D6.18 – Continuous Integration and Validation v3

WP6 – EVALUATE: Piloting and Demonstrating

i4Q has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 958205.

Document Information

GRANT AGREEMENT NUMBER	958205	ACRONYM	i4Q
FULL TITLE	Industrial Data Services for Quality Control in Smart Manufacturing		
START DATE	01-01-2021	DURATION	41 months
PROJECT URL	https://www.i4q-project.eu/		
DELIVERABLE	D6.18 – Continuous Integration and Validation v3		
WORK PACKAGE	WP6 – EVALUATE: Piloting and Demonstrating		
DATE OF DELIVERY	CONTRACTUAL 30-Apr-2024	ACTUAL	30-Apr-2024
NATURE	Other	DISSEMINATION LEVEL	Public
LEAD BENEFICIARY	ITI		
RESPONSIBLE AUTHOR	ITI		
CONTRIBUTIONS FROM	8-BIBA, 20-BIESSE, 1-CERTH, 2-ENG, 6-EXOS, 21-FACTOR, 23-FARPLAS, 24-FIDIA, 3-IBM ISRAEL, 7-IKERLAN, 5-KNOWLEDGEBIZ, 22-RIA STONE, 10-TUB, 11-UNINOVA, 9-UPV, 25-WHI		
TARGET AUDIENCE	1) i4Q Project partners; 2) industrial community; 3) other H2020 funded projects; 4) scientific community; 5) EC Services		
DELIVERABLE CONTEXT/ DEPENDENCIES	This document is the third iteration of D6.9 due by Dec 2022 and and D6.17 due by Jun 2023. Its relationship to other documents is as follows: - D6.9 Continuous Integration and Validation. - D6.17 Continuous Integration and Validation v2. - Deliverables from the Build Work Packages: WP3, WP4 and WP5. - Deliverables from the Evaluate Work Package: WP6. - D9.4 Ops Setup and Quality Control Report v1.		
EXTERNAL ANNEXES/ SUPPORTING DOCUMENTS	None		
READING NOTES	None		

ABSTRACT	This document is the last release of a set of three, of the Continuous Integration and Validation report. It includes an analysis of the integration of the different i4Q Solutions that are part of the pipeline of each one of the i4Q Pilots and a list of the problems that appeared during the integration and the corrective measures applied.

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
0.1	04-Sep-2023	ToC	Table of Contents	ITI
0.2-0.4	17-Apr-2024	Draft	1st Draft submitted for internal review	BIBA, BIESSE, CERTH, ENG, EXOS, FACTOR, FARPLAS, FIDIA, IBM ISRAEL, IKERLAN, KNOWLEDGEBIZ, RIA STONE, TUB, UNINOVA, UPV, WHI, ITI
0.5	18-Apr-2024	Review	Conduction of internal review	IKER, CERTH
0.6	29-Apr-2024	Draft	Final Draft available for quality check	ITI
1.0	30-Apr-2024	Final Doc	Quality check and issue of final document	CERTH

Disclaimer

Copyright message

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS/ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
CA	Certificate Authority
CNC	Computer Numerical Control
CSV	Comma-Separated Values
DB	Database
DR	Data Repository
EOL	End of Life
FMU	Functional Mock-up Units
GUI	Graphical User Interface
HSM	Hardware Security Module
i4Q^{AD}	Analytics Dashboard
i4Q^{AI}	AI Models Distribution to the Edge
i4Q^{BC}	Blockchain Traceability of Data
i4Q^{BDA}	Big Data Analytics Suite
i4Q^{DA}	Services for Data Analytics
i4Q^{DIT}	Data Integration and Transformation Services
i4Q^{DR}	Data Repository
i4Q^{DT}	Digital Twin Simulation Services
i4Q^{EW}	Edge Workloads Placement and Deployment
i4Q^{IM}	Infrastructure Monitoring
i4Q^{LRT}	Manufacturing Line Reconfiguration Toolkit
i4Q^{PA}	Prescriptive Analysis Tools
i4Q^{PQ}	Data-Driven Continuous Process Qualification
i4Q^{QD}	Rapid Quality Diagnosis
i4Q^{QE}	QualiExplore for Data Quality Factor Knowledge
i4Q^{SH}	IIoT Security Handler
i4Q^{TN}	Trusted Networks with Wireless and Wired Industrial Interfaces
IIoT	Industrial Internet of Things
IP	Internet Protocol
IWSN	Industrial Wireless Sensor Network
JSON	JavaScript Object Notation

KPI	Key Performance Indicator
LSTM	Long Short-Term Memory
MB	Message Broker
ML	Machine Learning
REST	Representational State Transfer
RIDS	Reliable Industrial Data Services
RNN	Recurrent Neural Network
SDN	Software Defined Networking
UI	User Interface
WP	Work Package

Executive summary

i4Q Project aims to provide a complete set of solutions consisting of IoT-based Reliable Industrial Data Services (RIDS), the so called 22 i4Q Solutions (17 software tools and 5 guidelines), able to manage the huge amount of industrial data coming from cheap cost-effective, smart, and small size interconnected factory devices for supporting manufacturing online monitoring and control.

Besides, different pipelines including some of the i4Q Solutions will be used to improve the current industrial processes of the following pilot scenarios:

- Pilot 1: Smart Quality in CNC Machining.
- Pilot 2: Diagnostics and IoT Services.
- Pilot 3: White Goods Product Quality.
- Pilot 4: Aeronautics and Aerospace Metal Parts Quality.
- Pilot 5: Advanced In-line Inspection for incoming Prime Matter Quality Control.
- Pilot 6: Automatic Advanced Inspection of Automotive Plastic Parts.

In this project, the development work is aimed at producing high-quality code and enhancing the integration of all the i4Q Solutions in the Pilots' pipelines.

The current deliverable provides the final set of steps followed for the deployment and integration of each one of the i4Q Solutions in the Pilots.

D6.18 is including:

- An analysis of the integration of the different i4Q Solutions that are part of the pipeline of each one of the i4Q Pilots.
- A list of the problems that appeared during the integration and the corrective measures applied.

Document structure

Section 1: Introduction. This section describes the purpose and the approach of the Integration and Validation procedures in the i4Q Project.

Section 2: Solutions Integration. This section analyses the integration of the different i4Q Solutions that are part of the pipeline of each one of the i4Q Pilots.

Section 3: Analysis of Results and Solutions Update. This section shows the problems experienced during the integration and the proposed correction actions.

Section 4: Conclusions. This section summarises the main results of the deliverable.

1. Introduction

The main objective of Task T6.8 is to perform integration and functionality tests to solve possible integration problems between the solutions from the BUILD Work Packages:

- WP3 Manufacturing Data Quality.
- WP4 Manufacturing Data Analytics for Manufacturing Quality Assurance.
- WP5 Rapid Manufacturing Line Qualification and Reconfiguration.

A total of 22 i4Q Solutions have been produced and tested, using 6 industrial pilot cases corresponding to the partners of this project, plus the generic pilot defined in T6.7.

The implementation of this task has made it possible to detect functional and/or integration problems at an early stage. Since this feedback has been provided to the development tasks, those problems have been addressed before the official release of i4Q Solutions. Consequently, this task has reduced the risk of facing those problems after the end of the project, which would make it more expensive to resolve in the future. Therefore, this task has contributed to improve the quality and reliability of the solutions and has made them more suitable to be used in real industrial scenarios.

This deliverable addresses the second phase of the deployment and integration of the i4Q Solutions in the i4Q Pilots' infrastructure.

First, the pipelines of the different i4Q Pilots and the most important modifications they have experienced during the period from M31 to M40 are explained. With this information, the appropriate modifications are made and, afterwards, the integrations between the different i4Q Solutions are analysed, as well as the information they send or receive, and the communication mechanism used.

Finally, this deliverable analyses the problems encountered during the deployment and integration phases of the different i4Q Solutions as well as the actions that have been carried out or that are proposed to be done in the future for correcting these problems or for improving the performance of the corresponding solution.

2. Solutions Integration

As it was already done in Deliverable D6.17 [1], this section analyses the integration of the different i4Q Solutions that are part of the pipeline of each one of the i4Q Pilots. For this purpose, first, the most recent version of the pipeline of each pilot is shown. Then, if any modifications have been made since the previous version, they are explained in a justified manner. Finally, solution providers explain, in a brief and concise manner, the interaction between the different solutions, what information they exchange, the communication mechanism used, as well as any other information that is relevant to consider (topics used by the communication mechanism to exchange information, databases or MinIO buckets created to store the information, artefacts generated by the different solutions, etc.).

2.1 Pilot 1: FIDIA - Smart Quality in CNC Machining

This subsection analyses the pipeline of the *Smart Quality in CNC Machining* pilot, which corresponds to the i4Q Solutions deployed at the facilities of the industrial partner FIDIA and explains in a justified way the modifications made since the last deliverable until the present day.

After verifying it with the pilot and the solution providers, the updated version of the pipeline is presented in Figure 1.

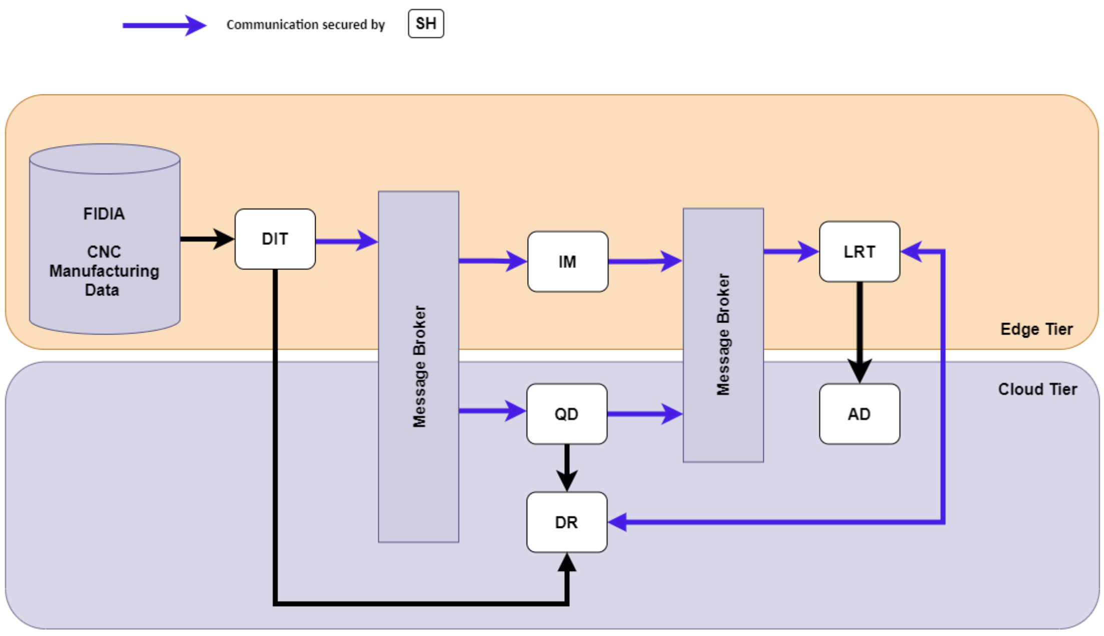

Figure 1. Pilot 1 pipeline diagram

The following modifications have been made with respect to the pipeline defined in Deliverable D6.17 [1].

- The interaction between i4Q^{DIT} and i4Q^{DR} Solutions has been modified. Now, instead of using the Message Broker to send the information from the former to the latter, the i4Q^{DIT} Solution will store the processed information directly in the i4Q^{DR} Solution. To reflect this change, the arrow from the Message Broker to the i4Q^{DR} Solution has been removed and a new one has been added from the i4Q^{DIT} to the i4Q^{DR} Solution.

- The direction of the arrow connecting i4Q^{QD} and i4Q^{DR} Solutions has been changed to indicate that the first solution will store the results produced in the second one and that the i4Q^{DR} Solution will not send any information to the i4Q^{QD} Solution.

2.1.1 i4Q^{DIT} – Data Integration and Transformation Services

2.1.1.1 Inbound integration with factory machines

- **Input solution**. i4Q^{DIT} Solution draws data directly from factory machines.
- **Information received**. The i4Q^{DIT} Solution obtains the measurements produced in real-time by the sensors installed on the FIDIA factory machine. The raw data is cleaned and processed to form one or more datasets to be further analysed.
- **Communication mechanism**. The integration between the solution and the factory machines occurs when the i4Q^{DIT} makes requests to the REST API created by FIDIA, to obtain the data produced by the machine's sensors.

2.1.1.2 Outbound integration with i4Q^{DR}

- **Output solution**. The data created by the i4Q^{DIT} Solution is sent to the i4Q^{DR} Solution.
- **Information sent**. First, i4Q^{DIT} receives the data from the REST API. It then performs a cleaning, pre-processing, and feature extraction process. Finally, the data processed by the solution is stored in a MongoDB instance deployed in the i4Q^{DR} Solution.
- **Communication mechanism**. The integration between these two solutions is done in a direct way, as the i4Q^{DIT} Solution stores the data produced in the MongoDB instance of the i4Q^{DR} Solution. Specifically, the data is stored in a collection with the name *i4q_dit_data*, which is located inside a database with the name *fidia*. In this collection, the following fields are stored: *XMotorCurrent*, *XMotorCurrent_rolling_median*, *YMotorCurrent*, *YMotorCurrent_rolling_median*, *ZMotorCurrent*, *ZMotorCurrent_rolling_median*, *XAbsPosition*, *XAbsPosition_rolling_median*, *YAbsPosition*, *YAbsPosition_rolling_median*, *ZAbsPosition*, *ZAbsPosition_rolling_median*, *XVibration*, *YVibration*, *ZVibration*, *timestamp*, *XVibration_FFT*, *YVibration_FFT*, *ZVibration_FFT*, *XVibration_BW*, *YVibration_BW*, *ZVibration_BW*.

2.1.1.3 Outbound integration with i4Q^{IM}

- **Output solution**. The dataset of sensor signals created by the i4Q^{DIT} Solution is also sent to the i4Q^{IM} Solution.
- **Information sent**. i4Q^{DIT} provides a dataset of preprocessed sensor signals, filtered and enhanced with extracted features.
- **Communication mechanism**. The two solutions communicate through the Message broker.

2.1.1.4 Outbound integration with i4Q^{QD}

- **Output solution**. The dataset of sensor signals created by the i4Q^{DIT} Solution is also sent to the i4Q^{QD} Solution.

- **Information sent**. i4Q^{DIT} provides a dataset of preprocessed sensor signals, filtered and enhanced with extracted features.
- **Communication mechanism**. The two solutions communicate through the Message broker.

2.1.2 i4Q^{SH} – IIoT Security Handler

The i4Q^{SH} Solution is distinguished by its ability to generate Certificate Authorities (CAs) appropriate for each individual pilot, using their specific information. This approach ensures that each pilot has its own unique CA, which adds an additional layer of security and personalisation to the digital certificate generation process. By using individual pilot data, such as names and unique identifiers, a CA is created that is intrinsically linked to each user's identity, increasing the trust and integrity of the security system.

The integration of the i4Q^{SH} Solution with the rest of the solutions in the pilot is different from the integration carried out by the others. Given that the i4Q^{SH} Solution generates a CA for each pilot, it is assumed that it is integrated with another solution when one of the others uses this CA as a mechanism to secure communications. In this case, it can be said that the solution is directly integrated with the i4Q^{DR} Solution and the Message Broker, which use this CA to add a layer of security to communications. It also integrates, in an indirect way, with the solutions that send/receive data to/from the Message Broker and with those that obtain data from the i4Q^{DR} Solution, which are: i4Q^{DIT}, i4Q^{IM}, i4Q^{QD} and i4Q^{LRT}.

- **Input solution**. i4Q^{SH} does not receive information from the other solutions, therefore, this does not apply to this solution.
- **Output solution**. The CA is produced for the use of the solutions deployed in a pilot environment; however, the solution does not send data to any of the other solutions involved in the pilot. Thus, this does not apply to the solution.
- **Communication mechanism**. As the solution does not send or receive information to/from other solutions, there is not defined a way of communication between this, and the other solutions deployed in the pilot.

2.1.3 i4Q^{IM} – Infrastructure Monitoring

In Pilot 1, the i4Q^{IM} Solution is responsible for identifying the degradation of components in CNC machining processes, via the utilisation of Machine Learning algorithms and the exploitation of machine sensor signals related to the spindle position, motor current, etc.

2.1.3.1 Inbound integration with i4Q^{DIT} Solution

- **Input solution**. The i4Q^{IM} is directly connected to the i4Q^{DIT} Solution to receive the sensor signals.
- **Information received**. The solution receives a preprocessed and enhanced dataset from the i4Q^{DIT} comprising all the necessary features for developing and training the component degradation detection ML model.

- **Communication mechanism**. The Message Broker is used as a communication mechanism to consume the data prepared by the i4Q^{DIT} Solution using the topic *DIT_topic_1*.

2.1.3.2 Outbound integration with i4Q^{DR} Solution

- **Output solution**. The i4Q^{IM} Solution stores its analytic results in the *i4q_im_results* collection of the i4Q^{DR} Solution.
- **Information sent**. The prediction results made by the CNC component degradation detection model, along with the associated sensor data, are formatted properly to be stored in the i4Q^{DR} Solution.
- **Communication mechanism**. Communication between the two solutions is established by connecting directly to the MongoDB instance of the i4Q^{DR} Solution.

2.1.4 i4Q^{QD} – Rapid Quality Diagnosis

In Pilot 1, the i4Q^{QD} Solution is responsible for identifying the occurrence of chatter during the operation of the CNC machines, by employing Machine Learning algorithms and exploiting the accelerometer sensor signals to capture machine oscillation information.

2.1.4.1 Inbound integration with i4Q^{DIT} Solution

- **Input solution**. The i4Q^{QD} is connected to the i4Q^{DIT} Solution to receive the sensors data.
- **Information received**. The solution receives a preprocessed and enhanced dataset from the i4Q^{DIT} comprising all the necessary features (accelerometer data in time & frequency domain) for developing and training CNC chatter detection algorithm.
- **Communication mechanism**. The Message Broker is used as a communication mechanism to consume the data prepared by the i4Q^{DIT} Solution using the topic *DIT_topic_1*.

2.1.4.2 Outbound integration with i4Q^{DR} Solution

- **Output solution**. The i4Q^{QD} Solution stores its analytic results in the *i4q_qd_results* collection of the i4Q^{DR} Solution.
- **Information sent**. The prediction results made by the CNC chatter detection model, along with the associated sensor data, are formatted properly to be stored in the i4Q^{DR} Solution.
- **Communication mechanism**. Communication between the two solutions is established by connecting directly to the MongoDB instance of the i4Q^{DR} Solution.

2.1.5 i4Q^{DR} – Data Repository

In Pilot 1, the i4Q^{DR} Solution is integrated with four solutions: with the i4Q^{SH}, as it uses the Certificate Authority (CA) generated by this solution to deploy a secured scenario; with the i4Q^{DIT} and i4Q^{QD} Solutions, from which it receives the processed or produced data; and with the i4Q^{LRT} Solution, with which it has a bidirectional integration, as it sends and also receives data from it.

To make it easier and more understandable to the reader, each integration is separately described in more detail below.

2.1.5.1 Inbound integration with i4Q^DIT Solution

- **Input solution**. i4Q^DR receives data from the i4Q^DIT Solution.
- **Information received**. The i4Q^DIT Solution obtains real-time data from the FIDIA factory machine. Then, it performs a cleaning, preprocessing and feature enrichment process. After this, it stores the data in a MongoDB instance deployed in the i4Q^DR Solution.
- **Communication mechanism**. The integration is done in a direct way, the i4Q^DIT Solution being the one that stores the data produced in the MongoDB instance of the i4Q^DR Solution. Specifically, the data is stored in a collection with the name *i4q_dit_data*, which is located inside a database with the name *fidia*. In this collection, the following fields are stored: *XMotorCurrent*, *XMotorCurrent_rolling_median*, *YMotorCurrent*, *YMotorCurrent_rolling_median*, *ZMotorCurrent*, *ZMotorCurrent_rolling_median*, *XAbsPosition*, *XAbsPosition_rolling_median*, *YAbsPosition*, *YAbsPosition_rolling_median*, *ZAbsPosition*, *ZAbsPosition_rolling_median*, *XVibration*, *YVibration*, *ZVibration*, *timestamp*, *XVibration_FFT*, *YVibration_FFT*, *ZVibration_FFT*, *XVibration_BW*, *YVibration_BW*, *ZVibration_BW*.

2.1.5.2 Inbound integration with i4Q^QD Solution

- **Input solution**. i4Q^DR receives the data processed by the i4Q^QD Solution to store it in a database.
- **Information received**. The i4Q^DR Solution receives the JSON alerts produced by the i4Q^QD indicating that a faulty machined product has been detected. After this, the data is stored in a MongoDB instance.
- **Communication mechanism**. The communication between these two solutions is done directly, the i4Q^QD Solution being the one that stores the data produced in the MongoDB instance of the i4Q^DR Solution. Specifically, the data is stored in the *i4q_qd_results* collection, which is located inside the *fidia* database. Within this collection, the following fields can be found: *XVibration_Values*, *YVibration_Values*, *ZVibration_Values*, *XVibration_Values_FFT*, *YVibration_Values_FFT*, *ZVibration_Values_FFT*, *XVibration_Values_BW*, *YVibration_Values_BW*, *ZVibration_Values_BW*, and *Predicted Chatter*.

2.1.5.3 Inbound integration with i4Q^LRT Solution

- **Input solution**. i4Q^DR Solution receives the data processed by i4Q^LRT to store it in a database.
- **Information received**. A file corresponding to the ML model generated by the i4Q^LRT Solution.
- **Communication mechanism**. The communication between these two solutions is done in a direct manner, which means that i4Q^LRT Solution connects to the MinIO instance deployed by i4Q^DR to store the file in the corresponding bucket.

2.1.5.4 Outbound integration with i4Q^{LRT} Solution

- **Output solution**. i4Q^{LRT} Solution obtains from the i4Q^{DR} Solution the data previously stored by the i4Q^{DIT} Solution in the *i4q_dit_data* collection.
- **Information sent**. Processed manufacturing data that will be used by the i4Q^{LRT} Solution to train the ML model it generates.
- **Communication mechanism**. The communication between these two solutions is done directly, which means that i4Q^{LRT} Solution will connect to the *fidia* database created in the MongoDB instance deployed by i4Q^{DR} to retrieve the necessary information from the *i4q_dit_data* collection.

2.1.6 i4Q^{LRT} – Manufacturing Line Reconfiguration Toolkit

2.1.6.1 Inbound integration with i4Q^{DR} Solution

- **Input solution.** i4Q^{LRT} receives data stored in the i4Q^{DR} Solution.
- **Information received.** This solution connects to the MongoDB instance managed by the i4Q^{DR} Solution to obtain the factory data that was previously processed and stored by the i4Q^{DIT} Solution. Then, this data is used to train the ML model that will be generated by the i4Q^{LRT} Solution.
- **Communication mechanism.** The communication between these two solutions is done directly, as the i4Q^{LRT} Solution will connect to the *fidia* database created in the MongoDB instance deployed by i4Q^{DR} to retrieve the necessary information from the *i4q_dit_data* collection.

2.1.6.2 Outbound integration with i4Q^{QD} Solution

- **Output solution.** i4Q^{LRT} sends the results produced by the reconfiguration algorithm to the i4Q^{QD} Solution.
- **Information sent.** When the i4Q^{QD} Solution detects a faulty machined part, it will make a request to the i4Q^{LRT} to launch the reconfiguration algorithm. Then, this algorithm will generate a possible reconfiguration solution and will return it to the i4Q^{QD} Solution.
- **Communication mechanism.** The communication between these two solutions is done through the Message Broker, as the i4Q^{LRT} sends the results produced by the reconfiguration algorithm via the *LRT_Output* topic.

2.1.7 i4Q^{AD} – Analytics Dashboard

The i4Q^{AD} Solution is a reporting interface that allows monitoring industrial data with fully flexible visualisation drill-down charts and a flexible dashboard to provide meaningful analytics to users on a real-time basis using incremental algorithms. In Pilot 1, the solution is used to create dashboards for monitoring the KPIs related to this pilot.

2.1.7.1 Inbound integration with i4Q^{LRT} Solution

- **Input solution.** i4Q^{AD} can receive data from the Message Broker or from any other solution, such as i4Q^{LRT}.
- **Information received.** This solution receives the results produced by the i4Q^{LRT} Solution in the form of one or more datasets. Then, the data received is processed with the assistance of Apache Druid. This tool provides batches of data that can be consumed by Apache Superset. Finally, this other tool generates the different dashboards according to the needs indicated by the user.
- **Communication mechanism.** This solution uses the Message Broker as a communication mechanism to receive information from the different solutions involved in the pilot. In this case, the data received from the i4Q^{LRT} Solution will arrive via the *LRT_topic_1*, but the data may arrive via other topics, depending on the solution through which it is sent. Apache Druid will subscribe to these topics and pre-process the data before consuming it.

2.1.8 Message Broker

The Message Broker is the communication component established for the i4Q Project that drives the interoperable nature of the i4Q Solutions suite. The Message Broker acts as an intermediary between the different solutions, allowing the exchange of data in real-time. Message Broker is powered by Apache Kafka, which is a distributed streaming platform designed to handle large volumes of data. It operates on a publish-subscribe model, where producers publish data to topics, and consumers subscribe to these topics to receive the data. Kafka's architecture consists of brokers that manage the storage of data, ensuring fault tolerance and scalability. Overall, the Message Broker provides a reliable and scalable solution for the i4Q suite, facilitating the implementation of real-time data pipelines and stream processing applications.

2.2 Pilot 2: BIESSE - Diagnostics and IoT Services

This subsection analyses the pipeline of the *Diagnostics and IoT Services* pilot, corresponding to the i4Q Solutions deployed in the facilities of the BIESSE industrial partner, and explains in a justified manner the modifications made since the last deliverable [1] was submitted until the present day.

Based on the analysis of the integration between the i4Q Solutions conducted with solution providers and the pilot leader, it has been concluded that the most recent version of the pipeline is the one shown in Figure 2.

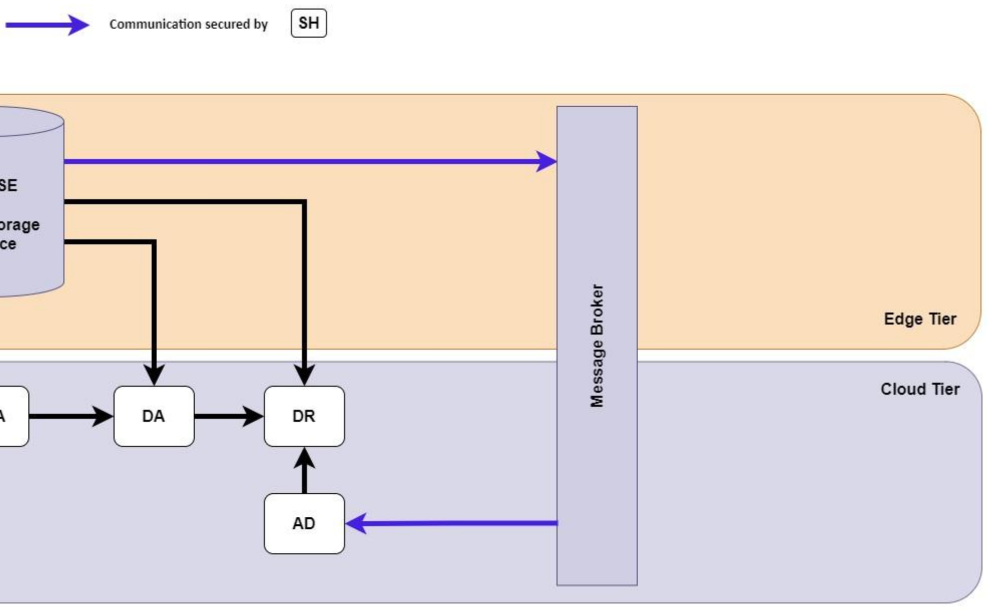

Figure 2. Pilot 2 pipeline diagram

Regarding the pipeline diagram of this pilot, no modifications have been introduced in comparison with the version presented in the previous deliverable. Therefore, there is nothing to comment on in terms of modifications.

2.2.1 i4Q^{BDA} – Big Data Analytics Suite

In this Pilot, the i4Q^{BDA} Solution is used to configure and create a deployable software bundle containing the i4Q^{DA} Solution and all necessary technologies to be deployed on the pilot's infrastructure so that these technologies are optimised for deployment and execution according to the pilot infrastructure's hardware specifications.

This solution does not receive input from any other solution and sends the optimal configuration parameters to the i4Q^{DA} Solution to be deployed on the pilot's infrastructure.

2.2.2 i4Q^{DA} – Services for Data Analytics

In Pilot 2, the i4Q^{DA} Solution is used to provide analytics and the ability to create data workflows without programming knowledge that are stored, and their execution scheduled. This solution is also integrated with the following solutions present in the pilot:

- **i4Q^{DR} Solution**. It is used to feed historical data to perform analytics on, as well as to store results of the workflows created by the user with i4Q^{DA}.
- **i4Q^{AD} Solution**. The results produced by the i4Q^{DA} Solution can be displayed in the I4Q^{AD}.
- **Message Broker**. The i4Q^{DA} Solution can use data coming from the Message Broker to perform analytics as well as send messages if needed in a workflow.

2.2.3 i4Q^{DR} – Data Repository

In Pilot 2, the i4Q^{DR} Solution has two types of integrations: an inbound integration with the i4Q^{DA} Solution, as the results produced by the i4Q^{DA} are stored in the i4Q^{DR}; and an outbound integration with the i4Q^{AD} Solution, as the information stored in the i4Q^{DR} is used to produce dashboards in the i4Q^{AD} Solution. To facilitate the reading, each integration is described in more detail below in a separate way.

2.2.3.1 Inbound integration with i4Q^{DA} Solution

- **Input solution**. i4Q^{DR} receives data from the i4Q^{DA} Solution.
- **Information received**. After performing an analysis of the raw data coming from the edge, the solution stores the results produced in an i4Q^{DR} data collection.
- **Communication mechanism**. The integration between these two solutions is done in a direct way, as the i4Q^{DA} Solution is responsible for storing the results it produces in a data collection of the i4Q^{DR}. For privacy reasons, no further details regarding the type of data or the structure of the stored data can be provided.

2.2.3.2 Outbound integration with i4Q^{AD} Solution

- **Output solution**. The i4Q^{AD} Solution fetches the information stored in the i4Q^{DR}. Since the information is transferred from the i4Q^{DR} to the i4Q^{AD}, the integration performed by the former solution is in the outbound direction with respect to the second one.
- **Information sent**. The i4Q^{AD} Solution retrieves the necessary information from the i4Q^{DR} to generate graphs and dashboards that can be useful for plant operators. For privacy reasons, no further details of the information retrieved, and the graphs generated can be provided.
- **Communication mechanism**. The integration between the two solutions has been carried out indirectly, as the Apache Druid and Trino tools have been used to make the data stored in the i4Q^{DR} Solution accessible to generate the appropriate graphs in the i4Q^{AD} Solution.

2.2.4 i4Q^{AD} – Analytics Dashboard

In Pilot 2, the solution is used to create dashboards for monitoring the KPIs related to this pilot. For ease of reading, each integration is described in more detail below in a separate way.

2.2.4.1 Inbound integration

- **Input solution**. According to Figure 2, the i4Q^{AD} Solution receives data only from the Message Broker but does not receive data directly from other solutions.
- **Information received**. This solution receives data sent through the Message Broker by other solutions and processes it using Apache Druid. This tool provides batches of data that can be consumed by Apache Superset. Finally, this other tool generates the different

dashboards according to the needs indicated by the user.

- **Communication mechanism**. This solution uses the Message Broker as a communication mechanism to receive information from the different pilot solutions. Therefore, the topic by which it will receive the information will depend on the solution sending the data. Apache Druid will subscribe to these topics and pre-process the data before consuming it.

2.2.5 i4Q^{SH} – IIoT Security Handler

As for the i4Q^{SH} Solution, the information concerning the integration of this with the rest of the i4Q Solutions in the pilot is exactly the same described in subsection 2.1.2. To avoid repetition of information, it is recommended to refer to this subsection for more information.

2.2.6 Message Broker

As for the Message Broker, the information concerning the integration of this tool with the rest of the i4Q Solutions in the pilot is the same as described in subsection 2.1.8. For minimising information redundancy, it is recommended to consult this subsection for obtaining more details.

2.3 Pilot 3: WHIRLPOOL – White Goods Product Quality

In this subsection, the pipeline of the White Goods Product Quality pilot, corresponding to the i4Q Solutions deployed at WHIRLPOOL's industrial partner's facilities, is analysed and the modifications made since the last deliverable [1] was presented until today are explained in detail.

According to this analysis, it has been agreed with the pilot leader and the solution providers involved that the final version of the pipeline is the one shown in Figure 3.

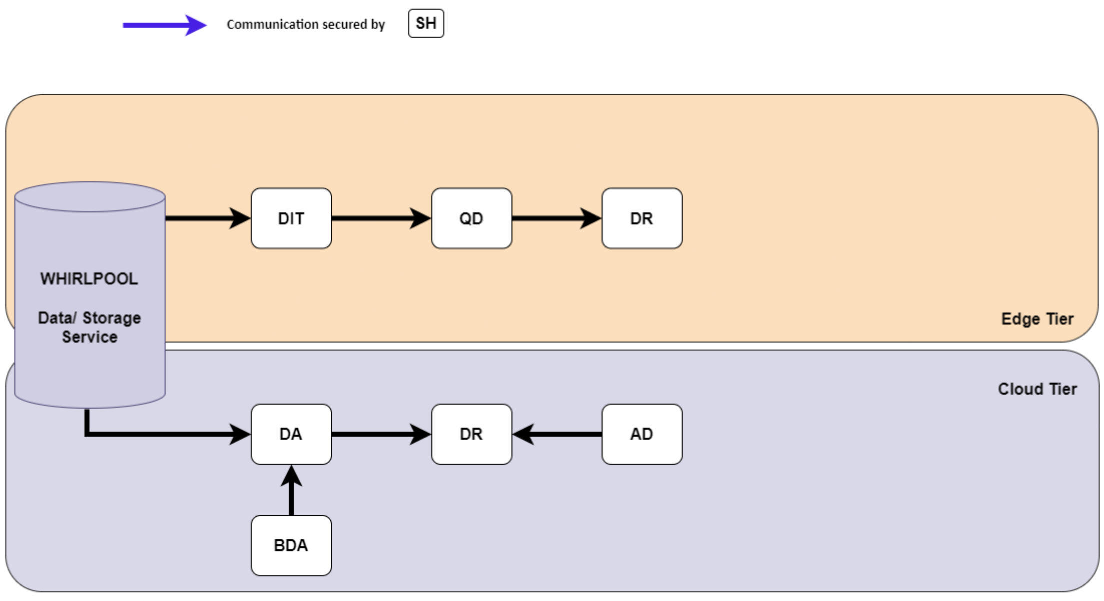

Figure 3. Pilot 3 pipeline diagram

No modifications have been introduced in the pipeline diagram of this pilot since the version presented in the preceding deliverable. For that reason, there is nothing to comment on regarding modifications.

2.3.1 i4Q^{DIT} – Data Integration and Transformation Services

In this pilot, the i4Q^{DIT} Solution receives an initial set of raw sensor data, applies cleaning and preprocessing functions and subsets generation, since the initial dataset was quite large.

After this, the integrations of this solution with the others in the pilot are analysed. In this case, the i4Q^{DIT} Solution has an inbound integration with the Google Cloud Platform using the i4Q^{DR} Solution, and an outbound integration with the i4Q^{QD}, which are described in detail below for ease of readability.

2.3.1.1 Inbound integration with database stored in the factory's Google Cloud Platform

- **Input solution**. i4Q^{DIT} receives data from a data collection stored in the factory, specifically, in Google Cloud platform, through the i4Q^{DR} Solution.
- **Information received**. The solution receives the raw sensor signals and applies cleaning and preprocessing functions as well as feature selection and subsets generation.
- **Communication mechanism**. The integration of i4Q^{DIT} and the communication with the factory's storage was conducted through the solution i4Q^{AD}.

2.3.1.2 Outbound integration with i4Q^{QD} Solution

- **Output solution**. The i4Q^{DIT} Solution sends the prepared data to i4Q^{QD}.
- **Information sent**. i4Q^{DIT} Solution provides i4Q^{QD} with a processed and clean dataset, as well as various subsets of the initial data, on demand.
- **Communication mechanism**. The communication between the two solutions is done through the Message Broker.

2.3.2 i4Q^{QD} – Rapid Quality Diagnosis

In Pilot 3, the i4Q^{QD} Solution is responsible for detecting products that don't conform with the set quality standards. This is achieved by utilising Machine Learning algorithms and exploiting the sensor signals during the EOL testing of the washing machine.

Regarding integrations, this solution has two different types: an inbound one with the i4Q^{DIT} Solution, and an outbound one with the i4Q^{DR} Solution. To facilitate the reading of this document, these are described in more detail in separate subsections below.

2.3.2.1 Inbound integration with i4Q^{DIT} Solution

- **Input solution**. The i4Q^{QD} is connected to the i4Q^{DIT} Solution to receive the sensors data.

- **Information received**. The solution receives a preprocessed and enhanced dataset from the i4Q^{DIT} comprising all the necessary information for developing and training product defect detection algorithm.
- **Communication mechanism**. The prepared data is ingested to the i4Q^{QD} Solution using the PostgreSQL database provided by the i4Q^{DR} Solution.

2.3.2.2 Outbound integration with i4Q^{DR} Solution

- **Output solution**. The i4Q^{QD} Solution stores its analytic results in the i4Q^{DR} Solution.
- **Information sent**. The evaluation metric of product defect detection model, along with its feature importances and prediction results, are formatted properly to be stored in the i4Q^{DR} Solution.
- **Communication mechanism**. Communication between the two solutions is established by connecting directly to the PostgreSQL instance of the i4Q^{DR} Solution.

2.3.3 i4Q^{DR} – Data Repository

In Pilot 3, i4Q^{DR} is integrated with three main solutions: i4Q^{QD}, i4Q^{DA}, and i4Q^{AD}.

- i4Q^{QD}. The i4Q^{QD} Solution stores the AI model to be run by the i4Q^{DA}.
- i4Q^{DA}. The i4Q^{DA} Solution stores the results of the data analysis performed on raw data coming directly from the edge. Moreover, a custom component has been designed to be run on i4Q^{DA}, which programmatically ingests data coming from the WHIRLPOOL's Google Cloud Platform to provide always up-to-date measurements from WHIRLPOOL's data lake to the i4Q^{DR} Solution.
- i4Q^{AD}. The i4Q^{AD} Solution fetches the information stored to graph and display dashboards to the plant operators.

2.3.4 i4Q^{BDA} – Big Data Analytics Suite

In Pilot 3, the information concerning the integration of the i4Q^{BDA} Solution with the rest of the i4Q Solutions involved is the same as described in subsection 2.2.1. For reducing the repetition of information, it is recommended to refer to this subsection for more details.

2.3.5 i4Q^{DA} – Services for Data Analytics

In Pilot 3, the information concerning the integration of the i4Q^{DA} Solution with the rest of the i4Q Solutions involved is the same as described in subsection 2.2.2. To minimise information redundancy, it is recommended to consult this subsection in order to obtain detailed information.

2.3.6 i4Q^{AD} – Analytics Dashboard

In Pilot 3, the solution is used to create dashboards for monitoring the KPIs related to this pilot. For ease of reading, each integration is described in more detail below in a separate way.

2.3.6.1 Inbound integration

- **Input solution**. The solution can receive data from the Message Broker or from any other solution, such as i4Q^LRT^.
- **Information received**. This solution receives one or more datasets, which are processed with the assistance of Apache Druid. This tool provides batches of data that can be consumed by Apache Superset. Finally, this other tool generates the different dashboards according to the needs indicated by the user.
- **Communication mechanism**. This solution uses the Message Broker as a communication mechanism to receive information from the different pilot solutions. Therefore, the topic by which it will receive the information will depend on the solution sending the data. Apache Druid will subscribe to these topics and pre-process the data before consuming it.

2.4 Pilot 4: FACTOR - Aeronautics and Aerospace Metal Parts Quality

This subsection analyses the pipeline of the *Aeronautics and Aerospace Metal Parts Quality* pilot, which corresponds to the i4Q Solutions deployed at the facilities of the industrial partner FACTOR and explains in a justified manner the possible modifications made since the last deliverable [1] was presented until today.

After checking the integrations among the different solutions with all the partners involved, it has been concluded that the final version of the pipeline diagram is as shown in Figure 4.

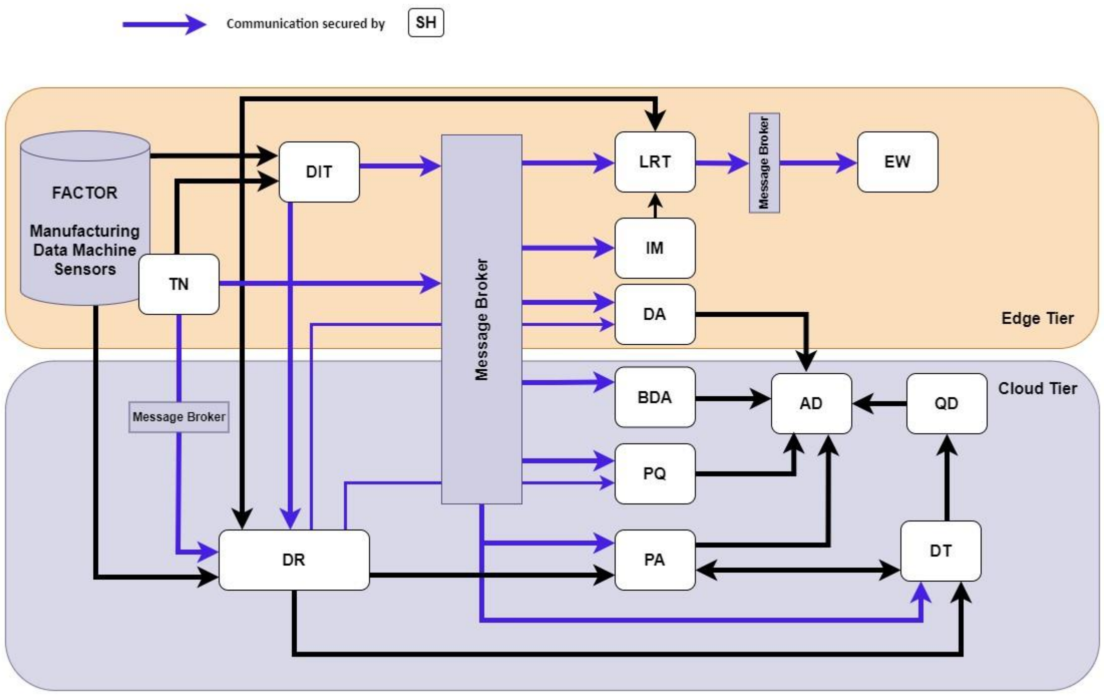

Figure 4. Pilot 4 pipeline diagram

The pilot diagram is the same as the one presented in the previous deliverable, so it can be concluded that no modifications have been made during this time.

2.4.1 i4Q^{TN} – Trusted Networks with Wireless and Wired Industrial Interfaces

Regarding the system integration of the i4Q^{TN} Solution in Pilot 4, the system is integrated with the Message Broker (Kafka) to publish all the gathered information from industrial sensors connected to the Industrial Wireless Sensor Nodes. Then, the solution oversees different industrial sensors from the factory and different processes, centralise all this data through the gateway and the SDN controller and expose all the information to the main Message Broker. This integration enables new data sources to the entire architecture, connecting isolated legacy machines or new sensors not available until now.

2.4.2 i4Q^{DIT} – Data Integration and Transformation Services

2.4.2.1 Inbound integration with the factory's infrastructure

- **Input solution.** This solution receives data directly from the pilot's infrastructure, so it does not have an input solution that provides the data.
- **Information received.** The i4Q^{DIT} Solution receives measurements from sensors that monitor the function of the machines.
- **Communication mechanism.** The solution communicates with the system Node-RED of the factory through the Message Broker.

2.4.2.2 Outbound integration with Infrastructure Monitoring

- **Output solution**. The i4Q^{DIT} sends the prepared data directly to the i4Q^{IM} for the latter to build a prediction model.
- **Information sent**. The i4Q^{DIT} provides CNC data to the i4Q^{IM} Solution.
- **Communication mechanism.** The two solutions communicate through the Message Broker.

2.4.3 i4Q^{DR} – Data Repository

In Pilot 4, the i4Q^{DR} Solution is responsible for providing data persistence to those solutions that consider it necessary to maintain a history of the data that they have produced. To do so, this solution manages a MongoDB database, in which different collections have been created to store the data generated by each of the solutions separately.

In terms of integrations, this solution involves many different integrations. For this reason, and with the objective of facilitating further reading, it has been decided to define each one of them in separate subsections.

2.4.3.1 Inbound integration with FACTOR's manufacturing data

FACTOR's manufacturing data is collected by an instance of MTLINK software, which is already deployed at FACTOR's infrastructure. The data gathered by the MTLINK is stored into a database created at MongoDB server, which will be referred as "*FACTOR's MongoDB*" in the rest of this section.

Since other i4Q Solutions need access to FACTOR's manufacturing data, part of the data stored in Factor's MongoDB is being replicated into a MongoDB instance deployed by i4Q^{DR}, called "*DR's MongoDB*" in the following for disambiguation purposes. The objective of this replication is to minimize the interference of this pilot use case in the company's production process, so that possible problems in the implementation of the pipeline do not cause side-effects.

The replication of data mentioned consists of the following steps:

1. Use of the i4Q^{DR} to deploy an instance of MongoDB for the "Single Server with security" scenario.
2. Creation of the database in the DR's MongoDB to store the replica of the manufacturing data.
3. Implementation of a script to export the relevant data from FACTOR's MongoDB and store it in the DR's MongoDB, namely in the database created in the step before. To perform both operations, connections to the corresponding database are opened. For readability purposes, this script is called "*exporter script*".
4. Initial execution of the exporter script to retrieve from FACTOR's MongoDB the data generated in the last 6 months.
5. Afterwards, periodically execution of the exporter script (probably on a daily basis) to update the DR's MongoDB with the latest manufacturing data.

To sum up, the most relevant aspects of this inbound integration with the i4Q^{DR} Solution are:

- **Input component**. FACTOR's MongoDB.
- **Information received**. Manufacturing data stored into FACTOR's MongoDB.
- **Communication mechanism**. This integration does not involve the Message Broker. Instead, the "exporter" script described above is used to replicate the most relevant data stored into FACTOR's MongoDB into the DR's MongoDB.

2.4.3.2 Inbound integration with i4Q^{DIT}

- **Input solution**. i4Q^{DR} receives data from the i4Q^{DIT} Solution.
- **Information received**. A CSV file that needs to be accessible for further use.
- **Communication mechanism**. This integration does not involve the Message Broker. The i4Q^{DIT} Solution opens a connection to the MinIO instance deployed by the i4Q^{DR} for storing the file in a bucket.

2.4.3.3 Inbound integration with i4Q^{LRT}

- **Input solution**. i4Q^{DR} receives data from the i4Q^{LRT} Solution.
- **Information received**. A file corresponding to the ML model generated by i4Q^{LRT} which needs to be accessible for further uses.

- **Communication mechanism**. This integration does not involve the Message Broker. The i4Q^{LRT} Solution opens a connection to the MinIO instance deployed by the i4Q^{DR} for storing the model into a bucket.

2.4.3.4 Inbound integration with i4Q^{TN}

- **Input solution**. i4Q^{DR} receives data from the i4Q^{TN} Solution.
- **Information received**. Sensor's information gathered and sent by IWSN nodes deployed at FACTOR's premises. Such information is received as messages in JSON format.
- **Communication mechanism**. The integration among both solutions will be made via the Message Broker. More specifically, i4Q^{DR} will consume the information published by i4Q^{TN} in topic *p4_tn2dr*. The structure and content of the received messages depends on the sensor that originally sent the data. The received information will be stored in a database created in the DR's MongoDB instance.

2.4.3.5 Outbound integrations with i4Q^{DA}, i4Q^{DT}, and i4Q^{PQ}

- **Output solutions**. i4Q^{DA}, i4Q^{DT}, and i4Q^{PQ} Solutions.
- **Information sent**. FACTOR's manufacturing data stored in a database created in DR's MongoDB instance.
- **Communication mechanism**. This integration does not involve the Message Broker. The output solutions connect to the database created in the DR's MongoDB storing the replica of FACTOR's manufacturing data, to extract the specific data required in each case.

2.4.3.6 Outbound integration with i4Q^{LRT}

- **Output solution**. i4Q^{DR} sends data to the i4Q^{LRT} Solution.
- **Information sent**. Manufacturing data generated in the last 6 months to train the model generated by the i4Q^{LRT}.
- **Communication mechanism**. This integration does not involve the Message Broker. The i4Q^{LRT} Solution opens a connection to the MinIO instance deployed by the i4Q^{DR} for storing the model into a bucket.

2.4.3.7 Outbound integration with i4Q^{PA}

- **Output solution**. i4Q^{DR} sends data to the i4Q^{PA} Solution.
- **Information sent**. Files corresponding to models to be used by the i4Q^{PA} Solution.
- **Communication mechanism**. This integration does not involve the Message Broker. The i4Q^{PA} Solution connects to a bucket created in the MinIO instance deployed by the i4Q^{DR}.

2.4.4 i4Q^{LRT} – Manufacturing Line Reconfiguration Toolkit

As part of Pilot 4, the i4Q^{LRT} Solution integrates with the Message Broker for real-time data processing and analysis. This integration allows the solution to access continuous streams of data from various sources, such as sensors and monitoring systems on the production line. The algorithm implemented in i4Q^{LRT} takes advantage of this capability to perform real-time analysis of the data received, which facilitates the detection of patterns, trends and anomalies in the operation of the line.

The FACTOR algorithm utilises a pre-trained Recurrent Neural Network (RNN) model to make predictions on incoming data. Its operation involves initializing the model and preparing data through pre-processing, making predictions using the loaded model, and finally post-processing the predictions to classify events. This algorithm provides an efficient way to identify specific patterns and events in the data.

As part of the training process, an algorithm trains an RNN model to make binary predictions based on sequential data. Its stages include data loading and pre-processing, defining and training the LSTM (Long Short-Term Memory) neural network model, and evaluating the model to measure its performance. This process allows for adjusting the model's hyperparameters to optimise its accuracy and effectiveness in event prediction.

2.4.5 i4Q^{IM} – Infrastructure Monitoring

In Pilot 4, the i4Q^{IM} Solution is responsible for ensuring the optimal condition of the Nakamura2 CNC machine by identifying potential malfunction during its operation and proactively alerting the operators. The solution employs Machine Learning algorithms and historical machine sensor signals to correlate the occurrence of machine malfunctions to specific operation conditions.

2.4.5.1 Inbound integration with i4Q^{DIT} Solution

- **Input solution**. The i4Q^{IM} is directly connected to the i4Q^{DIT} Solution to receive the appropriate machine sensor signals.
- **Information received**. The solution receives the necessary CNC after they have been properly preprocessed and enhanced by the i4Q^{DIT} Solution. This data is then being used to train the underlying ML model responsible for the detection of machine malfunctions.
- **Communication mechanism**. The Message Broker is used as a communication mechanism to consume the data prepared by the i4Q^{DIT} Solution using the topic *DIT_topic_1*.

2.4.5.2 Outbound integration with i4Q^{DR} Solution

- **Output solution**. The i4Q^{IM} Solution stores its analytic results in the *i4q_im_results* collection of the i4Q^{DR} Solution.
- **Information sent**. The prediction results made by the CNC machine malfunction detection model, along with the associated sensor data, are formatted properly and stored in the i4Q^{DR} Solution, for long term access and analysis.
- **Communication mechanism**. Communication between the two solutions is established by connecting directly to the MongoDB instance of the i4Q^{DR} Solution.

2.4.6 i4Q^{BDA} – Big Data Analytics Suite

In this Pilot, the information concerning the integration of the i4Q^{BDA} Solution with the rest of the i4Q Solutions involved is the same as described in subsection 2.2.1. For reducing the repetition of information, it is recommended to refer to this subsection for obtaining more details.

2.4.7 i4Q^{DA} – Services for Data Analytics

In Pilot 4, the information concerning the integration of the i4Q^{DA} Solution with the rest of the i4Q Solutions involved is the same as described in subsection 2.2.2. To minimise information redundancy, it is recommended to consult this subsection for obtaining more details.

2.4.8 i4Q^{PQ} – Data-Driven Continuous Process Qualification

In Pilot 4, the i4Q^{PQ} Solution integrates with two key components: the i4Q^{DR} for inbound data integration and Message Broker (Kafka) for real-time data processing and analysis. These integrations are crucial for the i4Q^{PQ} Solution's functionality in continuous process qualification and quality assurance. For ease of understanding, each integration is described in more detail below.

2.4.8.1 Integration with i4Q^{DR}

- **Input solution**. i4Q^{PQ} receives data directly from the i4Q^{DR} Solution.
- **Information received**. The i4Q^{DR} Solution aggregates and stores processed data from various sources, including real-time and historical manufacturing data. i4Q^{PQ} utilises this consolidated data to analyse process capability and quality range adherence.
- **Communication mechanism**. This integration is facilitated through a direct connection, where i4Q^{PQ} accesses the MongoDB instance managed by i4Q^{DR}. Specifically, i4Q^{PQ} queries the data stored within the database, utilizing collections that encompass machine performance metrics and quality indicators.

2.4.8.2 Integration with Message Broker

- **Input solution**. i4Q^{PQ} subscribes to topics within the Kafka Message Broker to receive real-time data streams.
- **Information received**. Kafka channels deliver real-time data on manufacturing processes, including machine performance, quality metrics, and operational efficiency indicators. This data is critical for i4Q^{PQ} to perform immediate analysis and quality control measures.
- **Communication mechanism**: The integration with Kafka is established through a subscription model. i4Q^{PQ} subscribes to specific topics configured within Kafka that are relevant to the manufacturing process's quality and performance. As data is published on these topics, i4Q^{PQ} consumes the information in real-time, enabling dynamic process qualification and adjustments.

2.4.9 i4Q^{EW} – Edge Workloads Placement and Deployment

In Pilot 4, the i4Q^{EW} Solution integrates with two key components: the i4Q^{AI}, with which it integrates to achieve required integration between the AI models and workloads. In addition, it integrates with the i4Q^{LRT} Solution, which provides the artefacts to be deployed at the edge.

For ease of understanding, each integration is described in more detail below.

2.4.9.1 Integration with i4Q^{AI}

- **Input solution**. i4Q^{EW} collaborates on deploying artefacts on the edge, mainly with the i4Q^{AI} Solution.
- **Information received**. This solution gets information from the i4Q^{LRT} Solution via Git to understand when they need to deploy/redeploy artefacts.
- **Communication mechanism**. This integration is facilitated through hooks to a Git repository. Both, i4Q^{AI} and i4Q^{EW}, get notifications from Git to understand whether their intervention is required. Whenever there are new artefacts to be deployed, the information is pulled from Git and gets deployed on eligible managed edge clusters.

2.4.10 i4Q^{AI} – AI Models Distribution to the Edge

In Pilot 4, the i4Q^{AI} Solution integrates with two key components: the i4Q^{EW}, with which it integrates to achieve required integration between the AI models and workloads. In addition, it integrates with the i4Q^{LRT} Solution, which provides the artefacts to be deployed at the edge.

For ease of understanding, each integration is described in more detail below.

2.4.10.1 Integration with i4Q^{EW}

- **Input solution**. i4Q^{EW} collaborates on deploying artefacts on the edge, mainly with i4Q^{AI}. I4Q^{AI} and i4Q^{EW} Solutions collaborate for deploying AI workloads and models at the edge.
- **Information received**. This solution gets information from the i4Q^{LRT} Solution via Git to understand when they need to deploy/redeploy artefacts.
- **Communication mechanism**. This integration is facilitated through hooks to a Git repository. Both i4Q^{AI} and i4Q^{EW}, get notifications from Git to understand whether their intervention is required. Whenever there are new artefacts to be deployed, the information is pulled from Git and gets deployed on eligible managed edge clusters.

2.4.11 i4Q^{AD} – Analytics Dashboard

In Pilot 4, the solution is used to create dashboards for monitoring the KPIs related to the project.

2.4.11.1 Inbound integration

- **Input solution**. The solution can receive data from the Message Broker or from any other solution, such as i4Q^{LRT}.
- **Information received**. This solution receives one or more datasets, which are processed with the assistance of Apache Druid. This tool provides batches of data that can be consumed by Apache Superset. Finally, this other tool generates the different dashboards according to the needs indicated by the user.
- **Communication mechanism**. This solution uses the Message Broker as a communication mechanism to receive information from the different pilot solutions. Therefore, the topic by which it will receive the information will depend on the solution sending the data. Apache Druid will subscribe to these topics and pre-process the data before consuming it.

2.4.12 i4Q^{DT} – Digital Twin Simulation Services

The i4Q^{DT} Solution is a tool whose main functionality is that of building models that could be exploited by other i4Q Solutions. Thus, it consumes static data and generates static data, being not applicable a real-time scenario for its use. Nevertheless, there is some integration with other solutions as described below:

- i4Q^{PA}. The i4Q^{PA} Solution can consume the models built with the i4Q^{DT} Solution, both from a database where they can be stored or directly from the Message Broker.
- i4Q^{DR}. The data need to carry out the simulations (in form of CSV files) can be consumed from the i4Q^{DR} Solution. The needed adaptation of the raw data from the machines of Pilot 4 is done through the i4Q^{DIT} Solution, which stores the transformed data in the i4Q^{DR}.
- i4Q^{QD}. The main functionality of this solution in Pilot 4 is that of developing and executing different data-driven quality models. As the approach taken for i4Q^{DT} in the context of this pilot has been finally a data-driven one, it has been decided that the i4Q^{DT} should absorb this task to avoid any overlapping of functionalities between solutions.

2.4.13 Message Broker

As for the Message Broker, the information concerning the integration of this tool with the rest of the i4Q Solutions in the pilot is the same as described in subsection 2.1.8. For minimising information redundancy, it is recommended to consult this subsection for obtaining more details.

2.5 Pilot 5: RIASTONE - Advanced In-line Inspection for Incoming Prime Matter Quality Control

In this subsection, the pipeline of the *Advanced In-line Inspection for Incoming Prime Matter Quality Control* pilot, corresponding to the i4Q Solutions deployed at RIASTONE's industrial partner's facilities, is analysed and the possible modifications made from the last deliverable [1] to the current time are explained.

After reviewing the integration between the solutions with the partners involved, it has been concluded that the final version of the pipeline diagram is as presented in Figure 5.

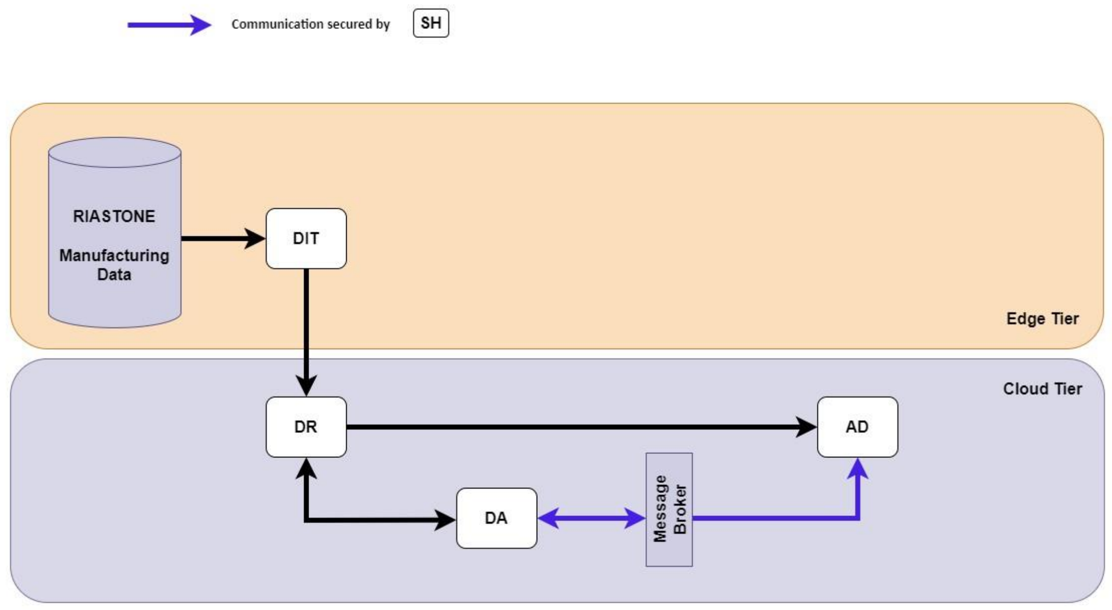

Figure 5. Pilot 5 pipeline diagram

The pilot pipeline diagram remains unchanged from the version presented in the last deliverable. Therefore, there are no comments to add concerning modifications in the integrations.

2.5.1 i4Q^{DIT} – Data Integration and Transformation Services

In Pilot 5, the i4Q^{DIT} Solution processes the spectrometer sensor data as well as the quality control data and applies a series of transformations to generate a clean and filtered dataset. After this series of transformations, an outbound integration between this solution and the i4Q^{DR} takes place, as the cleaned dataset is stored in a database managed by the i4Q^{DR}.

2.5.2 i4Q^{DR} – Data Repository

In Pilot 5, three integrations affecting the i4Q^{DR} Solution take place, these are: an inbound integration with the i4Q^{DA} Solution, as the results produced by this solution are stored in a database managed by the i4Q^{DR}; an inbound integration with the i4Q^{DIT} Solution, as the data transformed by this solution is stored in the i4Q^{DR}; and an outbound integration with the i4Q^{AD} Solution, as the information stored in the i4Q^{DR} is used for the generation of dashboards and charts.

2.5.3 i4Q^{DA} – Services for Data Analytics

In Pilot 5, the information concerning the integration of the i4Q^{DA} Solution with the rest of the i4Q Solutions involved is the same as described in subsection 2.2.2. To minimise information redundancy, it is recommended to consult this subsection for obtaining more details.

2.5.4 i4Q^{AD} – Analytics Dashboard

In Pilot 5, the solution is used to show data coming from all the production processes, from raw matter quality to final product quality. For ease of understanding, each integration is described in more detail below.

2.5.4.1 Inbound integration

- **Input solution**. The solution can receive data from the Message Broker or from any other solution, such as i4Q^{LRT}.
- **Information received**. This solution receives one or more datasets, which are processed with the assistance of Apache Druid. This tool provides batches of data that can be consumed by Apache Superset. Finally, this other tool generates the different dashboards according to the needs indicated by the user.
- **Communication mechanism**. This solution uses the Message Broker as a communication mechanism to receive information from the different pilot solutions. Therefore, the topic by which it will receive the information will depend on the solution sending the data. Apache Druid will subscribe to these topics and pre-process the data before consuming it.

2.5.5 i4Q^{SH} – IIoT Security Handler

Regarding the i4Q^{SH} Solution, details concerning the integration of this with the rest of the i4Q Solutions in the pilot are exactly the same as described in subsection 2.1.2. To avoid repeating information, it is recommended to consult this subsection for more information.

2.5.6 Message Broker

As for the Message Broker, the information concerning the integration of this tool with the rest of the i4Q Solutions in the pilot is the same as described in subsection 2.1.8. For minimising information redundancy, it is recommended to consult this subsection for obtaining more details.

2.6 Pilot 6: FARPLAS - Automatic Advanced Inspection of Automotive Plastic Parts

This subsection discusses the pipeline of the *Automatic Advanced Inspection of Automotive Plastic Parts*, which corresponds to the i4Q Solutions deployed at the facilities of the industrial partner FARPLAS and describes in a justified way the possible modifications made since the last deliverable [1] was submitted until the present time.

After checking the integrations between the different solutions with all the actors involved, it has been agreed that the most updated version of the pipeline diagram is the one presented in Figure 6.

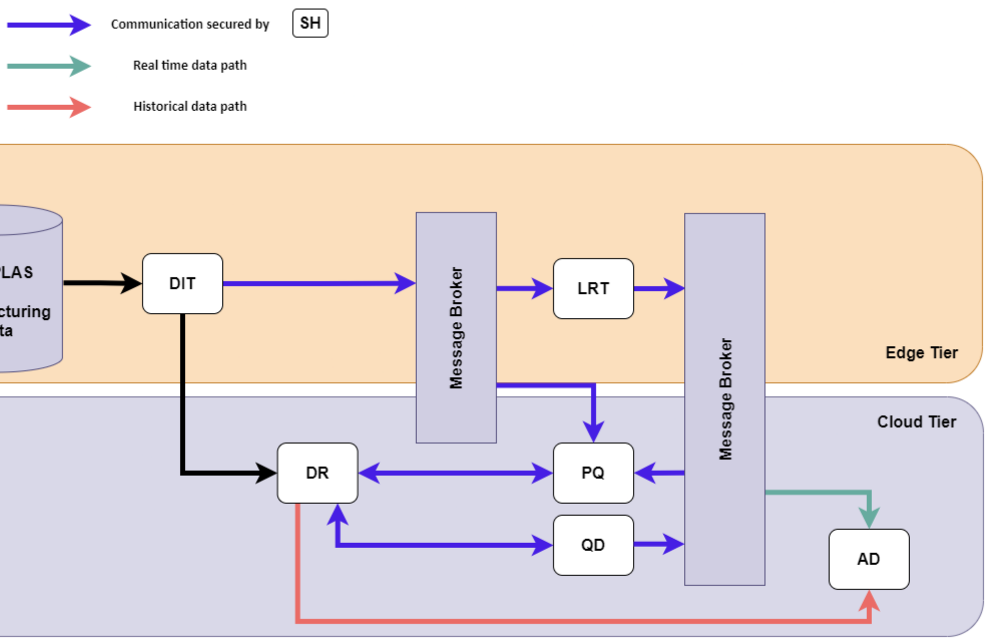

Figure 6. Pilot 6 pipeline diagram

Observing the diagram in the previous image and comparing it with that of Deliverable D6.17, the following modifications can be appreciated:

- Integration between the Message Broker and the i4Q^{DIT} Solution has been removed. This has been simplified so that i4Q^{DIT} uses the pilot's Kafka broker to consume the messages generated by the factory's machine.
- Integration between the i4Q^{LRT} and i4Q^{DR} Solutions has been removed. Currently, i4Q^{LRT} stores the necessary artefacts in MinIO and sends the results produced through a Message Broker topic. In this way, any solution that wants to consume this information can do so by subscribing to the corresponding topic.
- A direct integration between i4Q^{DIT} and i4Q^{DR} Solutions has been added. In this way, data produced by i4Q^{DIT} can be consumed in real-time by the Message Broker and, at the same time, can be stored in a data collection managed by i4Q^{DR}, without the need for i4Q^{DR} to consume data from a Message Broker topic.
- The arrow in the integration between the i4Q^{PQ} and i4Q^{DR} Solutions has been modified to represent a bidirectional communication, as the i4Q^{PQ} Solution makes a request to the i4Q^{DR} for obtaining the data previously stored by the i4Q^{DIT} and then, stores the results produced by the solution in a data collection of the i4Q^{DR}.
- The direction of the arrow in the integration between the i4Q^{PQ} Solution and the Message Broker has been modified, as it uses the Message Broker to obtain the data it needs, but the results produced by this solution are not sent via the Message Broker.
- The direction of the arrow in the integration between the i4Q^{QD} and i4Q^{DR} Solution has been changed to represent that the former makes a request to the latter for obtaining a set of data.
- An arrow has been added between the i4Q^{DR} and i4Q^{AD} Solution, which represents that the

i4Q^AD Solution makes requests to the i4Q^DR to obtain a set of historical data that will later be represented by different charts in one or more dashboards.

2.6.1 i4Q^DIT – Data Integration and Transformation Services

The i4Q^DIT Solution processes the sensor data from the injection moulding machines in this pilot and applies the corresponding transformations for adapting the output format of the data so that it can be used by other solutions, such as the i4Q^QD.

With regard to integrations, this solution participates in two: an inbound integration with the factory machine; an outbound integration with the i4Q^DR Solution and another outbound integration with the i4Q^QD Solution using the Message Broker. To facilitate the readability and detailed explanation of these, each one of them is analysed independently below.

2.6.1.1 Inbound integration with factory's machines

- **Input solution**. The i4Q^DIT is connected to the factory's machines to receive the sensors' data.
- **Information received**. The solution obtains the measurements produced in real-time by the sensors installed on the FARPLAS factory machine. The raw data is cleaned and processed to form one or more datasets to be further analysed.
- **Communication mechanism**. The integration between the solution and the factory machine occurs when the i4Q^DIT consumes the data produced in the *e117_cyc* topic by the Kafka broker installed in the FARPLAS factory.

2.6.1.2 Outbound integration with i4Q^DR Solution

- **Output solution**. The i4Q^DIT Solution sends the final dataset to the i4Q^DR Solution.
- **Information sent**. The data processed by the i4Q^DIT Solution has been stored in a MongoDB collection with the name *e117_cyc_dit_data*, which belongs to the database with the name *farplas*.
- **Communication mechanism**. Communication between these two solutions has been carried out directly, being the i4Q^DIT Solution the one in charge of storing the data in the aforementioned collection.

2.6.1.3 Outbound integration with i4Q^QD Solution using Message Broker

- **Output solution**. The i4Q^DIT Solution sends the final dataset to the i4Q^QD Solution.
- **Information sent**. The initial data has been filtered and techniques suitable for imbalanced samples have been applied. Thus, the i4Q^QD Solution receives a dataset with simulated values for building a better model.
- **Communication mechanism**. Communication between these two solutions has been established by the Message Broker. In this way, the results produced by the i4Q^DIT Solution

have been published in a certain topic, which can then be consumed by the i4Q^{QD} Solution.

2.6.2 i4Q^{SH} – IIoT Security Handler

As for the i4Q^{SH} Solution, the information concerning the integration of this with the rest of the i4Q Solutions in the pilot is exactly the same described in subsection 2.1.2. To avoid repetition of information, it is recommended to refer to this subsection for more information.

2.6.3 i4Q^{DR} – Data Repository

In Pilot 6, the i4Q^{DR} Solution is responsible for providing data persistence to those solutions that consider it necessary to maintain a history of the data that they have produced. To do so, this solution manages a MongoDB database, in which different collections have been created to store the data generated by each of the solutions separately.

In terms of integrations, this solution is involved in six different integrations, being these the following: three inbound integrations with the i4Q^{DIT}, i4Q^{PQ}, and i4Q^{QD} Solutions; and three outbound integrations with the i4Q^{PQ}, i4Q^{QD}, and i4Q^{AD} Solutions. For explaining each of these in detail and facilitate further reading, it has been decided to define each one of them in separate subsections.

2.6.3.1 Inbound integration with i4Q^{DIT} Solution

- **Input solution**. i4Q^{DR} receives data from the i4Q^{DIT} Solution.
- **Information received**. The i4Q^{DIT} Solution obtains real-time data from the FARPLAS factory machine. Then, it performs a cleaning, preprocessing and feature enrichment process. After this, it stores the data in a MongoDB instance deployed in the i4Q^{DR} Solution.
- **Communication mechanism**. The integration is done in a direct way, the i4Q^{DIT} Solution being the one that stores the data produced in the MongoDB instance of the i4Q^{DR} Solution. Specifically, the data is stored in a collection with the name *e117_cyc_dit_data*, which is located inside a database with the name *farplas*.

2.6.3.2 Inbound integration with i4Q^{PQ} Solution

- **Input solution**. i4Q^{DR} receives data from the i4Q^{PQ} Solution.
- **Information received**. The i4Q^{PQ} Solution receives a dataset and uses it to perform an analysis of the process capability and quality range adherence. Then, it stores this data in a MongoDB instance managed by the i4Q^{DR} Solution.
- **Communication mechanism**. The integration is done in a direct way, the i4Q^{PQ} Solution being the one that stores the results produced in the MongoDB instance of the i4Q^{DR} Solution. Specifically, the data is stored in a collection with the name *e117_cyc_pq_results*, which is located inside a database with the name *farplas*.

2.6.3.3 Inbound integration with i4Q^{QD} Solution

- **Input solution**. i4Q^{DR} receives data from the i4Q^{QD} Solution.
- **Information received**. The i4Q^{DR} Solution receives the JSON alerts produced by the i4Q^{QD} indicating that a faulty machined product has been detected. After this, the data is stored in a MongoDB instance.
- **Communication mechanism**. The communication between these two solutions is done directly, being the i4Q^{QD} the one that stores the data produced in the MongoDB instance of the i4Q^{DR} Solution. Specifically, the data is stored in the *e117_cyc_qd_results* collection, which is located in the *farplas* database.

2.6.3.4 Outbound integration with i4Q^{PQ} Solution

- **Output solution**. i4Q^{DR} receives a request from the i4Q^{PQ} Solution and returns the required dataset.
- **Information sent**. The data processed by the i4Q^{DIT} Solution is stored in a MongoDB data collection managed by the i4Q^{DR}. Then, the requested dataset is sent to the i4Q^{PQ} Solution for the analysis of process capability and quality range adherence.
- **Communication mechanism**. The communication between these two solutions is done directly. First, the i4Q^{PQ} performs the query of the necessary data to the corresponding database and data collection. In this case, this data will be found in the *e117_cyc_dit_data* collection, which is inside the *farplas* database, as this is where the i4Q^{DIT} has stored the processed data. Finally, the i4Q^{DR} is responsible for sending this data so that it can be used to perform the corresponding analysis.

2.6.3.5 Outbound integration with i4Q^{QD} Solution

- **Output solution**. i4Q^{DR} receives a request from the i4Q^{QD} Solution and returns the required dataset.
- **Information sent**. The data processed by the i4Q^{DIT} Solution is stored in a MongoDB data collection managed by the i4Q^{DR}. Then, the requested dataset is sent to the i4Q^{QD} Solution so that it can analyse the data and generate alerts for those parts that do not meet the established quality standards.
- **Communication mechanism**. The communication between these two solutions is done directly. First, the i4Q^{QD} performs the query of the necessary data to the corresponding database and data collection. In this case, this data will be found in the *e117_cyc_dit_data* collection, which is inside the *farplas* database, as this is where the i4Q^{DIT} has stored the processed data. Finally, the i4Q^{DR} is responsible for sending this data so that it can be used to analyse the parts quality.

2.6.3.6 Outbound integration with i4Q^{AD} Solution

- **Output solution**. i4Q^{DR} receives a request from the i4Q^{AD} Solution and returns the requested dataset.

- **Information sent**. The data processed by the i4Q^{DIT} Solution is stored in a MongoDB data collection managed by the i4Q^{DR}. Then, the requested dataset is sent to the i4Q^{AD} Solution so that the corresponding dashboards and charts can be created in the solution web interface.
- **Communication mechanism**. The communication between these two solutions is done directly. First, the i4Q^{AD} performs the query of the necessary data to the corresponding database and data collection. In this case, this data will be found in the *e117_cyc_dit_data* collection, which is inside the *farplas* database, as this is where the i4Q^{DIT} has stored the processed data. Next, the i4Q^{DR} is responsible for sending this data so that the i4Q^{AD} can access it. Finally, the i4Q^{AD} will be responsible for generating the dashboards and graphs indicated by the user in its web interface, which will help to interpret the values measured by the machine.

2.6.4 i4Q^{LRT} – Manufacturing Line Reconfiguration Toolkit

As part of Pilot 6, the i4Q^{LRT} Solution integrates with the Message Broker for real-time data processing and analysis. This integration allows the solution to access continuous streams of data from various sources, such as sensors and monitoring systems on the production line. The algorithm implemented in i4Q^{LRT} takes advantage of this capability to perform real-time analysis of the data received, which facilitates the detection of patterns, trends, and anomalies in the operation of the line.

2.6.5 i4Q^{PQ} – Data-Driven Continuous Process Qualification

In Pilot 6, the i4Q^{PQ} Solution integrates with two key components: the i4Q^{DR} for inbound data integration and Message Broker (Kafka) for real-time data processing and analysis. These integrations are crucial for the i4Q^{PQ} Solution’s functionality in continuous process qualification and quality assurance. For ease of understanding, each integration is described in more detail below.

2.6.5.1 Inbound integration with i4Q^{DR}

- **Input solution**. i4Q^{PQ} receives data directly from the i4Q^{DR} Solution.
- **Information received**. The i4Q^{DR} Solution aggregates and stores processed data from various sources, including real-time and historical manufacturing data. i4Q^{PQ} utilises this consolidated data to analyse process capability and quality range adherence.
- **Communication mechanism**. This integration is facilitated through a direct connection, where i4Q^{PQ} accesses the MongoDB instance managed by i4Q^{DR}. Specifically, i4Q^{PQ} queries the data stored within the database, utilising collections that encompass machine performance metrics and quality indicators.

2.6.5.2 Inbound integration with Message Broker

- **Input solution**. i4Q^{PQ} subscribes to topics within the Kafka Message Broker to receive real-time data streams.

- **Information received**. Kafka channels deliver real-time data on manufacturing processes, including machine performance, quality metrics, and operational efficiency indicators. This data is critical for i4Q^{PQ} to perform immediate analysis and quality control measures.
- **Communication mechanism**. The integration with Kafka is established through a subscription model. i4Q^{PQ} subscribes to specific topics configured within Kafka that are relevant to the manufacturing process's quality and performance. As data is published to these topics, i4Q^{PQ} consumes the information in real-time, enabling dynamic process qualification and adjustments.

2.6.6 i4Q^{QD} – Rapid Quality Diagnosis

In Pilot 6, the i4Q^{QD} Solution is responsible for identifying the occurrence of products that do not conform with the set quality standards during the plastic moulding injection process. Utilising Machine Learning algorithms, the solution can capture the correlation between the defective products and specific manufacturing conditions by exploiting the collected machine sensor signals.

Regarding integrations, this solution has integrations: an inbound one with the i4Q^{DIT} Solution, and an outbound one with the i4Q^{DR} Solution. For facilitating the reading of this document, these are described in more detail in separate subsections below.

2.6.6.1 Inbound integration with i4Q^{DIT} Solution

- **Input solution**. The i4Q^{QD} is connected to the i4Q^{DIT} Solution to receive the sensors data.
- **Information received**. The solution receives a pre-processed and enhanced dataset from the i4Q^{DIT} comprising all the necessary features for developing and training the product defect detection algorithm.
- **Communication mechanism**. The Message Broker is used as a communication mechanism to consume the data prepared by the i4Q^{DIT} Solution using the topic *DIT_topic_1*.

2.6.6.2 Outbound integration with i4Q^{DR} Solution

- **Output solution**. The i4Q^{QD} Solution stores its analytic results in the *e117_cyc_qd_results* collection of the i4Q^{DR} Solution.
- **Information sent**. The prediction results made by the product defect detection model, along with the associated sensor data, are formatted properly to be stored in the i4Q^{DR} solution.
- **Communication mechanism**. Communication between the two solutions is established by connecting directly to the MongoDB instance of the i4Q^{DR} Solution.

2.6.7 i4Q^{AD} – Analytics Dashboard

In Pilot 6, the solution is used to create dashboards for monitoring the KPIs related to this pilot.

Specifically, in this pilot, two communication channels have been established: one for real-time data (inbound integration with the Message Broker) and another for historical data (inbound integration with the i4Q^{DR} Solution). For ease of reading, each integration is described in more detail below in a separate way.

2.6.7.1 Inbound integration with the Message Broker

- **Input solution.** The i4Q^{AD} Solution receives data from the Message Broker.
- **Information received**. The data collected and processed by the i4Q^{DIT} Solution is sent via the Message Broker and is received by the i4Q^{AD} in real-time. Then, this data is processed with the assistance of Apache Druid and chunks of data are provided for their consumption via Apache Superset. Finally, this other tool generates the different dashboards according to the needs indicated by the user.
- **Communication mechanism.** This solution uses the Message Broker as a communication mechanism to receive information from the different pilot solutions. Therefore, the topic by which it will receive the information will depend on the solution sending the data. Apache Druid will subscribe to these topics and pre-process the data before consuming it.

2.6.7.2 Inbound integration with the i4Q^{DR}

- **Input solution.** The i4Q^{AD} Solution receives data from the i4Q^{DR}.
- **Information received**. The data collected and processed by the i4Q^{DIT} Solution is stored in a database of the i4Q^{DR} Solution. Afterwards, the i4Q^{AD} makes requests to the corresponding data collections in order to obtain historical data for a set of variables. Then, this data is processed by the Apache Druid tool and, finally, the Apache Superset receives this data and generates the different dashboards and charts according to the needs indicated by the user.
- **Communication mechanism.** This solution receives the data directly from the database managed by the i4Q^{DR}.

2.6.8 Message Broker

As for the Message Broker, the information concerning the integration of this tool with the rest of the i4Q Solutions in the pilot is the same as described in subsection 2.1.8. For minimising information redundancy, it is recommended to consult this subsection for obtaining more details.

3. Analysis of Results and Solutions Update

Once the integration phase defined in section 2 has been completed, solution providers have explained in detail the problems they have encountered during the integration of their solutions with the others. In addition, they have explained in a brief manner the actions implemented in order to correct these problems.

For knowing the problems encountered when integrating each of the solutions, the different solution providers have been asked to complete the following information in their corresponding table:

- **Problem Id**. This field represents a unique problem identifier for each of the i4Q Solutions. For this, the following format will be used: **SOLUTION_INITIALS-XXX**, where **XXX** will represent a numerical value starting at 001 and ending at the number of problems encountered in that solution.
- **Status**. Indicates the current status of the problem. This field can have the following values:
    - To be fixed, if a problem has been found but work on its solution has not yet begun.
    - Work in progress, if a problem has been found and the work on its solution has been started, but it has not been solved yet.
    - Fixed, if the problem has been corrected.
    - Discarded, after solving one problem, it may occur that it is no longer necessary to solve other problems arising from it, and this may lead to discarding the need to solve a problem that was initially detected.
- **Problem description**. Brief description of the problem encountered when trying to complete the integration between two solutions. This description may be helpful for understanding the problem and proposing possible corrective measures in the future.
- **Integration with**. It indicates the name of the solution(s) with which the integration was being attempted when the problem was detected.
- **Corrective measures**. In this field, solution providers should briefly explain the actions they carried out to correct the problem or, in case the problem has not been corrected yet, what they would try to do to correct this problem. This can be very useful, as it may help to correct similar problems in the future.

3.1 i4Q^{AD} – Analytics Dashboard

Problem Id.	Status	Problem description	Integration with	Corrective measures
AD-001	Discarded	Dashboard should be compatible with Python libraries to display the visualization.	Generic Pilot	Dashboard does not support Python scripts.

Table 1. Integration problems detected on the i4Q^{AD} Solution

3.2 i4Q^{AI} – AI Models Distribution to the Edge

Problem Id.	Status	Problem description	Integration with	Corrective measures
AI-001	Fixed	Synchronisation with i4Q^{EW}, so that workloads and models co-reside.	i4Q^{EW}	Tight internal and inclusive design.
AI-002	Fixed	Synchronisation with i4Q^{LRT}, so that when new versions of the model are available, they get picked up automatically.	i4Q^{LRT}	Design of interfaces and web hooks so that versions are deployed automatically.
AI-003	Fixed	Gather requirements so we can determine which base technologies to use.	i4Q^{LRT}	Tests of deployment of different kinds of artefacts on different backend components.
AI-004	Work in progress	Ensuring that the edge components are deployable by the software provided.	i4Q^{LRT}	Tight integration tests. Follow the evolving i4Q^{LRT} artefacts so we ensure components are synchronised.

Table 2. Integration problems detected on the i4Q^{AI} Solution

3.3 i4Q^{BC} – Blockchain Traceability of Data

Problem Id.	Status	Problem description	Integration with	Corrective measures
BC-001	Fixed	Failed to start the micro-service mc-management-service due to connection problem with the Message Broker.	Message Broker	Update the message broker relevant configuration. Disable auto-connect to the Message Broker.
BC-002	Fixed	Failed to login new users due to user verification failed with orion-server (Blockchain DB).	Generic Pilot	Ensure the new added users connected with users in the Orion-server (Blockchain DB).

Table 3. Integration problems detected on the i4Q^{BC} Solution

3.4 i4Q^{BDA} – Big Data Analytics Suite

Problem Id.	Status	Problem description	Integration with	Corrective measures
BDA-001	Fixed	New UI in React of i4Q^{DA} Solution was not being launched properly.	i4Q^{DA}	Updated the solution configuration to support the React Frontend.

Table 4. Integration problems detected on the i4Q^{BDA} Solution

3.5 i4Q^{DA} – Services for Data Analytics

Problem Id.	Status	Problem description	Integration with	Corrective measures
DA-001	Fixed	Due to the deployment of a large number of solutions on the server at the same time, it ran out of resources and became unresponsive.	Pilot 4 Solutions	Updated server hardware. Solution deployment by phases.
DA-002	Fixed	Deployment problems due to utilisation of same internal IP ports.	i4Q^{DR}	Deploy the solution modifying the port on the configuration.

Table 5. Integration problems detected on the i4Q^{DA} Solution

3.6 i4Q^{DIT} – Data Integration and Transformation Services

Problem Id.	Status	Problem description	Integration with	Corrective measures
DIT-001	Fixed	i4Q^{DIT} Solution provided more than the required features to some solutions.	i4Q^{IM} and i4Q^{QD}	In each of these cases, the script was updated in order to provide only the required variables to the other solutions.

Table 6. Integration problems detected on the i4Q^{DIT} Solution

3.7 i4Q^{DR} – Data Repository

Problem Id.	Status	Problem description	Integration with	Corrective measures
DR-001	Fixed	The initial format of the generated certificates did not allow the data to be stored in a secured manner.	i4Q^{SH}	After various meetings, the origin of the problem was identified and several modifications were made to the format of the certificates, which made possible the secure storage of the data.

Table 7. Integration problems detected on the i4Q^{DR} Solution

3.8 i4Q^{DT} – Digital Twin Simulation Services

Problem Id.	Status	Problem description	Integration with	Corrective measures
DT-001	Fixed	The first approach in Pilot 4 was to develop a physics-based model of the behaviour of one of the machines. However, this has ended up being non-viable, due to the high complexity of the	Pilot 4 Solutions	Instead of a physics-based model, a data-driven one has been developed, as a quality prediction model is more feasible to develop, and all the required inputs were accessible.

		system and the lack of the required inputs for designing a proper solution.		The physics-based workflow of the i4Q^{DT} Solution is tested and validated in the context of the Generic Pilot, thus exploiting both main functionalities of the solution in the context of the project Pilot phase.
DT-002	Fixed	The model consumes, for an individual quality prediction, all the data from a machine cycle, so a specific algorithm is required for identifying them, based in the signal properties.	Pilot 4, Message broker	Due to the uncertainty regarding which solution should do it, a specific machine cycle detection algorithm has been developed into the i4Q^{DT} Solution.
DT-003	Fixed	The data coming from the machines was not properly prepared for consuming in the i4Q^{DT} Solution for making quality simulations.	Pilot 4, i4Q^{DIT}	The i4Q^{DIT} has overseen the transformation and adaptation of the data to a more usable format for the i4Q^{DT} Solution.
DT-004	Fixed	For simulating the quality prediction model developed in Pilot 4 the project approach for real-time data does not apply, as the model needs at least all the data from a full machine cycle to be correctly executed.	Pilot 4	The i4Q^{DT} Solution only consumes CSV files in the context of the pilot for simulating the quality model.
DT-005	Fixed	As the approach taken for i4Q^{DT} in the context of this pilot has been finally a data-driven one, there is an overlapping in its functionalities with the i4Q^{QD} Solution.	Pilot 4, i4Q^{QD}	It has been decided that the i4Q^{DT} should absorb this task to avoid any overlapping of functionalities with the i4Q^{QD} Solution.
DT-006	Fixed	The FMU models required in the physics-based workflow of the i4Q^{DT} Solution cannot be run inside a Docker container if they have been developed in a Windows environment, thus limiting their applicability in the solution.	i4Q^{DT}, Generic Pilot	A binary source code recompilation function has been developed in the solution, to adapt it if the operating system is not compatible.

Table 8. Integration problems detected on the i4Q^{DT} Solution

3.9 i4Q^{EW} – Edge Workloads Placement and Deployment

Problem Id.	Status	Problem description	Integration with	Corrective measures
EW-001	Fixed	Synchronisation with i4Q^{AI}, so that workloads and models co-reside.	i4Q^{AI}	Tight internal and inclusive design.
EW-002	Fixed	Synchronisation with i4Q^{LRT}, so that when new versions of the workload are available, they get picked up automatically. Workloads are normally heavy and need not change too often.	i4Q^{LRT}	Design of interfaces and web hooks so that versions are deployed automatically.
EW-003	Fixed	Gather requirements so we can determine which base technologies to use.	i4Q^{LRT}	Tests of deployment of different kinds of artefacts on different backend components.
EW-004	Work in progress	Ensuring that the edge components are deployable by the software provided.	i4Q^{LRT}	Tight integration tests. Follow the evolving i4Q^{LRT} artefacts so we ensure components are synchronised.

Table 9. Integration problems detected on the i4Q^{EW} Solution

3.10 i4Q^{IM} – Infrastructure Monitoring

Problem Id.	Status	Problem description	Integration with	Corrective measures
IM-001	Fixed	The data payload produced by the i4Q^{DIT} Solution to the Message Broker contained additional information (features) not utilised by the predictive model of the i4Q^{IM}.	i4Q^{DIT}	After consuming the i4Q^{DIT} data using the Message Broker, the unnecessary data fields are discarded before feeding the predictive model.

Table 10. Integration problems detected on the i4Q^{IM} Solution

3.11 i4Q^{LRT} – Manufacturing Line Reconfiguration Toolkit

Problem Id.	Status	Problem description	Integration with	Corrective measures
LRT-001	Fixed	The “Helm Charts” of the solution must have specific parameters to work correctly with the i4Q^{EW} Solution.	i4Q^{EW}	Appropriate modifications have been made to the code in order to correct this problem.
LRT-002	Fixed	In case of a change of the endpoint address, both MinIO and algorithm, it was necessary to re-deploy the whole solution.	i4Q^{LRT}	A configuration panel has been added within the application to allow these status variables to be changed without having to re-deploy the solution.
LRT-003	Fixed	There was a dependency issue in one of the solutions that caused a cascading failure of the entire frontend when plugins are updated.	i4Q^{LRT}	It has been decided to upgrade the entire frontend to the latest version of NextJS, resulting in cleaner, faster, more optimised and less vulnerable code.
LRT-004	Work in progress	Difficulties connecting directly to Kafka with respect to Kafka’s security certificates.	i4Q^{LRT}	A solution is being developed to facilitate the upgrade for the end user without the need to re-deploy the solution.

Table 11. Integration problems detected on the i4Q^{LRT} Solution

3.12 i4Q^{PA} – Prescriptive Analysis Tools

Problem Id.	Status	Problem description	Integration with	Corrective measures
PA-001	Fixed	If the data to be transferred between the i4Q^{DT} and the i4Q^{PA} is too big, the i4Q^{PA} Solution keeps on with the process without waiting for the feedback from the i4Q^{DT} Solution.	i4Q^{DT}	Data transferred between the i4Q^{DT} and i4Q^{PA} contains an “END” line as an ending flag.
PA-002	Discarded	The i4Q^{PA} cannot prescribe data-driven model’s parameters, as these models work as a black box.	Pilot 4, i4Q^{PA} and i4Q^{DT}	Discarded due to the difficultly of modelling a physics-based model.
PA-003	Fixed	The Docker abstraction, being launched along with other solutions, and having to communicate with other solutions has resulted in communication problems.	Pilot 4 Solutions	The IP and port settings have been split into a configuration file invisible to ordinary users of the solution and a configuration tab in the solution interface where any user can define and save the database and Message Broker, IP and port, among others.

Table 12. Integration problems detected on the i4Q^{PA} Solution

3.13 i4Q^{PQ} – Data-Driven Continuous Process Qualification

Problem Id.	Status	Problem description	Integration with	Corrective measures
PQ-001	Fixed	Protocol for Kafka.	i4Q^{PQ}	Reviewed and removed the unnecessary code from software that was causing protocol issues with Kafka. After completing the code clean-up, connectivity and protocol compliance have been tested again.
PQ-002	Work in progress	Connection to the machine provided by the pilot is not working.	Pilot 6, i4Q^{PQ}	Engaged with the Pilot's IT team to diagnose the connectivity issues. Working on establishing a secure and reliable connection based on their network protocols and requirements.
PQ-003	Fixed	Docker image size too large due to included libraries for forecast and TensorFlow.	i4Q^{PQ}	Replaced heavy libraries with lightweight alternatives without compromising on functionality. In particular, removing TensorFlow as not critical to the application, to reduce the Docker image's size.

Table 13. Integration problems detected on the i4Q^{PQ} Solution

3.14 i4Q^{QD} – Rapid Quality Diagnosis

Problem Id.	Status	Problem description	Integration with	Corrective measures
QD-001	Fixed	The data payload produced by the i4Q^{DIT} Solution to the Message Broker contained additional information (features) not utilised by the predictive model of i4Q^{QD}.	i4Q^{DIT}	After consuming the i4Q^{DIT} data using the Message Broker, the unnecessary data fields are discarded before feeding the predictive model.
QD-002	Work in progress	The i4Q^{LRT} and i4Q^{QD} Solutions lack seamless interaction, leading to a disconnect between identifying a defective product and the necessary reconfiguration.	i4Q^{LRT}	An additional interactable field will be added to embed the i4Q^{LRT} functionality to the i4Q^{QD} Solution dashboard.

Table 14. Integration problems detected on the i4Q^{QD} Solution

3.15 i4Q^{QE} – QualiExplore for Data Quality Factor Knowledge

Problem Id.	Status	Problem description	Integration with	Corrective measures
QE-001	Work in progress	Content problems like hard-to-understand descriptions and a lack of contextual relevance.	i4Q^{QE}	Continuous improvement of the knowledge base.
QE-002	Work in progress	User management does not have a GUI. Users can only be added via configuration changes.	i4Q^{QE}	Integration of a simple administration GUI for regular users.
QE-003	To be fixed (outside of i4Q's scope)	Limited utility of the knowledge base without specific considerations of actual data in question (in the sense of a data analyser).	i4Q^{QE}	Implementation of a data sample upload, an analysis of that sample's quality, and matching with the knowledge base (new features).

Table 15. Integration problems detected on the i4Q^{QE} Solution

3.16 i4Q^{SH} – IIoT Security Handler

Problem Id.	Status	Problem description	Integration with	Corrective measures
SH-001	Fixed	Generation of CAs for each Pilot.	i4Q^{SH}	Generate the CAs using specific information of the pilots.
SH-002	Fixed	Integration of i4Q^{SH} Solution in the complete solution.	i4Q^{SH}	To integrate the solution the CAs are generated, and different solutions use it to perform secure functionalities.
SH-003	Discarded	Implement HSM in the i4Q Pilots.	i4Q^{QE}	It was necessary to perform different changes and insert HSM (hardware).

Table 16. Integration problems detected on the i4Q^{SH} Solution

3.17 i4Q^{TN} – Trusted Networks with Wireless and Wired Industrial Interfaces

Problem Id.	Status	Problem description	Integration with	Corrective measures
TN-001	Work in progress	The duration of the test is limited to the node's batteries, so validations are extended just for a period of business week.	Message Broker	Some nodes will be externally powered to extend the network lifetime, not for all nodes in the network (up to 20 nodes).

Table 17. Integration problems detected on the i4Q^{TN} Solution

3.18 Message Broker

Problem Id.	Status	Problem description	Integration with	Corrective measures
MB-001	Fixed	The size of the data stored by the Message Broker was filling up the server's disk space, causing performance and storage issues.	All i4Q Solutions	The data retention period was properly set to free up disk space after a particular amount of time (2-3 days) passed.

Table 18. Integration problems detected on the Message Broker

4. Conclusions

This document contains all the information related to i4Q Solutions and i4Q Pilots necessary to address the deployment and integration phases of the solution in the infrastructure of the industrial partners.

First, the updated versions of the i4Q Pilot pipelines have been presented, and the modifications introduced in each of them have been explained in a justified manner. Next, the integrations of the solutions that have been implemented during the period between M31 and M40 have been analysed, as well as the information exchanged between the different i4Q Solutions, and the communication mechanism used in each one of the integrations.

Afterwards, the solution providers have explained in detail the problems they have encountered during the deployment and integration phases of their solutions and the actions they have taken to correct these problems.

Finally, the aim was to analyse the modifications made during the last months in terms of solutions validation. However, there have been no significant changes in the behaviour of the i4Q Solutions since the previous deliverable was presented. Therefore, it makes no sense to comment again on the metrics of each solution. For more information about this topic, please refer to section 5 *Testing and Validation* of Deliverable D6.17 – *Continuous Integration and Validation v2*.

References

[1] D. Silveira, S. Santiago and S. Gálvez-Settier. "D6.17 – Continuous Integration and Validation v2," ITI – Instituto Tecnológico de Informática, Paterna, Valencia, Spain, Jun. 2023. [Online]. Available: https://knowledgebiz.sharepoint.com/:b:/r/sites/i4Q/Documentos%20Partilhados/Workpackages/WP06/T6.8/Deliverables/D6.17_Continuous%20Integration%20and%20Validation_v2/Working%20Versions/i4Q_Deliverable_D6.17_v0.8.pdf