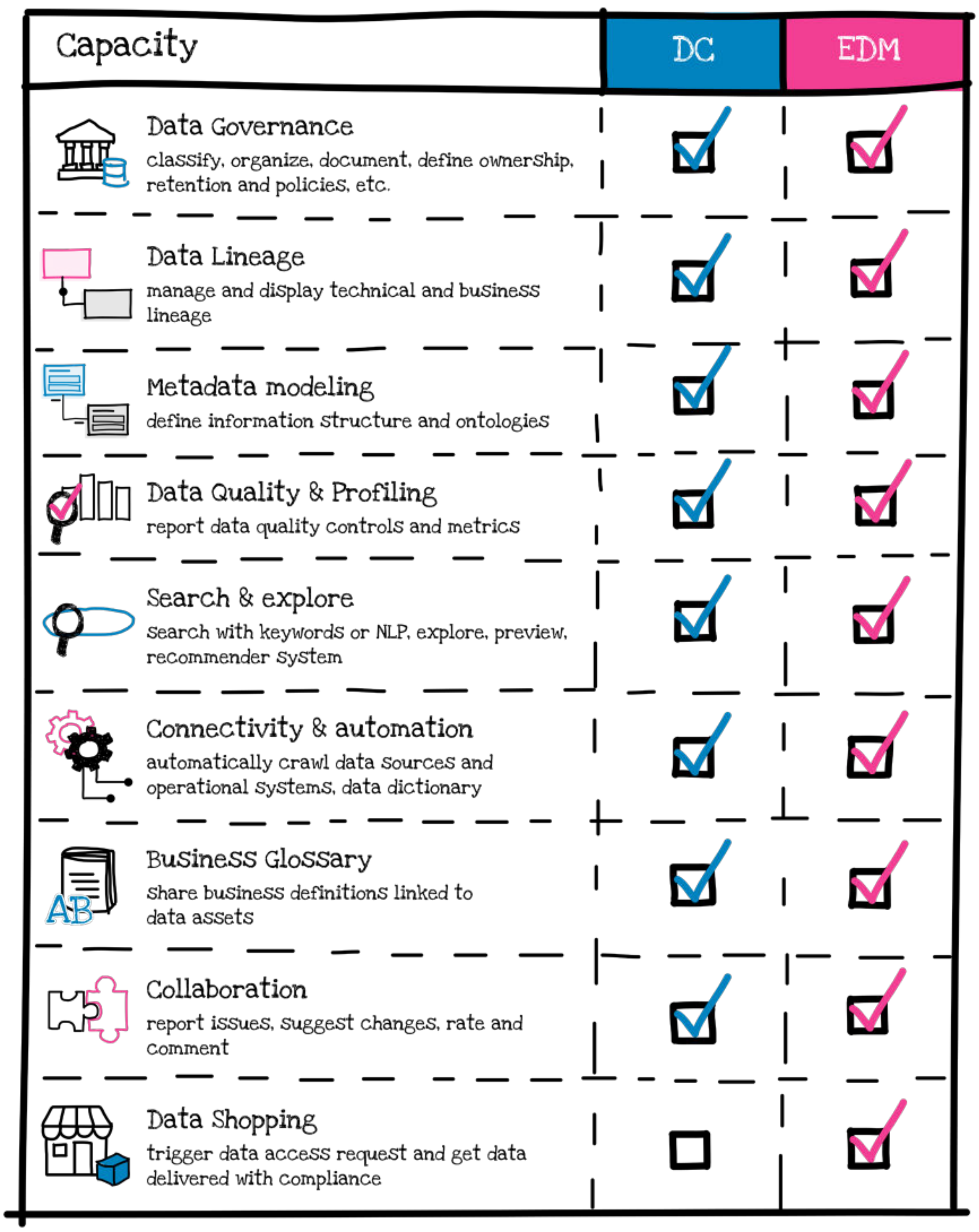


Domain catalogs must be integrated with the marketplace to avoid duplicating efforts in producing certain metadata - especially lineage, but also data dictionaries (schema), or even business definitions that will be present in both systems.



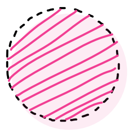
Domain catalogs potentially need to be integrated with each other - to share/synchronize certain information, primarily the business glossary but also some repositories.

When we look at the respective capabilities of an Enterprise Data Marketplace and a Data Catalog, we realize that these capabilities are very similar:

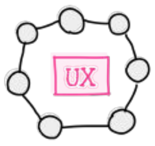


Capacity	DC	EDM
Data Governance classify, organize, document, define ownership, retention and policies, etc.	✓	✓
Data Lineage manage and display technical and business lineage	✓	✓
Metadata modeling define information structure and ontologies	✓	✓
Data Quality & Profiling report data quality controls and metrics	✓	✓
Search & explore search with keywords or NLP, explore, preview, recommender system	✓	✓
Connectivity & automation automatically crawl data sources and operational systems, data dictionary	✓	✓
Business Glossary share business definitions linked to data assets	✓	✓
Collaboration report issues, suggest changes, rate and comment	✓	✓
Data Shopping trigger data access request and get data delivered with compliance	☐	✓

In the end, on a strictly functional level, their capabilities are very similar. What distinguishes a modern Data Catalog from an EDM are:



Their scope - The Data Catalog is intended to cover all data, whereas the marketplace is limited to the objects shared by domains (data products and other domain analytics products).



Their user experience - The Data Catalog is often a fairly complex tool, designed to support governance processes globally - it focuses on data stewardship workflows. The marketplace, on the other hand, typically offers very simple ergonomics, heavily inspired by that of an e-commerce platform, and provides an experience centered on consumption - data shopping.

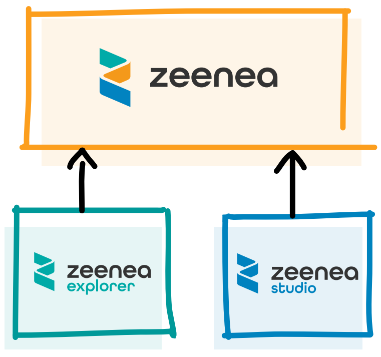
CHAPTER 3

The Data Mesh Supervision System: Zeenea's Solution

Recognizing the striking similarity between the Data Catalog and the Enterprise Data Marketplace in terms of functionality, Zeenea decided to evolve its data discovery platform to offer a unique system - one that covers both the need for a cross-domain marketplace and the need for a private data catalog for each domain.

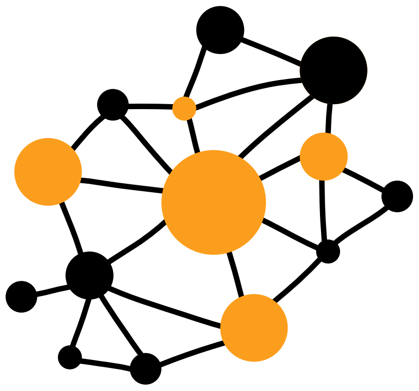
Technology and architecture of Zeenea Data Discovery Platform

Before delving into the solution's structure, let's review some of Zeenea's characteristics that make it a unique platform to support this duality:



Zeenea is designed as a marketplace - the platform already offers two distinct user experiences: *Zeenea Studio* serves as the back-office interface, enabling the definition of the metamodel, automation of data inventory, metadata feeding, and support for data producers' activities; *Zeenea Explorer* is a separate application that provides a search-focused experience and catalog exploration, inspired by leading e-commerce sites.

Zeenea relies on a customizable knowledge graph - allowing it to eschew reliance on a single hierarchical structure. The knowledge graph seamlessly accommodates multiple classification schemes, enabling different domains to organize their assets differently.

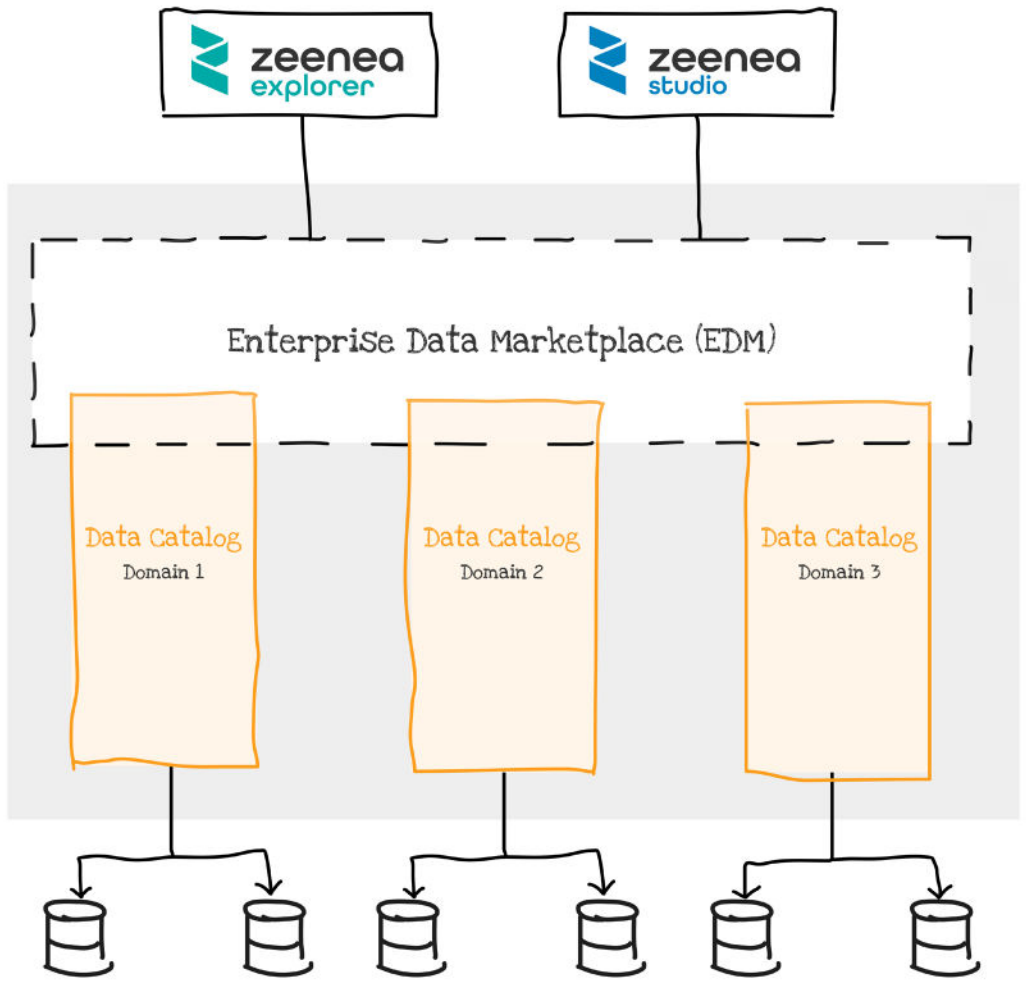


Zeenea already offers the capabilities of a modern Data Catalog - including a dictionary, governance, lineage, glossary, search, connectivity, automation, quality, etc.

To support marketplace usage, we have simply added a management layer that allows each domain to have a private catalog and to choose the objects they want to share with the other domains.

The marketplace becomes a subset of the graph, containing not only data products but also objects intended for sharing - business definitions, dashboards, ML models, etc.

The solution’s architecture is as follows:



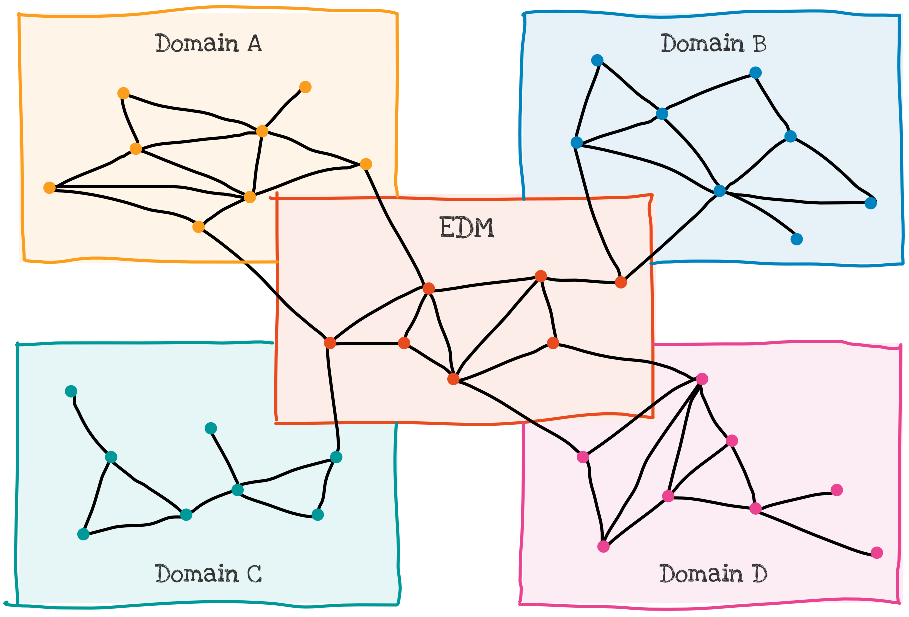
Zeenea's architecture

Each domain has its private space, where it can potentially:

- ✓ **Organize data as desired -** by defining a metamodel specific to it. Some elements of the metamodel can, of course, be shared or enforced by governance rules.
- ✓ **Integrate and ensure the feeding of its catalog -** from the data sources it owns.
- ✓ **Manage its users and their permissions.**
- ✓ **Identify the objects it wants to share with other domains -** and control which information will be shared.

The level of delegation is highly configurable, allowing for more autonomy in mature domains while maintaining tighter control over less mature domains - the system's topology can be modified at any time to reflect the progression of the data mesh in the organization.

Viewed logically, the information structure is as follows:



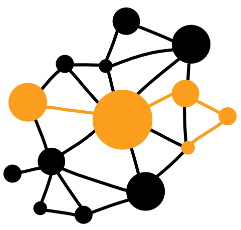
Private Data Catalogs per domains & marketplace (EDM): Information structure

The Zeenea federated graph: mirroring the mesh at the metadata level

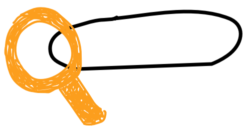
This structure is made possible by what has long distinguished Zeenea and sets it apart from most other catalog or metadata management solutions: an evolving knowledge graph.

A knowledge graph is a data structure that adheres to a set of rules defined in an ontology. The ontology describes the types of objects that the graph can contain, their attributes, and the relationships that these different types of objects can have with each other - thus defining a semantic universe.

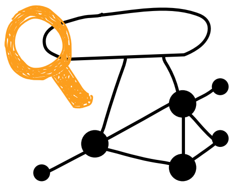
Knowledge graphs have unique properties for organizing and searching data:



They allow for exploring the knowledge universe starting from any node, then traversing the relationships to navigate the graph through the lens of each user - providing an experience vastly different from traversing a rigid and uniform hierarchical structure.



They form the basis of modern search engines - which can leverage not only the attributes of objects but also their relationships in the graph to rank them in the list of results.



They support an associative exploration model - which dominates the web world and especially classical marketplaces: users conduct a search, then navigate the links between the search results and other associated objects.

In Zeenea's platform, each domain is responsible for a subset of the graph, which it can model according to its needs and evolve at its own pace.

This ability to give autonomy to domains in how they organize their catalog seems fundamental to us - too often, the quest for a unique catalog structure leads to excessive complexity in the hierarchical organization, making it incomprehensible and unusable. This is the shared drive syndrome, where the hierarchy gradually becomes so complex that it becomes very difficult to find anything, except the documents one has classified oneself.

With this federated graph structure, Zeenea's platform mirrors the mesh at the metadata level - a mirror that can be continually adapted to reflect the evolution of the data mesh's topology and the architecture of the data platform as they deploy throughout the organization.

Structured as an evolving federation of interconnected graphs, the Zeenea platform thus provides the ideal foundation for building the data mesh supervision system and supporting its long-term deployment. In this structure, the marketplace is a simple subset of the graph, containing objects shared by the domains - especially data products.

In a data marketplace, searching and discovering data products are fundamental. But they are only the first step in the user journey - the subsequent steps being, of course, ordering and delivering the data to the consumer.

The shopping experience in Zeenea's marketplace

All classic marketplaces offer a very similar "checkout" experience, familiar to many people. The selected products are placed in a cart, then, when validating the cart, the buyer is presented with various delivery and payment options. The actual delivery is usually done outside the marketplace, which simply provides tracking functionalities. Delivery can be immediate (for digital products) or deferred (for physical products). Some marketplaces have their own logistics system, but most of the time, delivery is the responsibility of the seller. The delivery time is an important element of customer satisfaction - the shorter it is, the more satisfied users are.

How does this shopping experience translate into an Enterprise Data Marketplace? To answer this question, we need to consider what data delivery means in a business context and, for that, focus on the data consumer.

Delivery of Data Products

As we have seen previously, **a data product offers one or more consumption protocols - these are its *outbound ports.*** These protocols may vary from one data product to another, depending on the nature of the data - real-time data, for example, may offer a streaming protocol, while more static data may offer an SQL interface (and instructions for using this interface from various programming languages or in-house visualization tools). For interactive consumption needs, such as in an application, the data product may also offer consumption APIs, which in turn may adhere to a standard (REST, GraphQL, OData, etc.). Or simply download the data in a file format.

Some consumers may integrate the data product into their own pipelines to build other data products or higher-level uses. Others may simply consume the data once, for example, to train an ML model. It is up to them to choose the protocol best suited to their use case.

Whatever protocols are chosen, they all have one essential characteristic: they are secure. This is one of the universal rules of governance - access to data must be controlled, and access rights supervised.

With few exceptions, the act of purchase therefore simply involves gaining access to the data via one of the consumption protocols.

Access Rights Management for Data Products

However, in the world of data, access management is not a simple matter, and for one elementary reason: consuming data is a risky act. Some data products can be desensitized - somehow removing personal or sensitive data that pose the greatest risk. But this desensitization cannot be applied to the entire product portfolio: otherwise, the organization forfeits the opportunity to leverage data that is nonetheless highly valuable (such as sensitive financial or HR data, commercial data, market data, customer personal data, etc.). In one way or another, access control is therefore a critical activity for the development and widespread adoption of the data mesh.

In the logic of decentralization of the data mesh, risk assessment and granting access tokens should be carried out by the owner of the data product, who ensures its governance and compliance. This involves not only approving the access request but also determining any data transformations needed to conform to a particular use. This activity is known as *policy enforcement.*

Evaluating an access request involves analyzing three dimensions:

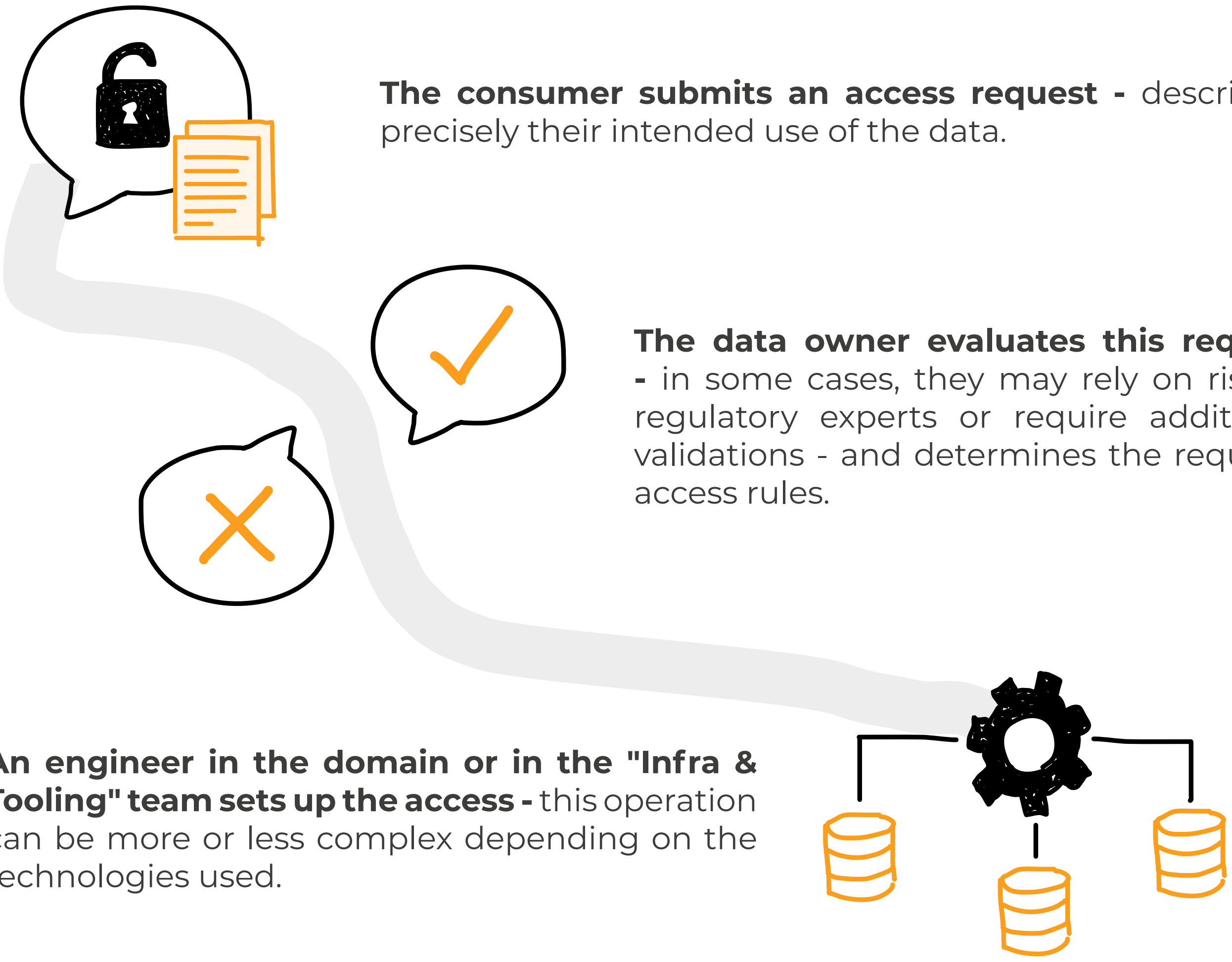
- The data themselves (some carry more risk than others) - the **what.**
- The requester, their role, and their location (geographical aspects can have a strong impact, especially at the regulatory level) - the **who.**
- The purpose - the **why.**

Based on this analysis, the data may be consumed as is, or they may require transformation before delivery (data filtering, especially for data not covered by consent, anonymization of certain columns, obfuscation of others, etc.). Sometimes, additional formalities may need to be completed - for example, joining a redistribution contract for data acquired from a third party, or complying with retention and right-to-forget policies, etc.

Technically, data delivery can take various forms depending on the technologies and protocols used to expose them.

For less sensitive data, simply granting read-only access may suffice - this involves simply declaring an additional user. For sensitive data, fine-grained permission control is necessary, at the column and row levels. Most modern data platforms support native mechanisms to apply complex access rules through simple configuration - usually using data tags and a policy enforcement engine. Setting up access rights involves creating the appropriate policy or integrating a new consumer into an existing policy. For older technologies that do not support sufficiently granular access control, it may be necessary to create a specific pipeline to transform the data to ensure compliance, store them in a dedicated space, and grant the consumer access to that space. This is, of course, a lengthy and potentially costly approach, which can be optimized by migrating to a data platform supporting a more granular security model or by investing in a third-party policy enforcement solution that supports the existing platform.

In the end, in a data marketplace, data delivery, which is at the heart of the consumer experience, translates into a more or less complex workflow, but its main stages are as follows:



The consumer submits an access request - describing precisely their intended use of the data.

The data owner evaluates this request - in some cases, they may rely on risk or regulatory experts or require additional validations - and determines the required access rules.

An engineer in the domain or in the "Infra & Tooling" team sets up the access - this operation can be more or less complex depending on the technologies used.

Data Product Shopping

Shopping for the consumer involves triggering this workflow from the marketplace.

For Zeenea's marketplace, we have chosen not to integrate this workflow directly into the solution but rather to interface with external solutions.

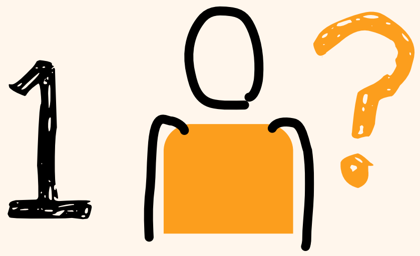
The idea is to offer a uniform experience for triggering an access request but to accept that the processing of this request may be very different from one environment to another, or even from one domain to another within the same organization. Again, this principle is inherited from classical marketplaces. Most offer a unique experience for making a purchase but connect to other systems for the operational implementation of delivery - the modalities of which can vary widely depending on the product and the seller.

This decoupling between the shopping experience and the operational implementation of delivery seems essential to us for several reasons.

The main reason is the extreme variability of the processes involved. Some organizations already have operational workflows, relying on a larger solution (data access requests are integrated into a general access request process, supported, for example, by a ticketing tool such as ServiceNow or Jira). Others have dedicated solutions supporting a high level of automation but whose deployment is not yet widespread. Still, others rely on the capabilities of their data platform, and some even on nothing at all - access is obtained through direct requests to the data owner, who handles them without a formal process. This variability is evident from one organization to another but also within the same organization - structurally, when different domains use different technologies, or temporally, when the organization decides to invest in a more efficient or secure system and must gradually migrate access management to this new system.

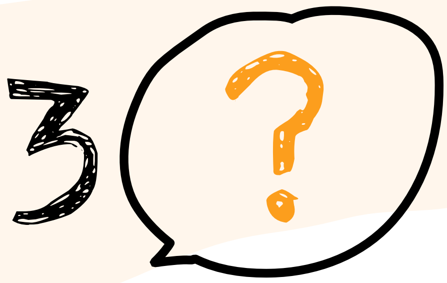
Decoupling, therefore, allows offering a consistent experience to the consumer while adapting to the variability of operational methods.

For a data marketplace customer, the shopping experience is very simple. Once the data product(s) of interest is identified, they trigger an access request by providing the following information:



Who they are - this information is already available.

Which data product they want to access - this information is also already available, along with the metadata needed for decision-making.



What they intend to use the data for - this is crucial since it drives risk management and compliance requirements.

With Zeenea, once the access request is submitted, it is processed in another system, and its status can be tracked from the marketplace - this is the direct equivalent of order tracking found on e-commerce sites.

From the consumer's perspective, the data marketplace provides a catalog of data products (and other digital products) and a simple, universal system for gaining access to these products.

For the producer, the marketplace plays a fundamental role in managing their product portfolio.

As mentioned earlier, in addition to the e-commerce system, which is intended for consumers, a classical marketplace also offers tools dedicated to sellers, allowing them to supervise their products, respond to buyer inquiries, and monitor the economic performance of their offerings. And other tools, intended for marketplace managers, to analyze the overall performance of products and sellers.

Zeenea's Enterprise Data Marketplace integrates these capabilities into a dedicated back-office tool, Zeenea Studio. It allows for managing the production, consolidation, and organization of metadata in a private catalog and deciding which objects will be placed in the marketplace - which is a searchable space accessible to the widest audience.

These activities primarily fall under the production process - metadata are produced and organized together with the data products. But it also allows for monitoring the use of each data product, notably by providing a list of all its consumers and the uses associated with them.

This consumer tracking helps establish the two pillars of data mesh governance:

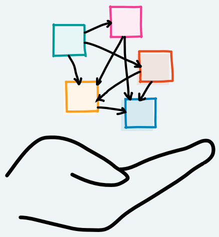
Compliance and risk management - by conducting regular reviews, certifications, and impact analyses during data product changes.

Performance management - the number of consumers, as well as the nature of the uses made of them, are the main indicators of a data product's value. Indeed, a data product that is not consumed has no value.

As a support tool for domains to control the compliance of their products and their performance, Zeenea's Enterprise Data Marketplace also offers comprehensive analysis capabilities of the mesh - lineage of data products, scoring and evaluation of their performance, control of overall compliance and risks, regulatory reporting elements, etc. This is the magic of the federated graph, which allows exploiting information at all scales and provides a comprehensive representation of the entire data asset.

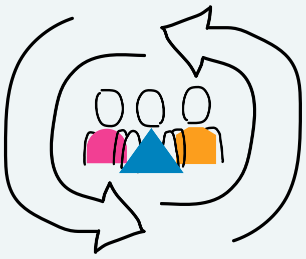
CONCLUSION

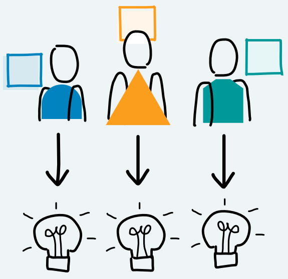
Key Takeaways from this Ebook



Data mesh is no longer just an intellectual curiosity but a mainstream practice undergoing massive adoption.

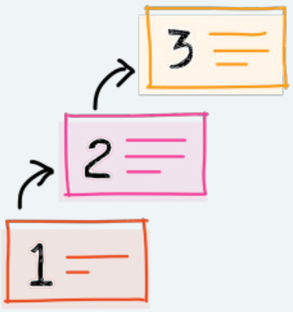
Implementing data mesh is primarily an organizational and cultural transformation - it does not require immediate investments in infrastructure.





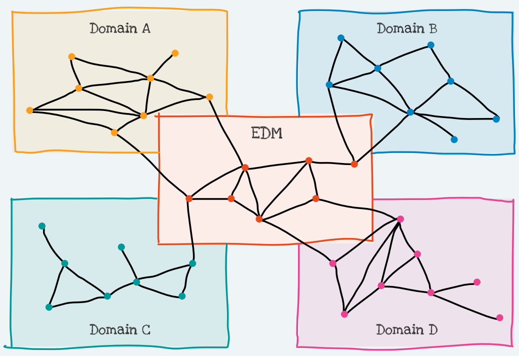
There is no canonical architecture for data mesh - each organization adopts solutions that are unique to them, and these solutions will evolve over time.

The implementation approach should be gradual and use Lean Manufacturing tools to properly focus efforts and investments by identifying real bottlenecks and addressing them.



To ensure the scalability of data mesh, it is necessary to have a comprehensive supervision system that allows:

- ✓ Connecting data producers and consumers;
- ✓ Providing consumers with marketplace capabilities to order data (shopping experience);
- ✓ Providing producers with the means to control and evaluate the performance of their products and manage their private data assets;
- ✓ Providing tools for global analysis of the mesh for risk management and strategic prioritization.



With its Enterprise Data Marketplace offering powered by a federated graph, Zeenea mirrors the data mesh at the metadata level - and builds a comprehensive and scalable supervision system perfectly integrated into data production and consumption processes.

zeenea

Want to find out more about Zeenea?

Contact us now for a personalized demonstration

GET A DEMO

www.zeenea.com - info@zeenea.com

Follow us on