Funded by the European Union

PROCTWIN

D8.1: Description for software system architecture and security

Version 1.0

Due date	30.06.2025
Dissemination level	Public
Work package title	Data management concepts
Lead editor	BFI
Internal reviewer	MCL
Status	Final

This project has received funding from the European Union under grant agreement NUMBER —101178721— ProcTwin

Authors

Name	Partner
Ahmad Rajabi	BFI
Albert Fort	AQS
Ana López	GSW
Christoph Nölle	BFI
Daniel Scheiber (reviewer)	MCL
Gunnar Mathiason	HiS
Irene García	Celsa Opco
Orhan Gafurovic	SSAB
Petri Kannisto	BFI

History

Date	Version	Authors	Description
30.4.2025	0.1	Petri Kannisto Christoph Nölle Albert Fort Gunnar Mathiason Ahmad Rajabi	Initial version
2.5.2025	0.2	Ana López	GSW descriptions
14.5.2025	0.3	Petri Kannisto	Review and finalize multiple chapters and sections
16.5.2025	0.4	Petri Kannisto	Write inputs to introduction; write about data models and security to Chapter 8 (platform) and edit existing texts; write conclusion
6.6.2025	0.5	Christoph Nölle Orhan Gafurovic	Write missing requirements; resolve multiple comments throughout the document; add description of SSAB use case
12.6.2025	0.6	Petri Kannisto Christoph Nölle	Late-stage preparations for review
18.06.2025	0.7	Daniel Scheiber	Review of deliverable Identified Todos: - Consistent use of BE or AE (modelling vs modelling) - Complete abbreviation list - Check page numbering after p 32

			- Check for references (reference section contains 4 references, some numbering in text suggests 20 references) - Clarify on comments throughout the document
23.6.2025	0.8	Petri Kannisto	Resolve many of the comments and TODOs
26.6.2025	1.0	Ahmad Rajabi Christoph Nölle	Resolve remaining comments

Abstract

This report is deliverable D8.1 for the research project ProcTwin. It describes the architectural principles that will guide the more detailed design of the components required for data management and integration. The description includes, among other requirements, the overall design principles for security.

The architecture has the following aims: (1) provide the generic design principles for the data-driven integration platform, based on the tangible requirements for production optimization and (2) establish a high-level organization with the teams and roles required to support software development and component integration. Considering the organization, the approach is based on the Data Mesh concept to recognize the natural interfaces between the organizations and organizational units. On the other hand, the integration platform aims to support storing and accessing both historical data and message-based online communication in a scalable manner.

Table of contents

Glossary

Abbreviation	Meaning
AI	Artificial Intelligence
API	Application Programming Interface
BFO	Basic Formal Ontology
CAT	Data product catalogue (in this document)
CC	Continuous Casting
CFD	Computational Fluid Dynamics
DAC	Data access (in this document)
DIS	Data ingestion and storage (in this document)
DTR	Data transformation (in this document)
ERP	Enterprise Resource Planning
ETL	Extract-Transform-Load
FEM	Finite Element Method
GOV	Data governance (in this document)
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IDM	Information models and data models (in this document)
IoT	Internet of Things
IS	Interoperability and standardisation (in this document)
JSON	JavaScript Object Notation
LD converter	Linz-Donawitz converter
MES	Manufacturing Execution System
ML	Machine Learning
MON	Monitoring (in this document)
MQTT	Message Queueing Telemetry Transport
OCI	Open Container Initiative
PLC	Programmable Logic Controller
PSA	Performance, scalability and availability (in this document)
RDF	Resource Description Framework
REST	Representative State Transfer
RUN	Application runtime (in this document)
SAP	Systemanalyse Programmentwicklung
SDK	Software Development Kit
SEC	Security and access control (in this document)
SSH	Secure Shell Protocol
SQL	Structured Query Language
TLS	Transport Layer Security
UI	User interface
UTM	Unified Temperature Model
VPN	Virtual Private Network
XML	Extensible Markup Language

1. Introduction

1.1 Scope

Architecture design is paramount for systems that perform reliably and are scalable, maintainable, and secure at present and in the future. Without a proper architecture, there is a risk that even systems that fulfil the desired functionality cannot meet quality-related requirements or become too expensive to maintain and extend. On the other hand, a good architecture guides the developers as they work on the more detailed design. The architecture is based on the analysis of requirements. Although agile methods are common, such an analysis for the start is important to avoid later problems that often follow from a pure ad hoc development approach.

Because software development is iterative and the development of the system continues beyond the writing of this document, this document aims to specify only the basis, including the following:

- Practices to support software development.
- Generic software components for data storage, communication, and system integration.

1.2 Deliverable role within the project

Figure 1 (left) illustrates the scope of the deliverable within the project as well as the relationship to other software-development-related deliverables.

- D8.1 (public) builds upon the requirements of the system and the principles of the Data Mesh architecture, defining an overall architecture for the communication platform being developed in the project.
- Later, D8.2 (public) specifies the concrete components design and the related Application Programming Interfaces (API).
- Finally, D9.1 (public) describes the implementation of the platform.

Furthermore, Figure 1 (right) illustrates the scope of D8.1. Although the project aims at developing optimization methods and models, these are out of scope in D8.1. The models will, however, be connected with the communication platform, which enables the models to receive information from the data sources as well as communicate with other models.

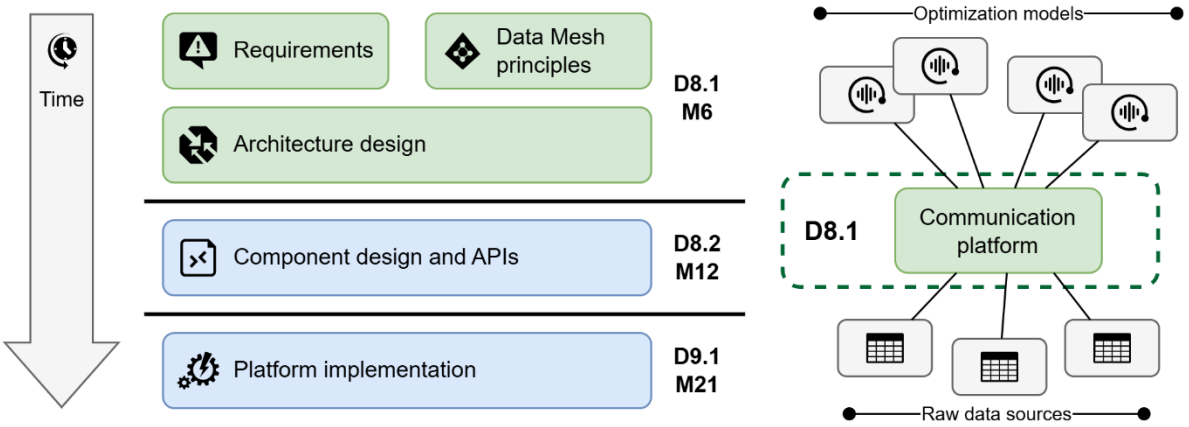


Figure 1. Left: D8.1 and the related deliverables. Right: D8.1 focuses on the communication platform, whereas the design of the models is a separate aspect.

Additionally, the following deliverables are related but focus on offline data storages instead of online data like this deliverable:

- D1.1 “Data management plan” (sensitive) focuses on offline data sets and their sharing.

- D10.1 “Early data storage” (sensitive) describes an initial version of the data storages used in the model development, particularly Machine Learning.

1.3 Outline

The remainder of this document is structured as follows:

- Chapter 2: architecture development process
- Requirements
  - Chapter 3: industrial use cases and distributed machine learning (ML)
  - Chapter 4: functional requirements
  - Chapter 5: non-functional requirements
- Architecture design
  - Chapter 6: Data Mesh principles
  - Chapter 7: main components of the platform
- Chapter 8: conclusion

2. Architecture design process

The development of the architecture is a process that includes discussions among the project partners to form a solid understanding of the project needs. Each step is explained below.

Collect requirements to serve stakeholders and external components. In this stage, the project partners discuss and utilize written documents to understand the landscape, goals, and restrictions related to the system being developed.

Formulate a requirements specification. Based on the requirements recognized, the partners write a requirements specification. The goal of these is to converge the varying requirements recognized. On the other hand, the partners will use their judgment to recognize implicit requirements that follow from the ones directly expressed earlier.

Design the architecture of the platform. Once the requirements have been formulated, the architecture is designed.

Iterative implementation (out of scope). It must be noted that the process is not sequential, but the stages occur in parallel and are re-iterated as needed. This is necessary due to the slow buildup of understanding, as a human being usually discovers the correct questions once some answers already exist. Despite the iterative process, this documentation follows a sequential structure for a better understanding, as suggested by Parnas & Clements[1]. Because this implementation exceeds the timeline of writing this document, it will be document in future deliverables:

- D8.2 Description of APIs, communication protocols, and interoperability
- D9.1 Data platform implementation

[1] Parnas, D.L. & Clements, P.C. 1986."A rational design process: How and why to fake it." IEEE Transactions on Software Engineering, vol. SE-12, no. 2, pp. 251-257. doi:10.1109/TSE.1986.6312940

3. Use cases and goals

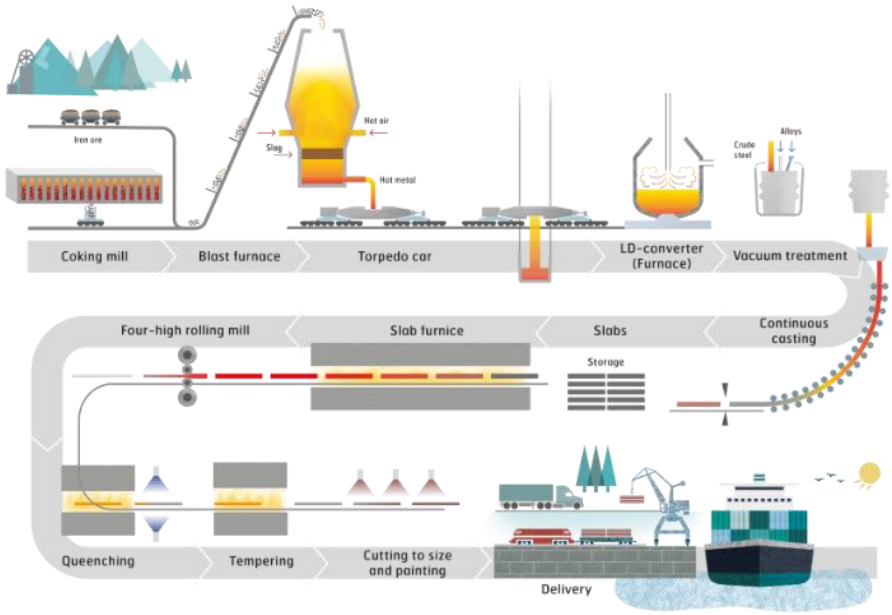
The motivation of ProcTwin project comes on one hand, from production optimization in the industrial use cases, and on the other hand, from generating scientific novelty with distributed ML. These factors are considered to establish the generic framework of the project. The following sections elaborate these requirements: SSAB use case, GSW use case, and distributed ML.

3.1 SSAB use case

SSAB in Oxelösund is world-leading manufacturer of high strength and energy-efficient steel, thereby increasing lifespan, lowering costs and reducing the environmental impact of end products throughout their lifecycle. Together with customers, unique solutions are developed that best utilize the characteristics of each steel type. Steel is made resistant to extreme wear, which is extra strong. The strength is an advantage in machine steels, making machines more efficient and protecting people.

Integrated production: integrated steel production is a steel production that covers the entire production chain from ore to finished steel and involves many different process steps.

- The coking mill converts bituminous coal into coke through dry distillation, which is used as fuel in the blast furnaces
- The blast furnace is fed around the clock with iron ore, coke and additives and hot metal at the temperature of 1,500 degrees Celsius is tapped at regular intervals throughout the night and day
- The hot metal is transported in a torpedo car to the Linz-Donawitz (LD) converter, where sulphur is removed, and the hot metal is converted into crude steel with a carbon content of less than 1 percent
- Various alloys are added in the after-treatment process
- In the continuous casting the liquid steel is converted into slabs. Ready-cut slabs have a maximum length of 11 meters and a maximum weight of 30 tons
- Slabs are rolled in the four-high rolling mill and by controlling the rolling force and temperature, the plates obtain various qualities
- Quenching and annealing endow the steel with its ultimate qualities, which depend on the type of steel that is to be produced



3.1.1 Plant environment infrastructure and data sources

The project will leverage data from various sources, including:

1. **Process Control Systems (e.g., SCADA, DCS):**
   - Format: Real-time data streams, typically in CSV or time series databases.
2. **Laboratory Results:**
   - Format: Periodic test results in Excel or PDF format.
3. **Manual Observations and Logs:**
   - Format: Structured logs or text files entered by operators.
4. **Sensors and IoT Devices:**
   - Format: High-frequency data in structured formats like JSON or API outputs.

We will ensure that data from all these sources is integrated into a centralized database or analysis platform for streamlined processing and decision-making. In terms of data sources, it is defined separately for each of the process:

- **Rolling mill**
  - **LEVEL 3**: Plate Vips, SSABs level 3 software, where information regarding slabs is available.
  - **LEVEL 2**: Structured Query Language (SQL) Server
  - **LEVEL 1**: Signal registers (Programmable Logic Controllers or PLCs) and information from Proview (SSABs open-source system for process control and automation) and information on the software system Iba™
- **Reheating furnace**
  - **LEVEL 3**: Plate Vips, SSABs level 3 software, where information regarding plates treatment chain and status is available,
  - **LEVEL 2**: SQL Server
  - **LEVEL 1**: Signal registers (PLCs) and information from Proview
- **Leveller**
  - **LEVEL 3**: Plate Vips, SSABs level 3 software, where information regarding plates planned treatment chain is available
  - **LEVEL 2**: SQL Server
  - **LEVEL 1**: Signal registers (PLCs) and information from Proview (SSABs open-source system for process control and automation)

3.1.2 Models

3.1.2.1 Temperature model

SSAB uses its proprietary software called UTM (Unified Temperature Model) when the plate temperature needs to be calculated while within furnace. The temperature model is design for plate finite difference model. It calculates temperature in five nodes in thickness direction.

As inputs, it takes physical measurement data, e.g. thermocouple readings, fuel flow, electrical power, constitute inputs for modelling temperatures of wall, gas, flames and emission & absorption data of the combustion products. These models form heat flux on the plate surface, which is calculated separately for upper and lower part of the furnace.

Linear interpolation between zone controlling thermocouples in continuous furnaces, virtual thermocouples, when necessary, start and end positions of the furnace.

Inputs: Geometrical data on furnace and plate, Fuel flow, Air flow, Electrical power, Fan speed, Temperature readings

Internal Models: Wall temperature model, Gas temperature model, Flame temperature model, Gas emission and absorption, Gas velocity, Heat flux on plate surface

Outputs: Incremental calculation of temperature for the plates in 5 nodes in thickness direction

The control systems shall call the UTM_FURNACETEMPMODEL function first in order to update the furnace model with the latest information before calling the UTM_PLATETEMPMODEL function to increment the plate temperature.

3.1.2.2 Quenching model

The quenching method uses individually adapted parameter settings related to the plate's quality and dimensions. SSAB owns all rights to the industrial parameters associated with quenching of its product portfolio.

3.1.2.3 Leveller model

The leveller model, LCM_MODEL, used for direction, consists of different steps such as Setup, Check and Drive. Each of them has its specific function. Working with the model requires general adaptation and revision of the subroutines so that they can be called and reused from both SETUP, CHECK and DRIVE.

3.2 GSW use case

3.2.1 Plant environment

The GSW Group[2] is Europe's leading producer of high value-added wire rod. It includes an electric arc furnace, a rolling mill and the Wire Works Division, which consists of six integrated downstream companies for the transformation of wire rod. Located in Santander (Spain), the GSW Group is also part of the CELSA Group™[3], Europe's leading circular steel producer, with 8 million tonnes of scrap recovered annually and 7 million tonnes of steel transformed and sold each year. It is a continuous process, with two main stages: the melting shop and the rolling mills.

In the melting shop, various scraps are used as the basic raw material, which is melted (together with other fluxes and ferro-alloys) in an electric arc furnace. This process results in a semi-finished product called a billet through continuous casting on 6 lines. The electric furnace has a capacity of 150 tonnes and uses a bottom pouring system to prevent the escape of slag. The billets produced have a square section of 180 mm and weigh between 2,500 and 3,000 kg.

The next stage is the rolling or laminating mill. This process begins with the entry of the billets obtained in the melting stage, which are heated in a gas furnace. Once hot, they are rolled into 140 mm square billets in a 4-pass roughing mill. Rolling continues in a continuous wire mill; the first 11 passes are common, and from the intermediate train the two rolling lines are separated by a total of 4 loop formers, depending on the diameter required.

Some of GSW's end customers are in the automotive or railway sectors, which require high quality standards for the steel supplied. As a result, high quality standards are required throughout the production process to minimise quality defects, which mainly originate in the melting process.

3.2.2 Software infrastructure and data sources

GSW has several data sources to allocate data and information (i.e., signals and metrics), which evokes the need for an edge architecture. Because there is no central repository to store and analyse the huge

[2] https://globalsteelwire.com/en/
[3] https://www.celsagroup.com/en/

amount of data generated, one main goal in this project is to deploy the edge architecture to process data closer to their original source rather than storing in centralized data centers or the cloud. This approach has several advantages:

1. **Reduced latency:** By processing data locally, response times are minimized, which is crucial for real-time applications like, in our case, industrial Internet of Things (IoT) systems.
2. **Bandwidth efficiency:** It reduces the amount of data that needs to be transmitted over the network, improving bandwidth efficiency.
3. **Enhanced security and privacy:** Keeping data close to its source reduces the risk of exposure during transmission.
4. **Cost-effectiveness:** It lowers the costs associated with sending large volumes of data to remote data centers.

In terms of data sources, we can define them depending on the part of the process. Please note that the numbering of the levels differs from the usual automation pyramid.

- **Meltshop**
  - **LEVEL 3**: Enterprise Resource Planning (ERP) with "Systemanalyse Programmentwicklung" (SAP); business data
  - **LEVEL 2**: Structured Query Language (SQL) Server; information available for steel grade, shift, and heat number; some aggregated data
  - **LEVEL 1**: signal registers (Programmable Logic Controllers or PLCs) and information on Iba system, other equipment (the Mold oscillation monitoring system KTI™, Condrive)
- **Rolling mill**
  - **LEVEL 3**: ERP with SAP; business data
  - **LEVEL 2**: Tracking information for steel grade, shift, heat number, billet number, timestamps
  - **LEVEL 1**: Signal registers (PLCs) and information on Iba system, other equipment (iSend)

3.2.3 Models

3.2.3.1 Existing models

The Reheating Furnace Model is a 2-dimensional Finite Element Method (FEM) model providing the temperature of steel billets. The software is already installed and used productively by industrial GSW. Online connectivity to all data sources is available and output data is stored in local database.

3.2.3.2 Models planned (as far as is known)

During the first phase of the project, individual models of specific stages of the steel manufacturing process at the steelworks will be developed:

- Machine learning model for the **continuous casting** (CC) process to predict billet quality parameters, such as bulging and rhomboidity, from process data, including data generated by the new sensor systems and the solidification model. In a later step we also aim to predict the impact of the casting process on quality parameters of the final wire rod, such as number and severity of surface defects. The model is intended to run in near real-time.
- **Solidification Model for Continuous Casting Mould** models flow and temperature fields as well as formation solidified shell in CC mould. It is based on physical laws – a Computational Fluid Dynamics (CFD) model. It cannot run directly in real time but it can be used to generate an online surrogate model.

- **Hot rolling** of long products and profiles in the rolling mill. The model cannot be run in real time. The program can potentially take parameters from the process in real time, but the simulation time and processing of results may be longer than the production time.
- Model for evolution of **material properties** in the hot rolling process with a physical basis, if possible, or with a semi-empirical model otherwise.

The intended models are instances of development towards a fully dynamic and distributed AI model. Each such an instance is a refinement step of the previous, in the following order:

1) A non-linear regression model predicting targets of each user case, including data from one, then several machine operations. The purpose is to investigate the information content in data available from subsequent operations, to understand how including multiple operational data can increase prediction accuracy. This results in a global model including data from multiple operations and a common target.
2) A distributed model where each operation's data set is represented as a separate data entity in a communication/data infrastructure, and where local sub models learns primarily the local operation and local operation target, then interacts with other sub models for a common and emergent model that predicts the common target. Data from each machine is held as separate entities.
3) A fully dynamic and distributed model where sub model (agent) coordination and aggregation are done through the data mesh, by publishing data and capabilities of each agent in that, both for agent input and output data and inter sub model updates.

3.3 Distributed machine learning

Distributed ML combines the concepts of distributed systems and ML for a more capable, more adaptable optimization. In a distributed system, there are multiple nodes that communicate to solve some common global goal. Also, each node works locally to serve both local usage and the global goal. In a production system, the machines can be seen as distributed operations. These operations can be modeled by ML and by other models, and in combination. There are local models for each production machine that optimizes for the local production target and output quality. However, the common output from a production line depends on both the local optimization and the global optimization of production towards the target. To achieve this, we use distributed ML where local Artificial Intelligence (AI) agents (i.e., local nodes) collaboratively optimize both for the local and the global production target.

3.3.1 Data services

There is a need for a communication infrastructure to share model information that the local AI nodes need to share based on what their algorithms find to be useful for sharing. The local ML is based on the local data from each machine, so each local processing should access the local data effectively. The global ML is based on model exchange over the information infrastructure. What is exchanged is typically an aggregation of local data, or model derivatives, considered essential for the collaborative production optimization.

Thus, the following essential functions are needed from a communication infrastructure:

a) Fast, efficient access to local data at each node.

b) Effective model knowledge exchange between nodes.

In addition to this, there is a need to look up data which is available at other nodes, so there is a need for a data publication system to express data availability and to provide data access.

3.3.2 Computation services

This project will require two different computation setups for ML model training and usage:

a) Computation is performed **offline** from production data flow, and data that is used is already available in the storage structure as historic data. Computation takes place both locally and in collaboration among the nodes. This is typically done for a certain period of historic production. This data must have version handling. The models must be computed for different periods of production, and the data can be revised for each such period. This requires that different versions of the data are traced. Also, historic data should be integrated into a live model.
b) Computation is performed **online** with data streaming from production. The model is trained on data that becomes available and learns while production runs. This is a setting in the live final system. There exist versions of the data available in an online system, since the data availability may change when, for instance, new sensors are added as a result of findings using online or offline model analysis.

For the preparation of data to fit model training, there must be a computation for data pre-processing. This is done both for off-line mode of training and online model training.

The data versioning of historic data is required for offline production analysis using ML. Having different models trained for different periods allows to find different behaviours in different production periods. Some periods may represent good production while other periods may represent problematic production. Training different models using data from different periods, the reasons for the differences can be explored by comparing these models. Any data-API-related updates need to be tracked in online ML to enable incorporating extended data and sensor availability.

4. Functional requirements

Functional requirements describe *what* the desired solution should do but not *how*. On the other hand, functional requirements exclude qualitative aspects (such as security and performance) and resource consumption. Such a separation enables a systematic approach to justify the design choices during the detailed design of the solution, which is a separate aspect. The functional requirements build upon the general requirements from the industrial use cases.

4.1 Data ingestion and storage (DIS)

Various data sources must be connected to the data platform, mostly from the industrial automation systems of the steel plants. This includes in particular level 2 and level 3 (e.g., Manufacturing Execution System or MES) automation systems.

DIS-1. **Connectivity of data sources.** The data platform must enable data ingestion from the identified relevant source systems.
DIS-2. **Extensibility.** The platform must allow for the simple integration of new data sources.
DIS-3. **Data format support.** The platform must be capable of storing different types of data, such as structured data, timeseries, images or video files, in appropriate storage technologies, such as databases (e.g. SQL or NoSQL databases, column stores, etc.) or object storage.
DIS-4. **Legacy integration.** It shall be possible to integrate existing storage solutions, such as databases or file systems, into the data platform without the need for data duplication.

4.2 Data transformation (DTR)

Data transformation enables the data to be adapted for the data consumers. The following requirements have been identified:

DTR-1. **Separation after refinement degree.** There should be separate spaces for raw data, curated data & business data.
DTR-2. **ETL pipelines.** Data processing with case-specific Extract-Transform-Load (ETL) pipelines is supported, recognizing the arbitrary needs of the optimization models.

4.3 Data access (DAC)

Data stored in the data platform must be accessible to different kinds of consuming applications, such as business intelligence tools or simulation models.

DAC-1. **Near real time access.** For online data, there must be a possibility to deliver the data in near real time to enable timely reactions in the optimization and monitoring tools. This does not mean that strict real-time capabilities are required.
DAC-2. **Historical data.** The platform must be able to store historical data and provide an access to this. This batch data interface should support filters on the time interval and other properties to enable efficient data retrieval.
DAC-3. **Reproducibility of model training.** The platform must enable the creation and versioning of datasets from selected sources, so that a model trained on a specific subset of data can be reproduced later when new data has been ingested.

4.3.1 Information models and data models

While the platform must allow for simple integration of new data sources, it shall also offer the ability to provide rich metadata describing the data stored. Metadata shall be applicable to all types of data,

including raw, curated and business data, and also describe the relationship between data from different domains, different data sources and between different types of data.

IDM-1. **Data Model.** The platform shall enable the storage of a data model describing relevant entities, the relationships between entities, and the meaning of the data stored in the platform.

IDM-2. **Model hierarchy.** A hierarchy of models shall be supported, with a generic high-level data model at the top that defines common aspects such as material traceability, location information, or time, and domain-specific models for the different industrial processes, such as continuous casting or hot-rolling.

IDM-3. **Annotated and compact representations.** The platform shall offer the options to retrieve either rich annotated data or more easily processable "flat data", at the discretion of the data consumer. For an example on how this can be realized, confer the NGSI-LD normalized and simplified representations[4].

4.3.2 Application programming interfaces (API)

To enable a data access for external applications, there must be well-defined, documented APIs. These provide a harmonized data access layer independent of the underlying storage technologies. The APIs shall be based on well-established, standardized technology, such as Representative State Transfer (REST) over HyperText Transfer Protocol (HTTP), websockets, and/or Message Queueing Telemetry Transport (MQTT). To support the distributed state-of-art developments methods, the favoured API type is "web API" rather than a component-to-component API for local, in-computer connections.

To support the development of applications tailored to the industrial domains, a software development kit (SDK) for the Python programming language shall be provided alongside the platform, which internally retrieves and writes data via the APIs. The SDK removes the burden of contacting the APIs directly from the developer and ensures that software best practices are followed. The SDK shall leverage the domain models (see Metadata section above) to foster specialized application development.

API-1. **API access.** Application programming interfaces shall be provided for accessing the data stored in the platform and for writing data. Specific APIs for specific types of data, such as timeseries, shall be provided.

API-2. **Streaming APIs.** Besides APIs for batch data access, methods for accessing streams of data shall be provided.

API-3. **SDK.** A software development kit (SDK) for the Python programming language shall be provided.

4.3.3 Discoverability and data product catalogue (CAT)

Discoverability refers to the capabilities provided that enable the user (for instance, the app developer) of the platform to identify the relevant datasets, to search for data sources according to different criteria, etc. Tools that enable discoverability include catalogues and data classification, for instance, by means of a data model.

CAT-1. **Data product catalogue.** The platform should provide a catalogue functionality with an overview of the provided data products, including information about data lineage, data type, domain, ownership, refinement degree (raw, curated, or business data, cf. DTR-1), data location and access options.

[4] ETSI : NGSI-LD API v. 1.8.1 : https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.08.01_60/gs_cim009v010801p.pdf

CAT-2. **Metadata inventory.** Similarly to the data catalogue, an inventory of available data models/metadata-sets shall be provided.

CAT-3. **Data exploration.** Alongside the data catalogue, a data exploration playground can support the initial analysis of available datasets.

4.4 Data governance (GOV)

Data governance refers to a structured approach of managing data during its complete life cycle, covering aspects such as ownership of data, data security, storing, processing and disposing of data, and others.

GOV-1. **Data ownership.** The platform must support the concept of data ownership. Data owners should be able to configure data connections, access rights, and metadata.

4.5 Application runtime (RUN)

A container orchestration system shall be provided for hosting applications interacting with the data platform, such as simulation models. The container platform shall support containers compatible with Open Container Initiative (OCI), allowing to run applications written in all major programming languages.

RUN-1. **Container orchestration.** Allow for the simple deployment of containerized applications.

RUN-2. **Distributed data processing runtime.** For the efficient processing of large sets of data a distributed data processing runtime, such as Apache Spark, would be useful.

4.6 Monitoring (MON)

MON-1. **Monitoring dashboard.** The platform must provide a monitoring dashboard for admin users that displays relevant status metrics, including CPU and RAM usage, node availability, and received requests. Specific functionality for data owners (cf. GOV-1) should include access statistics for datasets.

MON-2. **Alerting.** The platform should provide a configurable alerting mechanism that informs a user in case of anomalous operating conditions.

MON-3. **Monitoring extensibility.** Both the dashboard and alarming system should be extensible, in that users can create customized dashboards and alarms. Furthermore, it shall be possible to configure additional, application-specific metrics.

4.7 User interfaces (UI)

UI-1. **Browser-based user interface.** The platform must provide a way for applications to deploy a browser-based user interface. End users connected to an appropriate enterprise network must be able to access these web sites in the browser.

UI-2. **Data owner interface.** Dedicated browser-based user interfaces for data owners should be provided by the platform for relevant administration functions, such as the delegation of access rights, configuration of metadata, and possibly others.

5. Non-functional requirements

The non-functional requirements are mainly related to the *quality* of the system and, thus, do not describe any features or functionality. Instead, these requirements restrict *how* the system can be implemented to reach the desired qualities.

5.1 Security and access control (SEC)

SEC-1. **Data location control.** The owner of the data must be able to control where the data is located and utilized geographically.

SEC-2. **Confidentiality.** It must be possible to preserve the confidentiality of the data being processed. Practically, the data utilized in the system should either be stored and processed in a secure "sandbox", or if delivered elsewhere, encrypted connections must be applied.

SEC-3. **User authentication.** The system must provide a mechanism to authenticate the users to enable access control.

SEC-4. **Authorization.** Authorization should be applied to the extent required by the use cases to control which user or user roles can do what (for example, administrator, data owner, developer, or end user). This administration of access rights to specific data products should be managed by the respective data owners.

SEC-5. **Data availability and integrity.** Depending on the use case, there can be requirements for data availability and integrity, including backups or redundant systems.

5.2 Performance, scalability and availability (PSA)

PSA-1. **Computation and memory.** The infrastructure must provide a sufficient performance required for the computational needs, including both CPU and memory capacity. Sufficient performance is defined by the use-case specific applications and thus cannot be defined precisely upfront. In order to satisfy this requirement, it should therefore be possible to add more compute resources to the platform on demand.

PSA-2. **Network.** The network must provide a sufficient capacity for the software components to exchange data.

PSA-3. **Scalability for storage.** The architecture must enable scalability to growing data amounts in terms of data storage. The storage must be either large enough or extensible.

PSA-4. **Scalability for data transmission.** The architecture must scale to growing data amounts in terms of transmission.

PSA-5. **Scalability to network node count.** The architecture must scale to a growing overall number of network nodes exchanging information. For instance, point-to-point connections should have an alternative, such as a message broker, not to cause unnecessary network traffic that could result from broadcasting.

PSA-6. **Scalability to data consumer count.** The architecture must offer a method for a data source to deliver data to an arbitrary number of data consumers without sacrificing the performance of the data source. For instance, point-to-point connections should have an alternative, such as a message broker, not to overwhelm any data source if the number of clients grows.

PSA-7. **Availability to end users and developers.** The platform should be available to the end users and developers without major outages. When an outage is detected, it should be possible to resolve the situation in a timely manner, preferably in a few hours instead of days if a part of system becomes defect.

5.3 Interoperability and standardisation (IS)

IS-1. **Knowledge reuse.** To reduce redundant work, existing standards, industry standards, and/or open standards should be utilized where possible.

IS-2. **Interoperability with existing specifications.** To promote the interoperability of the results with project-external instances, existing standards, industry standards, and/or open standards should be utilized where possible.

IS-3. **Interoperability support.** The interoperability between the components should be supported by the architecture, including not only interface descriptions but also a process to support the related maintenance.

6. Data Mesh principles

As a design decision, the development work will be governed with the Data Mesh approach. This chapter describes the underlying principles and what these mean in the context of the data platform.

6.1 Background

The Data Mesh helps in managing, developing and maintaining the information systems of an organization. The Data Mesh can be seen as an ecosystem of independent yet interconnected Data Products[5] (Perrin & Broda, 2024, p. 15). However, although data, its sources, and its consumers have a high priority, the organization as well as the related development and maintenance processes are equally important. No part of the system or organization should be the constant bottleneck in development. Figure 2 illustrates the key aspects of the Data Mesh: organization, people, policies, supportive services, and data products.

Figure 2. The key aspects of the Data Mesh.

In this document, the architecture builds upon the following list of aspects for the Data Mesh ecosystem. The list is taken from[6] but the meanings are explained as appropriate for this document.

- **Data Products.** The ecosystem consists of Data Products that represent the various data-oriented services and service consumers.
- **Domains and Domain Teams.** Each Data Product belongs to a Domain and is owned by a Domain Team, which translates to a de-facto or de-jure organizational unit. In this project, a Domain Team can span over multiple legal organizations.
- **Enabling Team.** The development and maintenance of the concept must be supported by people. The Enabling Team helps the Domain Teams in development tasks.
- **Governance Group.** The Governance Group maintains the common policies of the Data Mesh, such as security, privacy, interoperability, and documentation.
- **Data Platform Team.** The Data Platform Team develops and maintains common services for the Domain Teams. These can include, for instance, data storage, message brokering, and Data Product catalogues.

It is notable that due to the lightweight organization of the research project, some project participants likely belong to multiple teams. Still, it is considered important to recognize the existence of the teams as providers of explicit roles.

[5] Perrin, J.-G. & Broda, E. 2024. "Implementing Data Mesh." O'Reilly Media, Inc., first ed.
[6] "Data Mesh Architecture." No date. https://www.datamesh-architecture.com/ [Visited on 21 Mar 2025]

6.2 Data Products and Domains

The Data Product represents a software component in the Data Mesh. We interpret that it is called a "product" now that it is supposed to provide:

- *Added value* in the form of data or information.
- *Connectivity* with other items (or Data Products) even in the long run.

Each Data Product must comply with a **Data Contract**. This describes the Application Programming Interface (API) to the Data Product. The contract should describe what is necessary for interaction.

The Data Contract can be created either **"top down"** or **"bottom up"**, or as a combination of these two. A pure top down approach means that the Data Contract is a based on an existing API specification. In contrast, a bottom up contract results when the Data Product is developed without modelling the API first.

Each Data Product is owned by a Domain that develops and maintains the Data Product. The Domain can span multiple organizations. For instance, two organizations can maintain an optimization model (a Data Product) collaboratively and, thus, logically form a Domain.

6.3 Enabling Team

Because software development is a people-intensive process, a well-performing architecture must not focus merely on software tools. The development process will progress even without any guidance, but only active guidance can guarantee that the principles of the architecture are followed.

The development processes of Domains can receive support from the Enabling Team at least in the following ways:

- Spreading awareness of the Data Mesh and its benefits.
- Person-to-person guidance and support.
- Code examples to help in the development of Data Products.
- Maintaining and communicating best practices.

6.4 Governance

The main aspects of governance in this project are:

- Security Policy and Privacy Policy.
- Interoperability Policy.
- Documentation Policy.

Security builds upon the requirements of each use case. Privacy can be seen as a part of security. In this research project, now that personal data is likely not processed, the importance of the privacy aspect *might* be lower than other confidentiality requirements. Overall, in the conventional "CIA" triad, confidentiality is likely the most important item, with integrity considered an inherent feature of the data processing tools and availability relying on personal backups as well as ICT support teams.

Interoperability is considered the ability of two components to interact in a meaningful way. Although this can cover multiple aspects, it is considered that the system requires attention at least on *semantic interoperability* through data models. Depending on the case, it may be necessary to document complex interaction workflows to enable *functional interoperability*. On the other hand, the lower levels, which include *technical interoperability*, are considered to be solved once a suitable communication protocol is selected.

Documentation is a tool for communication, providing information without a direct person-to-person connection. Its purpose is to cover what is necessary to develop and maintain the components of the

system. The perfect, optimal documentation provides exactly what is needed, nothing more and nothing less, and it should be correct and up to date. Excessive or poorly structured documentation can hamper finding the most essential information and add difficulty to maintenance. No perfect documentation exists, but even bad documentation is usually better than no documentation at all.

To enable governance, a Data Mesh Handbook has been written for the project and will be maintained throughout. It describes the policies and the organizational structure, forming the principles of the Data Mesh.

6.5 Software and components in Data Mesh

Although the Data Mesh can be seen as a management concept, it has a solid connection with the concrete component architecture of the platform. Regarding the components, the concepts Data Product and Self-serve Data Platform are important.

The Data Products include the actual software components developed and connected to the platform. They generate added value from data. They are including the optimization models as well as the ETL components to refine the data as it is received.

The Self-serve Data Platform provides tools to support the developers in their work. It has two main parts:

1. **Data platform** includes software services for the online operation of the components. These can include, at least, data storages, data brokering, data processing, monitoring, and container orchestration. This aspect is explained in more detail in Chapter 7.
2. **Development services** are tools not directly connected to the software during execution. These tools are used in the background to help in the development work. They can include at least version control, Data Product catalogue, and documentation storage.

Figure 3 illustrates the position of the components within the Data Mesh as explained in the earlier paragraphs.

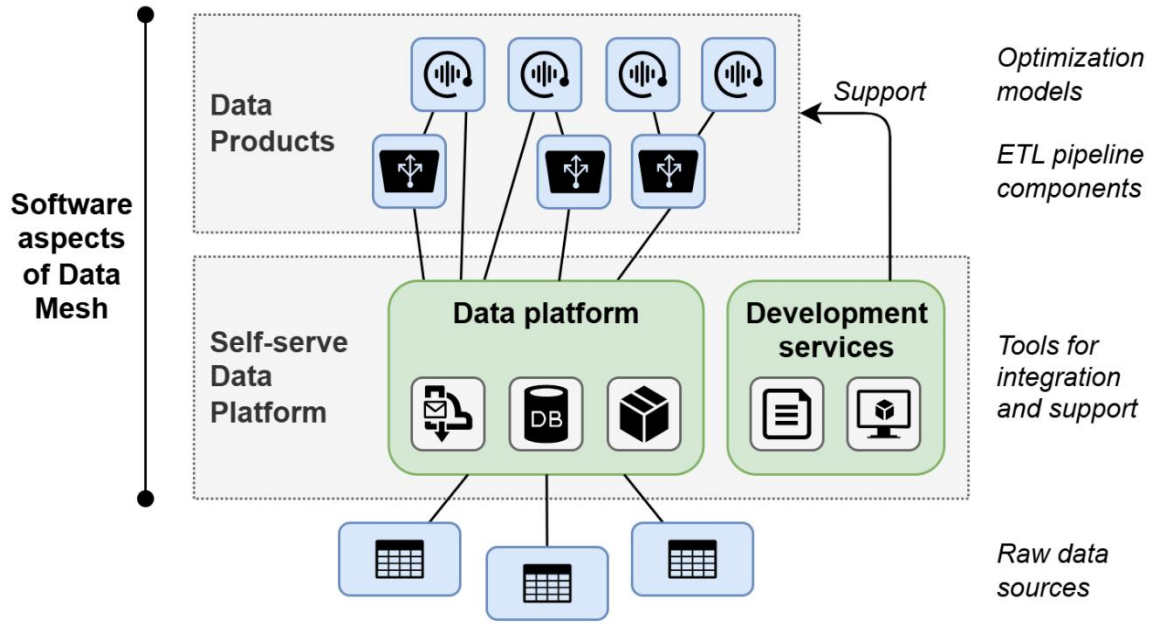


Figure 3. The position of software components in the Data Mesh, excluding any pure management aspects.

7. Main components of the platform

Although the final data platform will gain its final form thanks to the inputs from optimization system designers, the overall goal enables a generic platform design. This chapter will detail the features recognized. Despite the generic nature of the components, some must consider case-specific requirements.

7.1 Overview

The design of the data platform is based on a microservice architecture. This will contribute to the scalability of the resources and will enable the separation of physical or cloud resources (hardware) from the software components used.

Figure 4 illustrates the platform design, the 3 key sections being:

- Platform Tools: Architecture of all the components that will make up the Data platform and how they will interact with each other.
- Orchestration Tools: Orchestrator of all the components that will make up the Data Platform.
- Monitoring Tools: Monitoring system for the entire Data platform infrastructure.

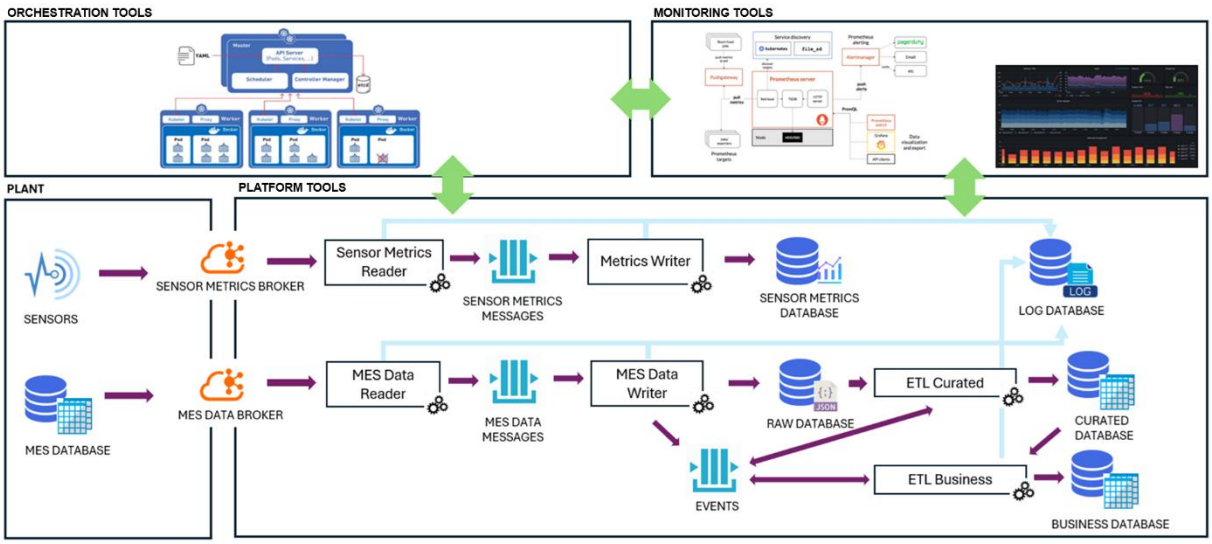


Figure 4. Platform components.

The following sections will discuss the aspects included in each section of the architecture.

7.2 Data storages

7.2.1 Sensor Metrics Database

This is a temporal database, for time series, storing all raw data from the metrics received from the various sensors. The data are organized according to the metadata as configured in the platform.

7.2.2 Raw Database

This raw database is an unstructured database (i.e., document oriented) that stores all raw process data from the messages and/or events received from the plant's production system. Presumably, this database will be our most extensive source of received data, but it is not recommended to use this database directly as the source for application consumption due to the low degree of refinement.

7.2.3 Curated Database

This is a structured (SQL) database where the raw data registered in the Raw Database will be stored in a relational format according to the messages. It will validate that the content received meets the minimum constraints to accept this information as valid.

7.2.4 Business Database

This is a structured (SQL) database where the data that has been registered and previously curated in the Curated Database will be stored and recorded with the already modeled structure in order to consume this data in an aggregated or converted form according to the needs of the applications.

7.2.5 Log Database

This database enables storing logs generated from the activity of the deployed Data Products, providing a structured way. The logs can be viewed with the Monitoring System.

7.3 Data brokering

The data brokers generic products are therefore independent of the case-specific data being delivered. Still, these must be configured for the use cases at least considering any topic names if relevant.

7.3.1 Inbound Broker: Sensor Metrics Broker

This is a data ingestion broker, where the plant sends temporary data generated by all sensors. This broker is configured to receive lightweight messages at a very high frequency. This component represents a coupling point between the automation system (Operational Technology) and the data platform, and it must be ensured that any issues with the platform do not impact the plant's production processes.

7.3.2 Inbound Broker: MES Data Broker

This is a data ingestion broker, where the plant sends data generated from its production process considering the higher levels of control, particularly MES and/or ERP. This broker is configured to receive complex messages at a lower frequency compared to the sensor data. However, it ensures that every message sent from the plant is received by the broker. Like for the Sensor Metrics Broker, it must be ensured that any issues with the platform do not impact the plant's production processes.

7.3.3 Message broker: Sensor Metric Messages

This message broker will temporarily store all sensor-generated temporary messages received from the plant. This component is crucial to ensure that every message received by the ingestion broker is correctly processed, guaranteeing that no message is lost between the reading process from the input and the writing to the database.

7.3.4 Message broker: MES Data Messages

This message broker will temporarily store all messages containing production process data received from the plant. This component is crucial to ensure that every message received by the ingestion broker is correctly processed, guaranteeing that no message is lost between the reading process from the input and the writing to the database.

7.3.5 Message broker: Events

This message broker will temporarily store events generated in the data logs of the various process databases (Raw, Curated, and/or Business Database). These events can be consumed by other processes that need to execute as close to real-time as possible. This component is essential to ensure that production data processing occurs as close to real-time as necessary, whenever this is relevant.

7.4 ETL pipeline system

Although the ETL infrastructure can be generic, the actual components must consider the requirements of the use cases. This means that the ETLs are, inevitably, classified as Data Products in the Data Mesh.

7.4.1 Reader Service: Sensor Metrics Reader

This service is responsible for consuming all messages received in the Sensor Metrics Broker and temporarily storing them in the corresponding queue of the Sensor Metric Messages broker, adding the necessary metadata.

7.4.2 Reader Service: MES Data Reader

This service is responsible for consuming all messages received in the MES Data Broker and temporarily storing them in the corresponding queue of the MES Data Messages broker, adding the necessary metadata.

7.4.3 Writer Service: Metrics Writer

This service is responsible for consuming all messages temporarily stored in the Sensor Metric Messages broker and registering them in the Sensor Metrics Database (time series) according to the metadata and the originating queue.

7.4.4 Writer Service: MES Data Writer

This service is responsible for consuming all messages temporarily stored in the MES Data Messages broker and registering them in the Raw Database (unstructured/document database) according to the metadata and the originating queue. For each message consumed and correctly registered in the database, this broker will create a new event, which will be recorded in the Events broker.

7.4.5 ETL Curated

These services are responsible for consuming, validating, and transforming data from the Raw Database to the Curated Database. These services can also generate events in the messaging broker Events. They can also be configured to execute every time they receive an event in the same Events broker.

7.4.6 ETL Business

These services are responsible for consuming, validating, and transforming data from the Curated Database to the Business Database. These services can also generate events in the messaging broker Events. They can also be configured to execute every time they receive an event in the same Events broker.

7.5 Container orchestrator

The container orchestrator is an environment to deploy and execute containers and to set up the required communication networking. It is responsible of automating the operational effort required to run containerized workloads and services. It automates various aspects of the containers' lifecycle, including provisioning, deployment, scaling, networking, load balancing, traffic routing, and more. By automating these tasks, container orchestration simplifies the management of containers at scale and ensures their efficient operation within a distributed environment. This platform can also automatically spin containers up, suspend these, or shut these down when needed.

Container orchestration mediates between the apps or services and the container runtimes and performs services management, resource management, and scheduling.

7.6 Monitoring

The Monitoring System is responsible for monitoring and analysing the performance, availability, and health of various components, such as servers, networks, applications, databases, and cloud systems. The main functionalities are:

- **Real-time monitoring**: They detect issues or anomalies instantly.
- **Alerts and Notifications**: They send alerts in case of system failures or outages.
- **Performance Analysis**: They collect data to identify bottlenecks or performance degradations.

7.7 Security aspects

The security measures will follow the rules and policies of the data owner, that is, the industrial use cases GSW and SSAB. The concrete measures will become clear during the project, but multiple mechanisms can be anticipated.

First, it is expected that, as an effective security measure, the online systems will operate in isolation instead of the open Internet. This means that the software components operate behind a firewall, which makes is much more difficult for a malicious actor to access the system. Therefore, a Virtual Private Network (VPN) or a similar mechanism is necessary, unless the developers can work on-site. VPN will enforce both connection encryption and user authentication.

Second, the approach can be strengthened with a defence-in-depth mechanism by applying multiple security layers. This means that, in addition to the presumed VPN, any access to software will perform an additional user authentication and possibly encryption. These measures can come from, for instance, Secure Shell Protocol (SSH) or remote desktop technologies, depending on the server operating system which is chosen. Additionally, any web user interfaces or APIs shall enforce connection encryption with Transport Layer Security (TLS) to enforce encryption with HTTP or MQTT, together with user authentication and access control checks. Which measures are to be used depends on both the application and the security policy of the enterprise.

7.8 Data Models

This section describes the first draft of the metamodel (for data modelling) to be adopted by ProcTwin. It is expected that the metamodel will be refined during the coming months of the project, to reach a first stable state with the publication of D8.2 - *Description of APIs, communication protocols, and interoperability,* scheduled for December 2025.

The metamodel prescribes common modelling principles for the development of the ProcTwin data models. In general, a distributed approach will be chosen, with domain-specific models developed by the corresponding domain teams. Common modelling principles applicable to all of these domain-specific models include the basic definitions of terms, such as *entities*, *properties*, or *relationships*, constraints on the set of allowed names, terminology to be adopted, etc.

7.8.1 State of the art

7.8.1.1 Graph models

Common metamodels found in the literature are *RDF (Resource Description Framework) graphs* and *property graphs*. These two are quite similar, both of them allowing for graph-like representations of data in the form of nodes, vertices and values, as illustrated in Figure 5. Non-value nodes of the graph are usually called *resources* or *entities*, and they represent physical (*real world*) or abstract entities. In our example, a billet and a continuous casting machine.

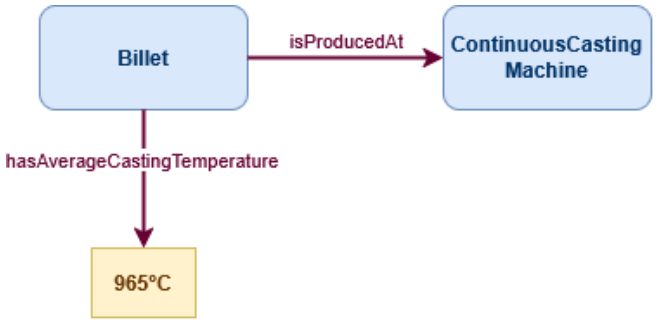


Figure 5: Example of a graph representation consisting of two nodes, a value, a relationship (vertex between nodes) and a property (vertex between node and value).

The main difference between the two graph models is that property graphs allow for more flexibility by permitting *properties* (vertices between a node and a value) and *relationships* (vertices between two nodes), collectively called *attributes*, to possess additional properties and relationships themselves[7]. These higher-level attributes can be thought of as *metadata*, which further qualifies the main attribute. This is illustrated in Figure 6, which differs from Figure 5 by the addition of a relationship *isMeasuredBy*, which further qualifies the *hasAverageCastingTemperature* property. Furthermore, property graphs relax some of the constraints imposed by RDF regarding the existence of multiple relationships of the same type between two nodes, and thus simplify the modelling of many-to-many relationships between entities[8].

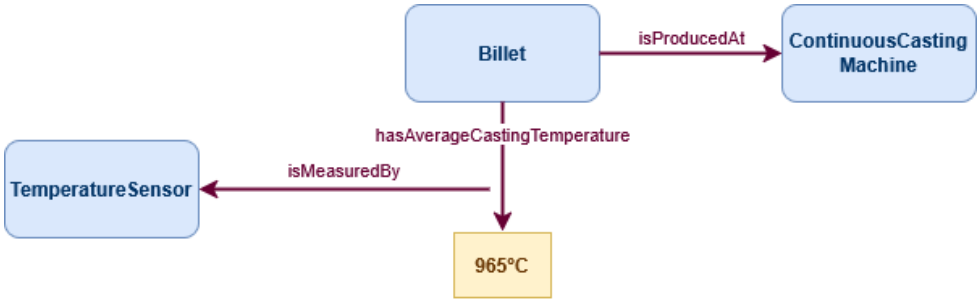


Figure 6: Example of a graph representation with a higher-level relationship (metadata) on the hasAverageCastingTemperature property.

7.8.1.2 Typing and Identification

A classification of the different entities is achieved by assigning *types*, or *classes*, to them. In the property graph model, types can also be assigned to properties and relationships. These types restrict the kind of attributes that can be assigned to an entity or attribute. For instance, a type *TemperatureSensor* may specify that instances have a field *value* (or alternatively, *reading, measurement*, etc.), of floating-point type. Class attributes can be mandatory or optional, usually it is good practice to define most attributes as optional. A flexible typing system should support:

- Multi-typing: instances can be assigned more than one type
- Sub-typing: types can extend other types (and multiple other types)

Recommendations on typing flexibility can be found for instance in[7], Appendix A.1.

[7] ETSI, « NGSI-LD Information Model », v.1.2.1 (2023): https://www.etsi.org/deliver/etsi_gs/CIM/001_099/006/01.02.01_60/gs_CIM006v010201p.pdf
[8] Neo4j, « RDF vs. Property Graphs: Choosing the Right Approach for Implementing a Knowledge Graph » (2024) : https://neo4j.com/blog/knowledge-graph/rdf-vs-property-graphs-knowledge-graphs/

In addition to types, every entity should be assigned a unique id. Many standards prescribe the use of specific formats for these ids, such as that they be valid URIs. NGSI-LD recommends the usage of entity identifiers of the form *urn:ngsi-ld:Person:28976543*, where the *Person* part is the type of the entity.

7.8.2 ProcTwin Metamodel

7.8.2.1 General approach

Since the focus of this research project is not on modelling principles, we simply adopt the well-established property graph model, as specified by the NGSI-LD information model.

In addition, there are many overarching aspects relevant to multiple or all domains, which should be handled in a unified way, such as location information, time-related aspects, composition of entities, or material tracing data. These cross-domain aspects are best specified in a common base ontology. Examples of existing attempts to specify such a cross-domain ontology include Basic Formal Ontology (BFO)[9] and the NGSI-LD cross-domain ontology[7]. Furthermore, the Common Core Ontologies may be relevant[10], which are based on the BFO.

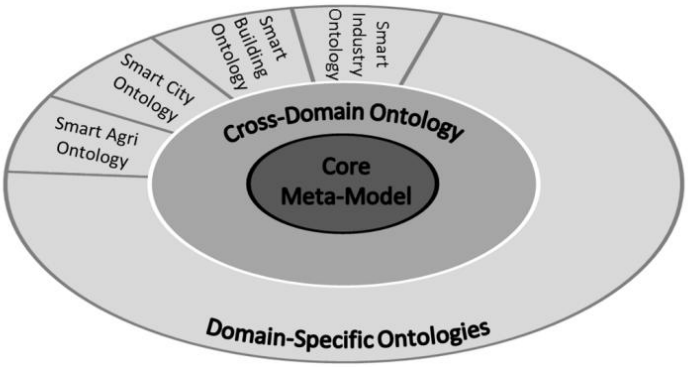


Figure 7: Schematic view of the layered ontology concept of NGSI-LD. From ref. [7].

7.8.3 Cross-cutting domains

In the case of the ProcTwin project, we can identify several cross-cutting domains of general relevance, including:

- **Process hierarchy:** industrial processes can be modelled at different resolutions; for instance, the hot rolling of some intermediate product typically consists of multiple steps including (re-) heating, pickling, rolling at multiple rolling stands, and potentially others. Depending on the requirements of a use-case, a coarser or a more fine-grained view must be adopted. Processes can be modelled at an abstract level, for instance as a generic hot-rolling process, and at a concrete level, referring to the actual realization of the hot-rolling process at a specific plant. The abstract view may be useful for representing common capabilities of different concrete processes, although to a certain extend it is also possible to provide such information using common classes for specific processes in the model. Whether or not the abstract view is useful for the purposes of the project is left to be determined here.
- **Equipment hierarchy:** similarly to the decomposition of a process into sub-processes, equipment can be decomposed into smaller parts, for instance, a hot-rolling line may include a reheating furnace, a pickling line, and multiple rolling stands, all of which in turn consist of multiple smaller parts, such as a burner, acid bath or rolling cylinders. Again, a model should support these different views of the equipment according to the needs of the use cases supported. Equipment models can refer to individual plants or machines, or they can refer to

[9] The Basic Formal Ontology : https://basic-formal-ontology.org/
[10] https://www.nist.gov/system/files/documents/2021/10/14/nist-ai-rfi-cubrc_inc_004.pdf

manufacturing models (one may think of a specific car model, for instance). The latter view may be useful if multiple instances of the same model must be represented, all sharing a set of common properties and capabilities.

- **Material hierarchy/traceability:** another cross-cutting concern is the traceability of material, in particular of (semi-)products through the steel production process chain. Aspects to be considered in this regard are the possibility of misidentifications by tracking systems, or even the absence of tracking information between certain processes, in which case only tracking at coarser resolution, such as one casting heat instead of individual slabs or billets, may be possible. Furthermore, tracking in general should be capable of modelling many-to-many relationships, supporting both the cutting of a product into multiple pieces, but also the joining (in the case of steel, usually rather welding) of multiple pieces into one. Furthermore, the loss of scrap due to shearing or the rolling process can lead to changes in the geometry of a product which should also be traceable.

Existing ontologies and taxonomies that might be relevant to the ProcTwin model include the IEC 62264 series of standards (Enterprise-Control System Integration) [11], the Smart Applications Reference Ontology (SAREF) and in particular the SAREF extension for the industry and manufacturing domain [12], the SmartManufacturing contribution of the Smart Data Models initiative [13], the Asset Administration Shell submodels [14], the Industrial Ontologies Foundry Core Ontology [15], or the Elementary Multiperspective Material Ontology (EMMO), which also contains a lot of manufacturing terms[16]. Overviews of relevant industrial ontologies are provided by the review and classification of manufacturing ontologies[17], the OntoCommons project report on the industrial domain ontologies registry[18] or the website of the IndustryPortal project[19].

Existing steel-specific models include the Core Reference Ontology for Steelmaking Process Knowledge Modelling and Information Management (CROS)[20] and the Steel Cold Rolling Ontology (SCRO)[21].

7.8.4 Data references

One weak point of many existing models is the lack of support for complex data references. As an example, consider a continuous casting machine in a steel mill for which sensor data from many sources is stored in a time-synchronized way, typically 1 value per second. There will be one or multiple temperature sensors attached to the caster, for instance in the tundish, the vessel that holds the liquid steel before it is poured into one of the casting strands. Now multiple entities have a relationship to the sensor value at a specific point in time: the casting machine, the tundish, the heat (batch of liquid steel), the solidified semi-products produced from the heat (such as slabs or billets). However, many modelling schemes do not support multiple such data references, if they model them at all. This is true for instance for NGSI-LD, which explicitly supports historic data storage but not with

[11] https://webstore.iec.ch/en/publication/6675
[12] https://saref.etsi.org/saref4inma/v1.1.2/
[13] https://github.com/smart-data-models/SmartManufacturing
[14] https://industrialdigitaltwin.org/content-hub/teilmodelle
[15] https://github.com/iofoundry/ontology/tree/master/core
[16] The Elementary Multiperspective Material Ontology (EMMO) : https://emmo-repo.github.io/emmo.html#module-manufacturing, https://github.com/emmo-repo/EMMO
[17] Sapel et.al., A review and classification of manufacturing ontologies (2024), https://link.springer.com/article/10.1007/s10845-024-02425-z
[18] OntoCommons D2.2: TLO/MLO Landscape Analysis Report(2022) https://zenodo.org/records/6504440
[19] https://industryportal.enit.fr/ontologies
[20] https://github.com/caoppg/CROS, https://cronfa.swan.ac.uk/Record/cronfa59000
[21] https://www.mdpi.com/2078-2489/12/8/304

multi-references, or for the Asset Administration Shell. This is an aspect where existing modelling schemes will need to be extended.

7.8.5 Serialization

Serialization specifies how an abstract information model is converted into a machine-readable format that can be stored in files or transferred via a network. Although there are many potential serialization formats with different levels of support for graph structures, the JSON format[22] has emerged as the most widely used data exchange format for the web. It is supported by all relevant programming languages and is very easy to debug due to being text-based and hence human-readable. Its main drawback is a somewhat larger size than can be achieved by binary representations.

Built on top of the basic JSON format, the JSON-LD variant (*JSON for Linked Data*)[23] provides an efficient and standardized way to represent typed graph data, by means of so-called *context documents*. Since JSON-LD documents are just special JSON documents, clients can choose to ignore the linked data aspect and just need to be able to work with JSON data. JSON-LD has been adopted as the main serialization format by the FIWARE-driven NGSI-LD specification. An example of a JSON-LD document is shown in Figure 8.

```
{
  "@context": "https://proctwin.eu/",
  "@type": "Billet",
  "id": "Billet:1",
  "averageCastingTemperature": {
    "@value": 965,
    "unit": "°C"
  },
  "producedAt": "ContinuousCastingMachine:1",
  "cutTime": "2025-06-28T10:17:21Z"
}
```

Figure 8 : Sample JSON-LD representation of a billet

The JSON schema specification[24] provides a (JSON-based) format that allows to specify constraints on JSON documents, and therefore is well-suited to represent the constraints defined by data model classes. JSON schema has been adopted among others by the FIWARE-affiliated Smart Data Models initiative[25], which aims to provide a set of data models for a variety of IoT domains.

[22] https://www.json.org
[23] https://json-ld.org/
[24] https://json-schema.org/
[25] https://smartdatamodels.org/

8. Conclusion

This document, deliverable D8.1 for ProcTwin research project, introduces the architectural principles to guide the subsequent software development. The project aims to develop novel optimization methods for steel production, including but not limited to distributed machine learning. Two industrial use cases are considered.

The document has the following main contributions: collect the requirements of the optimization systems, introduce a Data-Mesh-based organization structure, and suggest a data platform architecture to support the optimization activities, including modelling guidelines for the data model development. Now that the research project is in an early stage, currently month 6 out of 48, it is impossible to recognize all requirements. This is why the outcomes, both the Data Mesh and the data platform, aim at being generic. Both will be developed further during the project.

Regarding security, the developers of the architecture and software components will follow state-of-art principles and tools. Security is considered in non-functional requirements, the security policy of the Data Mesh, and the design of the data platform, that is, in all key stages of the design. This will enable the management of security risks considering both information and cyber security.

Next, the project proceeds to develop the software components and the connecting APIs further as well as the data platform. The former will be documented in deliverable D8.2 and the latter in D9.1, respectively.