LeArning and robuSt decision SupporT systems for agile mANufacTuring environments

Project Acronym:

ASSISTANT

Grant agreement no: 101000165

Deliverable no. and title	**D6.2 – Data Fabric Architecture Report**	
Work package	WP 6	Secure and intelligent data fabric
Task	T 6.2	Data fabric architecture design
Lead contractor	Institut Mines-Telecom (IMT) Alexandre Dolgui, mailto: alexandre.dolgui@imt-atlantique.fr	
Deliverable responsible	FLM - Flanders Make Johan Van Noten, johan.vannoten@flandersmake.be	
Version number	v1.0	
Date	29/10/2021	
Status	Release Candidate	
Dissemination level	Public (PU)	

Authors

Participant no.	Part. short name	Author name	Chapter(s)
4	FLM	Bart Meyers Johan Van Noten	All not mentioned below
6	BITI	P-O Östberg	3.2
8	INTRA	Vangelis Xanthakis	3.3
All	All	All data fabric users	Per user sections of 4

Document History

Version	Date	Author name	Reason
v0.1	2020-12-04	Félicien Barhebwa-Mushamuka	Initial Template
v0.2	2021-09-01	FLM	Initial D6.2 document breakdown
v0.4	2021-09-27	BiTi, Intra, UCC, FLM	Integrated draft for review by users.
v0.5	2021-09-29	BiTi, Intra, UCC, FLM	Integrated BiTi and Intra for review by users.
v0.6	2021-10-08	All	Integrated users' input. Refined content. Ready for review by WP collaborators.
v0.7	2021-10-15	BiTi, Intra, UCC, FLM	Final rework before internal review
v0.8	2021-10-22	UCC	Internal Review
v0.9	2021-10-26	BiTi, Intra, FLM	Final integration & rework
v1.0	2021-10-29	FLM	Release for publication

Publishable Executive Summary

In this document, the contributors on WP6 propose the architecture of the ASSISTANT Data Fabric. In addition to usual Data Fabric concerns, the ASSISTANT Data Fabric is structured specifically for handling the concerns of adaptive manufacturing scenarios.

The Data Fabric is only a part of the entire infrastructure for ASSISTANT. The following descriptions are focused on that Data Fabric aspect within the overall architecture. Four zones are distinguished that interact with the Data Fabric.

- First, there is the Interactive Consumer zone, where a user interactively investigates the available historical information. This data scientist looks for new information in the available data, tries to get analytics questions answered, and restructures data so that it can be fed into artificial intelligence model training. Among the four zones, this is the only one where a user (typically a data scientist) will interact directly with the Data Fabric.
- Next, the Automated Consumer zone, where all applications, dashboards, digital twins, AI models, etc. are hosted for runtime. Here the end-users do not directly interact with the Data Fabric, but instead interact with the applications that are hosted in this zone. It are the applications that interact with the Data Fabric..
- The third zone, the Operational Technology (OT) zone, is where a significant part of the data originates from and what WP5 will interact with most extensively.
- Finally, the Data Fabric for ASSISTANT does not necessarily cover all legacy compute and data infrastructure of the company. The ones that are necessary for the project, but not covered within the ASSISTANT Data Fabric, belong to the External Resources zone. Links or imports can be organized where required.

The most important aspect of the Data Fabric is how it organizes data and provides users access to exploring, reading, or writing such data. It is essential to make a distinction between historical data and live data. The latter is streamed directly between the producers and consumers of the live data, e.g., a device producing data in the OT zone and an application in the Automated Consumer zone that consumes this live data. The former is persistently stored first and only consumed thereafter, e.g., by a data scientist training a model. This does not mean that historical data is old data. Also, very recent contemporary data is considered part of this. Because of this difference in nature, the Data Fabric exposes both data kinds to the users through different Data Access interfaces. On top of these technical interfaces, the Data Fabric is also concerned with providing a suitable data structure and enhancing the transparency of the information and the interoperability between the different work packages (as described in deliverable D2.1). The Knowledge Graph provides this global view on data and its structure.

Finally, multiple instantiations of the Data Fabric will be created by each of its users. There are generic work packages WP3, 4, 5; industrial users AC, PSA, SE; and the demonstrator WP7. It is important to validate whether the proposed architecture fits their needs. Therefore, all users describe in this deliverable how they currently foresee interacting with the Data Fabric. As several of the project's contributions are precisely on this topic of interaction, it is expected that the vision on the Data Fabric internally and externally (from those who interact) will evolve during the project when new insights mature the Data Fabric architecture. Also WP2 will continuously check whether the architecture adheres to their concerns.

Table of contents

List of figures

List of tables

1. Introduction

1.1 Objective and scope of the document

This document defines a reference architecture for a Data Fabric tailored for AI in manufacturing. Several components from this Data Fabric architecture are highlighted, since ASSISTANT intends to go beyond the state-of-the-art on these components. Other components can be reused from state-of-the-art or state-of-the-practice. The architecture is intended to be used and deployed in several ways, using several existing platforms, to be generically usable in the manufacturing domain. It is not intended to limit itself to particular interfaces and protocols.

Next to describing the reference architecture itself, this deliverable briefly describes the instantiation of the reference architecture within the ASSISTANT project, for which an implementation will be created in Task 6.3. Within the ASSISTANT project, we identify seven users of the Data Fabric: WP3, WP4, WP5, industrial use cases of PSA, AC, SE, and the flexible assembly line demonstrator created in WP7.

This deliverable describes the first milestone of the architecture based on current insights and requirements. During the project insights as well as requirements are expected to evolve. After being validated and tested throughout the project, the final version of the architecture will be available.

1.2 Structure of the deliverable report

Section 2 describes the reference architecture at the logical level. It describes the main components of the Data Fabric, its context, and interactions. Furthermore, it discusses how to instantiate the architecture for a particular use, deployment, and security context.

Section 3 describes the architecture in detail, focusing on three main parts of the Data Fabric that enable the project's research:

- The knowledge graph and its interface to the existing data, information and knowledge that is served by the Data Fabric;
- The historical storage architecture that manages historical data, information and knowledge;
- The live streaming architecture that handles streaming data from the operational technology.

Section 4 describes the mapping of each of the users' needs to the reference architecture.

2. Logical architecture

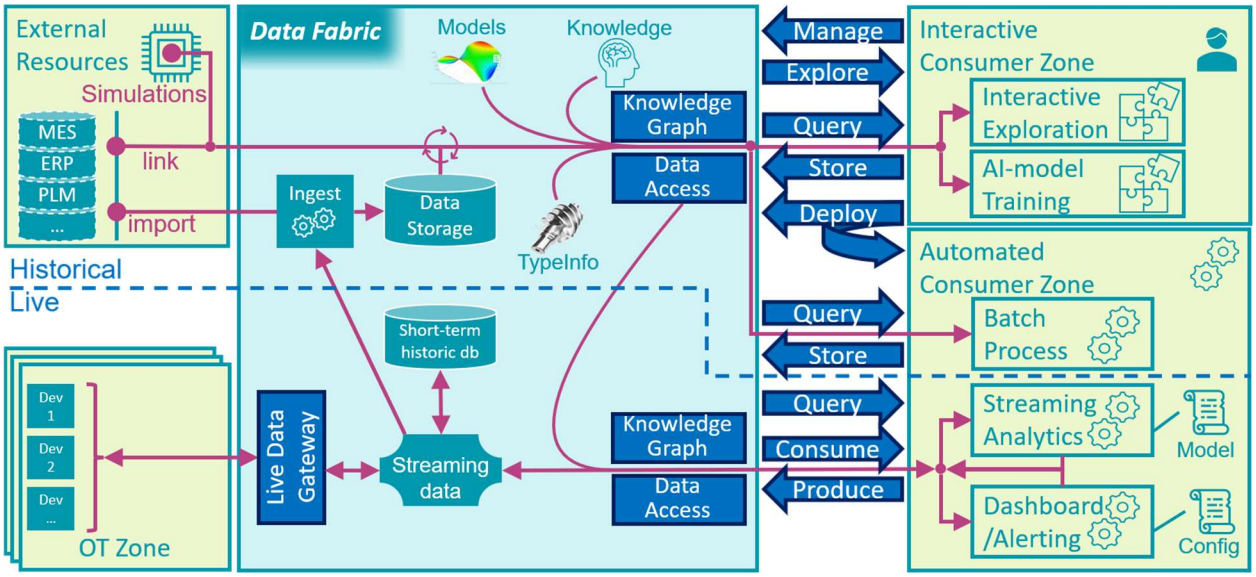

Figure 1: Logical architecture

2.1 The Data Fabric and its context

The logical architecture visualized in Figure 1 distinguishes two main parts, separated by a dashed line:

- **Historical** (= stored data): The upper part of Figure 1 is concerned with historical data, data at rest. This is where all historical data is ingested from OT devices and organizational systems such as MES or ERP. Data can also be stored by users or applications in both Consumer zones. Such data covers measurements, as well as stored models, product type information, and production knowledge.
- **Live** (= streamed data): The lower part of Figure 1 is about live data, data on the move. This is where field data originates, is collected and is processed in a real-time fashion.

Historical data does not imply "old data". Historical data also covers very recently collected data, becoming available once it is ingested in the Historical part of the Data Fabric. As opposed to "Live data" it is not consumed as a stream of data, but instead it is stored first and queried later.

In addition, the figure shows four zones:

- Centrally, the **Data Fabric** is the focus of this deliverable and is discussed in detail in the following sections.
- The **OT zone** contains the "operational technology", being the devices on the factory floor. This zone can contain machines, operator interfaces, and inspection stations, among others. Those devices are the main sources of data used in the project.
- The **External Resources** are mentioned explicitly on the figure since the Data Fabric is not intended to hold a copy of all of a company's data. Data sources, such as MES systems, may have their own data storage, either imported partially into the Data Fabric or linked to it through APIs. Also, some data might be obtained from live simulations. Those simulations are not hosted on the Data Fabric's computing resources but instead on external resources and accessed by the Data Fabric.

- The **Interactive Consumer zone** covers all kinds of interactive data analytics, typically performed by data science users. This means ad-hoc interactive investigation of the data (learning what is available, drawing first conclusions, etc) as well as the development and execution of analysis algorithms, simulations, AI-trainings, etc.
  In the context of the project, the Interactive Consumer zone is interacting exclusively with the Historical part of the Data Fabric.
- The **Automated Consumer zone** covers all kinds of automated processes that process data, and potentially draw conclusions from them or even react to them. A first kind of automated processes periodically react to historical data and are typically called batch processes. Other processes can be live dashboards showing real-time production statistics, or streaming analytics applications monitoring the OT zone caring for production optimization or anomaly detection. All of those automated processes are typically constructed in and deployed from the Interactive Consumption zone.

2.2 The scope of the Data Fabric

The term “Data Fabric” has a very broad definition with lots of aspects, as explained in D6.1. As suggested in deliverable D2.1 section 3.6.3., the responsibilities of each component in the ASSISTANT context needs to be clearly described. Together with the previous section, this section clarifies the responsibility of the Data Fabric.

In general terms, the Data Fabric covers all data infrastructure as highlighted in Figure 1. This implies that the Data Fabric covers data storage, communication, and transformation required to enable the OT and Consumer Zones’ functionality from a live as well as a historical point of view. Moreover, the Data Fabric is itself a client to the external resources. The zones indicated in yellow in Figure 1 are outside of the Data Fabric’s responsibility.

All work packages also need computational resources. This is the case for the digital twins, their simulations as well as the near-real-time activities required by WP5 (see deliverable D5.1 requirement R6.3). Such computational services can be separated into two parts:

- Inside the Data Fabric: The Data Fabric offers appropriate computational resources to perform data ingestion between live and historical zones and to care for information delivery and caching.
- Outside the Data Fabric: The Data Fabric does not provide computing resources for hosting case-specific computations, e.g., computing resources to train an AI model on historical data or to allow the runtime version of such model to react to incoming OT data, computing resources to perform simulations. Such computing resources need to be hosted in a Consumer zone and so outside the Data Fabric scope.

Moreover, as clarified in deliverable D6.1, it is not the project’s ambition to improve the state-of-the-art of Data Fabrics in all their aspects. Instead, the Data Fabric activities for ASSISTANT focus on those aspects specific to a Data Fabric suitable for “AI in manufacturing”. Therefore, the generic Data Fabric aspects can be separated in three categories:

- We will ignore those aspects that Data Fabrics often support but that are not considered relevant to this project's scope because neither the research nor the applications need this functionality.
- We will apply the current state-of-the-practice concerning Data Fabrics for the practical enabling of the project’s digital twins and the project’s research. For this purpose, the project uses standard technology covering topics such as:
  - How to host an ETL process that transforms a part of the historical data in a new and more usable (=curated) format?

  - How to realize the ingestion of OT data towards historical databases? This is for instance one of the main roles for Intrasoft's StreamHandler (see section 3.3).
- We will focus on the following key contributions in ASSISTANT concerning the Data Fabric for adaptive manufacturing:
  - Interaction mechanism between the Consumer zones and the Data Fabric's Knowledge Graph and Data Access (see section 2.3);
  - How to formalize knowledge into a Knowledge Graph, especially knowledge concerning uncertainty (see section 3.1);
  - Propose a generic KG structure on which the project partners can build their KG (see section 3.1);
  - A data storage architecture including simulation- an optimization-based tools for its intelligent management and orchestration (see section 3.2);

2.3 Interactions between the Consumer zones and the Data Fabric

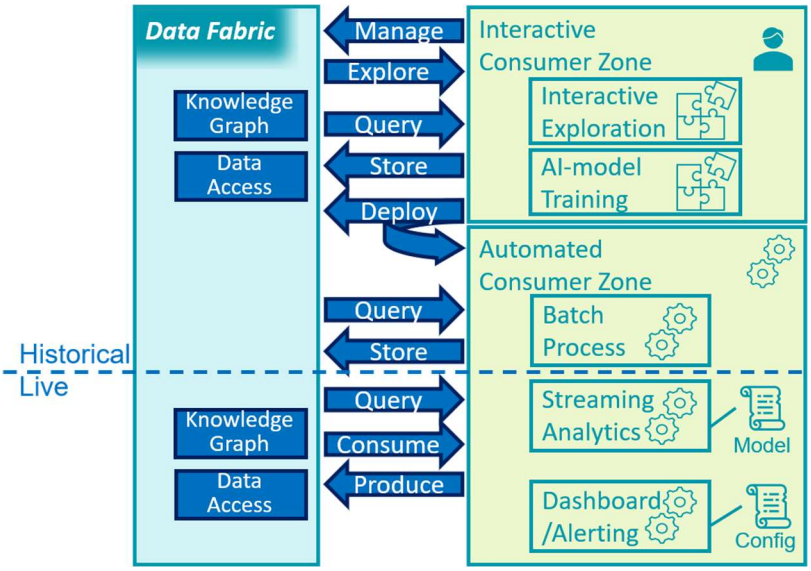

Figure 2: Interactions between the Consumer zones and the Data Fabric

Figure 2 repeats the main Data Fabric interfaces towards the Consumer zones and their interactions, already depicted in Figure 1. These are the Consumer interactions that drive the project's contributions on the Data Fabric state-of-the-art. As indicated in deliverable D2.1, it is important to provide to the user a clear and transparent view on the system. In the case of the Data Fabric, this transparency mainly applies to the interfaces accessed by the Interactive Consumer zone and to the role the Knowledge graph plays in it.

The internal structure of the Data Fabric is explained in more detail in chapter 0, but the main parts are motivated and introduced briefly here.

2.3.1 Knowledge Graph

- What is the Knowledge Graph (KG)?
  - It provides an overview of all data, information and knowledge that is available in the Data Fabric.
  - It mainly focuses on the descriptive part of the data (the metadata, the types, concepts and the links between them, additional knowledge, etc.)
- Interactions from the Interactive Consumer zone

- Explore – The user interactively explores the KG. Such exploration is typically navigation, visualization, etc.
- Query – The user retrieves parts of the KG in order to reprocess it for his own needs.
  Example: The KG may model products, processes and process steps. The user might want to query these elements from the KG and use them in an optimization model.
- Store – The user adds new knowledge into the KG. This knowledge can have two natures without a hard line to distinguish both:
  - A user can add new concepts, attributes or relationships. This changes the structure of the KG: after the addition, new kinds of information can be captured in individuals of those new concepts (being instances, occurrences of these concepts).
    Example: The user wants to go and define a new attribute "Tolerance" to a previously existing concept "PhysicalDimensions". From then on, the user can capture new values for Tolerance on each PhysicalDimension.
  - A user can add new individuals.
    Example: The user has performed a data analysis experiment and found a correlation between two parameters. The user stores this new knowledge into the KG. It then becomes available for future Explore/Query interactions.
- Deploy – When adding new concepts, attributes or relationships, a user may also need to specify how these link to specific data items on the Data Access interfaces (in case of a Virtual KG, see 3.1.4).

- Interactions from the Automated Consumer zone
  - Query – Applications in this zone know which information they need to retrieve. Not all of them will know where to find that information directly in the Data Fabric interface. They can query the KG in order to obtain the required metadata.
    Example: an application needs live data for the temperature of the inlet pump, but it does not know where to look for that information on that particular OT environment. It asks the KG for the correct Data Access location for this particular data stream (e.g. a specific Kafka topic). Once known, it starts consuming the desired data from the Data Access.

2.3.2 **Data access**

- What is Data Access?
  - It provides technical access to all data available in the Data Fabric to the extent of the user's access rights.
  - The precise access method can depend on the nature of the data the user is looking for.
- Interactions from the Interactive Consumer zone
  - Query – The user formulates a query for a specific subset of the Data Fabric's data.
    Example: Get the temperature measurements for sensor X between dates Y and Z. Get the ids of the machines involved in the process steps for product X.
  - Store – The user can store additional data into the Data Fabric. This is typically data computed by the user, e.g. in the context of a data science experiment.
- Interactions from the Automated Consumer zone
  - Consume / Query – Similarly to the Interactive Consumer zone, the applications can obtain their data here. Whereas the Interactive Consumer zone is typically requesting historical data, the Automated Consumer zone is usually (but not

exclusively) interested in live data. Therefore, the queries can be formulated as continuous queries or connections to incoming data streams, as well as traditional queries over short-term historical data.
    - Produce – A process in the Automated Consumer zone typically does not store additional data but rather sends output data as a continuous stream for consumption by other participants.

2.3.3 Generic interactions

Some generic interactions are not further detailed as we will implement these according to state-of-the-practice:

- "Deploy" interactions: When users create additional functionality, they need to deploy it to a computational platform. Such contributed functionality comes in various kinds:
    - An automation, such as an improved ETL process (a user's specific ingestion scripts)
    - A user-facing functionality, such as a dashboard for visualizing occurring production anomalies
    - A batch process periodically recalculating optimal planning for the next day of production.

  Some of these contributions need to be hosted on the Data Fabric. Others become hosted in the Automated Consumer zone or on some External Resources. In all cases, this is a kind of deployment.
- "Manage" interactions: Like any IT infrastructure, the Data Fabric also has management interfaces to handle security, user access, resource scalability, etc.

In conclusion, although Deploy and Manage are essential for the Data Fabric users, they do not match with any of the research contributions of the project. Therefore, these interactions are handled using default state-of-the-practice techniques and are not further detailed in this document unless required.

2.4 Data Fabric instantiation

This document describes the architecture of the Data Fabric. Such an architecture is a template of what a Data Fabric for ASSISTANT should look like. It does not mean that there should only be one single Data Fabric instance (= concrete installation) in use within the project, nor does it mean that all Data Fabric instances should be identical. Every user can deploy his instance. Such a decision takes into account several trade-offs:

- A single instance allows for better collaboration between its users because a single Knowledge Graph and Data Access provide access to all available knowledge and information.
- Separating instances allows for better data separation if the industrial partners of ASSISTANT need to store company-specific data. From a corporate governance point of view, the industrial partners will never allow confidential data inside a proof-of-concept implementation hosted outside of their control.
- Separating instances also allows for more independent evolution. For example, version evolutions or structural experiments can be performed more easily if they do not disturb any users using the same Data Fabric instance.

For good governance of the project, several instances of the Data Fabric architecture will be maintained by their respective users. The exact split needs to be decided during the project execution, but it is expected that the project ends up with the following instances:

- Research instance: covers the needs of WP6, allows for several experiments without disturbing users.
- Development instance: covers the needs of WP2, WP3, WP4, WP5.
    - It is advantageous that this is a single instance for the four work packages since it allows to research the relationships between the data models and the work packages.
    - During their individual development, work packages may still create a private instance.
    - It is essential to:
        - Maintain good governance for the different WPs to evolve their contributions smoothly, e.g. when a new version of WP3's domain model is deployed (see section 2.5).
        - Clarify which data will be put inside this common Data Fabric instance. For example, if WP3 desires working with confidential company data, such data should not be exposed to WP5. It is up to these work packages to check which data they can collectively host inside their common Data Fabric.
- User instance: each industrial partner can create a separate Data Fabric instance for their internal validation work. In this case, the industrial partner can choose to host this instance in a trusted environment and to fill it with confidential data.
- Demo instance: covers the needs of WP7. This is used for building the overall results in a demonstrable environment.

2.5 Governance

2.5.1 Dependency and version management

Throughout the project, the Data Fabric will evolve while it is used by several stakeholders simultaneously. In order to manage this, good governance needs to be defined. Although dependency and version management are not the only aspects of governance, they are currently perceived as the biggest integration risks because partners will evolve at different speeds, while still having the desire to integrate. Applying the rules below avoids significant versioning conflicts and inefficiencies.
The explanation below uses concepts that are introduced in section 3 only. In order to understand the details, the reader might consider to read that section first.

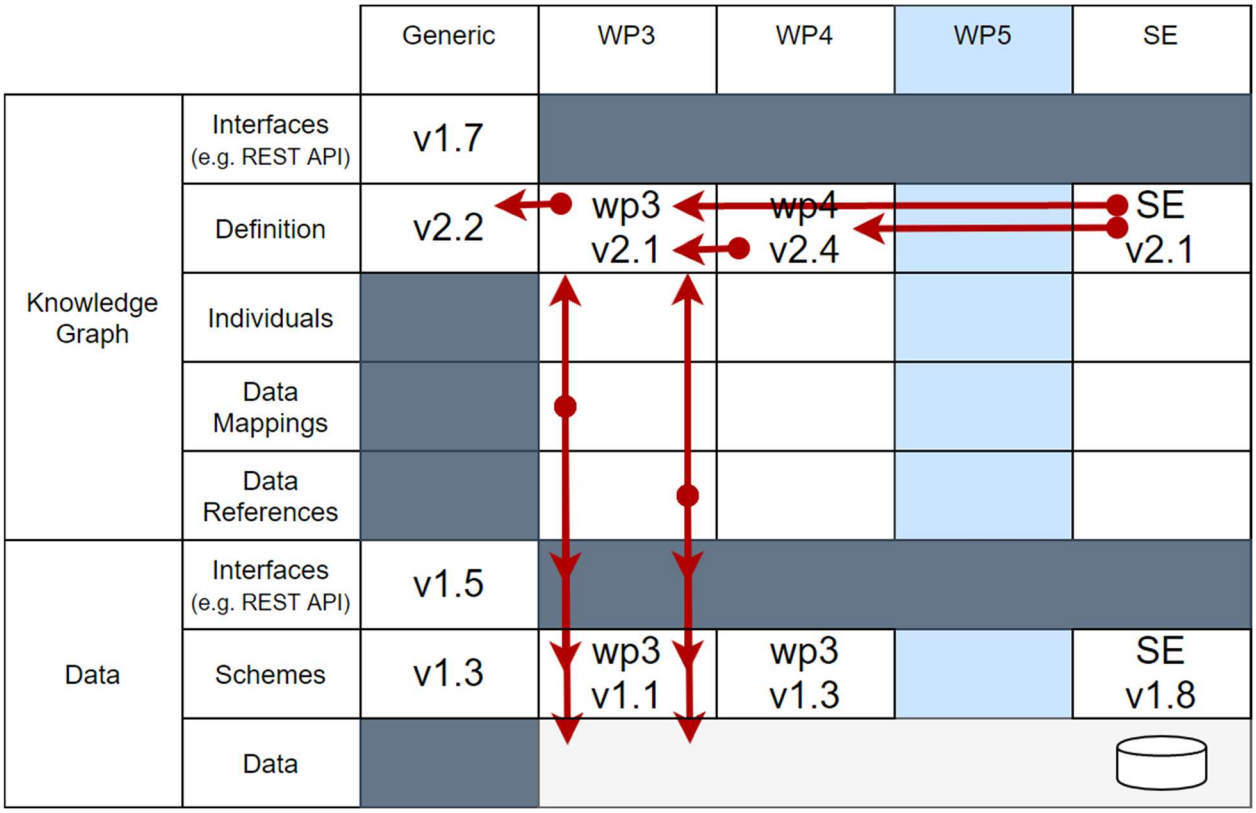

Figure 3: Version dependencies within an example SE Data Fabric instance

Figure 3 graphically represents the different entities involved in realizing the Data Fabric instance for one of the use cases, the SE industrial use case. The dependencies in the figure are for clarification purposes only, and do not correspond to the (still to be created) real artifacts. The red arrows indicate dependency relationships. Reading through the "Generic" column of the figure, one can conclude:

- The Generic KG interfaces currently carry a version number v1.7.
- The Generic KG concepts are defined in an ontology with version number v2.
- The Generic Data Interfaces are at v1.5
- The Generic Data schemes are at v1.3.
- The Generic components do not have their own individuals, data mappings, data references or data, as these are application-specific. (dark cells)

Next, work packages or users bring their own Data Fabric elements. As an example for WP3:

- WP3 does not change anything to the technical interfaces of the KG or of the Data. Those are centrally, generically defined and are therefore WP6 ownership. WP3 reuses these. (dark cells)
- WP3 extends the generic KG (carrying v2.2) definition with concepts and relationships specific to product planning, depending on v2.2 of the Generic KG definition. In the figure, WP3 itself currently assigns v2.1 as a version for these extended specific KG concepts. This extension mechanism is explained in more detail in section 3.1.3.
- WP3 also defines what the schemes should be of the data that represent the planning concepts. The figure shows v1.1 for these schemes. Also these schemes might relate to the generic schemes that currently carry v1.3.
- WP3, therefore also has to define the Data Mappings and potentially Data References it wants to make between the KG definition and the corresponding data below on the figure. These mappings and references make use of the data interfaces and schemes.

Similarly, WP4 depends on WP3's KG definition if WP4 desires to use the product planning info in its own scheduling algorithms. Likewise, the industrial use case of Siemens Energy (SE)

depends on WP3's KG definition and WP4's if they want to experiment with those WP's contributions. The SE case does not handle real-time functionality, so the SE Data Fabric instance has no dependencies on WP5, hence the light blue column for WP5.

The data itself is provided by SE, as shown on the lowest row of the figure. Although WP6, WP3 and WP4 define what the data should look like, SE provides the underlying data for the SE use case.

The example illustrates that the Data Fabric instances and their dependencies become unmanageable if every contributor modifies every aspect of the ecosystem without taking the dependencies into account. This problem is very close to what is encountered with dependency management in software development or in any service-oriented architecture system.

The following rules will be followed to improve the manageability:

- Strict version & release management. Modifications to all aspects of the Data Fabric (except for pure data) are to be:
  - versioned and made accessible in the central repository for the project
  - released according to the following convention:
    - Major version number: can break dependencies
    - Minor version number: never breaks dependencies, just adds functions / attributes / abilities
    - Releases that depend on other releases carry the same major version number (although their minor part can vary independently).
- Synchronized upgrades that are approved by a committee happen on a per instance basis. Every Data Fabric instance is allowed to make use of any desired existing releases of the elements it depends on as long as they are mutually consistent.
- Evolution of ontology definitions or schema evolutions will cause impact on the mappings, references and underlying data structure. It will be the responsibility of the Data Fabric instance owner to adjust these accordingly. The project does not take care of any form of automatic data structure evolution.

2.5.2 **Issue management**

Next to version management, the Data Fabric also needs issue management. This allows the Data Fabric collaborators as well as its users to report issues and follow them up.
Issues need to be submitted on the appropriate topics:

- WP6 - The generic Data Fabric issues
- WPx – Issues with the contributions of the specific WP
  (e.g., the model of WP4)
- InstanceX – Issues with a specific instance of the Data Fabric
  (e.g., the WP7 Data Fabric instance)

Each of those issue topics gets a responsible assigned who will delegate the issue if required.

2.6 Cloud or Edge Deployment

The logical architecture does not make any statements on the deployment location of the different zones. Naturally, our production scenarios' OT zone is on premises since it contains the physical machinery itself. Similarly, the historical part of the Data Fabric typically has a connotation of cloud deployment since more and more companies make use of the flexibility and reliability that cloud solutions offer today.
The live part of the Data Fabric and the Automated Consumer zone are often positioned close to the OT zone (i.e. on the edge) since some of their functions might require close-to-real-time

and highly reliable operation. On the other hand, the live part of the Data Fabric and the Automated Consumer zone can also be hosted in a cloud environment. In general, both are the most obvious candidates for a distributed physical deployment: one live Data Fabric and live Automated Consumer zone on each edge combined with one or several cloud deployments, each performing their function at the most appropriate location.
The project's architectural view does not impose any of these deployments. Each use case or digital twin in this project may decide a deployment that suites them best.

2.7 Security

Security is an essential concern for any IT system, and as discussed in deliverable D2.1 section 3.5, the ASSISTANT project respects security requirements throughout the architecture. All zones visualized in Figure 1 are vulnerable to security risks. Applications in the Consumer zones and installations in the OT zones can be attacked or leak sensitive information. In the context of this Data Fabric architecture document, we consider only the risks that directly impact the Data Fabric. Indirect threats such as information leaks using a compromised OT zone are not considered but left as the responsibility of those zones. Those indirect threats will be handled in the general project architecture of WP2.

Security is a complex combination of many aspects and the final level of a system's security is as low as that of its weakest part. Therefore a thorough threat analysis (using FMEA, FTA, and other risk management processes) would be performed on all systems intended for production use. The Data Fabric is intended to run as a proof-of-concept only within this project, so the threat analysis and security aspects are performed less thoroughly. Most of the security implementation depends on the implementation technologies chosen for the individual digital twins and demos.

The following security aspects are to be covered in the Data Fabric in order to obtain a basic level of security:

- **Authentication**:
  - Access to the Knowledge Graph and Data Access interfaces of the Data Fabric is only possible after successful authentication.
  - Deliverable D2.1 section 3.5.2 advises a central authentication system (CAS) for this purpose. The Data Fabric will play the role of a CAS client for those deployments that have a CAS server available.
- **Authorization:**
  - Depending on the authenticated user's role, some or all of the information will be accessible for reading or writing. Deliverable D2.1 section 3.5.1 suggested the implementation of a role-based access control (RBAC) mechanism.
  - For the Data Fabric, this RBAC is limited to the Knowledge Graph and Data Access APIs. Its granularity depends on the individual services offered during the project but will be reduced to essentials since the precise role assignments are not the project's focus.
- **Confidentiality**:
  - Communication between the users and the Data Fabric needs to be protected against eavesdropping or other non-authorized access as described in deliverable D2.1 section 3.5.3.
  - The Data Fabric establishes state-of-the-practice communication channel encryption (TLS/SSL) to cover this risk for data in transit. Additionally, the data stored in the Data Fabric will be encrypted at rest using state-of-the-practice algorithms (AES).
- **Privacy:**

  - Closely linked to confidentiality is the concept of privacy. Since the project covers GDPR-sensitive data, this data needs to be anonymized to the level that it cannot be traced back to individual persons. The Data Fabric itself does not implement any data anonymization techniques and supposes that the project partners anonymize the data before storing the data into the Data Fabric.
  - On the other hand, as suggested by deliverable D2.1 section 3.6.5, the Data Fabric needs to watch privacy in the audit or usage log data generated internally.

Some additional security aspects are of lower importance to the proof-of-concept level of the Data Fabric in this project:

- **Integrity:** Protecting the data against manipulation of its content.
- **Non-repudiation:** Making sure that the sender of the data has proof of delivery.
- **Availability:** Avoiding potential loss of data of unavailability of (parts of) the Data Fabric.

Those aspects would only be essential for production purposes.

Because of its status as proof-of-concept Data Fabric, the project partners will not trust the Data Fabric with any sensitive data and stick to the following rules:

- Deploy the Data Fabric in a trusted environment (perimeter security)
- Only allow access to that perimeter by trusted personnel
- Ensure that all data stored into the Data Fabric is free of privacy or Intellectual Property concerns in those cases where the perimeter security does not satisfy the company's security level.

3. Data Fabric architecture

This section describes the architecture of the Data Fabric in more detail by highlighting its different parts:

- 3.1 The Knowledge Graph
- 3.2 Storage architecture
- 3.3 Live streaming architecture

This refines the distribution of responsibility suggested in deliverable D2.1 section 3.6.3 to the different aspects within the Data Fabric architecture.

3.1 The Knowledge Graph

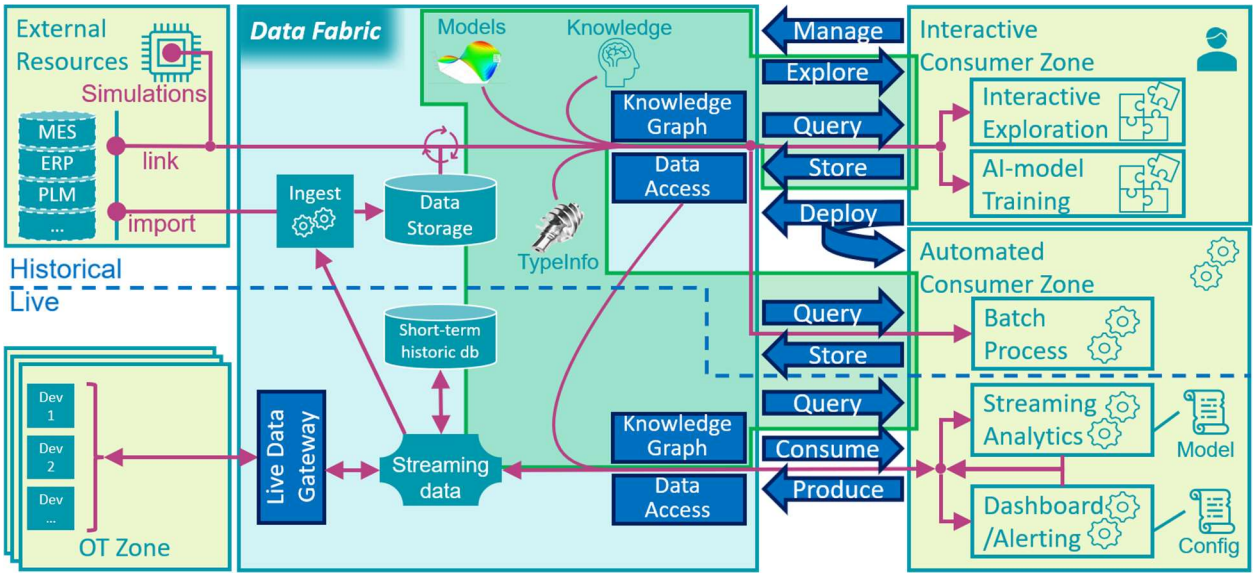

Figure 4: Position of the Knowledge Graph in the Data Fabric (Green area)

The green area within the Data Fabric zone in Figure 4 marks the Knowledge Graph and its interactions.

The following user stories are used throughout the section to guide the motivation:

- User story KG1: As an AI expert in the Interactive Consumer zone, I can interact with the Knowledge Graph to create an AI model using historical data and exploit it later on the Automated Consumer zone.
- User story KG2: As a domain modeler, I can create a domain-specific Knowledge Graph to make all relevant data, information, and knowledge of my domain accessible.

Many more user stories can be defined and refined, but it is not intended to be complete here. Instead, they serve as clarifications and motivations.

3.1.1 Incremental Knowledge Graph creation

Building a Knowledge Graph is an incremental process. Each company owns, stores and streams a lot of data, information and knowledge. It is unrealistic to go and include all of this data in the KG at once. Instead, the KG is to be built up incrementally. This way, the KG slowly covers more and more of the company's knowledge.
Often, this build-up is a manual process. Automated cataloguing systems exist, are constantly evolving, and can certainly take away a significant part of the manual burden. However, they

are limited to cataloguing information that is available in plain data (e.g., databases and documents). A KG can be much broader than that. It can contain knowledge produced in data science experiments. For example, a data scientist can store his experiment for future reproduction and store traceability links to the data on which he based his conclusions. Furthermore, he can store a calculation for a derived attribute that he computed in the experiment, together with this derived attribute's uncertainty and validity frame. Such information will not be catalogued automatically from existing data but will be contributed through the "Store" interaction on the historical KG.
One of the research contributions in ASSISTANT concerns that latter aspect precisely: how to express the knowledge concerning experiments and uncertainty in a helpful way in the KG, how does it evolve?

3.1.2 Generic Knowledge Graph structure

The project proposes a generic structure of a KG to:

- assist Data Fabric users to build their own KG as an extension to this generic structure (this composition mechanism is explained in section 3.1.3);
- assert the compatibility of the user-specific KG with the Data Fabric architecture;
- provide a means of standardization of generic concepts in the domain.

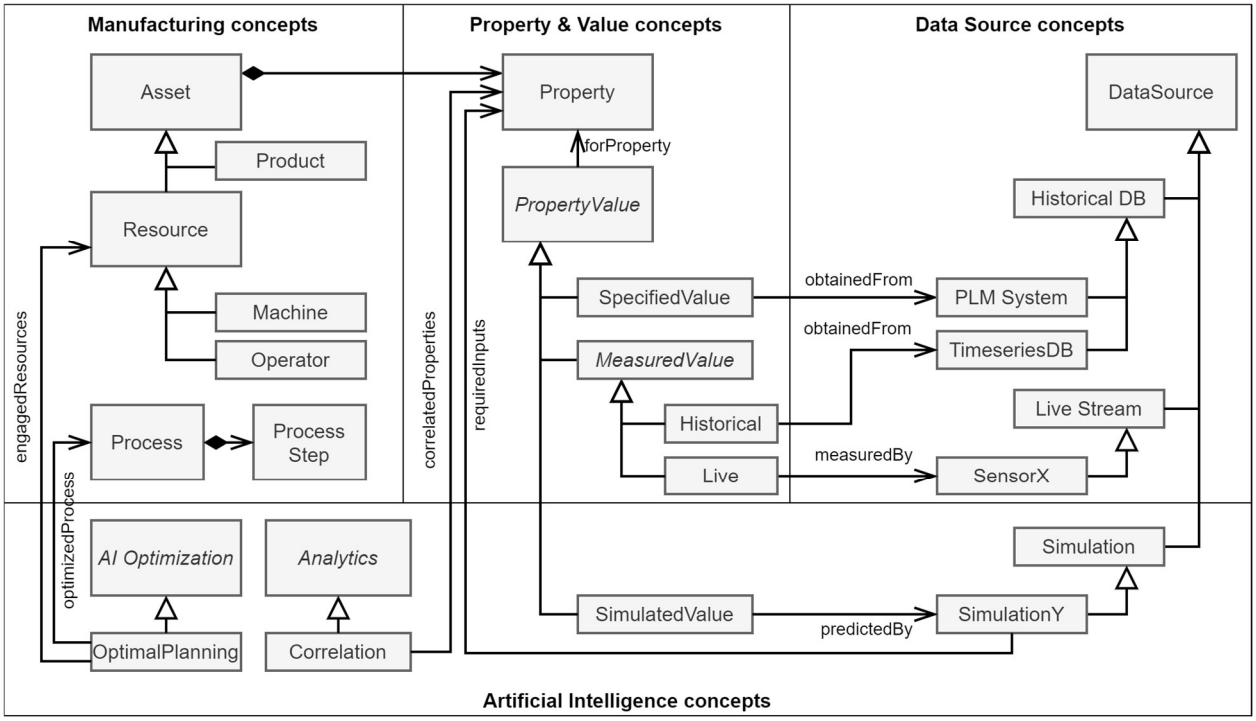

Figure 5: Minimalistic generic Knowledge Graph structure

Figure 5 shows an illustrative (but minimalistic) example of the generic Knowledge Graph structure as a UML class diagram. Without going into the details of the semantics of a UML class diagram, some examples of how to read the figure:

- A Machine is a kind of Resource, which in turn a kind of Asset. (hollow arrowhead)
- A Process owns Process Steps. (diamond + arrow)
- An OptimalPlanning references one or more Resources and considers these as its "engagedResources". (arrow + relationship name)

One of the research goals of ASSISTANT is to finalize the generic structure so that it is compatible with the needs of ASSISTANT. The project draws inspiration from existing standards and ontologies.

As a refinement of User story KG1, the following user stories are typically followed in succession (in multiple iterations):

- User story KG1.1: As an AI expert in the Interactive Consumer zone, I can use the Knowledge Graph to find historical data to perform data analytics that extract features and insights, which I can store back into the Knowledge Graph.
- User story KG1.2: As an AI expert in the Interactive Consumer zone, I can use the Knowledge Graph to find the historical data of a given property so that I can train an AI model.
- User story KG1.3: As an AI expert in the Interactive Consumer zone, I can use the Knowledge Graph to find the suitable input parameter values (possibly defined by User story KG1.1) of my AI model from the Knowledge Graph so that I can execute my AI model (inferencing).
- User story KG1.4: As an AI expert in the Interactive Consumer zone, I can explore the knowledge graph so that I can learn what data, information and knowledge exists.

The generic structure consists of three connected parts, shown in Figure 5:

- **Manufacturing concepts**: this part contains a generic ontology for manufacturing with connected concepts like Asset, Resource, Machine, Process , etc. It provides semantics and structure to all information in an organization. Using the hollow arrow inheritance notation, it indicates that Machines are a kind of Resource and Resources are a kind of Asset. When exploring and querying the Knowledge Graph, this structure is used to navigate through all information. Inspirational standards are: ISO-95, SOSA (see deliverable 6.1).
- **Property & Value concepts**: this part classifies and structures different manifestations of physical properties (speed, temperature, etc.), like measured values (e.g., from a sensor), expected values (e.g., from a PLM system). These values can be single or composite (e.g., time series, images), units can be attached, etc. Inspirational standards are SysML QUDV, etc. The values can be used for AI in each of the above refinements of User story KG1.1.
  Some examples:
  - Historical measurement data can be found in the Knowledge Graph for data analytics or for training an AI algorithm.
  - Insights from data analytics can be stored in the Knowledge Graph, such as a calculated probability distribution function representing the uncertainty of the execution time of a task or a correlation between the time executed and the end quality.
  - Input for inferencing a value based on a machine learning algorithm that can predict quality based on the temperature during milling.

  Note that this Property concept is different from the object and data properties as defined in the context of ontologies. The Property concept here represents an asset's physical or logical properties and is modelled as an ontology class.
- **Data Source concepts**: this part provides different aspects of data, namely historical data or live data. The different possibilities to link to data sources are explained in section 3.1.4, and the way to access the actual data is explained in sections 3.2.3 and 3.3.2.
- **Artificial Intelligence concepts**: this part includes some concepts to represent elements in the artificial intelligence workflow. If WP4 produced an AI Simulation model, it should be possible to store in the KG that this model provides values for a specific (set of) properties.

3.1.3 Knowledge Graph composition

The KG in ASSISTANT will be combined from several parts. A central KG models the domain concepts, relationships, and data in common for all project partners. In an own separate KG each partner references the generic KG, complements it, and enriches it with elements specific to the partner's research domain or use case. This is an essential enabler for the information interoperability explained in deliverable D2.1 section 3.4.1. From the point of view of a user exploring/querying the KG, this distinction is fully transparent and interoperable.

We identify the following user story as a refinement of User story KG2:

- User story KG2.1: As a domain modeler, I can extend the Knowledge Graph with the concepts relevant to my domain so that all relevant data, information and knowledge of my domain becomes accessible.

The domain concepts of the generic KG (developed in WP6 and explained in 3.1.2) are being extended by the contributions from the different domains as shown in Figure 6:

- Work-package specific contributions for WP3, 4 and 5
- Industrial user specific contributions from AC, PSA, SE
- Demonstration specific contributions from WP7 for flexible assembly

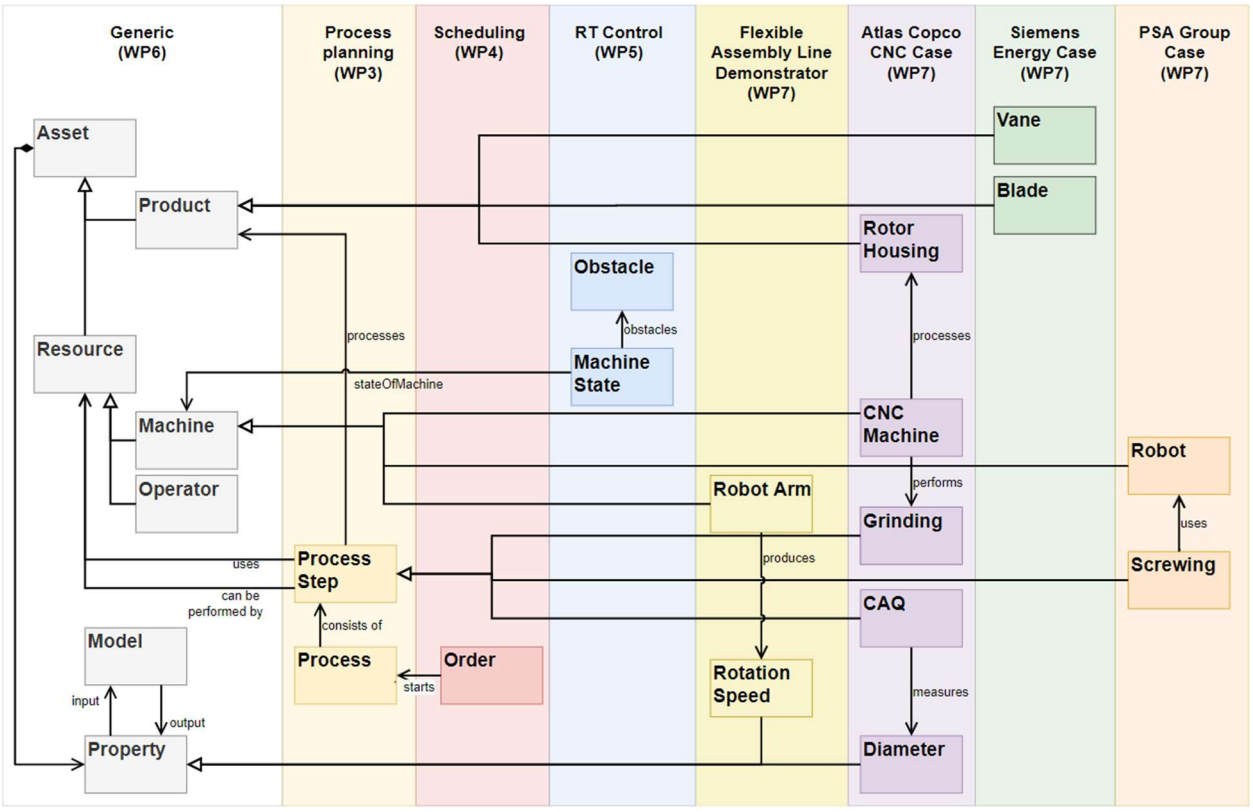

Figure 6: Domain models - generic vs specific

In addition, WP2 plays an essential role as a watchdog of the ethical and human-centric vision. The combined Knowledge Graph nor the Data Storage itself should contain or expose information that violates the guidelines of WP2, and it should contain those pieces of information required to realize WP2's ethical and human-centric vision. Therefore WP2 plays an important reviewing role during the creation and composition of the Knowledge Graph and data models.

3.1.4 The Knowledge Graph's link with data: direct, virtual, or referencing

A Knowledge Graph contains concepts, properties and relationships. In addition, a KG can contain so-called individuals.

- Concepts can be: Resource, Machine (kind of Resource), etc.
- Individuals are the instances of these concepts: all the machines in a company's OT zone are individuals of the "Machine" concept.

Typically, those individuals are considered to be the Knowledge Graph's "data". However, that does not mean that the KG needs to contain all the data available in a company. This would be undesirable and usually even unrealistic. So then, how do the KG and the data relate? Several options exist, and typically, a KG assembles these aspects, balancing flexibility, cost, and performance.

Direct individuals

A KG can contain all of its individuals inside its infrastructure. This is the traditional way of constructing a KG: the KG contains all concepts and individuals.

We identify the following user story as a refinement of User story KG2:

- User story KG2.2: As a domain modeler, some of my domain-specific concepts can store their data (individuals) in the KG itself. They do not need to be stored in separate data sources.

Virtual Knowledge Graph

While a KG can contain the individuals for its concepts, it does not need to physically contain them in all cases. A virtual KG dynamically obtains some of its individuals from other storage kinds and provides them to the user on request. This causes them to scale better when the number of individuals increases.

We identify the following user story as a refinement of User story KG2:

- User story KG2.3: As a domain modeler, I can define mapping rules between the concepts in my Knowledge Graph and data sources in the Data Fabric so that, when queried, individuals can be dynamically and automatically be retrieved from those data sources, thus avoiding the need for duplicating all data in the Knowledge Graph.

In the above example, the Machine individuals might not be stored in the Knowledge graph but in a database. A Virtual KG then acts as if it contains the individuals, but when the user requests these, the Virtual KG retrieves them from the underlying database and returns them to the user as if they were part of the KG.
For the implementation of its Virtual KG, the Data Fabric relies on Ontop (https://ontop-vkg.org/). This method comes with its limitations:

- Ontop only supports virtualizing data with SQL interface;
- Ontop does not allow updating data. Instead, updates need to be applied directly to the underlying database.

Referencing Knowledge Graph

Sometimes storing individuals in the KG as well as merely accessing them through the KG is explicitly undesired. The sheer size of the related data might be prohibitive, or a KG might not be suitable for representing those individuals.
This is, for example, applicable to time series data, which are very prevalent in the project's production scenarios. However, a KG does not have the appropriate structure to efficiently

handle massive amounts of time-based measurements for vast amounts of operational physical properties (temperature of the motor of driveline X in machine Y, pressure of pump 3…). Instead, the KG should handle such values as leaves in its structure: instead of containing the values as individuals, it points to the appropriate data access entry of the Data Fabric.

We identify the following user story as a refinement of User story KG2:

- User story KG2.4: As a domain modeler, I can define references to data sources in the Knowledge Graph so that, when queried, the location of data (e.g., a query and url to a timeseries database) can be retrieved, thus avoiding the need for converting all data to Knowledge Graph individuals.

This is followed by a user story as a refinement of User story KG1.2:

- User story KG1.2.1: As an AI expert in the Interactive Consumer zone, I can ask the Knowledge Graph where I can find the timeseries data for my property of interest. The KG provides me with a query that I can execute on the Data Access interface to retrieve all desired historical data.

The example with time series for sensor data leads to a KG structure similar to the one depicted in Figure 7. The outermost KG concepts (e.g., leaves such as InfluxDB) act as a proxy for the real, externalized data source (i.e. the reference to the actual Influx instance, the database, measurement, and field names). Therefore, a query on the KG does not return the underlying data; instead returns the contents of the blue boxes shown on the right. For obtaining the actual data, the Data Access interface of the Data Fabric needs to be accessed, specifying the information in the blue box. The information currently mentioned in the blue boxes is stated in legacy terms (such as an Influx link), but during the execution of the project, these references will be transformed in terms of the concepts offered by the Data Fabric's Data Access interfaces.

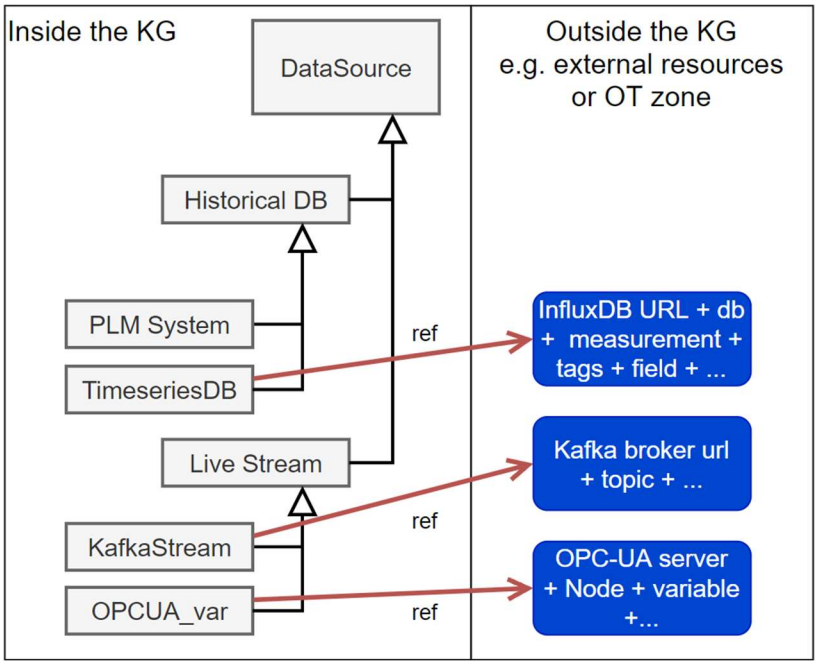

Figure 7: Referencing Knowledge Graph (references in red, external links in blue)

3.1.5 Historical versus Live Knowledge Graph

As explained above, the KG occurs in the historical part of the Data Fabric and the Live part. It does play the same role in both cases, but its coverage is different. The Live KG does not

represent “all” knowledge contained in the Data Fabric. Instead, it only represents that subset of the knowledge that might be relevant to answer Live queries.
An example can clarify this based on User story KG1. Suppose a user in the Interactive Consumer zone needs to develop an AI model, intelligently reacting to temperature variations of an inlet pump of a specific machine type. Such a user will typically follow these steps:

- Interactively investigate the KG to obtain the required knowledge (i.e., User story KG1.4)
- Perform some queries to obtain representative historical training data (i.e., User story KG1.2)
- Train a model based on this data
- Perform another query to obtain some validation data (i.e., User story KG1.2)
- Validate the model based on this data
- Iterate till the model seems to be ok
- Then the user deploys the model to operate on 10 Automated Consumer zones, deployed in the ten sites of the factory. Unfortunately, some sites have the pump connected through OPC-UA. Some sites have it on CAN-bus then distributed over DDS. The user does not want to customize the data input for the AI model. Instead, the AI model takes the generalized input “MachineTypeX/Pump/Temperature”.
- After deployment on-site, the AI model first queries the Live Knowledge Graph to know where the “MachineTypeX/Pump/Temperature” can be found on that site. Next, it uses the correct connector to subscribe to the incoming data.

For such a scenario, the Live KG becomes site-specific and only contains the mappings between the generic names and the concrete, site-specific implementation details.

Remark: This example serves the sole purpose of clarifying the scope difference between a historical KG and a live one. An AI model probably does not need to perform this kind of mapping, but instead, an on-site gateway will perform this kind of mapping. That gateway needs similar configuration input, though, which can be found in the KG if desired.

3.1.6 Technical interfaces

Several techniques can be used to implement Knowledge Graphs. It is not the project’s purpose to move the state-of-the-art for KG technology. The project’s focus is on the KG’s contents. So the technical implementation of the KG will be performed with the following standard ontological tooling. This can be seen in Table 1.

Table 1: Overview of KG technical implementations

Tool	How to use / interface
Protégé https://protege.stanford.edu/	A free, open-source ontology editor and framework. This is an interactive Java application, usable in the Interactive Consumer Zone for: • Query & Store: editing the Knowledge Graph’s structure: adding concepts, relationships, attributes,… • Explore & Query: Editing and testing SPARQL queries on the KG
Ontop https://ontop-vkg.org/	A Virtual Knowledge Graph System. This can be used in the Consumer zone:

	• Query & Store: As a plugin to Protégé to edit the mappings between the ontology and the underlying SQL-alike data (Interactive Consumer zone only) • Query: As an http-endpoint to launch SPARQL queries from either Consumer zone
VOWL http://vowl.visualdataweb.org/	Visual Notation for OWL Ontologies This can be used in the Interactive Consumer zone: • Exploration: browsing through the ontology, finding desired concepts or relationships
SPARQL https://www.w3.org/TR/rdf-sparql-query/	Query Language for RDF This can be used in either Consumer zone: • Query: retrieve desired information from the KG. • Store: store additional information in the KG (Interactive Consumer Zone only).
Apache Jena https://jena.apache.org/	A free and open source Java framework for building RDF graphs and Linked Data applications. Since Ontop on its own cannot support the update of data (it is intended solely for querying), a usable environment hides Ontop behind a solution such as Apache Jena, that cares for query federation.
Ontotext GraphDB https://graphdb.ontotext.com/	GraphDB is an enterprise ready Semantic Graph Database. In order to store ontologies and individuals on scale, the basic abilities of files are not sufficient. A database such as GraphDB will be integrated to support more advanced storage and levels of federation that are not easily attainable using Jena.

Based on the technical tooling mentioned in this section, the Data Fabric provides access to the Knowledge Graph. Figure 8 shows a minimalistic deployment only using Ontop's OBDA approach and is extended with the additional tools during the project. The figure shows how:

- a Browser can be used to explore the ontology through the OWL viewer
- Protégé and Ontop's Protégé plugin can be used to define an ontology and its mappings which then need to be stored (the ontological concepts) and deployed (the mappings)
- Finally, Ontop provides a SPARQL endpoint, which is a standardized http-endpoint that understands the SPARQL protocol. The SPARQL clients mentioned in the figure depending on the Consumer's technological choices. Clients exist for environments such as Python, Databricks, Java, and many more. The indirect access through the KG to the Data Access does not exclude direct access by the consumers to the Data Access interfaces. What is more, the Referencing Knowledge Graph (see 3.1.4) by definition, returns only references to the Data Access interfaces rather than returning the data itself. The precise format of these references is determined during T6.3.

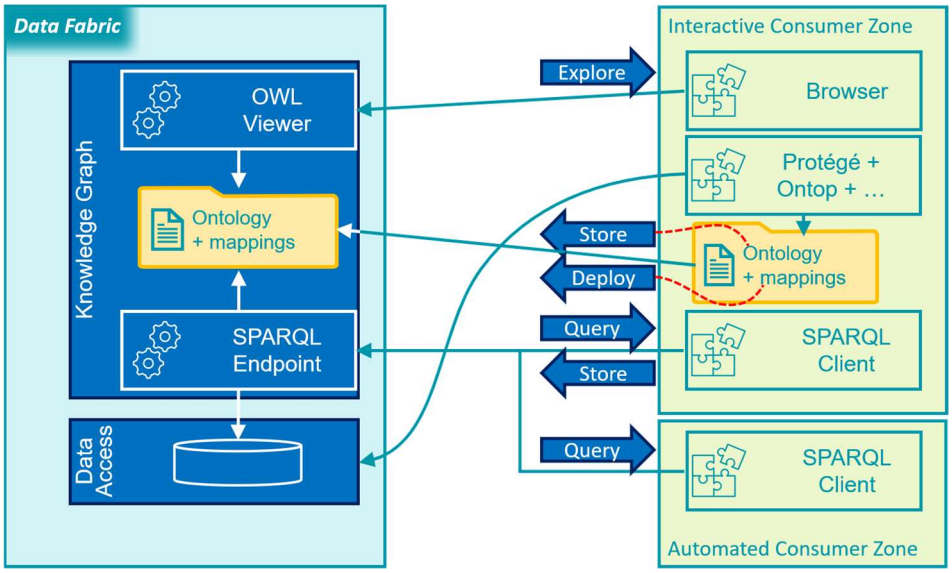

Figure 8: Knowledge Graph interaction example for a deployment with Ontop.

3.2 Storage architecture

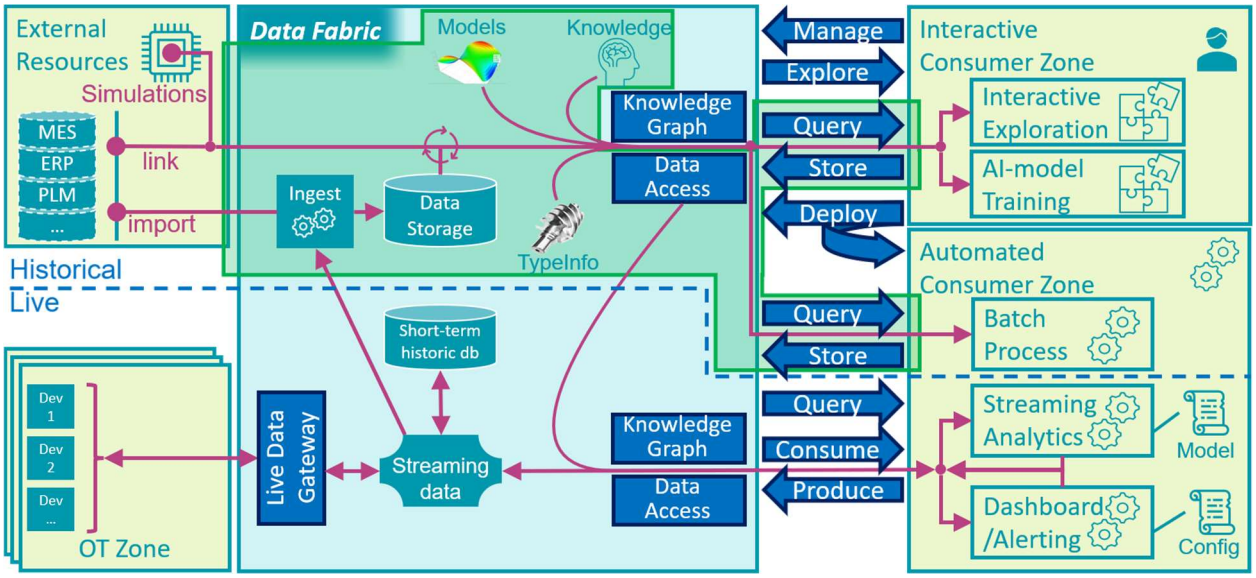

Figure 9: Position of historical data in the storage architecture of the Data Fabric (Green area)

A Data Fabric is a system that provides a unified architecture for the management and provisioning of data. To promote flexibility and scalability, Data Fabric storage architectures are typically realized as service-oriented distributed systems where (sets of) services provide consistent interfaces to, and mechanisms for, access to data and storage capabilities. The distributed nature of Data Fabrics allows scaling, flexible deployment, and adaptation of systems, and is often leveraged to, e.g., facilitate the integration of systems across organizational boundaries or combine the use of on-premise and cloud-based resources. In addition, while primarily designed as substrates for data management and system-to-system communication, Data Fabrics can also expose interfaces and tools to end-users to facilitate the development of mechanisms for convenient management, search, and analysis of data.

In pursuit of the objectives outlined in deliverable D6.1, ASSISTANT will make the following contributions towards the development of a Data Fabric storage architecture for adaptive manufacturing:

a) define a reference storage architecture for Data Fabric systems for adaptive manufacturing,
b) provide a proof of concept implementation of the storage architecture, and
c) provide simulation- and optimization-based tools for intelligent management and orchestration of the Data Fabric storage architecture subsystems.

The Data Fabric storage architecture will provide data storage and management functionality for the tools developed in ASSISTANT. As manufacturing data are by their nature complex and diverse, the Data Fabric will (as previously described) define integration interfaces and tools for data representations using state of the art domain models. The remainder of this and the following subsections of Section 3.2 outline and present the design rationale for the historical data storage architecture of the Data Fabric (illustrated in Figure 9).

3.2.1 Multi-layer service structure

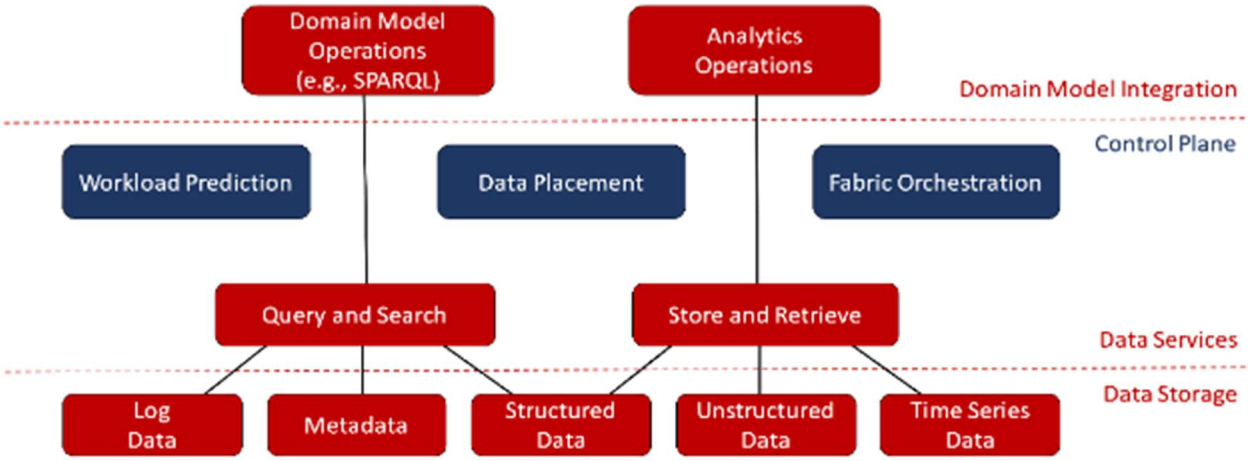

Figure 10: Conceptual overview: Layered service-oriented storage architecture for the Data Fabric

The storage architecture of the Data Fabric will be fully defined at implementation level (i.e. with service instantiations, interface specifications, and documentation) in deliverable D6.3. Here the Data Fabric storage architecture is presented with conceptual functionality groups as illustrated in Figure 10 (detailing subsystems of the highlighted "historical data" area of Figure 9).
As illustrated, the Data Fabric architecture is defined as layered service-based architecture with functionality stratified in four layers (bottom-up):

- a data storage layer provides functionality to store, access, and manipulate data;
- a data service layer provides contextual access and workflows for methods to locate and access data;
- a (for external clients hidden) control plane layer provides functionality to orchestrate and optimize the internal operations of the Data Fabric services;
- and (at the top) an integration layer facilitates integration of the core Data Fabric services and external (e.g., digital twin based) tools.

To facilitate deep and efficient integration with users and external tools in the application domain, the Data Fabric bases the integration layer on a set of domain models that can

represent a deeper understanding of the requirements and operational contexts of external tools.

As illustrated in Figure, the Data Fabric storage architecture exposes functionality in the form of network-accessible services. More specifically, the storage architecture services are rendered as RESTful web services providing JSON data over HTTP request/response exchanges. To facilitate ease of use and integration, the storage architecture also provides an abstraction layer on top of this in the form of APIs available in the (for the ASSISTANT project) most common programming languages and environments. For ease of adoption, example clients detailing the use of the services in common scenarios are also planned to be provided (in conjunction with full documentation of the service interfaces provided in deliverable D6.4). In addition, for optional high-performance exchanges, service implementations building on Google protocol buffers are also planned to be investigated. If such services are provided they will in interface conform to the REST service interfaces (i.e. not impact the overall design of the storage architecture) and be wrapped using the same type of APIs. Together with the previous section, this complies to the needs expressed in deliverable D2.1 section 3.4.1 concerning information & technical interoperability.

In summary, the full technical realization of the Data Fabric architecture will be documented in deliverables D6.3 and D6.4. The remainder of this section focuses on the design philosophy and the technical design choices of the system at architecture level. As outlined in Figure 10, the Data Fabric storage architecture is designed as a layered data services architecture. This design allows individual services (and clients to the services) to be implemented using different types of well-established technologies such as Java, Python, REST and Google protocol buffers. Services are abstracted behind well-documented APIs for easy integration, and example client implementations will be provided in representative languages. The architecture defines a set of core services with exchangeable reference implementations and has customization points in the form of plug-ins for adaptations and integrations. As previously described, the Data Fabric supports the use of knowledge graphs for integration with other tools and has built-in metadata structures enabling lightweight computational systems for, e.g., reactive data curation, advanced search, and data federation. In summary, the main design builds on open, freely available technologies and targets flexibility to promote adoption in the manufacturing industry.

3.2.2 **Data storage**

Key knowledge outcomes of the initial use case interviews and requirements gathering process documented in deliverable D6.1 included that the vast majority of all data needing to be stored, processed and used in the ASSISTANT digital twins can be considered (what we refer to as) historical data: data produced by either equipment or tools that will be utilized in other parts of production pipelines later on, but not necessarily at a known time (i.e. the data is potentially produced and used at different times).
Another learning outcome closely connected to the storage aspect is that data is rarely produced in the format it will be consumed, i.e. the data needs to be curated, filtered, aggregated, or processed before it can provide value to other systems or tools. This is generally true for most AI-based systems (and perhaps in particular for AI systems incorporating machine learning elements), and as such the ASSISTANT Data Fabric storage architecture is designed to meet these requirements. As illustrated in Figure 10, the Data Fabric storage architecture recognizes five different types of data:

- Structured data: data where the structure and format of the data carries information itself, in addition to the content of the payload data. This can include, e.g., workflow specifications, JSON files, schema validated XML documents, database tables or CSV files

where the presence and structure of the documents themselves indicate or influence the semantics for interpretation of the data.

- Unstructured data: in contrast to structured data, unstructured data is here defined as data where the structure of the data does not carry information in itself (beyond semantics needed to access and decode data, e.g., compression formats) at the time of storage. Unstructured data can be, e.g., video surveillance data, screenshots, binary log files, or other types of data that are temporarily stored as binary files for later processing.

- Time series data: the key characteristic of time series data is that they are composed of sequences of data points indexed in time order. Typically time series measurements involve repeated measurement of simple (one-dimensional) or complex (aggregated) metrics streamed for processing. Their value lies in the observation or analysis of statistics of these metrics. For time series data, the Data Fabric storage architecture systems are primarily geared towards storage, access, and offline data processing. Intrasoft's Streamhandler platform instead performs online analysis of streaming data (see section 3.3).

- Metadata: Metadata (from "meta" - Greek "after" or "beyond") is "data about data". In the storage architecture, all data is associated with a metadata document that contains information about the data or the process/context in which the data was created. This could include, for example, information about what tables are present in a database, what machines were used when data were produced, what parameters were set in simulation experiments, or data needed to track the provenance of data in pipelines that process data in multiple steps.

- Log data: Finally, the log data stored in the storage architecture are collected from the Data Fabric services themselves. These data are used for configuration control, debugging, and resource management of the Data Fabric control plane services.

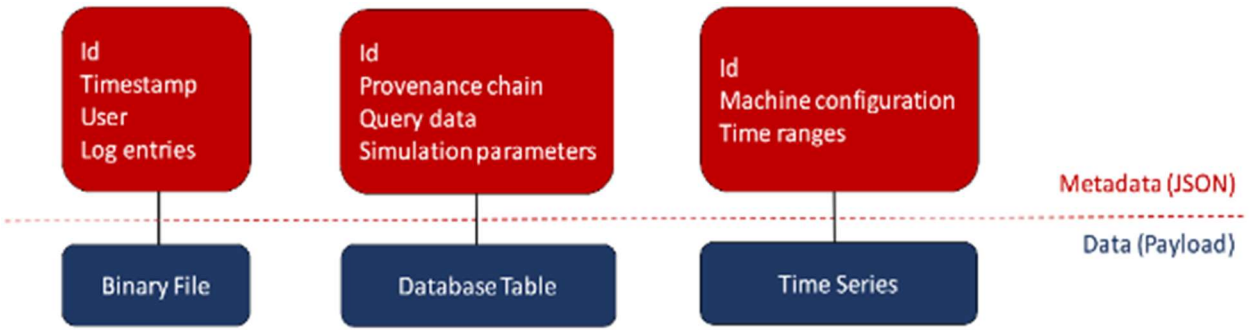

Figure 11: All storage architecture data items are stored with associated metadata documents. Example data illustrating tags with details of the data origin and provenance.

With this view and classification of data, the Data Fabric storage architecture can classify and store different types of data in different ways, enabling differentiable treatment and processing of data. Note that data is not static in the storage architecture, unstructured data can, for example, be transformed to structured data through processing (e.g., if a binary log is parsed by a program to produce a semi-human-readable JSON file). Furthermore, due to requirements for data persistence, data can optionally be stored with copy-on-write semantics, having new copies of modified data sets/items created when data is processed and updated. In contrast, the original data remains unaltered (defined due to an expressed need for always having original data sets available for processing).

A key benefit for defining an architecture for data storage as a service-oriented architecture is that a level of indirection between the storage mechanism (data sink/source) itself and the service interfaces is gained. For example, the storage architecture can provide a unified service interface to store multiple types of structured data and automatically redirect the storage of JSON documents to a file system and relational data sets to a SQL database. Furthermore, data can also be redirected to storage solutions based on policies and metadata interpretations, e.g., storing CSV files in spreadsheet or SQL databases. Additionally, the service structure also provides levels of indirection among the services, allowing, e.g., abstraction of service interfaces, load balancing among services, resilience and dynamic updates of service implementations without service interruption etc.

For the proof of concept implementation of the storage architecture, a reference implementation of services populating each of the layers in Figure will be provided in deliverable D6.3, complete with connections to common data sources such as SQL databases, NoSQL databases, time series databases, spreadsheets, CSV files, and file system based storage (including text-resolved storage formats such as JSON and XML) as needed by the project. To avoid implementation and performance overhead in the reference implementations, service implementations will expose native data and query formats when needed and provide facilities to, e.g., make domain language transformations for higher-level query languages in domain models. At the architectural level, all design decisions have been made with the perspective that the architecture should be extensible to adapt to new requirements and support flexible integration of new technologies.

3.2.3 **Data services (search, queries, and access)**

Aggregating the functionality of the data storage layer, the storage architecture provides advanced functionality for search, queries, and access of data in the data service layer. As illustrated in Figure, these services are organized into two main groups:

- Query and Search services provide functionality for identifying, locating, organizing, searching, and aggregating information in data. These services primarily operate on metadata, e.g., composing virtual data sets based on matching metadata tags associated with stored data, but can also operate directly on data of known structure, e.g., returning lists of data set identifiers for structured data sets matching queries.
- Storage and Retrieval services provide interfaces for managing data in more advanced ways than the lower-level interfaces provided by the data storage layer services. For example, a data storage layer service may provide an interface for storing data in a file with a specific filename or in a specific database server, whereas a storage and retrieval service may choose where and how to store the data for the requesting entity (be it system or end-user). Similarly, lower-level services may provide interfaces for retrieving data from specified sources, while higher level retrieval services expose interfaces for retrieving data based on global namespace identifiers rather than storage-level identifiers such as server IP numbers and local filenames.

Based on these abstractions, access to and exposure of data is intended to be provided at a more abstract level, facilitating integration with domain models and providing a level of indirection for data management as knowledge entities rather than bits and bytes stored at physical devices. Service interfaces will naturally stipulate limitations on e.g., the format of identifies and structure of metadata documents, but the design philosophy is to decouple data from conventions imposed by underlying platforms, systems, and technologies to pursue more abstract ways to represent and process knowledge and value from data.

Translation of data from underlying storage representations to useful reasoning system representations is application dependent and either captured by domain models or

encapsulated in plug-in logic (see section 3.2.4). The Data Fabric storage system provides mechanisms to store, process, and manage data but does not generally understand the content of the data beyond what information is made available in the associated metadata. For this reason, the design rationale of the storage architecture is “provide mechanism, not policy”, or in other words, facilitate processing and management of data but not assume the responsibility of owning data or being part of applications. Note that this does not exclude placing parts of application workflows inside the Data Fabric (or even in the storage architecture, see plug-ins below), but the perspective of the Data Fabric is to provide a system for unified access and management of data, not to be part of applications of digital twin systems.

Figure 12: Example data fabric workflow
External equipment and systems store data in the data fabric. Metadata tags are scanned upon storage and plug-ins are triggered based on preconfigured rules to perform light-weight processing of data, e.g., data curation. Once stored, data is made available to external tools through service interfaces. Data fabric clients, e.g., digital twins, make use of the data and may further interact with the data fabric systems as part of their processes.

To illustrate this distinction, consider the simple example illustrated in Figure 12: a digital twin aims to produce AI-based models to predict manufacturing capacity for a factory. The factory equipment stores data in the Data Fabric using the storage services, and plug-ins automatically tag all stored data with metadata about the data's origin, time span, and context. The digital twin then identifies and assembles a virtual data set composed of all relevant data (using the query and search services to identify data sets based on the metadata, and retrieves selected data using the storage and retrieve level services) and proceeds to train an AI model on the data using resources located outside the Data Fabric (e.g., machine learning algorithms executing on an off-site compute cluster. Then stores the trained model for later use in the digital twin inside the Data Fabric. To enable this scenario, plug-ins that can recognize and interpret incoming data must be pre-deployed and contain logic for correctly processing the data/metadata. In essence, this design embodies the principle of *separation of concerns*: the role of the Data Fabric storage architecture is to manage data, and while it may provide facilities for (lightweight) processing of data through the use of the plug-ins, the external systems (e.g., the digital twin) using the Data Fabric remain responsible for providing semantical understanding and logic for processing of data.

This design perspective extends (by necessity) also to the treatment of the format of data. Data is naturally occurring in many different formats and shapes in manufacturing, and a Data Fabric storage system must be able to deal with the diversity and heterogeneity of data (a key learning outcome of deliverable D6.1). As such, data can be stored in many different formats in the Data Fabric, and a design choice needs to be made whether to try to provide the earlier discussed data abstraction at storage or processing level. Following the earlier discussed design philosophy, the storage architecture forwards data in its stored format to processing logic (i.e. logic housed in plug-ins or external tools) and does not attempt to provide substantial abstractions in its service interfaces.

3.2.4 Compute and processing

To facilitate lightweight data processing and curation, the storage architecture exposes customizable plug-in structures at the service level where data processing and interpretation logic can be inserted to, e.g., convert data representations, filter outliers in data, or trigger customizable reaction to events in data streams (as illustrated in Figure 13). For language, deployment, and platform flexibility, plug-in logic is encapsulated and deployed as containerized services that can be invoked from storage architecture services (but not externally). Using this scheme, service realizations can be implemented using different languages and toolkits, e.g., Java RESTlets, and flexibly combined with data processing plug-ins implemented in other languages, e.g., Python scripts.

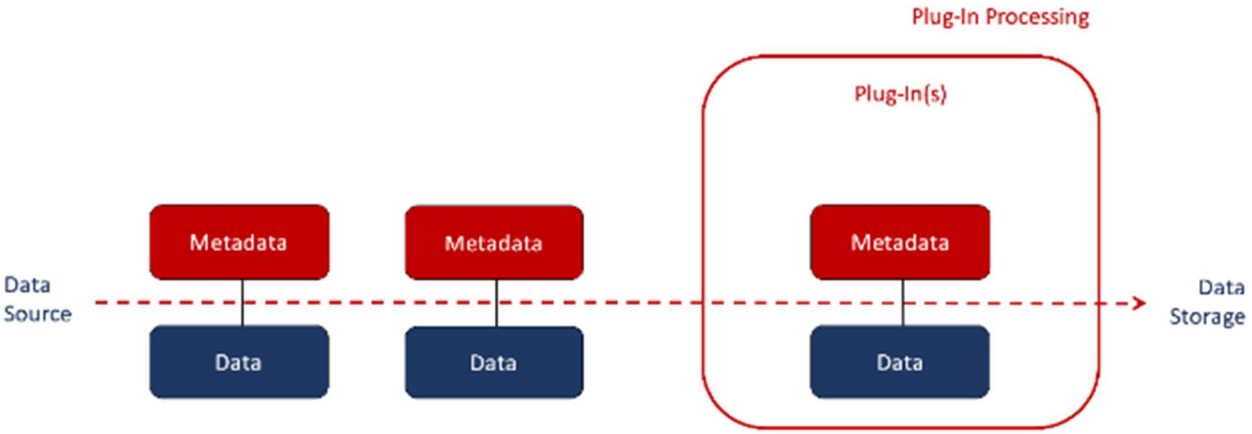

Figure 13: Detailed workflow: plug-in may be reactively triggered based on detection of preconfigured metadata tags. Plug-ins may operate on metadata or data, and may update, delete, or create new data items or metadata tags as part of their processes.

For technical reasons, limitations on the allowed processing time and resource capacity (e.g., amount of CPU cores or RAM) are imposed on plug-ins. The plug-in structures of the storage architecture are not intended to be viewed as a general compute cloud construct and should not be used for computationally intensive or time-consuming tasks such as simulations, machine learning model training or data analytics tasks. Instead, they are intended for short-lived limited complexity tasks such as data format translations or metadata analysis and classification. Data produced by plug-in processing are either stored as conventional data (e.g., in data format translation or filtering) or as metadata (e.g., in the case of metadata tagging or aggregation of data sets).

3.2.5 Control Plane

As illustrated in Figure 10, the storage architecture houses functionality for resource management within the Data Fabric in a dedicated control plane. The control plane services define functionality for primarily three areas:

- Workload prediction: algorithms and tools that predict the workloads and capacity needs for the Data Fabric storage architecture services. These services and tools are used to (using statistical approaches) analyze the resource needs and utilization rates of infrastructure resources and can also be used to dynamically adjust service deployment patterns to meet changes in resource requirements.
- Data placement: heuristics and algorithms that evaluate technical and business trade-offs for data placement within the Data Fabric system. Operating on metrics, such as cost efficiency and technical performance (e.g., response time or query throughput), and constraints, such as GDPR regulations and legal or policy compliance; data

placement tools can inform and guide operators and systems in the management of the Data Fabric services.

- Data Fabric orchestration: algorithms and systems for policy-guided automation of system deployment and configuration. Data Fabric services are designed to be flexibly using modern deployment tools and are envisioned to be managed like cloud native applications deployed across heterogeneous resource landscapes (e.g., combining on-premise, edge, and cloud resources).

The storage architecture control plane will be designed for inter-operation with the Data Fabric simulator (concurrently developed in Task T6.3), which will also be used as the basis for the demonstration of orchestration results developed in Task 6.5.

3.3 Live streaming architecture

This section highlights the streaming capabilities of the ASSISTANT solution. As depicted in Figure 14, four main components compromise the streaming solution: the data gateway, the data access, the data short-term storage, and the streaming infrastructure. An explanation of each component will be given next.

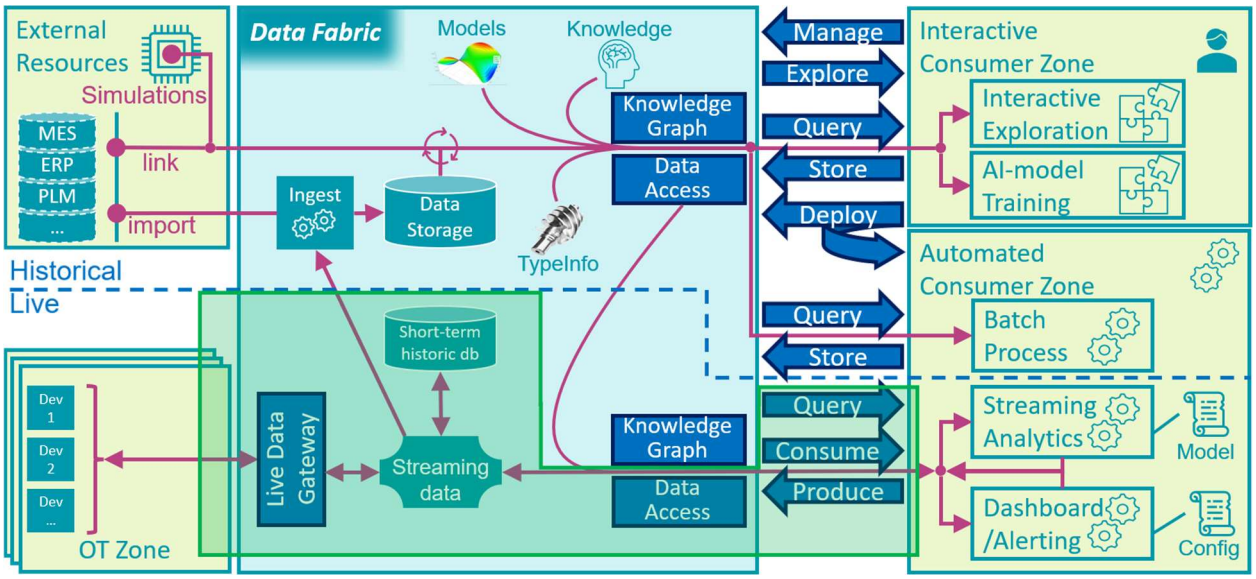

Figure 14: Position of Live data in the Data Fabric (Green area)

3.3.1 Streaming infrastructure (Streamhandler platform)

This component is a distributed event streaming platform that enables event-driven monitoring and event processing and a distributed messaging system providing high availability and horizontal scaling. Figure 15 depicts the platform architecture. In the middle section of the figure - based on the Apache Kafka - is the Streaming core platform with the data connectors and the schema registry. On top are the platform administrator tools for monitoring and fine-tuning the platform and security management tools. On the left upper side relies the data sources (OT Zone) while on the left bottom side is the Data Fabric persistence storage. Finally on the right side of the figure (Automated Consumer Zone) lie the streaming client application that use the streaming data for various applications including data analytics, dashboarding etc.

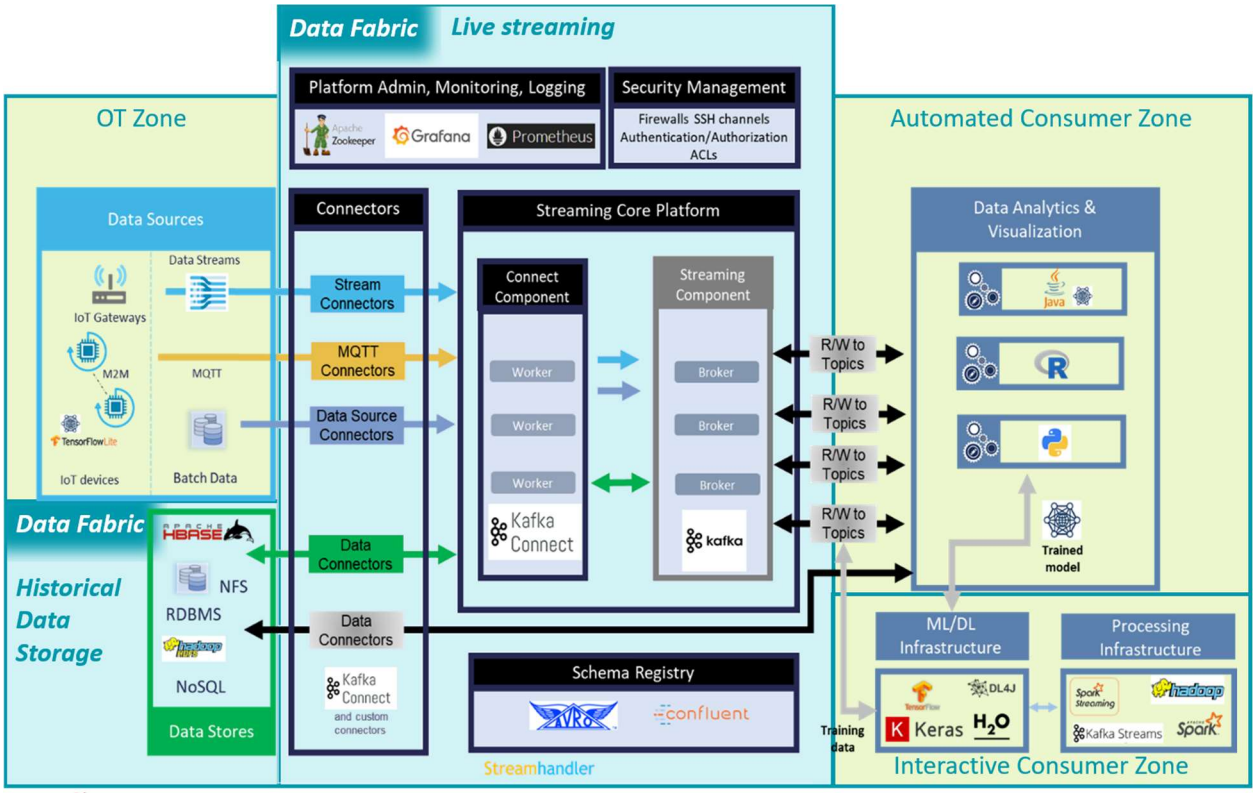

Figure 15: Streamhandler platform centrally in the Live streaming zone

The main concept of the Streamhandler platform is that events and/or messages consist of records. When writing data to Streamhandler, then for each record, a key/value pair is required. The value is the record itself the key is representing the nature/context of the data. For reading values from the platform it must be provided a key which means that it must be notified the platform for which records (nature/context) the reader wants to be notified.

3.3.1.1 Short term historic db

Streamhandler as having in its core Apache Kafka and leverages Kafka's functionalities. Kafka is designed to serve as a "source of truth store" for data, and as such, it can be used as temporal and permanent storage if the use case requires it. Moreover, it provides a fault tolerant way of managing messages, storing them into files (log files) which can be replicated across different machines like a distributed file system (allowing scaling). Nevertheless, the Data Fabric architecture does not impose a need of permanent data storage but temporary storage will facilitate the development of data processing jobs that perform computation on the data and store their result, allowing them to change their processing code and recompute the results (important especially in the development phase). Additionally, streaming analytics may use this functionality to avoid persistent storage and populate their internal cache from the temporary storage every time the application starts re-reading all the messages stored in the temporary storage

3.3.2 Data Gateway & Data Access components

In principle, the main actors of the platform are the data producers (databases, data gateways, etc) and are depicted on the left side of the platform architecture (Figure 15), while on the same figure on the right side are the data consumers (Data Analytics & visualization, persistence storage etc.) Figure 16 shows the data producers on top pushing the data in the platform (Kafka cluster) and the consumers at the bottom receiving data. There are also two other minor actors which are the connectors and the stream processors. The connectors in an efficient way provide

data stored on external databases to be used by the data consumers, while the stream processors are a special kind of data consumer which pushes back the result of the data processing.

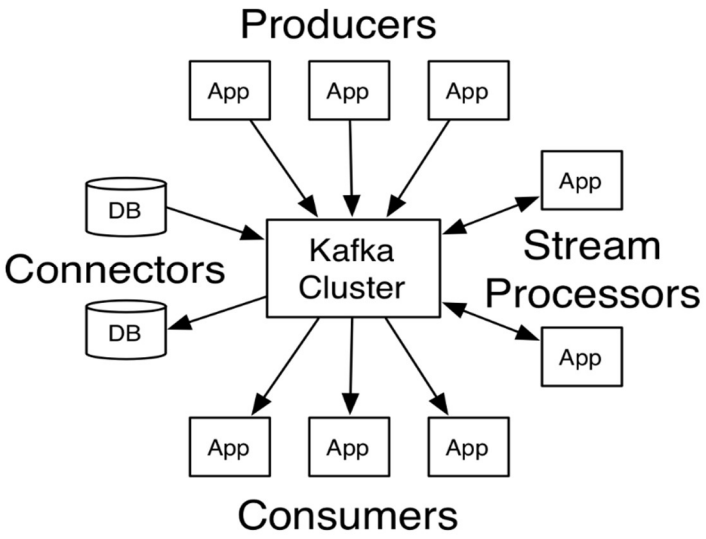

Figure 16: Data producers and data consumers

3.3.2.1 Data Gateway

The Data Gateway component is a data producer in the Streamhandler ecosystem. The gateway plays the role of a mediator between the platform itself and the actual devices on the shopfloor which generate the data. Moreover, it integrates these devices, offering a unified way of accessing the data itself. The gateway's role in ASSISTANT is to monitor the changes happening on the shopfloor and forward them to the core platform to be analysed and stored. As data producer, the data gateway has to implement a specific API provided by the platform supporting multiple programming languages. Although the Data gateway is shown as a single box on the figure, a final deployment probably consists of many smaller data gateways, each focusing on the part of the data.

3.3.2.2 Data Access

The Data Access component is a data consumer in the Streamhandler ecosystem. Implementing a specific API (provided by the platform supporting multiple programming languages) is notified upon data existence as preconfigured.

4. User view

During the project, the Data Fabric will have several users:

- The research work packages: WP3, WP4, and WP5
- The industrial users: AC, PSA, SE
- The demonstration work package: WP7

As described in D6.1 Data Fabric Requirements, the Data Fabric should be suitable to serve all of these users' needs. This does not mean there should be a single Data Fabric instance that addresses all needs. Users need to be able to deploy and govern their own Data Fabric. Indeed, large parts of the Data Fabric's implementation for a specific user will depend on the deployment needs of the individual user (see 2.4).
For each user, this chapter:

- mentions the concrete case for the user (more detail to be found in D6.1),
- indicates which parts of the architecture are relevant to the user,
- maps the generic interactions to the way each user will use these interactions. The Deploy interaction is not mentioned in this table, since all users will need to deploy the mappings between the user-specific Knowledge Graph and the underlying historical or calculated data. The difference between users' deployed contents does not imply any technical differences in the infrastructure.

The purpose of this chapter is to mitigate any inconsistencies between the users' expectations and the proposed architecture. More detailed deployment and implementation decisions will be taken in the following stages of the project.

4.1 WP3 - Process Planning

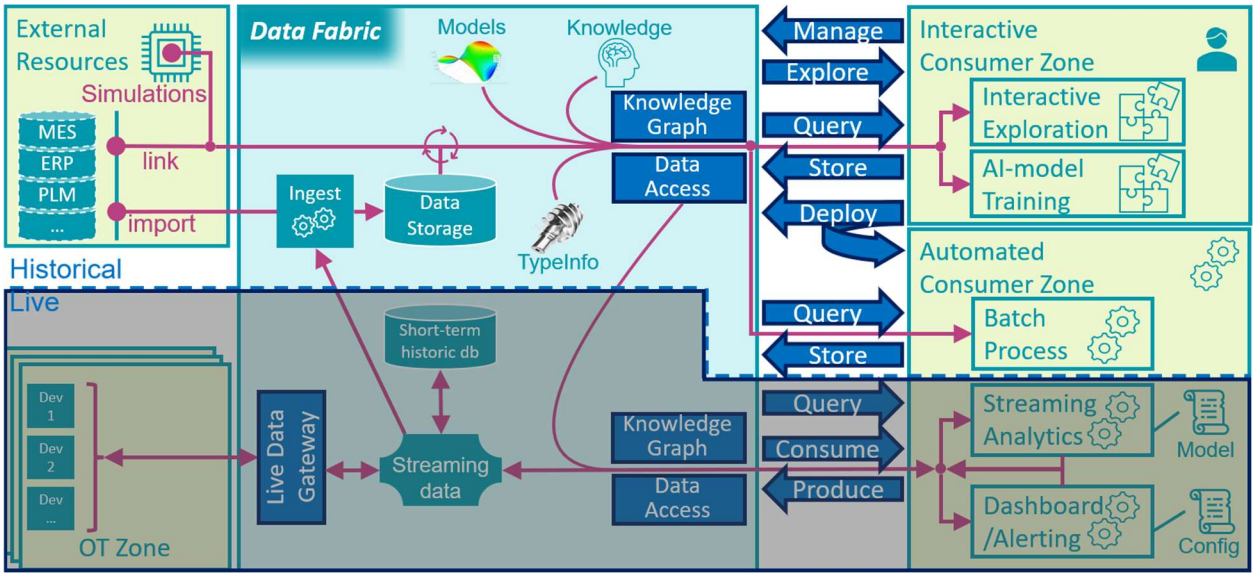

Figure 17: Scope of interest to WP3 (shaded area out of scope)

WP3 uses the Data Fabric for the following tasks::

- T3.2 Process engineer in the loop
- T3.3 Digital twins enrichment and simulation for process planning
- T3.4 Predictive analytics for process planning
- T3.5 Prescriptive analytics for robust process planning

In order to accomplish these tasks, WP3 has the following interactions with the Data Fabric:

			Interaction
Historical	KG	Explore	Understand the data which is available concerning process planning. • Production System layout • Resource library (devices allocated at the production lines and devices in the storage – reconfiguration) with included skills/capabilities • Product (variants) and processes (requirements) • Process plans
		Query	Find historical data, find model as blob (e.g., CAD) and extract model structure and constraints (e.g., of intended process graphs). WP3 queries data through the KG (stored in the data fabric). • Product data (CAD; process data) • Production/assembly system data (CAD models and layout (e.g. JSON)) • Resources (CAD models, skill models, resource parameters) • Historical data (Changes; KPIs - Times, Quality, Costs)
		Store	• Store additional concepts and relationships in the Knowledge Graph suitable for WP3 planning context (data formats: .owl; .json; etc.) • Store output models of WP3 modules (e.g., product structure, production system model, process plans, associated KPIs of each process step). • Store result of data analytics, providing right parameters (e.g., stochastic) for process plan optimization.

	Data	Query	The Data Fabric ingests company-specific systems relevant to process planning (ERP, PLM…).
			WP3 queries this data during its analysis.
			• Product data
			  ○ CAD models
			  ○ Additional process data (e.g. screwing parameters, joining force)
			  ○ Additional requirements data (e.g. process relevance; critical process --> need of monitoring; e.g. process data needed due to licensing (medical sector))
			• Production/assembly system data
			  ○ CAD models and layout (e.g. JSON)
			• Resources (process execution and logistics)
			  ○ CAD models
			  ○ Skill models
			  ○ Resource parameters
			• Historical data
			  ○ KPIs per process-resource combination (process and product quality – e.g. success in percentage)
			  ○ Historical process times
			  ○ Historical process quality
			  ○ Historical process quality
			  ○ Historical change data (change description and impact (resulting KPIs))

		Store	WP3 stores its predicted planning and corresponding models into the Data Fabric. • Process plans (and process monitoring plans --> as .json)

			• Product, Production, Process Graph • Continuous data upgrade
Live	KG	Query	WP3 does not directly interact with the live zone. The historical production data arrives from the OT zone into the historical Data Fabric, but these aspects are not the focus of WP3.
	Data	Consume	
		Produce	

4.2 WP4 – Production Planning & Scheduling

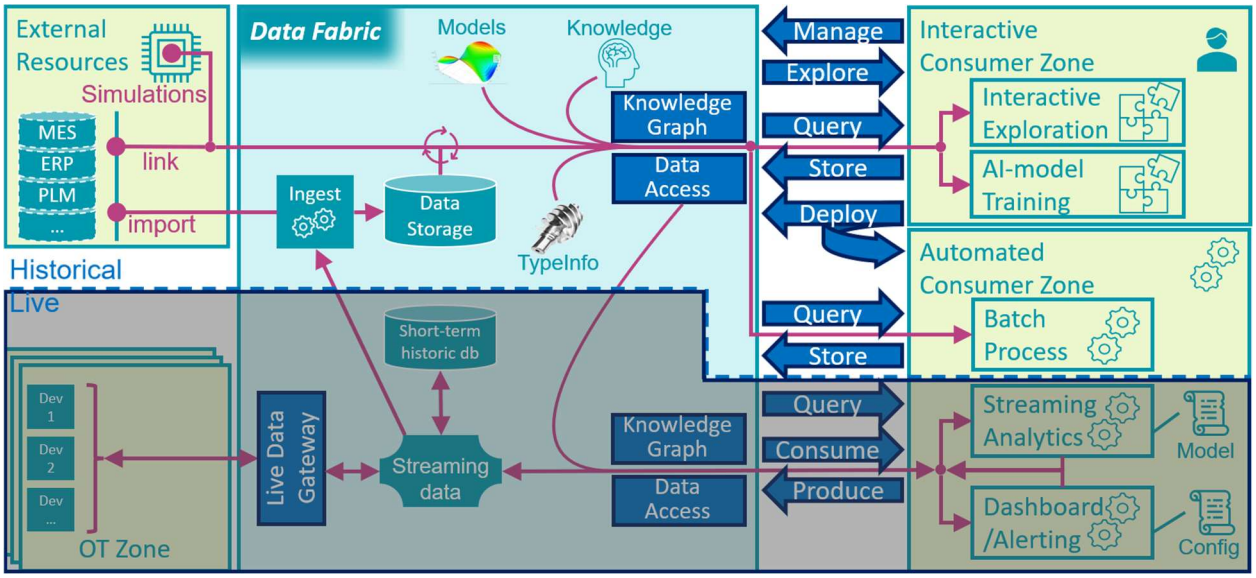

Figure 18: Scope of interest to WP4 (shaded area out of scope)

WP4's needs for the Data Fabric are similar to WP3's at architectural level. The Data Fabric is used in the following tasks:

- T4.2 Production Manager in the loop
- T4.3 Digital twins enrichment and simulation
- T4.4 Predictive and prescriptive analytics for production planning
- T4.5 Predictive and prescriptive analytics for scheduling

The next paragraph contains the WP4 interactions from three different perspectives because each of these aspects use the Data Fabric slightly differently:

- Simulation perspective
- Production planning perspective
- Model acquisition and scheduling perspective

4.2.1 WP4's interactions from Simulation perspective

WP4 has the following interactions with the Data Fabric from a simulation perspective:

			Interaction
Historical	KG	Explore	Understand, check, and count the available data and their relations concerning production planning and scheduling. These are • Production system data (products, BOM, processes, resources, …)

			• Production load data (reference to order lists, e.g. taken and stored as snapshot from ERP) • Simulation models (ID, reference to file) • Simulation data (simulation run ID, simulation model, use case scenario, configurations, references to input/load data, references to output/result data, KPIs) • ML data (ID, method, use case scenario, simulation runs, results) • Trained decision model (ID, reference to ML data the model is based on) • (References to) calculated plans/schedules
		Query	• Find (historic) data of the production system and production load • Find simulation models • Find previous simulation runs • Find previous ML training progress and results • Find trained decision model • Find calculated plan/schedule
		Store	• Store simulation models • Store simulation data (after execution) • Store ML data (during training) and acquired model (after training) • Store calculated plan/schedule
	Data	Query	• Get production system data (to create a simulation model) • Get production load data (to create the input data for a simulation run) The Data Fabric ingests data (production system data and production load data) from the industrial use cases (SE & AC) relevant to planning and scheduling (artificial data derived and distorted from ERP data). The precise import of this data from the use case partner systems is out of scope for the project. • Get previous simulation outputs (to be used in ML) • Get calculated plans/schedules (to view and implement)
		Store	• Simulation input/output data • ML data (training progress) • Calculated plans/schedules (using ML-trained models) (stored in the Data and referenced by the KG)
Live	KG	Query	WP4 does not directly interact with the live zone.
	Data	Consume	
		Produce	

4.2.2 **WP4's interactions from a Planning perspective**

WP4 has the following interactions with the Data Fabric from a production planning perspective:

	Interaction

Historical	KG	Explore	Understand and check available data concerning production planning. These are: • Past production plans • Products cycle time targets • Resources available for the production line • Set of existing products mix • Shift models • Capacity of the production line • Production line KPIs • Past set up models • Factory maintenance plans and historical data on the frequency of machines breakdown • Current production line uncertainties (parameters/variables)
		Query	• Find the demand per period and per end-item (provide historical data to create the forecast) • Find the amount of time consume per operation on each resource. • Find the amount of hour per period each resource is available (without extra hours) • Find the percentage of quality item resulting production lot (provide historical data to create the forecast) • Find the number of period between the release of an order decided by production planner and the period in which the order is produced/delivered (provide historical data to create the forecast) • Find the amount of time the machines/worker were running during a period.
		Store	• Store production planning models • Store production plans • Store purchasing plans • Store planned inventory • Store resource planned capacity consumption • Store information regarding the impact of uncertainty on production plans and formula learned
	Data	Query	• Set of resources (Primary resource and tools/workers shared among resources) • Flexible bill of material and bill of processes • Setup, inventory, unit production costs, extra capacity cost per resource, ... • Inventory levels for end-items and components. • Targeted KPI values given by the user. • Actual production at the end of a period (quantity of each item produced/delivered) • KPIs values from end users
		Store	• Production quantity per item/ per period • Extra capacity required per period per resource • Quantity sub-contracted • Quantity to order to suppliers • Output KPIs values

Live	KG	Query	WP4 does not directly interact with the live zone.
	Data	Consume	
		Produce	

4.2.3 **WP4's interactions from model acquisition and scheduling perspective**

WP4 has the following interactions with the Data Fabric from a model acquisition and scheduling perspective:

			Interaction
Historical	KG	Explore	Understand and check available data concerning scheduling. These data are: • Allocation and sequencing decisions from past factory production schedule
		Query	• Import some tables (e.g. a table containing the tasks and a table describing the resources) describing a schedule from which a model will be acquired. • Get the information regarding what in these tables are the input parameters and what are the variables. • Get the information of what are the global cost (if relevant). • Import a set of acquired constraints and the corresponding model.
		Store	• Record a set of acquired constraints as well as the corresponding model. • Record the schedule which was generated by executing a given model.
	Data	Query	• Set of resources • Bill of processes • Resource availability
		Store	• UML model, class diagrams in the model, textual form of the acquired model with link back to the data from which the model was acquired • Gantt chart and resource utilization • Optimization cost and main KPIs
Live	KG	Query	WP4 does not directly interact with the live zone.
	Data	Consume	
		Produce	

4.3 WP5 – RT control & actuation

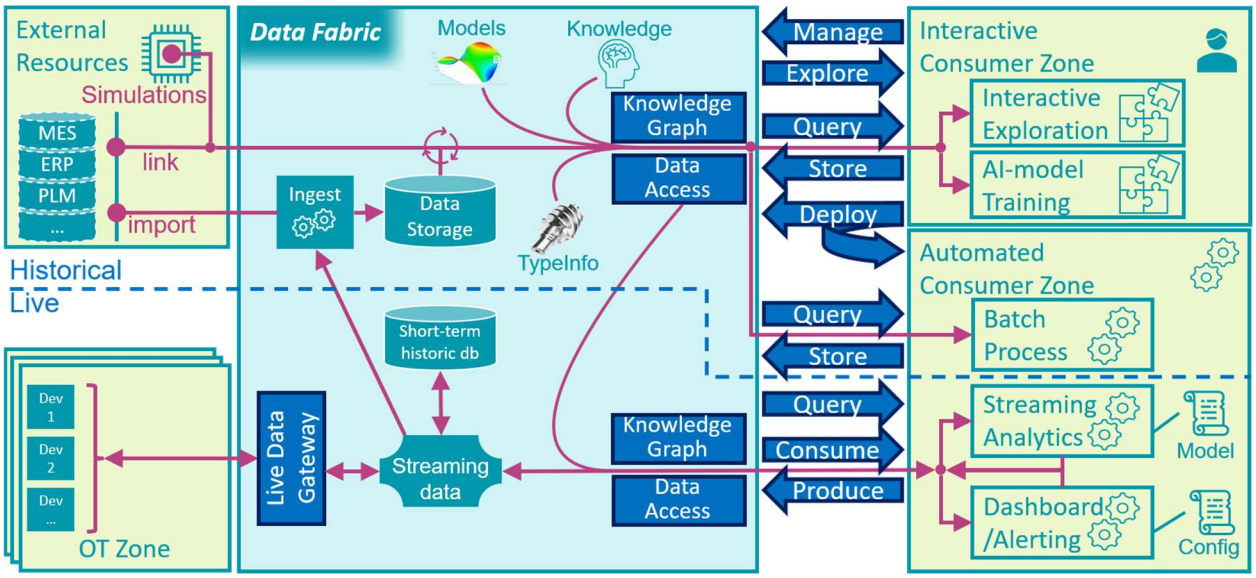

Figure 19: Scope of interest to WP5 (no shaded area -> full system in scope)

The data fabric is used in the following tasks:

- T5.2 Smart interfaces to operators
- T5.3 Digital Twin for Execution
- T5.4 Fenceless H-R collaboration
- T5.5 AI for dynamic process execution and quality control

In order to accomplish these tasks, WP5 has the following interactions with the Data Fabric:

			Interaction
Historical	KG	Explore	Understand the data available in WP3 & WP4. Determine which WP5 specific concepts and individuals are available (e.g. which sensors)
		Query	Find historical data, and find the location of streaming data.
		Store	Store additional concepts and relationships in the Knowledge Graph suitable for WP5's real-time execution context. e.g. Store resource suitability per process/task
	Data	Query	Query historical data (e.g. flow of human actions, repeated mistakes) for the purpose of training models for the Digital Twin for Execution, structured (SQL alike) and time series. Query data and models from WP3 & WP4.
		Store	Store simulation models, application configuration data, etc.
Live	KG	Query	Find the right sensors and data streams, etc.
	Data	Consume	Ingest OT data for applications in the Automated Consumer Zone, such as Monitoring, Dashboarding, Digital Twin for Execution models. This includes live sensor data.
		Produce	Store knowledge obtained during operation (e.g., number of robot adjustments due to obstacle), and from live simulations (e.g. Resource execution errors, Safety violations, Robot Trajectories, Process Quality results, Simulation models, Current Process, Valid process plans, Updated data or constrains such as new duration of operations,

			geometry/collision errors for the corresponding resource or operations).

4.4 Atlas Copco

The scope of the use case's interest is the full Data Fabric, including historical and live data.

The use case serves two goals:

- Measure as little as possible while maintaining the same quality for machined parts.
- Make new operators as experienced as an employee with 20 years of working experience in the machining area.

			Interaction
Historical	KG	Explore	Browse & understand the domain-structure of the data, independent of its underlying technical structure.
		Query	All structural data will be queried through the KG (most of it virtually).
		Store	Experiments performed to develop the AI model will be stored for future reference & reproduction.
	Data	Query	Lots of time series of physical properties measured, and the results of quality tests performed during past production runs are required for AI model training.
		Store	Ingestion of time series from the live zone. Import of PLM/MES data related to product and planning. Such historical data imports are out-of-scope for the project, since they remain under strict control of AC.
Live	KG	Query	None foreseen
	Data	Consume	Consumption of live data will only happen to a limited degree. The proof-of-concept Data Fabric will not be directly connected to a real AC production environment. Instead, the required OT devices will be stimulated to produce the required inputs. Quality case: • Consume the property values required for the computation of the observations of the AI model. Mostly temperature readings, remaining useful life of the used tools, vibrations, and more. Operator instructions case: • The AI algorithm monitors all properties of the circumstances in which an operator is working (day of week, skills, previous errors, temperature, current task…) for consumption.
		Produce	Quality case: • Produce the verdict "measure yes/no", i.e. classify a part as having or not the sufficient quality level o continue being processed in the manufacturing process. This will be consumed by the operator and the historic side. The latter will use it to monitor / re-train the AI algorithm. Operator instructions case:

			• Produce the selection of instructions that are suitable for this operator in these circumstances.

4.5 PSA – Stellantis

The scope of the use case's interest is the full Data Fabric, including historical as well as live data.

The use case serves the following goal:

- Optimization of assembly process of an electrical drive-line

			Interaction
Historical	KG	Explore	
		Query	Production schedule Assembly graph Resource suitability table All valid Process graphs Desired/Expected KPIs
		Store	
	Data	Query	CAD files related to process, workplace layout
		Store	Reported issues corresponding to the execution of a process by resource, Achieved KPIs, Recordings of process data for future training usage
Live	KG	Query	
	Data	Consume	
		Produce	Resource working time Resource execution status Resource execution errors Safety violations Robot Trajectories Process Quality results Current Process CT

4.6 Siemens Energy

The use case's scope of interest is equal to that of WP3/4, so making use on the historical part of the Data Fabric only.

The use case serves the following goal:

- Optimization of a up to two-year planning and a shorter scheduling horizon on operational data. Simulation (e.g. based on Tecnomatix Plant Simulation) will be an essential part of this optimization.

			Interaction
Histo	KG	Explore	Understand, check, and count the available data and their relations with respect to production planning and scheduling. These are

			• Production system data (products, BOM, processes, resources, …) • Production load data (reference to order lists, e.g. taken and stored as snapshot from ERP) • Simulation models (ID, reference to file) • Simulation data (simulation run ID, simulation model, use case scenario, configurations, references to input/load data, references to output/result data, KPIs) • ML data (ID, method, use case scenario, simulation runs, results) • Trained decision model (ID, reference to ML data the model is based on) • (References to) calculated plans/schedules
		Query	• Find (historic) data of the production system and production load • Find simulation models • Find previous simulation runs • Find previous ML training progress and results • Find trained decision model • Find calculated plan/schedule
		Store	• Store simulation models • Store simulation data (after execution) • Store ML data (during training) and acquired model (after training) • Store calculated plan/schedule
	Data	Query	• Get production system data (to create a simulation model) • Get production load data (to create the input data for a simulation run) The Data Fabric ingests data (production system data and production load data) from SE relevant to planning and scheduling (artificial data derived and distorted from ERP data). The precise import of this data from SE systems is out of scope for the project. • Get previous simulation outputs (to be used in ML) • Get calculated plans/schedules (to view and implement)
		Store	• Simulation input/output data • ML data (training progress) • Calculated plans/schedules (using ML-trained models) (stored in the Data and referenced by the KG)
Live	KG	Query	The SE use case does not directly interact with the live zone.
	Data	Consume	
		Produce	

4.7 WP 7 – Demonstrator

Since WP7 is intended to demonstrate the contributions of all parts of the project, its scope is obviously the full Data Fabric, including historical as well as live data.

This work package serves the following goal:

- Demonstration of all the project's novelties on a flexible assembly setup

			Interaction
Historical	KG	Explore	Explore and understand the available data and its sources, which are of interest to the flexible assembly setup.
		Query	Find and retrieve historical data generated by past configurations and simulations by the flexible assembly, which could be spread across several data sources. Queries to the Data Fabric should be possible in function of time periods or configurations made in the past (e.g. via a unique ID). The retrieved knowledge graphs can then be used in analytical post-processing tasks.
		Store	Store relationships, mapping information and other meta data between available data stores (RDBMS data stores (SQL), Object data stores (noSQL), alternative blob data stores, etc.)
	Data	Query	Query historic data for the purpose of training models for the Digital Twin for Execution, structured (SQL alike) and time-series. Query data and models from WP3 & WP4.
		Store	Data-streams coming from the flexible assembly during execution and/or simulation runs should be stored in an Operational Historian, typically found in OT zones. Preferable this is in a form of a time-series database. Open question if a RDBMS (SQL) or object database (e.g. MongoDB) is to be used. This is dependent on the type of information that has to be stored, coming from the flexible assembly, e.g.: 1. configuration (PLM-info), 2. recipe of assembly/production, 3. commands from executors, 4. alarms and events, 5. warnings and errors, 6. ... Queries to the data store should be possible in function of time periods or configurations made in the past (e.g. via a unique ID).
Live	KG	Query	Find and retrieve data-streams coming from e.g. sensory devices internal or external to the flexible assembly, regardless of their source or communication protocol.
	Data	Consume	Ingest OT data for applications in the Automated Consumer Zone (Monitoring, Dashboarding, Digital Twin for Execution models)
		Produce	Store knowledge obtained during operation (e.g., number of robot adjustments due to obstacle) and live simulations. Same information mentioned in historical/data/store.

5. Conclusions

The Data Fabric architecture proposed in this deliverable considers the needs of the manufacturing use cases that are handled throughout the ASSISTANT project. Based on the discussions between the WP4 partners, there is a good agreement on describing, scoping and features matching the goals of the ASSISTANT project and the needs of its use cases and industrial users.
New insights will pop up during the project's execution when implementing the architecture (T6.3) or any of the digital twins. Through continuous synchronization between the stakeholders and the WP6 contributors, these insights will mature the Data Fabric's architecture, design and implementation.

6. Appendix

6.1 Abbreviations

Table 2: Abbreviations

Abbreviation	Meaning
ASSISTANT	LeArning and robuSt decision SupporT systems for agile mANufacTuring environments
DF	Data Fabric
KG	Knowledge Graph
OT	Operational Technology (devices on the factory floor)
VKG	Virtual Knowledge Graph