Open in app ↗ Sign up Sign In

Published in Adevinta Tech Blog

Adevinta Follow

Jun 17, 2021 · 10 min read · Listen

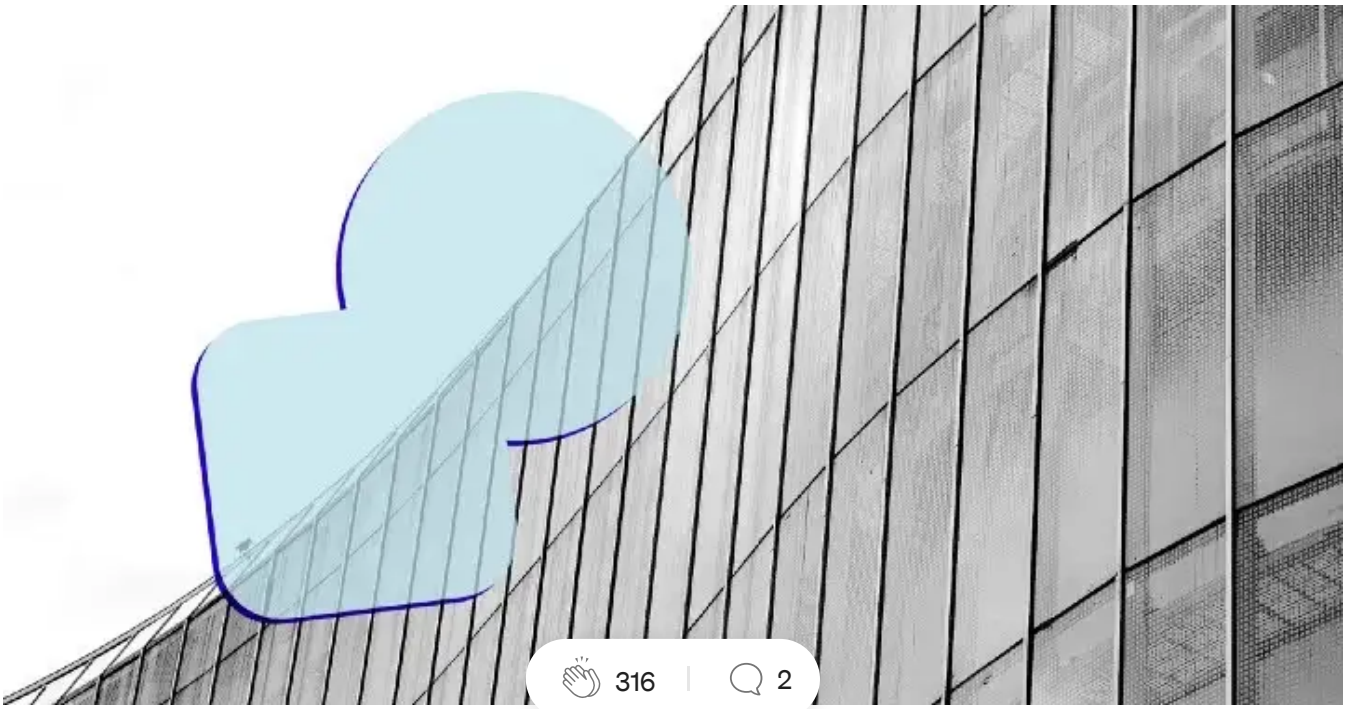
Save

Building a data mesh to support an ecosystem of data products at Adevinta

What worked and didn't work in our journey towards a data mesh

By Xavier Gumara Rigol, Senior Engineering Manager for Experimentation and Analytics Solutions and Sandra Real, Product Manager for Analytics Solutions at Adevinta.

In this article, we'll explain our journey towards a data mesh architecture. It began in 2018 (prior to the data mesh concept), when our Analytics Solutions team received the mandate to enable convenient access to our marketplaces' data…



316 2

The Central Product and Tech department (P&T) in Adevinta provides a wide range of data products that our marketplaces can leverage (amongst other services). Some examples include an ads moderation service, recommender systems, image recognition algorithms and chat services.

In order to measure the impact these products have on our end-users and make the best possible decisions, we use data. It is therefore crucial to capture, store, clean and make data easily available to decision makers in the company: Data Analysts, Data Scientists and Product Managers.

To do so, we've embraced the concept of a data mesh as proposed by Zhamak Dehghani: a federated model for self-service management of the data from our Central P&T components.

What we got right

We spent quarters steadily delivering incremental value to the organisation. But it was only once we were able to provide domain-specific curated datasets for exploratory analysis and reporting (aka tidy data or Business Intelligence), that we knew we had succeeded. We now provide analytical datasets considered by the organisation as "the Analysts' paradise" and overall the initiative has had a positive return on investment, as measured by time saved by decision makers (from Data Analysts to Senior Management).

What we're sharing in the sections below are the success factors for this initiative.

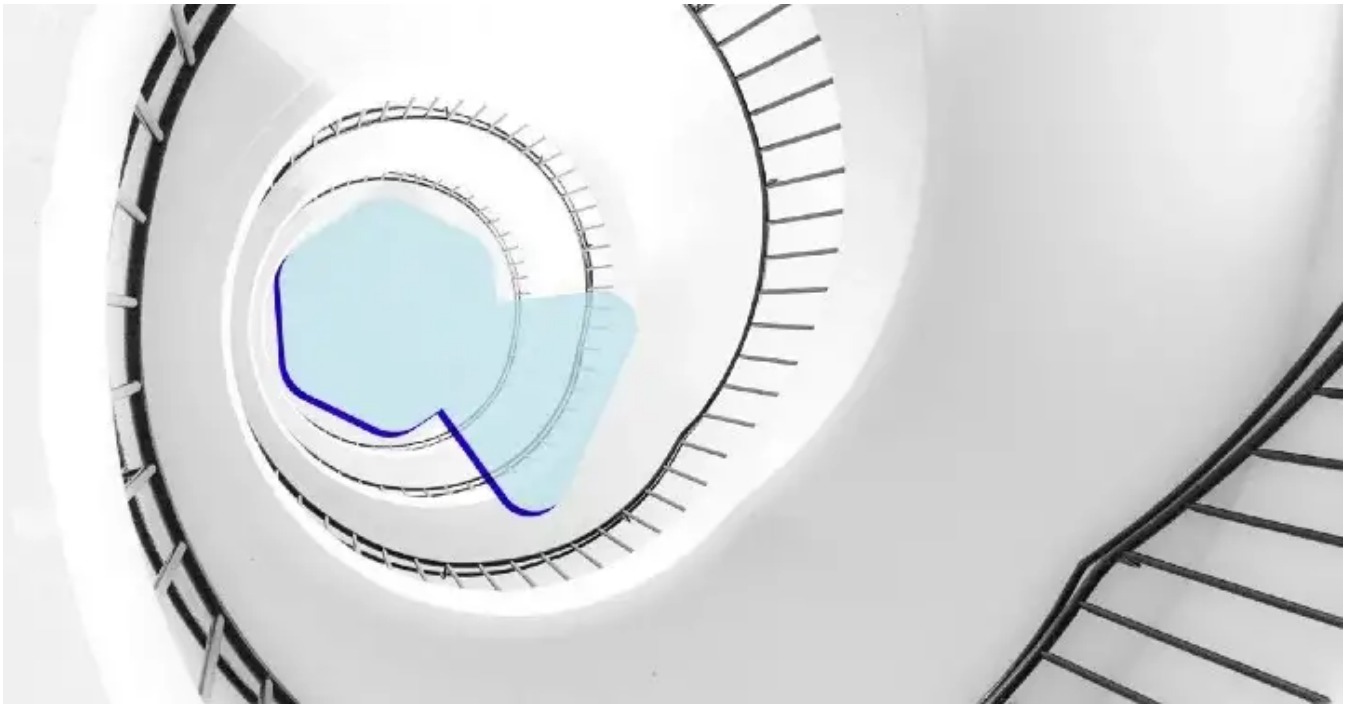
SQL access is a must

For Data Analysts, having data at their fingertips must be fast and easy, and it should take just a few seconds to run an SQL query. That's why we wanted to make every dataset available via SQL, like in traditional Data Warehousing.

For behavioural (clickstream) data and components that generate a lot of data, the traditional Data Warehousing approach doesn't work. This is due to the huge amount of data that we manage due to our scale. In the past, our folks had to use a notebook every time they wanted to access any type of data.

Providing an SQL layer on top of these big data datasets has accelerated decision making at Adevinta because Data Analysts can now explore more granular data to

answer specific questions. You can read more about this in a previous article from our blog.



Datasets as products

Understanding the context of a dataset or knowing how to access it is a challenge that becomes more painful as the organisation and the amount of data generated grows. It was crucial for us to think about which features our datasets should include, apart from the data processing aspects. Our datasets transitioned from being simply a path in a file system to becoming the data products our analysts love.

Knowing how your library of datasets is sorted is crucial for efficiency (Source: https://unsplash.com/photos/VfHoMBagDPc)

We identified the following fundamental properties that every single dataset should have:

- A description including the domain it belongs to, which type of data it contains and the granularity of the observations.
- How and where we can access this data. We determined that each dataset should have at least two interfaces: SQL as a fast access and programmatic access via notebooks, in case more complex processing is needed.
- The schema of the dataset, including field types and a detailed example for each field, to understand the data as you read the schema.
- Mappings and transformations used to generate the dataset. Making data meaningful means that some transformations will be applied (using enrichments or mappings). It was very important to us that the consumers understand how the data they're using is generated.
- Input preconditions. Datasets as data products have an input and output but what are the conditions for the dataset to be generated? What's the source? How

often is it generated?

- Data exploration examples. Each dataset includes a notebook with some simple processing examples and a query catalog with SQL query examples using the dataset.

All of these properties are exposed as part of our catalogue of datasets and are part of the acceptance criteria we use to consider whether a dataset is ready to be put into production. This allows us to say that our documentation is always up to date.

In retrospect and without realising it, these fundamental properties allowed us to fulfill the discoverability, addressability and self-describing aspects of "data as a product", as explained by Dehghani.

As some of us come from a traditional Data Warehousing environment, we took into consideration the interoperability between datasets and applied standard nomenclature for fields, tables and partitions. We also introduced some common rules and conventions such as: columns from different datasets that contain the same data should be named the same.

Finally, for the trustworthiness and security aspects, we leveraged our common Data Highway infrastructure. This facilitated data quality monitoring and authorisation tooling which allowed us to access data according to privacy regulations and company and team requirements.

Domain-oriented data

We define 'domains' as a field of influence and a 'domain team' as a team responsible for the products and services within that domain. In this context, it makes sense that our Messaging team owns the messaging domain and so on.

Similarly, we distinguish between 'domain datasets' and 'core datasets'. The domain datasets belong to a specific domain and the core datasets are those that are useful for more than one domain.This last sentence is important because it also helped us prioritise. We started modeling the core business domain and made the datasets that were going to be useful for more people available first.

Once we had a good coverage of questions that could be answered thanks to these core datasets, we started implementing the domain specific datasets, domain by domain.

Centralised implementation and governance

At the beginning, the domain-specific datasets were built by our Central Analytics Solutions team. This might seem counterintuitive based on the principles of the data mesh, but the data mesh is a complex system and quoting John Gall: *"A complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a working simple system"*.

Central governance and a talented team of data engineers were two of the key factors in the development of domain-specific analytics datasets; it is our *simpler system that worked*.

On top of that, the team needed to develop standardised transformation libraries, abstract read and write logic, build aggregations frameworks and so on in order not to build duplicated code or end up having a "pipeline jungle", as commented in Machine Learning: The High Interest Credit Card of Technical Debt.

Save pipeline jungles for your holidays, not your data product! (Source: Xavier Gumara Rigol)

The purpose was always to create datasets that were usable and had a significant impact in the understanding of analytics data across Adevinta.

What got us here won't get us there

Shortly after, our Analytics Solutions team faced a bottleneck: the amount of requests and the need to work on reusable components and libraries to ease our workload impeded our ability to work on new stuff at the velocity the business required.

Moreover, the team started to become too specialised and siloed from the business.

As written by Dehghani, the data mesh paradigm surfaces as a response to three failure modes of traditional data architectures. The first and third prevented us from scaling our initial successes:

- Centralised and monolithic
- Coupled pipeline decomposition
- Siloed and hyper-specialised ownership

As more data became available, the ability to harmonise it under the control of one team diminished and the long response time became a point of friction.

Moreover, as the Analytics Solutions team is a group of hyper-specialised Data Engineers not focused on where the data originates or where it is used, it makes it harder to have autonomy over the creation of domain-specific datasets.

In early 2020, we started having conversations with the business in order to prepare for a change in our operational model. We wanted to leverage the expertise of domain teams in order to accelerate the development of analytical datasets and better measure the performance of our central data products.

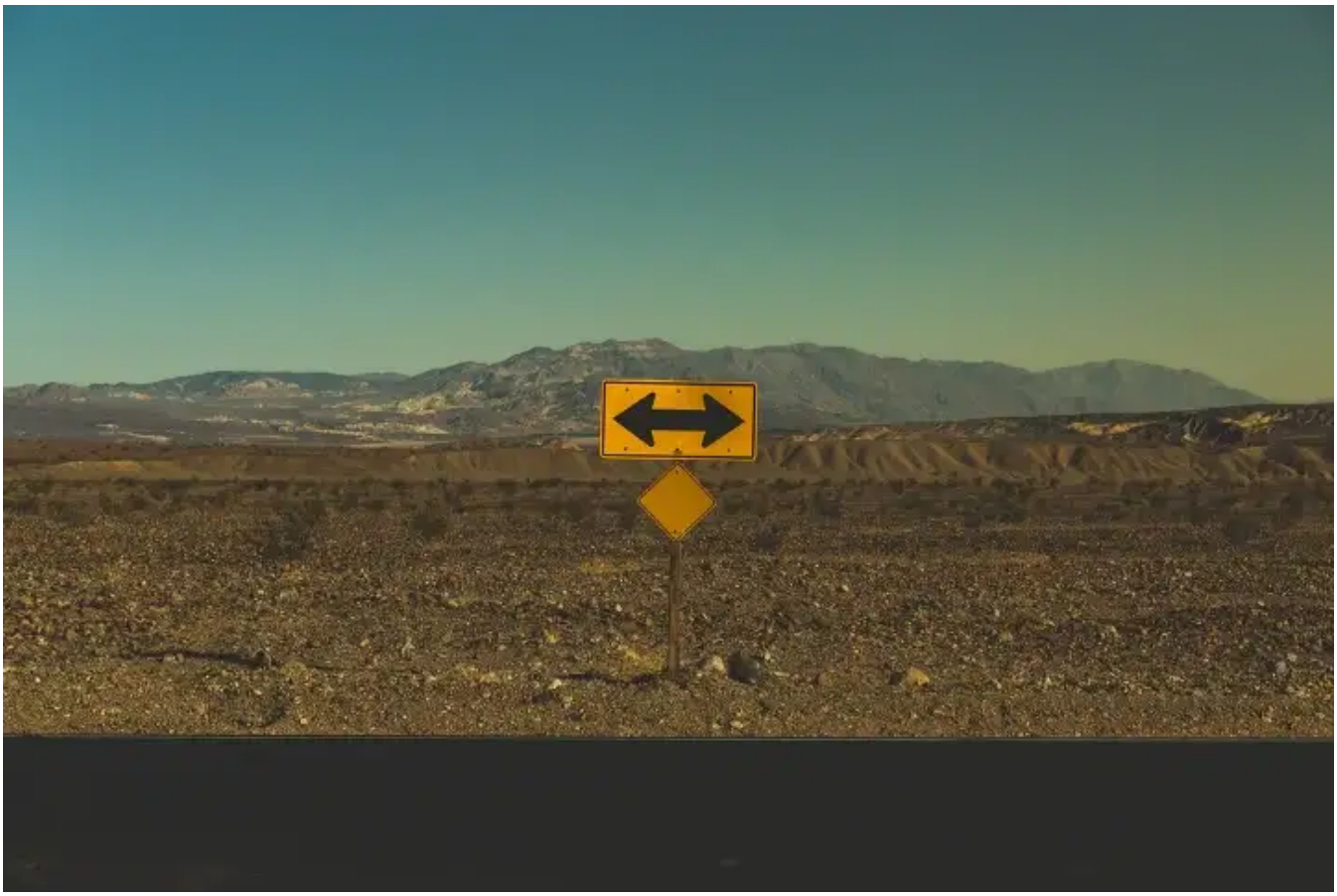
It wasn't easy. Moving towards a more decentralised ownership was a cultural challenge for our product teams. It took time to help people realise that this change was in everyone's best interests. Some of the things that helped with the transition included:

- Communicating that no disruption of the service was going to be made
- Offering all possible support and pairing sessions with Engineers on the domain teams so that they understood the system and its challenges
- Not handing over anything until the receiving team was ready

In order to move to the next stage, we needed to hire Data Engineers in the domain teams so they could have full ownership of their datasets and be responsible for their quality. The Data Engineers were able to work autonomously and didn't need to invest data engineering resources to (re)build tools.

On the other side, our Analytics Solutions team retained ownership of the core datasets, ensuring their trustworthiness, and kept building and maintaining specialised tools and libraries to enable analysts to create derived datasets.

Apart from initialising organisational changes and providing the technology to enable it, we also needed to put in place new processes for the system to work. In a data mesh world, how do you maintain the quality of datasets? How do you make sure the decentralised teams embrace the concept of datasets as products and don't just create datasets as a byproduct for other goals?



What got us here won't get us there (Source: https://unsplash.com/photos/lPQlndZz8Mo)

What worked in our case was to set some working agreements.

We started with the main principles of the paradigm:

1. Any data element that is specific to a domain should be created and maintained by the domain team; it is the responsibility of this team to own it and ensure its quality.
2. Any data element that is useful to more than one domain (core data elements) should be governed, created and maintained by the Analytics Solutions team in order to preserve interoperability and trustworthiness.
3. Data quality issues should be fixed at the source or as close to the source as possible.
4. Core and domain datasets should provide enough stability to build analytics products on top.
5. Core and domain datasets must be GDPR-compliant.

Followed by these working agreements:

1. The Analytics Solutions team owns the governance of core and domain data elements including: nomenclature, paths, documentation format and definitions as defined in the conventions [we're not including them here because they're very solution-specific]. Some processes are also available to help set this governance: joined requirement discovery, weekly syncs, integrated documentation.
2. The Analytics Solutions team owns and offers tooling for Data Engineers to orchestrate, build and backfill core and domain datasets. This tooling can be used by domain teams as a service.
3. If some data elements are duplicated in domain datasets, a process to add them to the core will be created and the duplications removed from the domain-specific datasets.
4. Pipelines to build core datasets run in the same cluster and the Analytics Solutions team is responsible for maintaining the pipelines that generate the core datasets.
5. Pipelines to build domain datasets run in specific domain teams' clusters with no access for Analytics Solutions team members (except for temporary joint

development phases). Domain teams are responsible for the maintenance of domain pipelines.

6. The Analytics Solutions team's efforts will be allocated on a quarterly basis with some bandwidth for ad hoc requests during the quarter.
7. A core data element can only be deprecated at the end of the following quarter when the deprecation is announced.

If a domain team agrees to these principles and working agreements, the Analytics Solutions team will support their requests. But if the domain specific team shows low alignment on these governance rules, they're left out of the scope of the umbrella of curated datasets as products.

Not being aligned on these principles is in direct relation to the fact that the work produced by the non-aligned team is non-discoverable, as if there is a lack of documentation, it's impossible to check the trustworthiness of the data, etc.

Final words

At Adevinta, we didn't wake up one day and decide to build a data mesh. The data mesh is the natural evolution of our data architecture, based on pain points we experienced in the past.

Evolving from a centralised approach to data for analytics to a data mesh where datasets are considered as products and every business domain is responsible for their own datasets was the right approach in order to stay one step ahead. It is our particular Red Queen's race.

However we're willing and curious to learn from companies that started their data architecture efforts from scratch with the data mash paradigm in mind. Feel free to reach out to us if you want to share your learnings or want to know more about our journey.

Data Mesh Data Architecture Data Engineering Data Driven Business

Data And Machine Learning

About Help Terms Privacy

Get the Medium app