AI·PROFICIENT

Artificial intelligence for improved production efficiency, quality and maintenance

Deliverable 5.8

D5.8: System integration and deployment

WP 5: AI-PROFICIENT system integration and deployment

T5.5: AI-PROFICIENT platform deployment

Version: 1.0

Dissemination Level: PU

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957391.

Table of Contents

List of Figures

Acronyms

ACL	Access Control List
AD	Anomaly Detection
AES	Advanced Encryption Standard
AI	Artificial Intelligence
DB	Database
DCS	Distributed Control System
DQR	Data quality tool
DT	Digital Twin
GHO	Generative Holistic Optimization
GUI	Graphical User Interface
HMI	Human-Machine Interface
HTTPS	Hypertext Transfer Protocol Secure
ICT	Information and Communication Technology
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
NVI	Natural Voice interaction
OPC	OLE for Process Control
PEAA	Post-hock Explainable Anomaly Analysis
RAM	Random Access Memory
RDF	Resource Description Framework
RDMS	Relational Database Management System
SDDM	surrogate data-driven models
SPAA	Short-term Post hoc Anomaly Analysis
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
UC	Use Case
VPN	Virtual Private Network
XML	eXtensible Markup Language

Disclaimer

This document contains description of the AI-PROFICIENT project work and findings.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any responsibility for actions that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of the AI-PROFICIENT consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 28 Member States of the Union. It is based on the European Communities and the Member States cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors (http://europa.eu/).

AI-PROFICIENT has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957391.

Title: D5.8 System integration and deployment

Lead Beneficiary:	IMP
Due Date:	30.06.2023.
Submission Date:	30.06.2023.
Status	Final
Description	AI-PROFICIENT system integration and deployment activities
Authors	IMP, TEK, IBE, VTT, ATC, UL, TF, CON, INOS
Type	Report
Review Status	PC + TL accepted
Action Requested	For acknowledgement by partners

VERSION	ACTION	OWNER	DATE
0.1	Initial ToC	IMP	08.03.2023.
0.2	Contribution	TEK	20.04.2023.
0.3	Contribution	IBE	26.04.2023.
0.4	Contribution	ATC	10.05.2023.
0.5	Contribution	VTT	15.05.2023
0.6	Contribution	UL	18.05.2023.
0.7	Contribution	TF	31.05.2023.
0.8	Contribution	CON	12.06.2023.
0.9	Contribution	INOS	15.06.2023.
1.0	Final Version	IMP	29.06.2023

Executive Summary

In this document, it is provided the description of the activities related to the system integration and deployment of AI-PROFICIENT platform. This is a public document that gives an overview of the main achievements related to system integration and deployment within WP5. The WP encompasses five tasks, where the successful completion of these tasks has played a crucial role in ensuring effective system integration and deployment:

- Task 5.1 focused on enabling connectivity to field-level equipment, platform services, and plant management systems. Through the implementation of robust communication protocols and interfaces, the AI-PROFICIENT platform can seamlessly interact with various equipment and systems. This achievement ensures smooth data exchange and enhances interoperability. Its results have been reported in D5.1 and D5.7.
- In Task 5.2, the involved project partners developed a semantic repository and databases to efficiently store and organize data. This repository allows for structured and standardized data management, facilitating seamless integration and access to information. The provision of a reliable and scalable database solution enhances system integration and deployment processes. The results of this task have been reported in D5.2.
- Task 5.3 is focused on data protection and security . The involved partners implemented comprehensive measures to ensure the confidentiality, integrity, and availability of data. By incorporating encryption, access controls, and backup mechanisms, the system can effectively safeguard sensitive information during integration and deployment. Its results have been provided in D5.3.
- Task 5.4 included creating an interface with AI4EU data sources and services, which is presented in D5.4.
- In Task 5.5, the project team focused on deploying an AI-proficient platform. Through planning and implementation, the platform was successfully deployed, enabling the integration of AI services from other WPs into the pilot sites.

1 Introduction

1.1 Scope

The aim of this document is to provide the description of the activities in relation to system integration and deployment of AI-PROFICIENT platform. More specifically, it will provide the summary of the main activities that have been performed in WP5.

1.2 Audience

The intended audience for this document is primarily the members of AI-PROFICIENT consortium and Project Officer. More specifically, the consortium members in charge of development, integration, AI service deployment and platform implementation are expected to benefit from the content presented here. Nevertheless, since this document is public, research community and other actors working in the development of similar ICT platform are expected to benefit from the content presented in this document.

1.3 Relations to other tasks and work packages

As it is shown in Figure 1, Task 5.5 is closely related to other work packages where different components of AI-PROFICIENT platform are developed (services, HMI, etc.). In addition, this Task takes inputs from other Tasks of WP5:

- Task 1.5 – provides the system architecture
- Task 5.1 – enables connectivity to field-level equipment, platform services and plant management systems
- Task 5.2 – provides semantic repository and databases
- Task 5.3 – ensures that data are managed by considering data protection and security measures
- Task 5.4 – provides interface with AI4EU data sources and services

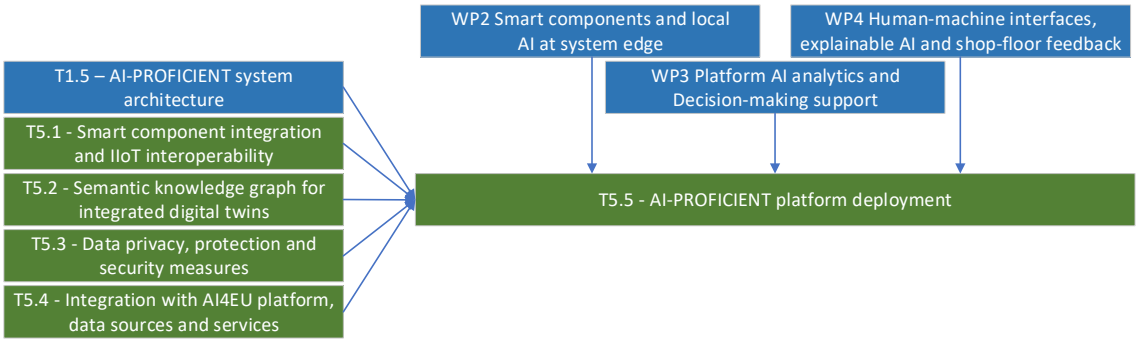

Figure 1: Relation of Task 5.5 with other Tasks and Work Packages

It is important to note that the information provided in this document focuses on the main achievements of WP5 regarding system integration and deployment. In particular, this document summarizes the main WP5 topics such as: data protection and security, communication middleware and interoperability, integration with AI4EU AI on demand platform and AI service deployment. More details regarding each of the aforementioned topics can be found in the corresponding deliverables provided within WP5.

1.4 Structure

The present document is divided into the following sections:

- **Section 1** provides the introduction to the content presented in this document
- **Section 2** provides more details regarding platform deployment:
  - Section 2.1: communication middleware and deployment
  - Section 2.2: data privacy and security measures
  - Section 2.3: integration with AI4EU platform data sources and services
  - Section 2.4: AI service deployment
- **Section 3** concludes the document

2 AI-PROFICIENT platform deployment

2.1 Communication middleware and IIoT interoperability

In order to support the AI-PROFICIENT services aimed at data processing, it was needed to deploy a platform middleware.

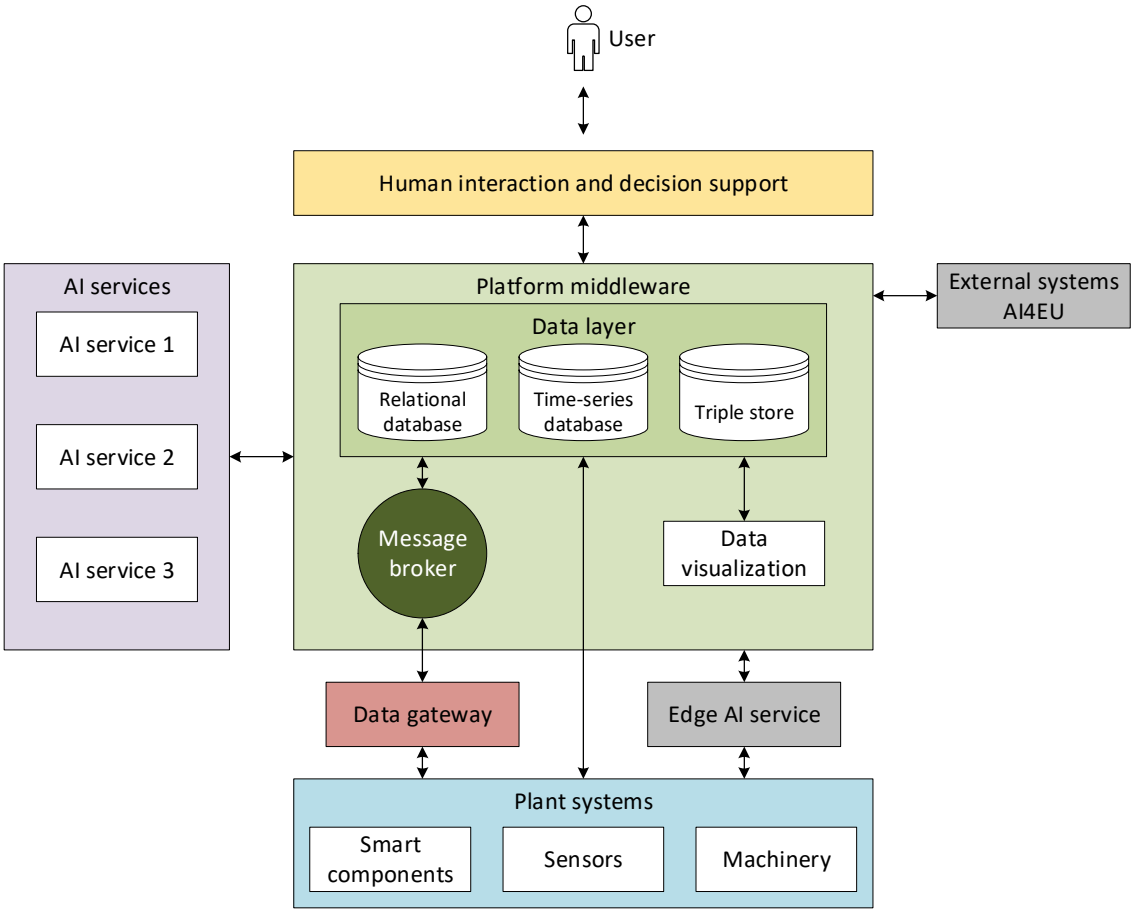

Figure 2: AI-PROFICIENT platform architecture

Figure 2 revisits the architecture of the AI-PROFICIENT platform, previously introduced in Deliverable D1.5. In the sequel, it is provided more details regarding platform middleware (depicted as a green box in Figure 2), which aims to integrate the AI-PROFICIENT services and other system components. The AI-PROFICIENT services are developed as part of WP2, WP3, and WP4 tasks. The design and implementation of the middleware was performed in Task 5.1, whereas the results are presented in more details in D5.1 (public) and D5.7 (confidential) documents. More specifically, in Figure 2, we illustrate the components comprising the AI-PROFICIENT middleware, which include a message broker and data layer which is presented in the sequel.

For Continental pilot, in Figure 3, we present different components which have been deployed and which are described in the sequel:

- Data flow from plant devices and sensors through OPC-UA server, Telegraf and Influx database
- Docker containers that represent the services deployed in different use cases (CONTI-2 to CONTI-10)

- MySQL databases that store the results of services
- Human interaction and decision support used to present the service results to the end users.

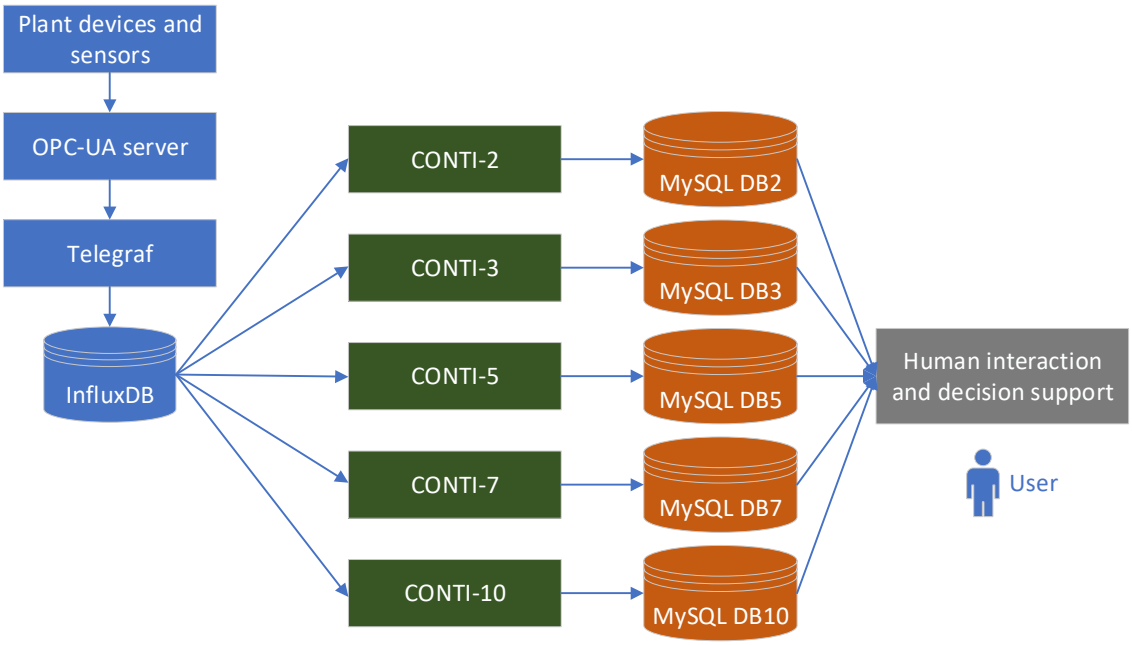

Figure 3: AI-PROFICIENT platform deployment on Continental platform

2.1.1 Data collection from plant sensors and devices

Initially, it was planned that the MQTT message broker will be deployed. However, due to pilot constraints in terms of obtaining data from plant sensors and devices, another approach that uses OPC-UA server has been deployed. However, within CONTI-7 use case, it is planned to use the MQTT broker to obtain the measurements from the Edge PC (see Section 2.4.3) For completeness, we will provide description of both of them.

2.1.1.1 MQTT message broker

The message broker plays a crucial role in facilitating communication between different system components using the publish/subscribe message exchange pattern. The widely used MQTT protocol, commonly employed in Internet of Things (IoT) applications, supports various payload types, including text and binary. The publish/subscribe pattern offers several advantages, such as allowing a single message to be sent from one publisher to multiple subscribers and enabling a subscriber to receive messages from multiple publishers. Moreover, publishers and subscribers are not required to have knowledge of each other's existence.

In contrast to the traditional client/server architecture, publishers and subscribers do not communicate directly in the publish/subscribe pattern. Instead, they rely on a messaging broker to handle the communication between them. The message broker filters messages received from publishers and delivers them to the appropriate subscribers. This approach eliminates the need for publishers and subscribers to be connected simultaneously and instead requires them to know the connection details of the message broker.

The MQTT protocol employs subject-based message filtering, where each message is associated with a topic. The message broker utilizes this topic information to deliver messages only to clients that have previously subscribed to the corresponding topic. MQTT topics consist of multiple levels separated by forward slashes (e.g., plant/machine3/sensor4/pressure). Importantly, an MQTT client does not need to create the topic in advance before publishing or subscribing to it. Instead, the broker automatically accepts the topic.

Overall, the message broker's role in the publish/subscribe pattern goes beyond simple message routing. It enables efficient message filtering based on topics, decouples publishers and subscribers, and centralizes control and coordination of message distribution. These functionalities enhance the scalability, flexibility, and reliability of system integration and deployment processes.

2.1.1.2 OPC-UA server

In the process of system integration and deployment for the AI-PROFICIENT platform a different approach was adopted to handle the data flow within the system. The data acquisition was automated through an OPC-UA server in Continental plant, which facilitated the seamless collection of data from various sources.

An OPC server acts as a gateway or middleware between the devices or software systems that collect process data (such as sensors, controllers, or data historians) and the applications that consume or utilize that data (such as SCADA systems, HMI interfaces, or data analysis tools). It abstracts the complexity of different communication protocols and data formats, providing a unified interface for accessing and interacting with data from diverse sources. The OPC server acts as a server-side component, offering a standardized API (Application Programming Interface) that enables client applications to read, write, and subscribe to real-time data, as well as access historical data. It handles the communication details and protocol conversions required to exchange data between the connected devices and the client applications.

Once the data was obtained from the OPC-UA server, it was stored in the database by using Telegraf component (see Figure 3). These data were readily accessible by the AI services (deployed as Docker containers) integrated into the platform to perform advanced analytics.

2.1.2 Data layer

The data layer of the AI-PROFICIENT platform is a hybrid collection of data storage and data management technologies fit for purpose. For the sensor readings, data technology optimised for time series has been used (InfluxDB). For providing a one-to-one connection with the semantics behind the data, an RDF triple store (Virtuoso) has been deployed. Finally, a common data technology (RDMS) for sharing the results of the AI services has been deployed (MySQL). In the figure below, the time series DB is used to collect and share the plant events, while the RDF store is used to describe the plant configuration information (e.g. the kind of sensor, its location and its purpose). The results of an AI service are stored in a MySQL database.

As deployment technology, containerization has applied (Docker). Docker is an open-source platform that allows automation of the deployment, scaling, and management of applications using containerization. Containers are lightweight, isolated environments that package an application with all its dependencies, making it easy to run consistently across different computing environments. Docker enables developers to create, distribute, and run applications seamlessly, providing a consistent and efficient way to package and deploy software. Using a containerization approach is beneficial for facilitating a complex collaboration between partners. On the one hand, they can simulate in their local environment the behaviour of their AI services without deploying it on the production environment. At the same time, integration activities when deploying the components on the target environment will not alter the behaviour of the AI service as developed. As such, Docker provides AI-PROFICIENT platform a consistent runtime environment abstracting the concrete setup of the target deployment machine. More precisely, those detailed requirements of the target deployment machine become visible in this approach, instead of being buried in many distinct configuration steps.

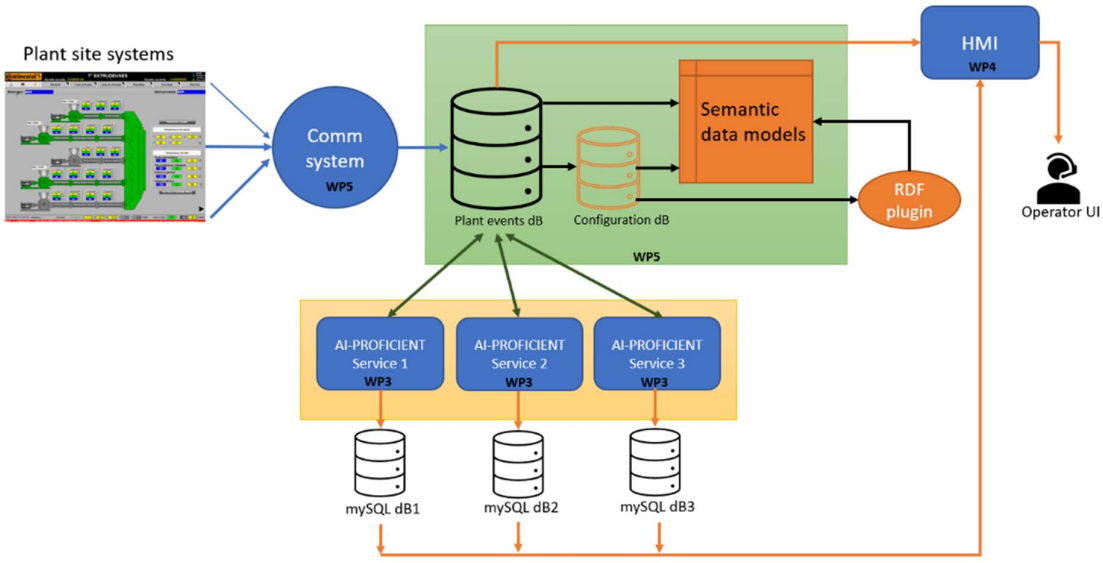

Figure 4: AI-PROFICIENT data layer representation

It has been decided that each AI PROCIFIENT service is responsible for defining and managing its own relational database containing the results it needs to share with other AI services (see Figure 4). This follows the concept of microservices (e.g. mysql-service-FCIR-TF), treating the database of each service itself as a separate microservice. Each microservice has its own dedicated database instance, which encapsulates its data and provides specific database-related functionalities which promotes isolated data management, scalability and performance, data base specific optimization and deployment flexibility.

This isolation of docker containers ensures that each database-as-a-microservice operates independently and does not interfere with other containers. This approach provides better security by limiting the potential attach surface as well as versioning of container images. It is therefore easier to manage different versions of database instances, which can be useful when performing database migrations or rolling back to previous version. Since each microservice has its own docker file, the configuration of each docker container is defined in the yml file (i.e. docker-compose.yml). The configuration of the docker containers includes the name of the docker image, container name, port number, credentials to access the relational database and volumes where the data from each service is stored.

2.1.2.1 Semantic data model

When considering a data processing system, the key to understand the potential of the system is to understand the data that is being shared. The semantic data model provides that perspective. It gives meaning to the data that flows within such system. The semantic data model provides a system agnostic view on the data. For instance, the timeseries databases provide information about the events that happen in the plant. The role of the semantic data model is to describe in a system agnostic way these events (e.g. temperature readings by a 1-degree sensitive sensor located at the entrance of the processing machine).

Unique for the semantic approach is to express this information in a human and machine processable format. In this way the actual data from the timeseries can be connected with a human readable representation in a transparent way.

Within AI-PROFICIENT, this approach has been used to document the Continental use case. The deployment of the semantic data model involved several steps to ensure efficient utilization and accessibility of the model. Firstly, the model was expressed in triples using the Resource Description Framework (RDF) and the XML Schema Definition (xsd) vocabularies. This approach allowed for the

representation of data and its relationships in a standardized and interoperable manner. The modeling process followed strict guidelines to ensure the accuracy and integrity of the data model, resulting in the creation of Turtle files (ttl) as artifacts.

To deploy the semantic data model, a Docker environment was set up along with a Docker Compose configuration. Docker provides a containerization platform that allows for easy deployment and management of applications and services. Docker Compose, on the other hand, simplifies the orchestration of multiple containers. The deployment utilized a Virtuoso database, which is a powerful and scalable RDF triple store that supports SPARQL query language. The Virtuoso graph upload feature was employed to load the Turtle files into the database, thereby creating the necessary RDF graph representation of the semantic data model. This deployment setup ensured efficient storage, retrieval, and querying of the data, enabling seamless utilization of the semantic data model in various applications and scenarios.

2.1.2.2 Deployment of timeseries database

In the AI-PROFICIENT project, a significant volume of measurements is expected to be collected from the pilot sites. These measurements can be represented as time series data, including the measured value, timestamp, and metadata such as sensor identifiers. To effectively store and manage this type of data, a timeseries database is necessary. Among the various options available in the market, the consortium members have chosen to utilize InfluxDB based on their positive prior experience.

InfluxDB is specifically designed to handle high write and query loads, making it well-suited for scenarios where large-scale plants generate real-time data from hundreds or thousands of digital and analog signals. It enables efficient querying of the collected data, which is crucial for performing data analytics and extracting valuable insights. In addition to its legacy InfluxQL querying language, InfluxDB has also introduced Flux since version 2.x. Flux is a powerful, fourth-generation programming language optimized for data scripting, monitoring, and alerting. It offers the ability to structure queries by separating common logic into functions and libraries, enabling easier code sharing and faster development. Furthermore, Flux supports integration with other SQL data stores, including Postgres, Microsoft SQL Server, SQLite, and SAP Hana. Additionally, it can be used to combine time series data with cloud-based data stores like Google Bigtable, Amazon Athena, and Snowflake. This capability allows for enriching time series data with additional information from various sources, providing valuable context and enabling deeper insights into the collected data.

InfluxDB offers more than simple querying capabilities; it also provides various functions for data manipulation, such as aggregation, integration, sum, and mean. These functions can be accessed through the same API on a specific port.

To understand the schema of InfluxDB, several concepts need to be grasped. First, a database serves as a logical container for different time-series data, users, retention policies, and continuous queries. Each database holds one or more measurements, which represent the actual data stored in associated fields. Tags are used to store metadata and consist of key-value pairs. Tag keys are indexed, enabling faster queries when performed on tag keys. On the other hand, series represent collections of data that share a common measurement, tag set, and retention policy.

Field keys and values store both metadata and actual measurements. These fields can be of various types, including integer, float, string, and boolean. Unlike tags, fields are not indexed, and each field value must be associated with a timestamp. Querying field values requires searching through all points that match the specified time range, which may affect performance. Retention policies determine how long data will be retained in the database. By default, the retention policy is set to infinite, meaning that no data is automatically removed unless specified otherwise.

Understanding these concepts is essential for effectively utilizing InfluxDB and designing an appropriate schema for organizing time-series data. It allows for efficient querying, indexing, and managing data retention policies, enabling optimal performance and data storage within InfluxDB.

Overall, the selection of InfluxDB as the timeseries database in the AI-PROFICIENT project is based on its ability to handle high write and query loads, its support for Flux as a powerful querying language,

and its integration capabilities with other SQL and cloud-based data stores. These features ensure efficient data management, enable advanced analytics, and facilitate the extraction of valuable insights from the collected time series data. In Continental pilot plant, InfluxDB is used to store the measurements obtained from OPC-UA server that has already been installed in the plant.

2.2 Data privacy, protection and security measures

The AI-PROFICIENT platform places significant emphasis on data protection and security throughout its development and implementation. These measures aim to ensure the confidentiality, integrity, and availability of data collected, stored, and exchanged within the platform.

- **Access Control:** Strict control over data access is crucial to prevent unauthorized access and protect sensitive information. The platform employs robust user authentication and authorization mechanisms. This includes the use of strong passwords, access control lists (ACLs) to limit access to authorized individuals or applications.
- **Encryption:** Encryption plays a vital role in safeguarding data both at rest and in transit. By transforming information into an unreadable format, encryption ensures that even if data is compromised, it remains inaccessible without the decryption key. The AI-PROFICIENT platform employs strong encryption algorithms, such as the Advanced Encryption Standard (AES), to protect sensitive data.
- **Data Backup:** Regular backups of data are performed and stored in secure locations. This practice mitigates the risk of data loss due to system failures, natural disasters, or cyberattacks. By maintaining up-to-date backups, the platform ensures data availability and enables efficient recovery in the event of a data breach or system failure.
- **Data Retention Policies:** Establishing data retention policies is essential for managing data effectively. These policies define how long data should be stored and when it should be securely deleted. By adhering to data retention policies, the platform minimizes the risk of unauthorized access or disclosure of data that is no longer required.
- **Physical Security:** Physical security measures play a crucial role in protecting the facilities where data storage takes place. Access controls, such as biometric authentication or key card systems, restrict entry to authorized personnel only. Robust physical security measures ensure that data storage facilities are adequately protected from physical breaches.

Overall, implementing these data protection and security measures within the AI-PROFICIENT platform ensures the confidentiality, integrity, and availability of data. Regular review and updates of these measures are necessary to address evolving threats and technologies. Continental and INEOS pilots have proactively implemented robust data security and protection standards within their factories to safeguard their valuable information assets.

2.2.1 Continental pilot security implementation

Continental network is accessible via VPN connection for the allowed users. Since this document is public, no further information can be provided here.

2.2.2 INEOS pilot security implementation

The INEOS-2 use case consists of an application which functions on a mobile device within the plant. The mobile device is stored securely within the plant and is authentication locked. The device is connected to the network on the plant floor, which is able to reach the internet. The device connects to the TenForce backend component and communicates via HTTPS.

INEOS's local network blocks all public internet traffic.

The communication from an internal network in INEOS with the DCS and the TenForce backend has to be routed through the internet. To enable that connection and securely communicate with each other over the internet, Azure Service Bus is used. Azure Service Bus provides a secure channel to communicate via HTTPS, and to sync the databases with each other. The flow of the Azure Service Bus is illustrated in the diagram below, shown in Figure 5. INEOS' DCS and TenForce's backend can push updates and receive updates from each other through Azure service bus's subscription service. The TenForce backend is behind a firewall set to disallow any other traffic from outside the network.

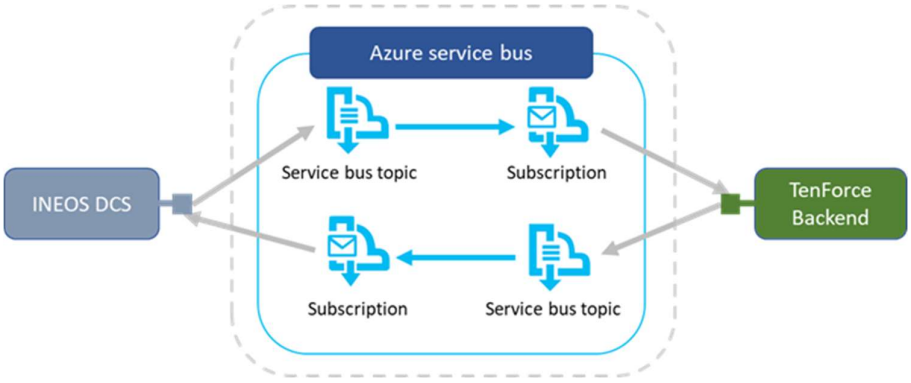

Figure 5: INEOS DCS and TenForce backend connection

For data security, INEOS' and TenForce's databases are periodically backed up to provide redundancy of the data.

2.3 Integration with AI4EU platform, data sources and services

2.3.1 AI4EU integration plan

AI-PROFICIENT's actions in Task 5.4 were dedicated to align development activities with the results of AI4EU project funded under the ICT-26-2018-2020 Call. This includes building upon the results of the AI on-demand platform developed in the context of AI4EU. This aims both at alignment of development activities and at exploitation of the AI4EU's results as part of the AI-PROFICIENT solution. Since AI4EU is aimed to deliver the AI on-demand platform to serve as a one-stop-shop, this task seeks for means to integrate its building blocks and key AI enablers. As the initial point of actions is defined, exploitation of the datasets for early testing of AI services, as well at the fast prototyping studio, serving as a prototyping environment, for the definition of the workflows, algorithms and data formats had to be considered.

It was defined at the beginning of development activities, that AI services and tools relevant to AI-PROFICIENT's scope and objectives, which are already hosted by the AI on-demand platform need to be thoroughly analysed and taken into account for exploitation and integration into the AI-PROFICIENT platform. Following this decision, technical partners of the consortium were instructed to explore the AI4EU platform for already existing software modules and datasets relevant to the services they had undertaken to develop in the project. The next step is to carefully examine the capabilities offered by the existing modules and assess whether it is possible to extend the model/algorithm functionalities, build on it with further code development, thus achieve code reusability and exploit the components in AI4EU platform to this new instance. It is also foreseen, that activities of this task will try to build upon the results and lessons learnt from AI4EU's pilot experimentations, in particular the AI4Robotics (Intelligent Performance Analytics for Industrial Robots) and AI4Industry (AI-Driven Digital Companion for Production Facility) case studies, always provided that these case studies are relevant to software components needed in the context of AI-PROFICIENT.

Finally, at the beginning of task activities, an internal agreement among AI-PROFICIENT technical partners was done to try for further integration of the project modules with AI4EU, by onboarding modules developed in terms of the project to AI4EU platform if this does not violate internal corporate rules about proprietary software of the partner(s) who developed the software component(s).

2.3.2 Actual implementation steps

In the course of implementation actions, AI-PROFICIENT partners performed both steps of trying to integrate modules already uploaded in AI4EU platform and trying to integrate the results of AI-PROFICIENT development to the platform.

An initial effort was done to allocate modules appropriate for integration in AI-PROFICIENT. After clarifying the Use Cases to be implemented by the project and consequently concluding end-user needs derived from the Use Cases, all technical partners engaged in software development performed an outline of what they need to develop as part of their software implementation. According to this definition of needs and priorities, the partners explored the marketplace of AI4EU platform to find existing modules with similar models, algorithms or other end user functionalities as the ones that they have decided to include in their software components. During this effort several concerns arose regarding the possibility to select the appropriate components from AI4EU, which are thoroughly explained in Deliverable *D5.4-* "*Integration with AI4EU's AI on-demand platform*".

As a short overview, we can allocate three main reasons for not being able to not fully integrate AI4EU's existing modules in the AI-PROFICIENT:

a. AI-PROFICIENT requires for software modules based on models which need to be developed upon and trained with data specific to the Use Cases they were developed for.
b. Tools already uploaded in the AI4EU platform either with a structure too specific for the Use Case they were developed for, either too generic – in both cases not fitting to AI-PROFICIENT development needs, derived from very specific end users (factories) requirements.

c. Several tools already existing in AI4EU platform, whilst seemingly appropriate to integrate in AI-PROFICIENT, upon examination were proven to be either completely non-functional, either not operating to a satisfactory level.

Due to these reasons, the technical partners were confronted either with the issue that modules are not available for their needs (case a above), either with the issue that existing modules are totally not appropriate for their needs (case b, above), either with the fact that modules existing in AI4EU platform are not operation (case c, above).

Finally, regarding the AI4Robotics (Intelligent Performance Analytics for Industrial Robots) and AI4Industry (AI-Driven Digital Companion for Production Facility) case studies, these were examined by the partners, but since the content of the case studies proved to be not relevant to AI-PROFICIENT scope and objectives, no further action was taken to integrate software tools originating from these case studies.

Resulting from the above, unfortunately, no modules from AI4EU platform have been integrated in the AI4EU platform.

Regarding the objective set internally by AI-PROFICIENT partners to integrate tools in AI4EU platform this resulted to onboard two tools in AI4EU platform, as thoroughly described in performed, as explained in Deliverable *D5.4*-"*Integration with AI4EU's AI on-demand platform*".

2.4 AI service deployment

In the AI-PROFICIENT project, the general approach to service deployment involves utilizing Docker containers to deploy services. Docker provides a lightweight and portable platform for packaging applications, along with their dependencies, into containers. By using Docker containers, services can be deployed consistently across different environments, ensuring reproducibility and ease of deployment. The services in the AI-PROFICIENT project interact with various components and data sources. They retrieve data from the semantic repository and timeseries measurements from the InfluxDB database. The semantic repository serves as a centralized store of structured metadata and provides a means for organizing and querying the data effectively. InfluxDB, being a high-performance timeseries database, is well-suited for storing and querying large amounts of timeseries data efficiently.

Once the services retrieve the necessary data, they process them by using AI algorithms developed in other related work packages. The results obtained from these operations are then stored in a MySQL database. The stored results in the MySQL database can be accessed and visualized through a user interface (UI). The UI acts as a dashboard or graphical interface that allows users to interact with and explore the data visually. This enables users to gain insights, monitor system performance, and make informed decisions based on the analysed data. In addition, it allows them to provide feedback which is later used as an input to AI services.

By adopting the Docker containerization approach and utilizing data from the semantic repository, InfluxDB, and MySQL, the AI-PROFICIENT project ensures a modular and scalable deployment of services. This architecture allows for flexibility in managing and scaling the services as needed, while the integration of various data sources enables comprehensive data analysis and visualization through the UI.

2.4.1 CONTI-2 and CONTI-5 service deployment

Even if these two Use Cases tackle completely different problems (CONTI-2 the set-up of an extrusion and CONTI-5 the monitoring of a cutting blade) in the course of the project they almost converged at service level. In this way there are three services that can be considered generic in both UCs:

- **Feature extraction service/module**: This module takes the raw measurements and creates structured data by means of data integration and signal processing. The indicators of features computed can be either used for inference or they can be stored for later retraining of the models.
- **Model building service:** This service takes the features that have been stored and retrains the model so that it can benefit from the latest data records improving its accuracy.
- **Model serving service:** This service is in charge of taking the model that has been trained, taking the features extracted from the new observations and executing an inference of the model. The outputs of this service are stored in standardized databases so that they can be displayed to the final users.

As an example, in Figure 6 and Figure 7, we depict the services for CONTI-5.

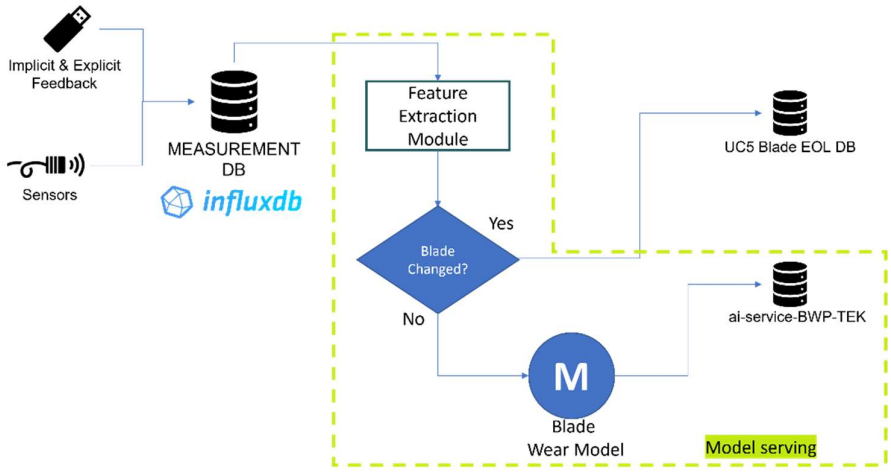

Figure 6: CONTI-5 Model serving including feature extraction.

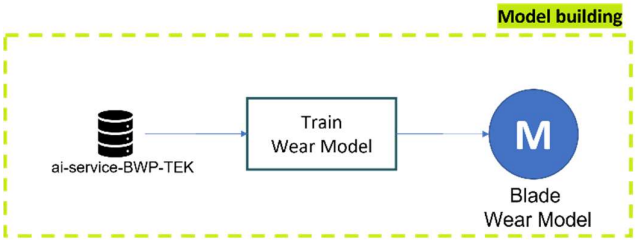

Figure 7: CONTI-5 Model building service

2.4.2 CONTI-3 service deployment

For CONTI-3 use case, the first step involves the offline training of the model utilizing data fetched from InfluxDB (see Figure 8). Post the completion of the training phase, the resultant model is incorporated into aip-service-NRPI container, where the service is deployed for usage.

The service deployed in the aip-service-NRPI container is composed of three distinct modules (see Figure 9):

- **Pre-processing Module:** This module retrieves the data newly stored, verifies whether the operations are within production phase and, following this, checks the existence of a derivative in production.
- **Prediction Model Module:** Post verification, the model trained offline is applied to predict the existence of a deviation.
- **Interpretation Module:** When deviations are accurately identified by the model, this module is activated. Here, we engage in an interpretation of the model, pinpointing the three primary features contributing substantially to the observed deviation.

The extracted information regarding these principal features is subsequently stored in a separate Docker container - aip-my-sql-NRPI. The stored data is thereby made accessible for presentation to the operator.

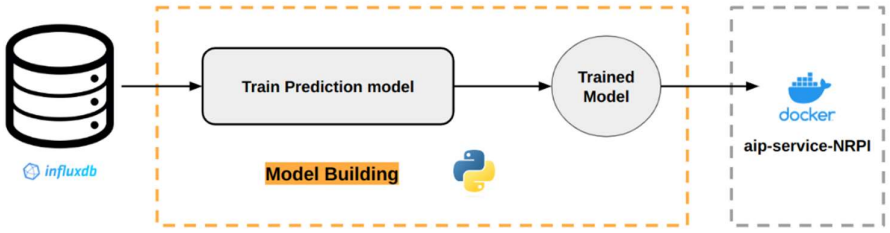

Figure 8: Training pipeline of CONTI-3 model

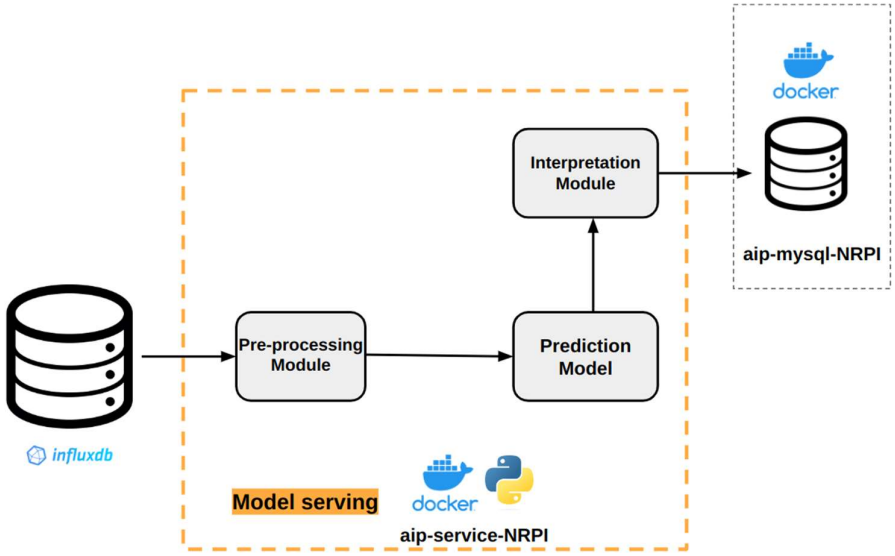

Figure 9: Usage pipeline of CONTI-3 model

2.4.3 CONTI-7 service deployment

In this use case, we use machine vision to detect the location of tire treads on the feeder belts of a tread packing mechanism. We try to construct a predictive model based on the position trend to indicate wear per stage zone in the form of numerical or (possibly) qualitative wear indicators. The result of executing the model on current data is output as wear indicators and maintenance suggestions. A secondary system is using more classical, tolerance-based approach for raising immediate alarms. In Figure 10, Figure 11, and Figure 12, we present the proposed pipelines for this use case.

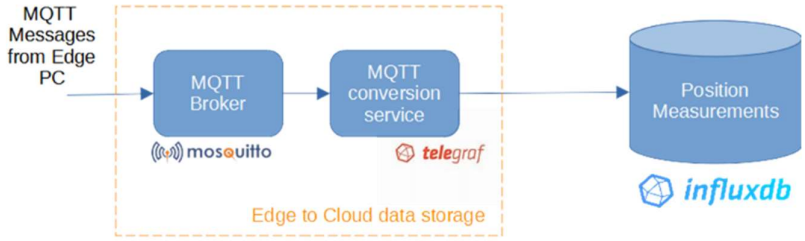

Figure 10: Data transfer pipeline of CONTI-7 model

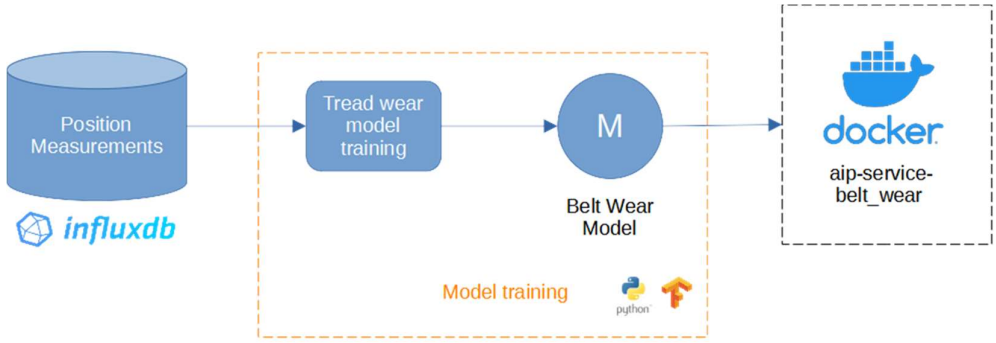

Figure 11: Training pipeline of CONTI-7 model

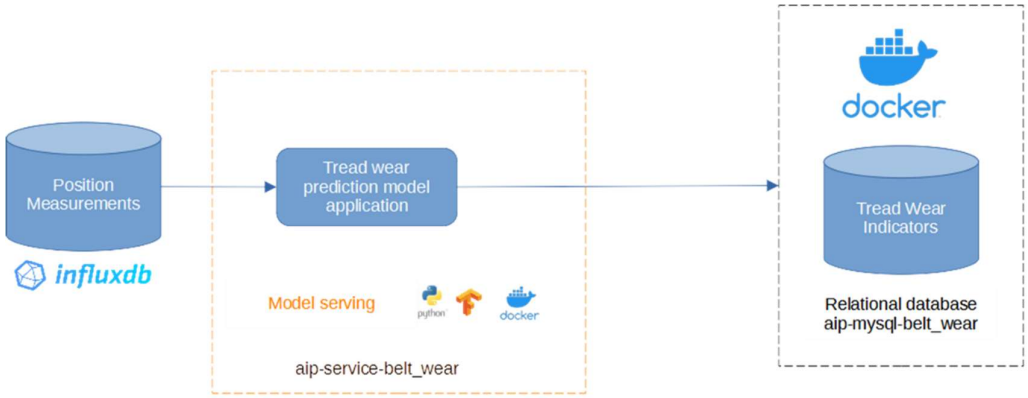

Figure 12: Usage pipeline of CONTI-7 model

2.4.4 CONTI-10 service deployment

This use case aims to automate the process of investigating the causes of the final product (tread) quality deviations. Such that, it will bring the dedicated Quality analysis tool, consisted of several sub-services, that are described in the following, with the special focus on the deployment and integration details.

2.4.4.1 GHO service integration

The Generative Holistic Optimization (GHO) service has been designed and developed to assist the operator by suggesting readjustments of control parameters and to, indirectly, address the issues in production line and prevent the production of out-of-range tire thread products. The optimizer relies on the MOEA algorithm as its core, generating and evaluating potential solutions iteratively based on objective functions derived from predicted values obtained through Surrogate Data-Driven Models (SDDM service). To ensure timely execution, the optimizer incorporates a time-series clustering module, facilitating stable and efficient convergence. Upon achieving convergence, the service presents the operator with a sub-optimal set of control parameters. The final versions of the GHO pipeline, has been depicted in Figure 13.

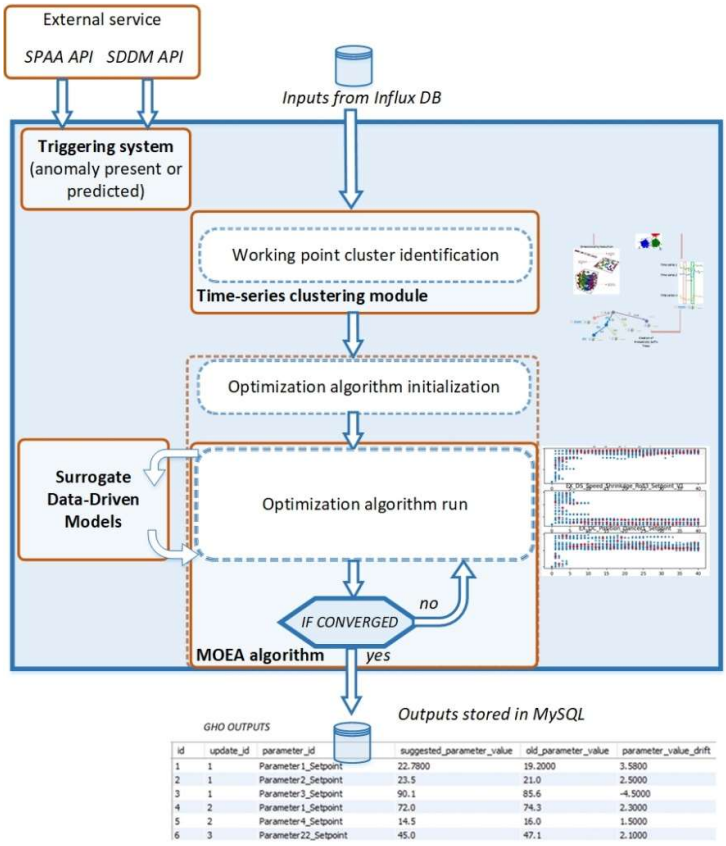

Figure 13: CONTI-10 GHO optimizer pipeline

The CONTI-10 optimizer service is developed using Python and deployed using Docker technology. Docker images are created for the optimizer and the MySQL output database. The service is connected to both the MySQL database and the Influx database, which gathers input data from the production plant. The GHO operates exclusively during the nominal regime of production, on demand, being triggered by the inner triggering system, or by two other external services, which role play Short-term Post-hoc Anomaly Analysis (SPAA) and Surrogate Data-Driven Models (SDDM) services, encapsulated together, in the Quality analysis tool.

Initially, a Natural Voice interaction module (NVI) was developed as the optimizer interface, allowing the operator to provide feedback through voice commands, which could not be utilized due to the noisy production environment. Instead, the final interface for the operator is the dashboard, serving as a visualization platform for all deployed services in the Continental pilot. The GHO frame can be accessed in the *Notification screen* of the dashboard, along with other CONTI-10 Quality analysis tool services. The dashboard provides notifications with recommended adjustments for process parameters and their values. The operator can choose to accept or reject recommendations for specific parameters or the

overall output. It's important to note that this interaction is optional, and feedback can be derived implicitly.

2.4.4.2 SDDM service integration

The Surrogate Explainable Data-Driven Model (SDDM) serves as an early anomaly detection module, an explainability and transparency service, and a process digital twin. It continuously forecasts product characteristics based on current process parameters, enabling proactive measures against degradation. Upon foreseeing degradation, the GHO service adjusts process conditions, while SDDM identifies probable causes and guides optimization of crucial parameters. Throughout the optimization process, SDDM evaluates recommendations, facilitating the selection of optimal solutions. SDDM was developed in Python, utilizing ML libraries like Keras, TensorFlow, scikit-learn, Lime, and SHAP, and deployed via Docker on a virtual machine at Continental. Integration with the AI-PROFICIENT data platform facilitates real-time parameter retrieval from InfluxDB and storage of outputs in MySQL DB. The MySQL DB includes tables for estimating product characteristics and capturing influential inputs. It serves to validate service performance, integrate with the HMI, and allows operators to monitor estimated characteristics and their color-coded states. Fast information exchange for GHO integration is achieved through an API.

2.4.4.3 PEAA service integration

Besides the base instructions and integration basis for the common services of AI PROFICIENT services, the three subservices have some limitation in execution and maximum capacity requirements must be considered before deploying on the target structure.

The three services basically have a demanding main function, which is processing the analysis and producing the reports. After obtaining the results, the reports and variables are stored in the results_db which can be shared afterwards.

None of the three services is interactive (they run a background process on the server or a detached service). However, while DQR service takes a few minutes to execute, the FS_XAI service and the AD service can take over 15 minutes with a highly demanding resource from the host machine.

The strategy to avoid bottle necks and service locks, is to create another service that will embrace the three services. The main service (aip-service-peaa) will have the main common functions for the other three services (they have their own container with ports and volumes). The three services are served in API mode and using aip-service-peaa also as a reverse proxy for each endpoint. Figure 14 is how they would appear when images are built and deployed. Figure 15 shows the architecture of the service.

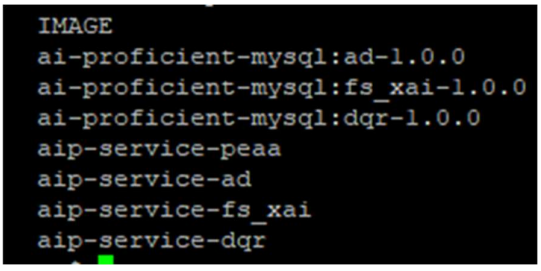

Figure 14: PEAA container images deployed.

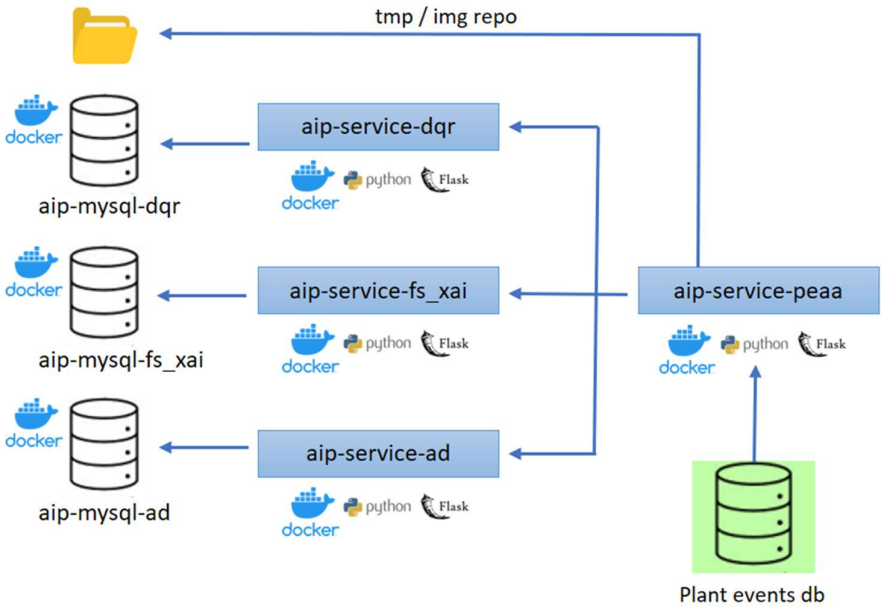

Figure 15: PEAA service integration.

2.4.4.4 NVI service integration

In the case of the NVI service integration, it was given a complex challenge with two main approaches: Voice analysis services have quite demanding hardware requirements. An analysis of the CONTI services target infrastructure was considered in this process. In the case of the voice interaction module, the security and operative CONTI requirements with the current interface were not the most feasible for the operators to use in the factory. In fact, it was suggested that the reinforcement learning process could be better performed without the intervention of operators. That is, the operator will receive the instructions of the optimizer. It will adjust the parameters in the machines, and to carry out the reinforcement learning process, the interface will have to read the signals from the machines and compare them with the output of the generative optimizer. So, it would be known whether the operator will have accepted the answer. However, due to some limitations, this was not feasible to have access to the signals on the machines once the parameters were adjusted. Secondly, this solution did not fit the requirements of tasks 4.1 and 3.5, which are related to reinforcement learning for CONTI-10.

NVI service needed a demanding core infrastructure (in terms of RAM available on the target machine) and dedicated hardware. Instead of using a dedicated container to do the voice processing there was the possibility of using the Azure service on the Cloud. However, this solution stands for security issues and depends on network availability. The preliminary tests on the other hand, were executed successfully on the deployment of the preliminary NVI module. Figure 16 shows the architecture of this service.

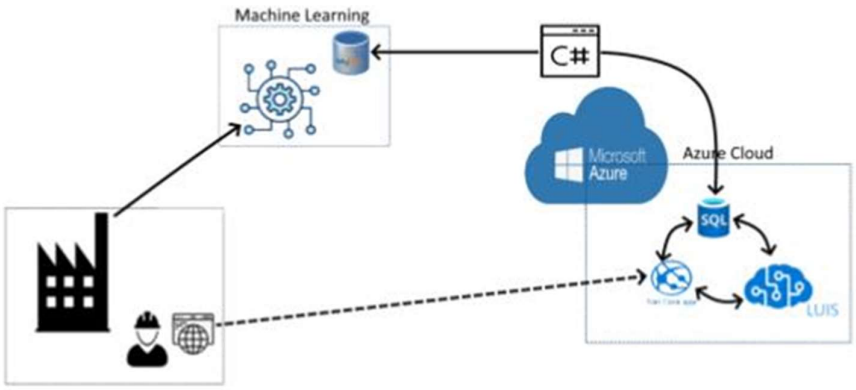

Figure 16: PEAA service integration.

2.4.5 INEOS-1 service deployment

Due to limited number of occurrences and the wide variety of conditions where process instabilities occur, the solution for INEOS-1 use case is based on a digital twin that is constructed through a hybrid approach, as shown in Figure 17. The approach combines data analysis with physical modelling carried out on three different levels (1D process model, 3D coarse grained model of the reactor and a fine-grained model for reactor sectors). The core of the digital twin is the 1D process model which was planned also to serve as the basis for Generative holistic optimization. Due to the long computation time, the digital twin (DT) proved to be too slow for optimization purpose.

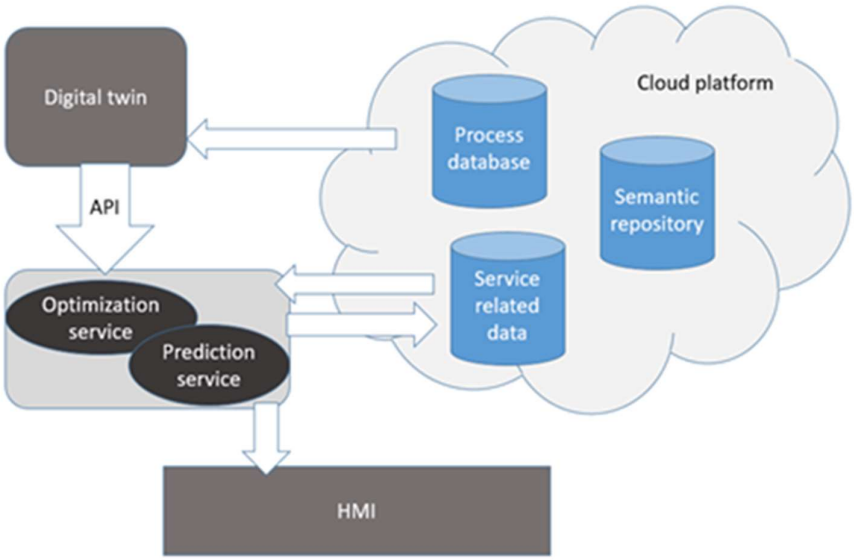

Figure 17: Interaction between services and the platform in the cloud as initially planned for INEOS-1 UC

Due to the long computation time of the digital twin, optimization service cannot be based on the DT, and it is omitted from the solution. The digital twin service is built on the following:

- Both the input and the output variables of the model are tagged with unique key values (id).
- Model inputs are the available measured time-dependent plant data (timestamp, variable id, variable value).

- Model output contain the full state of the process.
- The state of the digital twin model can be used to initialize a separate prediction calculation. This calculation can be initiated and controlled independently (from the digital twin model) through a GUI by an end-user for testing with different time-dependent scenarios.
- A separate interface is made to initialize, connect and run the digital twin model. This interface is responsible for passing measured plant data to the digital twin model, and for passing the model outputs to the end-user and to any data logging system.
- Programming language in the digital twin is Fortran while the interface is written in Python.

Due to the complexity of the process, a long offline testing period of the digital twin is required before moving to online testing and thus fully operational online solution and deployment can't be accomplished in the project. Instead, the DT and its interface are tested offline to prove the concept.

2.4.6 INEOS-2 service deployment

In this use case optical character recognition (OCR) is mainly used to convert text from images to useful information for an operator. The goal is to be able to decode images and display only necessary data to an operator so he can verify that information and communicate this information with the systems at INEOS.

The flow of this use case is the following: the operator takes a picture and passes these images to the OCR via an application which runs on a mobile device. The OCR service converts all text on an image to text. This text is then run through a matching AI service which extracts the necessary information and matches the OCR's results with a dictionary of all the possible outcomes. This result is shown to the operator for confirmation. After confirmation from the operator, the information is sent to a control system within INEOS, then the operator gets notified of the control system's output.

The control system notifies the operator if the information was rejected or accepted.

The operator is notified by the application of an uncertain recognition if any of the following occurs:

- If the OCR's extracted information of the OCR's result does not exist in the dictionary
- If the extracting and matching AI service is uncertain of its results
- If the OCR recognition is below 100% certainty of its result

Speech-to-text (STT) AI services are also used for a hands-free control of the application. A combination of *WearHF Speech Recognition* and *OpenAI's Whisper model* is used to control the mobile device.

2.4.7 INEOS-3 service deployment

Within INEOS Cologne pilot site, one use case was envisioned. This particular use case envisioned to develop recommendation system which would provide suggestions for process parameter adjustments with the aim of improving certain quality parameters, in particular, rheological characteristics. From the beginning of the project, it was clear that this use case was quite challenging, since establishing correlation between the process parameters and targeting characteristics was not an easy task, due to the rare rheological characteristics sampling.

Hence, extensive three rounds of data analysis were carried out to validate this particular use case and decide how to proceed within the project. Details from this analysis could be found within D3.4. As a result of this analysis, process parameters that have the biggest influence on the targeting characteristics have been selected. Since this analysis required extensive time effort, INEOS plant concluded that it is not feasible to develop, test and deploy fully operational solution for process control until the end of the project and it was decided to stop this use case. In other word, the results of this particular use case are conclusions from the extensive offline data analysis, rather than the online working services. Therefore, services have not been and will not be developed and deployed within INEOS Cologne plant. Instead, several "theoretical" results (in terms of modelling) are available allowing to develop scientific paper as requested by a RIA project.

3 Conclusion

In conclusion, this document provides a comprehensive overview of the activities related to the system integration and deployment of the AI-PROFICIENT platform. It highlights the main achievements within Work Package 5 and emphasizes their crucial role in ensuring effective system integration and deployment.

Robust communication protocols and interfaces have been implemented, allowing the AI-PROFICIENT platform to seamlessly interact with various equipment and systems. Efficient data storage and data modelling have been achieved through the development of a semantic repository and databases. This structured and standardized data management solution facilitates seamless integration and access to information, thereby enhancing the overall system integration and deployment processes.

The implementation of comprehensive measures for data protection and security has been a primary concern. Through the incorporation of encryption, access controls, and backup mechanisms, the AI-PROFICIENT platform effectively safeguards sensitive information during integration and deployment. The successful development of an interface with AI4EU data sources and services has provided seamless connectivity and access to advanced analytics and AI capabilities. This integration enhances the overall functionality and decision-making capabilities of the platform. Finally, the successful deployment of the AI-PROFICIENT platform has enabled the integration of AI services from other work packages into the pilot sites.

4 Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957391.