POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Data Mesh: a new paradigm shift to derive data-driven value

Supervisor

Prof. DANIELE APILETTI

Candidate

MICHELE DICATALDO

Academic year 2022/2023

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 About this work

The aim of the presented work, is to propose a first implementation of a Data Mesh Data architecture solution, for an energetic company to manage different kinds of data extracted from multiple data sources. With this work, all the main notions of Data Mesh architecture and the related advantages in terms of business intelligence are fully exploited, such as the adopted technologies for the ingestion and transformations processes. In particular, the various provisioning services and its resources are described: the usage of Python as programming language, Spark as engine, ADF (Azure Data Factory) and its resources as ETL and data ingestion platform. It will also be explained the motivations which led to a Data Mesh architecture, and presented the implemented use cases. "Cluster Reply" is the host company that has allowed the study of this topic and it's one of the first company to approach Data Mesh in a concrete way.

1.2 About the company

Cluster Reply is the company of the Reply group, specialized in consultancy services and system integration on Microsoft technologies in Italy. It proposes challenging projects in the fields of Artificial Intelligence Machine Learning, Data Analytics, Hyperautomation, Cloud Computing, DevOps, modern applications with Cloud Native architectures, Modern Digital Workplace and Business Applications. Those works, are related to all the sectors of Financial Services (Banking, Credit and Insurance), Manufacturing, Healthcare, Utilities, Telco Media, Services, with a particular orientation towards Business Applications, Applications and Azure Infrastructures , Data AI.
Data Mesh is the most recent data architecture proposed and leveraged by this company: it allows the transition to *data-driven* culture and enables corporate agility and scalability, reduces time to market, lowers maintenance costs, all through a decentralized structure composed by different nodes (Data Product) with the aim of extracting large-scale analytical data.

Chapter 2

Big Data

In this chapter what is Big Data and why it is at the core of this work, will be explained. Moreover it is shown also the motivation that led to an architectural shift for data management and storage.

2.1 What is Big Data: the 5 V's

Big Data is defined as large amounts of heterogeneous data from many sources and formats that can be examined in real time. The world is changing rapidly due to the emergence of innovational technologies, and consequently to the amount of data produced by the large number of devices used by individuals. These technologies and applications are useful for data analysis and storage [1]. In the early 2000s, we defined big data with 3 characteristics: volume, velocity and variety. In the last decade, while this concept was becoming more concrete in the companies, two more features were assigned to Big Data: veracity and variability. Let's now analyse these points:

1. Volume: we all in our everyday life generate a massive amount of data. Companies need to face this huge volume of data which cannot be managed

by traditional technologies: companies themselves produce data from multiple sources including business transactions, IoT devices, industrial equipment, video, social media and more.

2. Velocity: with the expansion of the Internet of Things, huge amount of data flows must be managed in a fast and unprecedented manner. Sensors collect data in real time, so the challenge for the companies is not just to collect them, but also analyse them rapidly, in order to take business decisions as quickly as possible.

3. Variety: with variety, we refer to the diversity of data types, nowadays available, coming from heterogeneous sources. Structured and unstructured data, not only acquired inside the company, but also outside.

4. Veracity: "*Bad data is worse than no data*". Data must be accurate and dependable. Data management technologies evolves, and with this the rate at which data is produced expands, so quality and integrity must be the pillars for conducting trustworthy analyses.

5. Variability: much more data, in different formats and coming from different contexts. Data scientists should work with domain experts, to take in account the variety of their significance over time.

Nowadays data are considered as "the new petrol" since they are source of big value. So we can talk about a new "V". By definition the data is a codified representation of an event, entity or a transaction, and the knowledge extracted from it usually has a meaning just for who work in that domain. So to allow the extraction of useful information from those data, to use in the company processes creating knowledge for the improvement of performances, Data Analytics tools are needed. That's why we need to consider the 6-th "V": *Value* [2].

2.2 Big Data and Business

Companies use big data in their systems to improve operations, provide better costumer services, create personalized marketing campaign and take actions which, ultimately, can increase revenue and profits. Businesses that use it effectively hold a potential competitive advantage over those that don't because they're able to make faster and more informed business decisions. Think about valuable insights that companies collect on big data related to their customers: they can be analyzed to assess the evolving preferences of consumers or corporate buyers, enabling businesses to become more responsive to customer wants and needs [3].
The benefits coming from the usage of big data in business are:

- Dialogue with consumers: with big data, you will get actionable data that you can use to engage with your customers one-on-one in real-time, checking infos on the product he/she is complaining about.
- Re-Develop products: Big Data is one of the best ways to collect and use feedback. The knowledge of what users think about your product, help in re-developing it.
- Perform risk analysis: Big data allow predictive analysis tasks, so scanning social-media feed and newspaper reports, helps in keeping up with speed on the latest trends in the industry.
- Data safety: Big Data tools allow the mapping of the data landscape across the company, monitoring eventual internal threats. So you can keep all sensitive informations protected in an appropriate manner and stored according regulatory requirements.
- Create new revenue streams: as said before, a proper Big Data analysis, and the inspection of the resulting insights, can lead companies to a competitive

advantage [4].

2.3 Architectures for Big Data

To handle the big data ingestion, processing and analysis, an appropriate architecture must be designed. The general logical components of a Big Data platform, are shown in the following figure: The data flow can have different behaviours

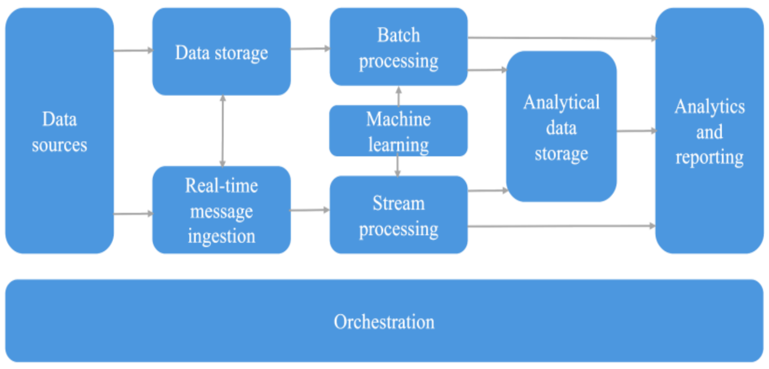


Figure 2.1: Big Data architecture

based on the type of workload (Batch or Stream). By the way generally the main components of a Big Data architecture are:

- Data Sources: is where data came from. They include application data stores (e.g. relational databases), static files produced by applications (e.g. log files), real-time data sources such as IoT devices.
- Data Storage: data for batch processing is typically stored in a distributed file system which can hold large volume of data, in different formats.
- Batch processing: this is a solution to face the fact that datasets are too large. It consists in the usage of long-running batch jobs to filter, aggregate and prepare data for analysis.

- Real-time message ingestion: when the process includes real-time sources, the architecture must provide a way to store the messages for stream processes. This might be simply a data store where incoming messages are dropped into a folder.
- Stream processing: after the data streams are captured, they need to be processed and then written on a sink. Here technologies like SQL queries, are perpetually run to manage unbounded streams of data.
- Analytical data storage: big data solutions contemplate the preparation of data, before serving them for analysis. Relational data stores like Data warehouses can be used to query this kind of data, or data can also be displayed through low-latency NoSQL technologies, which abstract metadata about data files in the distributed file system.
- Analysis and reporting: reporting tools are fundamental to deliver data insights. Data modeling layer (e.g. OLAP model), are often included to perform an appropriate data analysis.
- Orchestration: an orchestration technology is needed to deal with the continuous and repeated processes on big data: transform source data, move data between multiple sources and sinks, load the processed data into an analytical data store, or push the results straight to a report or dashboard [5].

2.3.1 Data Warehouse

The Data warehouse (DW) technology was developed to integrate heterogeneous information sources for analysis purpose. Information sources are autonomous and they can change their schema independently of the DW. Such changes must be supported in the DW, in fact this technology is seen as the process of good decision making since it provides necessary tools for data analysis such as the

On Line Analytical Processing (OLAP): it means that it can handle interactively huge amount of integrated data to process ad-hoc queries [6]. In other words, DW is the storage system which contains structured data, derived from the ETL (Extract-Transform-Load) process and meant to analytical processes. Structured data means that DW requires them to be represented by relationships that can be expressed using hard tables and diagrams.

2.3.2 Datalake

Datalakes can be defined as largely scalable repositories denoted as a significant mass of data, where it is stored in its "As-Is" form, remaining in that state until the need of processing systems, which ingest data without compromising the data structure. The data lakes are typically built to handle large and quickly arriving volumes of unstructured data (in contrast to data warehouses' highly structured data), thus they use not pre-built analytical applications. The architectural diversity by the DWs (in which folders anf files follow a hierarchical structure) , is that it does not require a rigid schema: elements are identified by a unique ID and a set of extended metadata tags, and require the maintenance of the order of data arrival [6].
Summarizing, the differences between the two architectures are mainly the kind of data they deal with (structured for DWs and non/semi-structured for datalakes), the storage costs (higher for DWs) and lastly their users: data warehouses are mainly used by professionals business users, while data lakes mainly target data scientists. Moreover data lakes theoretically scales to an infinite amount of raw data, and are considered to be more efficient and effective to deal with heavy workloads [7].

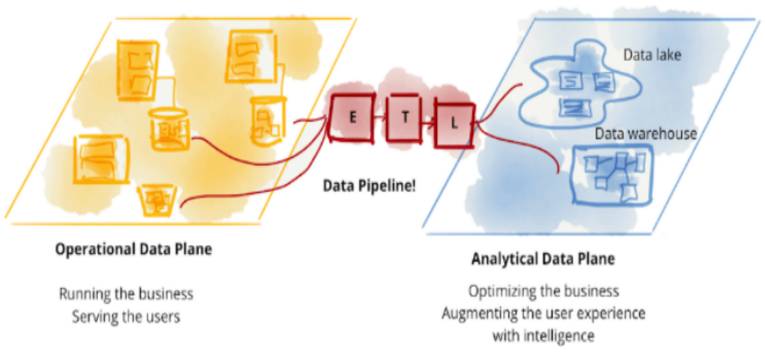


Figure 2.2: Latest generation architectures

In the following chapter we will understand why these approaches are no more enough, and the need of a paradigm shift to face the disproportionate enhancement of big data generation.

Chapter 3

Data Mesh

Data mesh is the nudge that puts us a new trajectory in how we approach data, how we imagine data, how we capture and share it, and how we create value from it, at scale and in the field of analytics and AI [8]. It can be summarized as a decentralized architecture, in which each node is a domain-driven data ownership that is treated as a product.

3.1 Why Data Mesh: the inflection point

Data Mesh is the result of an inflection point, where the current architectures are in a state of crisis, due to the incapability of responding to the scale, complexity, and data aspirations of company's business. The fact is that nowadays, building a data-oriented business is one of the top strategic goals of executives, so there are great expectations of data: companies want to provide the best customer experience based on data and personalization, reduce operational costs and time, empower employees to make better decisions through trends analysis and business intelligence (BI). To face these expectations, a new data management approach is needed, one which can seamlessly fulfill the diversity of the constantly growing uses for data, the misalignment between the the organizational and costumer needs, and the architecture instituted. The truth is that the current centralized approach (Data warehouse and Data lake), cannot easily deal with the continuous context changes in which data are applied, with data teams that usually have not prior knowledge of the data. So, Data Mesh arises as a decentralized paradigm that occurs at both technological and organizational level, with the purpose of solving the problems related to the loss of data nature, costs related the management of monolithic architectures and the significant pressure on data teams.

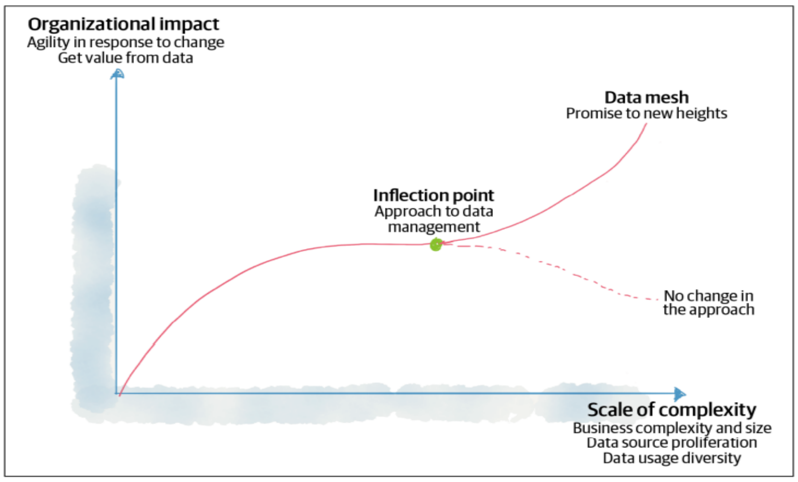


Figure 3.1: The inflection point in the analytical data management approach

3.2 Data Mesh Shifts

Data mesh introduces multidimensional technical and organizational shifts from earlier analytical data management processes:

- *Organizationally*: it shift the centralized ownership of data by specialists who run the data platform technologies, to a decentralized data ownership pushing accountability of data back to the business domains where data is produced from or is used.

- *Architecturally*: it shifts from collecting data in monolithic warehouses and data lakes, to a distributed mesh of data products accessible through standardized protocols.
- *Technologically*: it shifts from solutions where data are seen as by-product of running pipeline code, to solutions which treat data and code as one lively autonomous unit.
- *Operationally*: it shifts data governance from a top-down centralized model with human interventions, to a federated model with policies embedded in the node.
- *Principally*: it shifts our values from data as an asset to be collected, to data served as product to data users (externally or internally to the organization).
- *Infrastructurally*: it shifts from two sets of integrates services (one for analytics and one for applications and operational systems), to a well-integrated set of infrastructure for both operational and data systems.

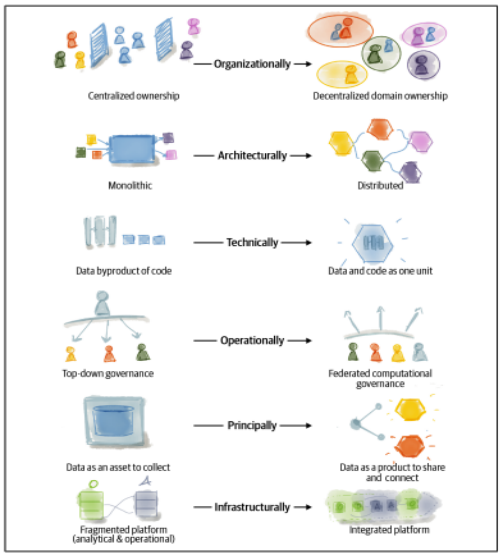

Centralized ownership
Organizationally
Decentralized domain ownership
Monolithic
Architecturally
Distributed
Data byproduct of code
Technically
Data and code as one unit
Top-down governance
Operationally
Federated computational governance
Data as an asset to collect
Principally
Data as a product to share and connect
Fragmented platform (analytical & operational)
Infrastructurally
Integrated platform

Figure 3.2: Data mesh shifts

3.3 The Principles

There are four principles that capture the idea behind data mesh's logical architecture and operating model. These principles have the objective of make us progress towards the increasing value from data at scale and sustain agility as an organization grows.

3.3.1 Principle of Domain Ownership

The first concept of data mesh relates to the organization of the data in line with the business itself, which is divided in areas of operations called domains. Data mesh postulates the existence of a distributed responsibility by organization's teams, that can better understand and produce data of their specific business domain. In this sense data ingestion obeys the nature of data and decentralize ownership. Then, we can classify three kind of domain data:

- Source-aligned analytical data, produced by teams which owned the operational systems;

- Aggregates, analytical data in the middle between producers and consumers, compositions of multiple data products. Aggregates can be owned by producer or consumer team;

- Consumer-aligned analytical data, designed for a particular data consumption scenario.

This kind of logical design is inspired by the so called Domain-driven design (DDD) [8].

3.3.2 Data as a Product

According to J. Majchrzak et al., a *"data product is an autonomous, read-optimized, standardized data unit containing at least one dataset (Domain Dataset), created for satisfying user needs"*. Each node of the data mesh is a data product, which lives in a context-bounded domain but the same domain can contains more data products. There are six principles that must be addressed to guarantee the quality

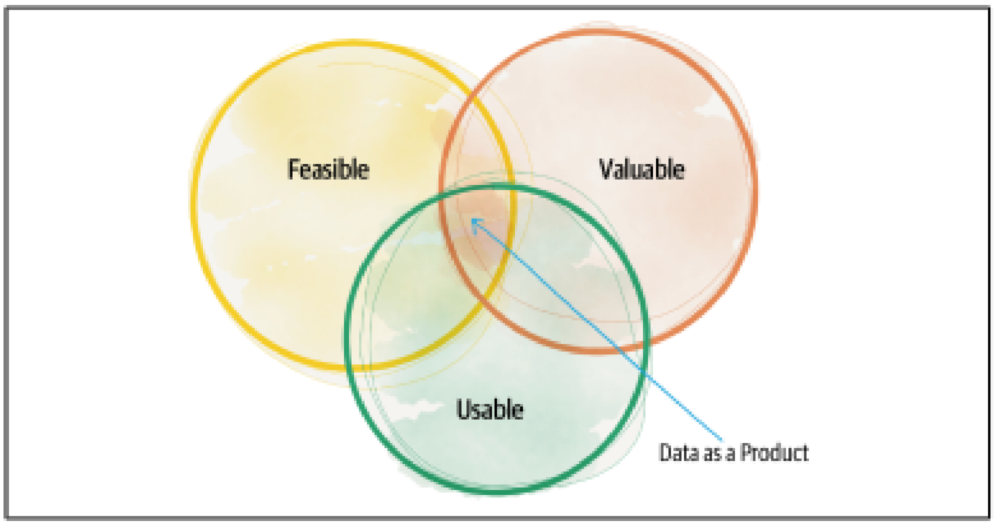


Figure 3.3: Characteristics of successful data product

and efficieny of data mesh, and those are (DATSIS principles): Discoverable, Addressable, Trustworthy, Self-describing, Interoperable and Secure [6]. Then, the concept of data product as *"product thinking"* is used to delight the experience of data users (e.g data scientists, data analysts, data explorers), and anyone in between. In short, it means that ensuring the data product meets a specific business need, and provide some tangible value, has a long-term time horizon with a well-defined owner that acts also in the costumer's interest. This new orientation, results in the appearance of two new roles: the data product owner and the data product developer. The first, is concerned with the satisfaction of the consumer, so it is responsible for the life-cycle, the decision-making process and must make his work measurable (through KPI). The data product developer instead, is responsible for building, maintaining and serving the data product. Moreover, they are not exclusive to one domain, and based on the company needs they can switch it. In this scenario, we can define the *architectural quantum* as data product: it is the smallest unit developable, including all its structural components [6].

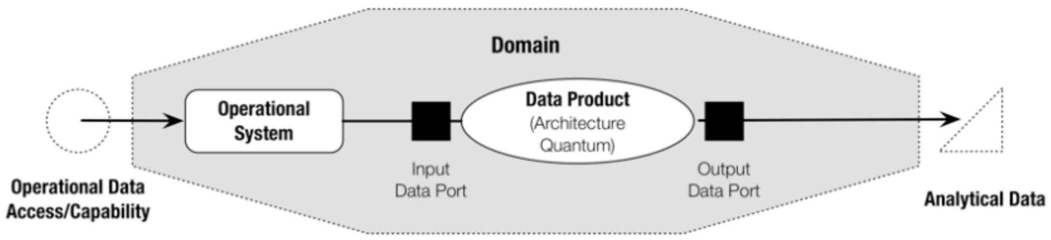


Figure 3.4: Architecture quantum in a domain

A data product includes three main components:

- Code: it includes the data pipeline, application to access data and metadata, and code for the policy access. The code get data from upstream via the input data, and provides interfaces to generate output data via polyglot output ports.

- Data and Metadata: data can be provided in different formats (events, batch files, relational tables...) but they need to maintain the same semantic. Data must be usable, so a set of metadata (intrinsic in the data) regarding its computational documentation, semantic and syntax declaration must be saved.

- Infrastructure: this component allows access to data and metadata, and running the code associated to the data product.

3.3.3 Self-serve Data Platform

Previous principles can have undesired consequences: more efforts in each domain, more operation cost, and large-scale inconsistencies across domains. So, how we manage these costs if each domain needs to build its own data? Or what is the technology that I should use to provide all the data product usability features? It is necessary that teams have access to an high-level infrastructure, capable of encapsulating all data product's complexity. Therefore, the notion of self-serve platform emerges: interoperability and connectivity must be always ensured and well defined, thus the self-serve platform must be able to provide tools and user interfaces for the maintenance of the data product, without the needs of highly specialized knowledge. Then, the key differentiation of data mesh platform is to make changes to existing platforms so they can scale-out sharing, accessing and using analytical data in a decentralized manner for a new population of generalist technologists[8]. For the logical infrastructure of a data mesh platform, the notion

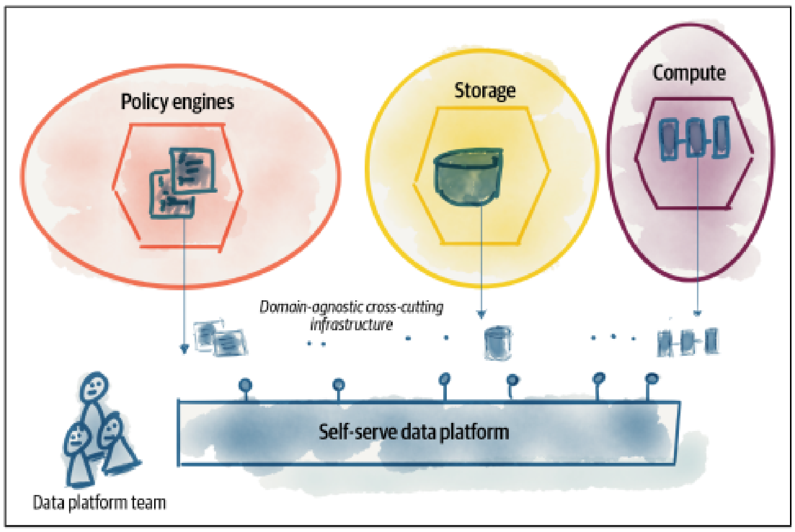


Figure 3.5: Extraction and collection of domain-independent infrastructure and tools into a separate one

of plane is used: it is a collection of capabilities with complementary objectives, high functional cohesion to provide an end-to-end outcome, and its role is to abstract the complexity of the platform, offering a set of interfaces (APIs) to ensure these capabilities [8]. Deheghani proposes three kinds of plane:

1. Data infrastructure (Utility) plane: it is responsible for dealing with low-level resources which compose and run the mesh, e.g., storage, identity system, etc.
2. Data product experience plane: higher level plane used to build, maintain and exploit data products, built using the infrastructure plane. It contains interfaces to allow developers of managing data product's life-cycle.
3. Mesh experience plane: this plane abstract the mesh (graph of connected data products) capabilities to search and explore the best data product for a given use case.

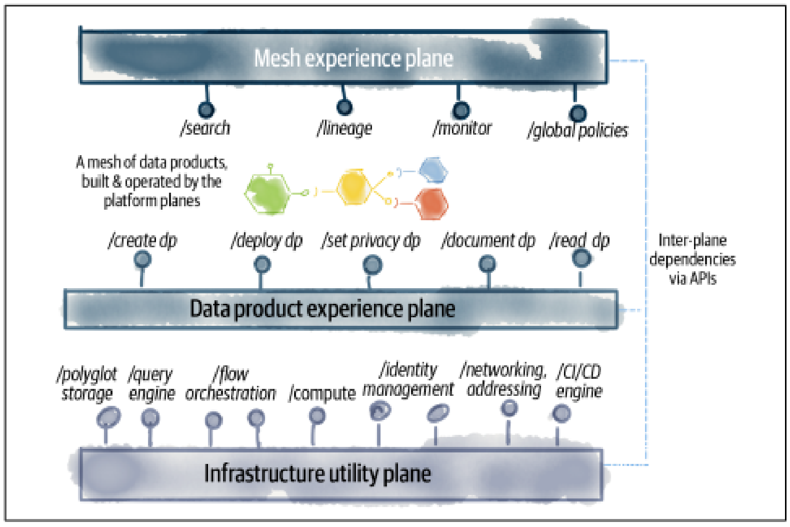


Figure 3.6: Multiple planes of the self-serve data platform

3.3.4 Federated Computational Governance

Governance is the mechanism which ensures security, trust and most importantly the extraction of value from the independent data products in the mesh. This last pillar of data mesh architecture, relates to central teams and processes that become bottlenecks in serving and using data: they must ensure availability of safe, consistent, compliant, privacy-respecting data across the company with managed risk. Federated and computational governance can be summarized as a decision-making model led by all the independent domain data product and platform's owners, which take decisions locally (in their respective domain), but adhering to a set of global rules. This raise a new set of concerns for data mesh to address: interoperability, which is essential to extract values from lot of independent data products, and the standardization of data communication, e.g. standardization of data presentation and querying across all domains [8]. Deheghani proposes the following logical architecture to deal with the existence of global rules and standards, while preserving the domain's autonomy:

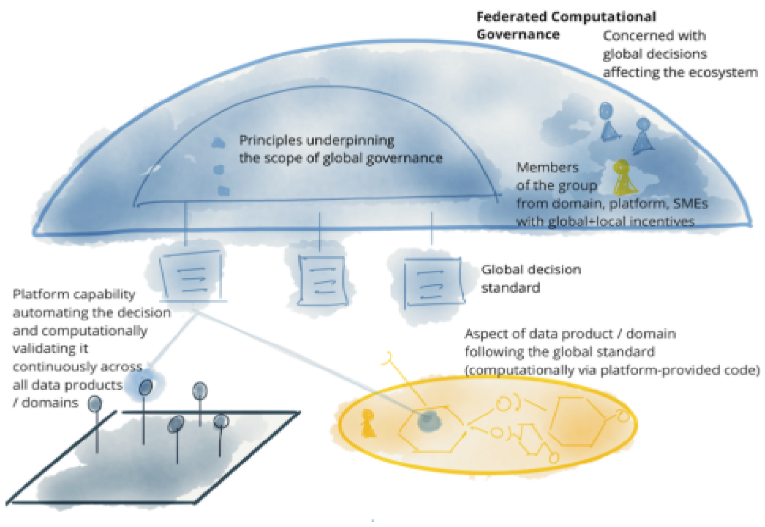

Federated Computational Governance
Concerned with global decisions affecting the ecosystem
Principles underpinning the scope of global governance
Members of the group from domain, platform, SMEs with global+local incentives
Global decision standard
Platform capability automating the decision and computationally validating it continuously across all data products / domains
Aspect of data product / domain following the global standard (computationally via platform-provided code)

Figure 3.7: Federated Computational Governance

3.4 After the inflection point

The inflection point named at the beginning of the chapter, is important because is the point where we look at what has come before, learn from it and choose a new direction in our organization. Now that we have analyzed the principles and advantages of data mesh, we can highlight the data mesh strong points as:

- Respond gracefully to change: the complexity of businesses systems, composed by many domains each with its accountability structure and goal, need to face the volatility and rapid changes of the markets. So how can business manage the impact of this complexity on its data? The solution is to embracing changes in a complex organization. Dehegani proposes different ways to achieve it, such as breaking the business complexity into independently managed parts (so aligning business,tech and analytical data), or closing the gap between analytical and operational data, to ensure data reflecting the business truthfulness.

- Sustain agility in the face of growth: the success of business results in new acquisitions, new products, new geolocation expansion and so on. Thus, becomes difficult to manage all the new source of data and data-driven use case, slowing down the delivering value process. The data mesh agility in the face of growth can be summarized in techniques which aim to remove the monolithic and centralized bottlenecks, coordination of data governance and synchronization.

- Increase the ratio of value from data to the investment: data mesh proposes a new data platform that abstracts today's technical complexity through open data interfaces, which allow data sharing across organizational boundaries or physical locations and through applying product thinking to improve experience of data users. In this way data are more accessible, and so more value can be derived [8].

Chapter 4

Adopted Technologies

This chapter illustrates the technologies adopted for the implementation of the proposed solution. ADF is the ingestion platform, with all its services (Storage Account, Integration Runtime, Key-Vault, SQL server), while Databricks allow to run notebooks written in Python, Scala and R, and it's used for the ETL pipeline and the ML algorithm.

4.1 Overview of Azure Data Factory

Azure Data Factory is the Microsoft service that orchestrates and operationalize processes to refine very large relational and non-relational stores of raw data without proper context or meaning, into actionable business insights. It is the managed cloud service built for complex hybrid ETL, ELT and data integration projects. ADF allows the creation and scheduling of data-driven workflows (called pipeline) that can ingest data from various data sources. The ETL processes can also be built with compute services that are integrated in ADF, like Azure HDInsight Hadoop, Azure Databricks, and Azure SQL Database. Then, the transformed data can be organized and published into meaningful data stores such as Azure Synapse, SQL server and Datalakes, for Business Intelligence applications to consume.

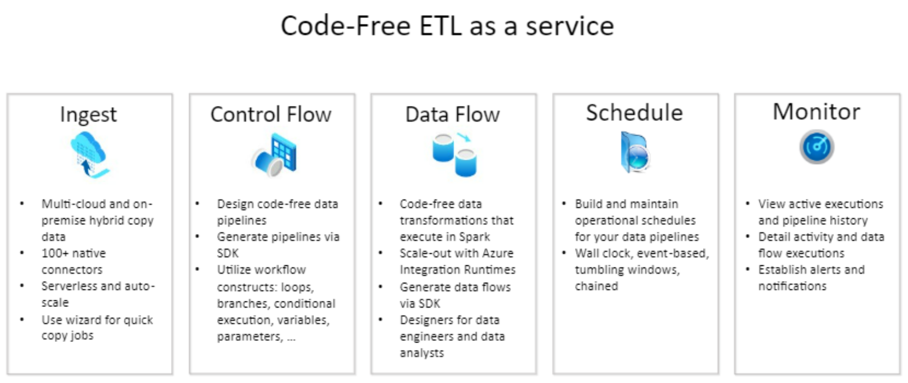
The main operations which make ADF a good platform solution for the realization of a data product are:

- Connect and collect: The first step in building an information platform is to connect all data sources and data processing systems, such as SaaS services, databases and FTP web services. Without data factory, enterprises need to collect various type of data located in disparate sources in the cloud, structured or unstructured and arriving at different times and speeds, in a custom data movement component or create custom services to integrate these data sources or processing. ADF provides various kind of no-code functionalities, called *activities*, which allow the managing of data collection. With the Copy Data activity, for example, you can move data from both on-premises and cloud source data stores to a centralized data store in the cloud for further analysis, using Azure Data Lake Analytics or Azure Databricks.

- Transform and enrich: ones data are stored in a centralized data store in the cloud, they need to be processed or transformed. ADF Data Flow, enable engineers to build and maintain data transformation graphs executed in Spark, without having a deep knowledge of it. It is also possible to write transformation code by hand, using compute services named before.

- CI/CD and publish: CI/CD is a mechanism to frequently integrate and deliver changes made on your codebase automatically. In ADF it means moving Data Factory pipelines from one environment (development, test, production) to another. This allows to incrementally develop and deliver the ETL processes before publishing the finished product. After data are ready for business consuming, they can be load in Data warehouses, SQL Databases or whichever engine the business users can point to from their tools.

- Monitor: after the successful deployment of the integration pipeline and the provided business value from the data, ADF provides a built-in support for pipeline monitoring of the scheduled activities for success and failure rates.



Code-Free ETL as a service

Ingest
- Multi-cloud and on-premise hybrid copy data
- 100+ native connectors
- Serverless and auto-scale
- Use wizard for quick copy jobs

Control Flow
- Design code-free data pipelines
- Generate pipelines via SDK
- Utilize workflow constructs: loops, branches, conditional execution, variables, parameters, …

Data Flow
- Code-free data transformations that execute in Spark
- Scale-out with Azure Integration Runtimes
- Generate data flows via SDK
- Designers for data engineers and data analysts

Schedule
- Build and maintain operational schedules for your data pipelines
- Wall clock, event-based, tumbling windows, chained

Monitor
- View active executions and pipeline history
- Detail activity and data flow executions
- Establish alerts and notifications

Figure 4.1: ADF provides a series of interconnected systems for an end-to-end platform

Azure Data Factory works through some components to provide the platform on which is possible to create a data-driven workflows to move and transform data [9]. These components are listed and described in the following paragraphs.

4.1.1 Pipelines

Pipelines are logical grouping of activities that perform a task, allowing you to manage activities as a set instead of individually. For example, a pipeline can have a set of activities that ingest and clean log data, and then kick off a mapping data flow to analyze them. Activities in the pipeline define what to do on the data: you can exploit Copy Data activity to move data from SQL server to the Azure Blob Storage, then use Databricks Notebook activity to process and transform them from the blob storage, and store the final results on a datalake or whatever storage system for BI reporting. Each activity that compose the pipeline, take one or more input datasets and produces one or more output datasets, and they can grouped as:

1. Data movement activities: Copy Data activity is the main activity used for this task. It copies data from a source data store to a sink one. There are many types of data stores supported by data factory such as Blob Storage, Cosmos DB (SQL API), Data Lake Gen2, Delta Lake, Database for MySQL and so on.

2. Data transformation activities: a transformation activity executes in a computing environment such as Databricks or HDInsight (Hadoop). Transformations can be executed as: mapping data flows, which allow to develop graphical data transformation logic without writing code, Data wrangling through Power Query, which allows to do code-free data preparation at cloud scale iteratively, and managing your transformations writing code on external compute environments.

3. Control flow activities: examples are *Look up* activity to read a record/table name/value from an external sources, *Execute Pipeline* to invoke and execute another pipeline, *If Condition*, and iterative control activity like *For Each*.

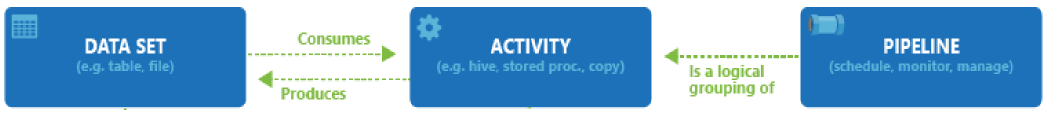
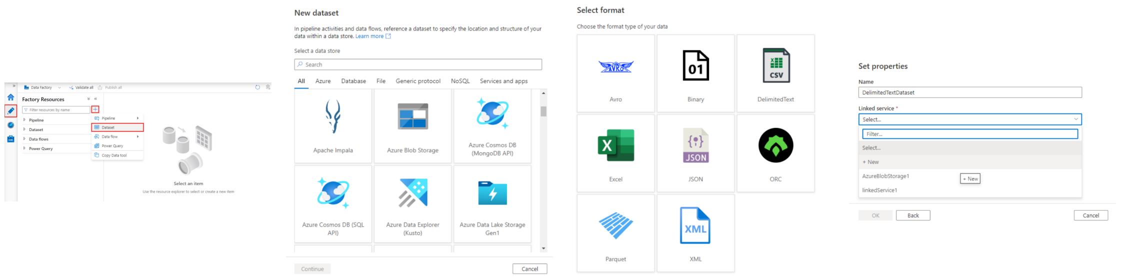


Figure 4.2: Relationship between dataset, activity and pipeline

For the pipelines, is also possible to define parameters and variables. Parameters, once set, are constant during the whole time of the pipeline run. You can read them but not modify. Variables instead, can be set and modified during a pipeline run through a *"Set Variable"* activity [10].

4.1.2 Datasets

As said, ADF can have one or more pipelines, which are group of activities that perform a task. Now datasets in ADF, are simply views of data that points and reference the data used as input or output in the activities. Dataset refers to data in a given data stores, such as table, folder or documents. For example an Azure Blob dataset identifies data in a given container and folder, in the Blob Storage they belong. ADF support different dataset formats: json, csv, xlsx, bin, parquet and others. In Copy Data activities and in Data Flow, they can be used as source or sink, and you can also infer them a specific Schema. Before the creation of a dataset, ADF needs to be linked to a *Linked Service* towards the data store. Linked Services simply define the connection information needed for the service to link with external data sources. Basically datasets can be thought as the representation of the structure of the data in the data store, and the corresponding linked service represents the connection to the store [11].


New dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. Learn more

Select a data store

Set properties

Name: DelimitedTextDataset

Linked service *: Select...

Figure 4.3: Steps to create a dataset

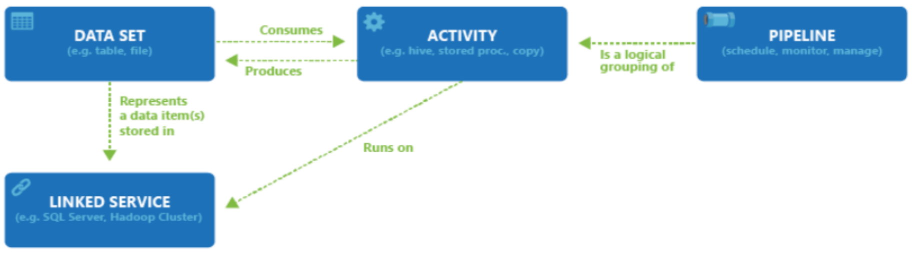

DATA SET
(e.g. table, file)
Consumes
Produces
ACTIVITY
(e.g. hive, stored proc., copy)
Is a logical grouping of
PIPELINE
(schedule, monitor, manage)
Represents a data item(s) stored in
Runs on
LINKED SERVICE
(e.g. SQL Server, Hadoop Cluster)

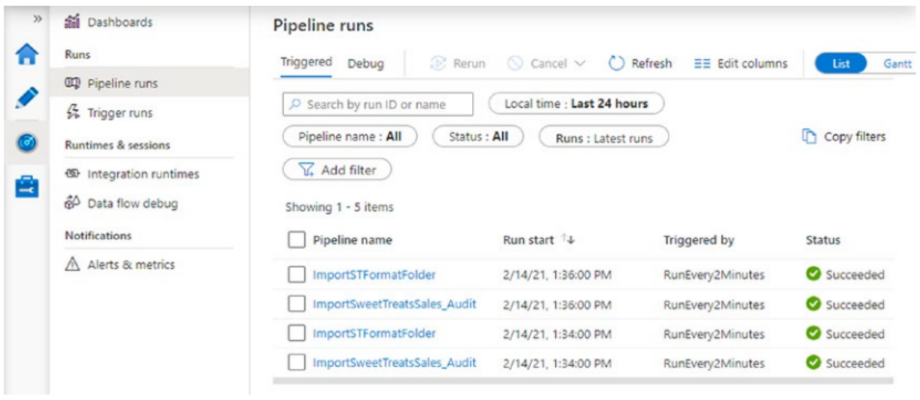
Figure 4.4: Relationship between dataset, activity, pipeline and linked service

4.1.3 Pipeline execution and triggers

A pipeline run in ADF, defines an instance of pipeline execution. In fact, a pipeline can be executed in different instants, and one instance will be created for each run with the corresponding run ID, which uniquely identifies that particular run. Pipelines can be run manually or using a *trigger*. Triggers represents the automatic way to define the conditions for which a pipeline execution must be kicked off:

- A *scheduled trigger* simply define the date and time at which run the pipeline;
- An *event-based trigger* runs when a file is created or deleted in the blob storage;
- A *tumbling window trigger* is run at a specific date and time, but unlike the scheduled trigger, defines the interval (window) between two successive executions. Start and end time can be passed as parameters to the pipeline to make data processing time-sliced.

A trigger can be associated to one or more pipelines, and when its conditions are met, it runs. The trigger run determines the execution of each of its associated pipelines [12].



Pipeline runs

Local time : **Last 24 hours**

Pipeline name : **All** | Status : **All** | Runs : Latest runs

Showing 1 - 5 items

Pipeline name	Run start	Triggered by	Status
ImportSTFormatFolder	2/14/21, 1:36:00 PM	RunEvery2Minutes	✓ Succeeded
ImportSweetTreatsSales_Audit	2/14/21, 1:36:00 PM	RunEvery2Minutes	✓ Succeeded
ImportSTFormatFolder	2/14/21, 1:34:00 PM	RunEvery2Minutes	✓ Succeeded
ImportSweetTreatsSales_Audit	2/14/21, 1:34:00 PM	RunEvery2Minutes	✓ Succeeded

Figure 4.5: Examples of pipeline executions after the corresponding trigger run

4.1.4 Integration Runtime

The Linked Services represent connections to storage systems or computing resources which are external to data factory. ADF provides internal computing system (it does not have internal storage systems) which manage all the activities in defined in the Data Factory instance, such as Copy Data, Lookup activity and Data Flow. This system is the Integration Runtime (IR). An integration runtime provides the bridge between activities and linked service, and constitute the environment where the activity runs or gets dipatched from. In this way, activities can be performed on the closest region where the data store lies, and compute services in the most performant way while meeting security and compliance needs. Each data factory instance is equipped with its default IR, named *AutoResolveIntegrationRuntime*, but new integration runtimes can be instantiated due to geographical reasons, or increase the default time-to-live (TTL), for a just in time (JIT) Databricks cluster. There is also a different type of IR, called *Self-Hosted Integration Runtime*. Its purpose is to provide secure access to private networks, for example to data sources located in an Azure virtual network, like file systems, SQL server instances and ODBC-compliant database services. Since a self-hosted IR in a private network can be associated to only one data factory, a common solution is to create a new data factory instance (shared data factory) which encapsulates and share it with other ADF instances. Thus, they don't need a direct relationship to the underlying gateway implementation [13].

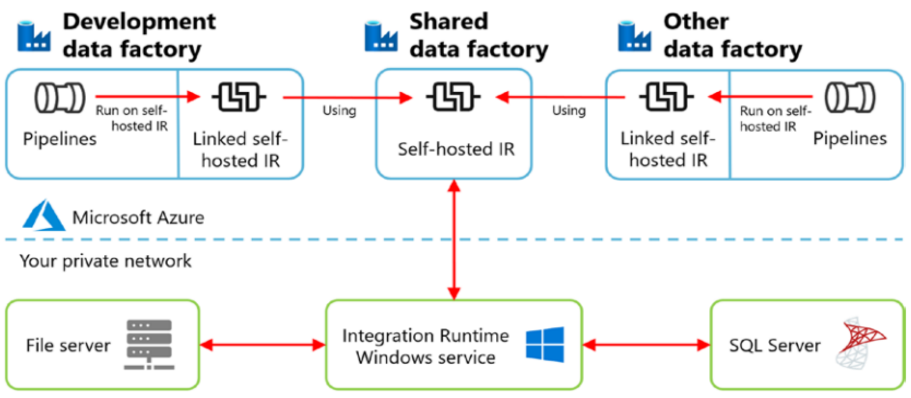


Figure 4.6: How self-hosted IR works with other ADF instances

4.2 Azure Databricks

Databricks is the distributed computational environment which is integrated in ADF and with all its services to transform, visualize and extract insights from large amount of data. It is built on Apache Spark, a distributed and cluster-based computing system with capabilities that provide speed and ease of use, and it can run very quickly on large datasets thanks to its in-memory processing design, which allow to run with few I/O operations. Databricks allows the processing of highly reliable data pipelines, and data science tasks, the easy external data storage integration (like Blob Storage and SQL server), and the automatic cluster management. Databricks also supports different languages, such as Scala, Python and R, to make the definition of ETL pipelines and transformations for developers easier: it is facilitated by the notebook-like integrated workspace, which allows data teams to have a real time, cross-functional collaboration. Notebooks can be used to create jobs which can be later scheduled, meaning that locally development code, can be easily deployed to production [14].

4.2.1 Databricks architecture

A databricks cluster is a databricks application composed by a set of virtual machines, managed as Azure resources and as a single group. This is deployed with a virtual network called *VNet*, a security group to manage the resources permissions, and a storage account used, among other things, as file system. Once everything is deployed, users can manage clusters through the Azure Databricks UI. The most

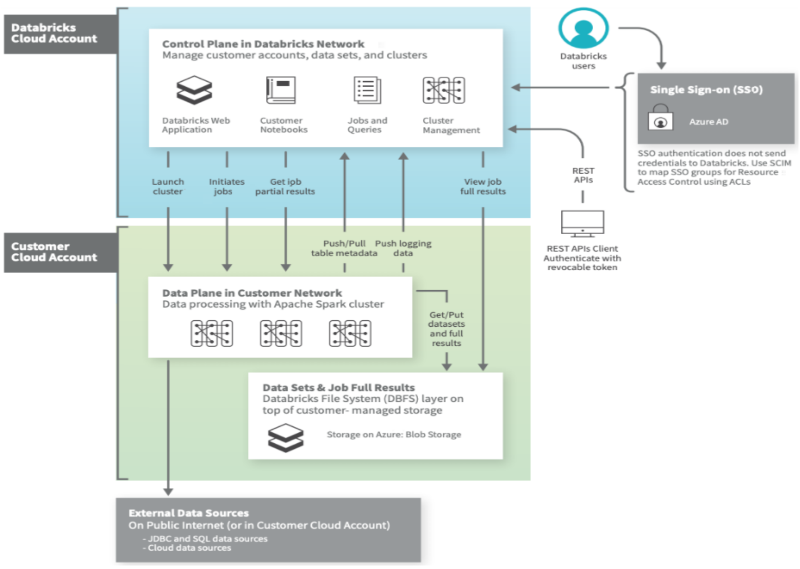

Databricks Cloud Account
Control Plane in Databricks Network
Manage customer accounts, data sets, and clusters
Databricks Web Application
Customer Notebooks
Jobs and Queries
Cluster Management
Databricks users
Single Sign-on (SSO)
Azure AD
SSO authentication does not send credentials to Databricks. Use SCIM to map SSO groups for Resource Access Control using ACLs
REST APIs
REST APIs Client Authenticate with revocable token
Launch cluster
Initiates jobs
Get job partial results
View job full results
Customer Cloud Account
Push/Pull table metadata
Push logging data
Data Plane in Customer Network
Data processing with Apache Spark cluster
Get/Put datasets and full results
Data Sets & Job Full Results
Databricks File System (DBFS) layer on top of customer- managed storage
Storage on Azure: Blob Storage
External Data Sources
On Public Internet (or in Customer Cloud Account)
- JDBC and SQL data sources
- Cloud data sources

Figure 4.7: Databricks Architecture

important benefit this architecture gives, is the seamless connection to Azure which allows Databricks to be linked with any resource within the same resource group, and have a centrally managed Databricks from the Azure control center, without any additional setups. Figure 4.7 shows that Databricks is composed by a *control plane* and a *data plane*: the first one includes backend services that databricks manages in its own Azure account, and commands and workspace configurations. The second one is managed by user's Azure account, and is where data resides, maintaining control and ownership. It includes VNet, NSG (Network Security Group) and the storage account known as DBFS [15].

4.2.2 Core concepts

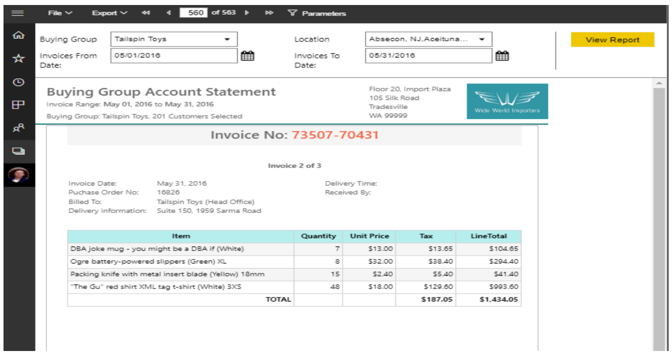
To have a better understand of what databricks is and why it is so useful, we need to familiarize with the following core concepts:

- Workspace: is the environment where users can share and access to jobs, notebooks, libraries, data and models, all organized in folders.
- Notebook: it is a web interface were cells of code can be run. The result is a document easy to visualize and share. The other option to run code, is through jobs. In order to be able to run cell code, cells need to be connected to a cluster, but this connection is not necessarily permanent: notebooks can be scheduled as jobs to create pipelines, run ML models, or update dashboards.
- Cluster: a cluster is a set of connected servers that work together collaboratively, as if they are a single computer. This structure, allows better performance and the distribution of the workload among more machines. There are *all-purpose* clusters and *job clusters.* The former are the ones on which we work collaboratively using notebooks, while the latter are used to execute automatic and more concrete jobs.
- Job: it is a task that we run when executing a notebook, JAR, or Python file on a certain clusters.
- Apache SparkContext/environment: it is the main application in Apache Spark running internal services and providing the Spark execution environment.

- Azure Databricks workspace filesystem: databricks is deployed with a distributed filesystem, which is mounted in the workspace and allows user to mount storage systems and exploit them through paths [14].

4.3 Power BI

Power BI provides a collection of services, apps and connectors to turn unrelated sources of data into coherent, visually immersive, and interactive insights [16]. It allows the easily visualization of data and sharing of them, in order to discover what is important.
The main usage of Power BI in our context, is for creating reports and dashboards. One common workflow begins by connecting a data source to build a report (from *Power BI Desktop*) and then publish it to *Power BI Service*, and share it so business users can interact with it. For this purpose, a particular kind of reports are developed: *paginated reports*. They are called *"paginated"* because they fit well on a page. They display all the data in a table, even if the table spans multiple pages [17].



Buying Group	Tailspin Toys	Location	Absecon, NJ,Aceituna...
Invoices From Date:	05/01/2016	Invoices To Date:	05/31/2016

Buying Group Account Statement
Invoice Range: May 01, 2016 to May 31, 2016
Buying Group: Tailspin Toys, 201 Customers Selected

Floor 20, Import Plaza
105 Silk Road
Tradesville
WA 99999

Invoice No: 73507-70431

Invoice 2 of 3

Invoice Date:	May 31, 2016	Delivery Time:	
Puchase Order No:	16826	Received By:	
Billed To:	Tailspin Toys (Head Office)		
Delivery Information:	Suite 150, 1959 Sarma Road		

Item	Quantity	Unit Price	Tax	LineTotal
DBA joke mug - you might be a DBA if (White)	7	$13.00	$13.65	$104.65
Ogre battery-powered slippers (Green) XL	8	$32.00	$38.40	$294.40
Packing knife with metal insert blade (Yellow) 18mm	15	$2.40	$5.40	$41.40
"The Gu" red shirt XML tag t-shirt (White) 3XS	48	$18.00	$129.60	$993.60
TOTAL			$187.05	$1,434.05

Figure 4.8: Example of a paginated report

Chapter 5

Case Study

This chapter presents the case study analyzed. It describes each component of the data mesh solution, with its functionality, developed for an energy provider company. The main objective, is to identify different domains and use case, which can be extracted from the huge amount of data collected by IoT devices, and provide a report tool, to better visualize and analyze the *"entities"* associated to those data. The entities in exam are quantities extracted from the work machines (gas consumptions, gas flow, temperature...). The presented work, proposes two main solutions, each regarding to specific domains of the business.

5.1 Domain and Use Cases identification

The first analysis, is about the individuation of the domains. A *Top-Down* approach is followed, starting from the source systems and mapping the business processes for each sub-area. The considered areas (*clusters*), with the respective owners are described in the following table:

Cluster	**Owner**
Forecast & Bidding	Owner1
Sales	Owner1
BI & Data Platform	Owner1
Maintenance	Owner2
Programming	Owner2
Monitoring	Owner2
Delivering	Owner2
HSE	Owner2

OWNER1 and *OWNER2* in this work are the data-platforms, working on the reported areas, regarding a specific aspect of the business.
The following table summarizes, the identified domains:

Domain	**Owner**
Forecast & Bidding - Market	Owner1
Sales - Commercial	Owner1
Production	Owner2
Maintenance	Owner2
Delivering	Owner2
Health Security Environment (HSE)	Owner2
Trading & Hedging	External
Finance	External

The *EXTERNAL* owner is outside the considered case study (which contains only *OWNER1* and *OWNER2*). The first solution is implemented to manage the consumptions and yields reports for the examined station, so the following use cases are identified:

- P1_C1 Consumptions Monitoring: monitor in real-time plants consumes to support the decision-making on the productions with respect to the thresholds of reference;

- P1_C2 Yields Monitoring: monitor in real-time the delta between actual yield and optimal (estimated) yield;
- P1_C3 Enhancement yield delta: batch-evaluation of the difference of the yield, starting from actual yield, optimal yield, conversion constants and data from the considered source (Gas Flow, PCI, CO2 Price and Gas Price).

The following figure shows the entity map, which represents the entities analyzed in each use case:

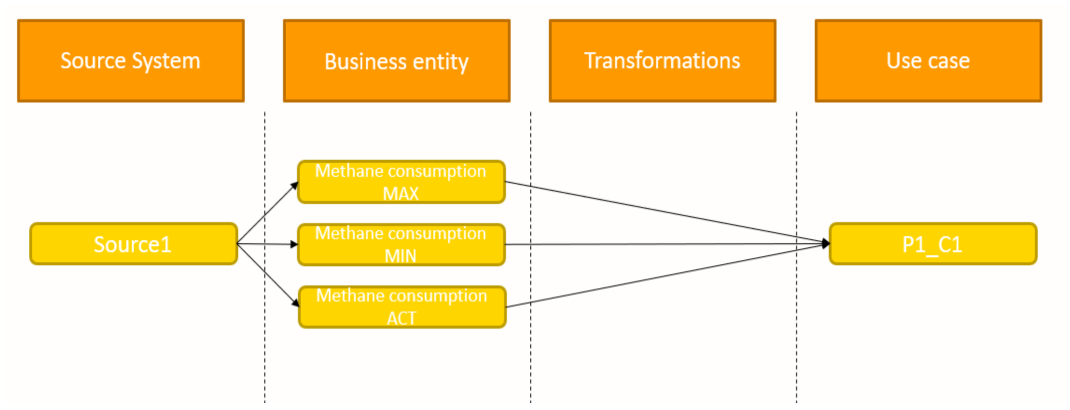


Figure 5.1: Entity map P1_C1

The first solution is *source aligned*, so there are no transformations on the source data, and they are reported as they are in the final dashboard.

The involved quantities are:

- Methane consumption MAX: consumption prediction when the unit works at its maximal power;
- Methane consumption MIN: consumption prediction when the unit works at its minimum power;

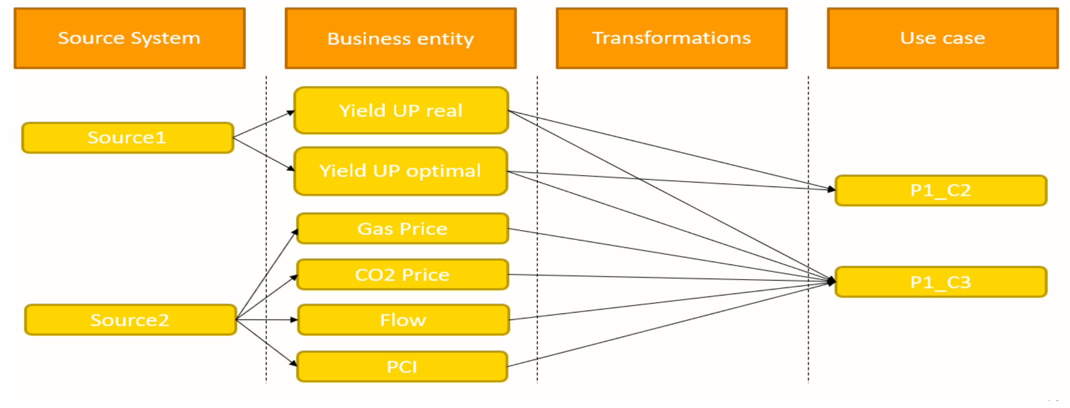


Figure 5.2: Entity map P1_C2 and P1_C3

- Methane consumption ACT: consumption prediction when the unit works at its actual power;

The second solution is implemented to populate the consumptions (about other entities w.r.t the first solution) and algorithm performance reports, and the following use cases are identified:

- P2_C1 Consumption monitoring: batch-monitoring of electric (EE) and thermal (ET) energy.
- P2_C2 Programming error monitoring: batch-monitoring of the average error percentage between the final consumption and the last planning sent (last scenario).
- P2_C3 Expected error monitoring: batch-monitoring of the average error percentage between the last consumption and the last expected consumption.

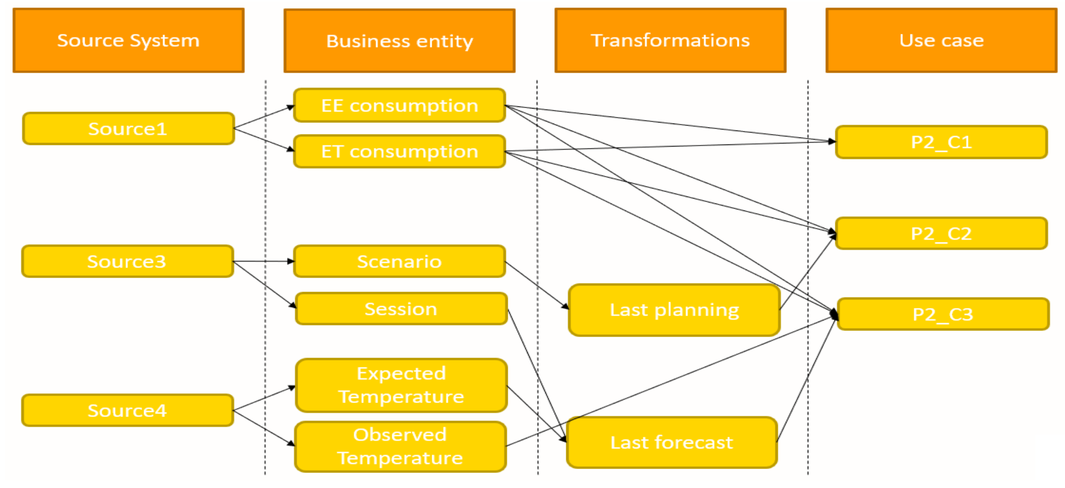


Figure 5.3: Entity map of the three use cases of the second pilot

The reported business entities represent the quantities and variables the business wants to monitor in the final dashboards.

5.2 Mesh Architectures

In this section are described the conceptual data mesh map for each solution, the data flow and the role of each Azure service.

5.2.1 Data Product Map - First solution

Figure 5.4, shows the Data Product Map for the first solution. The solution is implemented to manage the consumptions and yields reports.

Two data products are developed, each working on its domain and entities. The rightmost node, is the customer platform, where the reported results can be consulted via PowerBI.

In the first source, we can see the IoT Hub, which is an Azure Service to manage IoT devices. It is used to read the data from a web server through an HTTP request and route only the telemetry messages towards the Event Hub. Those messages

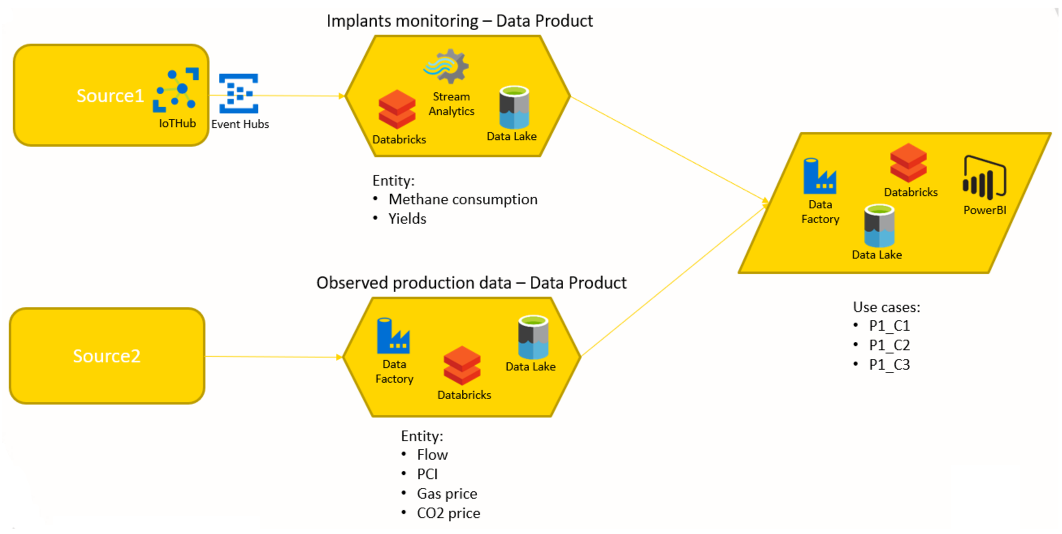
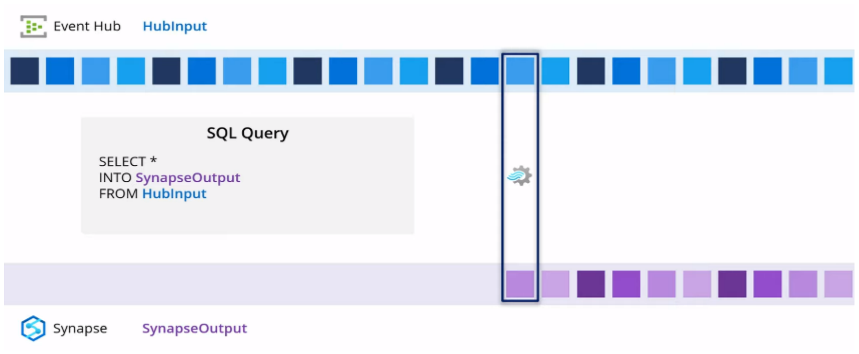


Figure 5.4: Entity map of the three use cases of the second pilot

are identified through a *"tag"*. The Event Hub is an event streaming service which is organized in partitions, each taking the data we define for each flow. Then, the Stream Analytics service is used to apply the transformations over those data. The transformations (*jobs*) are SQL query which can work over streams of data.



SQL Query

```
SELECT *
INTO SynapseOutput
FROM HubInput
```

Figure 5.5: Example of a job over stream of data

When new data comes, the job is triggered and the query is run on the batch of data.

5.2.2 Data Ingestion - First solution

Yield and consumption data are taken from *SOURCE1* and made available on PowerBI minimizing the publication on the source and the visualization on the dashboard. Moreover, data are stored also in the Datalake, and examinable through analysis tool (Databricks) with the following structure:

Name	**Type**	**Note**
Tag	string	The quantity we want to take in exam
TimestampShortStr	string	
TimestampUtcStr	string	Data will be available every quarter of hour.
Value	double	
ValueUoM	string	Unit of measure
ReferenceUtc	Datetime	From TimestampUtcStr

In the reports, the following informations are made available:

- Site and Unit, based on the tag;

- Day, hour (from 1 to 23/24/25) and the quarter of hour (0-15-30-45).

Yields and consumptions are taken from SOURCE1, and as seen in the entity map, they don't need transformations.
The ingestion of data from SOURCE2, follow the same steps of the SOURCE1, and the structure for the consultation on the Datalake is:

Name	**Type**	**Note**
Codice	string	TAG name
Data	string	
Tipologia	string	Validated, Published, Billed
DescrizioneTipologia	string	
ReferenceTsStr	string	Timestamp (UTC + 15min)
Valore	double	Value of the tag
StatoValidazione	string	YES/NO
MessageIdEmbedded	string	
MessageType	string	
MessageTypeCod	string	
MessageSourceName	string	
MessageTimestamp	string	UTC
DataCalcolata	string	yyyymmdd
QuartoCalcolato	int	from 1 to 92/96/100
Reference_Ts_UTC	datetime	
Partition_AnnoMese	string	yyyymm

In the reports, the following informations are made available:

- Site and Unit, based on the tag;
- Day, hour (from 1 to 23/24/25) and the quarter of hour (0-15-30-45);
- Value with "Tipologia" with the highest priority (Billed, Published, Validated), on the same Date/hour/tag.

5.2.3 Data Product Map - Second solution

The following figure, shows the Data Product Map for the second solution:

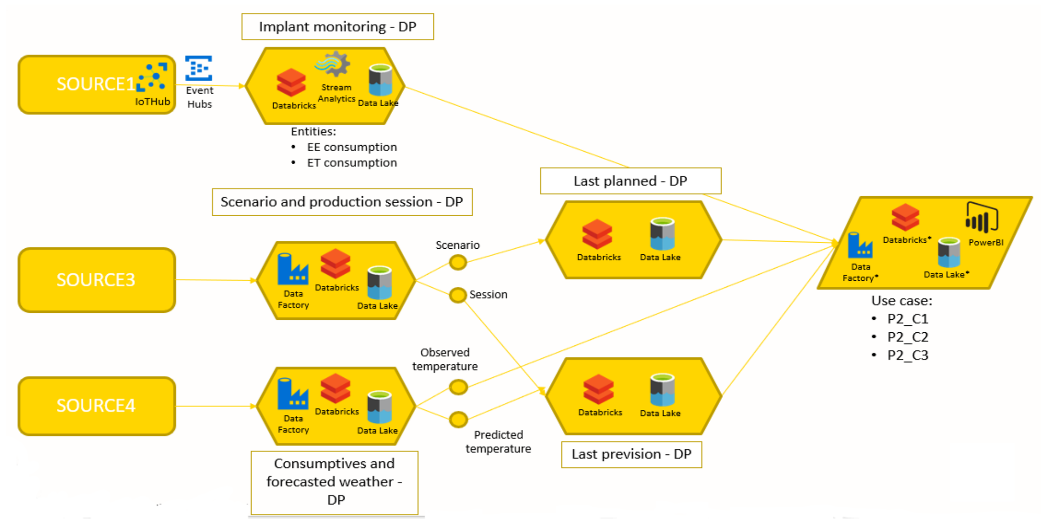


Figure 5.6: Data Product map of the second pilot

This architecture is implemented to manage the consumptions and the algorithm performances reports.

5.2.4 Data Ingestion - Second solution

Data from SOURCE1 (EE and ET) will be stored in the Datalake, and made available through Databricks, with the structure shown in the following format:

Name	**Type**	**Note**
Tag	string	
TimestampShortStr	string	Observed data will be extracted and made available every hour
TimestampUtcStr	string	
Value	double	
ValueUoM	string	
ReferenceUtc	Datetime	From TimestampUtcStr

SOURCE1 provides data about consumptives. The data will be available with the explicit informations of the day and hour (from 1 to 23/24/25). Data in SOURCE3 provides informations about *Session* and *Scenario*, which are made available with the following structure.

Name	**Type**	**Note**
ID Scenario	string	A scenario is a validated session.
ID Sessione	string	A session corresponds to an algorithm run.
CreationDateUTC	datetime	Date of starting session
Entity	string	W (scenario), P(session)
Tipo	string	ACT, PRG, LAST. (only PRG are of interest for the pilot)
Serie	string	Is the tag. Only EE and ET are of interest for the pilot
ReferenceDateUTC	datetime	
Valore	double	
DataCalcolata	string	Format yyyymmdd, computed from Reference_Ts_UTC
OraCalcolata	int	From 1 to 23/24/25, computed from Reference_Ts_UTC
Partition_AnnoMese	string	yyyymm computed from Reference_Ts_UTC

The "Scenario and Production session - DP", run the algorithm which makes the predictions (sessions). When a session is validated it becomes a scenario. A batch flow is implemented (twice per day, when the sessions are validated) for the ingestion of sessions and scenarios data from SOURCE3. Every time data are requested, all the sessions and scenarios validated from the last call, are made available, and if necessary, also those related to specific intervals can be retrieved.

In the final dashboard, only data related to the solution are reported. Sessions, are reported with the data ingested from SOURCE4, with the following features:

- "TempStatus" = F

- "Ingestion_UTC_Datetime" $\leq$ CreationDateUTC
- Higher "Ingestion_UTC_Datetime"

Also the consumptives metheorological data are reported, without the relative session, taking:

- "TempStatus" = "O"
- Last "Ingestion_UTC_Datetime"

The "Ingestion_UTC_Datetime" is an additional feature taken from the datalake of SOURCE3, which keep track of the ingestion date.

From the SOURCE4 the temperature data are ingested. Predicted and observed data are taken to be associated with the corresponding sessions, to populate the reports.

Those data are available in the following structure:

Name	**Type**	**Note**
WeatherStation	string	
Date	string	Date, format dd/mm/yyyy
Hour	string	Hour, format hh:mm
Temp	double	
TempStatus	string	F (forecasted), O (observed)
UpdateDate	string	
UpdateHour	string	
RefTimeTz	string	Timezone information
UpdateTimeTz	string	
DataCalcolata	string	format yyyymmdd
OraCalcolata	int	from 1 to 23/24/25
Reference_Ts_UTC	datetime	
Partition_AnnoMese	string	yyyymm

5.3 Mapping metadata

The following table summarizes the metadata associated to the data products, underlining the domains and the entities involved.

DP owner	Domain	Data product	DP type	Source/Input port	Output port	Entity	Consumer
TBD	Production	Implants monitoring	Source aligned	Source1	File system	Methane consumption (MIN, MAX, ACT)	P1_C1
TBD	Production	Implants monitoring	Source aligned	Source1	File system	Optimal yield	P1_C2. P1_C3
TBD	Production	Implants monitoring	Source aligned	Source1	File system	EE consumption	P2_C1, P2_C2, P2_C3
TBD	Production	Implants monitoring	Source aligned	Source1	File system	ET consumption	P2_C1, P2_C2, P2_C3
TBD	Production	Production consumptives	Source aligned	Source2	File system	Flow rate	P1_C3
TBD	Production	Production consumptives	Source aligned	Source2	File system	PCI	P1_C3
TBD	Production	Production consumptives	Source aligned	Source2	File system	Gas price	P1_C3
TBD	Production	Production consumptives	Source aligned	Source2	File system	CO2 price	P1_C3
TBD	Forecast & bidding - Market	Scenario and session optimization	Source aligned	Source3	File system	Scenario	Last planned DP
TBD	Forecast & bidding - Market	Scenario and session optimization	Source aligned	Source3	File system	Session	Last forecasted DP
TBD	Forecast & bidding - Market	Last planning DP	Consumer DP	Source3	File system	Last planning	P2_C2
TBD	Forecast & bidding - Market	Last forecasted DP	Consumer DP	Source3, Source4	File system	Last forecasted	P2_C3
TBD	Forecast & bidding - Market	Consumptives and forecasted weather	Source aligned	Source4	File system	Forecasted temperature	Last forecasted DP
TBD	Forecast & bidding - Market	Consumptives and forecasted weather	Source aligned	Source4	File system	Forecasted temperature	Last forecasted DP
TBD	Forecast & bidding - Market	Consumptives and forecasted weather	Source aligned	Source4	File system	Observed temperature	P2_C3

5.4 Data Consumption

Data consumption is allowed through the Power BI dashboards. Figure 5.7 shows the consumptions report of the first solution, with respect to thresholds of reference to support the decisions on the production. Is also possible to visualize the consumptions specifying the UP, the site or the Total which is simply the sum of all the sites.

The report shows the entities MIN, MAX, ACT and SEL consumptions. The latter is built by summing the selected UP and consumptions directly from the dashboard.

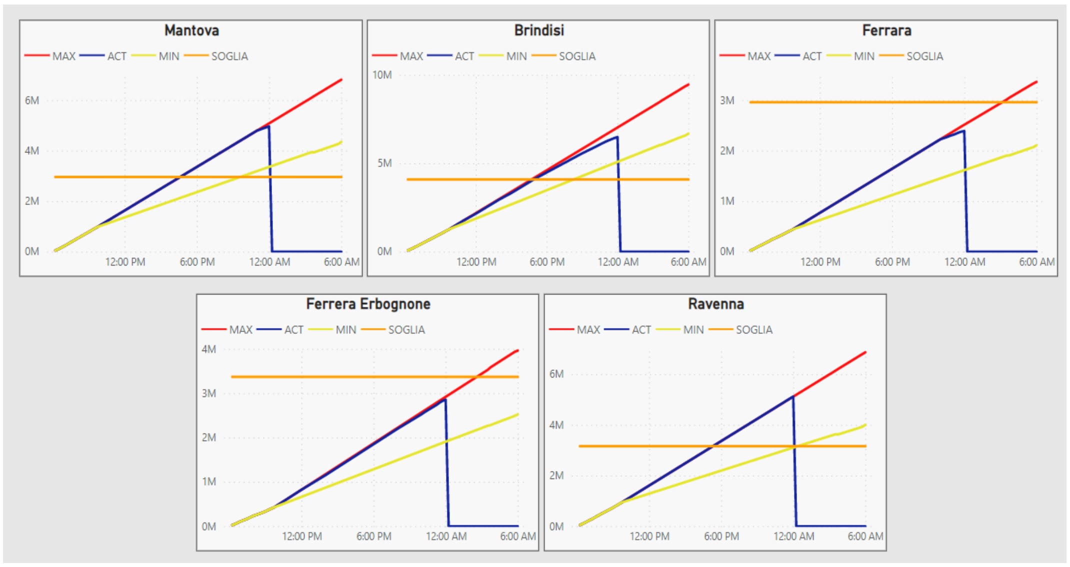


Figure 5.7: Consumption report per site

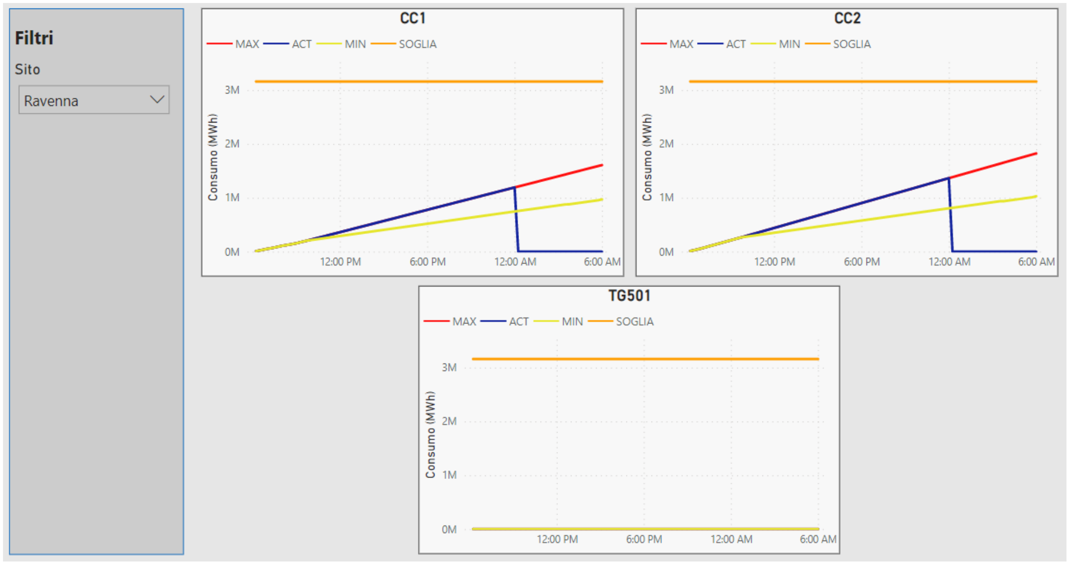


Figure 5.8: Consumption report per UP

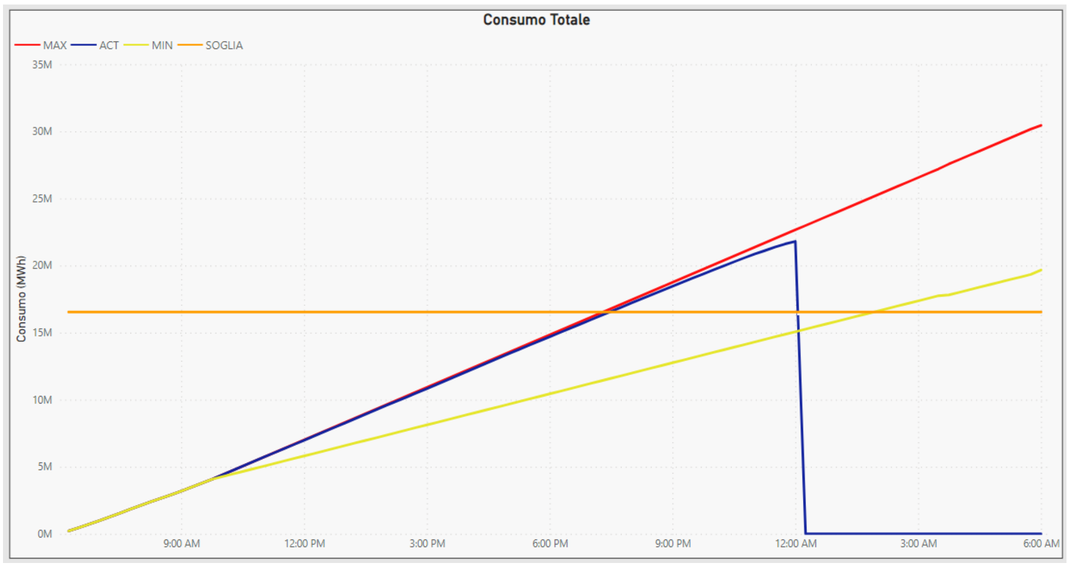


Figure 5.9: Total consumption report

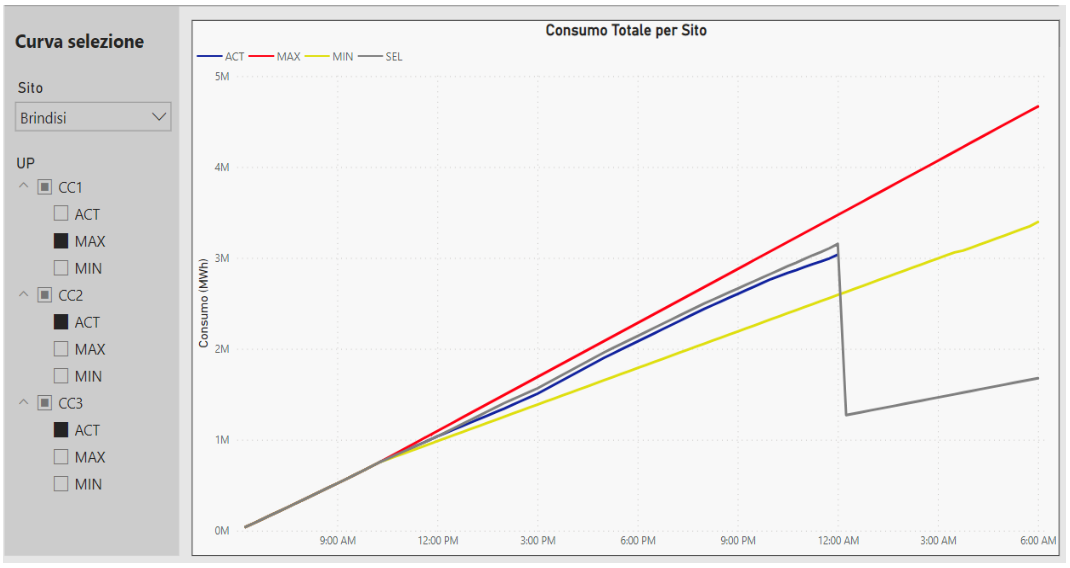


Figure 5.10: SEL consumption report

Figure 5.11 and 5.12 show the yields reports, related to first solution.

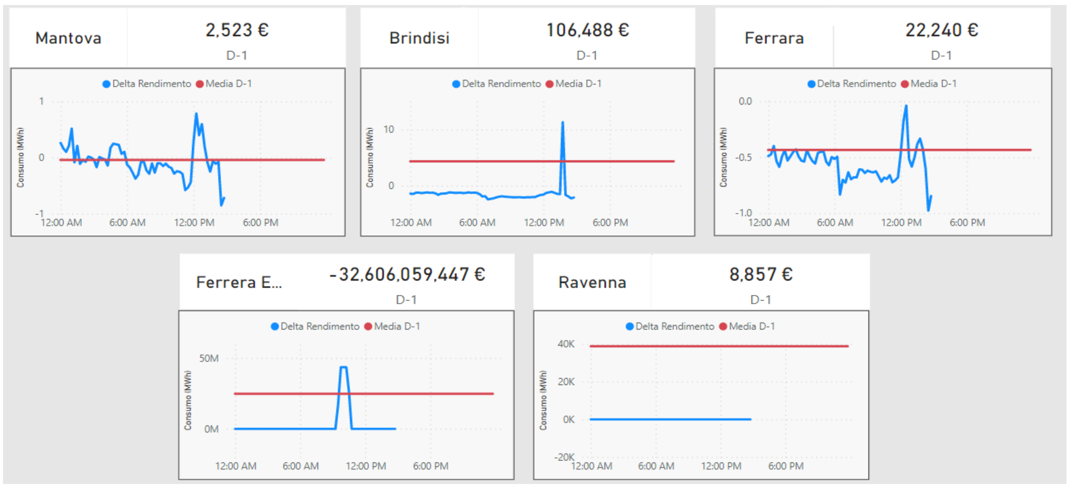


Figure 5.11: Yields per site

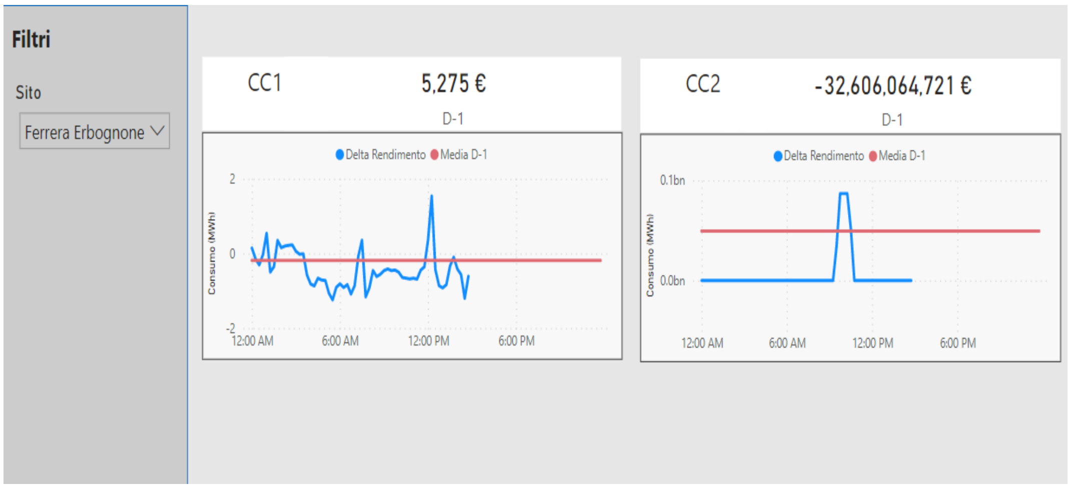


Figure 5.12: Yields per unit

Figure 5.13 and 5.14 show the EE and ET consumptions reports, and the algorithm performance one, related to the second solution.

For each report, is possible to select the period for which visualize the curves. The WMAPE is the average error between consumption and prevision, and is computed with the formula below.

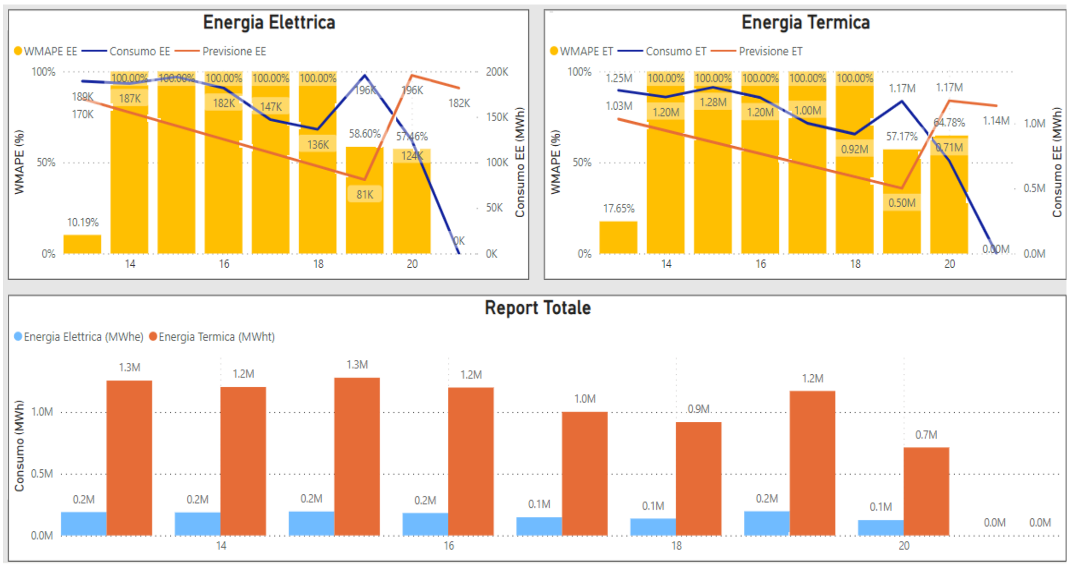


Figure 5.13: Consumptives and predicted consumption report

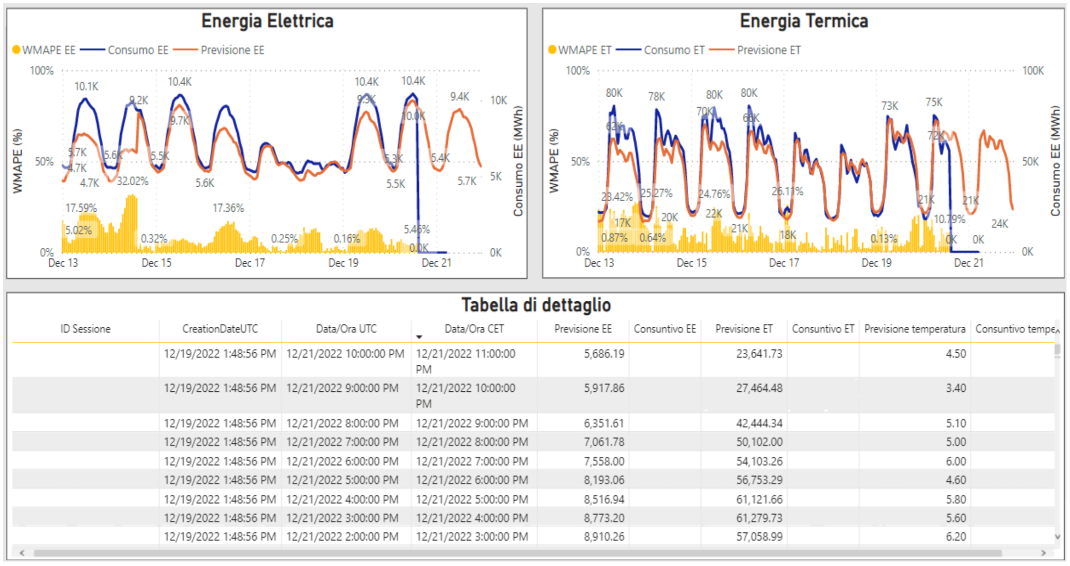


Tabella di dettaglio

ID Sessione	CreationDateUTC	Data/Ora UTC	Data/Ora CET	Previsione EE	Consuntivo EE	Previsione ET	Consuntivo ET	Previsione temperatura	Consuntivo tempe
	12/19/2022 1:48:56 PM	12/21/2022 10:00:00 PM	12/21/2022 11:00:00 PM	5,686.19		23,641.73		4.50	
	12/19/2022 1:48:56 PM	12/21/2022 9:00:00 PM	12/21/2022 10:00:00 PM	5,917.86		27,464.48		3.40	
	12/19/2022 1:48:56 PM	12/21/2022 8:00:00 PM	12/21/2022 9:00:00 PM	6,351.61		42,444.34		5.10	
	12/19/2022 1:48:56 PM	12/21/2022 7:00:00 PM	12/21/2022 8:00:00 PM	7,061.78		50,102.00		5.00	
	12/19/2022 1:48:56 PM	12/21/2022 6:00:00 PM	12/21/2022 7:00:00 PM	7,558.00		54,103.26		6.00	
	12/19/2022 1:48:56 PM	12/21/2022 5:00:00 PM	12/21/2022 6:00:00 PM	8,193.06		56,753.29		4.60	
	12/19/2022 1:48:56 PM	12/21/2022 4:00:00 PM	12/21/2022 5:00:00 PM	8,516.94		61,121.66		5.80	
	12/19/2022 1:48:56 PM	12/21/2022 3:00:00 PM	12/21/2022 4:00:00 PM	8,773.20		61,279.73		5.60	
	12/19/2022 1:48:56 PM	12/21/2022 2:00:00 PM	12/21/2022 3:00:00 PM	8,910.26		57,058.99		6.20	

Figure 5.14: Algorithm performance report

$$WMAPE = \frac{\sum_{period} |consumption - prevision|}{\sum_{period} consumption} \tag{5.1}$$

5.5 Data Marketplace

As future evolution of the proposed mesh architecture, a data marketplace is thought to be integrated.

A data marketplace is a platform where users can buy and sell data. It emerges after the necessity of deliver the data products, meeting the needs of data consumers and ensure the effective use of data enterprise-wide. It works as a “two-sided

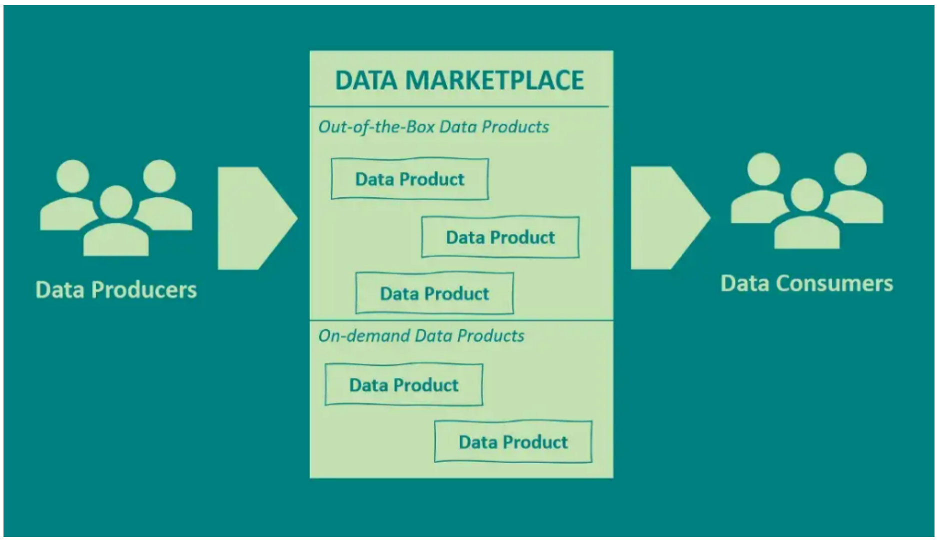


Figure 5.15: Data marketplace logical architecture

market”, where the data data provider commercialize his data assets, and the data buyer find a data source which meets his requirements. Data marketplaces work to the benefit of both parties, which is why more companies are turning to them to unlock successful data strategies [18].

5.5.1 Implement a data marketplace

Data marketplace enablement can be discussed through three aspects: people, processes (operating model), and technology. The people aspect includes roles and responsibilities associated with the marketplace. The main actors involved are obviously the data producer and data consumer. On the producer's side we have data owners, accountable for all aspects of the data products, and data stewards who are responsible for the data product design (the business context) and content. These two figures are supported by data custodians who are accountable for the technical implementation of data product and execution of data delivery. On the other side, data consumers use data products for business operations. Their main responsibilities is related to the specifications of business requirements which can lead to the creation of new or enhancement of existing data products.

The process/operating model aspect refers to the definition and implementation of the standards and repeatable processes under which the data marketplace will operate. These operations includes: create new data product, data product approval/publishing workflow, modifying data product, data product delivery. The technology aspect refers to the tools and platform used to support the data marketplace. It should be a collection of tools and services that supports the workflow management to enable the creation, publication and selling of data products, and the interaction between data producers and consumers [19].

Chapter 6

Conclusions

To sum up, the Data Mesh emerges as a necessary paradigm shift that will enable companies to become truly data-oriented, implementing an architecture that brings the opposite of the current models for efficient data product cooperation. From a more structural perspective, data is organized into domains and data teams manage themselves and carry out their own work in an agile and product-oriented way. This change has the purpose of solving the problems such as loss of the nature of the data itself, high costs related to the management of monolithic architectures (e.g., Data Lakes), significant pressure on data teams, among others. As described in Chapter 2, the main principles for an effective Data Mesh implementation are:

- Domain ownership: each team is responsible only for a specific business domain;
- Data as product: each node of the mesh lives in a context-bounded domain and constitute an autonomous read-optimized data unit containing at least one dataset;
- Self-serve data platform: it provides an high-level infrastructure which encapsulates the complexity of more data products, through the use of tools

and interfaces for the maintenance without the needs of highly specialized knowledge;

- Federated computational governance: it relates to the central teams and processes that become bottlenecks in serving and using data in a secure way and most important in extracting value from the data products. So, this principle ensure the existence of data communication and interoperability while preserving the domain's autonomy.

All these principles have been implemented for the case study described in chapter 5, using the Azure services cited in chapter 4. Other services can be exploit for the the Data Marketplace implementation (such as *Microsoft Purview*), to ensure a truly full and functioning implementation of an architecture that will change and enhance the way data is processed.

Bibliography

[1] Maria Ijaz Baig, Liyana Shuib, and Elaheh Yadegaridehkordi. «Big data in education: a state of the art, limitations, and future research directions». In: *International Journal of Educational Technology in Higher Education* 17.1 (Nov. 2020). DOI: 10.1186/s41239-020-00223-0. URL: https://doi.org/10.1186/s41239-020-00223-0 (cit. on p. 3).

[2] *Le 5V dei Big Data: dal Volume al Valore*. URL: https://blog.osservatori.net/it_it/le-5v-dei-big-data?hsLang=it-it (cit. on p. 4).

[3] URL: https://www.techtarget.com/searchdatamanagement/definition/big-data#:~:text=Why%5C%20is%5C%20big%5C%20data%5C%20important,can%5C%20increase%5C%20revenue%5C%20and%5C%20profits (cit. on p. 5).

[4] *How Big Data Can Help You Do Wonders In Your Business*. URL: https://www.simplilearn.com/how-big-data-can-help-do-wonders-in-business-rar398-article (cit. on p. 6).

[5] *Big data architecture style*. URL: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/big-data (cit. on p. 7).

[6] Inês Araújo Machado, Carlos Costa, and Maribel Yasmina Santos. «Data Mesh: Concepts and Principles of a Paradigm Shift in Data Architectures». In: *Procedia Computer Science* 196 (2022), pp. 263–271. DOI: 10.1016/j.procs.

2021.12.013. URL: https://doi.org/10.1016/j.procs.2021.12.013 (cit. on pp. 8, 14, 15).

[7] Wided Oueslati and Jalel Akaichi. «A Survey on Data Warehouse Evolution». In: (2010). DOI: 10.48550/ARXIV.1012.1565. URL: https://arxiv.org/abs/1012.1565 (cit. on p. 8).

[8] Zhamak Dehghani. *Data Mesh: Delivering Data-Driven Value at Scale.* Sebastopol, CA: O'Reilly Media, Inc., 2022 (cit. on pp. 10, 13, 17, 18, 20).

[9] *What is Azure Data Factory?* URL: https://learn.microsoft.com/en-us/azure/data-factory/introduction (cit. on p. 23).

[10] *Pipelines and activities in Azure Data Factory and Azure Synapse Analytics.* URL: https://learn.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities?tabs=data-factory (cit. on p. 24).

[11] *Datasets in Azure Data Factory and Azure Synapse Analytics.* URL: https://learn.microsoft.com/en-us/azure/data-factory/concepts-datasets-linked-services?tabs=data-factory (cit. on p. 25).

[12] *Pipeline execution and triggers in Azure Data Factory or Azure Synapse Analytics.* URL: https://learn.microsoft.com/en-us/azure/data-factory/concepts-pipeline-execution-triggers (cit. on p. 26).

[13] *Integration runtime in Azure Data Factory.* URL: https://learn.microsoft.com/en-us/azure/data-factory/concepts-integration-runtime (cit. on p. 28).

[14] Alan Bernardo Palacio. *Distributed Data Systems with Azure Databricks.* Livery Place 35 Livery Street Birmingham B3 2PB, UK: Packt Publishing Ltd., 2021 (cit. on pp. 29, 31).

[15] *Databricks architecture overview.* URL: https://docs.databricks.com/getting-started/overview.html (cit. on p. 30).

[16] *What is Power BI?* URL: `https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview` (cit. on p. 31).

[17] *What are paginated reports in Power BI?* URL: `https://learn.microsoft.com/en-us/power-bi/paginated-reports/paginated-reports-report-builder-power-bi` (cit. on p. 31).

[18] *Ultimate Guide to The Data Marketplace in 2023.* URL: `https://about.datarade.ai/data-marketplaces` (cit. on p. 48).

[19] *How to build an organization's data marketplace.* URL: `https://medium.com/the-future-of-data/how-to-build-an-organizations-data-marketplace-48e3a32fe3f6` (cit. on p. 49).