D6.17 – Continuous Integration and Validation v2

WP6 – EVALUATE: Piloting and Demonstrating

i4Q has received funding from the European Union's Horizon 2020 research and innovation programme 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	36 months
PROJECT URL	https://www.i4q-project.eu/		
DELIVERABLE	D6.17 – Continuous Integration and Validation v2		
WORK PACKAGE	WP6 – EVALUATE: Piloting and Demonstrating		
DATE OF DELIVERY	CONTRACTUAL 30-Jun-2023	ACTUAL	30-Jun-2023
NATURE	Report	DISSEMINATION LEVEL	Public
LEAD BENEFICIARY	ITI		
RESPONSIBLE AUTHOR	ITI		
CONTRIBUTIONS FROM	BIBA, IBM, ITI, IKER, UNI, TUB, CERTH, KBZ, ENG, EXOS, FIDIA, BIESSE, WHIRPOOL, FACTOR, RIASTONE, FARPLAS		
TARGET AUDIENCE	1) i4Q Project partners; 2) industrial community; 3) other H2020 funded projects; 4) scientific community		
DELIVERABLE CONTEXT/ DEPENDENCIES	This document is the second iteration of D6.9 due by Dec 2022 and has a third iteration in Dec 2023 (D6.18). Its relationship to other documents is as follows: - D6.9 Continuous Integration and Validation. - 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 second release of a set of three, of the Continuous Integration and Validation report. This second release provides a detailed exposition of the first set of steps followed for the deployment and integration of each one of the i4Q Solutions in the Pilots. This task will finish with the third release of the document by the end of the project implementation, and it will include a high-level view of the degree of completion of the integration and validation activities of i4Q Solutions and pilots including software security and quality management.

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
0.1	20-Dec-2022	ToC	1st ToC draft	ITI
0.2	10-Feb-2023	Draft	Version with the deployment infrastructure of the pilots.	ITI, FIDIA, BIESSE, WHIRPOOL, FACTOR, RIASTONE, FARPLAS
0.3	10-Mar-2023	Draft	Version with the solutions configuration.	BIBA, IBM, ITI, IKER, UNI, TUB, CERTH, KBZ, ENG
0.4	21-Apr-2023	Draft	Solutions Integration	BIBA, IBM, ITI, IKER, UNI, TUB, CERTH, KBZ, ENG, EXOS, FIDIA, BIESSE, WHIRPOOL, FACTOR, RIASTONE, FARPLAS
0.5	08-May-2023	Draft	Testing and Validation Analysis of results	BIBA, IBM, ITI, IKER, UNI, TUB, CERTH, KBZ, ENG, EXOS, FIDIA, BIESSE, WHIRPOOL, FACTOR, RIASTONE, FARPLAS
0.6	30-May-2023	Draft	Version ready for internal review	ITI
0.7	13-Jun-2023	Draft	Comments from reviewers received	TUB, ENG
0.8	20-Jun-2023	Draft	Draft version with modifications ready for final review	ITI
1.0	30-Jun-2023	Final Doc	Quality check and issue of final document	CERTH

Disclaimer

Any dissemination of results reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Copyright message

© i4Q Consortium, 2021
This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS/ACRONYMS

AI	Artificial Intelligence
ACM	Advanced Cluster Management
AWS	Amazon Web Services
CA	Certificate Authority
CNC	Computer Numerical Control
CPU	Central Processing Unit
CSV	Comma-separated values
FMU	Functional Mockup Unit
GB	Gigabytes
GHz	Gigahertz
GKE	Google Kubernetes Engine
HDD	Hard disk drive
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
IWSN	Industrial Wireless Sensor Network
KPI	Key Performance Indicator
MB	Message Broker

ML	Machine Learning
N/A	Not Available
OCM	Open Cluster Management
OS	Operating System
QC	Quality Control
RAM	Random Access Memory
RIDS	Reliable Industrial Data Services
SSD	Solid-state drive
SW	Software
TB	Terabytes
UPV	Universitat Politècnica de València

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, 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 first set of steps followed for the deployment and integration of each one of the i4Q Solutions in the Pilots. Note that a final deliverable is foreseen in Month 36. This final deliverable will include any detail not mentioned by Month 30.

D6.17 is including:

- Information on the deployment infrastructure of each one of the i4Q Pilots (use case, type of infrastructure, operating system, storage size, CPU information, RAM size and any other relevant aspects to be considered).
- Information on the configuration and deployment processes for each of the i4Q Solutions.
- The analysis of the different solutions integration that are part of the pipeline of the same pilot.
- The analysis of the results obtained from testing and validating the performance of the different solutions using the SonarQube tool.
- Analysis of the deployment and integration status of the different solutions in each of the pilots.

Document structure

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

Section 2: Deployment Infrastructure. Discusses the main features of the deployment infrastructure for each one of the i4Q Pilots. These include the operating system, storage size, CPU, RAM, and other important details to consider.

Section 3: Solutions Configuration. Contains information on the configuration and deployment processed, as well as the system requirements and software dependencies needed to deploy each i4Q Solution correctly on the corresponding pilot infrastructure.

Section 4: Solutions Integration. Reviews the integration between the different solutions that are part of the pipeline of the same pilot, as well as the information they exchange, and the communication mechanism used.

Section 5: Testing and Validation. Quality metrics of the source code of each i4Q Solution are discussed in this section, based on the reports generated by the SonarQube tool.

Section 6: Analysis of results. Analyses the deployment and integration status of the different solutions in each of the pilots by using a colour-coded matrix.

Section 7: Conclusions. 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.
- and WP5 Rapid Manufacturing Line Qualification and Reconfiguration.

A total of 22 i4Q Solutions will be 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 will allow to detect functional and/or integration problems at an early stage. Since this feedback can be given to the development tasks, those problems can be addressed before the official release of i4Q Solutions. Consequently, this task reduces the risk of facing those problems after the project finishes which, at the end of the day, could be expensive to solve in the future. Therefore, this task contributes to improve the solution's quality and reliability and makes them more likely to be used in real industrial scenarios.

In addition, modifications to the functionality of the use cases can be proposed, as long as they result in an improvement of the solutions.

This deliverable addresses the first phase of the deployment and integration of the i4Q Solutions in the i4Q Pilots infrastructure. For this purpose, a general review of the pilots' deployment infrastructure and the characteristics of the implemented solutions is carried out.

In addition to this information, the pipelines of the different pilots and the most important modifications they have experienced during the period from M24 to M30 are explained. With this information, the appropriate modifications are made and, afterwards, the integration between the different i4Q Solutions is analysed, as well as the information they send or receive, and the communication mechanism used.

Another priority of this task is to ensure that the i4Q Solutions developed meet a quality level. To this end, the quality metrics of the source code of each i4Q Solution are examined and, with the results obtained, the solution providers are proposed to make the appropriate adjustments to improve the software quality.

Finally, the deployment and integration of the different i4Q Solutions in the industrial partners' infrastructure is analysed. This data provides a picture of the status of the solutions and shows whether the integration and deployment phases are going as planned.

Note that the deployment of the i4Q Solutions is expected to be completed by the end of M30. Therefore, in most of the i4Q Pilots the integration between the different solutions has only just started. The results of this phase as well as possible related modifications will be presented in the subsequent release of this document, in M36.

2 Deployment Infrastructure

The objective of this section is to collect information on the deployment infrastructure of each one of the i4Q Pilots. For this purpose, pilot leaders have been asked to complete the corresponding table with the information explained below.

- **Use case**. Indicates the different KPIs that the pilot infrastructure intends to measure for each one of the defined business processes.
- **Type of infrastructure**. Explains whether the type of infrastructure deployed is on a local machine, on a private or on-premises cloud, or on a cloud service provider such as Amazon Web Services (AWS) or Microsoft Azure.
- **Operating system**. Here the operating system and version installed on the infrastructure is indicated.
- **Storage size**. Gigabytes (GB) of disk storage available for storing solutions data.
- **CPU information**. Contains information related to the processing unit installed in the infrastructure such as: model name, number of cores, number of threads, and base frequency of execution.
- **Quantity of RAM**. GB of RAM contained in the infrastructure.
- **Other details**. Indicate other relevant aspects to take into account that have not been mentioned above.

Remark: In case the infrastructure varies for different use cases, this table can be repeated as many times as necessary.

Pilot 1: Smart Quality in CNC Machining	
Name	**Description**
Use case	**P1_BP01:** Ensure final surface quality. **P1_BP02:** Chatter detection and avoidance. **P1_BP03:** Evaluation of machine tool condition.
Type of infrastructure	Local machine
Operating system	Linux (Ubuntu 22.04)
Storage size	512 GB SSD
CPU information	Model name: Intel® Xeon® W-2125 Number of CPU core(s): 16 Number of thread(s): 32 Basic frequency: 4.00 GHz
Quantity of RAM	32 GB
Other details	-

Table 1. Pilot 1 deployment infrastructure

Pilot 2: Diagnostics and IoT Services	
Name	**Description**
Use case	**P2_BP01:** Diagnostic of axis movement and torque monitoring. **P2_BP02:** Electrospindle Monitoring.
Type of infrastructure	Local Machine
Operating system	Windows 10 Enterprise LTSC
Storage size	256 GB
CPU information	Model name: Intel® Core™ i5-8500 Number of CPU core(s): 6 Number of thread(s): 6 Basic frequency: 3.00 GHz
Quantity of RAM	32 GB
Other details	The machine is shared with the CNC and dedicates 1 core to real-time tasks.

Table 2. Pilot 2 deployment infrastructure

Pilot 3: White Goods Product Quality	
Name	**Description**
Use case	**P3_BP01:** Full production product conformity automatic assessment.
Type of infrastructure	The infrastructure is stored on a server in CERTH's premises.
Operating system	Linux Server (Ubuntu 20.04)
Storage size	5 TB (1TB SSD + 4TB HDD)
CPU information	Model name: Intel® Core™ i9-10920X Number of CPU core(s): 12 Number of thread(s): 24 Basic frequency: 3.50 GHz
Quantity of RAM	128 GB
Other details	-

Table 3. Pilot 3 deployment infrastructure

Pilot 4: Aeronautics and Aerospace Metal Parts Quality	
Name	**Description**
Use case	**P4_BP01:** In-line product quality control. **P4_BP02:** Automatic online correction of the CNC machining process.
Type of infrastructure	Local Machine
Operating system	Windows 10 Pro 64-bits
Storage size	768 GB
CPU information	Model name: N/A Number of CPU core(s): 4 Number of thread(s): N/A Basic frequency: 3.50 GHz
Quantity of RAM	64 GB
Other details	In this server there are some data and software running that are needed for the daily activity of Factor. It is in the virtual machine VIR-007, please do not touch it.

Table 4. Pilot 4 deployment infrastructure

Pilot 5: Advanced In-line Inspection for Incoming Prime Matter Quality Control	
Name	**Description**
Use case	P5_BP01 – Data collection and analysis for raw matter quality control. P5_BP02 – Final product QC causal relation analysis.
Type of infrastructure	Local Machine
Operating system	Windows 10 Pro
Storage size	500GB
CPU information	Model name: N/A Number of CPU core(s): 4 Number of thread(s): N/A Basic frequency: 1.60 GHz
Quantity of RAM	8GB
Other details	This is a temporary server that will be used to accommodate the i4Q Solutions. It is envisaged that a better machine will be available for the second round of validation

Table 5. Pilot 5 deployment infrastructure

Pilot 6: Advanced Inspection of Automotive Plastic Parts	
Name	**Description**
Use case	**P6_BP01:** Autonomous parameter optimization for the injection process. **P6_BP02:** Automatic Quality Inspection.
Type of infrastructure	Local machine
Operating system	Ubuntu Server
Storage size	1 TB
CPU information	Model name: Intel® Xeon® Gold 6230 Number of CPU core(s): 16 Number of thread(s): 32 Basic frequency: 2.10 GHz
Quantity of RAM	64 GB
Other details	Two other similar servers can also be used as needed.

Table 6. Pilot 6 deployment infrastructure

3 Solutions Configuration

The objective of this section is to gather information on the configuration and deployment processes for each of the i4Q Solutions. For this purpose, solution providers have been asked to complete the corresponding table with the information explained below.

- **Type of infrastructure**. Brief explanation of the infrastructure type to be deployed. This can be a Docker image, source code managed from GitLab or a private repository, a dedicated server (physical or virtual) where the solution Will be deployed, etc.
- **Description**. Brief explanation of the solution and the functions it performs.
- **Location**. Indicate the environment in which the solution is intended to be deployed. This can be: a local environment, a cloud server, a server hosted in the pilot environment, the environment provided by an industrial partner, etc.
- **Storage size**. GB of disk space expected to be occupied by the solution (this value may vary).
- **SW tech and dependencies**. Software programs, packages, or libraries that must be installed in the pilot environment to be able to run the solution without problems.
- **System requirements**. Minimum hardware requirements that the pilot environment must fulfil in order to run the solution efficiently.

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

Feature	Explanation
Type of infrastructure	Virtual server on BIBA’s infrastructure

Description	i4Q^{QE} Solution is a virtual server with Docker environment to run the solution stack.
Location	BIBA's environment
Storage size	~100 GB
SW tech and dependencies	▪ Ubuntu OS ▪ Docker Engine v20.10.21 ▪ Docker Compose v2.12.2
System requirements	Docker requirements: ▪ 16 GB RAM ▪ 64-bit operating system ▪ Hardware virtualisation support

Table 7. i4Q^{QE} Solution configuration

The configuration of the i4Q^{QE} Solution by pilot is as follows.

- BIBA operates a remote service on their infrastructure accessible via web browser. No pilot-specific deployments are planned.

3.2 i4Q^{BC} – Blockchain Traceability of Data

Feature	Explanation
Type of infrastructure	Source code is handled via a GitLab repository. Also Includes deployment entities (Docker container).
Description	i4Q^{BC} is a solution that provides blockchain-based properties on top of a standard database, using cryptographic capabilities.
Location	On a server hosted by the pilot. Cloud-based deployment is also supported.
Storage size	The Docker image is less than 0.5 GB. The required storage is proportional to the number of users and the number of elements that are interacted with in the pilot.
SW tech and dependencies	▪ Linux OS ▪ Docker Engine v20.10.21 ▪ Go Lang compiler v1.19
System requirements	Docker requirements: ▪ 16 GB RAM ▪ 64-bit operating system ▪ Hardware virtualisation support

Table 8. i4Q^{BC} Solution configuration

At the moment, it seems that this solution shall be used in the generic pilot, within a demonstration for capabilities of securely storage of machine configuration data and changes thereof. Deployment should take place at the pilot plant server.

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

Feature	Explanation
Type of infrastructure	Preinstalled source code at IWSN gateway and industrial wireless nodes.
Description	i4Q^{TN} Solution is a mesh industrial wireless software defined network data generation converging towards i4Q Kafka broker through the IWSN gateway.
Location	Plant floor
Storage size	Not applicable
SW tech and dependencies	Not applicable
System requirements	Network connection to i4Q broker deployed at pilot virtualized server.

Table 9. i4Q^{TN} Solution configuration

The configuration of the TN by pilot is as follows:

- In Pilot 4, the IWSN subcomponent of the i4Q^{TN} Solution will be deployed, including the IWSN gateway and at least one IWSN node, depending on pilot/digitalization requirements during the deployment of the system. One or more industrial sensors will be acquired by the IWSN node.

3.4 i4Q^{SH} – IIoT Security Handler

Feature	Explanation
Type of infrastructure	Source code is managed through a GitLab repository.
Description	i4Q^{SH} is a solution that provides the CA for other solutions, so that they can use it to generate the certificates to perform secure operations.
Location	Local environment
Storage size	The size of the code is 750 bytes (the total size would depend on how it is deployed).
SW tech and dependencies	▪ OpenSSL, any recent version
System requirements	▪ 4 GB RAM ▪ 64-bit Linux operating system (tests performed in Ubuntu).

Table 10. i4Q^{SH} Solution configuration

This solution will be used in the same way in all the pilots and will be responsible for providing the CA to the other solutions, so that they can use it to generate the certificates they need to perform secure operations.

3.5 i4Q^{DR} – Data Repository

Feature	Explanation
Type of infrastructure	Source code as provided in the GitLab repository. Includes a deployer script that launches several Docker containers.
Description	i4Q^{DR} Solution is a tool aimed at providing, in a centralised fashion, the functionality related to the storage of data in the whole i4Q system.
Location	Computational infrastructure provided by industrial partners at their premises.
Storage size	Source code size < 150 MB Storage size necessary after deployment will depend on the selected scenarios.
SW tech and dependencies	▪ Ubuntu OS preferred (not tested in Windows). ▪ Bash, to run the scripts. ▪ Docker Engine and Docker Compose. Recent version accepting version 3.9 Docker Compose YAML files. ▪ OpenSSL, any recent version. ▪ curl ▪ jq, any recent version.
System requirements	Docker requirements: ▪ 4 GB RAM ▪ 64-bit operating system. ▪ Hardware virtualisation support

Table 11. i4Q^{DR} Solution configuration

The configuration of the i4Q^{DR} Solution by pilot is as follows.

- In Pilot 1, i4Q^{DR} deploys MongoDB for the Single Server scenario with Security, and MinIO for the Single Server scenario. MongoDB instance is being used to store manufacturing data. The MinIO instance is being used to store models generated by the i4Q^{LRT} Solution.
- In Pilot 2, i4Q^{DR} deploys PostgreSQL instance for the Single Server scenario. One database is being used to store data to be visualized by the i4Q^{AD} Solution.
- In Pilot 3, i4Q^{DR} deploys PostgreSQL instance for the Single Server scenario. One database is being used to store offline data in batch offered by Whirlpool, as well as analytics results, to be visualized by the i4Q^{AD} Solution.
- In Pilot 4, i4Q^{DR} deploys MongoDB for the Single Server scenario with Security, and MinIO for the Single Server scenario. The MongoDB instance is being used to store the following data:
    - Information generated by the MTLINK instance deployed by FACTOR at its premises, which obtains data generated by CNC machines. This data is read by the i4Q^{LRT} Solution.

    - Data provided by a CSV file manually generated by FACTOR employees, which includes information regarding quality aspects of the production process. This information is read by the i4Q^{DIT} Solution.
    - The MinIO instance will be used to store the models generated by the i4Q^{LRT} Solution.
- In Pilot 5, i4Q^{DR} deploys PostgreSQL for the Single Server scenario.
- In Pilot 6, i4Q^{DR} deploys MongoDB for the Single Server scenario with Security and MinIO for the Single Server scenario. In the one hand, the MongoDB instance is being used to store manufacturing data provided by FARPLAS, which is read by the i4Q^{QD}, i4Q^{PQ} and the i4Q^{LRT} Solutions. On the other hand, the MinIO instance is being used to store models generated by the i4Q^{PA} and i4Q^{LRT} Solutions.

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

Feature	Explanation
Type of infrastructure	Docker images will be available on GitLab along with a token that will allow the use of these images.
Description	i4Q^{DIT} Solution is a distributed server-based solution able to prepare manufacturing data for being efficiently processed by other analytical solutions.
Location	Local environment
Storage size	Image occupies around 1.5GB. The actual storage size required after implementation will depend on the selected scenarios and the amount of data used in the pilots.
SW tech and dependencies	Git and Docker.
System requirements	Minimum requirements for Docker. The performance of the algorithms will improve with more RAM or processing speed.

Table 12. i4Q^{DIT} Solution configuration

The configuration of the i4Q^{DIT} Solution by pilot is as follows.

- In Pilot 1, the i4Q^{DIT} Solution pre-processes the data and extracts features.
- In Pilot 2, the i4Q^{DIT} Solution transforms the data.
- In Pilot 3, the i4Q^{DIT} Solution performs preprocessing and fusion of different datasets and transform them for further use by other solutions.
- In Pilot4, the i4Q^{DIT} Solution fuses different datasets and provides a new enriched dataset.
- In Pilot 5, the configuration and deployment of the i4Q^{DIT} Solution has not started yet. Further details will be provided once the deployment on the RIASTONE infrastructure is completed.
- In Pilot 6, configuration and deployment of the i4Q^{DIT} Solution has not started yet. Further details will be provided once the deployment on the FARPLAS infrastructure is completed.

3.7 i4Q^{DA} – Services for Data Analytics

Feature	Explanation
Type of infrastructure	Server at UNINOVA premises. Source code available in the i4Q GitLab for local deployments.
Description	The i4Q^{DA} Solution is a Data Analytics experimentation environment that allows the creation and management of data analytics workflows in an intuitive, code-free manner.
Location	UNINOVA and local environment is some pilots[1]
Storage size	Minimum 16 GB of free space. Storage size necessary after deployment will depend on the selected scenarios
SW tech and dependencies	Docker Engine and Docker Compose.
System requirements	▪ 16 GB RAM ▪ Processor: Intel® Core™ i5-6500 or greater / AMD Ryzen 5 3600 or greater

Table 13. i4Q^{DA} Solution configuration

The configuration of the i4Q^{DA} Solution by pilot is as follows.

- In Pilot 2, the i4Q^{DA} Solution is being used to connect to different data sources and pre-process data before analytics models can be applied.
- In Pilot 3, the i4Q^{DA} Solution is being used to evaluate analytics models on given data.
- In Pilot 4, the i4Q^{DA} Solution is planned but still not implemented.

[1] In BIESSE and WHIRLPOOL pilots, the i4Q^{DA} Solution has been installed in their local environment.

- In Pilot 5, the i4Q^{DA} is being used to connect to production databases as well as data coming in real-time from a spectrometer for in-line raw matter quality control. This data is continuously being analysed to check if the raw matter meets the necessary standards.

3.8 i4Q^{BDA} – Big Data Analytics Suite

Feature	Explanation
Type of infrastructure	Server at UNINOVA premises. Source code available in the i4Q GitLab for local deployments.
Description	The i4Q^{BDA} Solution is a tool that enables the encapsulation of the Services for Data Analytics solution in a ready-to-deploy software bundle that can be deployed in any hardware infrastructure, whether it is a on-premises server or a cloud instance.
Location	UNINOVA
Storage size	Minimum 16 GB of free space. Storage size necessary after deployment will depend on the selected scenarios.
SW tech and dependencies	Docker Engine and Docker Compose.
System requirements	▪ 16 GB RAM ▪ Processor: Intel® Core™ i5-6500 or greater / AMD Ryzen 5 3600 or greater

Table 14. i4Q^{BDA} Solution configuration

The i4Q^{BDA} Solution was used for the pilots where the i4Q^{DA} Solution was deployed to create an optimized software bundle to the specific hardware infrastructure available at each pilot. This solution is deployed in the same way for all pilots and there are no plans for a pilot-specific deployment.

3.9 i4Q^{AD} – Analytics Dashboard

Feature	Explanation
Type of infrastructure	Source code as provided in the GitLab repository. Includes a deployer script that launches Docker container.
Description	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.
Location	Local environment
Storage size	Source code size < 100 MB. Storage size necessary after deployment will depend on the selected scenarios.
SW tech and dependencies	▪ Bash, to run the scripts. ▪ Docker Engine and Docker Compose. Recent version accepting version 3.9 Docker Compose YAML files.

	▪ Access to i4Q GitLab registry
System requirements	Docker requirements: ▪ 16 GB RAM ▪ 4 CPU ▪ 64-bit operating system ▪ Hardware virtualisation support

Table 15. i4Q^{AD} Solution configuration

The configuration of the i4Q^{AD} Solution by pilot is as follows.

- In Pilot 1, the i4Q^{AD} Solution is used to create dashboards for monitoring the KPIs related to the project.
- In Pilot 2, the i4Q^{AD} Solution is used to create dashboards for monitoring the KPIs related to the project.
- In Pilot 3, the i4Q^{AD} Solution is used to create dashboards for monitoring the KPIs related to the project.
- In Pilot 4, the i4Q^{AD} Solution is planned but still not implemented.
- In Pilot 5, the i4Q^{AD} Solution is used as the main visualisation and dashboarding solution to show data coming from all the production processes, from raw matter quality to final product quality.
- In Pilot 6, the i4Q^{AD} Solution is used to create dashboards for monitoring the KPIs related to the project.

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

Feature	Explanation
Type of infrastructure	Extensions to ACM / OCM available from GitLab
Description	The i4Q^{AI} Solution is a tool that enables the Artificial Intelligence models lifecycle management
Location	Cloud / edge / factory floor
Storage size	Around 1 GB
SW tech and dependencies	▪ Linux ▪ Kubernetes ▪ Access to i4Q GitLab registry ▪ Hardware virtualisation support
System requirements	16 GB RAM

Table 16. i4Q^{AI} Solution configuration

The configuration of the i4Q^{AI} Solution by pilot is as follows.

- In Pilot 4, the i4Q^{AI} Solution deploys and handles the lifecycle management of models produced by the i4Q^{LRT} Solution.

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

Feature	Explanation
Type of infrastructure	Extensions to ACM / OCM available from GitLab
Description	The i4Q^{EW} Solution is a tool that enables the lifecycle management of Artificial Intelligence workloads.
Location	Cloud / edge / factory floor
Storage size	Around 1 GB
SW tech and dependencies	▪ Linux ▪ Kubernetes ▪ Access to i4Q GitLab registry ▪ Hardware virtualisation support
System requirements	16 GB RAM

Table 17. i4Q^{EW} Solution configuration

The configuration of the i4Q^{EW} Solution by pilot is as follows.

- In Pilot 4, the i4Q^{EW} Solution deploys and handles the lifecycle of Artificial Intelligence (AI) workloads produced by the i4Q^{LRT} Solution.

3.12 i4Q^{IM} – Infrastructure Monitoring

Feature	Explanation
Type of infrastructure	Docker images will be available on GitLab along with a token that will allow the use of these images.
Description	The i4Q^{IM} Solution consists of a collection of monitoring tools and predictive failure alerting methods, with the goal of providing efficient alerts when a machine problem is detected. It uses Machine Learning algorithms to track and analyse industrial sensor signals, as well as monitor production lines and operations.
Location	Local infrastructure
Storage size	Images occupy roughly 1.5 GB. The storage size requirements after implementation will depend on the selected pilot scenario.
SW tech and dependencies	▪ Git ▪ Docker and Docker Compose
System requirements	Minimum requirements for Docker. The execution time of the algorithms may benefit from additional RAM or processing speed.

Table 18. i4Q^{IM} Solution configuration

The configuration of the i4Q^{IM} Solution by pilot is as follows.

- In Pilot 1, the i4Q^{IM} Solution deploys a Machine Learning (ML) based service for the detection of degraded CNC machine components and the provision of warning messages.
- In Pilot 2, the i4Q^{IM} Solution deploys a ML based service for the detection of CNC tool wear and the provision of warning messages.
- In Pilot 3, the i4Q^{IM} Solution deploys a ML based service for the detection of faulty products and the provision of warning messages.
- In Pilot 4, the i4Q^{IM} Solution deploys a ML based service for the detection of CNC machine malfunctions and the provision of warning messages.

3.13 i4Q^{DT} – Digital Twin Simulation Services

Feature	Explanation
Type of infrastructure	Docker image and source code available to download on GitLab.
Description	The i4Q^{DT} Solution is capable of running simulations both from Physics-based models (co-simulated from models built from FMU files) and Data-driven models (trained from ML models). It allows the user to build or train the models with the help of a friendly interface.
Location	Local infrastructure
Storage size	Images occupy almost 10 GB. The size of storage required after implementation will depend on the scenarios selected and the amount of data used by the pilots.
SW tech and dependencies	▪ Git ▪ Docker and Docker Compose
System requirements	Minimum requirements for Docker. The more RAM or processing speed the faster the simulations will be carried out.

Table 19. i4Q^{DT} Solution configuration

The configuration of the i4Q^{DT} Solution by pilot is as follows.

- In Pilot 4, the i4Q^{DT} Solution is deployed via Docker Compose in FACTOR's system. The created models are stored using the i4Q^{DR} Solution and then are exploited by the i4Q^{PA} Solution.

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

Feature	Explanation
Type of infrastructure	Docker images available to download on GitLab.
Description	The i4Q^{PQ} Solution is capable of checking ongoing manufacturing processes and provides interpretable KPIs and visualizations of the analytical output to determine if an intervention is necessary.
Location	Local infrastructure
Storage size	Images occupy roughly 1 GB
SW tech and dependencies	▪ Git ▪ Docker and Docker Compose
System requirements	Minimum requirements for Docker. The execution time of the algorithms may benefit from additional RAM or processing speed.

Table 20. i4Q^{PQ} Solution configuration

The i4Q^{PQ} Solution is deployed in the same way for all the pilots in which it is involved, so it does not require a specific configuration.

3.15 i4Q^{QD} – Rapid Quality Diagnosis

Feature	Explanation
Type of infrastructure	Docker images will be available from GitLab along with a token that will allow the use of these images.
Description	The i4Q^{QD} Solution is a microservice that seeks to offer an efficient, rapid diagnosis on product quality conformity and manufacturing process conditions. It employs state-of-the-art ML algorithms to industrial sensor signals to improve the quality of the end product.
Location	Local infrastructure
Storage size	Images occupy roughly 1.5 GB. The storage size requirements after implementation will depend on the selected pilot scenario.
SW tech and dependencies	▪ Git ▪ Docker and Docker Compose
System requirements	Minimum requirements for Docker. The execution time of the algorithms may benefit from additional RAM or processing speed.

Table 21. i4Q^{QD} Solution configuration

The configuration of the i4Q^{QD} Solution by pilot is as follows.

- In Pilot 1, the i4Q^{QD} Solution deploys a ML based service for the detection of machine chatter presence.
- In Pilot 4, the i4Q^{QD} Solution deploys a ML based service for the detection of faulty products in CNC machining.
- In Pilot 6, the i4Q^{QD} Solution deploys a ML based service for the detection of defective products in injection molding machines.

3.16 i4Q^{PA} – Prescriptive Analysis Tools

Feature	Explanation
Type of infrastructure	This solution can be deployed on both Docker and Kubernetes. The images will be available from GitLab, and a token will be provided to use them.
Description	The i4Q^{PA} Solution is a simulation, evaluation and optimization tool that prescribes the best configuration/parametrization of a model according to an evaluation function.
Location	Local environment
Storage size	Images occupy 2.5 GB approximately. The size of storage required after implementation will depend on the models selected and the number of simulations defined by the pilots.
SW tech and dependencies	▪ Git ▪ Docker and Docker Compose. ▪ K3s (If you choose to deploy on Kubernetes) ▪ Helm (If you choose to deploy on Kubernetes)
System requirements	Minimum requirements for Docker or Kubernetes. The more RAM or processing speed the faster the algorithms can go.

Table 22. i4Q^{PA} Solution configuration

The configuration of the i4Q^{PA} Solution by pilot is as follows.

- In Pilot 4, the i4Q^{PA} Solution deploys a MinIO instance, the buckets “models” and “i4qpa” are created, the models to be prescribed are loaded to the “models” bucket and the i4Q^{PA} Solution deploys a prescription service in which a model can be analysed to optimise a system according to an evaluation function.

3.17 i4Q^{LRT} – Manufacturing Line Reconfiguration Toolkit

Feature	Explanation
Type of infrastructure	This solution can be deployed on both Docker and Kubernetes. The images will be available from GitLab, and a token will be provided to use them.
Description	The i4Q^{LRT} Solution is a collection of optimization microservices that use simulation to evaluate different possible scenarios and propose changes in manufacturing line configuration parameters to achieve improved quality objectives.
Location	Local infrastructure
Storage size	Images occupy less than 1 GB. The size of storage required after implementation will depend on the scenarios selected and the amount of data used by the pilots.
SW tech and dependencies	▪ Git ▪ Docker and Docker Compose. ▪ K3s (If you choose to deploy on Kubernetes) ▪ Helm (If you choose to deploy on Kubernetes)
System requirements	Minimum requirements for Docker or Kubernetes. The more RAM or processing speed the faster the algorithms can go.

Table 23. i4Q^{LRT} Solution configuration

The configuration of the i4Q^{LRT} Solution by pilot is as follows.

- For Pilots 1, 4 and 6, the i4Q^{LRT} Solution implements a ML model for the prediction of possible errors.

3.18 Message Broker

Feature	Explanation
Type of infrastructure	This component can be deployed using Docker. The images will be available from GitLab, and a token will be provided to use them.
Description	The i4Q Message Broker (MB) is a component that provides a quick and safe way of inter-solution communication via data streaming.
Location	Local instance
Storage size	Images occupy roughly 6 GB.
SW tech and dependencies	▪ Git ▪ Docker and Docker Compose
System requirements	Minimum requirements for Docker. The performance of the message bus may benefit from additional RAM or processing speed.

Table 24. Message Broker configuration

The configuration of the MB for every pilot is as follows.

- A Kafka message bus will be deployed allowing the different i4Q Solutions to exchange messages in a secure manner.

4 Solutions Integration

This section analyses the integration of the different solutions that are part of the pipeline of the same pilot. To do so, the updated version of each pilot's pipeline is shown, and the most important modifications that have been made since the previous version are explained in a justified manner. Then, the interaction between the different solutions, the information they exchange, and the communication mechanism they use are explained.

4.1 Pilot 1: Smart Quality in CNC Machining

In this subsection, the pipeline of Pilot 1 *Smart Quality in CNC Machining* is analysed, and the most important modifications made from the time it was defined to the present version are explained in a justified manner.

After checking it with the pilot and the different solution providers, the updated version of the pipeline is presented in the following image.

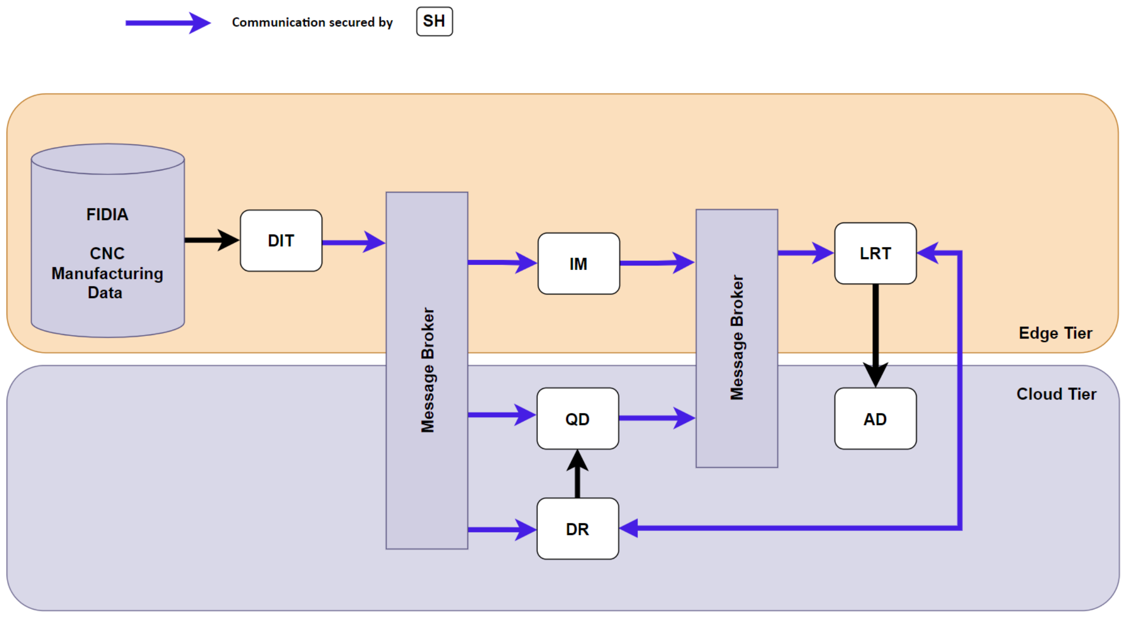


Figure 1. Pilot 1 pipeline diagram

Since the pipeline was defined, the main modifications made are as follows.

- **The i4Q^{PQ} Solution has been replaced by the i4Q^{AD}**. After discussion with the partners involved in the Pilot deployment, we realised that the requirements were better met by using the i4Q^{AD}, so it has been agreed that this will be the solution shown, and not the i4Q^{PQ} as indicated in previous versions of the pipeline.

- **The i4Q^{QD} Solution no longer uses the i4Q^{AD} Solution to display the data**. Since the i4Q^{QD} Solution implements its own user interface, it has been decided to discard the use of the i4Q^{AD} Solution, as both would perform the same functionality.

- **Modification in the representation of the Security Handler solution** (**i4Q^{SH}**). In the past, the use of certificates signed with the Certificate Authority generated by the Security Handler were represented using a box with the initials "SH". This representation has been replaced by purple arrows connecting those solutions that use the i4Q^{SH} Solution to secure communications by using certificates. In this way, it is clearer to see which communications are secured and which are not.

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

- **Input solution**. This solution receives data directly from the pilot's infrastructure, so it does not have an input solution that provides the data.
- **Output solution**. The data processed by this solution is sent to the i4Q^{DR}, i4Q^{IM}, and i4Q^{QD} Solutions.
- **Information received**. This solution receives the sensor readings from the CNC machines which are installed in the pilot's infrastructure.
- **Information sent**. This solution sends clean, preprocessed, and feature enriched datasets to the solutions mentioned above.
- **Communication mechanism**. Message Broker is used as a communication mechanism to send information to the solutions mentioned above via *DIT_topic1* and *DIT_topic2*.

4.1.2 i4Q^{SH} – IIoT Security Handler

In the case of the i4Q^{SH} Solution, there is not an information exchange, as it is not responsible for sending or receiving data from other solutions. Instead, it provides the CAs to the Message Broker and the i4Q^{DR} Solution, so that they can generate the necessary certificates, being this the same procedure to be followed in all the pilots.

This CA will be responsible for guaranteeing the authenticity, integrity, and reliability of the digital certificates used in communications. In this way, communications will be secure, and the certificates generated in the solutions will have authenticity and integrity assured.

4.1.3 i4Q^{IM} – Infrastructure Monitoring

- **Input solution**. This solution receives data from i4Q^{DIT} Solution.
- **Output solution**. The data processed by this solution is sent to the i4Q^{LRT} Solution.
- **Information received**. This solution receives an enriched dataset with features for the development and training of a Machine Learning model. Specifically, it contains information about the component degradation of Fidia's CNC machine.
- **Information sent**. This solution sends alerts for problem detection on the CNC machine.
- **Communication mechanism**. Message Broker is used as a communication mechanism to provide the i4Q^{LRT} Solution with the generated alert via the *IM_topic*. This alert is a JSON message that indicates the instances where a problem has been detected.

4.1.4 i4Q^{QD} – Rapid Quality Diagnosis

- **Input solution**. This solution receives data from the i4Q^{DR} and i4Q^{DIT} Solutions.
- **Output solution**. The processed data is not sent to any solution.
- **Information received**. This solution receives an enriched dataset with features for the development and training of a Machine Learning model. Specifically, it contains information about the quality conformity of the machined products.
- **Information sent**. This solution sends alerts for the detection of faulty machined products.
- **Communication mechanism**. Although the information is not sent to any specific solution, Message Broker is used as a communication mechanism to provide the generated alert via the *QD_topic* for potential use. This alert is a JSON message that indicates the instances where a faulty product has been detected.

4.1.5 i4Q^{DR} – Data Repository

In Pilot 1, besides the use of the CA generated by the i4Q^{SH} Solution to deploy the "with security" scenarios, the i4Q^{DR} Solution integrates with two solutions: i4Q^{DIT} and i4Q^{LRT}. In the last case, the integration is bi-directional. For readability purposes, further details are provided for each integration separately.

4.1.5.1 Inbound integration with i4Q^{DIT}

- **Input solution.** This solution receives data from the i4Q^{DIT} Solution.
- **Information received.** Clean, preprocessed, and feature enriched datasets produced by the i4Q^{DIT} Solution. This data is stored in a database created in a MongoDB instance deployed with the i4Q^{DR} Solution.
- **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^{DIT} in topic DIT_topic3[2]. At this stage, the concrete format and content of the exchanged information has not been defined, as it depends on the training algorithm to be used by i4Q^{LRT}.

4.1.5.2 Outbound integration with i4Q^{LRT}

- **Output solution.** The processed data is sent to the i4Q^{LRT} Solution.
- **Information sent.** Processed manufacturing data that will be used by the i4Q^{LRT} Solution to train the ML model it generates.
- **Communication mechanism.** This integration does not involve the MB. Instead, i4Q^{LRT} will connect to a database created in the MongoDB instance deployed by i4Q^{DR} to retrieve the necessary information. The concrete information to be retrieved by i4Q^{LRT} is not defined at this stage as the implementation of the ML model is currently in progress.

[2] This name of this topic is expected to be replaced by a more intuitive one.

4.1.5.3 *Inbound integration with i4Q^{LRT}*

- **Input solution.** The i4Q^{DR} Solution receives the data processed by i4Q^{LRT} in order to store it in a database.
- **Information received.** A file corresponding to the ML model generated by i4Q^{LRT}.
- **Communication mechanism.** The integration among both solutions does not involve the MB. The i4Q^{LRT} Solution will connect to the MinIO instance deployed by i4Q^{DR} to store the file in a bucket created for this purpose.

4.1.6 i4Q^{LRT} – Manufacturing Line Reconfiguration Toolkit

- **Input solution**. The solution can receive data from the Message Broker or from another solution, such as i4Q^{DIT}.
- **Output solution.** The solution can send the processed data to the Message Broker or directly to the solutions that have sent it input data.
- **Information received**. This solution receives one or more datasets, which will be used to train the ML model and make predictions.
- **Information sent**. Once the received data has been processed, the solution sends the results obtained by the prediction model.
- **Communication mechanism**. This solution uses the Message Broker as a communication mechanism to send information to the different pilot solutions.
    - The input data will be received through the topic established by the i4Q^{DIT} Solution.
    - For the output data, the default topic will be *LRT_Output*, but the user will be able to configure it and use the one desired.

4.1.7 i4Q^{AD} – Analytics Dashboard

- **Input solution**. The solution can receive data from the Message Broker or from any other solution, such as i4Q^{LRT}.
- **Output solution**. The processed data is not sent to any solution.
- **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 is in charge of generating 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.

4.2 Pilot 2: Diagnostics and IoT Services

In this subsection, the pipeline of Pilot 2 *Diagnostics and IoT Services* is analysed, and the most important modifications made from the time it was defined to the present version are explained in a justified manner.

After checking it with the pilot and the different solution providers, the updated version of the pipeline is presented in the following image.

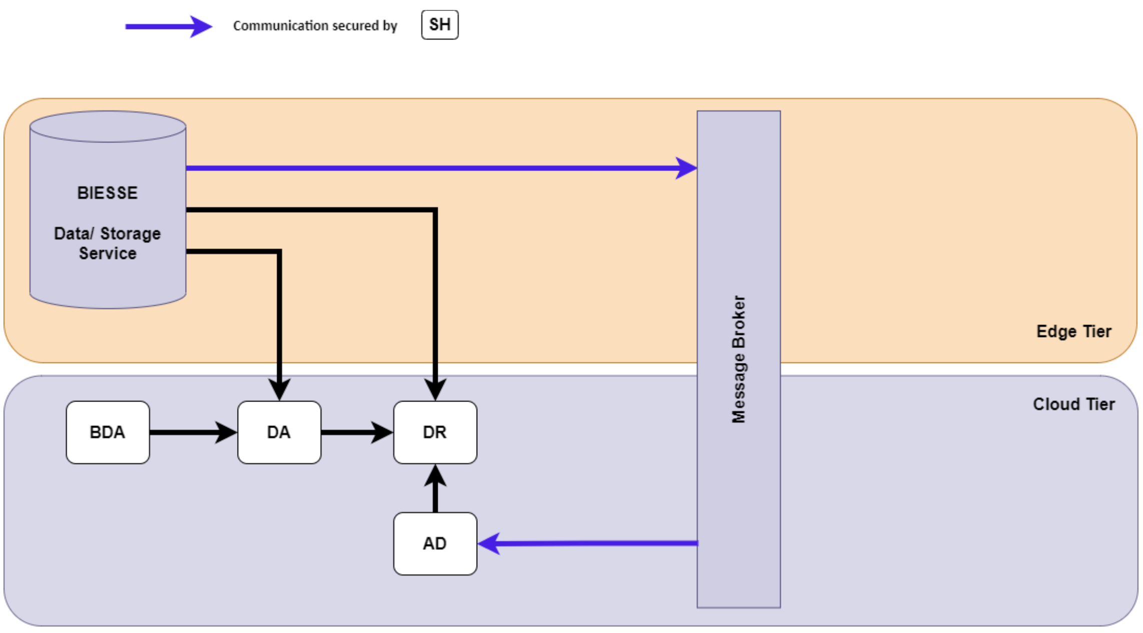


Figure 2. Pilot 2 pipeline diagram

Since the pipeline was defined, the main modifications made are as follows.

- **The i4Q^{LRT} Solution has been removed from the pipeline**. After discussion with the pilot and UPV, it has been decided to discard the use of the i4Q^{LRT} Solution, as the functionality provided does not help to meet the requirements established for the pilot.
- **The i4Q^{IM} and i4Q^{DIT} Solutions have been removed from the pipeline**. After discussion with BIESSE, it has been decided to discard these solutions from the pipeline, as the use of them implies the sharing of private know-how. In addition, the development of AI-based analysis models performed by BIESSE produced the same results as those obtained by the i4Q^{IM} Solution.
- **Modification in the representation of the Security Handler solution** (**i4Q^{SH}**). In the past, the use of certificates signed with the Certificate Authority generated by the Security Handler were represented using a box with the initials “SH”. This representation has been replaced by purple arrows connecting those solutions that use the i4Q^{SH} Solution to secure communications by using certificates. In this way, it is clearer to see which communications are secured and which are not.

4.2.1 i4Q^{DR} – Data Repository

- **Input solution**. This solution receives data from the i4Q^{DA} Solution.
- **Output solution**. This solution does not send data to any other.
- **Information received**. This solution receives data analysis and prediction models results and insight from i4Q^{DA} Solution and results from on-site analysis performed on the edge machines.
- **Information sent**. This solution does not send information to any other.
- **Communication mechanism**. This solution does not make use of Message Broker or any other communication mechanism, as the i4Q^{AD} Solution reads the information directly from i4Q^{DR}.

4.2.2 i4Q^{BDA} – Big Data Analytics Suite

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 optimized for deployment and execution according to the pilot infrastructure's hardware specifications.

- **Input solution**. This solution does not receive data from any other.
- **Output solution**. The data processed by this solution is sent to the i4Q^{DA} Solution.
- **Information received**. This solution does not receive information.
- **Information sent**. This solution sends the optimal configuration parameters for the i4Q^{DA} Solution to be deployed on the pilot's infrastructure.
- **Communication mechanism**. This solution does not make use of Message Broker or any other communication mechanism.

4.2.3 i4Q^{DA} – Services for Data Analytics

- **Input solution**. The solution receives data from the i4Q^{DR} Solution, which provides data stored in a database, and from the Message Broker, which provides real-time data.
- **Output solution**. The processed information is sent to the i4Q^{DR} Solution to be stored in a database. As for real-time information, this is sent to the Message Broker.
- **Information received**. The solution receives KPI-related data and client's spindle change data.
- **Information sent**. The solution sends data analysis and prediction model's results and insights.
- **Communication mechanism**. This solution uses the Message Broker as a communication mechanism to send real-time data. However, the topic used to send this information is unknown.

4.2.4 i4Q^{AD} – Analytics Dashboard

- **Input solution**. The solution can receive data from the Message Broker or from any other solution.
- **Output solution**. The processed data is not sent to any solution.
- **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 is in charge of generating the different dashboards according to the needs indicated by the user.
- **Information sent**. This solution does not send information to any other.
- **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.

4.3 Pilot 3: White Goods Product Quality

In this subsection, the pipeline of Pilot 3 *White Goods Product Quality* is analysed, and the most important modifications made from the time it was defined to the present version are explained in a justified manner.

After checking it with the pilot and the different solution providers, the updated version of the pipeline is presented in the following image.

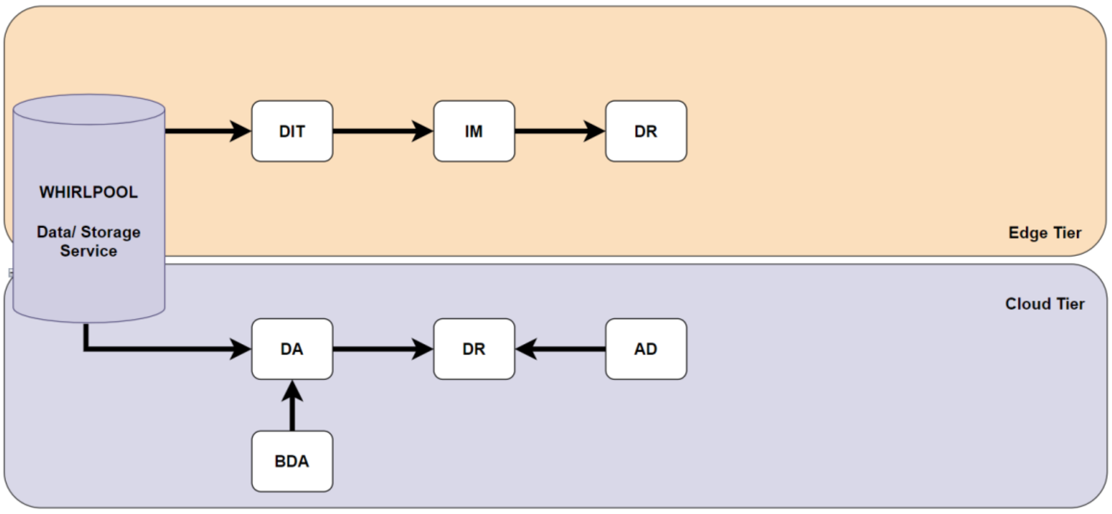


Figure 3. Pilot 3 pipeline diagram

Since the pipeline was defined, the main modifications made are as follows.

- **The i4Q^{DT} and i4Q^{PA} Solutions have been removed from the pipeline**. After discussion with the pilot and the partners involved, it has been decided to remove these solutions from the pipeline, as the work they perform is the same as the work already done with i4Q^{DA}. Therefore, keeping these solutions in the pipeline would lead to an overlapping of the functions performed, as the results obtained would be very similar to those obtained with i4Q^{DA}.

- **The use of Message Broker and the i4Q^{SH}** Solution has been discarded. Due to the use of Whirlpool's private cloud, it is unnecessary to implement these security mechanisms, as they are already implemented there.
- **The i4Q^{DA} Solution has been moved from the Edge Tier to the Cloud Tier**. The use of Google Kubernetes Engine (GKE) in Whirlpool's private cloud makes it possible to scale the i4Q^{DA} Solution with ease using Kubernetes features. For this reason, it has been decided to move the solution from one tier to the other.
- **A new instance of the i4Q^{DR}** Solution has been added to the Edge Tier. Results from models' evaluation are now stored directly on edge for further fine-tunings and then moved to the cloud for production use.

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

- **Input solution**. This solution receives data directly from the pilot's infrastructure, so it does not have an input solution that provides the data.
- **Output solution**. The data processed by this solution is sent to the i4Q^{QD} Solution.
- **Information received**. This solution receives datasets from various sensors from the production pipeline.
- **Information sent**. This solution sends processed, merged and transformed datasets.
- **Communication mechanism**. Message Broker is used as a communication mechanism to send information to the i4Q^{QD} Solution via *DIT_topic1*.

4.3.2 i4Q^{DR} – Data Repository

- **Input solution**. This solution receives data from i4Q^{DA} and i4Q^{IM} Solutions.
- **Output solution**. This solution does not send data to any other.
- **Information received**. This solution receives data analysis and analytics models results and insight from i4Q^{DA} Solution and analytics models evaluation from i4Q^{IM} Solution.
- **Information sent**. This solution does not send information to any other.
- **Communication mechanism**. This solution does not make use of Message Broker or any other communication mechanism, as the i4Q^{AD} Solution reads the information directly from i4Q^{DR}.

4.3.3 i4Q^{IM} – Infrastructure Monitoring

- **Input solution**. This solution receives data from i4Q^{DIT} Solution.
- **Output solution**. The processed data is sent to the i4Q^{DR} Solution to be stored in a database.
- **Information received**. This solution receives an enriched dataset with features for the development and training of a Machine Learning model. Specifically, it contains information about the quality conformity of the manufactured washing machines.
- **Information sent**. This solution sends alerts for the detection of faulty products.

- **Communication mechanism**. Although the information is not sent to any specific solution, Message Broker is used as a communication mechanism to provide the generated alert via the *IM_topic* for potential use. This alert is a JSON message that indicates the instances where a faulty product has been detected.

4.3.4 i4Q^{DA} – Services for Data Analytics

The i4Q^{DA} Solution is used to perform Data Processing, Analytics and Machine Learning tasks, by offering a visual, block-based programming environment to create data processing and AI-related workflows. In the case of pilot 3, the i4Q^{DA} Solution calculates and analyses the pilot's KPIs and production data.

- **Input solution**. The solution receives data from the i4Q^{DR} Solution, which provides data stored in a database.
- **Output solution**. The processed information is sent back to the i4Q^{DR} Solution to be stored in the database.
- **Information received**. The solution receives KPI-related and production data.
- **Information sent**. The solution sends data analysis results and insights.
- **Communication mechanism**. This solution does not make use of Message Broker or any other communication mechanism.

4.3.5 i4Q^{AD} – Analytics Dashboard

- **Input solution**. The solution can receive data from the Message Broker or from any other solution.
- **Output solution**. The processed data is not sent to any solution.
- **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 is in charge of generating 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.

4.3.6 i4Q^{BDA} – Big Data Analytics Suite

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 optimized for deployment and execution according to the pilot infrastructure's hardware specifications.

- **Input solution**. This solution does not receive data from any other.
- **Output solution**. The data processed by this solution is sent to the i4Q^{DA} Solution.

- **Information received**. This solution does not receive information.
- **Information sent**. This solution sends the optimal configuration parameters for the i4Q^{DA} Solution to be deployed on the pilot's infrastructure.
- **Communication mechanism**. This solution does not make use of Message Broker or any other communication mechanism.

4.4 Pilot 4: Aeronautics and Aerospace Metal Parts Quality

In this subsection, the pipeline of Pilot 4 *Aeronautics and Aerospace Metal Parts Quality* is analysed, and the most important modifications made from the time it was defined to the present version are explained in a justified manner.

After checking it with the pilot and the different solution providers, the updated version of the pipeline is presented in the following image.

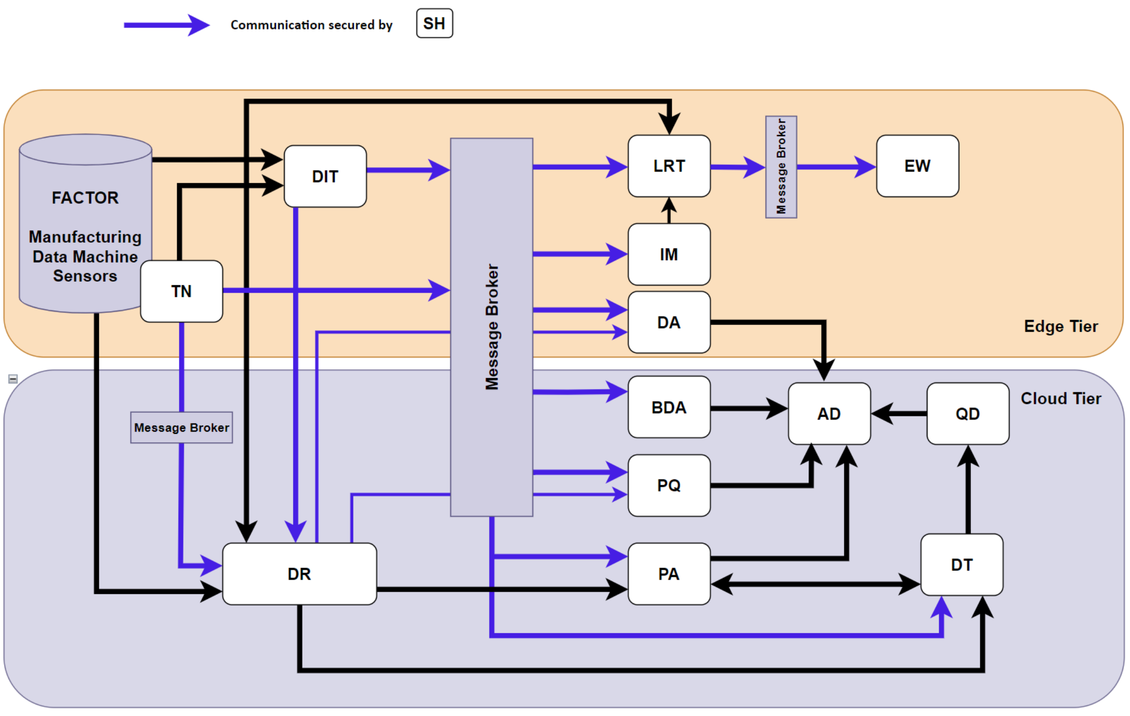


Figure 4. Pilot 4 pipeline diagram

Since the pipeline was defined, the main modifications made are as follows.

- **An instance of the Message Broker has been added between the i4Q^{LRT} and i4Q^{EW} Solutions**. After discussion with CERTH and IBM, it has been decided that the communication between these two solutions should be done in a secure way, therefore an instance of MB has been added.
- **Modification in the representation of the Security Handler solution** (**i4Q^{SH}**). In the past, the use of certificates signed with the Certificate Authority generated by the Security Handler were represented using a box with the initials "SH". This representation has been replaced by purple arrows connecting those solutions that use the i4Q^{SH} Solution to secure communications by using certificates. In this way, it is clearer to see which communications are secured and which are not.

- **The arrow from i4Q^{DR} to i4Q^{BDA} has been removed** since this integration is no longer necessary. The reason is that i4Q^{BDA} is deployed as a software bundle that contains the i4Q^{DA} Solution, and there is already an integration between i4Q^{DR} and i4Q^{DA}.

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

In this pilot the IWSN part of the i4Q^{TN} Solution will be deployed to gather different type of relevant sensor. The main process selected to digitize using the IWSN will be a horizontal bandsaw used to cut cylindrical rods. The purpose is to obtain information from the current consumption to detect the wear of the saw for different. This information, in combination with additional sensors and other spots of monitorization will be delivered to the main i4Q broker and interested solution will be able to acquire the data stream in real time or throw the historical collected at the i4Q^{DR} Solution. The proposed deployment will be based on a set of nodes between 10 and 20 (depending on availability) that's create a mesh network and collect different sensor and information delivered to the broker throw the IWSN gateway. The complete network covers not only the horizontal bandsaw but also other machines and local environmental variables in different zones of the pilot plant.

- **Input solution**. The information is generated from different sensors connected to the IWSN nodes.
- **Output solution**. The processed data is sent to the i4Q^{DR} and any interested solution in real time data stream.
- **Information received**. The solution receives information generated by the connected sensors.
- **Information sent**. The solution sends messages in JSON format including different collected sensor for each of the IWSN nodes.
- **Communication mechanism**. Message Broker is used to send the generated information to the i4Q^{DR}. Every solution that needs to receive information from the i4Q^{TN} will need a different topic to avoid emptying the Kafka queues for different solutions. The selected topic to send data to i4Q^{DR} is: *p4_tn2dr*. See below a message example in JSON format (additional pair name-value could be consider depending on the final sensor deployed).

```
{
      'nodeid':'1.1.2',
      'SeqN':75,
      '10':12,
      '1':4792,
      '34':0.0,
      'timestamp': '2023-05-07 10:42:34'
}
```

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

- **Input solution**. This solution receives data directly from the pilot's infrastructure, so it does not have an input solution that provides the data.
- **Output solution**. The data processed by this solution is sent to the i4Q^{DT} and i4Q^{LRT} Solutions.
- **Information received**. This solution receives datasets directly from the pilot infrastructure.
- **Information sent**. This solution sends merged and transformed datasets.
- **Communication mechanism**. For this solution neither the Message Broker nor any other communication mechanism has been used.

4.4.3 i4Q^{DR} – Data Repository

In this pilot i4Q^{DR} integrates with several components using different mechanisms. Thus, each integration is explained separately in the following subsections.

4.4.3.1 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 minimise 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 into the DR'S MongoDB, namely in the database created in the step before. In order 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 MB. Instead, the "exporter" script described above is used to replicate the most relevant data stored into FACTOR's MongoDB into the DR's MongoDB.

4.4.3.2 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 MB. i4Q^{DIT} opens a connection to the MinIO instance deployed by i4Q^{DR} in order to store the file in a bucket[3].

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

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

4.4.3.4 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 i4Q^{LRT}.
- **Communication mechanism.** This integration does not involve the MB. i4Q^{LRT} connects to the database created in the DR's MongoDB storing the replica of FACTORs manufacturing data.

4.4.3.5 Inbound integration with i4Q^{LRT}

- **Input solution**. i4Q^{DR} receives data from the i4Q^{LRT} Solution.
- **Information received.** 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 MB. i4Q^{LRT} opens a connection to the MinIO instance deployed by the i4Q^{DR} to store the model into a bucket.

[3] This might change in the future. Depending on the content of the CSV file, it might be more appropriate to store it into a database created in the MongoDB instance deployed by i4Q^{DR}.

4.4.3.6 *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 i4Q^{PA}.
- **Communication mechanism**. This integration does not involve the MB. i4Q^{PA} connects to a bucket created in the MinIO instance deployed by i4Q^{DR}.

4.4.3.7 *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.

4.4.4 i4Q^{LRT} – Manufacturing Line Reconfiguration Toolkit

- **Input solution**. The solution can receive data from the Message Broker or from another solution, such as i4Q^{DIT}.
- **Output solution.** The solution can send the processed data to the Message Broker or directly to the solutions that have sent it input data.
- **Information received**. This solution receives one or more datasets, which will be used to train the ML model and make predictions.
- **Information sent**. Once the received data has been processed, the solution sends the results obtained by the prediction model.
- **Communication mechanism**. This solution uses the Message Broker as a communication mechanism to send information to the different pilot solutions.
  - The input data will be received through the topic established by the i4Q^{DIT} solution.
  - For the output data, the default topic will be *LRT_Output*, but the user will be able to configure it and use the one desired.

4.4.5 i4Q^{IM} – Infrastructure Monitoring

- **Input solution**. The solution receives data from i4Q^{DR} and i4Q^{DIT} Solutions.
- **Output solution**. The data processed by this solution is sent to the i4Q^{LRT} Solution.
- **Information received**. This solution receives an enriched dataset with features for the development and training of a ML model. Specifically, it contains information about the malfunctions on Factor's CNC machines.
- **Information sent**. This solution sends alerts for the detection of problems in the CNC machine.

- **Communication mechanism**. Message Broker is used to provide the i4Q^{LRT} Solution with the generated alert via the *IM_topic*. This alert is a JSON message that indicates the instances where a problem has been detected.

4.4.6 i4Q^{DA} – Services for Data Analytics

- **Input solution**. The solution receives data from the i4Q^{DR} Solution, which provides data stored in a database.
- **Output solution**. The processed information is sent back to the i4Q^{DR} Solution to be stored in the database.
- **Information received**. The solution receives KPI-related and production data.
- **Information sent**. The solution sends data analysis results and insights.
- **Communication mechanism**. This solution does not make use of Message Broker or any other communication mechanism.

4.4.7 i4Q^{BDA} – Big Data Analytics Suite

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 optimized for deployment and execution according to the pilot infrastructure's hardware specifications.

- **Input solution**. This solution does not receive data from any other.
- **Output solution**. The data processed by this solution is sent to the i4Q^{DA} Solution.
- **Information received**. This solution does not receive information.
- **Information sent**. This solution sends the optimal configuration parameters for the i4Q^{DA} Solution to be deployed on the pilot's infrastructure.
- **Communication mechanism**. This solution does not make use of Message Broker or any other communication mechanism.

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

- **Input solution**. The solution can receive data from any solution that predicts continuous quality characteristics.
- **Output solution**. The solution does not send data to any other.
- **Information received**. This solution receives continuous quality characteristics.
- **Information sent**. This solution does not send information to any other.
- **Communication mechanism**. This solution supports different mechanisms to integrate with other solutions. In the one hand, it can receive information by using the Message Broker. In the other hand, it can load a CSV file, open a connection to a MongoDB database to read stored data, or perform HTTP requests to an API in order to retrieve data. In all cases the connection can be configured and established in the App.

4.4.9 i4Q^{PA} – Prescriptive Analysis Tools

- **Input solution**. The solution receives data from i4Q^{DT} and i4Q^{DR} Solutions.
- **Output solution**. The solution sends data to i4Q^{DT}, i4Q^{LRT}, and i4Q^{AD} Solutions.
- **Information received**. Depending on the source solution, different types of information can be received.
    - From the i4Q^{DT} Solution, simulation results are received.
    - From the i4Q^{DR} Solution, models are received.
- **Information sent**. Depending on the target solution, different types of information can be sent.
    - To the i4Q^{DT} Solution, different simulation requests can be sent, such as the model, input signals, and a map of scenarios, among others.
    - To the i4Q^{LRT} Solution, a table with evaluations can be sent.
    - To the i4Q^{AD} Solution, simulations and evaluation results can be sent.
- **Communication mechanism**. Message Broker can be used to obtain information from this solution and to make requests to the i4Q^{DT} Solution. On the other hand, models and input signals can be consumed directly from the i4Q^{DR} Solution.

  Regarding the use of the MB, the following topics can be used to exchange information:
    - *PA_output_model_scenarios*, PA_output_sim_results, and *PA_output_eval_results* are the general topics to communicate with the i4Q^{PA} Solution.
    - *PA_output_sim_request*, and *DT_output_sim_results* are the specific topics for exchanging information between the i4Q^{PA} and i4Q^{DT} Solutions.

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

- **Input solution**. This solution receives data from those that want to deploy on an edge target, such as the i4Q^{LRT}.
- **Output solution**. At a more advanced stage, the data processed by this solution will be sent to the i4Q^{IM} Solution to use it to understand current events and resources.
- **Information received**. The data received by this solution includes workloads to be deployed and a representation of the possible targets.
- **Information sent**. This solution sends the current state of affairs at the edge, so that components can have a clear view of the topology.
- **Communication mechanism**. This solution uses GitHub as communication mechanism. In this way, those solutions that want to deploy a new (or revised) workload pushes a new version to Git, which in turn makes that information available to our solution, which then gets the necessary information from Git and distributes it to the correct destination.

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

- **Input solution**. This solution receives input from those wishing to deploy new or revised AI model on an edge target (for example, the i4Q^{LRT} Solution in the FACTOR pilot).
- **Output solution**. At a more advanced stage, the output of this solution may pass information to the i4Q^{IM} Solution to use in their understanding of current affairs and resources. It can further send information to a cloud-based data scientist to evaluate the efficiency of the current deployed model.
- **Information received**. The input received by this solution includes models to be deployed and a representation of the possible targets.
- **Information sent**. This solution sends the current state of affairs at the edge, so that components can have a clear view of the topology. Mostly concentrated on feedback concerning the currently deployed and executing model.
- **Communication mechanism**. This solution uses GitHub as communication mechanism. In this way, those solutions that want to deploy a new (or revised) workload pushes a new version to Git, which in turn makes that information available to our solution, which then gets the necessary information from Git and distributes it to the correct destination.

4.4.12 i4Q^{AD} – Analytics Dashboard

- **Input solution**. The solution can receive data from the Message Broker or from any other solution.
- **Output solution**. The processed data is not sent to any solution.
- **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 is in charge of generating 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.

4.4.13 i4Q^{QD} – Rapid Quality Diagnosis

- **Input solution**. This solution receives data from the i4Q^{DR} and i4Q^{DIT} Solutions.
- **Output solution**. The processed data is not sent to any solution.
- **Information received**. This solution receives an enriched dataset with features for the development and training of a ML model. Specifically, it contains information regarding the quality conformity of injection molded products.
- **Information sent**. This solution sends alerts for the detection of faulty machined products.
- **Communication mechanism**. Although the information is not sent to any specific solution, Message Broker is used as a communication mechanism to provide the generated alert via

the *QD_topic* for potential use. This alert is a JSON message that indicates the instances where a faulty product has been detected.

4.4.14 i4Q^{DT} – Digital Twin Simulation Services

- **Input solution**. This solution receives data from i4Q^{PA}, i4Q^{DR}, and i4Q^{DIT} Solutions.
- **Output solution**. The processed data is sent to the i4Q^{PA} Solution.
- **Information received**. Depending on the source solution, different types of information can be received.
    - From the i4Q^{DIT} Solution, post-processed data is received from the pilot infrastructure.
    - From the i4Q^{PA} Solution, simulation requests are received.
- **Information sent**. The data sent by the solution are simulations/predictions generated by a model, which has been previously created. This data is related to the predicted quality of the manufactured pieces, which will be calculated based on the data provided by the input sensors. Finally, this data can be sent to the i4Q^{PA} Solution or consumed directly by the user through the corresponding topic.
- **Communication mechanism**. Message Broker can be used to obtain information from this solution and to make requests to the i4Q^{PA} Solution. On the other hand, input model/data can be consumed directly from the i4Q^{DR} Solution.

  Regarding the use of the MB, the following topics can be used to exchange information:
    - *DT_input_model*, *DT_input_data*, *DT_output_model*, and *DT_output_data* are the general topics to communicate with the i4Q^{DT} Solution.
    - *PA_output_sim_request* and *DT_output_sim_results* are the specific topics for exchanging information between the i4Q^{PA} and i4Q^{DT} Solutions.

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

In this subsection, the pipeline of Pilot 5 *Advanced In-line Inspection for Incoming Prime Matter Quality Control* is analysed, and the most important modifications made from the time it was defined to the present version are explained in a justified manner.

After checking it with the pilot and the different solution providers, the updated version of the pipeline is presented in the following image.

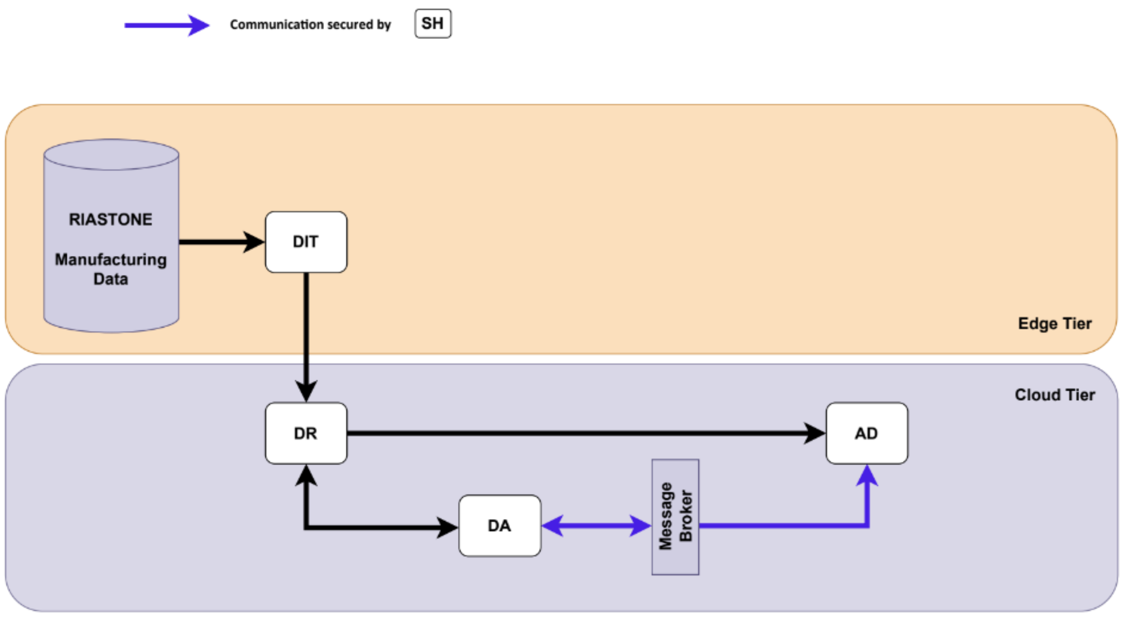


Figure 5. Pilot 5 pipeline diagram

Since the pipeline was defined, the main modifications made are as follows.

- **A box with the initials "TN" has been removed from the pipeline diagram**. After discussion with ITI and UNINOVA, it has been concluded that this box does not refer to the i4Q^{TN} Solution, as the use of this solution is not included in the pilot requirements. Therefore, we can conclude its inclusion in the previous version of the pipeline was due to an error and, in order to solve it, this box must be removed.
- **The i4Q^{LRT} Solution has been removed from the pipeline**. After discussion with the pilot and UPV, it has been decided to discard the use of the i4Q^{LRT} Solution, as the functionality provided does not help to meet the requirements established for the pilot.
- **Modification in the representation of the Security Handler solution** (**i4Q^{SH}**). In the past, the use of certificates signed with the Certificate Authority generated by the Security Handler were represented using a box with the initials "SH". This representation has been replaced by purple arrows connecting those solutions that use the i4Q^{SH} Solution to secure communications by using certificates. In this way, it is clearer to see which communications are secured and which are not.

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

This subsection analyses the interaction of the i4Q^{DIT} Solution with other solutions in the Pilot 5 pipeline. For this, the solutions from which it receives or to which it sends information are mentioned. In addition to this, the information sent or received from other solutions as well as the communication mechanism used to exchange this information are explained.

- **Input solution**. This solution receives data directly from the pilot's infrastructure, so it does not have an input solution that provides the data.
- **Output solution**. The data processed by this solution is sent to the i4Q^{DR} Solution.
- **Information received**. This solution does not receive information from any solution.

- **Information sent**. For now, the information processed by this solution is not sent to any other solution.
- **Communication mechanism**. As the solution's data is neither received nor sent to any other solution, no communication mechanism is used at the moment.

4.5.2 i4Q^{DR} – Data Repository

The i4Q^{DR} Solution is used as the main data storage solution for Pilot 5. Specifically, the pilot is using the i4Q^{DR} PostgreSQL instance and the i4Q^{DR} MongoDB instance. PostgreSQL is being used as the main data storage for both raw, pre-processed and results data. MongoDB is integrated with the i4Q^{DA} Solution to provide intermediate data storage between consecutive blocks on the workflows developed in the i4Q^{DA} Solution.

- **Input solution**. The solution receives data from i4Q^{DIT} and i4Q^{DA} Solutions.
- **Output solution**. The processed data is set to i4Q^{DIT}, i4Q^{DA} and i4Q^{AD} Solutions.
- **Information received**. This solution receives raw data (from pilot infrastructure), pre-processed data (from the i4Q^{DIT} Solution), and AI workflow results (from the i4Q^{DA} Solution).
- **Information sent**. This solution sends: raw data (to the i4Q^{DIT} Solution), pre-processed data (to i4Q^{DA} and i4Q^{AD} Solutions), and AI workflow results (to the i4Q^{AD} Solution).
- **Communication mechanism**. N/A.

4.5.3 i4Q^{DA} – Services for Data Analytics

The i4Q^{DA} Solution is used to perform Data Processing, Analytics and Machine Learning tasks, by offering a visual, block-based programming environment to create data processing and AI-related workflows. In the case of pilot 5, the i4Q^{DA} Solution calculates and analyses the pilot's KPIs and production data.

- **Input solution**. The solution receives data from the i4Q^{DR} Solution, which provides data stored in a database.
- **Output solution**. The processed information is sent back to the i4Q^{DR} Solution to be stored in the database.
- **Information received**. The solution receives KPI-related and production data.
- **Information sent**. The solution sends data analysis results and insights.
- **Communication mechanism**. This solution does not make use of Message Broker or any other communication mechanism.

4.5.4 i4Q^{AD} – Analytics Dashboard

- **Input solution**. The solution can receive data from the Message Broker or from any other solution, such as i4Q^{DR}.
- **Output solution**. The processed data is not sent to any solution.
- **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 is in charge of generating 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.

4.5.5 i4Q^{SH} – IIoT Security Handler

In the case of the i4Q^{SH} Solution, there is not an information exchange, as it is not responsible for sending or receiving data from other solutions. Instead, it provides the CAs to the Message Broker and the i4Q^{DR} Solution, so that they can generate the necessary certificates, being this the same procedure to be followed in all the pilots.

This CA will be responsible for guaranteeing the authenticity, integrity, and reliability of the digital certificates used in communications. In this way, communications will be secure, and the certificates generated in the solutions will have authenticity and integrity assured.

4.6 Pilot 6: Automatic Advanced Inspection of Automotive Plastic Parts

In this subsection, the pipeline of Pilot 6 *Automatic Advanced Inspection of Automotive Plastic Parts* is analysed, and the most important modifications made from the time it was defined to the present version are explained in a justified manner.

After checking it with the pilot and the different solution providers, the updated version of the pipeline is presented in the following image.

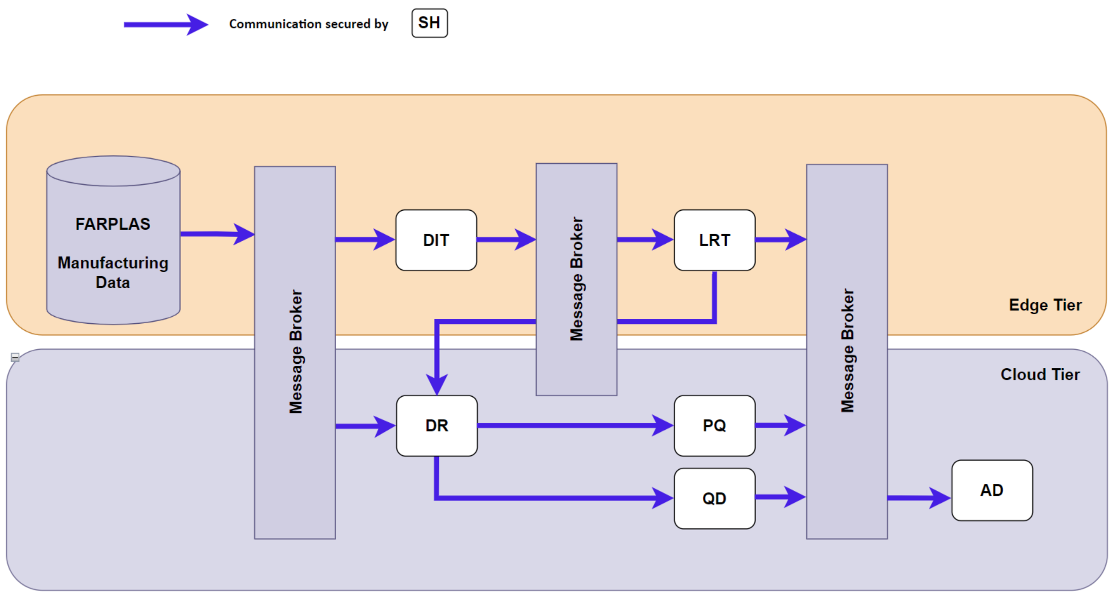


Figure 6. Pilot 6 pipeline diagram

Since the pipeline was defined, there have been no significant changes to this diagram.

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

This subsection analyses the interaction of the i4Q^{DIT} Solution with other solutions in the Pilot 6 pipeline. For this, the solutions from which it receives or to which it sends information are mentioned. In addition to this, the information sent or received from other solutions as well as the communication mechanism used to exchange this information are explained.

- **Input solution**. This solution receives data directly from the pilot's infrastructure, so it does not have an input solution that provides the data.
- **Output solution**. The data processed by this solution is sent to the i4Q^{LRT} Solution.
- **Information received**. This solution does not receive information from any solution.
- **Information sent**. For now, the information processed by this solution is not sent to any other solution.
- **Communication mechanism**. As the solution's data is neither received nor sent to any other solution, no communication mechanism is used at the moment.

4.6.2 i4Q^{SH} – IIoT Security Handler

In the case of the i4Q^{SH} Solution, there is not an information exchange, as it is not responsible for sending or receiving data from other solutions. Instead, it provides the CAs to the Message Broker and the i4Q^{DR} Solution, so that they can generate the necessary certificates, being this the same procedure to be followed in all the pilots.

This CA will be responsible for guaranteeing the authenticity, integrity, and reliability of the digital certificates used in communications. In this way, communications will be secure, and the certificates generated in the solutions will have authenticity and integrity assured.

4.6.3 i4Q^{DR} – Data Repository

In this pilot i4Q^{DR} integrates with several components using different mechanisms. Thus, each integration is explained separately in the following subsections.

4.6.3.1 Inbound integration with FARPLAS's manufacturing data

The purpose of this integration is to store data generated by sensors installed at FARPLAS premises. More specifically, this data refers to parameters of injection machines, such as temperature, volume, pressure, speed, etc.

The manufacturing data is currently streamed by a software artifact developed by FARPLAS via the MB as a message in JSON format at an approximate rate of once per minute per machine. The i4Q^{DR} Solution consumes such data and persists in a MongoDB database, so that other solutions can retrieve and process it.

- **Input solution**. The solution receives streaming manufacturing data from the FARPLAS' s software.
- **Information received**. This solution receives the value of different parameters of FARPLAS injection machines at a given moment in JSON format.
- **Communication mechanism**. This integration is performed via the MB. FARPLAS software artifact publishes the manufacturing data in topic "e117_cyc". The i4Q^{DR} Solution is subscribed to this topic in order to consume such data and periodically stores it in a database created in a MongoDB instance.

4.6.3.2 Inbound integration with i4Q^{LRT}

- **Input solution**. The solution receives data from the i4Q^{LRT} Solution.
- **Input received**. This solution receives a file corresponding to the ML model generated by the i4Q^{LRT} Solution, which needs to be accessible for further uses.
- **Communication mechanism**. This integration does not involve the MB. The i4Q^{LRT} opens a connection to the MinIO instance deployed by the i4Q^{DR} to store the model into a bucket.

4.6.3.3 Outbound integration with i4Q^{PQ} and i4Q^{QD}

- **Output solution**. The solution receives data from i4Q^{QD} and i4Q^{PQ} Solutions.
- **Information sent**. This solution sends the manufacturing data from FARPLAS.
- **Communication mechanism**. This integration does not involve the MB. The output solutions connect to the database created in the MongoDB instance deployed by i4Q^{DR} to store FARPLAS's manufacturing data, in order to extract the specific information required in each case.

4.6.4 i4Q^{LRT} – Manufacturing Line Reconfiguration Toolkit

This subsection analyses the interaction of the i4Q^{LRT} Solution with other solutions in the Pilot 6 pipeline. For this, the solutions from which it receives or to which it sends information are mentioned. In addition to this, the information sent or received from other solutions as well as the communication mechanism used to exchange this information are explained.

- **Input solution**. The solution can receive data from the Message Broker or from another solution, such as i4Q^{DIT}.
- **Output solution.** The solution can send the processed data to the Message Broker or directly to the solutions that have sent it input data.
- **Information received**. This solution receives one or more datasets, which will be used to train the ML model and make predictions.
- **Information sent**. Once the received data has been processed, the solution sends the results obtained by the prediction model.
- **Communication mechanism**. This solution uses the Message Broker as a communication mechanism to send information to the different pilot solutions.
    - The input data will be received through the topic established by the i4Q^{DIT} Solution.
    - For the output data, the default topic will be *LRT_Output*, but the user will be able to configure it and use the one desired.

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

- **Input solution**. This solution receives data from any solution capable of predicting continuous quality characteristics.
- **Output solution**. The processed data is not sent to any solution.
- **Information received**. This solution receives data from continuous quality characteristics.
- **Information sent**. The information processed by this solution is not sent to any other solution.
- **Communication mechanism**. This solution supports different mechanisms to integrate with other solutions. In the one hand, it can receive information by using the Message Broker. In the other hand, it can load a CSV file, open a connection to a MongoDB database to read stored data, or perform HTTP requests to an API in order to retrieve data. In all cases the connection can be configured and established in the App.

4.6.6 i4Q^{QD} – Rapid Quality Diagnosis

- **Input solution**. This solution receives data from i4Q^{DR}, i4Q^{DIT}, and i4Q^{DT} Solutions.
- **Output solution**. The processed data is not sent to any solution.
- **Information received**. This solution receives an enriched dataset with features for the development and training of a Machine Learning model. Specifically, it contains information about the quality conformity of the machined products.

- **Information sent**. This solution sends alerts for the detection of faulty machined products.
- **Communication mechanism**. Although the information is not sent to any specific solution, Message Broker is used as a communication mechanism to provide the generated alert via the *QD_topic* for potential use. This alert is a JSON message that indicates the instances where a faulty product has been detected.

4.6.7 i4Q^{AD} – Analytics Dashboard

- **Input solution**. The solution can receive data from the Message Broker or from any other solution.
- **Output solution**. The processed data is not sent to any solution.
- **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 is in charge of generating 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.

5 Testing and Validation

This section aims to analyse the quality of the software developed by the different solution providers. For this purpose, the SonarQube tool has been used to analyse the source code of each solution. Once the analysis has been completed, this tool generates a report with the following metrics.

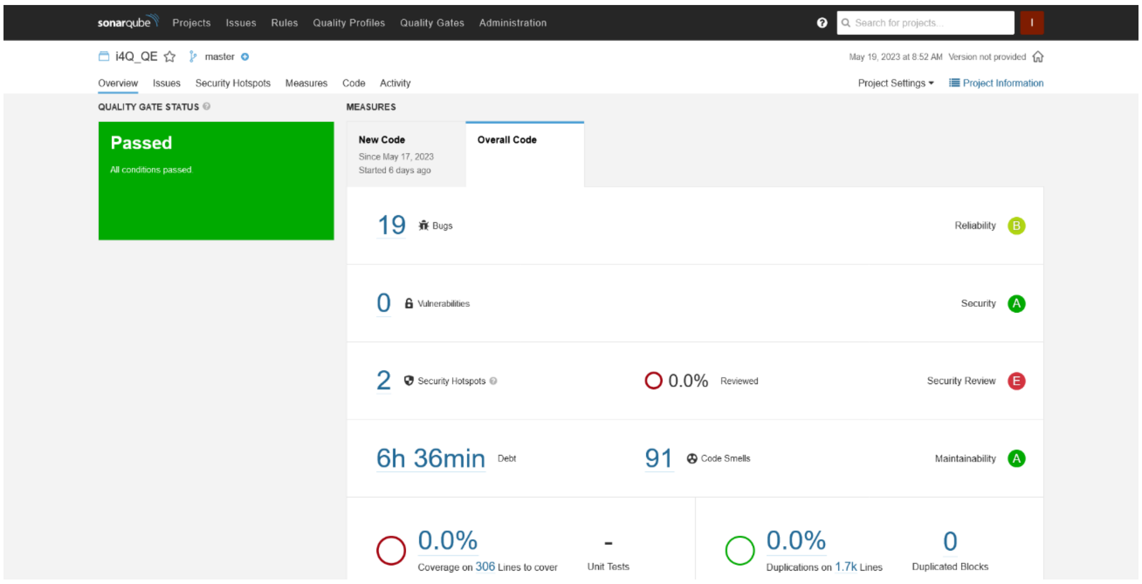
- **Bugs**. This metric indicates the number of problems found in the source code. These problems must be solved as soon as possible, as they may cause the solution to stop working correctly.
- **Vulnerabilities**. This measurement exposes the number of security-related problems that have been found in the source code. These problems are very dangerous, as they represent a backdoor access for attackers.
- **Security Hotspots**. This metric quantifies the number of code snippets that are security sensitive. These snippets need to be manually reviewed to discover if there are threats or if there is vulnerable code that needs to be fixed.
- **Technical Debt**. Measures the effort required in minutes to fix all code smells in the source code. When this measure is shown in form of days, it is assumed that a day has a duration of 8 hours (normal working day duration).
- **Code Smells**. Measures the quantity of problems related to code maintenance. A higher number of code smells means that it is harder to maintain the source code when changes must be introduced. This can also lead to introducing new bugs in the code when adding new functionalities.
- **Coverage**. This is a measure that combines line and condition coverage, to give an answer to the percentage of code that has been covered by unit tests.

- **Unit Tests**. This metric quantifies the number of unit tests performed on the source code.
- **Duplications and Duplicated Blocks**. These measure the percentage of duplicated lines and the number of duplicated blocks of lines.

In the following subsections, the reports generated for each of the solutions will be analysed and the results obtained will be discussed.

5.1 i4Q^{QE}: QualiExplore for Data Quality Factor Knowledge

This subsection analyses the source code quality of the *QualiExplore for Data Quality Factor Knowledge* (i4Q^{QE}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



May 19, 2023 at 8:52 AM Version not provided

Passed
All conditions passed.

New Code: Since May 17, 2023, Started 6 days ago

19 Bugs — Reliability B
0 Vulnerabilities — Security A
2 Security Hotspots — 0.0% Reviewed — Security Review E
6h 36min Debt — 91 Code Smells — Maintainability A
0.0% Coverage on 306 Lines to cover — - Unit Tests
0.0% Duplications on 1.7k Lines — 0 Duplicated Blocks

Figure 7. SonarQube report for the i4Q^{QE} Solution

At the moment, this solution has a considerable quantity of **bugs**. These may cause the solution not to work properly in the future. Therefore, it is proposed to review and refactor the source code as soon as possible.

In the case of **vulnerabilities**, currently none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, 2 fragments have been found that need to be reviewed in order to verify that they are not vulnerable code.

Regarding the **technical debt**, although it is considerable, it can be addressed, as the number of **code smells** is less than a hundred. However, an early refactoring of the code should be planned in order to minimise these metrics.

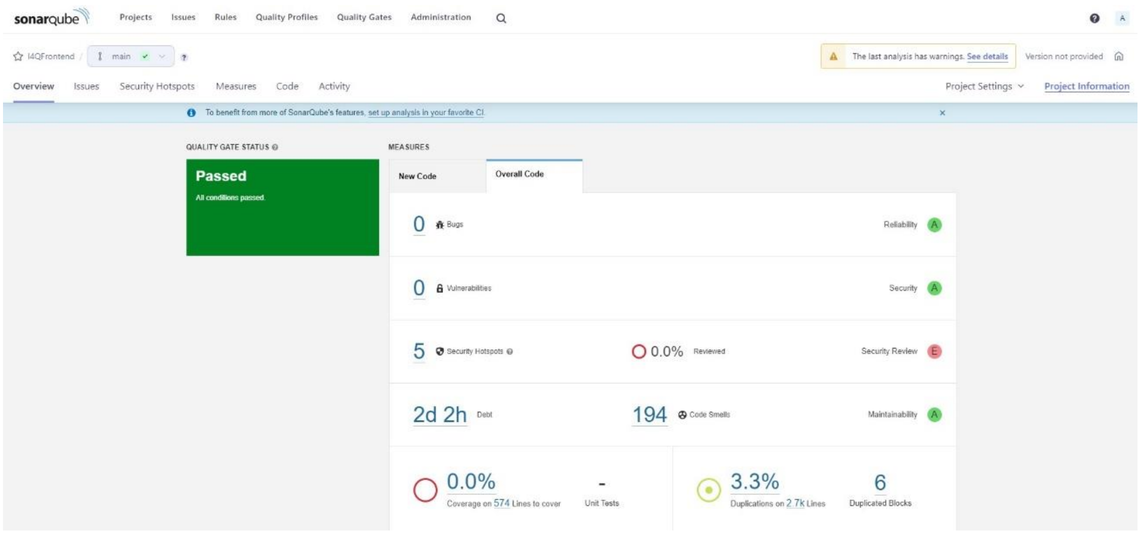
In terms of **coverage** and the number of **unit tests**, at present no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, no **duplications of lines or blocks** have been found, which indicates that the implemented code makes a good reuse of the defined elements, avoiding unnecessary code repetitions.

5.2 i4Q^{BC}: Blockchain Traceability of Data

This subsection analyses the source code quality of the *Blockchain Traceability of Data* (i4Q^{BC}) Solution. This solution consists of two components: the frontend and the management service. Below, the reports generated by SonarQube for each of these components are analysed, and then the results obtained are discussed.

The following image shows the results of the frontend component for each of the metrics explained at the beginning of section 5. After this, the results obtained for this component are discussed.



The last analysis has warnings. See details

Version not provided

To benefit from more of SonarQube's features, set up analysis in your favorite CI.

Passed
All conditions passed.

0 Bugs — Reliability A

0 Vulnerabilities — Security A

5 Security Hotspots — 0.0% Reviewed — Security Review E

2d 2h Debt — 194 Code Smells — Maintainability A

0.0% Coverage on 574 Lines to cover — - Unit Tests

3.3% Duplications on 2.7k Lines — 6 Duplicated Blocks

Figure 8. SonarQube report for the frontend component of i4Q^{BC} Solution

Currently, this component does not have any **bug**, which means that no erratic behaviour is expected in the component due to the presence of errors in the source code. For the moment, it is not necessary to perform a refactoring focused on fixing bugs, but it is important to be alert and be careful not to introduce new ones in the future.

In the case of **vulnerabilities**, by now, none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, 5 code snippets suspected to contain vulnerable code have been found, therefore, an early review of these snippets should be performed to verify the existence or not of vulnerable code.

Regarding the **technical debt**, it is considerable but can be addressed in two or three days of work. Given that the number of code smells is almost 200, an early refactoring of the code should be planned in order to minimise these metrics.

In terms of **coverage** and the number of **unit tests**, at present no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the component.

Finally, for the number of **duplicated lines and blocks** of code, the reported data indicates that there is a slight duplication in the code, but this is not very worrying, as it represents 3.3% of the total number of lines and only a total of 6 blocks of code. If possible, a code review is recommended to reduce the amount of duplicity, but this task is not urgent and can be done later.

Once the discussion of the frontend component results is finished, the following image shows the results of the management system component and, afterwards, the results obtained by this component for each of the metrics explained at the beginning of section 5 are discussed.

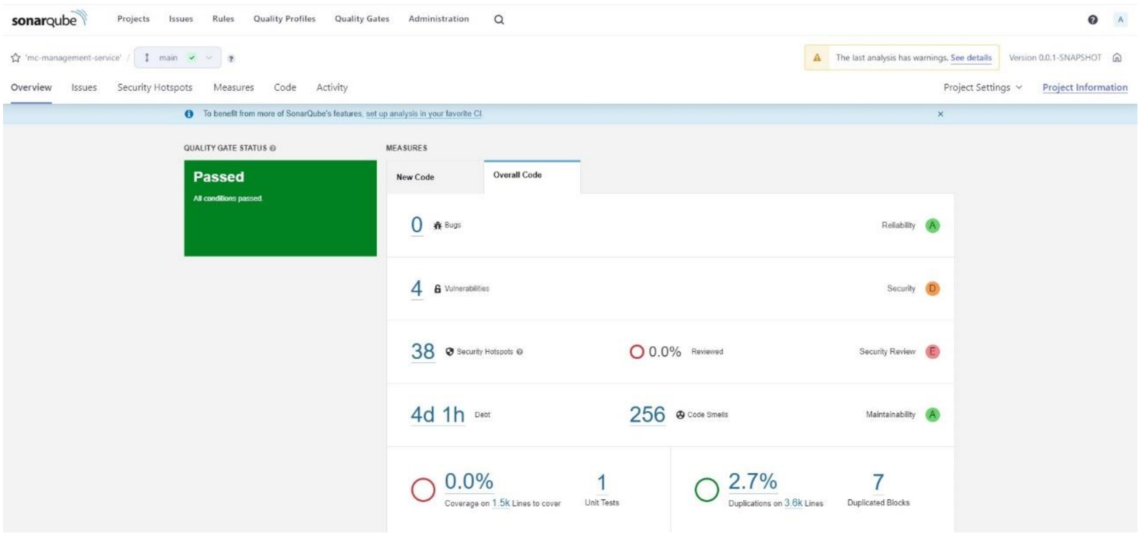

sonarqube
Projects Issues Rules Quality Profiles Quality Gates Administration
'mc-management-service' / main
The last analysis has warnings. See details
Version 0.0.1-SNAPSHOT
Overview Issues Security Hotspots Measures Code Activity
Project Settings
Project Information
To benefit from more of SonarQube's features, set up analysis in your favorite CI.
QUALITY GATE STATUS
Passed
All conditions passed.
MEASURES
New Code
Overall Code
0 Bugs
Reliability A
4 Vulnerabilities
Security D
38 Security Hotspots
0.0% Reviewed
Security Review E
4d 1h Debt
256 Code Smells
Maintainability A
0.0% Coverage on 1.5k Lines to cover
1 Unit Tests
2.7% Duplications on 3.6k Lines
7 Duplicated Blocks

Figure 9. SonarQube report for the management system component of i4Q^{BC} Solution

Currently, this component does not have any **bug**, which means that no erratic behaviour is expected in the component due to the presence of errors in the source code. For the moment, it is not necessary to perform a refactoring focused on fixing bugs, but it is important to be alert and be careful not to introduce new ones in the future.

In the case of **vulnerabilities**, 4 security-related problems have been reported in this component. This is a very serious issue, as the presence of vulnerabilities in the source code can facilitate backdoor access to attackers. For this reason, an immediate code review is recommended in order to mitigate these problems.

For the **security hotspots**, a total of 38 snippets suspected of containing vulnerable code have been reported in this component. This quantity is very worrying, as it indicates that there is a high probability that the code of this component is susceptible to security-related problems. For this reason, an urgent review of these code fragments should be undertaken to verify whether they represent a threat or not and, if they do, corrective measures should be implemented to minimise the risk of suffering an attack in the future.

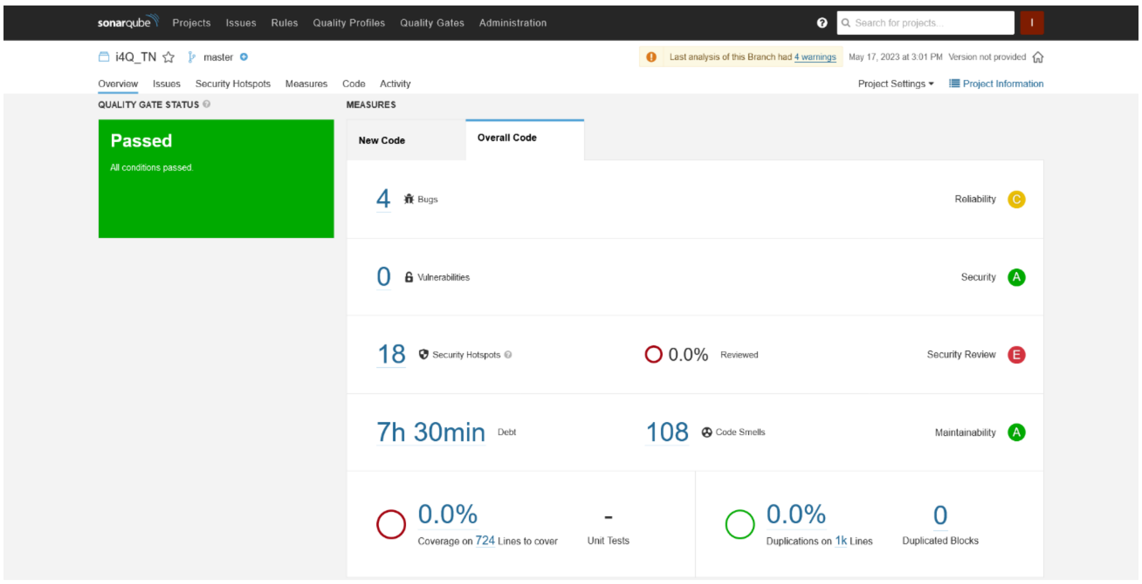
Regarding the **technical debt**, it is considerable but can be addressed in 4 days of works. Given that the number of **code smells** is over 200, an early refactoring of the code should be planned in order to minimise these metrics.

In terms of **coverage** and the number of **unit tests**, at present no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the component.

Finally, for the number of **duplicated lines and blocks** of code, the reported data indicates that there is a slight duplication in the code, but this is not very worrying, as it represents 2.7% of the total number of lines and only a total of 6 blocks of code. If possible, a code review is recommended to reduce the amount of duplicity, but this task is not urgent and can be done later.

5.3 i4Q^{TN}: Trusted Networks with Wireless and Wired Industrial Interfaces

This subsection analyses the source code quality of the *Trusted Networks with Wireless and Wired Industrial Interfaces* (i4Q^{TN}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



i4Q_TN master

Last analysis of this Branch had 4 warnings May 17, 2023 at 3:01 PM Version not provided

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

4 Bugs Reliability C

0 Vulnerabilities Security A

18 Security Hotspots 0.0% Reviewed Security Review E

7h 30min Debt 108 Code Smells Maintainability A

0.0% Coverage on 724 Lines to cover - Unit Tests

0.0% Duplications on 1k Lines 0 Duplicated Blocks

Figure 10. SonarQube report for the i4Q^{TN} Solution

Currently, this solution has a small number of **bugs**. Nevertheless, as these could lead to an erratic behaviour of the solution in the future, it is proposed to review and refactor the source code as soon as possible.

In the case of **vulnerabilities**, at the moment none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, 18 code snippets suspected to contain vulnerable code have been found, therefore, an early review of these snippets should be performed to verify the existence or not of vulnerable code.

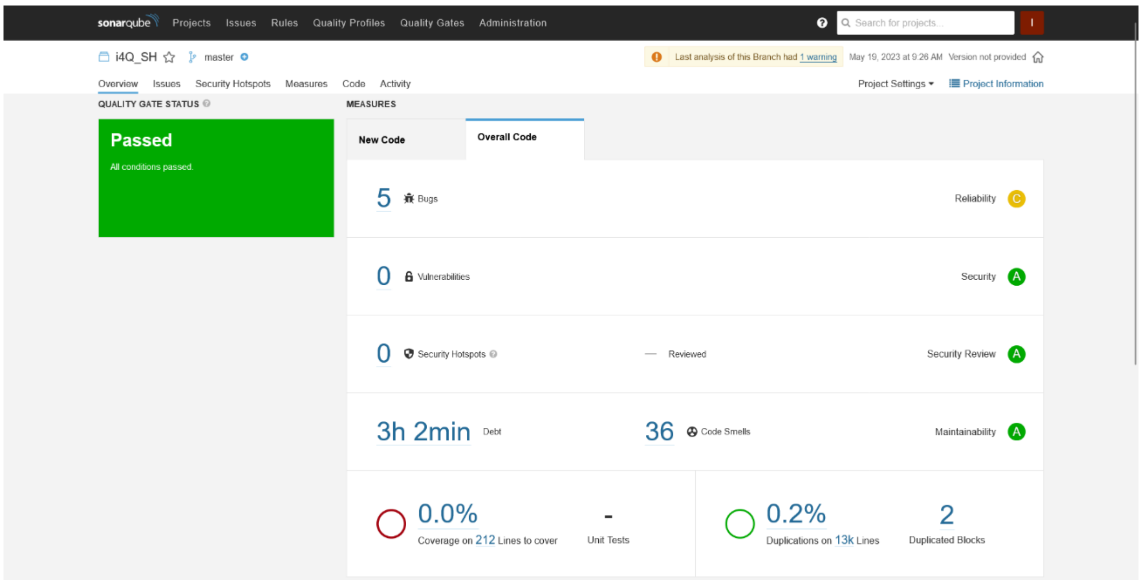
Regarding the **technical debt**, it is considerable but can be addressed in one or two days of work. Given that the number of **code smells** is over a hundred, an early refactoring of the code should be planned in order to minimise these metrics.

In terms of **coverage** and the number of **unit tests**, at present no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, no **duplications of lines or blocks** have been found, which indicates that the implemented code makes a good reuse of the defined elements, avoiding unnecessary code repetitions.

5.4 i4Q^{SH}: IIoT Security Handler

This subsection analyses the source code quality of the *IIoT Security Handler* (i4Q^{SH}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



Last analysis of this Branch had 1 warning May 19, 2023 at 9:26 AM Version not provided

QUALITY GATE STATUS

Passed
All conditions passed.

MEASURES

5 Bugs — Reliability C

0 Vulnerabilities — Security A

0 Security Hotspots — Reviewed — Security Review A

3h 2min Debt — 36 Code Smells — Maintainability A

0.0% Coverage on 212 Lines to cover — - Unit Tests

0.2% Duplications on 13k Lines — 2 Duplicated Blocks

Figure 11. SonarQube report for the i4Q^{SH} Solution

At the present time, this solution has a small number of **bugs**. However, as these could lead to an erratic behaviour of the solution in the future, it is proposed to review and refactor the source code as soon as possible.

In the case of **vulnerabilities**, currently none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, no fragments suspected of containing vulnerable code have been found either. This reinforces the idea that, at present, the code of this solution is up to date and does not contain any problem compromising its security, but it is important to be aware that this situation could change in the future.

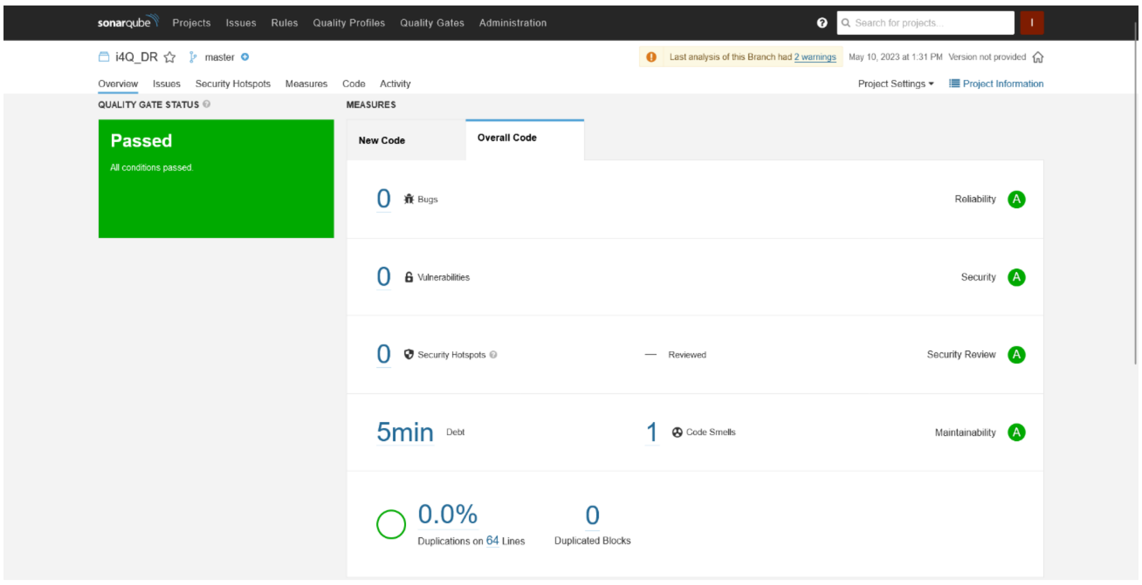
Regarding the **technical debt**, this is considerable but can be addressed in one working day. The number of **code smells** is less than forty but, in future code refactorings, the correction of these should be addressed to reduce this number to the minimum.

In terms of **coverage** and the number of **unit tests**, at present no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, the quantity of **duplicated lines and blocks** of code in this solution is minimal (0.2% of the total number of lines and 2 blocks of code). This indicates that the implemented code makes a good reuse of the defined elements, so it is not necessary to perform a refactoring focused on the correction of duplicities.

5.5 i4Q^{DR}: Data Repository

This subsection analyses the source code quality of the *Data Repository* (i4Q^{DR}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



Last analysis of this Branch had 2 warnings May 10, 2023 at 1:31 PM Version not provided

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

0 Bugs — Reliability A

0 Vulnerabilities — Security A

0 Security Hotspots — Reviewed — Security Review A

5min Debt — 1 Code Smells — Maintainability A

0.0% Duplications on 64 Lines — 0 Duplicated Blocks

Figure 12. SonarQube report for the i4Q^{DR} Solution

Currently, this solution does not have any **bug**, which means that no erratic behaviour is expected in the solution due to the presence of errors in the source code. For the moment, it is not necessary to perform a refactoring focused on fixing bugs, but it is important to be alert and be careful not to introduce new ones in the future.

In the case of **vulnerabilities**, at the moment none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, no fragments suspected of containing vulnerable code have been found either. This reinforces the idea that, at present, the code of this solution is up to date and does not contain any problem compromising its security, but it is important to be aware that this situation could change in the future.

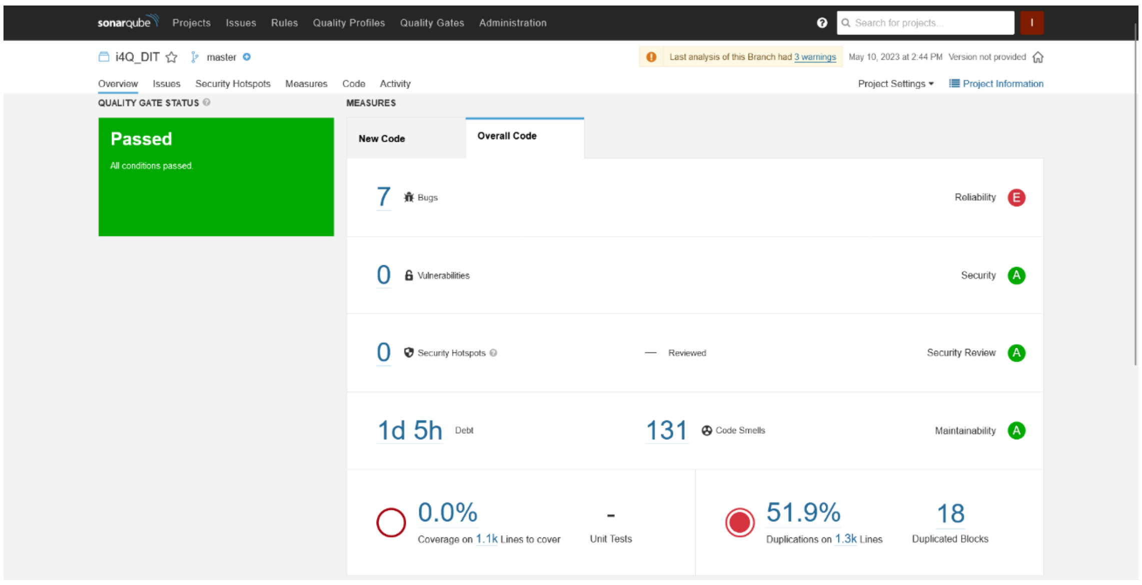
Regarding the **technical debt**, this can be solved in a small amount of time, as only one **code smell** has been found in the source code. As far as possible, it is recommended to try to solve this, but it is not necessary to perform a refactoring focused on the correction of code smells.

In terms of **coverage** and the number of **unit tests**, currently no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, no **duplications of lines or blocks** have been found, which indicates that the implemented code makes a good reuse of the defined elements, avoiding unnecessary code repetitions.

5.6 i4Q^{DIT}: Data Integration and Transformation Services

This subsection analyses the source code quality of the *Data Integration and Transformation Services* (i4Q^{DIT}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



Last analysis of this Branch had 3 warnings May 10, 2023 at 2:44 PM Version not provided

QUALITY GATE STATUS

Passed
All conditions passed.

MEASURES

7 Bugs — Reliability E

0 Vulnerabilities — Security A

0 Security Hotspots — Reviewed — Security Review A

1d 5h Debt — 131 Code Smells — Maintainability A

0.0% Coverage on 1.1k Lines to cover — - Unit Tests

51.9% Duplications on 1.3k Lines — 18 Duplicated Blocks

Figure 13. SonarQube report for the i4Q^{DIT} Solution

Currently, this solution has a small number of **bugs**. However, as these could lead to an erratic behaviour of the solution in the future, it is proposed to review and refactor the source code as soon as possible.

In the case of **vulnerabilities**, at the present time none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, no fragments suspected of containing vulnerable code have been found either. This reinforces the idea that, at the moment, the code of this solution is up to date and does not contain any problem compromising its security, but it is important to be aware that this solution could change in the future.

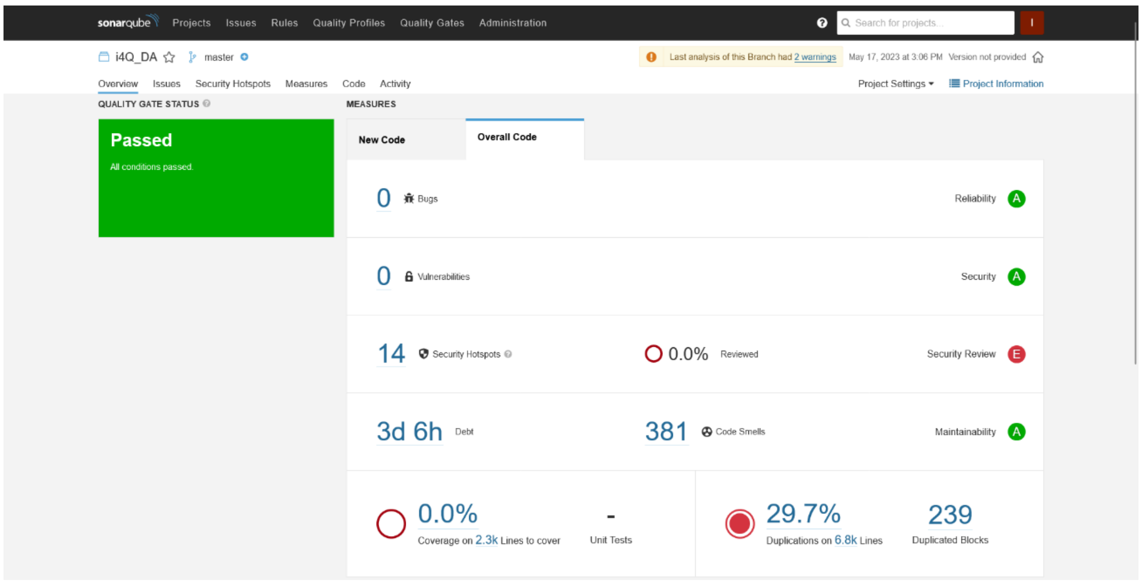
Regarding the **technical debt**, it is considerable but can be addressed in one or two days of work. Given that the number of **code smells** is over a hundred, an early refactoring of the code should be planned in order to minimise these metrics.

In terms of **coverage** and the number of **unit tests**, currently no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, the number of **duplicated lines and blocks** of code in this solution is very high. This can be due to two reasons. The first reason is the use of imported libraries or components that already contain a high number of duplicates. The other reason is that most of these duplicates have been introduced during the implementation of the solution. In any case, a review and refactoring of the solution is recommended in order to reduce these metrics as far as possible.

5.7 i4Q^{DA}: Services for Data Analytics

This subsection analyses the source code quality of the *Services for Data Analytics* (i4Q^{DA}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



Last analysis of this Branch had 2 warnings   May 17, 2023 at 3:06 PM   Version not provided

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code | Overall Code

0 Bugs — Reliability A

0 Vulnerabilities — Security A

14 Security Hotspots — 0.0% Reviewed — Security Review E

3d 6h Debt — 381 Code Smells — Maintainability A

0.0% Coverage on 2.3k Lines to cover — - Unit Tests

29.7% Duplications on 6.8k Lines — 239 Duplicated Blocks

Figure 14. SonarQube report for the i4Q^{DA} Solution

Currently, this solution does not have any **bug**, which means that no erratic behaviour is expected in the solution due to the presence of errors in the source code. For the moment, it is not necessary to perform a refactoring focused on fixing bugs, but it is important to be alert and be careful not to introduce new ones in the future.

In the case of **vulnerabilities**, by now none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, 14 code snippets suspected to contain vulnerable code have been found, therefore, an early review of these snippets should be performed to verify the existence or not of vulnerable code.

Regarding the **technical debt**, almost 400 **code smells** have been reported, which is a very high quantity. Solving this is not an impossible task, but it will require 3-4 days of work. Due to the high numbers reported, it would be highly recommended to review and refactor the solution's source code in order to reduce the values of these metrics.

In terms of **coverage** and the number of **unit tests**, at the moment no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, the quantity of **duplicated lines and blocks** of code in this solution is very high. As discussed with the solution provider, this is because the solution is based on existing open-source tools such as Apache Superset, Apache Druid, and Apache Kafka. Therefore, many of these duplicities come from the use of these tools and, for that reason, it will be difficult to minimise these metrics.

5.8 i4Q^{BDA}: Big Data Analytics Suite

This solution is responsible for creating a deployment bundle that contains the i4Q^{DA} Solution and all the necessary technologies to perform an optimised deployment based on the hardware specification of the pilot infrastructure. In this way, the solution does not perform a software implementation, as it is a deployment script. Therefore, it has been agreed with the solution provider that it will not be deployed in SonarQube.

5.9 i4Q^{AD}: Analytics Dashboard

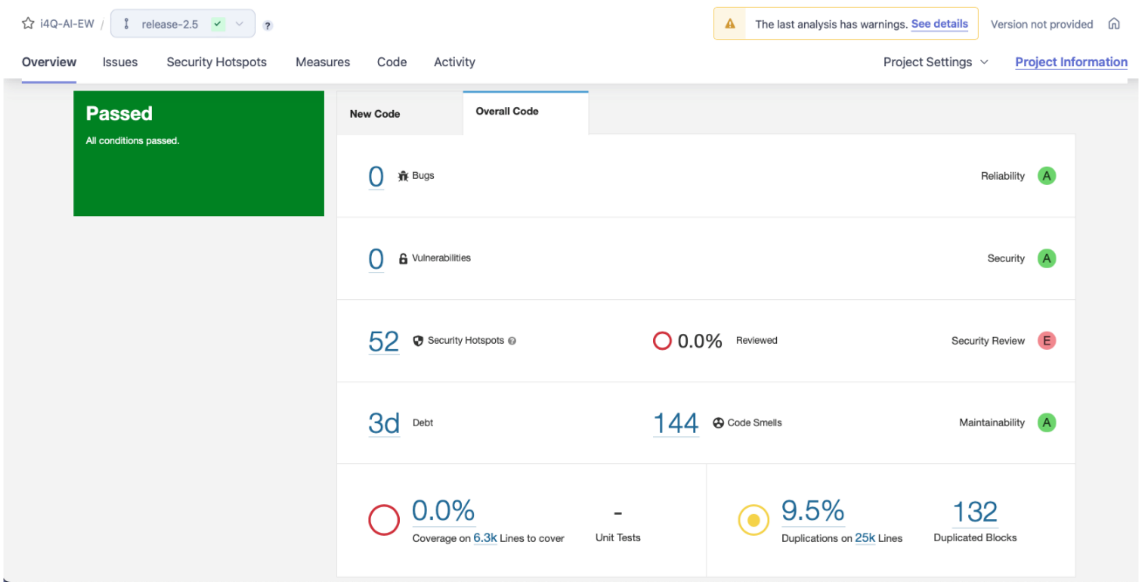
The implementation of this solution is based on the use of existing open-source tools such as Apache Superset, Apache Druid, and Apache Kafka. For this reason, the solution provider considers that it is not necessary to analyse the quality of the software implemented in this solution by using SonarQube.

5.10 i4Q^{AI}: AI Models Distribution to the Edge

Since the *AI Models Distribution to the Edge* (i4Q^{AI}) Solution is closely related to the *Edge Workloads Placement and Deployment* (i4Q^{EW}) Solution, as the code of the i4Q^{AI} is combined with the code of the i4Q^{EW} Solution, it has been agreed with the solution provider to analyse only the code quality of the i4Q^{EW} Solution.

5.11 i4Q^{EW}: Edge Workloads Placement and Deployment

This subsection analyses the source code quality of the *Edge Workloads Placement and Deployment* (i4Q^{EW}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



Passed
All conditions passed.

0 Bugs — Reliability A
0 Vulnerabilities — Security A
52 Security Hotspots — 0.0% Reviewed — Security Review E
3d Debt — 144 Code Smells — Maintainability A
0.0% Coverage on 6.3k Lines to cover — - Unit Tests
9.5% Duplications on 25k Lines — 132 Duplicated Blocks

Figure 15. SonarQube report for the i4Q^{EW} Solution

Currently, this solution does not have any **bug**, which means that no erratic behaviour is expected in the solution due to the presence of errors in the source code. For the moment, it is not necessary to perform a refactoring focused on fixing bugs, but it is important to be alert and be careful not to introduce new ones in the future.

In the case of **vulnerabilities**, by now none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, a total of 52 snippets suspected of containing vulnerable code have been reported in this solution. This quantity is very worrying, as it indicates that there is a high probability that the code of this solution is susceptible to security-related problems. For this reason, an urgent review of these code fragments should be undertaken to verify whether they represent a threat or not and, if they do, corrective measures should be implemented to minimise the risk of suffering an attack in the future.

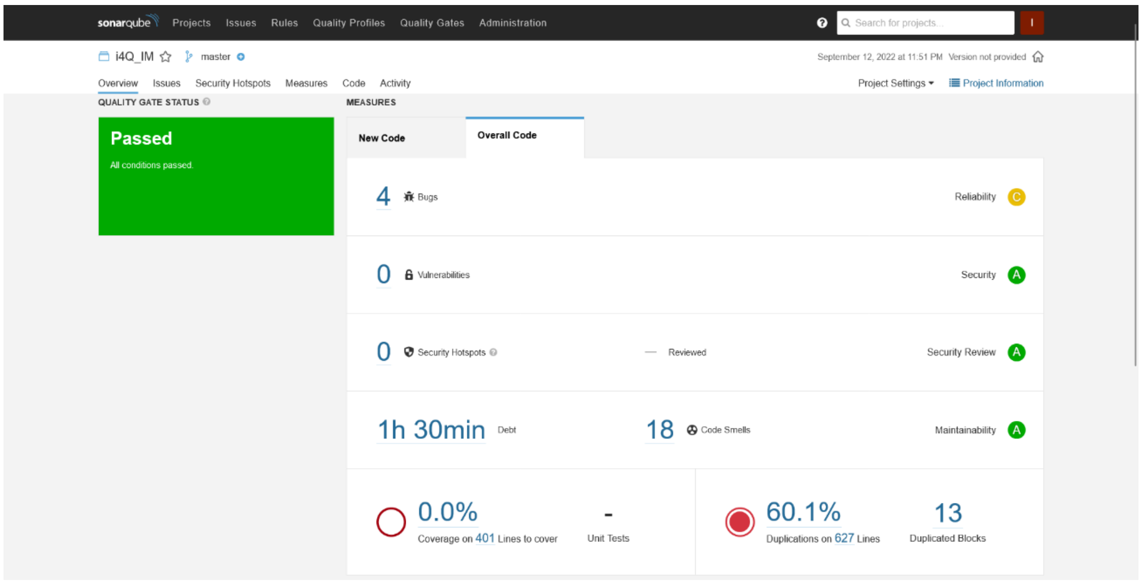
Regarding the **technical debt**, almost 150 code smells have been reported. This amount is considerable but can be addressed in 3-4 days of work. Given that the number of reported code smells exceeds a hundred, an early refactoring of the code should be planned to minimise these metrics.

In terms of **coverage** and the number of **unit tests**, at present no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, for the quantity of **duplicated lines and blocks** of code, the reported data indicates that there is a slight duplication in the code, but this is not very worrying, as it represents less than 10% of the total number of lines and a total of 132 blocks of code. If possible, a code review is recommended to reduce the amount of duplicity, but this task is not urgent and can be done later.

5.12 i4Q^{IM}: Infrastructure Monitoring

This subsection analyses the source code quality of the *Infrastructure Monitoring* (i4Q^{IM}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



i4Q_IM master

September 12, 2022 at 11:51 PM Version not provided

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code | Overall Code

4 Bugs — Reliability C

0 Vulnerabilities — Security A

0 Security Hotspots — Reviewed — Security Review A

1h 30min Debt — 18 Code Smells — Maintainability A

0.0% Coverage on 401 Lines to cover — - Unit Tests

60.1% Duplications on 627 Lines — 13 Duplicated Blocks

Figure 16. SonarQube report for the i4Q^{IM} Solution

At the present time, this solution has a small number of **bugs**. However, as these could lead to an erratic behaviour of the solution in the future, it is proposed to review and refactor the source code as soon as possible.

In the case of **vulnerabilities**, currently none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, no fragments suspected of containing vulnerable code have been found either. This reinforces the idea that, at present, the code of this solution is up to date and does not contain any problem compromising its security, but it is important to be aware that this situation could change in the future.

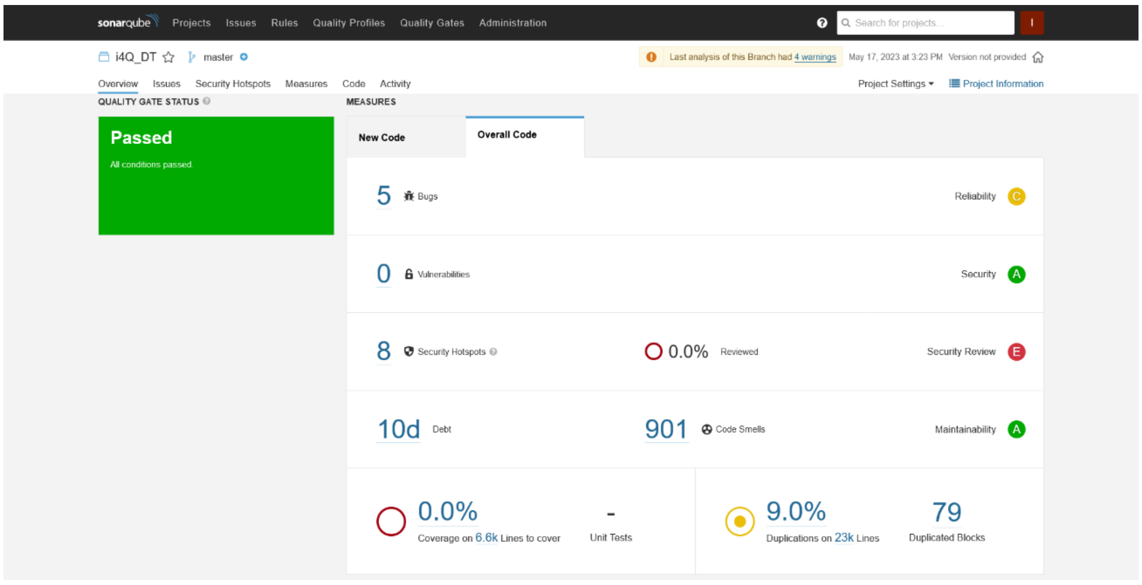
Regarding the **technical debt**, the resolution of the **code smells** found should not be very expensive in terms of time, as less than 20 have been reported. As far as possible, it is recommended to try to solve them, but it is not necessary to perform a refactoring focused on the correction of code smells.

In terms of **coverage** and the number of **unit tests**, currently no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, the number of **duplicated lines and blocks** of code in this solution is very high. This can be due to two reasons. The first reason is the use of imported libraries or components that already contain a high number of duplicates. The other reason is that most of these duplicates have been introduced during the implementation of the solution. In any case, a review and refactoring of the solution is recommended in order to reduce these metrics as far as possible.

5.13 i4Q^{DT}: Digital Twin Simulation Services

This subsection analyses the source code quality of the *Digital Twin Simulation Services* (i4Q^{DT}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



Last analysis of this Branch had 4 warnings   May 17, 2023 at 3:23 PM   Version not provided

QUALITY GATE STATUS

Passed
All conditions passed

MEASURES

New Code | Overall Code

5 Bugs — Reliability C

0 Vulnerabilities — Security A

8 Security Hotspots — 0.0% Reviewed — Security Review E

10d Debt — 901 Code Smells — Maintainability A

0.0% Coverage on 6.6k Lines to cover — - Unit Tests

9.0% Duplications on 23k Lines — 79 Duplicated Blocks

Figure 17. SonarQube report for the i4Q^{DT} Solution

Currently, this solution has a small number of **bugs**. However, as these could lead to an erratic behaviour of the solution in the future, it is proposed to review and refactor the source code as soon as possible.

In the case of **vulnerabilities**, at the moment none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, 8 code snippets suspected to contain vulnerable code have been found, therefore, an early review of these snippets should be performed to verify the existence or not of vulnerable code.

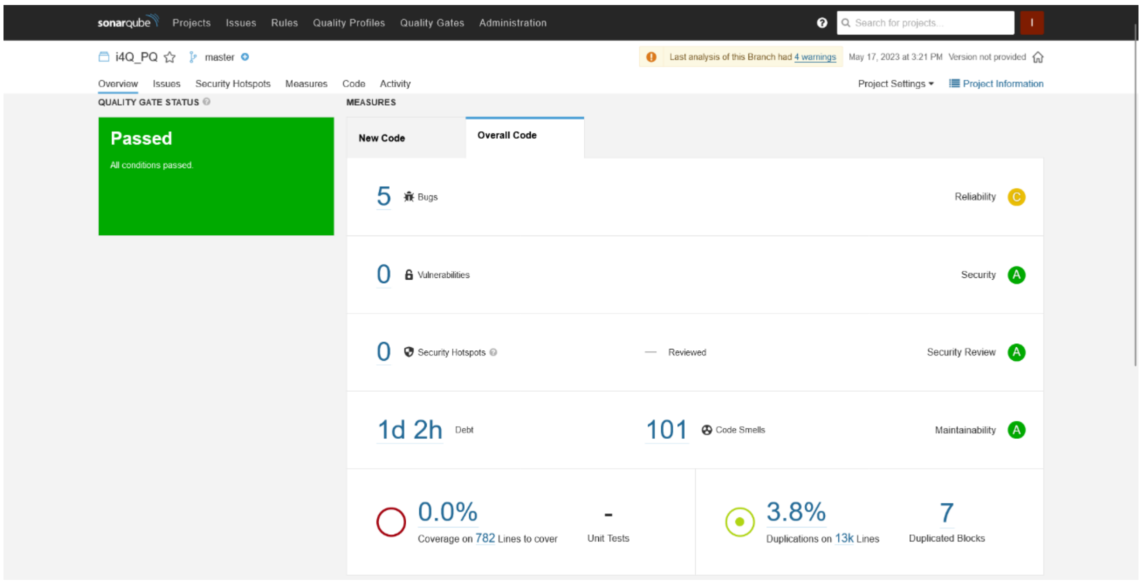
Regarding **technical debt**, this represents a very high number of hours, as large number of **code smells** have been reported. An early refactoring of the code should be addressed in order to reduce these metrics. However, in the case of wanting to tackle this task, it should be taken into account that more than 1 week of work will be necessary to resolve all the code smells detected. For this reason, it is proposed to undertake this task by phases or iterations, so that the task can be tackled in a manageable way.

In terms of **coverage** and the number of **unit tests**, at present no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solutions.

Finally, for the number of **duplicated lines and blocks** of code, the reported data indicates that there is a slight duplication in the code, but this is not very worrying, as it represents less than 10% of the total number of lines and a total of 79 blocks of code. If possible, a code review is recommended to reduce the amount of duplicity, but this task is not urgent and can be done later.

5.14 i4Q^{PQ}: Data-Driven Continuous Process Qualification

This subsection analyses the source code quality of the *Data-Driven Continuous Process Qualification* (i4Q^{PQ}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



Last analysis of this Branch had 4 warnings May 17, 2023 at 3:21 PM Version not provided

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

5 Bugs — Reliability C

0 Vulnerabilities — Security A

0 Security Hotspots — Reviewed — Security Review A

1d 2h Debt — 101 Code Smells — Maintainability A

0.0% Coverage on 782 Lines to cover — - Unit Tests

3.8% Duplications on 13k Lines — 7 Duplicated Blocks

Figure 18. SonarQube report for the i4Q^{PQ} Solution

At the moment, this solution has a small number of **bugs**. However, as these could lead to an erratic behaviour of the solution in the future, it is proposed to review and refactor the source code as soon as possible.

In the case of **vulnerabilities**, currently none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, no fragments suspected of containing vulnerable code have been found either. This reinforces the idea that, at present, the code of this solution is up to date and does not contain any problem compromising its security, but it is important to be aware that this situation could change in the future.

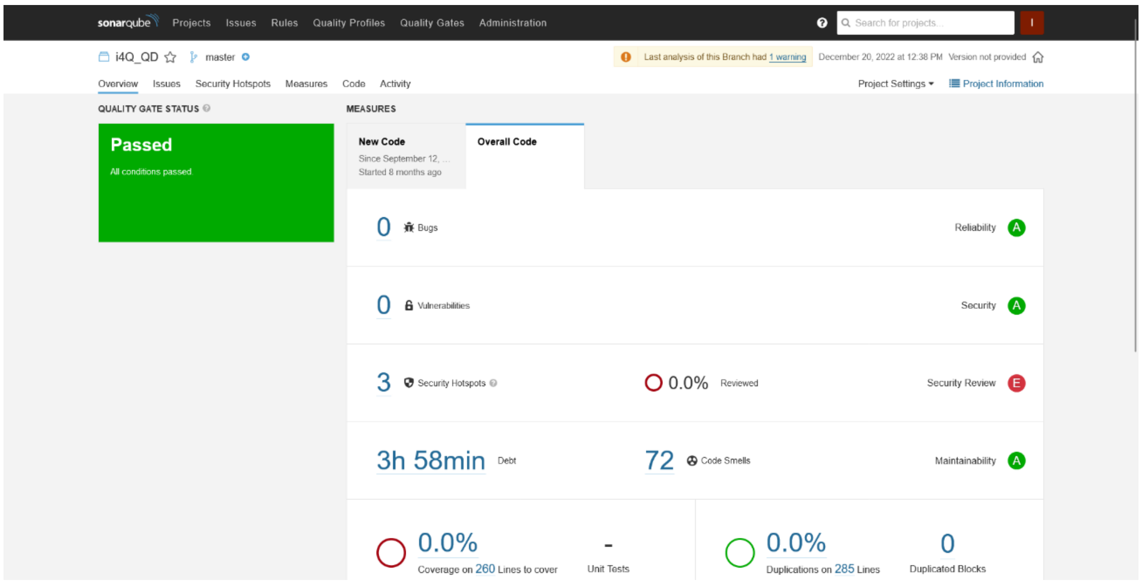
Regarding the **technical debt**, it is considerable but can be addressed in one or two days of work. Given that the number of **code smells** is over a hundred, an early refactoring of the code should be planned in order to minimise these metrics.

In terms of **coverage** and the number of **unit tests**, at present no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, for the number of **duplicated lines and blocks** of code, the reported data indicates that there is a slight duplication in the code, but this is not very worrying, as it represents 3.8% of the total number of lines and a total of 7 blocks of code. If possible, a code review is recommended to reduce the amount of duplicity, but this task is not urgent and can be done later.

5.15 i4Q^{QD}: Rapid Quality Diagnosis

This subsection analyses the source code quality of the *Rapid Quality Diagnosis* (i4Q^{QD}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



Last analysis of this Branch had 1 warning December 20, 2022 at 12:38 PM Version not provided

QUALITY GATE STATUS

Passed
All conditions passed.

MEASURES

New Code
Since September 12, ...
Started 8 months ago

Overall Code

0 Bugs — Reliability A
0 Vulnerabilities — Security A
3 Security Hotspots — 0.0% Reviewed — Security Review E
3h 58min Debt — 72 Code Smells — Maintainability A
0.0% Coverage on 260 Lines to cover — - Unit Tests
0.0% Duplications on 285 Lines — 0 Duplicated Blocks

Figure 19. SonarQube report for the i4Q^{QD} Solution

Currently, this solution does not have any **bug**, which means that no erratic behaviour is expected in the solution due to the presence of errors in the source code. For the moment, it is not necessary to perform a refactoring focused on fixing bugs, but it is important to be alert and be careful not to introduce new ones in the future.

In the case of **vulnerabilities**, by now, none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, 3 code snippets suspected to contain vulnerable code have been found, therefore, an early review of these snippets should be performed to verify the existence or not of vulnerable code.

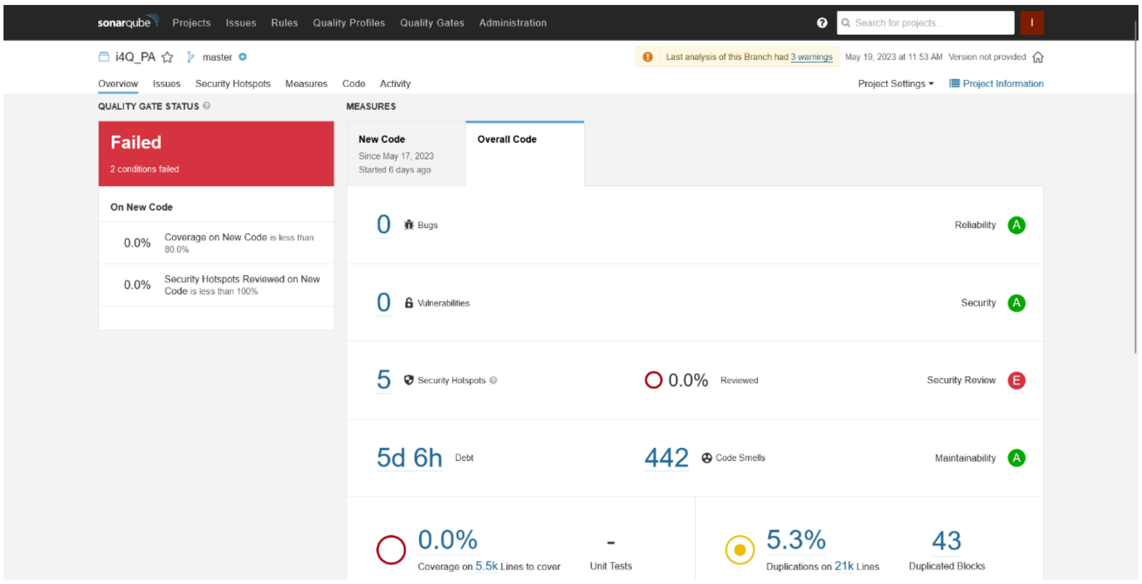
Regarding the **technical debt**, although it is considerable, it can be addressed, as the number of **code smells** is less than a hundred. However, an early refactoring of the code should be planned in order to minimise these metrics.

In terms of **coverage** and the number of **unit tests**, at the moment no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, no **duplications of lines or blocks** have been found, which indicates that the implemented code makes a good reuse of the defined elements, avoiding unnecessary code repetitions.

5.16 i4Q^{PA}: Prescriptive Analysis Tools

This subsection analyses the source code quality of the *Prescriptive Analysis Tools* (i4Q^{PA}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



Last analysis of this Branch had 3 warnings May 19, 2023 at 11:53 AM Version not provided

QUALITY GATE STATUS

Failed
2 conditions failed

On New Code

0.0% Coverage on New Code is less than 80.0%

0.0% Security Hotspots Reviewed on New Code is less than 100%

MEASURES

New Code Since May 17, 2023 Started 6 days ago | Overall Code

0 Bugs — Reliability A

0 Vulnerabilities — Security A

5 Security Hotspots — 0.0% Reviewed — Security Review E

5d 6h Debt — 442 Code Smells — Maintainability A

0.0% Coverage on 5.5k Lines to cover — - Unit Tests — 5.3% Duplications on 21k Lines — 43 Duplicated Blocks

Figure 20. SonarQube report for the i4Q^{PA} Solution

First of all, although it appears in red and with the message “Failed”, the execution of SonarQube in this solution is not failing. This message is displayed to warn that, after a second analysis of the source code, no unit tests have been implemented.

In terms of **bugs**, at present this solution does not have any, which means that no erratic behaviour is expected in the solution due to the presence of errors in the source code. For the moment, it is not necessary to perform a refactoring focused on fixing bugs, but it is important to be alert and be careful not to introduce new ones in the future.

In the case of **vulnerabilities**, by now, none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, 5 code snippets suspected to contain vulnerable code have been found, therefore, an early review of these snippets should be performed to verify the existence or not of vulnerable code.

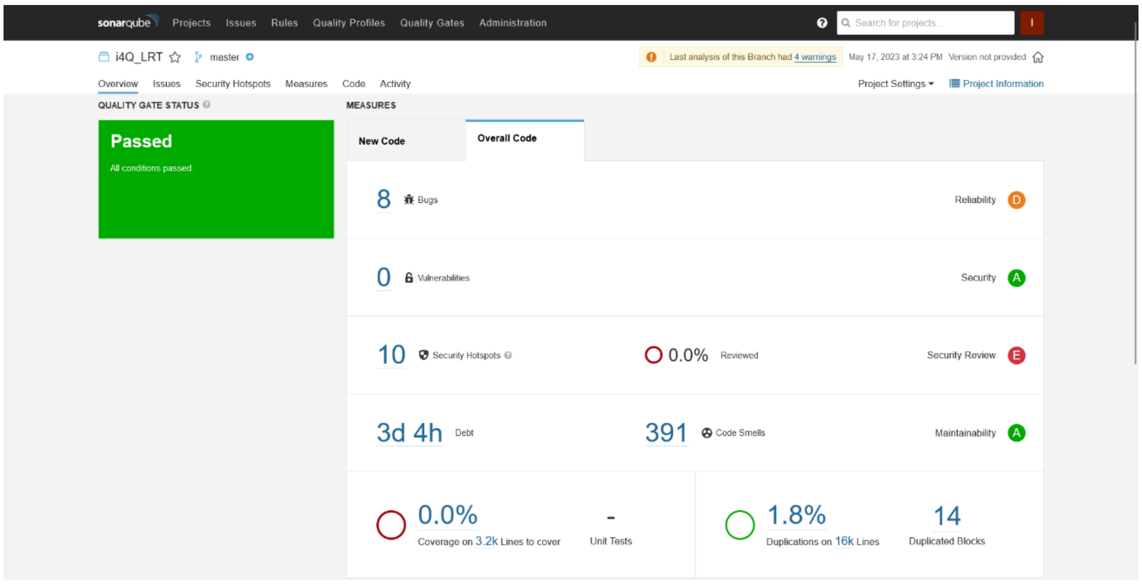
Regarding the **technical debt**, more than 400 **code smells** have been reported, which is a very high quantity. Solving this is not an impossible task, but it will require between 5 and 6 days of work. Due to the high numbers reported, it would be highly recommended to review and refactor the solution's source code in order to reduce the values of these metrics.

Concerning **coverage** and the number of **unit tests**, at present no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, for the number of **duplicated lines and blocks** of code, the reported data indicates that there is a slight duplication in the code, but this is not very worrying, as it represents 5.3% of the total number of lines and a total of 43 blocks of code. If possible, a code review is recommended to reduce the amount of duplicity, but this task is not urgent and can be done later.

5.17 i4Q^{LRT}: Manufacturing Line Reconfiguration Toolkit

This subsection analyses the source code quality of the *Manufacturing Line Reconfiguration Toolkit* (i4Q^{LRT}) Solution according to the report generated by the SonarQube tool. The following image shows the results for each of the metrics explained at the beginning of section 5 and, at the end of this subsection, the results obtained are discussed.



i4Q_LRT master

Last analysis of this Branch had 4 warnings May 17, 2023 at 3:24 PM Version not provided

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

QUALITY GATE STATUS

Passed

All conditions passed

MEASURES

New Code Overall Code

8 Bugs — Reliability D

0 Vulnerabilities — Security A

10 Security Hotspots — 0.0% Reviewed — Security Review E

3d 4h Debt — 391 Code Smells — Maintainability A

0.0% Coverage on 3.2k Lines to cover — - Unit Tests

1.8% Duplications on 16k Lines — 14 Duplicated Blocks

Figure 21. SonarQube report for the i4Q^{LRT} Solution

At the moment, this solution has a small number of **bugs**. However, as these could lead to an erratic behaviour of the solution in the future, it is proposed to review and refactor the source code as soon as possible.

In the case of **vulnerabilities**, currently none have been detected. This indicates that, in principle, the implemented code does not contain any security problem.

For the **security hotspots**, 10 code snippets suspected to contain vulnerable code have been found, therefore, an early review of these snippets should be performed to verify the existence or not of vulnerable code.

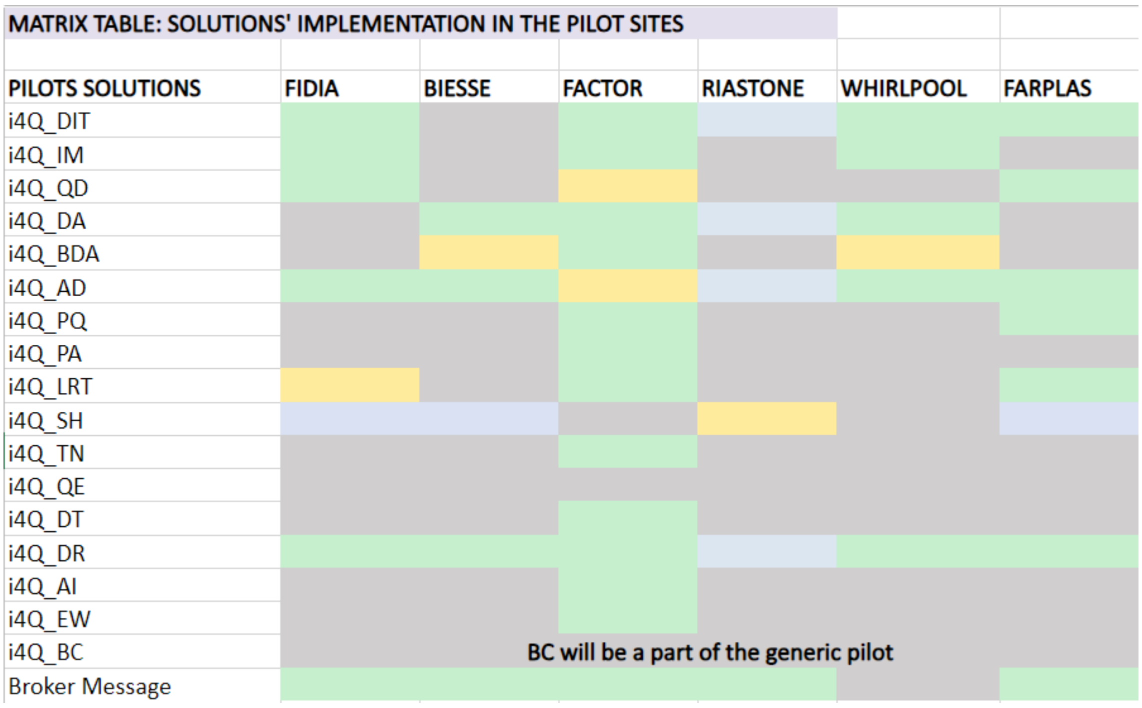
Regarding the **technical debt**, almost 400 **code smells** have been reported, which is a very high quantity. Solving this is not an impossible task, but it will require 3-4 days of work. Due to the high numbers reported, it would be highly recommended to review and refactor the solution's source code in order to reduce the values of these metrics.

In terms of **coverage** and the number of **unit tests**, at present no tests have been performed, so its coverage represents 0%. It would be highly recommended that future refactoring of the code also addresses the inclusion of unit tests to check the correct functioning of the solution.

Finally, for the number of **duplicated lines and blocks** of code, the reported data indicates that there is a slight duplication in the code, but this is not very worrying, as it represents 1.8% of the total number of lines and a total of 14 blocks of code. If possible, a code review is recommended to reduce the amount of duplicity, but this task is not urgent and can be done later.

6 Analysis of results

After explaining the infrastructure of each one of the pilots, specifying the configuration of the solutions, and defining the integration between them in sections 2, 3 and 4, in this section, a brief analysis of the current state of deployment and integration of the solutions in the infrastructure of the pilots will be made. To do so, a colour-coded matrix has been generated in order to distinguish the status of each solution in each one of the pilots.



MATRIX TABLE: SOLUTIONS' IMPLEMENTATION IN THE PILOT SITES

PILOTS SOLUTIONS	FIDIA	BIESSE	FACTOR	RIASTONE	WHIRLPOOL	FARPLAS
i4Q_DIT	🟩	⬜	🟩	🟦	🟩	🟩
i4Q_IM	🟩	⬜	🟩	⬜	🟩	⬜
i4Q_QD	🟩	⬜	🟨	⬜	⬜	🟩
i4Q_DA	⬜	🟩	🟩	🟦	🟩	⬜
i4Q_BDA	⬜	🟨	🟩	⬜	🟨	⬜
i4Q_AD	🟩	🟩	🟨	🟦	🟩	🟩
i4Q_PQ	⬜	⬜	🟩	⬜	⬜	🟩
i4Q_PA	⬜	⬜	🟩	⬜	⬜	⬜
i4Q_LRT	🟨	⬜	🟩	⬜	⬜	🟩
i4Q_SH	🟦	🟦	⬜	🟨	⬜	🟦
i4Q_TN	⬜	⬜	🟩	⬜	⬜	⬜
i4Q_QE	⬜	⬜	⬜	⬜	⬜	⬜
i4Q_DT	⬜	⬜	🟩	⬜	⬜	⬜
i4Q_DR	🟩	🟩	🟩	🟦	🟩	🟩
i4Q_AI	⬜	⬜	🟩	⬜	⬜	⬜
i4Q_EW	⬜	⬜	🟩	⬜	⬜	⬜
i4Q_BC	⬜	⬜	**BC will be a part of the generic pilot**	⬜	⬜	⬜
Broker Message	🟩	🟩	🟩	🟩	⬜	🟩

Figure 22. Solutions deployment and integration status matrix

- A **grey cell** indicates that the solution in question is not deployed in the infrastructure of that pilot.
- A **yellow cell** means that the solution is expected to be deployed in that pilot but, for some reason, it has not been deployed yet.
- A **green cell** represents that a given solution has been successfully deployed in the pilot infrastructure but has not been integrated yet with the solutions mentioned in section 4.
- A **blue cell** denotes that the solution in question has been successfully deployed in the pilot infrastructure and, in addition, its integration with the solutions specified in section 4 has been successfully completed.

As can be seen in the picture above, in the FIDIA pilot most of the solutions are already deployed. However, only the i4Q^{SH} Solution has been successfully integrated.

In the case of BIESSE's pilot, the situation is similar, with only one solution still to be deployed. However, as with FIDIA, only the i4Q^{SH} Solution has completed its integration with the other.

In the WHIRLPOOL pilot, all but one of the planned solutions are deployed. In contrast, in this pilot, no solutions have been integrated yet.

Regarding the FACTOR pilot, it has suffered some delays in the solutions deployment, due to problems with the infrastructure. As a result, there are still quite a few solutions to be deployed.

For the RIASTONE pilot, there are a couple of solutions that have not been deployed yet. Despite this, it is the most advanced, with 3 solutions that have already been successfully integrated.

Finally, FARPLAS's pilot also has 2 solutions still to be deployed and, as in FIDIA and BIESSE, the only solution that has been deployed is the i4Q^{SH}.

7 Conclusions

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

First, the characteristics of the deployment infrastructures of each i4Q Pilot have been analysed. To this end, industrial partners have provided some information about their pilots. In particular, the information provided explains the use cases or KPIs they are trying to cover, the type of infrastructure deployed, the operating system used, the storage size, CPU details, the amount of RAM, and other relevant details not discussed above.

Then, the characteristics of the i4Q Solutions have been discussed. For this purpose, solution providers have provided some information about their solutions. Specifically, the information provided explains the type of infrastructure they need, what their solution does, the environment in which it will be deployed, the amount of storage needed, software technologies and dependencies that must be installed, as well as the minimum requirements that the pilot infrastructure must fulfil for the solution to be deployed without problems.

Subsequently, the updated versions of the i4Q Pilot pipelines have been presented, and the modifications made to each of them have been explained in a justified manner. Then, the solution integration that will be addressed during the period of time between M31 and M36 have been analysed, as well as the information they will exchange and the communication mechanism they will use.

Afterwards, SonarQube tool has been used to review the source code quality of the different i4Q Solutions. This tool generates a report with metrics to analyse the behaviour of the solutions in terms of bugs, vulnerabilities and security issues, code maintainability, unit tests performed, and amount of code covered by them, and code duplications. With the information provided by these metrics, solution providers will be proposed to refactor their code starting from M31 in order to improve the quality of their code and avoid future problems.

Finally, the current status of the i4Q Solutions regarding the deployment and integration phases in each of the i4Q Pilots is shown. In general, the deployment phase has moved a little slower than expected and thus is a little behind schedule. However, this is not a significant cause for concern and can be solved during the period between M31 and M36.