D2.5 – Functional viewpoint

WP2 – DESIGN: i4Q Framework Design

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	D2.5 – Functional viewpoint		
WORK PACKAGE	WP2 – DESIGN: i4Q Framework Design		
DATE OF DELIVERY	CONTRACTUAL: May. 2020	ACTUAL	Sep. 2020
NATURE	Report	DISSEMINATION LEVEL	Public
LEAD BENEFICIARY	UPV		
RESPONSIBLE AUTHOR	-		
CONTRIBUTIONS FROM	CERTH (3PM), ENG (3PM), ITI (5PM), IKER (0.5PM), BIBA (2PM), UPV (6PM), UNI (2PM).		
TARGET AUDIENCE	1) i4Q Project partners; 2) industrial community; 3) other H2020 funded projects; 4) scientific community		
DELIVERABLE CONTEXT/ DEPENDENCIES	This document is functional viewpoint. Its relationship to other documents is as follows: • D1.9 Title: Requirements Analysis and Functional Specification v2 • D2.3 Title: i4Q Business viewpoint • D2.4 Title: i4Q Usage viewpoint • D2.6 Title: i4Q Implement viewpoint		
EXTERNAL ANNEXES/ SUPPORTING DOCUMENTS	None		
READING NOTES	None		
ABSTRACT	The main objective of deliverable D2.5 is to focus on the functional components in i4Q solutions, their structure, interrelation, interfaces and interactions between them using the reference architecture as a framework. On the other hand, the deliverable reflects the relationships and interactions of the i4Q solutions with external elements of the environment to support the activities and usages of the overall system.		

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
0.1	14-May-2021	ToC	Table concepts	UPV
0.2	01-Jul-2021	Working version	Introduction and Functional components	UPV / ITI / CERTH / ENG and all Solutions Providers.
0.3	01-Sep-2021	Working version	Mapping user task, functional domains and Product Life-cycle	UPV / ITI
0.4	09-Sep-2021	1st draft	First Version	UPV
0.5	16-Sep-2021	Working version	Review	FARPLAS / IKER
0.6	21-Sep-2021	2nd draft	Second Version	UPV / ITI
1.0	30-Sep-2021	Final doc	Final quality check and issue of final document	CERTH

Disclaimer

Copyright message

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS/ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented reality
AS	Aerospace Standard
AWS	Amazon Web Services
CAD	Computer Aided Desing
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CEN	European Committee for Standardization
CENELEC	European Committee for Electrotechnical Standardization
CLI	Command Line Interface
CNC	Computer Numerical Control
CRM	Costumers Relationship Management
CSM	Contact Management System
CWA	CEN Workshop Agreement
DB	Database
DLT	Distributed ledger technology
DSS	Document Security System
EC	European Commission
ERP	Enterprise Resource Planning
EU	European Union
FMS	Flexible Manufacturing System
FMU	Functional Mock up Units
FTO	Freedom to Operate
GA	Grant Agreement
GPU	Graphics Processing Unit
gRPC	gRPC Remote Procedure Calls
GUI	Graphical User Interface
HDL	High-Definition LiDAR
HSM	Hardware Security Module
HTML	Hypertext Markup Language
IACS	Industrial Automation and Control Systems
ICT	Information and Communications Technology
IEEE	Institute of Electrical and Electronics Engineers
IIoT	Industrial Internet of Things
IMC	Innovation Management Committee
IP	Intellectual Property
IPR	Intellectual Property Rights
IIRA	Industrial Internet Reference Architecture
ISO	International Organization for Standardization
IT	Information Technology
IWSN	On Industrial Wireless Sensor Network
LM	Legal Manager
MES	Manufacture Execution System

MES	Manufacturing Execution System
ML	Model Life
MQTT	Message Queuing Telemetry Transport
MS	Milestone
NA	Not applicable
NOK	Not Okay
NPI	New Product Introduction
OEE	Overall Equipment Effectiveness
OEM	Original Equipment Manufacturer
OPE	Overall Production Effectiveness
OS	Operating System
PC	Project Coordinator
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PEDR	Plan for Exploitation and Dissemination of Results
PKI	Public Key Infrastructure
PLC	Programmable Logic Controllers
QC	Quality Control
QCM	Quality Control Manager
QoS	Quality of Service
RBAC	Role-based access control
REST API	RESTful API
RIDS	Reliable Industrial Data Services
RoT	Root of Trust
SDN	Software defined networks
SME	Small Medium Enterprise
SQL	Structured Query Language
SWOT	Strengths, Weaknesses, Opportunities, Threats
TCP	Fieldbus communication
TM	Technical Manager
TPM	Trusted Platform Module
TRL	Technical Rediness Level
TSN	Time Sensitive Networks
UA	Unified Architecture
UI	User Interface
UK	United Kingdom
VR	Virtual Reality
VRM	Value Roadmapping Method
WP	Workpackage
WPL	Workpackage leader
WSN	Wireless Sensors Technologies
ZDMP	Zero Defects Manufacturing Platform
5G	Fifth generation

Executive summary

Deliverable D2.5 functional viewpoint describes the functional components for the business processes of the 17 i4Q solutions, extending the defintions and SysML diagrams provided in deliverable D1.9. Furthermore, D1.9 provides a deeper understanding and analysis of the needs and requirements of the pilots, as well as a functional specification of the i4Q Solutions, whereas this deliverable focuses on the interactions between the different components, framed in the reference architecture. It also considers the interoperability of the solutions by defining functional interfaces and eliciting the corresponding requirements. This is an input for other design tasks in work package 2, especially for task T2.6 implementation viewpoint.

The procedure to structure the document is based on IIRA (ICC, 2019). In D2.5, the focus is on the functional viewpoint, but there are links to the usage viewpoint and within the project, the primary objective is to promote a shared understanding of the functionalities of the different components and how they work together to implement the features requested in the different use cases. For this reason, the open-source systems modelling language SysML is used to highlight the boundaries between components, relate the functional component to the reference architecture and specify the life-cycle stage where the functionality is used, based on the the diagrams and definitions developed in task T 1.9.

This way, the main results of this document are detailed descriptions of the functional components for each solution and the mapping of these components to the IIRA domains and the usage viewpoint. Finally, the document includes some sequence diagrams to provide a better understanding of the life-cycle of applications using i4Q solutions.

Therefore, the main results of this document are:

- The overview of the functional components for each i4Q solution, linked with the i4Q reference architecture and IIRA functional domains.
- Definition of the models and ontologies to be used in each communication flow within the defined architecture and proposed solutions.
- Mapping of the functional components to the usage model provided in the usage viewpoint.
- The overview of some different solutions for each life-cycles defined in AI Models and Data stage.

Document structure

Section 1 – Introduction: Explains the objectives of this document, its context within the IIRA document and the different viewpoints defined therein, and the methodology to be followed in the different sections.

Section 2 – Functional model: Extends the definitions of the functional components of the different i4Q solutions found in D1.9 to align it with the reference architecture and the different life-cycle stages defined for the i4Q solutions. Next, it deepens into each of the components to better describe their functionality. Finally, it completes the functional model by linking the functional components to the different IIRA functional domains.

Section 3 – Digital Model and Ontologies: Looks at the digital models and ontologies applied to different communication flows in the architecture for i4Q solutions, namely data, control and workload flows.

Section 4 – Mapping of Functional Components to User Tasks: Maps the different functional components of the solutions with each of the user tasks defined in the T2.4 and defines additional acceptance criteria for each functional component to be taken into account during the validation phase.

Section 5 – Mapping of Functional Components to product life-cycles stages: Taking into account the life-cycles suggested by IIRA, this section develops a series of sequence diagram to describe the life-cycle of i4Q solutions.

Section 6 – Conclusion: The Conclusion provides a summary of the objectives archived throughout the document.

1. Introduction

1.1. Context

This deliverable is part of a series of deliverables that describe the architecture of the i4Q solutions according to the viewpoints specified in the Industrial Internet Reference Architecture (IIRA). This industrial internet architecture framework provides a standard-based framework – primarily based on the ISO 42010 System Architecture Description (ISO 42010, 2011) - to guide the development of IIoT systems, using a value-driven approach. The basic idea underlying the ISO Architecture Description standard is to analyse and resolve the specific concerns in each of the viewpoints, and as a consequence, creating different architecture models that represent the system architecture from different perspectives.

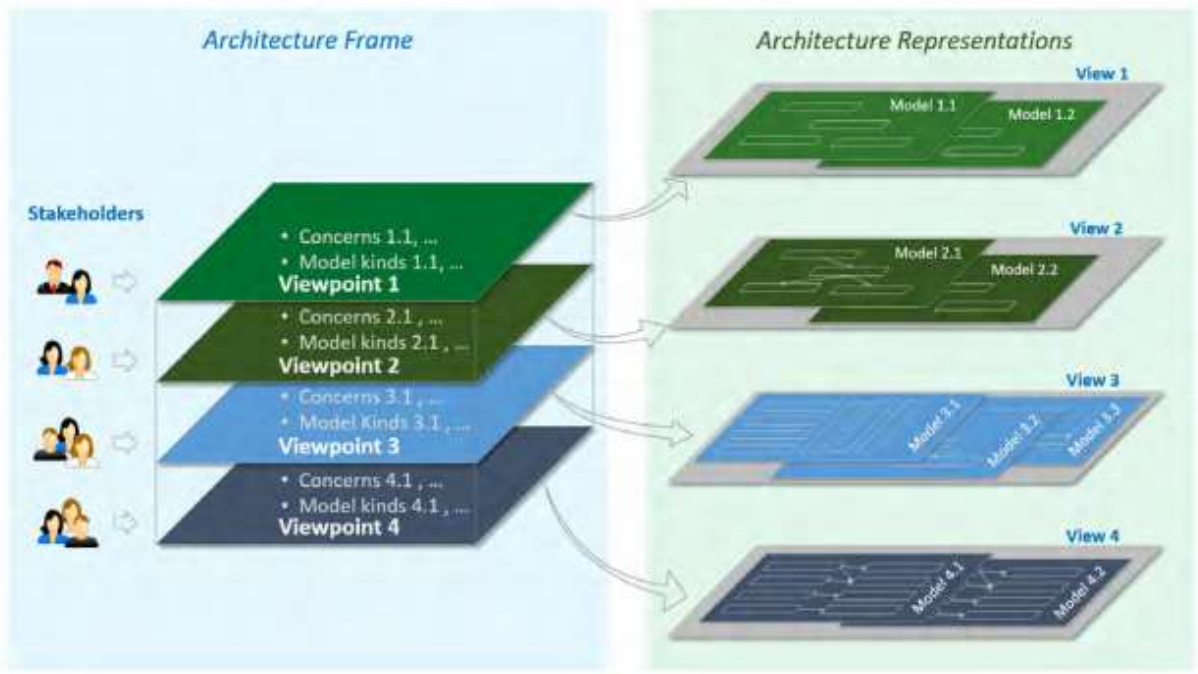

Figure 1. IIRA Architecture

The viewpoints defined in IIRA are:

- **Business viewpoint:** Addresses the specific concerns related to the business stakeholders, their vision and objectives in establishing the IIoT system. The business viewpoint of i4Q solutions is described in detail in deliverable D2.3.
- **Usage viewpoint**: Addresses the specific concerns related to the usage of the IIoT system. The usage viewpoint of i4Q solutions is described in detail in deliverable D2.4.
- **Functional viewpoint:** Addresses the specific concerns related to the functionality of the IIoT system and it is therefore closely related to the functional specification of the system. The functional viewpoint of i4Q solutions is described in detail in deliverable 2.5.
- **Implementation viewpoint:** Addresses the specific concerns related to the implementation the IIoT system. The implementation viewpoint of i4Q solutions is described in detail in deliverable D2.6.

1.1.1. Business viewpoint

In the Reference Architecture definition, one of the main objectives is to avoid the risk of a "technology-centric"' approach; for this reason, the viewpoints' definition started with the analysis of a business point of view. The business viewpoint has been defined in task T2.3 (business viewpoint) and focused on framing the vision, values, and key objectives.

The definition of the business viewpoint started with the analysis of stakeholders. They have been gathered into two main categories:

- Primary stakeholders, people and organizations who seek, receive, manage and provide IoT based services for quality improvement, having a direct impact on defining main solutions' functionalities;
- Secondary stakeholders, users of information maintained by the primary stakeholders' systems or providers of information needed by the primary stakeholders.

Concerning both categories, different types of stakeholders have been identified: **Decision-Makers** and **Technical Personnel**.

Starting from stakeholders' classification main elements have been considered: **Vision**, describing a future state of an organization; **Values**, reflecting how the vision may be perceived by the stakeholders involved in the implementation and usage of the i4Q Solutions; **Key objectives**, quantifiable high-level technical and ultimately business outcomes; **Fundamental capabilities**, referring to high-level specifications of the essential ability of the i4Q Solutions to complete specific major business tasks.

Output in terms of objectives and functional capabilities represents the input for the usage viewpoint analysis.

1.1.2. Usage viewpoint

The usage viewpoint consolidates various aspect of the system's usage, continuing the initial design efforts made in T2.3 where the business viewpoint is described. The key objectives and fundamental capabilities identified in the business viewpoint will help derive **usage activities and system requirements** of the usage viewpoint. The definition of these broad properties of the i4Q Reference Architecture allow for the identification of the usage viewpoint's three key elements of usage activities: **tasks, roles and parties**, considering both users and software systems.

The usage viewpoint guides the development of the functional and implementation viewpoints. The link between these works is defined within each task's **functional map and implementation map** which links each task with the different functions and implementation components.

1.1.3. Functional viewpoint

The functional viewpoint focuses on the functional aspects of i4Q Solutions. It considers their internal functional structure, defining its internal functional components, as well as the interfaces and interrelations with other solutions and external systems. These elements are coordinated with the usage viewpoint, linked to the user activities defined therein and with the business viewpoint, describing how the fundamental system capabilities are implemented from a functional viewpoint.

The Industrial Internet Reference Architecture (IIRA) document defines within the functional viewpoint a functional model consisted of different **functional domains** (control, operations, information, application and business), together with a set of **system characteristics** (safety, security, reliability, resilience, privacy, scalability, among others) and **cross-cutting functions** (connectivity, distributed data management, Industrial Analytics, Intelligent and resilient control).

The functional domains are (ICC, 2019):

- **Control domain:** It is a set of functions that are performed in industrial control systems. The solutions that apply these functions are implemented close to the physical systems that control them, so they may be geographically distributed and difficult to access.
- **Operations domain:** These are solutions whose function consists of provisioning, managing, monitoring, and optimizing solutions in the control domain.
- **Information domain:** These are solutions whose function is to collect data from several domains, especially the control domain, and analyze this data to acquire high-level intelligence about the overall system. To obtain the data, it uses the control domain solutions, which make it possible to control the physical systems.
- **Application domain:** A set of functions that implement logic, rules, and application models at a high and detailed level for application optimization.
- **Business domain:** This is a set of functions focused on end-to-end operations to support specific business processes.

The system characteristics are (ICC, 2018):

- **Safety:** is the condition of the system operating without causing unacceptable risk of physical injury or damage to the health of people, either directly or indirectly, as a result of damage to property or to the environment.
- **Security:** property of being protected from unintended or unauthorized access, change or destruction ensuring availability, integrity and confidentiality.
- **Reliability:** is the ability of a system or component to perform its required functions under stated conditions for a specified period of time.
- **Resilience:** property of being protected from unintended or unauthorized access, change or destruction ensuring availability, integrity and confidentiality.
- **Privacy:** is the right of an individual or group to control or influence what information related to them may be collected, processed, and stored and by whom, and to whom that information may be disclosed.
- **Scalability:** the property of a system to handle a growing amount of work by adding resources to the system.

The cross-cutting functions are:

- **Connectivity:** This is about all the components that enable Industrial Internet of Things System connectivity. These components can be sensors, controllers and other systems.

- **Distributed data management:** This deals with all the basic approaches to data exchange. It also includes the management components within the Industrial Internet of Things System.
- **Industrial analytics:** It is in charge of transforming all the data collected by the Industrial Internet of Things System. It processes the information that will then be used to make decisions and optimize systems.
- **Intelligent and resilient control:** It presents a conceptual model with different keys to build a smart and resilient control.
- **Dynamic composability and automatic integration:** These characteristics define the flexibility of the system to adapt and optimize services as environments change. In this way, interruptions can be avoided when a component is updated.

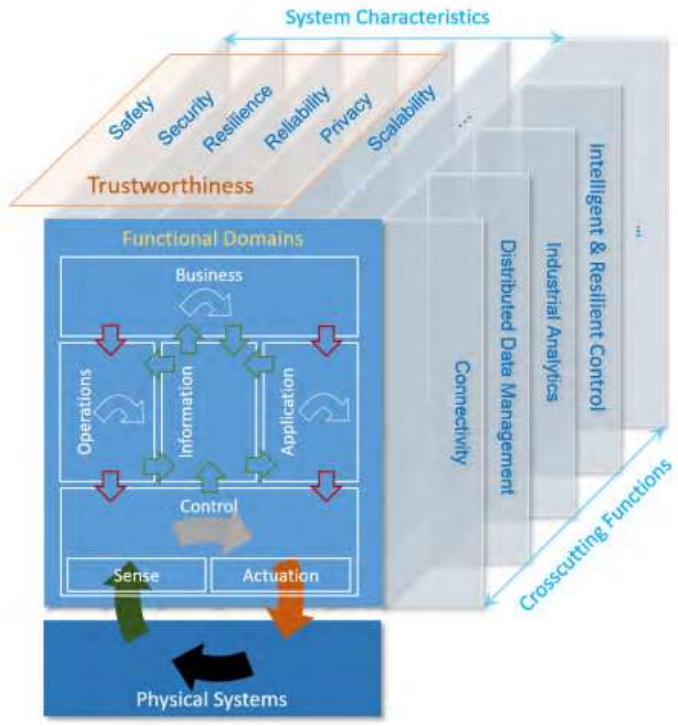

Figure 2. Functional viewpoint architecture

The data flows and control flows take place in and between these functional domains. The functional domains relate to each other through data and control flows. Control, coordination, and orchestration exercised from each of the functional domains have different granularities and run on different temporal cycles.

1.1.4. Implementation viewpoint

The implementation viewpoint, in deliverable D2.6, will describe the IIoT i4Q Architecture, its technologies, system components and interconnections between them for its implementation.

In order to achieve this, it will have as inputs, from **business** viewpoint (D2.3) its business-oriented approach that identifies stakeholders and their business vision, values and objectives to map them to system capabilities, from **usage** viewpoint (D2.4) the activities and tasks

identified to implement the capabilities and structure of the i4Q Framework for which it will provide implementation maps with their associated components, and finally from **functional** viewpoint (D2.5) the identified functional components, flows circulating among them and their typical operations.

Implementation viewpoint, based on cloud computing patterns, with technologies such as Edge computing, Microservice applications and Function as a Service (FaaS), will provide a detailed architecture considered instrumental to implement the software described in this viewpoint.

1.2. Methodology

The aim of this section is to provide a clear picture of the development of the functional viewpoint of the i4Q reference architecture. The functional viewpoint has a close relationship with the functional specification and therefore, with the activities conducted in Task 1.9 and described in deliverable 1.9. More specifically, the functional specifications use System Modeling Lenguage (OMG SYSML, 2003) to describe the functionalities of i4Q solutions, in what is called a Functional Specifications Diagram (FSD) and to link these functionalities to the requirements. Based on these definitions, the main objectives of this functional viewpoint are:

- Validate the functional models, ensuring that the relationships between i4Q Solutions (required inputs, and outputs) are consistent, using the reference architecture as a means to frame these interactions and gain common understanding on the way that the different solutions interact with each other.
- Provide further functional details that have not been considered in the development of the functional specification.
- Map the different functional models to the domains defined in IIRA.
- Define the life-cycle of i4Q solutions, by mapping their functionalities to the different phases of their life-cycle.
- Provide detailed descriptions of the data flows and interfaces between i4Q Solutions, and between i4Q Solutions and external systems.
- Provide a specification of the models and ontologies to support these data flows and ensure interoperability.
- Map the functional components to the usage model provided in the usage viewpoint as a means to link the requirements to user activities, through the SysML models.

To accomplish these objectives, within this task, partners have analysed the FSDs of the functional specification and have held bi-lateral meetings to validate that the interfaces (Inputs and Outputs) are consistent with the reference architecture, ensuring that the functionalities that are required from external solutions (either i4Q solutions or external systems) are taken into consideration.

The validation starts with the i4Q solutions in work package 5 as they are closely related to manufacturing quality control, moving then to work packages 4 and 3, which mainly provide system characteristics and cross-cutting functions to support them.

For the validation, solution providers are requested to fill in the following form for each of the functional blocks identified in the FSD:

Details	Functional Component	[Name] Name identifying the functional component.
	Description	What: [Definition] Brief definition of the functionality of the component. Who: Which user(s) identified in the usage model primarily use this function. Where: where in the reference architecture does this function belong. Why: [Objective] what is the objective of this function.
	Inputs	[Input name] + [description]. Name of the input and a description of the input parameters necessary for its operation.
	Outputs	[Output name] + [description]. Name of the output and a description of the output parameters. Provide a comprehensive definition of the outputs
	Life-cycle stage	[Lifecyle] [conceptualization, requirement, prototype, development, build, Test/validation, deployment, operation, evolution and disposal].

Table 1. Functional component input form

In deliverable 2.1 the different components of the i4Q architecture (**Figure 3**) have been defined. These components allow us to align the data inputs and outputs of the solutions with respect to the defined architecture.

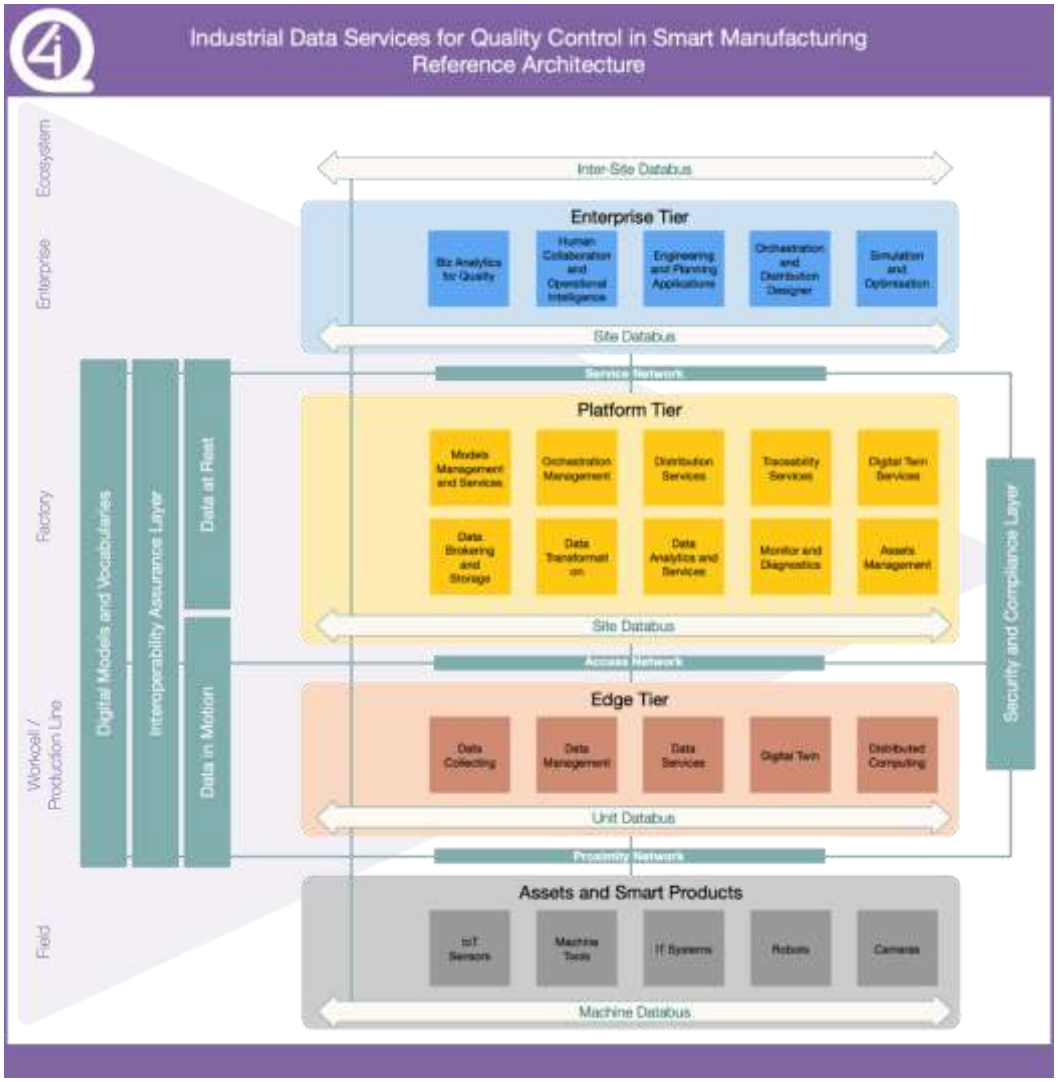

Figure 3. i4Q reference architecture

The components are the following (divided by Tiers):

Enterprise Tier:

- **Biz analytics for quality**: Applications built on top of underneath services (e.g.: Models Management and Services, Data Analytics and Services...) aiming to pursue excellence in quality focusing on zero-waste context.
- **Human Collaboration and Operational Intelligence**: CRM, CSM, DSS and ERP/MES and other services that provide smart alerting and quality diagnosis.
- **Engineering and Planning Applications:** CAD, CAE, CAM and AR/VR and everything related to the joint and integrated use of software systems for computer-aided design and computer-aided manufacturing.
- **Orchestration and Distribution Designer:** Engineering Environment for defining pipelines and distributions such as a scalable, easy to use, policy-based distribution mechanism to ease the task of distributing AI/ML models and other metadata to the edge.
- **Simulation and Optimisation:** Simulation and Optimisation applications for optimising manufacturing processes.

Platform Tier:

- **Models Management and Services:** Management of Models and related Services enabling holistic vision of data across i4Q Infrastructure.
- **Orchestration Management:** Services for orchestration management enabling high resiliency and responsiveness workloads.
- **Distribution Services:** Distribution and deployment services enabling plug & play reconfiguration and installation of workloads.
- **Traceability Services:** Robust and rapid traceability service (i.e., DLT) to provide an audit trail for all inserted data, guaranteeing immutability and finality.
- **Digital Twin Services:** Digital Twin management services that enable industrial companies virtual validation/visualisation and productivity optimisation using pre-existing and/or simulated data and data from different factory levels.
- **Data Brokering and Storage:** Services for data brokering and storage management supporting a high degree of digitisation in companies with most manufacturing devices acting as sensors or actuators and generating vast amounts of data.
- **Data Transformation:** Services for data transformation (typically post-processing).
- **Data Analytics and Services**: Services for data analytics on top of the data infrastructure with several incremental algorithms (i.e., operating on data streams with fast incremental updates) suitable for analytic processing of high-speed data streams.
- **Monitor and Diagnostics:** Services for near/real-time monitoring and diagnostics monitoring the health of workloads, predict problems and take corrective actions, predict failures and provide alerts.
- **Assets Management:** Services for managing factory assets, enabling firmware updates, status traceability, or logging functions.

Edge Tier:

- **Data Collecting:** Solutions for data ingestion, collecting raw data from the facilities and store them to the data lake or make them available for the processing.
- **Data Management:** Solutions for data management, transformation (typically pre-processing), harmonization and loading.
- **Data Services**: Data Services to enable ingestion use and maintenance.
- **Digital Twin:** Digital Twin (Operational Services) to achieve a connected 3D production simulation with a digital twin for manufacturing.
- **Distributed Computing:** Fog Computing/Edge nodes that allow deploying and running AI workloads on the edge enabling efficient analysis.

Furthermore, the life-cycle stages considered are:

- **Conceptualization:** the functional component is used in the early stages of the system definition. It is used to frame a situation or problem to face and envision the system-level goals of a solution.
- **Requirement**: the functional component is used to define and refine the requirements of the system.

- **Prototype:** the functional component is used to design and propose a first prototype of the solution to develop and evaluate such prototype as well.
- **Development:** the functional component is used to create a partial solution to face a specific task, solve a specific problem, provide a specific functionality, etc.
- **Build:** the functional component is used to build an ensemble of a number of partial solutions.
- **Test/validation:** the functional component is used to test and validate partial solutions, ensembles of solutions, the whole system or any other concern.
- **Deployment**: the functional component is used to deploy (i.e., make available) the system.
- **Operation**: the functional component is used in use-time, to operate the system.
- **Evolution:** the functional component is used to design, plan, and prepare the evolution of the system. This includes a vast set of concerns like realizing about new needs of the system, current needs not actually covered by the system, ways of using the system in unexpected ways, errors and problems identified, possible solutions, etc.
- **Disposal:** the functional component is used in any task related to the shutting down of the system. This includes the releasing of the resources occupied, the destruction of whatever assets that should not survive to the life span of the project (e.g. sensitive data) and the provision of agreed guarantees beyond such life span.

Based on these definitions, solution providers have completed the description of the functional components with the required information. The results of these activities are described in section 2. It is important to note that these activities are held in parallel to and in coordination with the definition of the functional specification. While the focus of the functional specification is to define the inner functional structure of each solution, the focus of this task is to ensure that the definitions are consistent with each other (no duplicated functionalities, consistent interconnections, no missing functionalities) and correctly mapped to the reference architecture.

This additional information is used as the basis to define the digital models and ontologies used by i4Q solutions to support the identified dataflows. These allow specifying a vocabulary related to a certain domain (classes, properties, functions or relations between components). Digital models and ontologies play a key role in resolving semantic interoperability between information systems and their use in context.

For the architecture proposed by i4Q in deliverable 2.1 it has been divided into three domains. The **Data Flow**, the **Control Flow** and the **Workload Distribution Flow**. These are described in Section 3 and the objective is to avoid communication problems due to lack of shared understanding which limits interoperability and therefore the possibility to reuse and share information. In the pursuit of interoperability, the standards described in deliverable 2.2 will be taken into account.

Once this process is completed, the information provided is used to make the functional map by mapping the different tasks defined in deliverable 2.4 (**usage viewpoint**), linked to the solutions that will be involved and the functional components that will act on them. In addition, a number of additional criteria will be added to validate the fulfilment of the different tasks with respect to the functional components. Finally, in the section 5, a series of sequence diagrams

will be developed for different i4Q Solutions, which will allow us to better interpret the functionality of these solutions within each of the **life-cycles stages**.

2. Functional Components

2.1. i4Q^{QE} Quali Explore for Data Quality Factor Knowledge

2.1.1. Mapping of Functional Structure Diagram to Reference Architecture

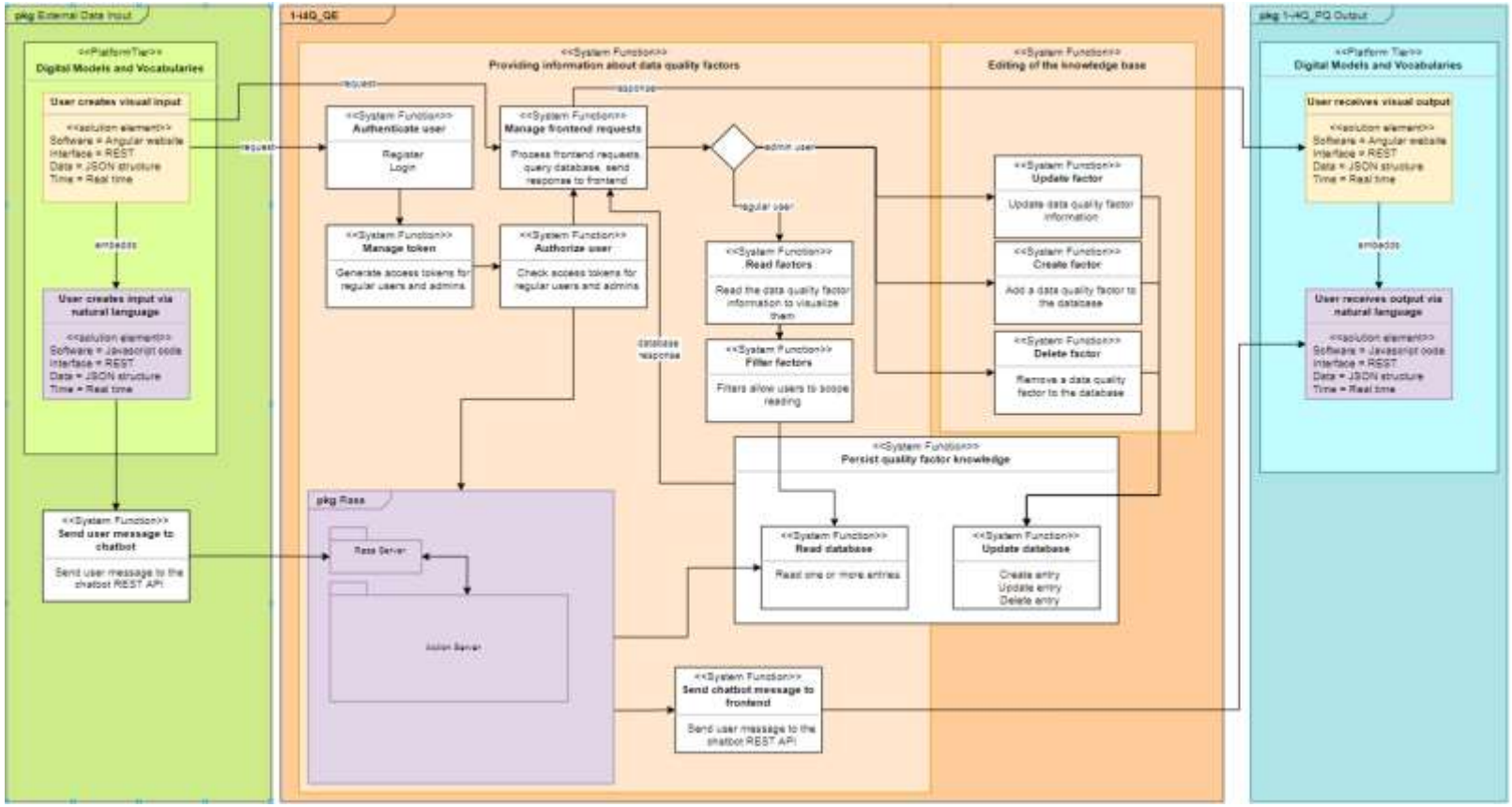

Figure 4. i4Q^{QE} Functional Components Diagram

2.1.2. Functional Components

Details	Functional Component	Graphical User Interface
	Description	What: Provides users a graphical user interface and allows viewing and manipulation of the Quali Explore database entries. Who: Any user with valid security token. Where: Digital Models and Vocabularies. Why: To visualize data and allow users to interact.
	Inputs	Text and clicks.
	Outputs	Text messages.
	Life-cycle stage	Operation.

Table 2. i4Q^{QE} Graphical User Interface

Details	Functional Component	Chatbot widget
	Description	What: Provides users a natural language interface. Who: Any user with valid security token. Where: Digital Models and Vocabularies. Why: To visualize data and allow users to interact via natural language.
	Inputs	User's natural language text.
	Outputs	Text messages.
	Life-cycle stage	Operation.

Table 3. i4Q^{QE} Chatbot widget

Details	Functional Component	User authentication.
	Description	What: Authenticates users. Who: Any person visiting the QualiExplore website. Where: Security and Compliance layer OR internal. Why: The control access to the solution.
	Inputs	Credentials (username and password).
	Outputs	Security token.
	Life-cycle stage	Operation.

Table 4. i4Q^{QE} User Authentication

Details	Functional Component	Manage token.
	Description	What: Checks if tokens are valid. Who: Any authenticated user. Where: Security and Compliance layer OR internal. Why: To constrain token validity (e.g. expiry).
	Inputs	Security token.
	Outputs	Renewed security token or redirect if expired.
	Life-cycle stage	Operation.

Table 5. i4Q[QE] Manage token

Details	Functional Component	Authorize user.
	Description	What: Control access to information. Who: Any user with valid security token. Where: Security and Compliance layer OR internal. Why: To allow user roles specific views on factors and actions (e.g. editing, deleting). Important for the editing environment.
	Inputs	Security token.
	Outputs	Access right.
	Life-cycle stage	Operation.

Table 6. i4Q[QE] Authorize user

Details	Functional Component	Persist quality factor knowledge.
	Description	What: Stores knowledge about the data quality factors in a graph database. Who: Internal to the solution. Where: Digital Models and Ontologies. Why: To have a persistent storage for knowledge.
	Inputs	Instructions from the website's middleware or queries from the Rasa action server.[1]
	Outputs	Returns information about factors as text.
	Life-cycle stage	Operation.

Table 7. i4Q^{QE} Persist Quality factor knowledge

2.1.3. Mapping of functional components to IIRA functional domain

i4Q^{QE} Quali Explore for Data Quality Factor Knowledge	
Graphical User Interface	Application
Chatbot widget	Application
User Authentication	Business
Manage Token	Operations
Authorize user	Operations
Persist Quality Factor Knowledge	Operations

Table 8. i4Q^{QE} Mapping of functional components to IIRA functional domains

[1] The action server runs custom Rasa actions (e.g., Python code) to create responses. https://rasa.com/docs/action-server

2.2. i4Q[BC] Blockchain Traceability of Data

2.2.1. Mapping of Functional Structure Diagram to Reference Architecture

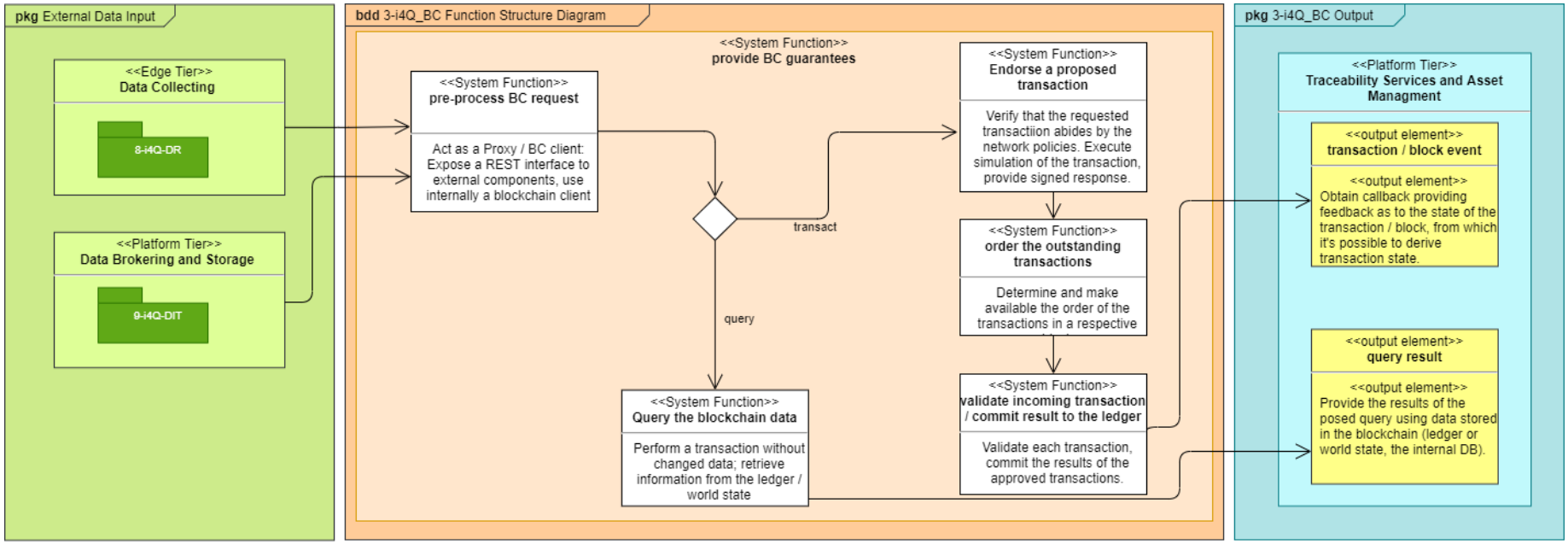

Figure 5. i4Q[BC] Functional Component Diagram

2.2.2. Functional Components

Details	Functional Component	Pre-process BC request.
	Description	What: Act as a Proxy / BC client: Expose a REST interface to external components, use internally a blockchain client. Who: developer / IoT device. Where: Platform tier. Why: Enter data to be stored on the blockchain or query for data already on the blockchain.
	Inputs	Information to carry out a blockchain transaction or query.
	Outputs	Transaction / query to be involved on the blockchain.
	Life-cycle stage	Operation

Table 9. i4Q^{BC} Pre-process BC Request

Details	Functional Component	Query the blockchain data.
	Description	What: Perform a transaction without changing data; retrieve information from the ledger / world state. Who: Business user. Where: Platform tier. Why: Query for data stored in the blockchain.
	Inputs	Query to be invoked on the blockchain.
	Outputs	Query results.
	Life-cycle stage	Operation

Table 10. i4Q^{BC} Query the blockchain data

Details	Functional Component	Endorse a proposed transaction.
	Description	What: Verify that the requested transaction abides by the network policies. Execute simulation of the transaction, provide signed response. Who: Internal to the solution. Where: platform tier. Why: A part of the internal solution process in charge of approving a transaction to be invoked on the blockchain. Performs simulated execution of the transaction.

	Inputs	Transaction proposal.
	Outputs	Signed transaction response, including approval and corresponding read and write sets.
	Life-cycle stage	Operation.

Table 11. i4Q^{BC} Endorse a proposed transaction

Details	Functional Component	Order the outstanding transaction.
	Description	What: Determine and make available the order of the transactions in a respective block. Who: Internal to the solution. Where: platform tier. Why: This stage ensures that all participants see transactions in the same order.
	Inputs	Transaction proposal.
	Outputs	Block of transactions.
	Life-cycle stage	Operation

Table 12. i4Q^{BC} Order the outstanding transaction

Details	Functional Component	Validate incoming transaction / commit result to the ledger.
	Description	What: Validate each transaction, commit the results of the approved transactions. Who: Internal to the solution. Where: Platform tier. Why: Determine which transactions are valid, and apply the valid transaction in the designated order.
	Inputs	Block of transactions.
	Outputs	Event indicating that a new block has been processed.
	Life-cycle stage	Operation.

Table 13. i4Q^{BC} Validate/Commit

2.2.3. Mapping of functional components to IIRA functional domain

i4Q^{BC} Blockchain Traceability of Data	
Pre-process BC request	Operations
Query	Information

Endorse	Application
Order	Operations
Validate / commit	Control

Table 14. i4Q[BC] Mapping of functional components to IIRA functional domain

2.3. i4Q[TN] Trusted Networks with Wireless & Wired Industrial Interfaces

2.3.1. Mapping of Functional Structure Diagram to Reference Architecture

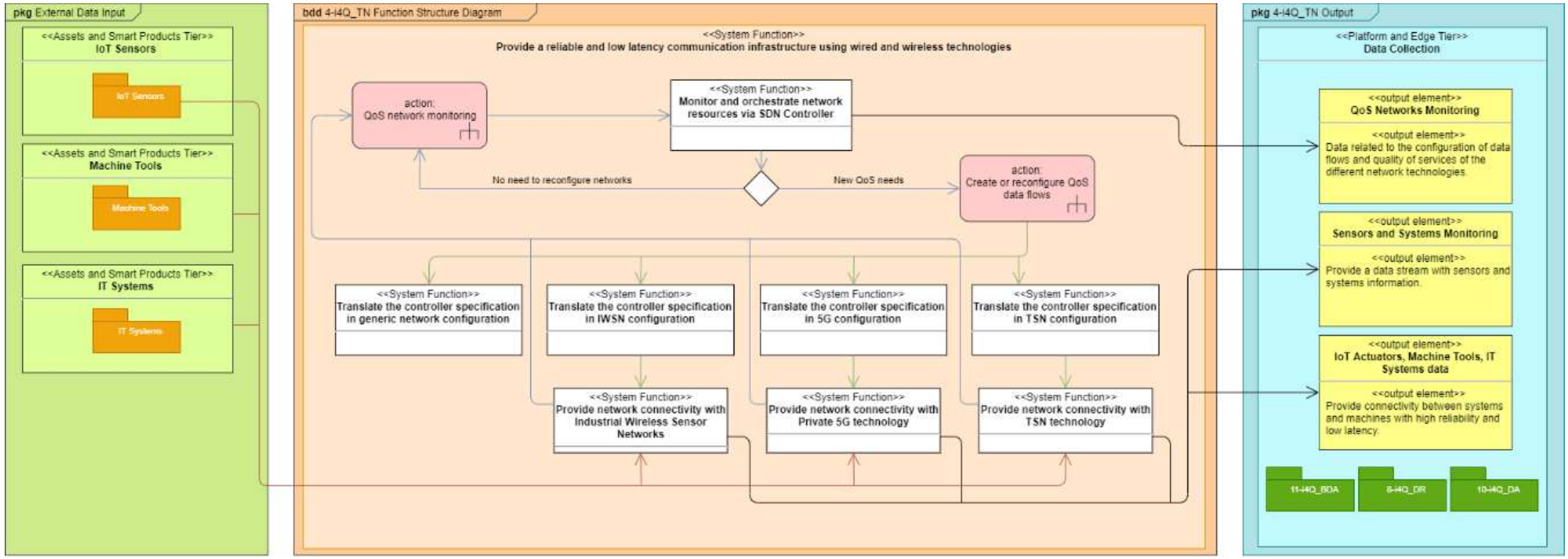

Figure 6. i4Q[TN] Functional Component Diagram

2.3.2. Functional Components

Details	Functional Component	Monitor and orchestrate network resources via SDN controller.
	Description	What: This element allows to manage the network resources of different communication technologies such as IWSN, TSN or private 5G networks. Who: Any user. Where: Edge Tier. Why: In order to satisfy an expected network QoS, a centralized controller orchestrate all the network resources to improve the network performance.
	Inputs	Different QoS network monitoring.
	Outputs	SDN Network management commands.
	Life-cycle stage	Deployment, Operation.

Table 15. i4Q[TN] Monitor and orchestrate network resrouces via SDN controller

Details	Functional Component	Translate the controller specification in generic network configuration.
	Description	What: Translates the commands received from the SDN controller into specific commands to manage network resources. Who: Any user. Where: Edge Tier. Why: An SDN agent is needed to integrate the SDN controller with a generic network.
	Inputs	SDN Network management commands.
	Outputs	Specific network management commands.
	Life-cycle stage	Deployment, Operation.

Table 16. i4Q[TN] Translate the controller specification in generic network configuration

Details	Functional Component	Translate the controller specification in IWSN configuration.
	Description	What: Translates the commands received from the SDN controller into specific IWSN commands to manage network resources. Who: Any user. Where: Edge Tier.

		Why: An SDN agent is needed to integrate the SDN controller with an IWSN.
	Inputs	SDN Network management commands.
	Outputs	Specific IWSN management commands.
	Life-cycle stage	Deployment, Operation.

Table 17. i4Q^{TN} Translate the controller specification in IWSN configuration

Details	Functional Component	Translate the controller specification in 5G configuration
	Description	What: Translates the commands received from the SDN controller into specific 5G commands to manage network resources. Who: Any user. Where: Edge Tier. Why: An SDN agent is needed to integrate the SDN controller with a 5G private network.
	Inputs	SDN Network management commands.
	Outputs	Specific 5G network management commands.
	Life-cycle stage	Deployment, Operation.

Table 18. i4Q^{TN} Translate the controller specification in 5G configuration

Details	Functional Component	Translate the controller specification in TSN configuration.
	Description	What: Translates the commands received from the SDN controller into specific TSN commands to manage network resources. Who: Any user. Where: Egde Tier. Why: A SDN agent is needed to integrate the SDN controller with a TSN network.
	Inputs	SDN Network management commands.
	Outputs	Specific TSN management commands.
	Life-cycle stage	Deployment, Operation.

Table 19. i4Q^{TN} Translate the controller specification in TSN configuration

Details	Functional Component	Provide network connectivity with Industrial Wireless Sensor Networks.
	Description	What: Reliable and secure Industrial Wireless Sensor Network infrastructure orchestrated by the SDN controller. Who: Any user. Where: Egde Tier. Why: Enables data collection from IoT sensors, industrial equipment and IT system monitoring and actuation.
	Inputs	Specific IWSN management commands; IoT Sensors and systems data collecting.
	Outputs	QoS feedback; IoT sensors and system data monitoring and actuator.
	Life-cycle stage	Deployment, Operation.

Table 20. i4Q^{TN} Provide network connectivity with Industrial Wireless Sensor Networks

Details	Functional Component	Provide network connectivity with Private 5G technology.
	Description	What: Ultra reliable and low latency private 5G infrastructure orchestrated by the SDN controller. Who: Any user. Where: Edge Tier. Why: Enables data collection from IoT sensors, industrial equipment and IT system monitoring and actuation.
	Inputs	Specific 5G network management commands; IoT Sensors and systems data collecting.
	Outputs	QoS feedback; IoT sensors and system data monitoring and actuator.
	Life-cycle stage	Deployment, Operation.

Table 21. i4Q^{TN} Provide network connectivity with Private 5G technology

Details	Functional Component	Provide network connectivity with TSN technology.
	Description	What: Reliable and low latency TSN infrastructure orchestrated by the SDN controller. Who: Any user.

		Where: Edge Tier. Why: Enables data collection from IoT sensors, industrial equipment and IT system monitoring and actuation.
	Inputs	Specific TSN management commands; IoT Sensors and systems data collecting.
	Outputs	QoS feedback; IoT sensors and system data monitoring and actuator.
	Life-cycle stage	Deployment, Operation.

Table 22. i4Q^{TN} Provide network connectivity with TSN technology

2.3.1. Mapping of functional components to IIRA functional domain

i4Q^{TN} Trusted Networks with Wireless & Wired Industrial Interfaces	
Monitor and orchestrate network resources via SDN controller	Control
Translate the controller specification in generic network configuration	Operations
Translate the controller specification in IWSN configuration	Operations
Translate the controller specification in 5G configuration	Operations
Translate the controller specification in TSN configuration	Operations
Provide network connectivity with Industrial Wireless Sensor Networks	Control
Provide network connectivity with Private 5G technology	Control
Provide network connectivity with TSN technology	Control

Table 23. i4Q^{TN} Mapping of functional components to IIRA functional domain

2.4. i4Q[SH] IIoT Security Handler

2.4.1. Mapping of Functional Structure Diagram to Reference Architecture

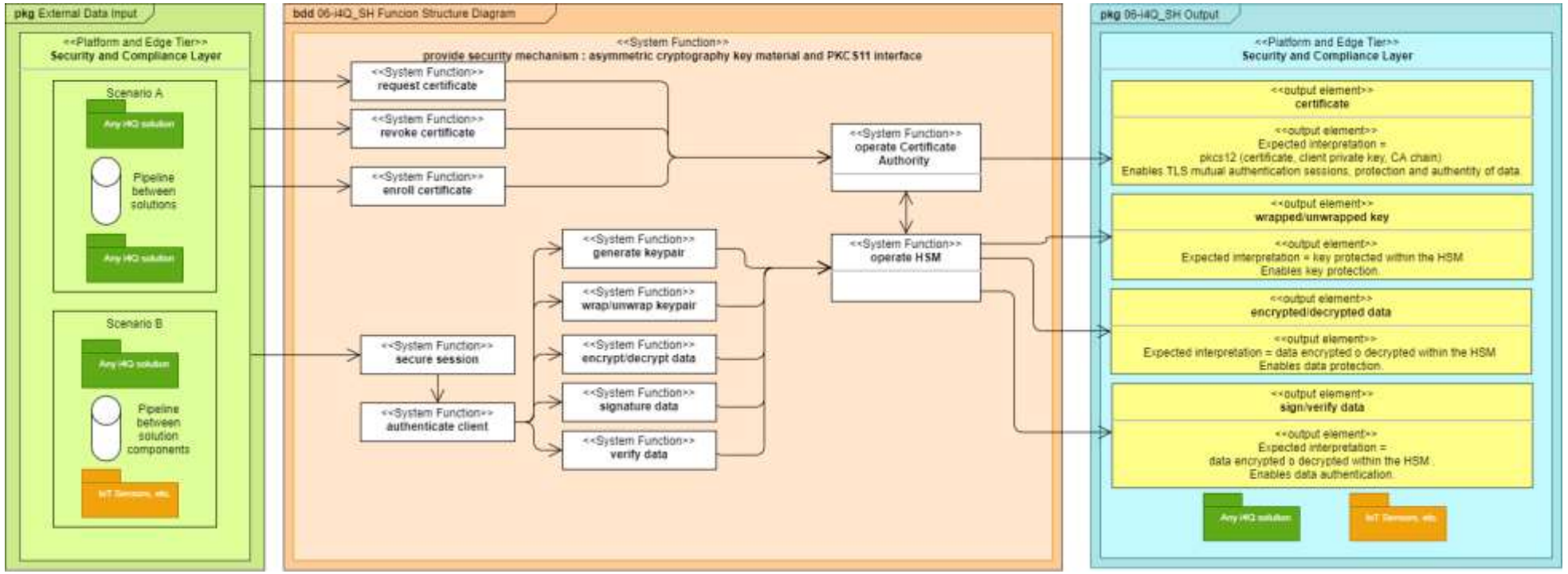

Figure 7. i4Q[SH] Functional Component Diagram

2.4.2. Functional Components

Details	Functional Component	Request certificate.
	Description	What: Issue a certificate and private key. Who: Other i4Q solutions or equipment from pilots such as a potential IoT gateway. Where: Security and Compliance Layer. Why: Provide a digital identity as a trust anchor point for a secure communication or cryptography operations such as signatures.
	Inputs	X509 certificate attributes: Common Name. Issuance attributes: Username and Token password.
	Outputs	X509 certificate in PEM format or PKCS12 format.
	Life-cycle stage	Operation.

Table 24. i4Q^{SH} Request Certificate

Details	Functional Component	Revoke certificate.
	Description	What: Revoke a certificate for being able to request certificate state through CRL or OCSP. Who: A system operator. Where: Security and Compliance Layer. Why: Sometimes certificates and private keys are compromised and should be revoked in order not to use them.
	Inputs	Certificate Serial Number.
	Outputs	CRL List and update of list in OCSP Server.
	Life-cycle stage	Operation.

Table 25. i4Q^{SH} Revoke Certificate

Details	Functional Component	Enroll certificate.
	Description	What: When only is necessary to issue a certificate with an existing key. Who: Other i4Q solutions or equipment from pilots such as a potential IoT gateway. Where: Security and Compliance Layer. Why: Provide a digital identity as a trust anchor point for a secure communication or cryptography

		operations such as signatures.
	Inputs	CSR: Certificate Signature Request.
	Outputs	X509 certificate in PEM format or PKCS12 format.
	Life-cycle stage	Operation.

Table 26. i4Q^{SH} Enroll Certificate

Details	Functional Component	Operate Certificate Authority.
	Description	What: Provide certificate life-cycle management with management of a CA (Certificate Authority) and maintenance operations. Who: System or PKI (public key infrastructure) operator and SH internal operations. Where: Security and Compliance Layer. Why: X509 certificates needs a Certificate Authority as a clearing method to agree between different entities which is the trust anchor point.
	Inputs	Web requests.
	Outputs	Web responses through a web console.
	Life-cycle stage	Operation.

Table 27. i4Q^{SH} Operate Certificate Authority

Details	Functional Component	Secure session.
	Description	What: it is necessary to secure the session with TLS, when accessing the HSM (hardware security module) operations. The same as accessing an API through a HTTPs session. Who: Other i4Q solutions or equipment from pilots such as a potential IoT gateway. Where: Security and Compliance Layer. Why: When accessing HSM resources, the transactions should be protected.
	Inputs	HTTPs API requests.
	Outputs	HTTPs API responses.
	Life-cycle stage	Operation.

Table 28. i4Q^{SH} Secure Session

Details	Functional Component	Authenticate client.
	Description	What: It is necessary to authenticate the session with an API authentication mechanism, when accessing the HSM (hardware security module) operations. Who: Other i4Q solutions or equipment from pilots such as a potential IoT gateway. Where: Security and Compliance Layer. Why: When accessing HSM resources, the transactions should be authenticated.
	Inputs	HTTPs API requests.
	Outputs	HTTPs API responses.
	Life-cycle stage	Operation.

Table 29. i4Q^{SH} Authenticate client

Details	Functional Component	Generate keypair.
	Description	What: Create a security token in the HSM. Who: Other i4Q solutions or equipment from pilots such as a potential IoT gateway. Where: Security and Compliance Layer. Why: It is necessary a security token to perform cryptography operations in the HSM. The security token will reside within the HSM.
	Inputs	Generate keypair requests.
	Outputs	Response with success/fail.
	Life-cycle stage	Operation.

Table 30. i4Q^{SH} Generate keypair

Details	Functional Component	Wrap/unwrap keypair.
	Description	What: When the security token is outside the HSM, it can be protected with a security token from the HSM. Who: Other i4Q solutions or equipment from pilots such as a potential IoT gateway. Where: Security and Compliance Layer. Why: It is necessary to protect keys when they are stored in a persistent memory media and unprotect

		them when they are used.
	Inputs	Keypair.
	Outputs	Keypair wrapped or unwrapped.
	Life-cycle stage	Operation.

Table 31. i4Q^{SH} Wrap/Unwrap keypair

Details	Functional Component	Encrypt/decrypt data
	Description	What: Encrypt or decrypt data with a security token that resides within the HSM. Who: Other i4Q solutions or equipment from pilots such as a potential IoT gateway. Where: Security and Compliance Layer. Why: Provides data privacy on information exchanged between different entities.
	Inputs	Data decrypted or encrypted.
	Outputs	Data encrypted or decrypted.
	Life-cycle stage	Operation

Table 32. i4Q^{SH} Encrypt/Decrypt data

Details	Functional Component	Signature data.
	Description	What: Sign data with a security token that resides within the HSM. Who: Other i4Q solutions or equipment from pilots such as a potential IoT gateway. Where: Security and Compliance Layer. Why: Signature provides an authentication mechanism over no-secured data.
	Inputs	Data to be encrypted.
	Outputs	Signature of the data. Data consists on one string value inside HTTPs API Request.
	Life-cycle stage	Operation.

Table 33. i4Q^{SH} Signature data

Details	Functional Component	Verify data.

	Description	What: Verify data with a security token that resides within the HSM. Who: Other i4Q solutions or equipment from pilots such as a potential IoT gateway. Where: Security and Compliance Layer. Why: Enables the verification of signed data.
	Inputs	Signature. Data consists on one string value inside HTTPs API Request.
	Outputs	Verification success or fail.
	Life-cycle stage	Operation.

Table 34. i4Q^{SH} Verify data

Details	Functional Component	Operate HSM.
	Description	What: Provide cryptography operations with a HSM and maintenance operations. Who: System or HSM operator and SH internal operations. Where: Security and Compliance Layer. Why: Trusted cryptography operations need security tokens backed by security hardware enabling the highest protection that can be achieved with the current state of the art.
	Inputs	Web requests.
	Outputs	Web responses.
	Life-cycle stage	Operation.

Table 35. i4Q^{SH} Operate HSM

2.4.3. Mapping of functional components to IIRA functional domain

i4^{QSH} IIoT Security Handler	
Request certificate	Control
Revoke certificate	Control
Enroll certificate	Operations
Operate Certificate Authority	Operations
Source session	Application
Authenticate client	Control

Generate keypair	Application
Wrap/unwrap keypair	Application
Encrypt/decrypt data	Application
Signature data	Application
Verify data	Application
Operate HSM	Operations

Table 36. i4Q^{SH} Mapping of functional components to IIRA functional domain

2.5. i4Q[DR] Data Repository

2.5.1. Mapping of Functional Structure Diagram to Reference Architecture

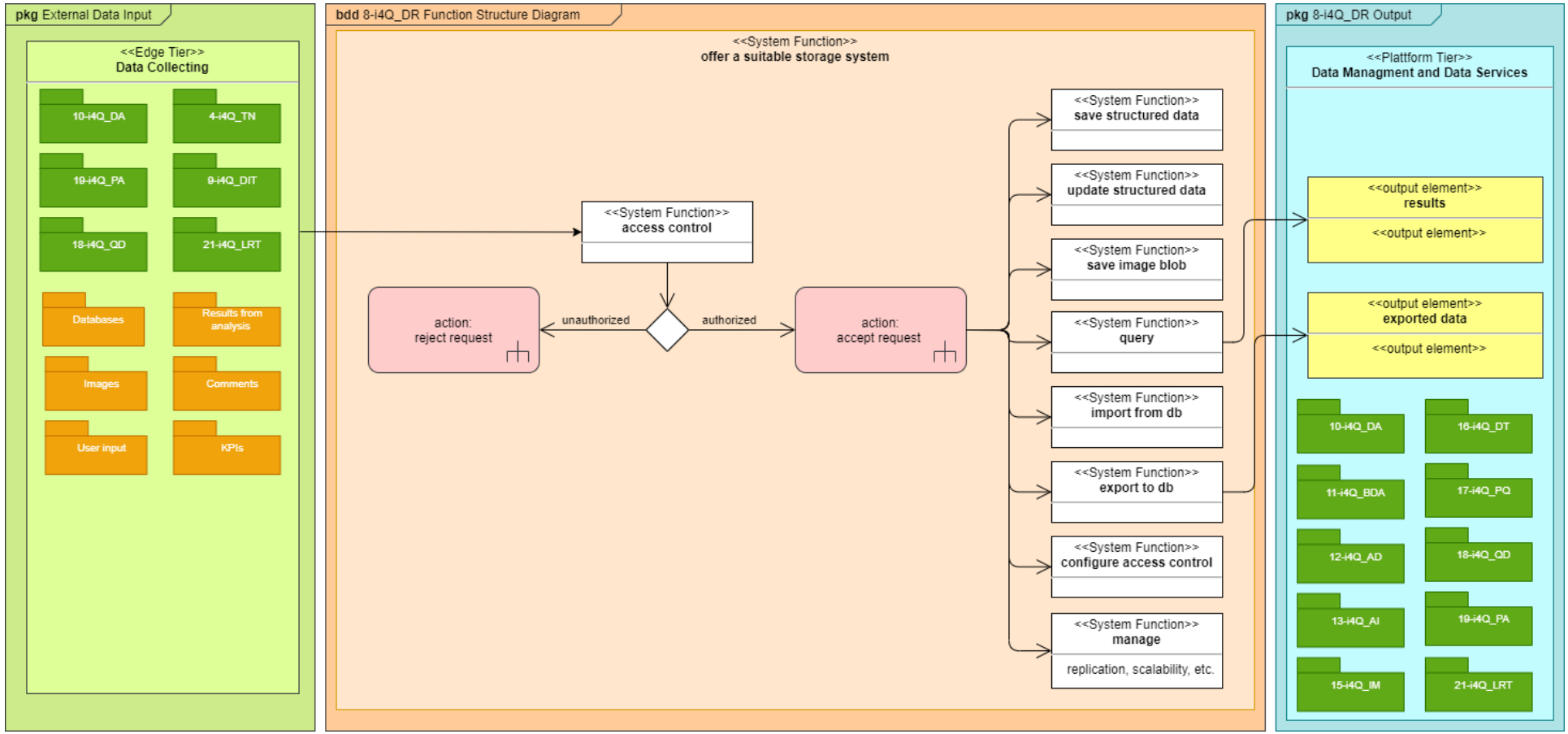

Figure 8. i4Q[DR] Functional Component Diagram

2.5.2. Functional Components

Details	Functional Component	Access control.
	Description	What: Decides if a user request can be performed or not depending on authorization concerns. Who: Any user. Where: Brokering and Storage (Platform Tier). Why: This is used to ensure that only authorized users perform authorized actions. This feature is typically offered by an external IAM (Identity and Access Management) tool. This includes a number of subfunctions which are listed with an informative-only purpose. In practice, as pointed out by the previous comment, these features are offered by an external tool, which is who decides which subfunctions it is composed of, how they work, etc.
	Inputs	User request to perform any action on the Data Repository.
	Outputs	Authorized (resp. unauthorized) flag in case the user is authorized (resp. not authorized) to perform the requested action.
	Life-cycle stage	Operation.

Table 37. i4Q^{DR} Access control

Details	Functional Component	Save structured data.
	Description	What: Save structured data. Who: Any authorized user. Where: Brokering and Storage (Platform Tier). Why: This is used to save structured data. This feature is typically offered by an external data store tool (RDBMS, NoSQL, etc.).
	Inputs	User request including the data to save.
	Outputs	None.
	Life-cycle stage	Operation.

Table 38. i4Q^{DR} Save structured data

Details	Functional Component	Update structured data.
	Description	What: Update structured data. Who: Any authorized user. Where: Brokering and Storage (Platform Tier). Why: This is used to update the structured data handled by the Data Repository. This feature is typically offered by an external data store tool (RDBMS, NoSQL, etc.).
	Inputs	User request including the required data (references to the data to update and the data changes to save).
	Outputs	None.
	Life-cycle stage	Operation.

Table 39. i4Q^{DR} Update structured data

Details	Functional Component	Manage blobs.
	Description	What: Performs actions to manage blobs: save a blob, update the metadata of a blob, delete a blob. Who: Any authorized user. Where: Brokering and Storage (Platform Tier). Why: This is used to manipulate the blobs by the Data Repository. The reason to distinguish the management of blobs from regular structured data is because blobs usually have special needs (e.g. amount of persistent storage, amount of bandwidth to perform a full read, etc.), some different semantics (e.g. an update typically means the deletion of the current blob and the upload of a new blob, besides the update of the current metadata). These features are typically offered by an external data store tool (RDBMS, NoSQL, etc.).
	Inputs	User request including the required data (the blob to save, the metadata to update or references to the blob to update or delete).
	Outputs	None.
	Life-cycle stage	Operation.

Table 40. i4Q^{DR} Manage blobs

Details	Functional Component	Query.
	Description	What: Performs actions to query the data handled by the Data Repository (structured data, blobs or any other type). Who: Any authorized user. Where: Brokering and Storage (Platform Tier). Why: This is used to query the data handled by the Data Repository. This feature is typically offered by an external data store tool (RDBMS, NoSQL, etc.).
	Inputs	User request including a query. The query may include criteria to filter the required data.
	Outputs	Data queried.
	Life-cycle stage	Operation.

Table 41. i4Q^{DR} Query

Details	Functional Component	Import from DB.
	Description	What: Imports data from an external database. Who: User with a specific permission to import and export data. Where: Brokering and Storage (Platform Tier). Why: This is used to import data from external sources.
	Inputs	Batch-like data or reference to the database to import.
	Outputs	None.
	Life-cycle stage	Operation.

Table 42. i4Q^{DR} Import from DB

Details	Functional Component	Export to DB.
	Description	What: Exports data to an external database. Who: User with a specific permission to import and export data. Where: Brokering and Storage (Platform Tier). Why: This is used to export data to an external source.
	Inputs	Reference of the data to export.

	Outputs	Batch-like data.
	Life-cycle stage	Operation.

Table 43. i4Q^{DR} Export to DB

Details	Functional Component	Config access control.
	Description	What: Performs actions to configure the access control mechanism. Who: User with a specific permission to configure the access control. Where: Brokering and Storage (Platform Tier). Why: This is used to prepare the access control to be used. This includes: 1) Registering the roles to authorize, typically in deployment time. 2) Registering the actions to authorize, typically in deployment time. 3) Registering the authorization rules (which roles allow which actions), typically in deployment time. 4) Registering the authorization bindings (which users are given which roles), typically in operation time.
	Inputs	Misc, including 1) List of roles to authorize. 2) List of actions to authorize. 3) Authorization rules. 4) Authorization bindings.
	Outputs	None.
	Life-cycle stage	Development, operation.

Table 44. i4Q^{DR} Config access control

Details	Functional Component	Manage.
	Description	What: Performs actions to manage the data repository. Who: User with a specific permission to manage the data repository. Where: Brokering and Storage (Platform Tier). Why: This is used to configure all internal concerns of the data repository (e.g. replication-related issues).

		This includes 1) replication-related actions (e.g. create replicas of the data store, reconfigure replicas, remove replicas, etc.).
	Inputs	None.
	Outputs	None.
	Life-cycle stage	Deployment, operation.

Table 45. i4Q^{DR} Manage data repository

2.5.3. Mapping of functional components to IIRA functional domain

i4Q^{DR} Data Repository	
Access control	Control
Save structured data	Application
Update structured data	Application
Manage blobs	Operations
Query	Information
Import from DB	Information
Export DB	Application
Config access control	Business
Manage	Business

Table 46. i4Q^{DR} Mapping of functional components to IIRA functional domain

2.6. i4Q^{DIT} Data Integration and Transformation Services

2.6.1. Mapping of Functional Structure Diagram to Reference Architecture

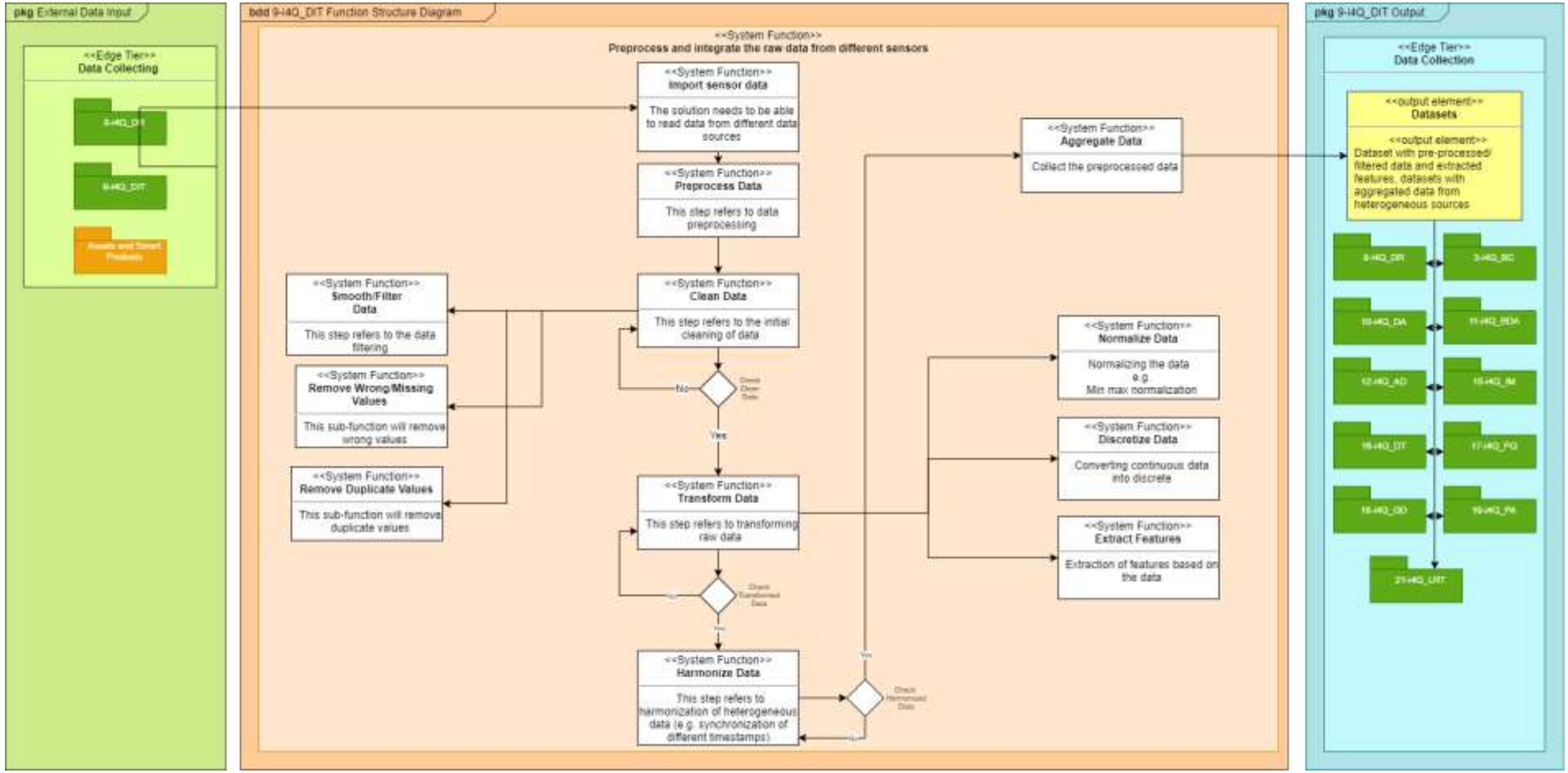

Figure 9. i4Q^{DIT} Functional Component Diagram

2.6.2. Functional Components

Details	Functional Component	Import Sensor Data.
	Description	What: This functional component is responsible for correctly importing sensor data from repositories or directly from sources. Who: Algorithm provider (internal to the solution). Where: Edge Tier (Data collecting). Why: This component is necessary to read all the incoming data for further processing by this or other solutions.
	Inputs	Sensor/machine data (whether at rest, or real time). Sensor data are streaming data, thus they will be in the form of time series variables. Other important variables that should be imported are timestamps and identifications (e.g. for a specific sensor, for a row etc).
	Outputs	Sensor/machine data.
	Life-cycle stage	Operation.

Table 47. i4Q[DIT] Import Sensor Data

Details	Functional Component	Pre-process Data.
	Description	What: This component is responsible for the pre-processing of raw data, which may include the cleaning of data and their transformation. Who: Algorithm provider (internal to the solution). Where: Platform Tier (Data transformation). Why: The input data from sensors may have wrong entries or repeated entries and since there are time series data, they usually need filtering to remove noise.
	Inputs	Raw sensor data: time series variables that were imported in the previous function.
	Outputs	Clean sensor data, filtered sensor data, in the form of a dataset that will not contain 'error' entries. E.g. the initial imported dataset includes 1000 rows responding to readings from the sensor, however there are 100 duplicate entries, thus after the cleaning, the resulted output would be a dataset

		with 900 rows.
	Life-cycle stage	Operation.

Table 48. i4Q[DIT] Pre-process Data

Details	Functional Component	Transform Data.
	Description	What: This component includes sub-functions like normalization and feature extraction. Who: Algorithm provider (internal to the solution). Where: Platform Tier (Data transformation). Why: Feature extraction is needed in the processing/analysis of time series data. In particular for the processes of the project, there are specific features that need to be computed, according to the requirements of the pilots.
	Inputs	Clean/filtered data: time series along with row identifiers and timestamps.
	Outputs	Features (new variables) calculated from the inputs: new variables will be computed from the initial sensor readings (e.g. by applying a function and a rolling time window). An example of extracted feature could be the mean value of every 10 readings of a sensor measurement.
	Life-cycle stage	Operation.

Table 49. i4Q[DIT] Transform Data

Details	Functional Component	Harmonize Data.
	Description	What: Bring the data of different sources and structures to a unified format. Who: Algorithm provider (internal to the solution). Where: Platform Tier (Data transformation). Why: Sensor data may arrive from different sources and with different structures. To further combine and analyse them, first they need to have the same structure (or schema).
	Inputs	Clean data (datasets were duplicate values and wrong entries have been removed) or feature sets (sets of variables extracted from the initial sensor readings) from different sources, with different format (e.g. the readings of sensor A are

		accompanied by a machine id but the readings of Sensor B are not).
	Outputs	Harmonized dataset(s): datasets that need to be used together are brought to the same format (or schema).
	Life-cycle stage	Operation.

Table 50. i4Q[DIT] Harmonize Data

Details	Functional Component	Aggregate Data.
	Description	What: This component performs an early fusion (combination) of data derived from previous functions. Who: Algorithm provider (internal to the solution). Where: Edge Tier (Data services). Why: The aggregation of sensor data may improve the result of the requested analysis, e.g. prediction of failures or assessing the condition of the manufacturing process.
	Inputs	Different datasets of harmonized and pre-processed sensor data.
	Outputs	A new dataset with combined data.
	Life-cycle stage	Operation.

Table 51. i4Q[DIT] Aggregate Data

2.6.3. Mapping of functional components to IIRA functional domain

i4Q[DIT] Data Integration and Transformation Services	
Import Sensor Data	Control
Pre-process Data	Operations
Transform Data	Application
Harmonize Data	Application
Aggregate Data	Application

Table 52. i4Q[DA] Mapping of functional components to IIRA functional domain

2.7. i4Q[DA] Services for Data Analytics

2.7.1. Mappin of Functional Structure Diagram to Reference Architecture

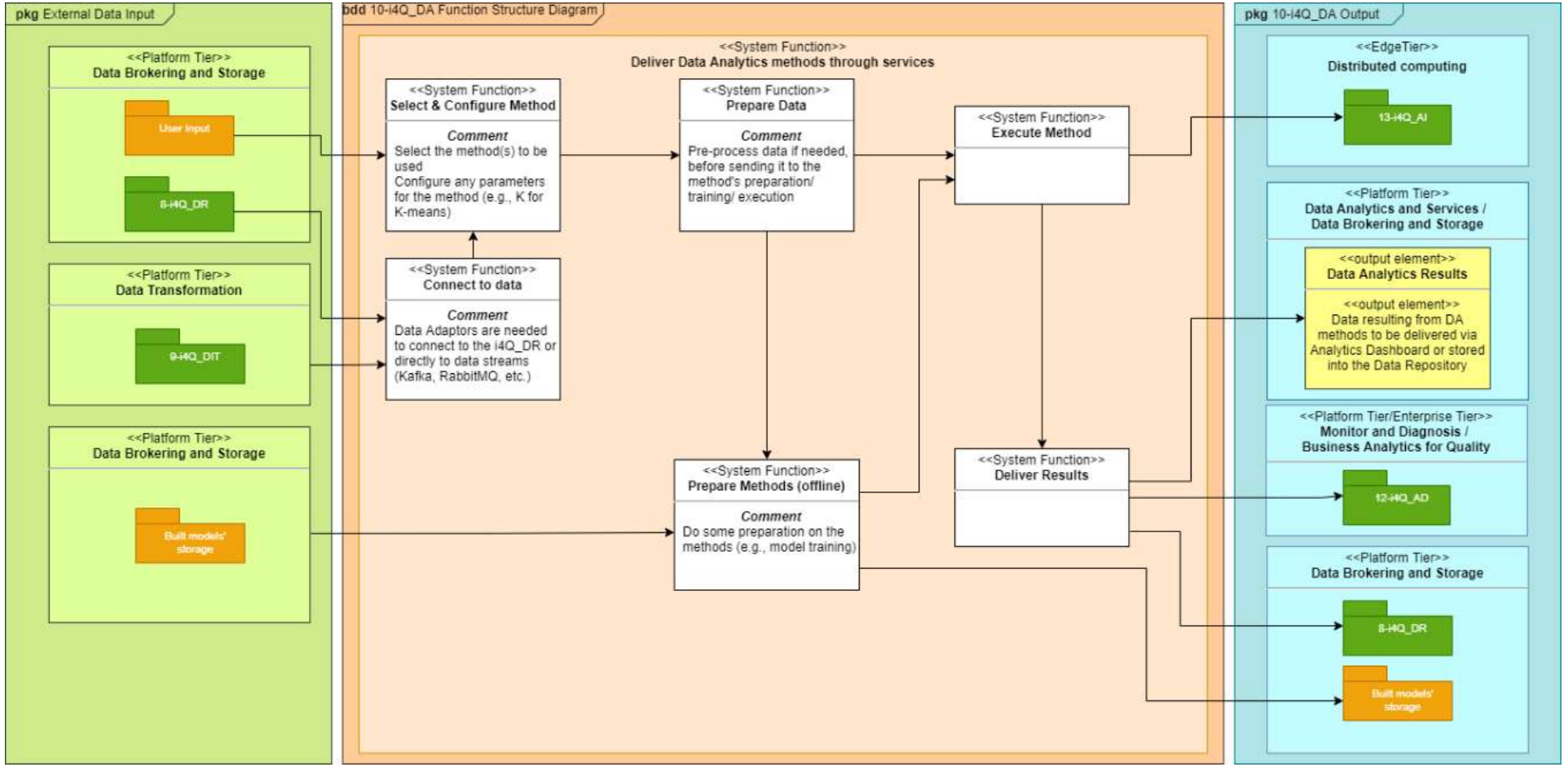

Figure 10. i4Q[DA] Functional Component Diagram

2.7.2. Functional Components

Details	Functional Component	Connect to data.
	Description	What: This component is responsible for connecting to the i4QDR or directly to data streams to retrieve data. Who: Any authorized user. Where: Edge Tier (Data Brokering and Storage). Why: To gather the necessary data to feed the analytics methods.
	Inputs	Data From i4QDR or streams.
	Outputs	Data that the user wants to apply analytics to, in an already clean and structured format.
	Life-cycle stage	Deployment.

Table 53. i4Q[DA] Connect to data

Details	Functional Component	Select and Configure method.
	Description	What: This function selects the analytic methods from a list. Who: Any authorized user. Where: Platform Tier (Models Management and Services). Why: The analytics methods require some customization by the user (variables to predict, etc.).
	Inputs	The user selects the desired methods from a list and the data to apply the methods on.
	Outputs	Analytic model.
	Life-cycle stage	Deployment.

Table 54. i4Q[DA] Select and Configure Method

Details	Functional Component	Prepare Data.
	Description	What: This function will prepare (split data, normalize, etc.) the selected data in order to apply the selected algorithms. Who: Internal to the solution. Where: Platform Tier (Data Transformation). Why: Some algorithms need additional data

		preparation.
	Inputs	Data that the user wants to apply analytics, in an already clean and structured format.
	Outputs	Data in the format required by the analytics method selected by the user.
	Life-cycle stage	Deployment.

Table 55. i4Q^{DA} Prepare Data

Details	Functional Component	Prepare Methods.
	Description	What: This component will prepare (train models, load previously trained models, etc.) the methods for execution. Who: Internal to the solution. Where: Platform Tier (Data Analytics and Services). Why: Some algorithms need training.
	Inputs	Prepared Data, analytic models.
	Outputs	Customized Model ready to deploy.
	Life-cycle stage	Build.

Table 56. i4Q^{DA} Prepare Methods

Details	Functional Component	Execute Methods.
	Description	What: This component will run the selected methods. Who: Internal to the solution. Where: Platform (Data Analytics and Services) / Edge tier (Distributed Computing). Why: The purpose of this function is to execute the selected method to obtain the desired result.
	Inputs	Customized Model.
	Outputs	Methods results.
	Life-cycle stage	Operation.

Table 57. i4Q^{DA} Execute Methods

Details	Functional Component	Deliver Results.

	Description	What: This component is responsible for transforming the raw output of the models and prepare it to visualization and storage. Who: Internal to the solution. Where: Platform Tier (Distribution Services). Why: The outputs of the analytics methods require some transformation to a human readable format.
	Inputs	Methods raw results.
	Outputs	Analytics' results.
	Life-cycle stage	Operation.

Table 58. i4Q^{DA} Deliver Results

2.7.3. Mapping of functional components to IIRA functional domain

i4Q^{DA} Services for Data Analytics	
Connect to data	Information
Select and Configure method	Operations
Prepare Data	Application
Prepare Methods	Application
Execute Methods	Application
Deliver Results	Business

Table 59. i4Q^{DA} Mapping of functional components to IIRA functional domain

2.8. i4Q[BDA] Big Data Analytics Suite

2.8.1. Mapping of Functional Structure Diagram to Reference Architecture

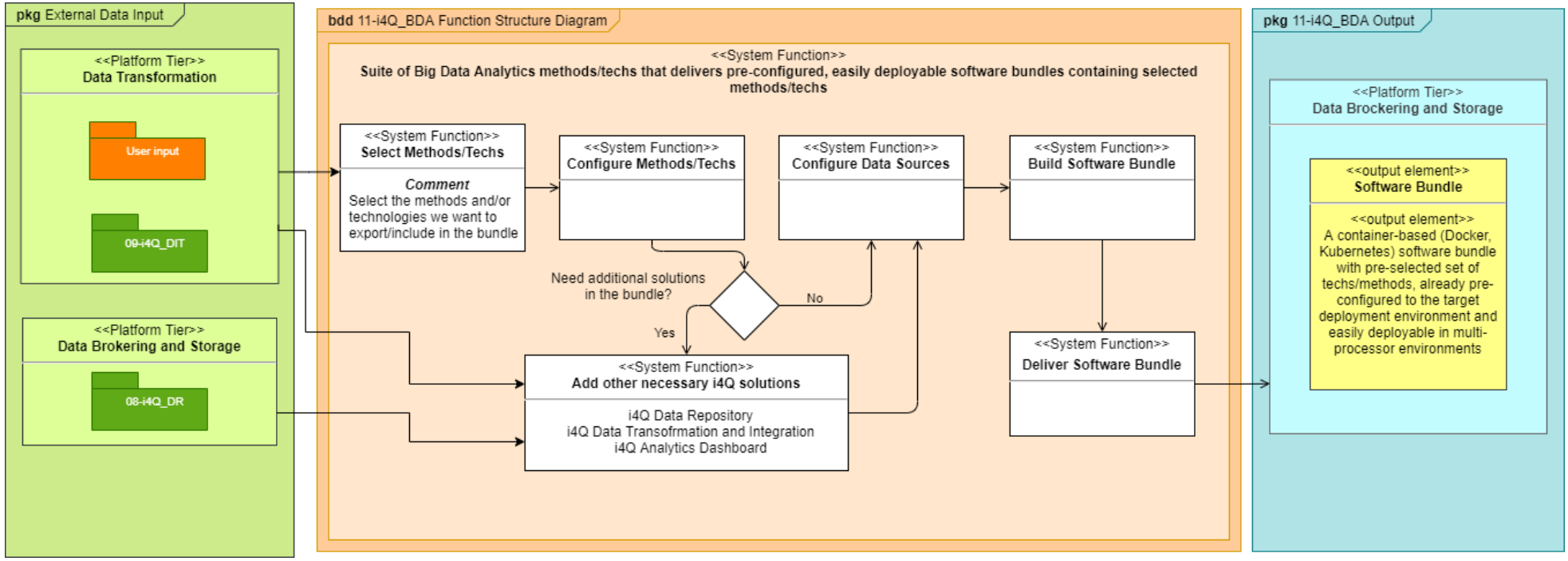

Figure 11. i4Q[BDA] Functional Component Diagram

2.8.2. Functional Components

Details	Functional Component	Select methods and techs.
	Description	What: This function selects the analytic methods from a list as well as the technologies required. Who: Any authorized user. Where: Platform Tier (Models Management and Services). Why: So that the solution is embedded with the desired tools.
	Inputs	User selects from a list of available methods and technologies the tools desired in the software bundle.
	Outputs	Desired methods and techs.
	Life-cycle stage	Deployment.

Table 60. i4Q[BDA] Select methods and techs

Details	Functional Component	Configure methods and techs.
	Description	What: This component is responsible for configure the client infrastructure (# processors, available Ram, etc.) and the selected methods. Who: Any authorized user. Where: Platform Tier. Why: To adapt the solution to the client hardware availability.
	Inputs	User configures the deployment infrastructure where the desired methods and techs will be deployed.
	Outputs	Mapping of solutions and infrastructure.
	Life-cycle stage	Deployment.

Table 61. i4Q[BDA] Configure Methods and techs

Details	Functional Component	Add other i4Q solutions.
	Description	What: This component is responsible for connecting/integrating additional i4Q solutions that might be necessary. Who: Any authorized user. Where: Platform Tier (Models Management and Services).

		Why: To allow the usage of these solutions on the client premises.
	Inputs	User selects, from a list of available i4Q solutions, the solutions he/she wishes to be integrated in the software bundle to be deployed locally.
	Outputs	Configuration file of the additional solutions.
	Life-cycle stage	Deployment.

Table 62. i4Q[BDA] Add other i4Q Solutions

Details	Functional Component	Configure Data Sources.
	Description	What: This component is responsible for the connection to the client data repositories. Who: Any authorized user. Where: Platform Tier (Data Brokering and Storage). Why: To gather the necessary data to feed the analytics methods.
	Inputs	User configures the connection to the desired data sources. This data sources may be local databases, files (csv, excel, etc.), databases in the cloud or the 8-i4QDR.
	Outputs	Connection to data sources.
	Life-cycle stage	Deployment.

Table 63. i4Q[BDA] Configure Data Sources

Details	Functional Component	Build Software Bundle.
	Description	What: This component is responsible for bundling all the necessary tools with the correct configurations in a single execution file. Who: Internal to the solution. Where: Platform Tier (Distribution Services). Why: To abstract the user from the complexity of the technologies.
	Inputs	Map of tecs/infrastructure, data source connection.
	Outputs	Single execution file (Docker compose file, Kubernetes deployment file "Yaml").
	Life-cycle stage	Deployment.

Table 64. i4Q[BDA] Build Software Bundle

Details	Functional Component	Deliver Software Bundle.
	Description	What: This component is responsible for delivering the solutions to the clients. Who: Internal to the solution. Where: Platform Tier (Distribution Services). Why: Thanks to this component, it will be able to provide a distributable file with the necessary elements to set up the service.
	Inputs	Single execution file (Docker compose file, kubernetes deployment file "Yaml") with the base configuration.
	Outputs	Single execution file (Docker compose file, kubernetes deployment file "Yaml") with configured software.
	Life-cycle stage	Deployment.

Table 65. i4Q[BDA] Deliver Software Bundle

2.8.3. Mapping of functional components to IIRA functional domain

i4Q[BDA] Big Data Analytics Suite	
Select methods and techs	Operations
Configure methods and techs	Operations
Add other i4Q solutions	Business
Configure Data Sources	Control
Build Software Bundle	Application
Deliver Software Bundle	Application

Table 66. i4Q[BDA] Mapping of functional components to IIRA functional domain

2.9. i4Q^{AD} Analytics Dashboard

2.9.1. Mapping of Functional Structure Diagram to Reference Architecture

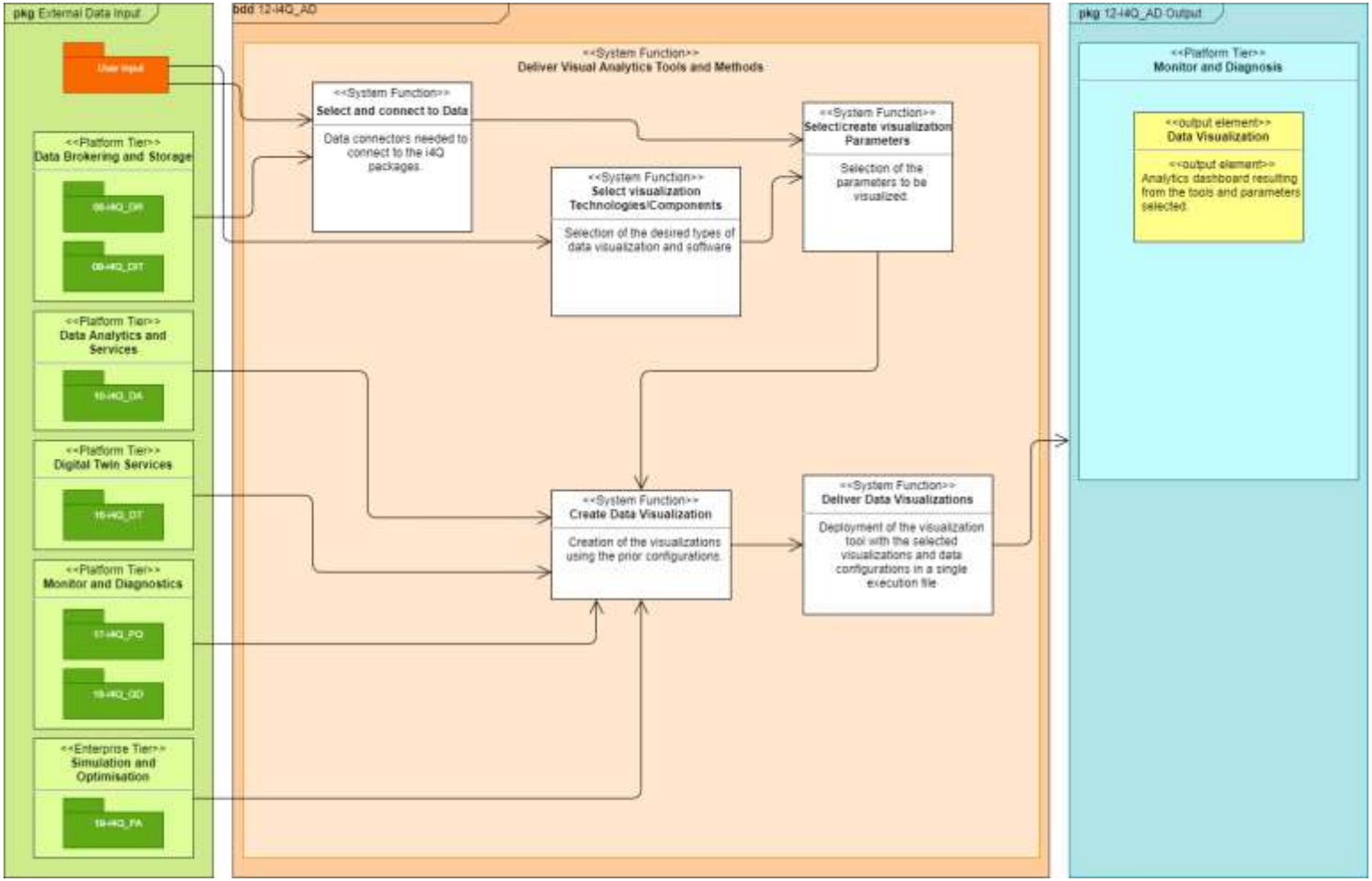

Figure 12. i4Q^{AD} Functional Component Diagram

2.9.2. Functional Components

Details	Functional Component	Select and connect to Data.
	Description	What: This component is responsible for connecting to the i4QDR or directly to data streams to retrieve data. The authorized user configures the connection to the desired data sources. Who: Any authorized user. Where: Platform Tier (Data Brokering and Storage). Why: To gather the necessary data to feed the analytic visualizations.
	Inputs	This data sources may be local databases, files (csv, excel, etc.), databases in the cloud or the 8-i4QDR.
	Outputs	Data that the user wants to visualize, in an already clean and structured format.
	Life-cycle stage	Build.

Table 67. i4Q[AD] Select and connect to data

Details	Functional Component	Select visualization Technologies/Components.
	Description	What: This component is responsible for the selection of the visualization components from a list as well as the technologies required. Who: Any authorized user. Where: Platform Tier (Models Management and Services). Why: So that the solution is embedded with the desired components.
	Inputs	Select a technology desired for the visualization.
	Outputs	Mapping of visualizations and technologies.
	Life-cycle stage	Build.

Table 68. i4Q[AD] Select visualization Tecnologies/Components

Details	Functional Component	Select/create visualization Parameters.
	Description	What: This component is responsible for data pre-processing and visualization parameters configuration. Who: Any authorized user.

		Where: Platform Tier (Data Analytics and Services). Why: So that the solution is customized according to the user needs.
	Inputs	Data that the user wants to visualize, in an already clean and structured format. Mapping of visualizations and technologies.
	Outputs	Mapping of required visualizations with the data required by the selected visualizations.
	Life-cycle stage	Build.

Table 69. i4Q^{AD} select/create visualization Parameters

Details	Functional Component	Create Data Visualization.
	Description	What: This component is responsible for creating the selected visualizations with the previously defined configuration. Who: Internal to the solution. Where: Platform Tier (Data Analytics and Services). Why: This will allow us to provide a clear and intuitive interface for the end user. In this way, the different data can be interpreted in a simple way.
	Inputs	Mapping of required visualizations with the data required by the selected visualizations.
	Outputs	Data visualizations (Bar graphs, data tables, etc.).
	Life-cycle stage	Build.

Table 70. i4Q^{AD} Create Data Visualization

Details	Functional Component	Deliver Data Visualizations.
	Description	What: This component is responsible for bundling all the required visualizations in a dashboard and the creation of a single execution file with the tools already configured and ready for deployment. Who: Internal to the solution. Where: Platform Tier (Distribution Services). Why: To abstract the user from the complexity of the technologies.
	Inputs	Data visualizations, Mapping of technologies.
	Outputs	Single execution file (Docker compose file).

	Life-cycle stage	Deployment.

Table 71. i4Q[AD] Deliver Data Visualizations

2.9.3. Mapping of functional components to IIRA functional domain

i4Q[AD] Analytics Dashboard	
Select and connect to Data	Information
Select visualization Technologies/Components	Operations
Select/Create visualization Parameters	Operations
Create Data Visualization	Application
Deliver Data Visualization	Application

Table 72. i4Q[AD] Mapping of functional components to IIRA functional domain

2.10.i4Q[AI] AI Models Distribution to the Edge

2.10.1. Mapping of Functional Structure Diagram to Reference Architecture

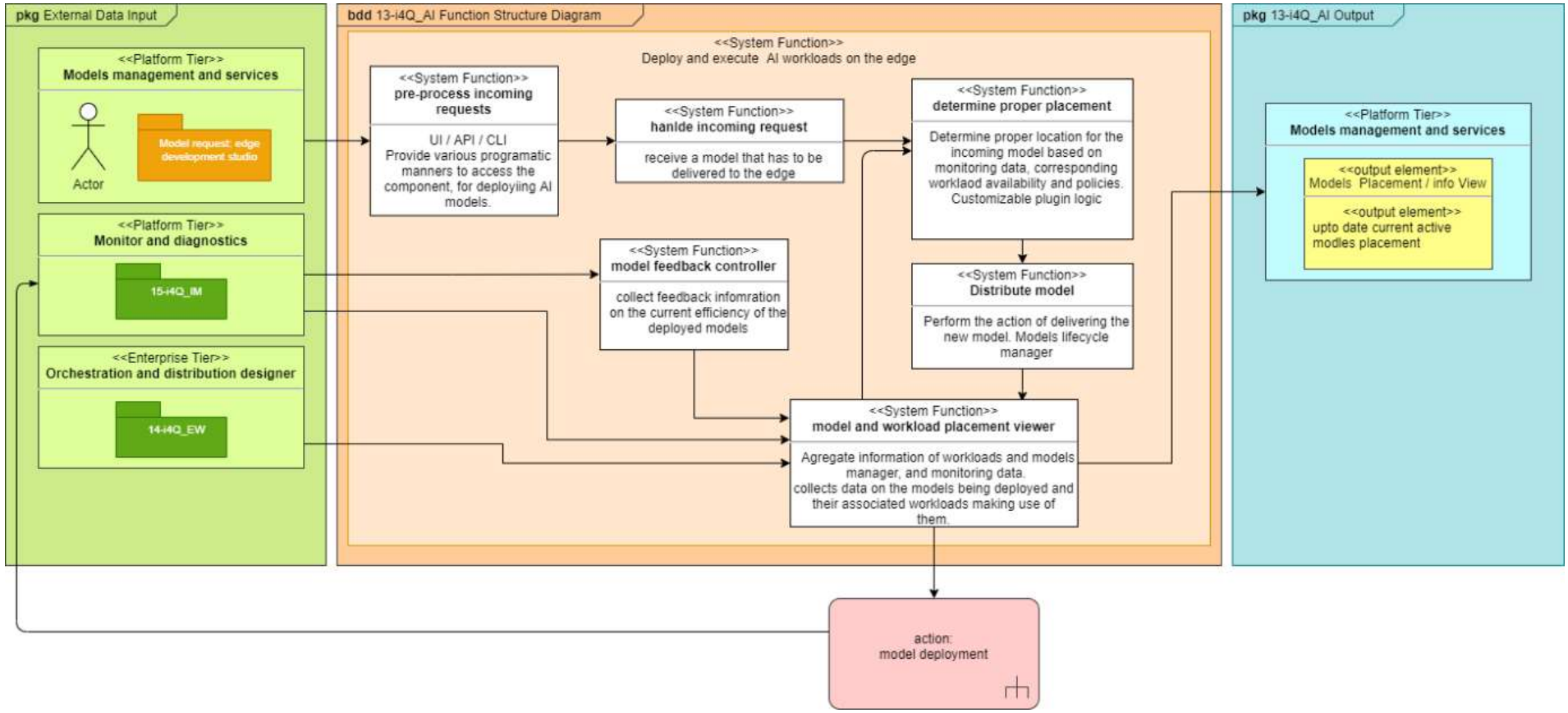

Figure 13. i4Q[AI] Functional Component Diagram

2.10.2. Functional Components

Details	Functional Component	Pre-process incoming requests.
	Description	What: Provide various programmatic manners to access the component, for deploying models (UI / API / CLI). Who: Developer / data scientist. Where: Platform tier (Models management and services). Why: Introduce a new or refined model to the system.
	Inputs	Model to be deployed or redeployed.
	Outputs	Model deployment request.
	Life-cycle stage	Development.

Table 73. i4Q^{AI} Pre-process incoming request

Details	Functional Component	Handle incoming request.
	Description	What: Receive a model that has to be delivered to the edge. Who: Developer / data scientist. Where: Platform tier (Models management and services). Why: Prepare the internal model deployment request.
	Inputs	Model to be deployed or redeployed.
	Outputs	Model deployment request.
	Life-cycle stage	Development.

Table 74. i4Q^{AI} Handle incoming Request

Details	Functional Component	Determine proper placement.
	Description	What: Determine proper location for the incoming model based on monitoring data, corresponding workload availability and policies. Customizable plugin logic. Who: Different systems within the Edge platform. Where: Platform tier (Models management and services). Why: Determine proper placement of the model.

	Inputs	Model to be deployed or redeployed.
	Outputs	Model deployment placement.
	Life-cycle stage	Development.

Table 75. i4Q^{AI} Determine proper placement

Details	Functional Component	Distribute model.
	Description	What: Perform the action of delivering the new model. Models' life-cycle manager. Who: Different systems within the Edge platform. Where: Platform tier (Distribution services). Why: Deploy the model in the designated location.
	Inputs	Model to be deployed or redeployed.
	Outputs	Model deployment.
	Life-cycle stage	Development.

Table 76. i4Q^{AI} Distribute model

Details	Functional Component	Model and workload placement viewer.
	Description	What: Aggregatee information of workloads and models manager, and monitoring data. Who: Different systems within the Edge platform. Where: Platform tier (Models management and services). Why: monitor the deployment and targets state.
	Inputs	Monitoring data.
	Outputs	State of deployments view.
	Life-cycle stage	Operation.

Table 77. i4Q^{AI} Model and workload placement viewer

Details	Functional Component	Model feedback controller.
	Description	What: Aggregate information of workloads and models manager, and monitoring data. Who: Different systems within the Edge platform. Where: Edge tier (Distributed computing). Why: Provide feedback on the deployed models at the edge.

	Inputs	Monitoring / feedback data from running workloads.
	Outputs	Input to estimate efficiency of running models.
	Life-cycle stage	Operation.

Table 78. i4Q^{AI} Model feedback controller

2.10.3. Mapping of functional components to IIRA functional domain

i4Q^{AI} AI Models Distribution to the Edge	
Pre-process incoming requests	Business
Handle incoming request	Control
Determine proper placement	Control
Distribute model	Control
Model and workload placement viewer	Operations
Model feedback controller	Business

Table 79. i4Q^{AI} Mapping of functional components to IIRA functional domain

2.11.i4Q[EW] Edge Workloads Placement and Deployment

2.11.1. Mapping of Functional Structure Diagram to Reference Architecture

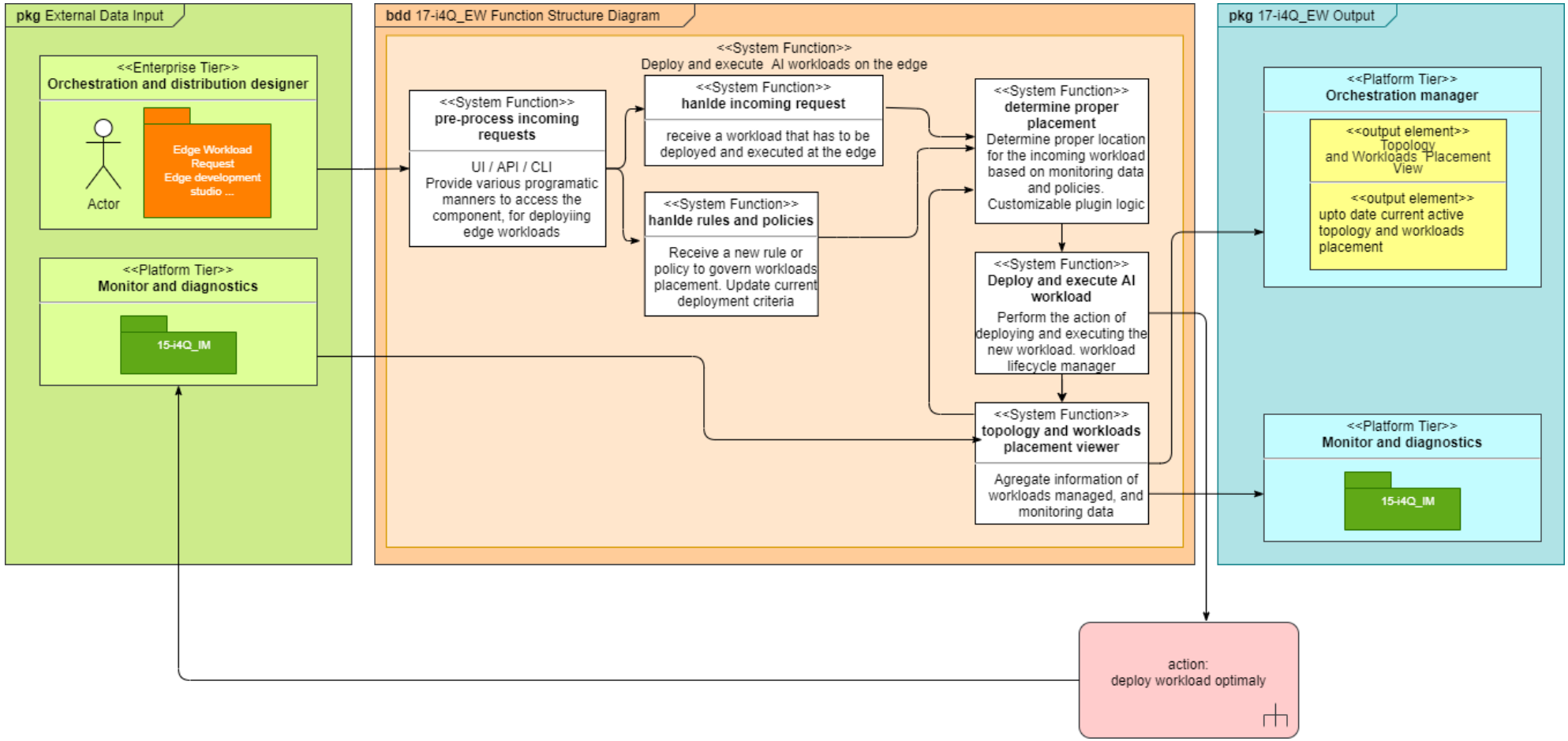

Figure 14. i4Q[EW] Functional Component Diagram

2.11.2. Functional Components

Details	Functional Component	Pre-process incoming requests.
	Description	What: Provide various programmatic manners to access the component, for deploying edge workloads (UI / API / CLI). Who: Developer. Where: Platform tier (Orchestration management). Why: Introduce a new or refined edge workload to the system.
	Inputs	Edge workload to be deployed or redeployed.
	Outputs	Edge workload deployment request.
	Life-cycle stage	Development.

Table 80. i4Q^{EW} Pre-process incoming requests

Details	Functional Component	Handle incoming request.
	Description	What: Receive an edge workload that has to be deployed to the edge. Who: Developer. Where: Platform tier (Orchestration management). Why: Prepare the internal edge application deployment request.
	Inputs	Edge application to be deployed or redeployed.
	Outputs	Edge application deployment request.
	Life-cycle stage	Development.

Table 81. i4Q^{EW} Handle incoming request

Details	Functional Component	Handle rules and policies.
	Description	What: Receive a new rule or policy to govern workloads placement. Update current deployment criteria. Who: Administrator. Where: Platform tier (Orchestration management). Why: Add or change existing placement rules.
	Inputs	New or changed deployment rule.
	Outputs	Internal state change of the placement rules.

	Life-cycle stage	Configuration.

Table 82. i4Q[EW] Handle rules and policies

Details	Functional Component	Determine proper placement.
	Description	What: Determine proper location for the incoming workload based on monitoring data and policies. Customizable plugin logic. Who: Edge platform. Where: Platform tier (Orchestration management). Why: Determine proper placement of the edge application.
	Inputs	Edge application to be deployed or redeployed.
	Outputs	Edge application deployment placement.
	Life-cycle stage	Development.

Table 83. i4Q[EW] Determine proper placement

Details	Functional Component	Deploy and execute AI workload.
	Description	What: Perform the action of deploying and executing the new workload. Workload life-cycle manager. Who: Edge platform. Where: Platform tier (Distribution services), and Edge tier (Distributed computing). Why: Deploy the edge workload in the designated location.
	Inputs	Edge workload to be deployed or redeployed.
	Outputs	Edge application deployment.
	Life-cycle stage	Deployment.

Table 84. i4Q[EW] Deploy and execute AI workload

Details	Functional Component	Topology and workloads placement viewer.
	Description	What: Aggregate information of workloads managed, and monitoring data. Who: Edge platform. Where: Platform tier (Orchestration management). Why: Monitor the deployment and targets state.
	Inputs	Monitoring data.

	Outputs	State of deployments view.
	Life-cycle stage	Operation.

Table 85. i4Q^{EW} Topology and workloads placement viewer

2.11.3. Mapping of functional components to IIRA functional domain

i4Q^{EW} Edge Workloads Placement and Deployment	
Pre-process incoming requests	Operations
Handle incoming request	Operations
Handle rules and policies	Business
Determine proper placement	Control
Deploy and execute AI workload	Control
Topology and workloads placement viewer	Operations

Table 86. i4Q^{EW} Mapping of functional components to IIRA functional domain

2.12.i4Q[IM] Infrastructure Monitoring

2.12.1. Mapping of Functional Structure Diagram to Reference Architecture

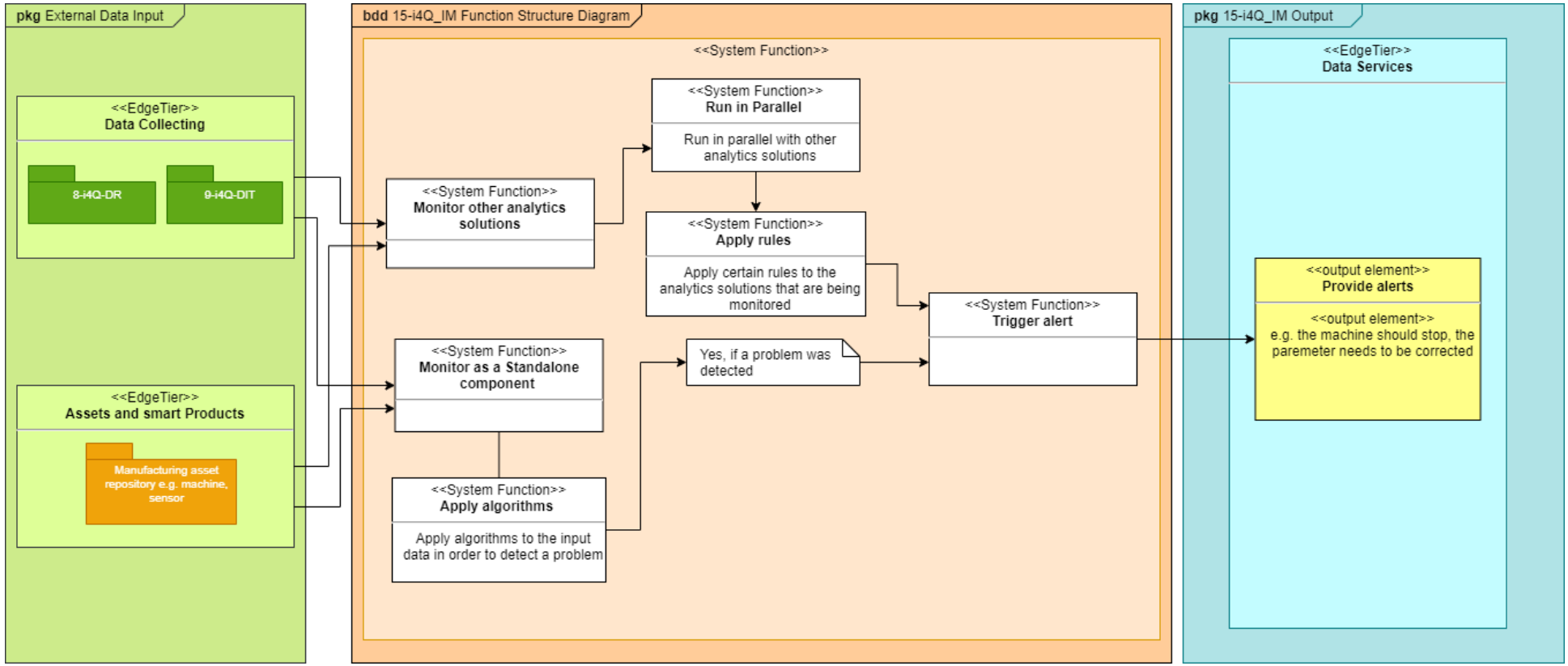

Figure 15. i4Q[IM] Functional Component Diagram

2.12.2. Functional Components

Details	Functional Component	Monitor other analytic solutions.
	Description	What: This component will monitor other analytic solutions and by applying certain rules will identify harmful events. Who: Algorithm provider. Where: Platform Tier (Monitor and diagnostics). Why: Since there are various analytic solutions in the i4Q, that provide predictive algorithms, the i4Q-IM can use the information they offer, for detecting failures.
	Inputs	Algorithms from other analytical solutions, that provide information about the processes that need to be monitored, 8-i4Q_DR, 9-i4Q_DIT, sensor data.
	Outputs	Alerts (e.g., in the form of a notification message).
	Life-cycle stage	Deployment.

Table 87. i4Q^{IM} Monitor other analytics solutions

Details	Functional Component	Run in parallel.
	Description	What: This component will allow the parallel examination of multiple analytical results. Who: Algorithm provider. Where: Edge Tier (Data Management). Why: This component offers multifactorial monitoring to achieve a more effective fault detection system.
	Inputs	Analytical solution outputs.
	Outputs	Processed analytical solution outputs.
	Lifecycle stage	Operation.

Table 88. i4Q^{IM} Run in parallel

Details	Functional Component	Apply rules.
	Description	What: This component will apply certain rules to the outputs of other analytic solutions to identify harmful events. Who: Algorithm provider. Where: Platform Tier (Models Management and Services).

		Why: This component allows the solution to detect/ predict possible failures in cooperation with other analytical solutions.
	Inputs	Processed analytical solution outputs.
	Outputs	Boolean. “0”: Do not trigger alert, “1”: trigger alert.
	Lifecycle stage	Development.

Table 89. i4Q^{IM} Apply rules

Details	Functional Component	Monitor as a standalone.
	Description	What: This component will have its own algorithms needed for detecting failures and harmful events. Who: Algorithm provider. Where: Platform Tier (Monitor and diagnostics). Why: This component allows the solution to work as a standalone and be used individually in the pilots.
	Inputs	Sensor data, 8-i4Q_DR, 9-i4Q_DIT.
	Outputs	Alerts.
	Life-cycle stage	Deployment.

Table 90. i4Q^{IM} Monitor as a standalone

Details	Functional Component	Apply algorithms.
	Description	What: This component will apply algorithms on real-time input data (sensors) and on data collected by 8-i4Q_DR and 9-i4Q_DIT solutions, in order to detect/predict an imminent problem. Who: Algorithm provider. Where: Platform Tier (Models Management and Services). Why: This component allows the solution to detect possible failures when it works as a standalone.
	Inputs	Processed Sensor data, 8-i4Q_DR, and 9-i4Q_DIT data.
	Outputs	Boolean. “0”: Do not trigger alert, “1”: trigger alert.
	Lifecycle stage	Development.

Table 91. i4Q^{IM} Monitor as a standalone

Details	Functional Component	Trigger alert.
	Description	What: This component will initiate alerts about the detection or prediction of harmful events and alert other solutions if corrective actions are needed. Who: Algorithm provider. Where: Platform Tier (Monitor and diagnostics). Why: To initiate the alert.
	Inputs	Boolean value derived either from monitoring other analytic solutions or monitoring as a standalone operation mode ("0": do not trigger alert, "1": trigger alert).
	Outputs	If the input is "1" provide an alert signal.
	Life-cycle stage	Deployment.

Table 92. i4Q^{IM} Trigger alert

Details	Functional Component	Provide alert.
	Description	What: This component will 'send' the alert to the users and inform them about a failure. Who: Internal to the solution. Where: Platform tier (Data Brokering and Storage). Why: To inform the machine operators about the harmful event in order to take corrective actions.
	Inputs	Alert signal.
	Outputs	Alert.
	Life-cycle stage	Deployment.

Table 93. i4Q^{IM} Provide alert

2.12.3. Mapping of functional components to IIRA functional domain

i4Q^{IM} Infrastructure Monitoring	
Monitor other analytic solutions	Deployment
Run in parallel	Operation
Apply rules	Development
Monitor as a standalone	Deployment
Apply algorithms	Development
Tigger alert	Deployment

Provide alert	Deployment

Table 94. i4Q[IM] Mapping of functional components to IIRA functional domain

2.13.i4Q^{DT} Digital Twin Simulation Services

2.13.1. Mapping of Functional Structure Diagram to Reference Architecture

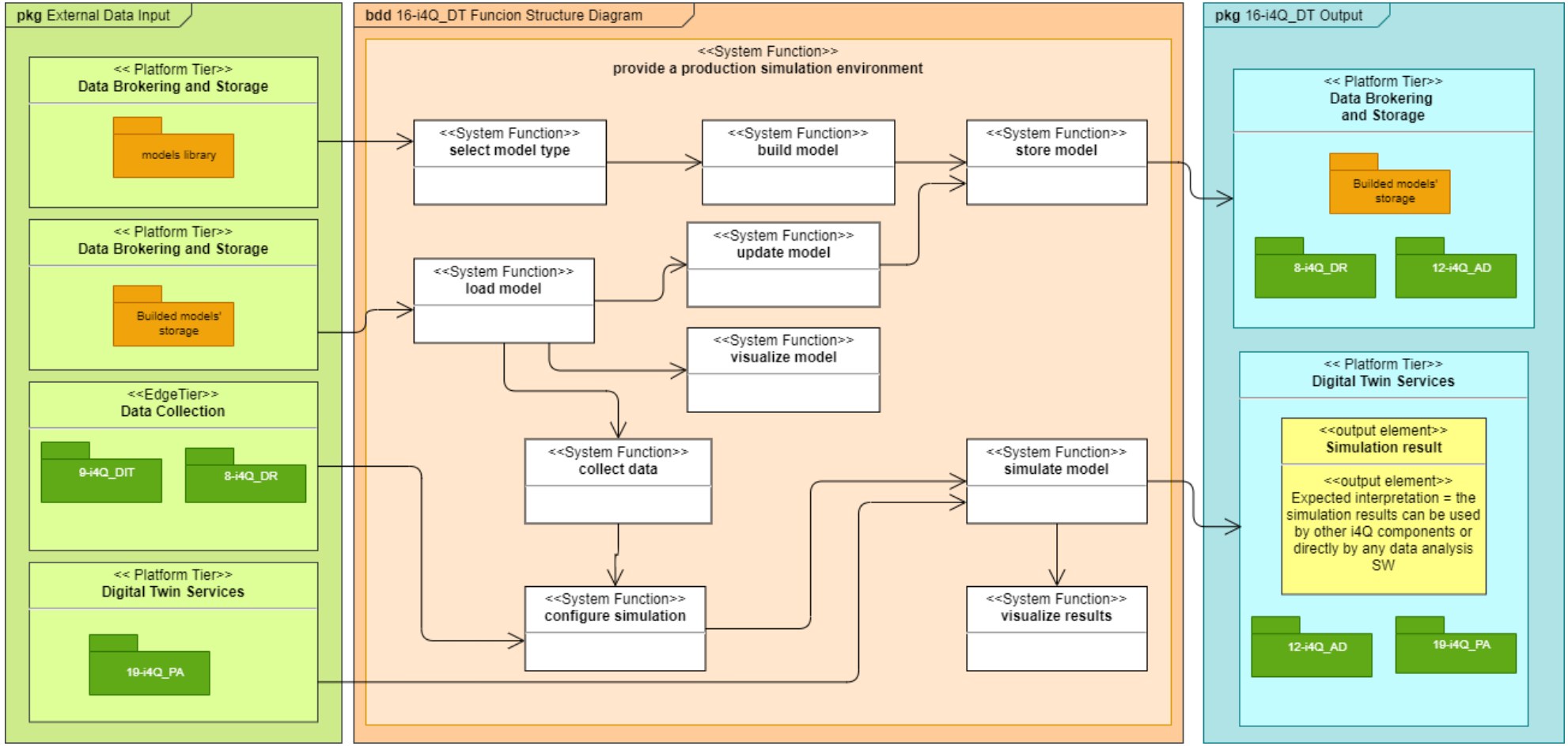

Figure 16. i4Q^{DT} Funcitional Component Diagram

2.13.2. Functional Components

Details	Functional Component	Select model type.
	Description	What: This function selects the typology of the model from a set of types of models. Who: Developer (system engineer). Where: Edge/Platform tier. Why: The selection of the model type implies the use of different subfunctions of the "build model" functional component.
	Inputs	Types of models (physics-based or data-driven, plant configuration, specific machine or processes).
	Outputs	Model type selected.
	Life-cycle stage	Operation.

Table 95. i4Q^{DT} Select model type

Details	Functional Component	Build model.
	Description	What: This function builds the model, taking as a basis the model type selection and a number of predefined models of the selected type. In some cases, data will also be necessary to construct the model. Who: Developer (system engineer). Where: Edge/Platform tier. Why: The Digital Twin consists of a model that needs to be generated.
	Inputs	Model type selection, model libraries and, in case of data-driven approaches, training data.
	Outputs	Built model.
	Life-cycle stage	Operation.

Table 96. i4Q^{DT} Build model

Details	Functional Component	Store model.
	Description	What: This function stores the model generated in the "Build model" functional component. Who: Developer (system engineer). Where: Edge/Platform tier.

		Why: The models need to be stored to be used afterwards, even if they are used in i4Q-DT or in other i4Q solutions.
	Inputs	Built model.
	Outputs	File in standardized format (FMU for physics-based models, ONNX for data-driven models, etc.).
	Life-cycle stage	Operation.

Table 97. i4Q^{DT} Store model

Details	Functional Component	Load model.
	Description	What: This function loads a model from those that have been built previously. Who: Production Team. Where: Edge/Platform tier. Why: In order to launch the digital twin, the model behind it needs to be selected and loaded.
	Inputs	List of built models.
	Outputs	Model selected.
	Life-cycle stage	Operation.

Table 98. i4Q^{DT} Load model

Details	Functional Component	Update model.
	Description	What: This function updates the parameters of the model according to data that represents with more fidelity the reality of the system to be modelled. Who: Developer (system engineer). Where: Edge/Platform tier. Why: The model needs to be as accurate as possible to obtain the best results.
	Inputs	Built model and data collected from previous functional components.
	Outputs	Built (updated) model.
	Life-cycle stage	Operation

Table 99. i4Q^{DT} Update model

Details	Functional Component	Visualize model

	Description	What: This function provides a visualization of the model. Who: Production Team. Where: Edge/Platform tier. Why: The visualization of the model allows to understand the model more clearly.
	Inputs	Built model from previous functional components.
	Outputs	-
	Life-cycle stage	Operation.

Table 100. i4Q^{DT} Visualize model

Details	Functional Component	Collect data.
	Description	What: This function selects the origin of the data and it loads them. Who: Production Team. Where: Edge tier. Why: There is a need of data to conduct the simulations.
	Inputs	Sensor/machine data stored in the data repository or produced from i4Q-DIT. The sensor readings have the form of time series.
	Outputs	Data (dataframe).
	Life-cycle stage	Operation.

Table 101. i4Q^{DT} Collect data

Details	Functional Component	Configure simulation.
	Description	What: This function allows the user to select the model parameters as well as the simulation parameters. Who: Production Team. Where: Edge Tier. Why: The virtual representation provided by the digital twin needs to be limited to the simulation requirements of the user.
	Inputs	Built model and data collected from previous functional components.

	Outputs	Configuration parameters (vector of values defining the model and simulation parameters).
	Life-cycle stage	Operation.

Table 102. i4Q^{DT} Configure simulation

Details	Functional Component	Simulate model.
	Description	What: This function performs the simulation of the digital twin. Who: Production Team. Where: Edge/Platform tier. Why: Simulations need to be performed to obtain the results from the digital twin.
	Inputs	Built model, data collected and configuration parameters from previous functional components.
	Outputs	Simulation results, in general in the form of time series (dataframe) or specific KPIs that can be computed by the model (dataframe).
	Life-cycle stage	Operation.

Table 103. i4Q^{DT} Simulate model

Details	Functional Component	Visualize results
	Description	What: This function visualizes the results of the simulation. Who: Production Team. Where: Edge/Platform tier. Why: The results need to be evaluated to consider their validity.
	Inputs	Simulation results obtained in previous functional components (time series, KPIs) in the form of dataframes.
	Outputs	-
	Life-cycle stage	Operation.

Table 104. i4Q^{DT} Visualize results

2.13.3. Mapping of functional components to IIRA functional domain

i4Q^{DT} Digital Twin Simulation Services	
Select model type	Operations

Build model	Application
Store model	Application
Load model	Application
Update model	Application
Visualize model	Application
Collect data	Information
Configure simulation	Business
Simulate model	Operations
Visualize results	Business

Table 105. i4Q^{DT} Mapping of functional components to IIRA functional domain

2.14.i4Q[PQ] Data-driven Continuous Process Qualification

2.14.1. Mapping of Functional Structure Diagram to Reference Architecture

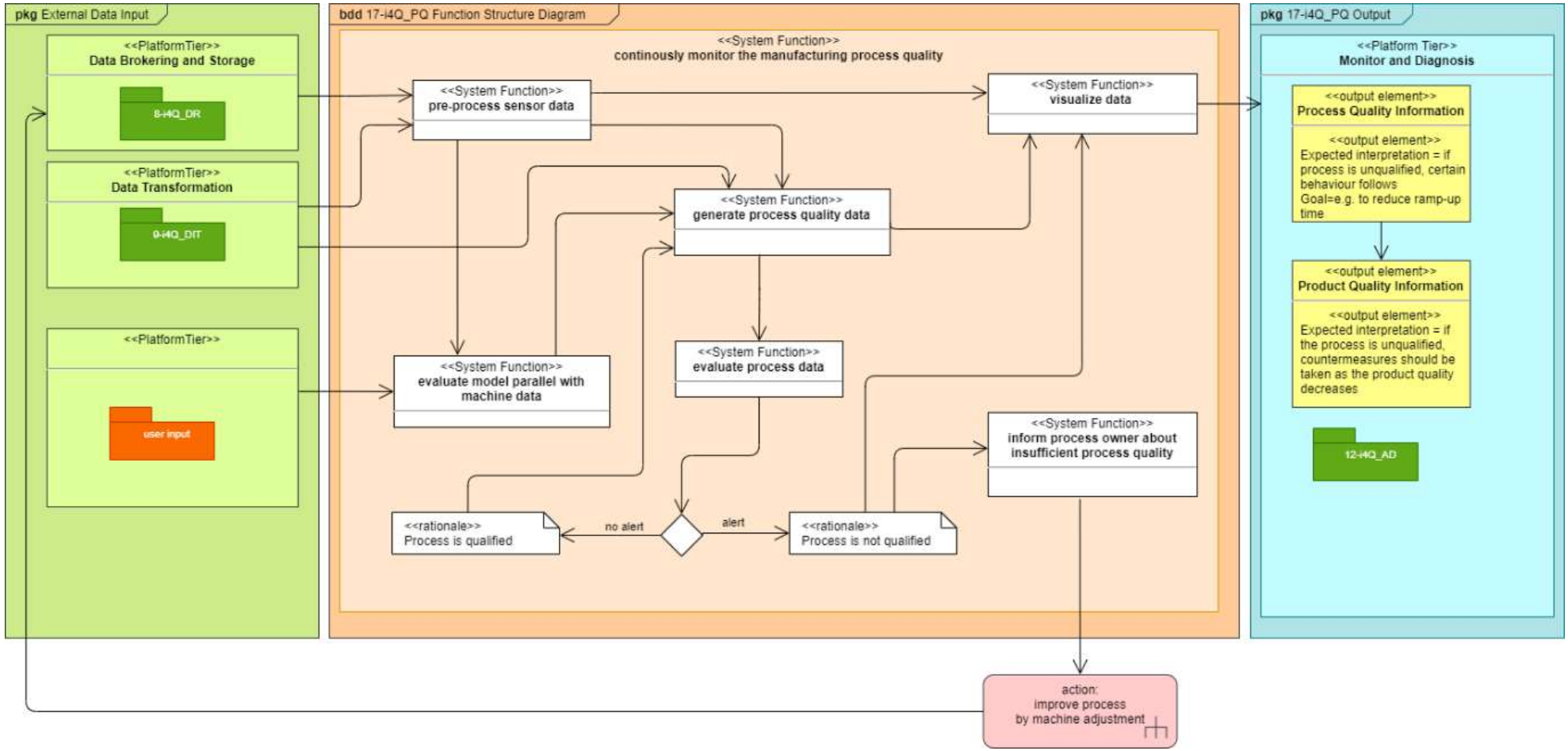

Figure 17. i4Q[PQ] Functional Component Diagram

2.14.2. Functional Components

Details	Functional Component	Pre-process sensor data.
	Description	What: In terms of finding the right scales for the solution to proceed, i4Q-PQ will have an individual step of data processing for the individual Algorithm. Who: Algorithm provider. Where: Edge Tier. Why: Feeding Algorithm with proper data.
	Inputs	All recorded sensor data in their original format but in an already processed way (e.g. 70°C) to further manipulate the data. Some steps involved might be typical data transformation for neural networks like scaling, grouping, one hot encoding… The data structure should be easy to use data frames in csv-format.
	Outputs	All recorded sensor data in their processed format.
	Life-cycle stage	Development/Operation.

Table 106. i4Q^{PQ} Pre-process sensor data

Details	Functional Component	Generate process quality data.
	Description	What: Finding a prediction with an individual time frame ahead of real time. Who: Algorithm provider. Where: Platform/Edge Tier. Why: Prediction is numerical value in- or outside the accepted range of quality conformity.
	Inputs	Model which was learned by historical data and saved in keras. The architecture of the model is saved in JSON or YAML. The weights are in a multi-grid data type HDF5. Real-time features are fed into the network using processed csv data.
	Outputs	Numerical value.
	Life-cycle stage	Deployment.

Table 107. i4Q^{PQ} Generate process quality data

Details	Functional Component	Evaluate process quality data.

	Description	What: Applying a rule to the Algorithms' output by comparing it to the individual range of quality conformity. Who: Algorithm provider. Where: Platform/Edge Tier. Why: If the predicted value is outside the quality conformity range, an alert is sent to an operator.
	Inputs	Predicted single numerical value (e.g. 20.015) which is the output from the Algorithm and the given user input by the process owner who defines the range of the conformity level by two single values (e.g. 19 and 21).
	Outputs	Information or alert will be sent when outside range or no alert when inside range.
	Life-cycle stage	Deployment.

Table 108. i4Q^{PQ} Evaluate process quality data

Details	Functional Component	Visualize data.
	Description	What: Output of the Algorithm and if necessary raw data can be plotted in an understandable way. Who: Algorithm provider, Visualization Software. Where: Platform Tier. Why: To ensure interaction and visibility with the solution and the process capability.
	Inputs	Pre-processed sensor data, algorithm output and conformity range.
	Outputs	Convenient visualization with a scorecard to see if the process is not in the desired range of quality. The critical limits are visualized in red, the prediction in green and the historical process quality is in grey.
	Life-cycle stage	Operation.

Table 109. i4Q^{PQ} Visualize data

Details	Functional Component	Evaluate model parallel with new machine data.
	Description	What: Constant evaluation by theoretically feeding new dependent variables into a parallel running algorithm for further evaluation of better models. Who: Algorithm provider.

		Where: Platform Tier. Why: To adapt model to new data over time during the production process.
	Inputs	Model which was learned by historical data and saved in keras. The architecture of the model is saved in JSON or YAML. The weights are in a multi-grid data type HDF5. Additionally, it includes numerical values from the neural networks' evaluation criteria like cost functions and/or accuracy.
	Outputs	Better trained model.
	Life-cycle stage	Development

Table 110. i4Q^{PQ} Evaluate model parallel with new machine data

Details	Functional Component	Inform process owner about insufficient process quality.
	Description	What: Process owner needs to be informed about instability of the process. Who: Operator, Process Engineer. Where: Platform/Edge Tier. Why: To reduce amount of failure parts.
	Inputs	Validation result.
	Outputs	An alert which needs to be defined by the use case. Possible functionality can be a message to a phone number, a direct link with an automated email or a warning sign in the machine.
	Life-cycle stage	Operation.

Table 111. i4Q^{PQ} Inform process owner about insufficient process quality

2.14.3. Mapping of functional components to IIRA functional domain

i4QPQ Data-driven Continuous Process Qualification	
Pre-process sensor data	Control
Generate process quality data	Application
Evaluate process quality data	Application
Visualize data	Operations
Evaluate model parallel with new machine data	Control

Inform process owner about insufficient process quality	Application

Table 112. i4Q^{PQ} Mapping of functional components to IIRA functional domain

2.15.i4Q^{QD} Rapid Quality Diagnosis

2.15.1. Mapping of Functional Structure Diagram to Reference Architecture

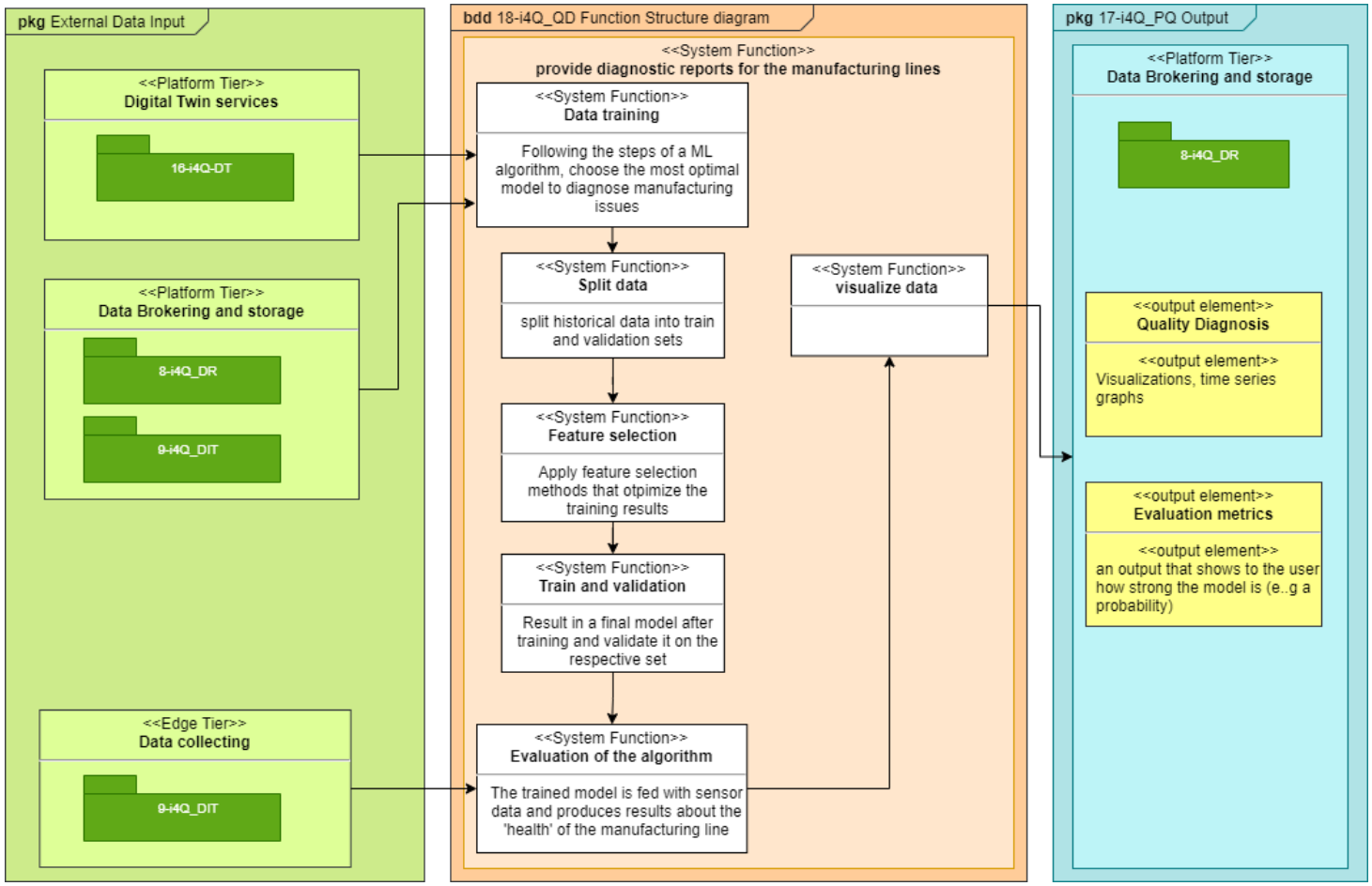

Figure 18. i4Q^{QD} Functional Component Diagram

2.15.2. Functional Components

Details	Functional Component	Data training.
	Description	What: This component, that is split into sub-components, trains various models on the (historical) data, in order to result to a final model. Who: Developer/ Algorithm provider. Where: Platform Tier (Data analytics and services). Why: Different models need to be trained in order to see which is the most suitable for extracting information from the data.
	Inputs	Data collected from 16-i4Q_DT, 8-i4Q_DR, 9-i4Q_DIT solutions constitute the input data.
	Outputs	Trained model.
	Life-cycle stage	Development.

Table 113. i4Q^{QD} Data training

Details	Functional Component	Split data.
	Description	What: The initial input data are divided to a training and validation set. Who: Developer/ Algorithm provider. Where: Edge Tier (Data management). Why: The training set is needed for training the model and the validation set is used to apply the trained model and maybe fine tune the model's parameters.
	Inputs	A dataset with the input data.
	Outputs	Train and validation datasets.
	Life-cycle stage	Development.

Table 114. i4Q^{QD} Split Data

Details	Functional Component	Feature selection.
	Description	What: This component includes various algorithms that choose the most valuable variables for a prediction/classification model. Who: Developer/ Algorithm provider. Where: Platform Tier (Data analytics and services). Why: To eliminate the volume of the initial data and

		improve the performance of the algorithms.
	Inputs	Train and validation datasets.
	Outputs	Data including the most contributing features.
	Life-cycle stage	Development.

Table 115. i4Q^{QD} Feature selection

Details	Functional Component	Train and validation.
	Description	What: This component will use the trained model on the validation set in order to fine tune the model's parameters. Who: Developer/ Algorithm provider. Where: Platform Tier (Data analytics and services). Why: To further improve the performance of the trained model.
	Inputs	Pre-processed datasets.
	Outputs	Final models.
	Life-cycle stage	Development.

Table 116. i4Q^{QD} Train and validation

Details	Functional Component	Evaluation of the algorithm.
	Description	What: This component will apply the trained model on the incoming (real-time) data to perform the diagnosis. Who: Developer, data scientist (users)/ Algorithm provider. Where: Platform Tier (Data analytics and services). Why: An algorithmic procedure is crucial for this solution, to make some decisions about the quality diagnosis.
	Inputs	Final models and data collected from 9-i4Q_DIT.
	Outputs	Algorithm output (e.g., output characterizing the diagnosis).
	Life-cycle stage	Deployment.

Table 117. i4Q^{QD} Evaluation of the Algorithm

Details	Functional Component	Visualize data.
	Description	What: This component is responsible for the visual representation of the quality diagnosis and the generation of insightful evaluation metrics. Who: Developer, data scientist (users)/ Algorithm provider. Where: Platform tier (Data analytics and services). Why: Assist the machine operators in monitoring and evaluating the condition of the production line through visual means.
	Inputs	Evaluation results.
	Outputs	Evaluation metrics and results.
	Life-cycle stage	Deployment.

Table 118. i4Q^{QD} Visualize data

Details	Functional Component	Quality diagnosis.
	Description	What: This component provides a graphical representation of the quality diagnosis results. Who: Developer, data scientist (users) / Algorithm provider. Where: Platform tier (Data brokering and storage). Why: Visualizations facilitate the observation of data over time and provide a more eligible way of examining information about the manufacturing line.
	Inputs	Evaluation results.
	Outputs	Graphical representation of evaluation results.
	Life-cycle stage	Deployment.

Table 119. i4Q^{QD} Quality diagnosis

Details	Functional Component	Evaluation metrics.
	Description	What: This component presents and stores various evaluation metrics resulting from the quality diagnosis. Who: Developer, data scientist (users) / Algorithm provider.

		Where: Platform tier (Data brokering and storage). Why: Evaluation metrics offer detailed information about the quality of the manufacturing process in a more statistical way.
	Inputs	Evaluation results.
	Outputs	Display and store evaluation metrics.
	Life-cycle stage	Deployment.

Table 120. i4Q^{QD} Evaluation metrics

2.15.3. Mapping of functional components to IIRA functional domain

i4Q^{QD} Rapid Quality Diagnosis	
Data training	Development
Split data	Development
Feature selection	Development
Train and validation	Development
Evaluation of the algorithm	Deployment
Visualize data	Deployment
Quality diagnosis	Deployment
Evaluation metrics	Deployment

Table 121. i4QQD Mapping of functional components to IIRA functional domain

2.16.i4Q[PA] Prescriptive Analysis Tools

2.16.1. Mapping of Functional Structure Diagram to Reference Architecture

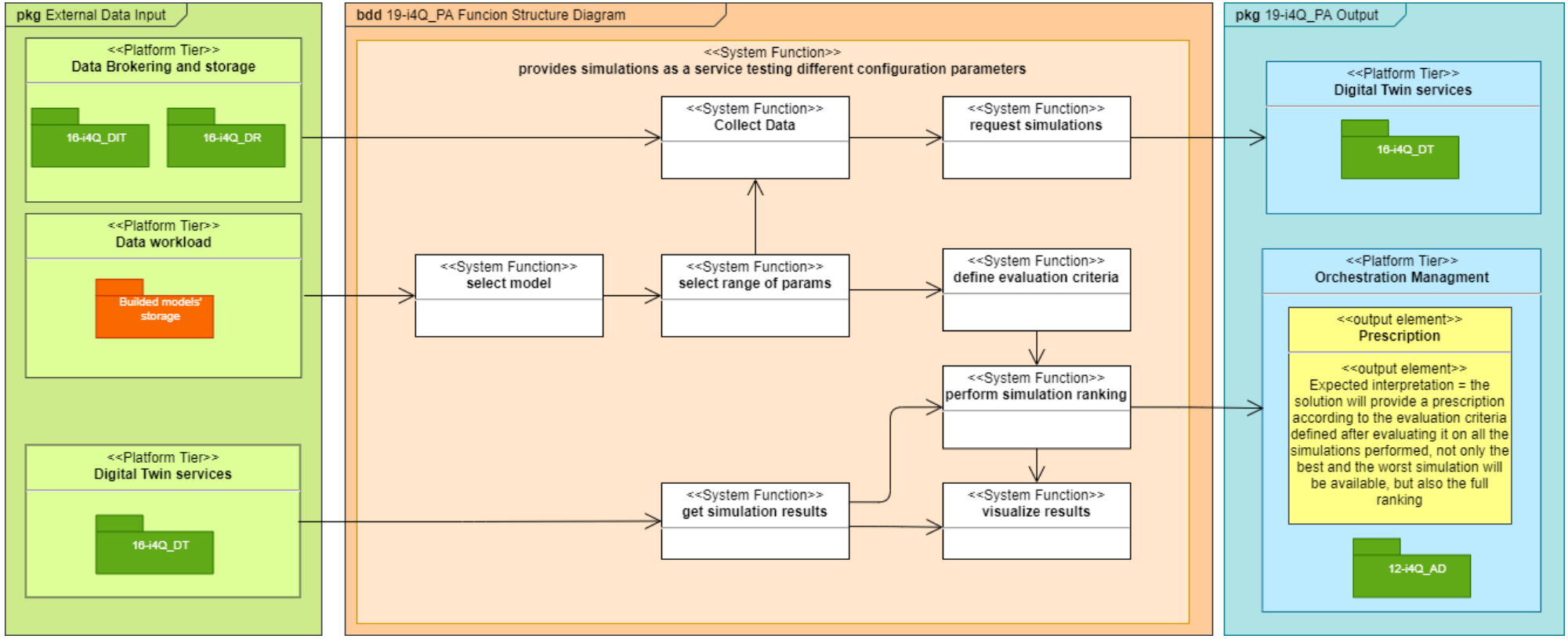

Figure 19. i4Q[PA] 20Functional Component Diagram

2.16.2. Functional Components

Details	Functional Component	Select Model.
	Description	What: This function selects and loads a model from those that have been built previously. Who: Production/Maintenance Team. Where: Edge/Platform Tier. Why: In order to launch the digital twin simulations, the model behind it needs to be selected and loaded, this will allow us to identify the inputs and parameters it has for next steps.
	Inputs	Built model.
	Outputs	Model selected (characterized by its inputs, outputs and parameters).
	Life-cycle stage	Operation.

Table 122. i4Q^{PA} Select Model

Details	Functional Component	Select Range of Parameters.
	Description	What: Among all the parameters of the model the user will select the set of parameters that will vary and will specify the specific ranges for each of them. Who: Production/Maintenance Team. Where: Edge/Platform Tier. Why: The parameters to vary and the ranges need to be defined in order to configure the simulations to be run.
	Inputs	Parameters/Metadata from the model.
	Outputs	Parameters that will vary in the simulation (minimum value, maximum value and step).
	Life-cycle stage	Operation.

Table 123. i4Q^{PA} Select Model

Details	Functional Component	Collect Data.
	Description	What: This function selects the origin of the data and it loads them. Who: Production/Maintenance Team. Where: Edge tier.

		Why: There is a need of data to conduct the simulations.
	Inputs	Data repositories.
	Outputs	It returns all the data necessary for the correct use of the simulations.
	Life-cycle stage	Operation.

Table 124. i4Q[PA] Collect Data

Details	Functional Component	Request simulations.
	Description	What: This function configures each simulation according to the ranges of parameters defined and launches calls to the Digital Twin in order to perform each needed simulation. Who: Production/Maintenance Team. Where: Edge/Platform Tier. Why: Each simulation needs to be run in order to evaluate it.
	Inputs	Built model, Ranges of parameters.
	Outputs	Simulation request.
	Life-cycle stage	Operation.

Table 125. i4Q[PA] Request simulations

Details	Functional Component	Define evaluation criteria.
	Description	What: an objective function will be defined taking into account the outputs the Digital Twin can provide, as well as the optimization criteria to be applied. Who: Production/Maintenance Team. Where: Edge/Platform Tier. Why: This will allow to evaluate each simulation and perform the ranking.
	Inputs	Digital Twin outputs.
	Outputs	Objective function, optimization criteria.
	Life-cycle stage	Operation.

Table 126. i4Q[PA] Define evaluation criteria

Details	Functional Component	Get simulation results.
	Description	What: This function received the output of the Digital Twin. Who: Internal to the solution. Where: Edge/Platform Tier. Why: The simulation result needs to be transferred to the PA solution.
	Inputs	Request output from Digital Twin.
	Outputs	Simulation results.
	Life-cycle stage	Operation.

Table 127. i4Q^{PA} Get simulation results

Details	Functional Component	Perform Simulation ranking.
	Description	What: As a first step, the evaluation criteria need to be computed for every simulation and then they will be ordered in ascending/descending order. Who: Internal to the solution. Where: Edge/Platform Tier. Why: The ranking needs to be performed in order to let the user know the best/worst scenarios.
	Inputs	Simulation results.
	Outputs	Ranking of all simulations performed.
	Life-cycle stage	Operation.

Table 128. i4Q^{PA} Perform Simulation ranking

Details	Functional Component	Visualize results.
	Description	What: This function visualizes the results of the simulation and/or the evaluation result of each simulation according to the evaluation criteria previously defined. Who: Production/Maintenance Team. Where: Edge/Platform Tier. Why: The results need to be evaluated to consider their validity.

	Inputs	Simulation results, criteria evaluation for each simulation, ranking performed.
	Outputs	It provides insight into the data for users.
	Life-cycle stage	Operation.

Table 129. i4Q[PA] Visualize results

2.16.3. Mapping of functional components to IIRA functional domain

i4QPA Prescriptive Analysis Tools	
Select Model	Operations
Select Range of Parameters	Operations
Collect Data	Information
Request simulations	Information
Define evaluation criteria	Operations
Get simulation results	Business
Perform Simulation ranking	Business
Visualize results	Operations

Table 130. i4Q[PA] Mapping of functional components to IIRA functional domain

2.17.i4Q[LRT] Manufacturing Line Reconfiguration Toolkit

2.17.1. Mapping of Functional Structure Diagram to Reference Architecture

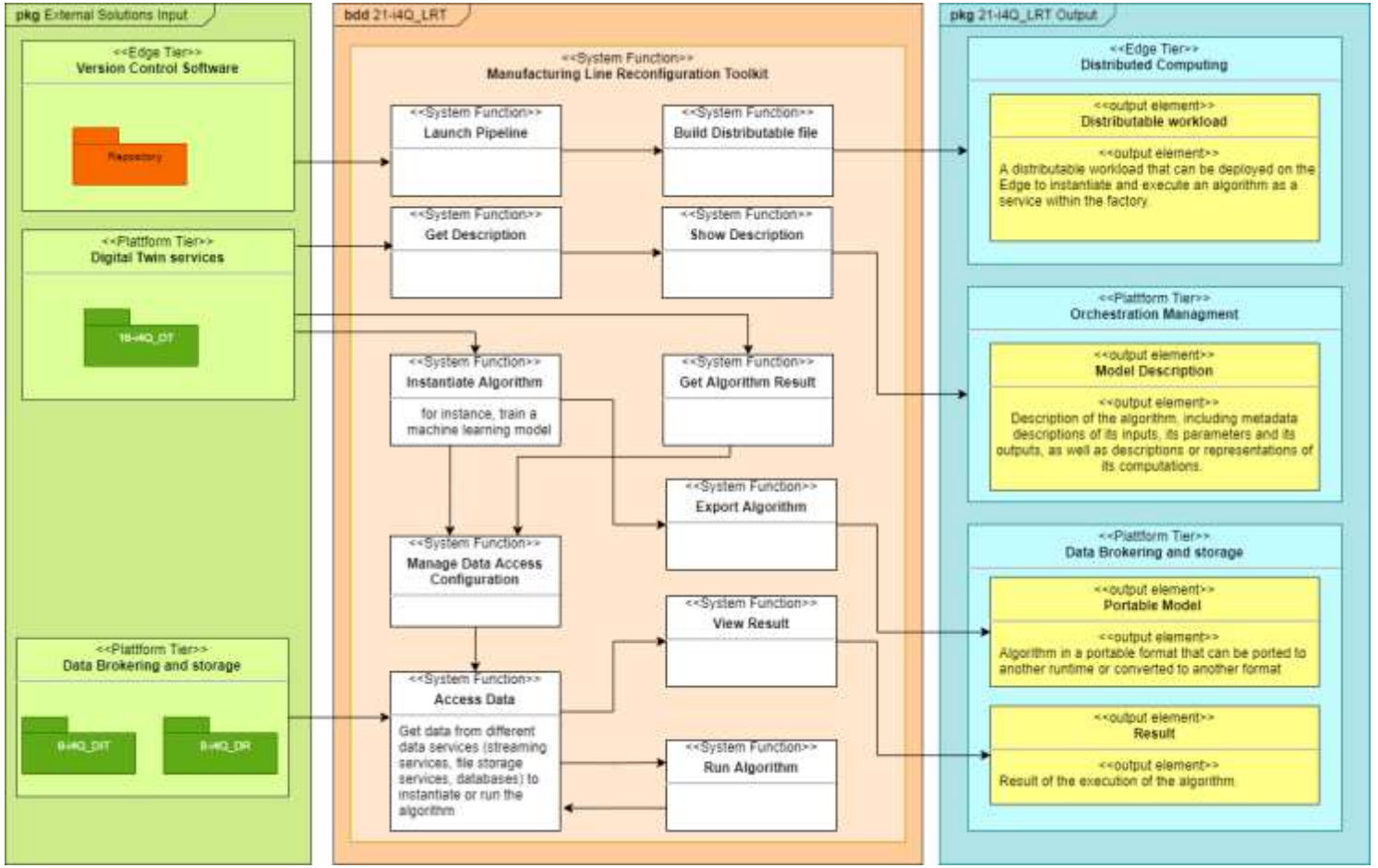

Figure 21. i4Q[LRT] Functional Component Diagram

2.17.2. Functional Components

Details	Functional Component	Launch pipeline.
	Description	What: This component is responsible for implementing the process that automates the generation of a distributable file. Who: Version Control Software. Where: Edge Tier. Why: This component is necessary to automate the image creation process under predefined characteristics.
	Inputs	--
	Outputs	--
	Life-cycle stage	Deployment.

Table 131. i4Q[LRT] Launch pipeline

Details	Functional Component	Build Distributable file.
	Description	What: This function provides a distributable file. This file can be deployed on the Edge to instantiate and execute algorithm as a service within the factory. Who: Any solution that can deploy distributable file. Where: Edge Tier. Why: To deploy this service within the factory and offer the algorithms as a service.
	Inputs	--
	Outputs	A distributable workload that can be deployed on the Edge to instantiate and execute an algorithm as a service within the factory.
	Life-cycle stage	Deployment.

Table 132. i4Q[LRT] GeLRT Build Distributable file

Details	Functional Component	Get Description.
	Description	What: This function provides information about the algorithm, including a description of inputs and outputs. Who: Any solution that requires information from an algorithm. Where: Edge/Platform Tier.

		Why: To learn how to use the algorithm.
	Inputs	Call to the service with the tag "/metadata" without any accompanying data.
	Outputs	--
	Life-cycle stage	Operation.

Table 133. i4Q[LRT] Describe Algorithm

Details	Functional Component	Show Description.
	Description	What: Return description of the algorithm, including metadata descriptions of its inputs, its parameters and its outputs, as well as descriptions or representations of its computations. Who: Any solution that requires information from an algorithm. Where: Edge/Platform Tier. Why: Allows to obtain information from the algorithm. With this component, he end-user will receive the information in a form understandable to him.
	Inputs	--
	Outputs	• Id: The "id" here refers to the key of the returned dictionary. • Description: description about algorithm. • Attributes: Describes all the inputs, parameters and outputs that the algorithm has.
	Life-cycle stage	Operation.

Table 134. i4Q[LRT] Show Description

Details	Functional Component	Instantiate Algorithm.
	Description	What: This function allows train a compute unit class with methods to instantiate and run the specific algorithm. Who: Any solution that requires to instantiate Algorithm for instance or train a machine learning model. Where: Edge/Platform Tier.

		Why: To instantiate algorithms in this service.
	Inputs	Id: The "id" here refers to the key of the returned dictionary. Format: Specifies the format of the input data parameter. Depending on the value of key "format", the key should be replaced with one of the following values: • filename: (Optional) For file/csv inputs, specifies the file name of the file containing the input data. Currently the files must be available in a Docker-volume made available using Docker-compose. Later the CSV files will be in the i4QDR. • mongo-collection: (Optional) For database/mongo inputs, specifies the collection used to fetch the data. Currently the collections are served from a locally running MongoDB database. Once the Storage component is available the database will reside there. • collection: (Optional) For collection data, contains the collection (data) in json format. • mqtt (Optional) For message bus data, is a dictionary with a key "topic" for message bus topic name, and a key "history length" for ring buffer (which stores the stream data) length.
	Outputs	--
	Life-cycle stage	Operation.

Table 135. i4Q[LRT] Instantiate Algorithm

Details	Functional Component	Export Algorithm.
	Description	What: A method to be able to take algorithms instantiated with LRT and export them for other runtimes. Who: Any solution that requires an Algorithm. Where: Edge/Platform Tier. Why: It makes it easy to export algorithms to other

		machines or servers in the fog. These algorithms can already be trained and tested.
	Inputs	--
	Outputs	Algorithm in a portable format that can be ported to another runtime or converted to another format.
	Life-cycle stage	Operation.

Table 136. i4Q[LRT] Export Algorithm

Details	Functional Component	Manage Data Access Configuration.
	Description	What: This function is responsible for managing the configuration data of the service call to use the client correctly. Who: Any solution that requires to instantiate or run Algorithm. Where: Edge/Platform Tier. Why: It allows us to handle different types of data.
	Inputs	--
	Outputs	--
	Life-cycle stage	Operation.

Table 137. i4Q[LRT] Manage Data Access Configuration

Details	Functional Component	Get Algorithm Result.
	Description	What: This function is called to query algorithms and view their results live. Who: Any solution that requires to run an Algorithm. Where: Edge/Platform Tier. Why: To execute algorithm live.
	Inputs	• Has the same input as "Instantiate Algorithm" Component.
	Outputs	--
	Life-cycle stage	Operation.

Table 138. i4Q[LRT] Get Algorithm Result

Details	Functional Component	Access Data.

	Description	What: Get data from different data services (streaming services, file storage services, databases) to instantiate or run the algorithm. Who: Any solution that requires to instantiate or run an Algorithm. Where: Edge/Platform Tier. Why: To provide and save data.
	Inputs	• Filename: (Optional) For file/csv inputs, specifies the file name of the file containing the input data. Currently the files must be available in a Docker-volume made available using Docker-compose. Later the CSV files will be in the Storage component. • Mongo-collection: (Optional) For database/mongo inputs, specifies the collection used to fetch the data. Currently the collections are served from a locally running MongoDB database. Once the Storage component is available the database will reside there. • Collection: (Optional) For collection data, contains the collection (data) in json format. • Mqtt (Optional) For message bus data, is a dictionary with a key "topic" for message bus topic name, and a key "history Length" for ring buffer (which stores the stream data) length.
	Outputs	--
	Life-cycle stage	Operation.

Table 139. i4Q[LRT] Access Data

Details	Functional Component	View Result.
	Description	What: Return result of the execution of the algorithm. Who: Any solution that requires to run an Algorithm. Where: Edge/Platform Tier. Why: This component that returns the results after being run by the algorithm and thus can be used by different clients or users.
	Inputs	--
	Outputs	• Id: The compute Unit returns the results as a

		python dictionary. The “id” here refers to the key of the returned dictionary. • Format: Specifies the format of the output data parameter. Depending on the value of the "format" key, (from the input of the Access data component), the key must be replaced by one of the keys defined above (filename, Mongo-Collection, Collection data, Mqtt).
	Life-cycle stage	Operation.

Table 140. i4Q[LRT] View Result

Details	Functional Component	Run Algorithm.
	Description	What: This function run the algorithm with the input configuration and data defined taken in the request. Who: Any solution that requires to instantiate or run an Algorithm. Where: Edge/Platform Tier. Why: To process the data in the algorithm.
	Inputs	--
	Outputs	--
	Life-cycle stage	Operation.

Table 141. i4Q[LRT] Run Algorithm

2.17.3. Mapping of functional components to IIRA functional domain

i4Q[LRT] Manufacturing Line Reconfiguration Toolkit	
Launch Pipeline	Application
Build Distributable file	Operations
Get Description	Information
Show Description	Business
Instantiate Algorithm	Control
Export Algorithm	Operations
Get algorithm Result	Information
Manage Data Access Configuration	Control
View Result	Business
Access Data	Control
Run Algorithm	Application

Table 142. i4Q[LRT] Mapping of functional components to IIRA functional domain

3. Digital Models and Ontologies

Throughout this section different digital models and ontologies that can be used to support the data, control and workload distribution flows will be analysed, within the defined architecture and the proposed solutions. To this purpose, this section factors in the inputs and outputs defined above and the standards, data models and ontologies identified in D2.2 to define models and ontologies to be used in each communication flow.

3.1. Data Flow

This section describes the main six data flows between the different functional components of the architecture and the relationship between data flows and i4Q solutions. It describes the requirements for the message structure at each communication interface and gives pointers to related standards that can be used.

The collection, processing and utilisation of data is the basis for the use of the different i4Q solutions. This section analyses the flow of data between the different solutions, from the asset and intelligent product level to the platform level. The section highlights how the different solutions manage the entire life-cycle of manufacturing data, implementing functions to store, clean or provision. The components or systems that implement data collection functions are often deployed close to the physical systems they control (edge tier).

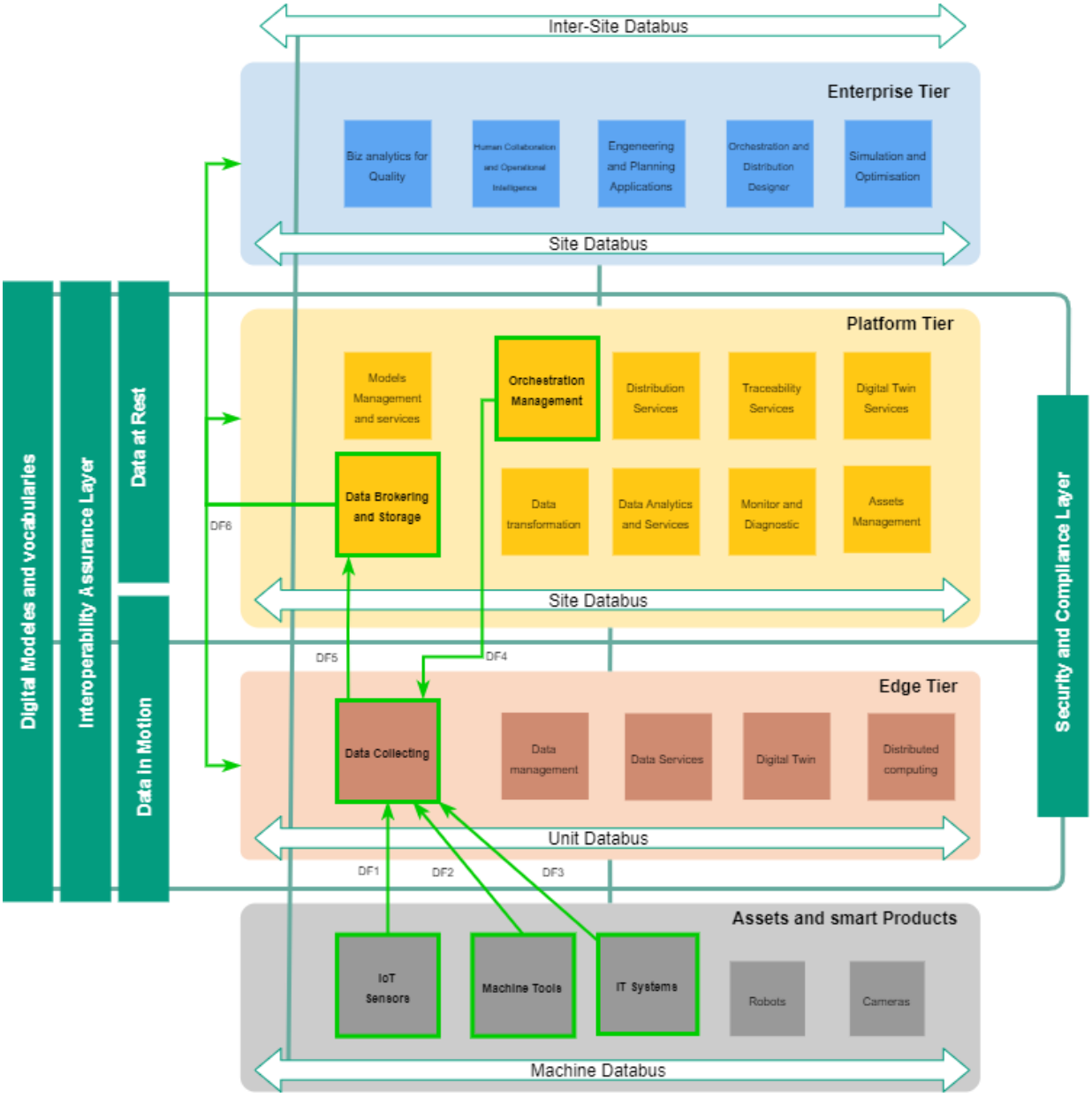

Figure 22. Data Flow

Details	Flow	DF1
	Components	From: IoT Sensors To: Data Collecting
	Solutions	• 4-i4QTN • 9-i4QDiT
	Message Data Structure	The messages exchanged through this interface primarily involve IoT sensor data readings collected from sensors in the field (e.g. condition monitoring sensors or environmental sensors), either directly or through proxies, gateway systems, or industrial

		controllers. The messages exchanged primarily transfer timestamped data and must contain: • Source Timestamp: A timestamp variable identifying the moment in time when the message is generated by the source. • Data Type: Data type of the data parameter (e.g. Binary, Double, String, etc.). • Value: Value of the sensor data or system property reading.
	Standards	FDI, OPC UA, MQTT

Table 143. Data Flow 1 (DF1) Details

Details	Flow	DF2
	Components	From: Machine Tools To: Data Collecting
	Solutions	• 4-i4QTN • 9-i4QDiT
	Message Data Structure	The messages exchanged through this interface involve event notification messages describing manufacturing process events registered in industrial variables programmed in manufacturing equipment controllers (e.g. a change in a machine configuration parameter). As in the previous interface, the messages exchanged contain timestamped data: • Source Timestamp: A timestamp variable identifying the moment in time when the message is generated by the source. • Data Type: Data type of the data parameter (e.g. Binary, Double, String, etc.). • Value: Value of the system property reading.
	Standards	FDI, OPC UA, MQTT

Table 144. Data Flow 2 (DF2) Details

Details	Flow	DF3
	Components	From: IT Systems To: Data Collecting
	Solutions	• 9-i4QDiT

	Message Data Structure	The messages exchanged through this interface primarily involve logging messages for IT system data monitoring and event notification messages describing manufacturing process events or changes in production KPIs collected from IT Systems like ERPs or MES systems (e.g. change of production order). The log messages must contain: • Source Identifier: An identifier of the data system property the event message refers to. • Severity: Indication of the level of severity of the logging messages (e.g. informative, warning, error). • Message: Logging message body. Regarding the manufacturing process events messages, they must contain: • Event Identifier: An identifier of the manufacturing process event the message refers to. • Asset Identifier: An identifier of the manufacturing asset (e.g. Workcenter or work unit) the event refers to. • Source Timestamp: A timestamp variable identifying the moment in time when the event occurred. • Data Type: Data type of the data that represents the event (e.g. Binary, Double, String, etc.). • Value: Value of the sensor data or system property reading that represents the event. Regarding the manufacturing KPI messages, they must contain: • Asset Identifier: A unique identifier of the manufacturing asset (e.g. Workcenter or work unit) the KPI event refers to. • KPI Identifier: A unique identifier of the KPI. • Source Timestamp: A timestamp variable identifying the moment in time when the KPI is calculated • Value: Value of the KPI.

	Standards	MIMOSA, KPI-XML, OpenO&M…

Table 145. Data Flow 3 (DF3) Details

Details	Flow	DF4
	Components	From: Orchestration Management To: Data Collecting
	Solutions	• 8-i4QBDA • 9-i4QDiT
	Message Data Structure	Messages exchanged through this interface contain the configuration information necessary to enable data ingestion, managing the connectivity between data collection components and manufacturing assets. Thus, messages must contain: • Source Identifier: Unique identifier of the data source. • Data Source Configuration: Specific service configuration information to collect data from the data source (service specific).
	Standards	FDI, OPC UA, MQTT

Table 146. Data Flow 4 (DF4) Details

Details	Flow	DF5
	Components	From: Data Collecting To: Data Brokering and Storage
	Solutions	• 15-i4QDiT • 8-i4QDR • 2-i4QBC
	Message Data Structure	Messages exchanged through this interface represent the actual ingestion of data from manufacturing assets and smart products, made available for other functional components through the data brokering services and/or stored in the data storage services to be used at a later stage. Optionally, the ingestion of data can be performed through the interfaces provided by the i4QBC solution to enable accountability. In this sense, the i4QDiT solution acts as a gateway between the services provided by the manufacturing assets and smart products and the data brokering services used by all other solutions in

		the platform and edge tiers. Messages exchanged through this interface include the following information elements: • Source Identifier: Unique identifier of the source of the message (e.g. sensor, machine controller, IT system). • Quality of Service (QoS) indicator: A numeric indicator that assesses the quality of service of the reading. • Source Timestamp: A timestamp variable identifying the moment in time when the message is generated by the source. • Data Type: Data type of the data parameter (e.g. Binary, Double, String, etc.). • Value: Value of the sensor data or system property reading.
	Standards	OData, OpenAPI, AsyncAPI

Table 147. Data Flow 5 (DF5) Details

Details	Flow	DF6
	Components	From: Data Brokering and Storage To: Platform Tier, Edge Tier and Enterprise Tier Services
	Solutions	• 8-i4QDR • 10-i4QDA • 11-i4QBDA • 12-i4QAD • 13_i4QAI • 15-i4QIM • 16-i4QDT • 17-i4QPQ • 18-i4QQD • 19-i4QPA • 21-i4QLRT
	Message Data Structure	Messages exchanged through this interface represent the exchange of data between edge, platform and enterprise tier services using the data access services provided by the i4Q Data Repository solution. Messages include configuration information to enable data exchange, logging messages, and data collected from smart manufacturing assets and smart

		products through the data collecting functional components. Messages through this interface include: • Entity Identifier: Identifier of the data entity this message refers to. • Method: Operation (Create, Request, Update, Delete) to perform on the data entity this message refers to. • Data: For Create and Update methods, it contains the data of the entity to be created or updated.
	Standards	OData, OpenAPI, AsyncAPI

Table 148. Data Flow 6 (DF6) Details

3.2. Control Flow

Several i4Q solutions consume data from the system under operation with a monitoring and control purpose. These solutions, usually placed in the Platform tier, or even in the Enterprise tier, can trigger several control actions that can have different purposes, for instance, from corrective to preventive. To trigger these actions, they process, interpret, model or, in general, make sense out of the data coming from the "Assets and Smart Products" and the Edge tiers through the data flows described in the previous section. For example, the control process may require executing fine-grained close-loops, applying rules and logic or applying mathematical models obtained through machine learning or other AI techniques. According to the application of these techniques, when a control action is required, these solutions will exercise control over the physical system through a set of physical actuators, closing the loop started with the data flows with a series of control flows that will define the actions or actuations required to achieve a particular purpose.

In particular, four main different flows have been identified from the tasks stated in the usage viewpoint and, in the long run, derived from the pilot use cases defined in i4Q. These control flows are "Update configuration" (CF1), "Stop Production" (CF2), "Restart Production" (CF3) and "Rework/Remanufacture" (CF4). For each one of these control flows this section lists the components acting as potential sources and the particular solutions triggering the flow, its potential sinks, a proposed data structure, standards that could be involved and a series of additional comments, mainly referring to examples of application of these control flows. Similarly, these control flows are also depicted in the i4Q reference architecture diagram. Table 149 - Table 152 and Figure 23 - Figure 26 show this information for each one of the four control flows identified.

Details	Flow	CF1 - Update configuration
	Components	From: “Monitor and diagnostics” or the “Data Analytics and Services” components of the “Platform Tier” or the “Simulation and Optimization”

		component of the "Enterprise Tier". To: “Machine Tools”, “IT Systems” or “Robots” components of the "Assets and Smart Products" tier
	Solutions	The following solutions could be the origin of the control flow: • 15-i4QIM • 18-i4QQD • 20-i4QLRT • 10-i4QDA • 11-i4QBDA
	Data Structure	It includes, at least, the following data: • **Id**: an identifier of the control command (alphanumeric string). • **Timestamp**: a timestamp of the moment in which the control command is issued (date, time and time zone). • **Params**: dictionary with parameters of the control command (set of (key-value) pairs). Optional data: • **Hrd**: a human-readable description of the command (string).
	Standards	Some of the i4Q solutions involved are associated with several proposed standards. In particular: • 11-i4QBDA: KPIML, OpenO&M • 13-i4QQD: AutomationML, CAEX, COLLADA
	Additional comments	After processing sensor data, the decission of modifying or updating the configuration of a machine is taken. Depending on the result or use case a message may be displayed in an HMI.

Table 149. CF1 - Update Configuration.

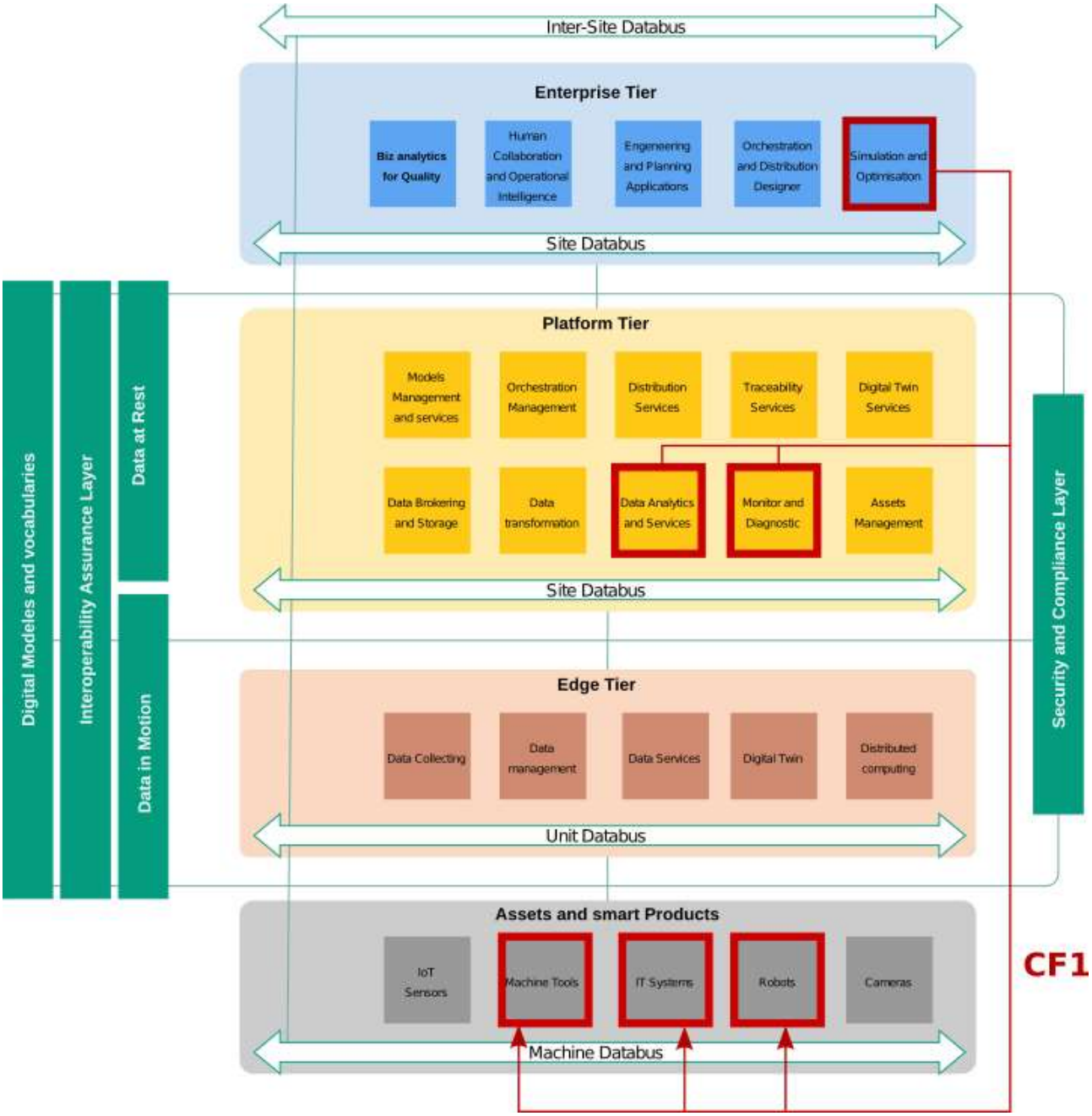

Figure 23. Components that could be involved in CF1.

Details	Flow	CF2 – Stop Production
	Components	From: "Data Analytics and Services" or the “Monitor and Diagnostic” components of the “Platform Tier”, or the "Orchestration and Distribution services" of “Enterprise Tier”. To: "Machine Tools", “IT Systems” or “Robots” components of the "Assets and Smart Products" tier.
	Solutions	The following solutions could be the origin of the control flow: • 10-i4QDA • 11-i4QBDA • 13-i4QAI • 15-i4QIM
	Data Structure	It includes, at least, the following data: • Id: an identifier of the control command (alfanumeric string). • Timestamp: a timestamp of the moment in which the control command is issued (date, time and timezone). • Params: dictionary with parameters of the control command (set of (key,value) pairs). Optional data: • Hrd: a human-readable description of the command (string)
	Standards	Some of the i4Q solutions involved are associated with several proposed standards. In particular: • 11-i4QBDA: KPIML, OpenO&M • 13-i4QAI: Open API
	Additional comments	From a predictive off-line analysis of sensor data, a decission is taken. Depending on the result (ok, warning or ko) a message is displayed in an HMI and a action to stop the machinery is issued.

Table 150. CF2 - Stop Production.

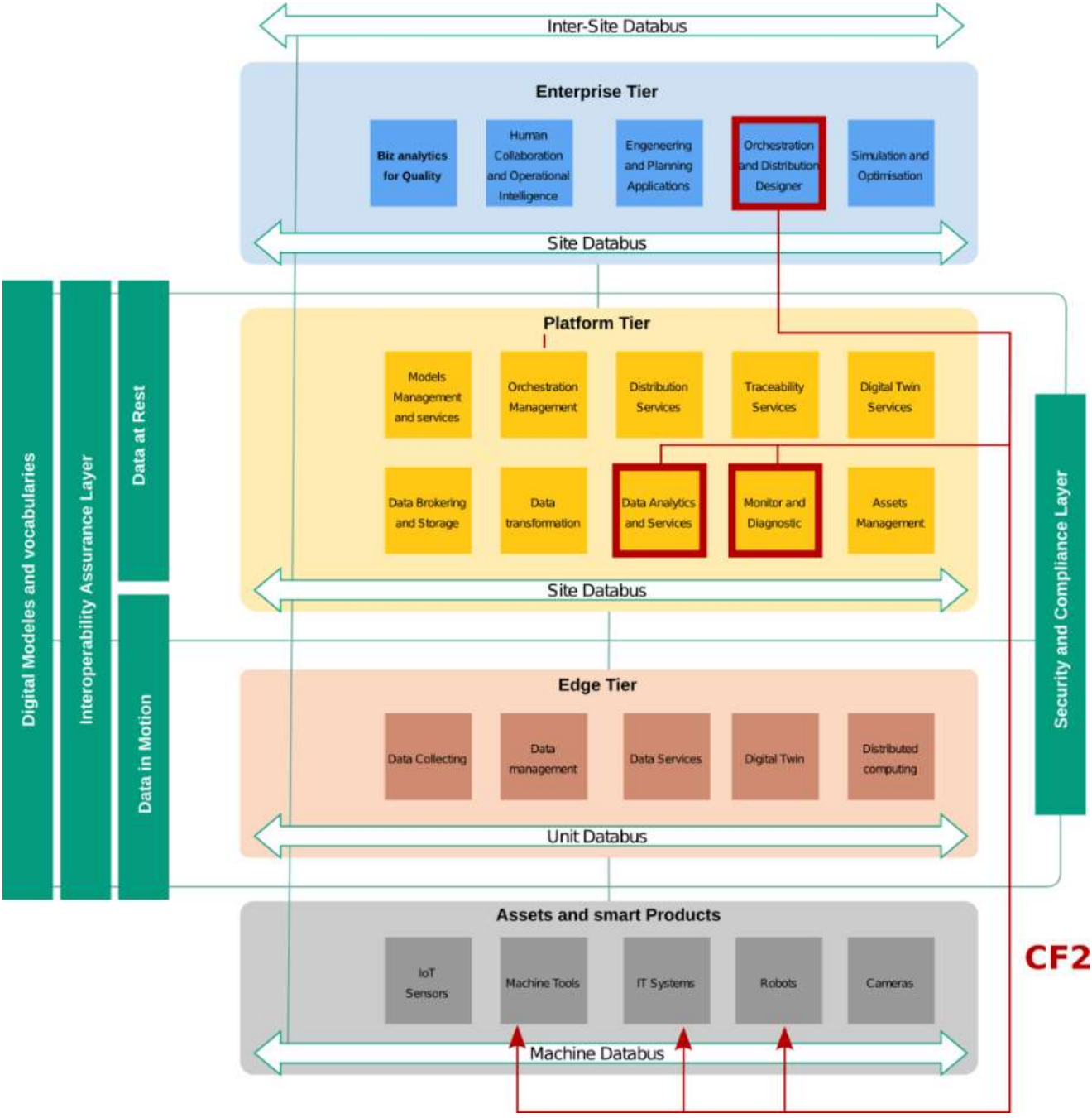

Figure 24. Components that could be involved in CF2.

Details	Flow	CF3 – Resume production
	Components	From: “Monitor and diagnostics” from the components of the “Platform Tier” or the “Simulation and Optimisation” component of the "Enterprise Tier”. To: “Machine Tools”, “IT Systems” or “Robots” components of the "Assets and Smart Products" tier
	Solutions	• 18-i4QQD • 20-i4QLRT
	Data Structure	It includes, at least, the following data: • Id: an identifier of the control command (alphanumeric string). • Timestamp: a timestamp of the moment in which the control command is issued (date, time and timezone). • Params: dictionary with parameters of the control command (set of (key,value) pairs). Optional data: • Hrd: a human-readable description of the command (string).
	Standards	Some of the i4Q solutions involved are associated with several proposed standards. In particular: • 13-i4QQD: AutomationML, CAEX, COLLADA
	Additional comments	Once a machine has been fixed/maintained or a production issue resolved, resume the production process.

Table 151. CF3 - Resume production.

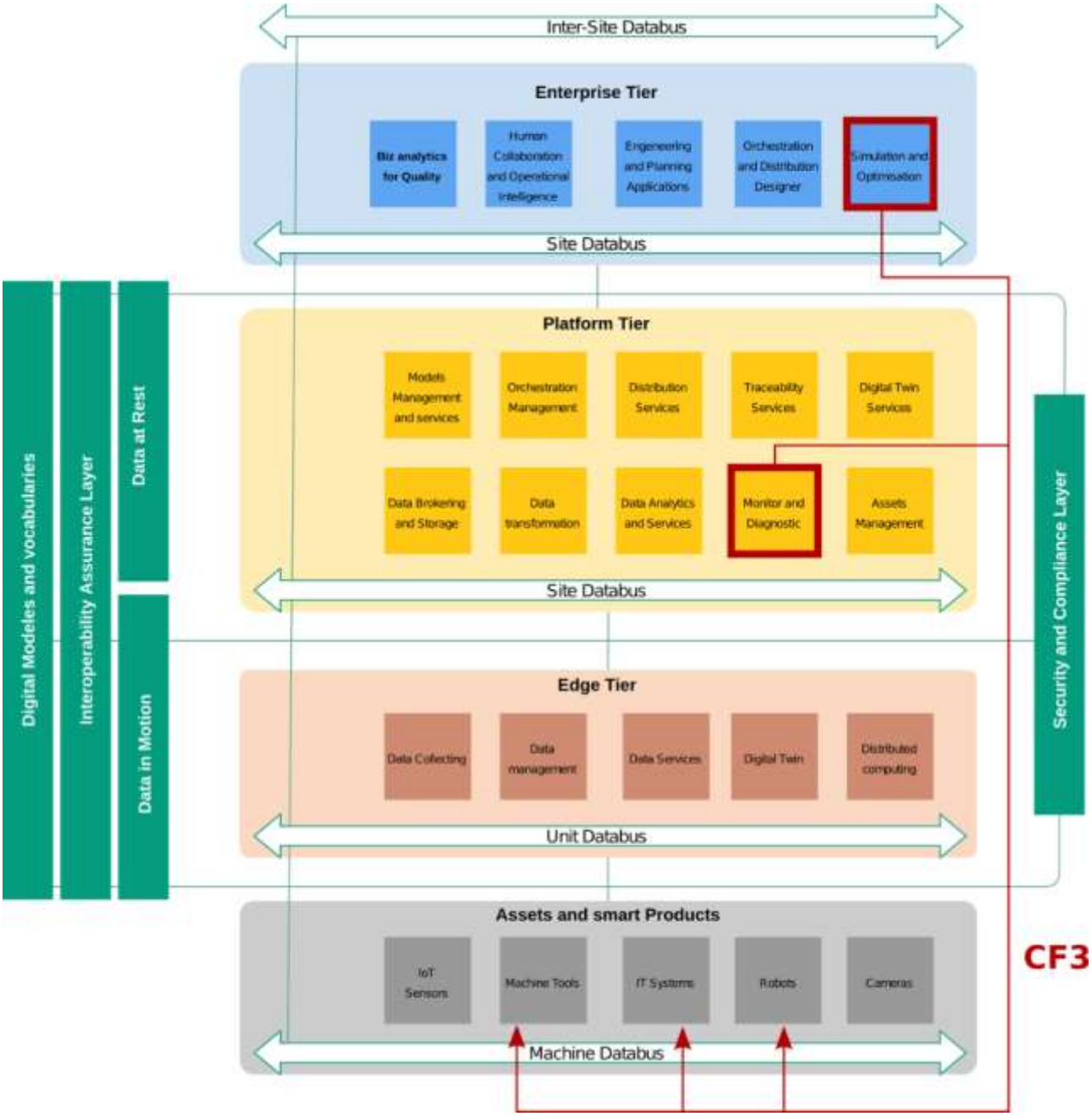

Figure 25. Components that could be involved in CF3.

Details	Flow	CF4 – Rework/Remanufacture
	Components	From: “Monitor and diagnostics” from the components of the “Platform Tier” or the “Simulation and Optimisation” component of the "Enterprise Tier”. To: “Machine Tools”, “IT Systems” or “Robots” components of the "Assets and Smart Products" tier
	Solutions	• 18-i4QQD • 20-i4QLRT
	Data Structure	It includes, at least, the following data: • Id: an identifier of the control command (alpha-arithmetic string) • Timestamp: a timestamp of the moment in which the control command is issued (date, time and timezone) • Params: dictionary with parameters of the control command (set of (key,value) pairs) Optional data: • Hrd: a human-readable description of the command (string)
	Standards	Some of the i4Q solutions involved are associated with several proposed standards. In particular: • 13-i4QQD: AutomationML, CAEX, COLLADA
	Additional comments	When the result of the analysis of a piece is not OK, try to rework it or remanufacture it.

Table 152. CF4 - Rework/Remanufacture.

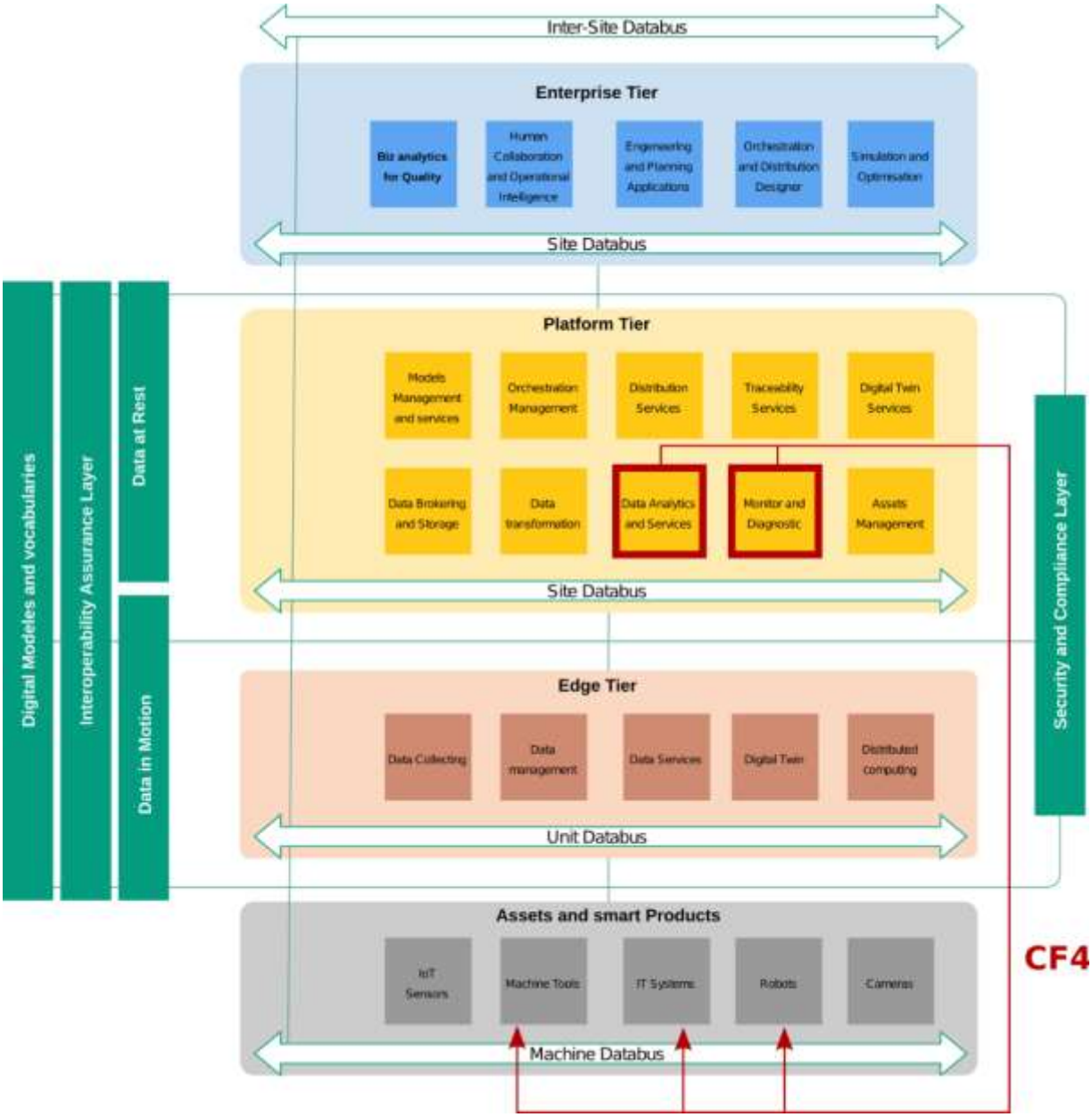

Figure 26. Components that could be involved in CF4.

3.3. Workload Distribution Flow

The **workload distribution architecture** uses distribution services in the reference architecture to scale solutions horizontally, primarily through the deployment of multiple instances of a resource across different network nodes. Two different architectures are considered for i4Q solutions: Cloud architecture, adopted by solutions that do not require that functional components are executed at the edge (close to the data source) and Cloud/Edge architecture for those that do require a distributed execution environment.

In the cloud, workload distribution is achieved through the use of load balancing functions that provide runtime logic that distributes the workload across the available IT assets. This model can be applied within the architecture with reference to the different IT resources required, as it is also commonly used with cloud storage devices, virtual servers or cloud services. In this sense, **the workload architecture** model basically functions as follows; Resource A and Resource B are exact copies of the same resource. Inbound requests from consumers are handled by the load balancer which forwards the request to the appropriate resource dependent on workload being handled by each resource.

Referring to the Industrial IoT architecture, you will find an Edge/Cloud model. This architecture allows concentrating device resources at the edge, processing data at the edge, filtering it and sending it to the cloud. On the other hand, it allows scalability by adding new nodes for each of the resources found and thus deploy new services.

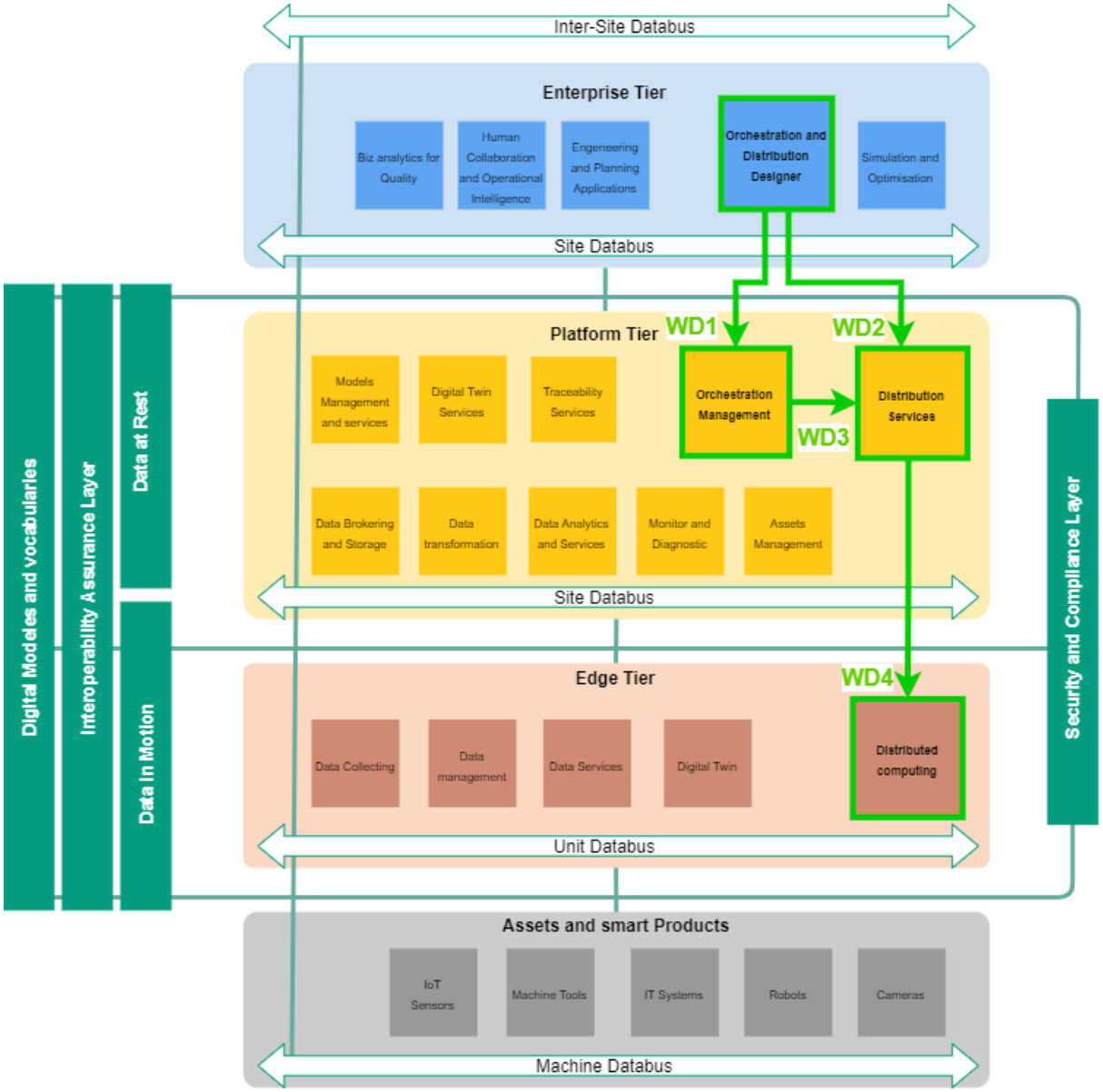

Figure 27. Workload Distribution Flow diagram

Details	Flow	WD1– Build
	Components	From: Orchestration and Distribution Designer To: Distribution Services
	Solutions	9 - i4QAD 11 - i4QBDA 20 - i4QLRT
	Data Structure	This interface is used for the transfer of distributable objects (workload images, binary format to transfer AI models, simulation models, analytic dashboards, etc) from the orchestration and distribution design functions of the i4QBDA and i4QAD solutions to the repositories (package and container registries) that

		store the distributable objects available in the distribution service component. The messages in this interface include, at least, the following data: • Name: Name of the distributable object. • Binary data: Binary data of the distributable object. • Version Tag: Version tag of the release.
	Standards	ONNx, FMI, PFA
	Additional comments	NA

Table 153. Workload Flow 1 (WD1) - Build Details

Details	Flow	WD2– AI model distribution management
	Components	From: Orchestration and Distribution Designer To: Orchestration Management
	Solutions	• 10 - i4QAI • 11 - i4QEW
	Data Structure	This interface is used to transfer the configurations defined by a human administrator to control the distribution of AI models to the edge and the edge workloads placement and deployment. The messages to be exchanged include rules and policies to configure the distribution of workloads and models and include: • Distributable object name: Name of the object (e.g. AI model, analytic dashboard) to distribute. • Placement rules: High level rules (e.g. Minimum computing resources) to regulate the placement and distribution of workloads and models. • Placement policies: Settings and procedures based on rules that the distribution must comply.
	Standards	ONNx, PFA

Table 154. Workload Flow 2 (WD2) - AI Model Distribution Management Details

Details	Flow	WD3– AI model distribution Control

	Components	From: Orchestration Management To: Distribution Services
	Solutions	• 10 - i4QAI • 11 - i4QEW
	Data Structure	There is not any data structured, only have the interface used to control the distribution of workloads to the edge.
	Standards	ONNx, FMI, PFA
	Additional comments	NA

Table 155. Workload Flow 4 (WD3) - AI Model Distribution Control Details

Details	Flow	WD4– AI model distribution
	Components	From: Distribution Services To: Distributed Computing
	Solutions	• 16 – i4QDT • 17 – i4QLRT
	Data Structure	This interface is used to transfer workloads and models to the edge (distributed computing) through the distribution services, orchestrated by the edge workload distribution functions of the i4QEW solution. The messages exchanged through this interface include: • Installation commands: Instructions to install and configure the workloads or model. When an edge node installs a workload or model, it will pull the distributable object from the registries of the distribution services component
	Standards	ONNx, FMI, PFA
	Additional comments	NA

Table 156. Workload Flow 4 (WD3) - AI Model Distribution Control Details

4. Mapping of Functional Components to User Tasks

After presenting the different functional components of those i4Q solutions where it applies in Section 2, Section 4 elaborates on the relation between solutions, their functional components, and the tasks that were identified in **D2.4 usage viewpoint**.

The aim of this section is to describe to which functions or functional components of each solution a task is assigned. As a result, the requirements and usage considerations will be reflected in the design of the functional components and in the choice of technologies and the actual implementation of the system. In addition, the characteristics of the system will be subject to different acceptance criteria, so that their level of compliance can be assessed and measured.

The following series of tables present the functional components to be mapped to a task, as well as the acceptance criteria required for its validation, for each one of the tasks that may potentially involve the use of one or multiple i4Q solutions. Note that, as mentioned, not all the tasks identified in the usage viewpoint is listed below, only those involving the use of i4Q solutions.

ID	T00	Task Name:	Produce (manufacture, build, ensemble, etc.) an item (part, product, etc.).
Solution		Functional Component Map	Acceptance Criteria
$i4Q^{IM}$		Monitor as a standalone.	The data related to the operation of the production process is correctly acquired.
		Trigger alert.	The proper alert is produced when the production process is not working properly.
$i4Q^{PQ}$		Pre-process sensor data.	The data from the sensor is processed properly.
		Inform process owner about insufficient process quality.	The proper alert is produced when the production process is not working properly.

Table 157. Solutions and functional components involved in T00

ID	T01	Task Name:	Sense data during the item production process.
Solution		Functional Component Map	Acceptance Criteria
$i4Q^{DIT}$		Import sensor data.	The data from the sensor is imported properly.
		Aggregate Data.	The pre-processed data is composed properly as datasets.
$i4Q^{IM}$		Monitor as a standalone.	The data related to the operation of the production process is correctly acquired.
		Trigger alert.	The proper alert is produced when the production process is not working properly.

Table 158. Solutions and functional components involved in T01

ID	T02	Task Name:	Predict the quality of an item in production time.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{PA}		Collect Data.	The required input data for the simulations are collected.
		Perform simulation ranking.	A number of simulations (from best to worse) are provided.
i4Q^{QD}		Evaluation of the algorithm.	The model is fed with sensor data.
		Visualize data.	The results of the evaluation are produced.
i4Q^{DR}		Save structured data.	The results of the evaluation are properly saved in the data repository.
i4Q^{BC}		Pre-process BC request.	The data being submitted must match what is expected by the API/Smart contract.

Table 159. Solutions and functional components involved in T02

ID	T08	Task Name:	Admit the item.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{AD}		Deliver Data Visualizations.	Quality data is retrieved and used to decide about.
i4Q^{QE}		Graphical User Interface	Relevant data must have been input into the solution beforehand.
		Chatbot widget	Relevant data must have been input into the solution beforehand.
		User authentication.	Quality inspectors must be able to log in so they can retrieve data.

Table 160. Solutions and functional components involved in T08

ID	T09	Task Name:	Reject the item.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{AD}		Deliver Data Visualizations.	Quality data is retrieved and used to decide about.
i4Q^{QE}		Graphical User Interface	Relevant data must have been input into the solution beforehand.
		Chatbot widget	Relevant data must have been input into the solution beforehand.
		User authentication.	Quality inspectors must be able to log in so they can retrieve data.

Table 161. Solutions and functional components involved in T09

ID	T10	Task Name:	Correct the item.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{AD}		Deliver Data Visualizations.	Quality data is retrieved and used to decide about.

Table 162. Solutions and functional components involved in T10

ID	T11	Task Name:	Decide whether to modify machine parameters.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{IM}		Monitor other analytic solutions.	There must be other analytic solutions monitoring the system.
		Monitor as a standalone.	The system being monitored must offer metrics or KPIs that are sufficient to trigger alerts if necessary.
		Trigger alert.	The system must have metrics, KPIs or components on which an alert can be triggered.
i4Q^{QD}		Evaluation of the algorithm.	A model must have been trained, validated and made available for use. Data to feed the corresponding algorithm must be available.
		Visualize data.	The algorithm must have run successfully.

Table 163. Solutions and functional components involved in T11

ID	T12	Task Name:	Propose new values for machine parameters.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{LRT} i4Q^{LRG}		Get Algorithm Result.	There must be a model/digital twin previously trained, available and selected. The algorithm must have been run successfully.
		Manage Data Access Configuration.	The operator must know which parameters to modify (if needed) to run the algorithm.
		Run Algorithm.	The parameters input must be correct and the algorithm must run successfully.
		View Result.	The algorithm must have run successfully.

Table 164. Solutions and functional components involved in T12

ID	T13	Task Name:	Decide if the problem can be solved with automatic changes.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{QD}		Evaluation of the algorithm.	A model must have been trained, validated and made available for use. Data to feed the corresponding algorithm must be available.
		Visualice data.	The algorithm must have run successfully.

Table 165. Solutions and functional components involved in T13

ID	T14	Task Name:	Automatically modify machine parameters.
Solution		Functional Component Map	Acceptance Criteria
i4Q[LRT]		Get Algorithm Result.	There must be a model/digital twin previously trained, available and selected. The algorithm must have been run successfully.
		Manage Data Access Configuration.	The parameters must be available and accessible.
		Run Algorithm.	The parameters input must be correct and the algorithm must run successfully.
		View Result.	The algorithm must have run successfully.

Table 166. Solutions and functional components involved in T14

ID	T16	Task Name:	Propose the shutdown of the machine or the production process.
Solution		Functional Component Map	Acceptance Criteria
i4Q[IM]		Monitor other analytic solutions.	There must be other analytic solutions monitoring the system.
		Monitor as a standalone.	The system being monitored must offer metrics or KPIs that are sufficient to trigger alerts if necessary.
		Trigger alert.	The system must have metrics, KPIs or components on which an alert can be triggered.
i4Q[PQ]		Evaluate process quality data.	A model must have been previously trained and the quality conformity ranges must have been properly defined.
		Evaluate model parallel with new machine data.	A model must have been trained and the solution must be properly connected to the system under evaluation. Similarly, the algorithm must be executed successfully.
		Visualize data.	Algorithm execution has to be successful.
		Inform process owner about insufficient process quality.	Algorithm execution has to be successful.
i4Q[QD]		Evaluation of the algorithm.	A model must have been trained, validated and made available for its use. Data to feed the corresponding algorithm must be available.
		Visualice data.	The algorithm must have run successfully.
i4Q[LRT] i4Q[LRG]		Get Algorithm Result.	There must be a model/digital twin previously trained, available and selected. The algorithm has had run successfully.
		Manage Data Access	The parameters must be available and

	Configuration.	accessible.

Table 167. Solutions and functional components involved in T16

ID	T17	Task Name:	Compute derived quality evaluation data (e.g., FFT).
Solution		Functional Component Map	Acceptance Criteria
i4Q^{DA}		Execute methods.	Methods must have been properly prepared previously.
		Deliver Results.	Data inputs for the selected methods must have been properly made available and prepared and the methods must have been executed successfully.
i4Q^{DIT}		Import sensor data.	The data from the sensor is imported properly or it is already available from previous tasks.
		Aggregate Data.	The pre-processed data is composed properly as datasets and it is coherent, so it can be aggregated.
		Transform Data.	Datasets must have been made available.
		Harmonize Data.	Datasets must have been made available.

Table 168. Solutions and functional components involved in T17

ID	T18	Task Name:	Maintain the production equipment (disassemble, calibrate, repair, reassemble, test, etc.).
Solution		Functional Component Map	Acceptance Criteria
i4Q^{LCP}		There are no functional components associated to this solution.	Production maintenance should be done following the guidelines and documentation.

Table 169. Solutions and functional components involved in T18

ID	T19	Task Name:	Publish model
Solution		Functional Component Map	Acceptance Criteria
i4Q^{AI}		Handle incoming requests.	The provided model must be prepared to be delivered to all the possible targets.
		Determine proper placement.	Information regarding workload, incoming data and policies must be available to decide the proper targets.
		Distribute Model.	Edge targets must be ready to receive a new working model.
i4Q^{EW}		Hanide incoming requests.	The provided model must be prepared to be delivered to all the possible targets.

	Determine proper placement.	Information regarding workload, incoming data and policies must be available to decide the proper targets.
	Distribute Model.	Edge targets must be ready to receive a new working model.

Table 170. Solutions and functional components involved in T19

ID	T20	Task Name:	Store data
Solution		Functional Component Map	Acceptance Criteria
i4Q^{DR} i4Q^{DRG}		Access control.	Only authorized users should be able to store data in the data repository.
		Save structured data.	Any new model, result or data relevant to the production process should be allowed to be stored in the repository.
		Update structured data.	Once any structured data handled by the system is modified, it should be updated in the data repository.
		Save image blob.	Relevant blobs, such as production inspection data must be temporally allocated in the data repository.
i4Q^{BC}		Pre-process BC request.	The data being submitted must match what is expected by the API/Smart contract.

Table 171. Solutions and functional components involved in T20

ID	T21	Task Name:	Stop production.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{QD}		Evaluation of the algorithm.	Quality diagnosis must be capable of detecting issues that should stop the production line.
i4Q^{LCP}		There are no functional components associated to this solution.	Production stop must always follow the guidelines and documentation.
i4Q^{IM}		Trigger alert.	As soon as a critical problem is detected an alert must be triggered in order to stop the production.

Table 172. Solutions and functional components involved in T21

ID	T22	Task Name:	Train a model with the provided data.

Solution	Functional Component Map	Acceptance Criteria
i4Q[BDA]	Select Methods / Techs.	In order to train a model, it is necessary that an expert user decides the optimal methods and technologies for the use case.
	Configure Methods / Techs.	The algorithm, to be built, will require the configuration of the used methods with specific data.
	Configure data sources.	Model training and execution will require valid data sources in order to infer the results from it.
	Build software bundle.	In order to obtain the final software bundle a specific configuration and data will be required.
i4Q[DT]	Select model type.	The specific configuration of the model being trained must be available so it can be properly trained.
	Build model.	The simulation provides important information that will be used in the training stage.

Table 173. Solutions and functional components involved in T22

ID	T26	Task Name:	Admission of raw matter / goods.
Solution		Functional Component Map	Acceptance Criteria
i4Q[AD]		Create data visualization.	Information regarding specific matters or goods will be required to obtain a data visualization.
		Deliver data visualization.	Data visualization should be provided to authorized users in an intuitive and useful manner to take the best decisions possible.
i4Q[QE]		Read database.	Quality inspectors may need to retrieve all the information regarding a matter or good just received.
		Authenticate users.	Quality inspectors must be able to log in so they can retrieve data.

Table 174. Solutions and functional components involved in T26

ID	T28	Task Name:	Discard lot.
Solution		Functional Component Map	Acceptance Criteria
i4Q[QE]		CRUD on factors database.	The factors surrounding the discarding of an

		item lot must be able to be updated on the i4Q knowledgebase.
	Authenticate user.	Assemblers and Quality inspectors must be able to log in with admin rights.
i4Q^{AD}	Deliver data visualizations.	Data visualizations will be used to evaluate and verify quality metrics of the discarded lot.

Table 175. Solutions and functional components involved in T28

ID	T36	Task Name:	Prepare data.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{DIT}		Pre-process and integrate the raw data from sensors.	Users of production data will require any number of functions in order to adapt the data output for their current task (e.g., Train AI models or product cost accounting).
i4Q^{DQG}		There are no functional components associated to this solution.	The contents captured in the guidelines have to apply to the datasets used

Table 176. Solutions and functional components involved in T36

ID	T37	Task Name:	Perform analysis.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{PA}		Collect Data.	It must be able to gather production parameter for current or previous batches in order to evaluate production costs.
		Request simulations.	It must be able to simulate and analyse different configurations and scenarios.
		Define evaluation criteria.	The analyst must be able to define the production variables to be analysed.
		Visualise results.	The analyst should be able to prepare a report based on the results or export them.

Table 177. Solutions and functional components involved in T37

ID	T39	Task Name:	Check raw materials or components in stock.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{DR}		Access control.	The inventory must have read/write access to inventory-related data.
		Query.	Stock data must be accessible.

Table 178. Solutions and functional components involved in T39

ID	T41	Task Name:	Propose production schedule.

Solution	Functional Component Map	Acceptance Criteria
i4Q^{IM}	Monitor other analytics solutions.	In order to propose a new schedule, the current status of inventory, production and quality assurance teams should be retrievable.
i4Q^{PQ}	Visualize data.	The production scheduler must be able to access process and product quality information.

Table 179. Solutions and functional components involved in T41

ID	T45	Task Name:	Check if scheduled or unscheduled maintenance is needed.
Solution		Functional Component Map	Acceptance Criteria
i4Q^{IM}		Trigger alert.	Upon detecting scheduled or unscheduled maintenance, the Maintenance service scheduler and processing operators must be alerted via Human-Machine interface, email or some other way.
		Monitor as standalone.	Each machine involved in the production process will have its own maintenance schedule.
i4Q^{PQ}		Visualize data.	Process and product quality information will be available in order to make informed decisions about the required maintenance.

Table 180. Solutions and functional components involved in T45

5. Mapping Functional Components to product life-cycles stages

The objective of this section is to provide a better understanding of the life-cycle of quality control for smart manufacturing applications using i4Q solutions. The following sections describe the life-cycle of quality control applications, focusing on two critical aspects: The life-cycle of analytical solutions (e.g. optimization models, simulation models) from development to operation, and the data life-cycle of data, from collection to use and storage. The different stages considered are mapped to the function components of solutions by means of user sequence diagrams.

5.1. Life-cycle of AI solutions

The life-cycle model of AI solutions considers the following stages:

- **Datasets creation:** Data scientists create datasets that represent the problem and that can be used to train and validate the AI model.
- **Model design:** Data scientists design the AI models and algorithms that will solve the problem.
- **Model Development:** Data scientists develop the AI model.
- **Build:** Data scientists build an instance of the model so that it can be trained and validated.
- **Training:** The instance of the model is trained.
- **Test/validation:** The performance of the model is validated.
- **Deployment**: System administrators deploy the model to a runtime in the cloud or edge tier.
- **Operation**: The model is used for its design purpose (inference, optimization, simulation).
- **Evolution:** Data scientist update the model or re-train the model using new operational data.
- **Disposal:** the model is decommissioned.

There are different patterns that can be derived from this basic life-cycle model, depending on the characteristics of the model and the deployment patterns used. In this document the following have been considered:

- **Learning approach:** As the life-cycle model highlights above, many AI data-driven solutions require a training stage to learn the relationships between inputs and outputs from available datasets. There are two main learning approaches to be considered. In **batch learning**, the models are trained over the available training data set. On the other hand, in **online learning**, the models are trained as new data is made available.
- **Model training/operation pattern:** There are different combinations possible depending on the tier where the models are trained and where they are operated: Models can be trained and operated in the **cloud**, trained in the cloud (using data collected from multiple edge locations) and operated in the edge (**cloud/edge**), or trained and operated in the **edge**.
- **Edge distribution pattern:** Finally, AI models can be distributed, both for deployment and for evolution, **embedded** into the workloads that are distributed to the edge or **stand-alone,** independent of the workload distribution.

Clearly, each pattern has different trade-offs in each specific use case. An in-depth discussion of the pros and cons of each alternative is out of the scope of this document. The objective, however, is to describe how the different i4Q solutions enable these patterns and what are the interactions between them involved in the different life-cycle stages. For this purpose, different life-cycles diagrams will be developed for the i4QLRT solution.

The diagram presented in figure 29 is the development and training of an AI model. This use case is located within the "Distribution Services" subcomponent of the i4Q Solutions architecture. It is presented with an actor that is going to develop the entire process of deploying an algorithm through the i4QLRT solution within the Edge platform.

First, the actor creates a model in its code repository. Once it is created, it launches a CI/CD pipeline that, connected to the i4QLRT solution, builds a distributable file with the model and updates or creates it within the workload repository. Once this distributable file is built, the deploy process starts with the image. If the worker decides to instantiate the algorithm, it will configure the client to access the data and train the model. This data access will be provided by the subcomponent "Data brokering and storage" were the i4QDR solution is found. Finally, the model is validated through the pipeline, which will run the model on the i4QLRT solution and will return the final results.

The diagram shows the life-cycle of the i4QLRT solution within **model development** (when data scientists develop the AI model) before moving to the Build phase. Taking into account the distribution pattern at the edge. AI models are distributed for deployment embedded in workloads at the edge or independently, regardless of the workload distribution. In the diagram (figure 28), we observe the life-cycle of the i4QLRT solution in the **build** phase, when data scientists build an instance of the model in order to train and validate it. As can be seen in the diagram, the CI/CD worker exports the model, builds the trained model and updates the distributable file in the workload repository. The following use case (figure 30) **validates** the performance of the model. For this, the actor launches the test through the CI/CD pipeline, which downloads the distributable file found in the workload repository. Once it is in the CI/CD worker, it runs the model on it, getting access to all the necessary data (which can be provided by other solutions) and returns the test results. In the last use case (figure 31), it represents the **deployment** life-cycle, when system administrators deploy the model to a runtime in the edge tier.

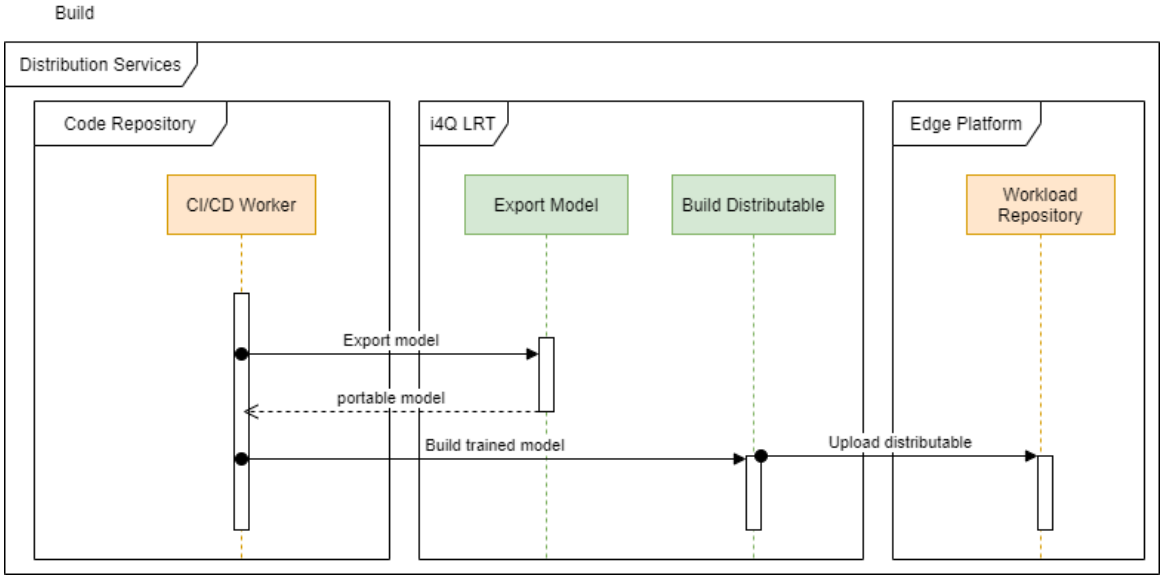

Figure 28. AI model build

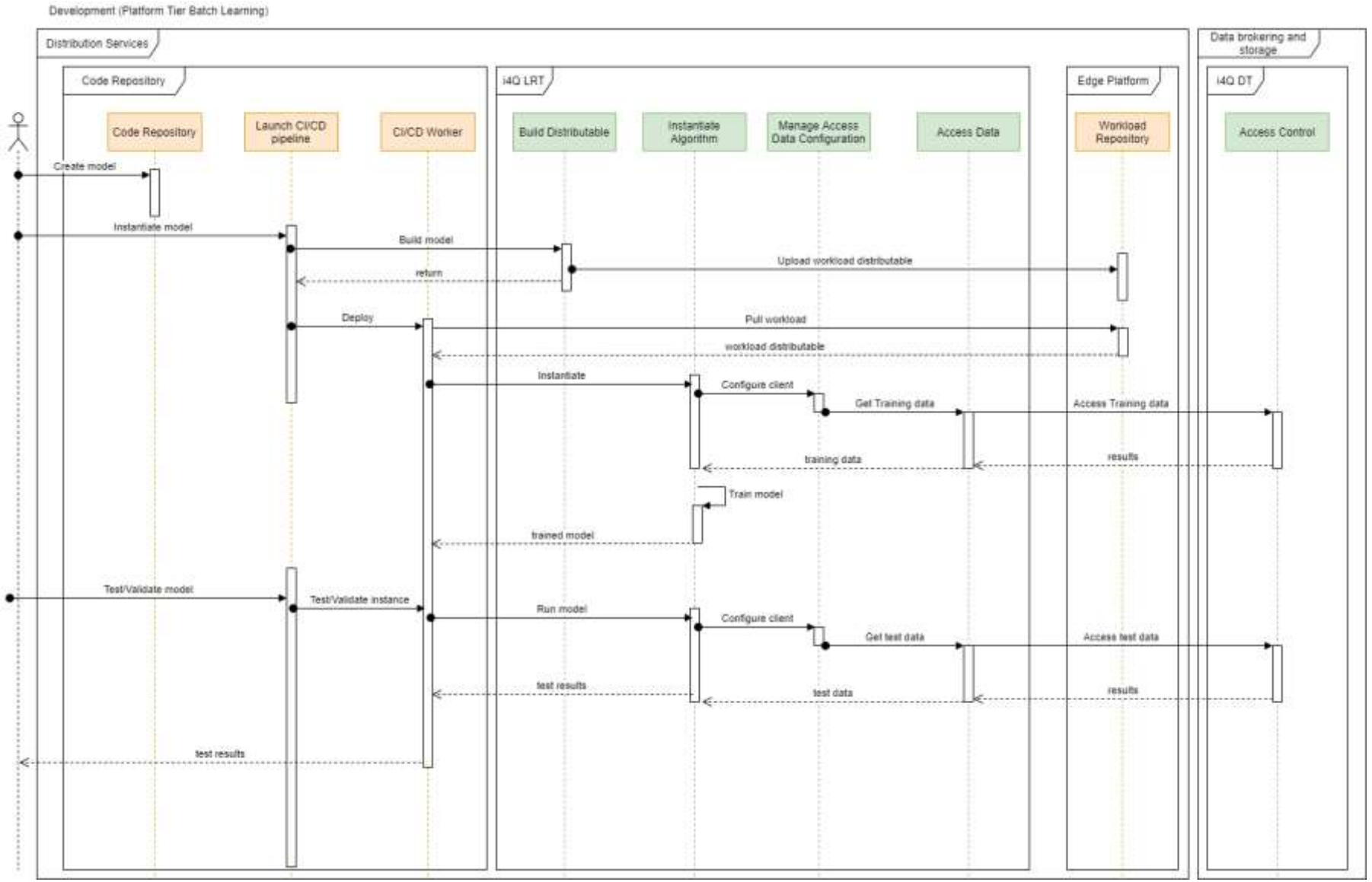

Figure 29. AI model development and training.

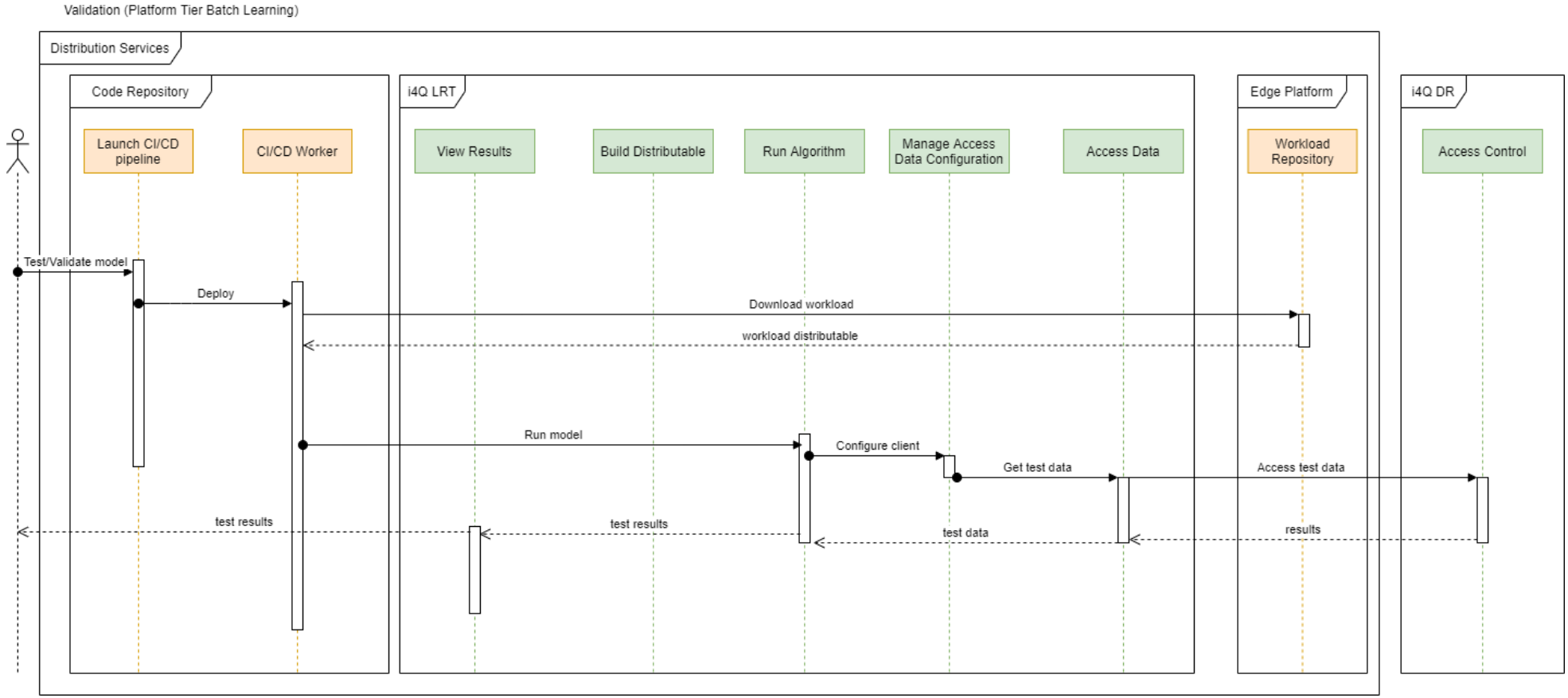

Figure 30. AI model validation

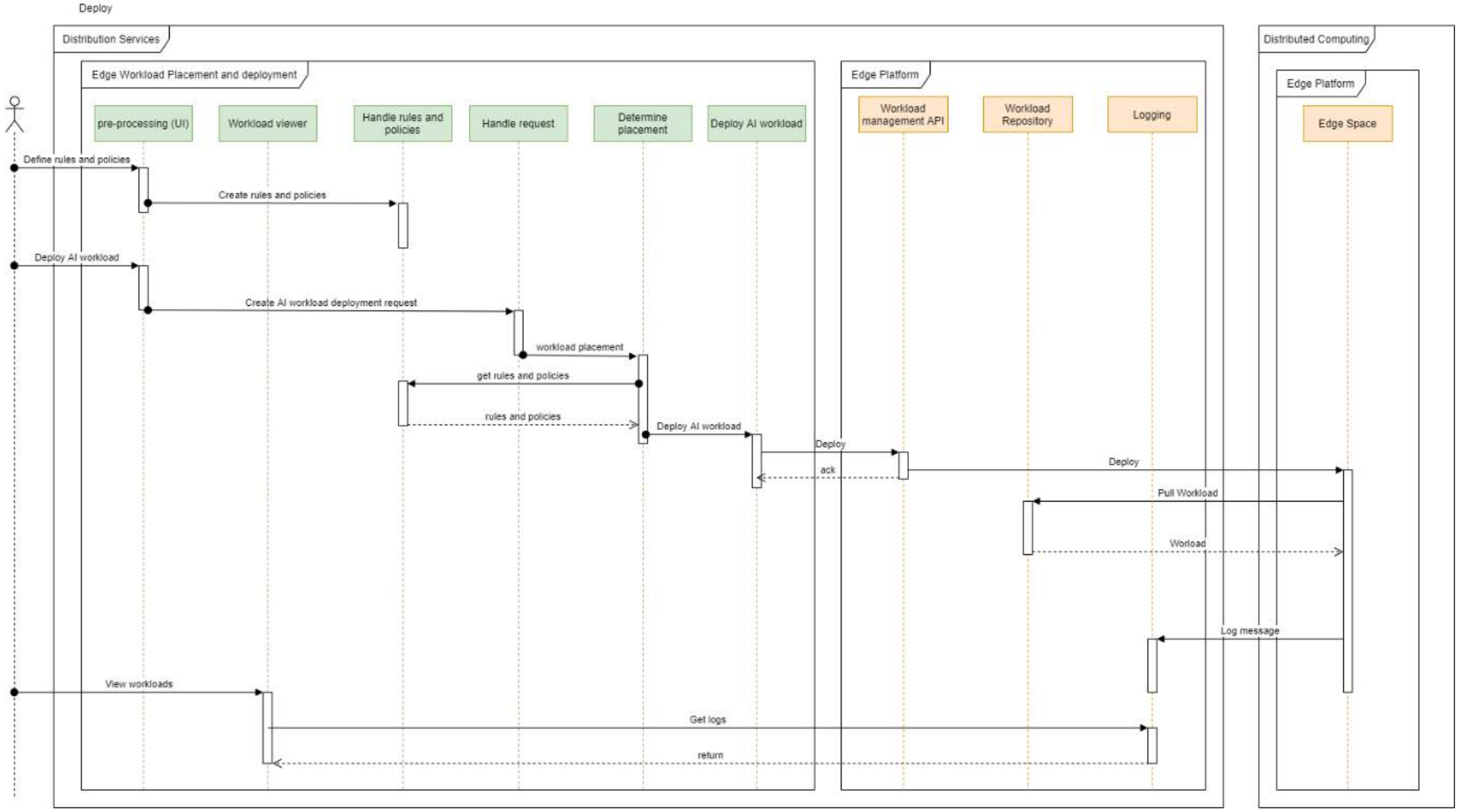

Figure 31. AI model Deploy

5.2. Life-cycle of Data solutions

The life-cycle of Data solutions is crucial for extracting the maximum utility from data. It goes from their creation or capture and storage, through their use in various processing activities, until they become obsolete and is deleted. The data produced during the operation life-cycle can be used over time in the different i4Q solutions.

There are multiple versions of a data life-cycle with differences attributable to variation in practices across domains. In this case it is relied on a 5-phase model (Create, Store, Use, Share and Delete). This deliverable analyses the functional integration of different solutions in three of these phases (Create, Store, Use and Share) linked to the operation life-cycle.

To analyse these phases in the life-cycle data, three use cases involving different i4Q solutions will be developed: a first use case where it can be found a process of storing the data obtained by the sensors and the query by the solution, a second use case where it will be seen how a solution can subscribe to a message broker and a third use case where the blockchain solution will be used for data traceability.

This use case (figure 32) involves different solutions integrated in different components of the i4Q architecture. In this way, it can get data from the sensors tier and delivers it to the services found at the platform Tier. It is aligned with the data **creation** life-cycle phase and **storage**. To understand how the solution works in this phase, we will explain how it works and the relationship between solutions using the diagram in figure 32.

The component Data collection, which will work in the same way in all three use cases, involves both the i4QTN and i4QDIT solutions. i4QTN solution ensures that digitalised processes and higher-level platforms can exchange information with guaranteed reliability and quality of service. To this end, it establishes a set of technologies focused on high-speed and reliable wireless exchange for large volumes of data with low latency. The data is provided to the i4QDIT solution. This solution is responsible for providing the imported datasets with a homogeneous structure, preparing the data for their use. This process is composed of two phases; a first phase consisting of the pre-processing of the data, including data cleaning and transformation. A second phase consists of sending the data to the different solutions.

In this case, it uses the data sent by the solution i4QDIT, in the component Data Brokering and the storage from the architecture, by the i4QDR solution. This solution is responsible for receiving, storing and serving data to the other components of the architecture. It requires credentials to access its services. Once authorized, it will be able to insert the data into the client, which will be stored in a storage service within the Tier. Finally, it encounters the i4QDT solution that wants to obtain stored data. This solution will send a request to the i4QDR solution, which after checking the access credentials, will access the storage service and return the query.

In the second use case (figure 33), a data model is defined in which a solution subscribes a message broker to use live data. This use case represents the life cycle of data **creation and use**. The first part of the data capture from the sensor to the i4QDIT solution is the same as in the previous use case. The difference is that instead of dumping the data into a solution that is

responsible for storing data, in this case, it sends the data directly to the message broker. The solution i4QPA will subscribe to this message broker.

Message brokers can validate, store, route and deliver messages to appropriate destinations. They serve as intermediaries between solutions, allowing i4QDIT to broadcast messages without knowing where the receivers are, whether they are active or not, or how many there are. This facilitates the decoupling of processes in the different i4Q solutions within the system. The most popular message brokers are RabbitMQ, Apache Kafka, Redis, Amazon SQS, and Amazon SNS.

In the last use case (figure 34), it is a data model using the i4QBC solution as a data manager. The first part of the data capture from the sensors of the i4QDiT solution and the **storage** data with the i4QDR is the same as in the first use case. The difference is that when it sends the request to storage the data in the storage service, this request is sent to the i4QBC solution. i4QBC service provides tools to ensure trust in data and full traceability; enhancing the level of trust by employing a blockchain based data service, improving acceptability by providing security and trust in the data that flows directly to the blockchain, serving as a single point of truth. Blockchain is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system. This solution focuses on the life-cycle of **sharing**, prioritising data security.

A blockchain is essentially a digital ledger of transactions that is duplicated and distributed across the entire network of computer systems on the blockchain. Each block in the chain contains a number of transactions, and every time a new transaction occurs on the blockchain, a record of that transaction is added to every participant's ledger. The decentralized database managed by multiple participants is known as Distributed Ledger. Blockchain uses in transactions a type of Ledge that allows to record with an immutable cryptographic signature called hash. Therefore, when a block changes its chain, it would be evident that it has been manipulated. Thank to this, it can be guaranteeing the traceability of the product and record all transactions in the Edge Tier.

i4QPQ is a micro-service consisting of simulations models as a service, taking as input the manufacturing resources, current production planning and process condition. It makes use of the i4QBC in order to enable traceability of product data and production data. This allows you to map part quality to process parameters. It can integrate a certificate authority for all services using the infrastructure, using public and private keys.

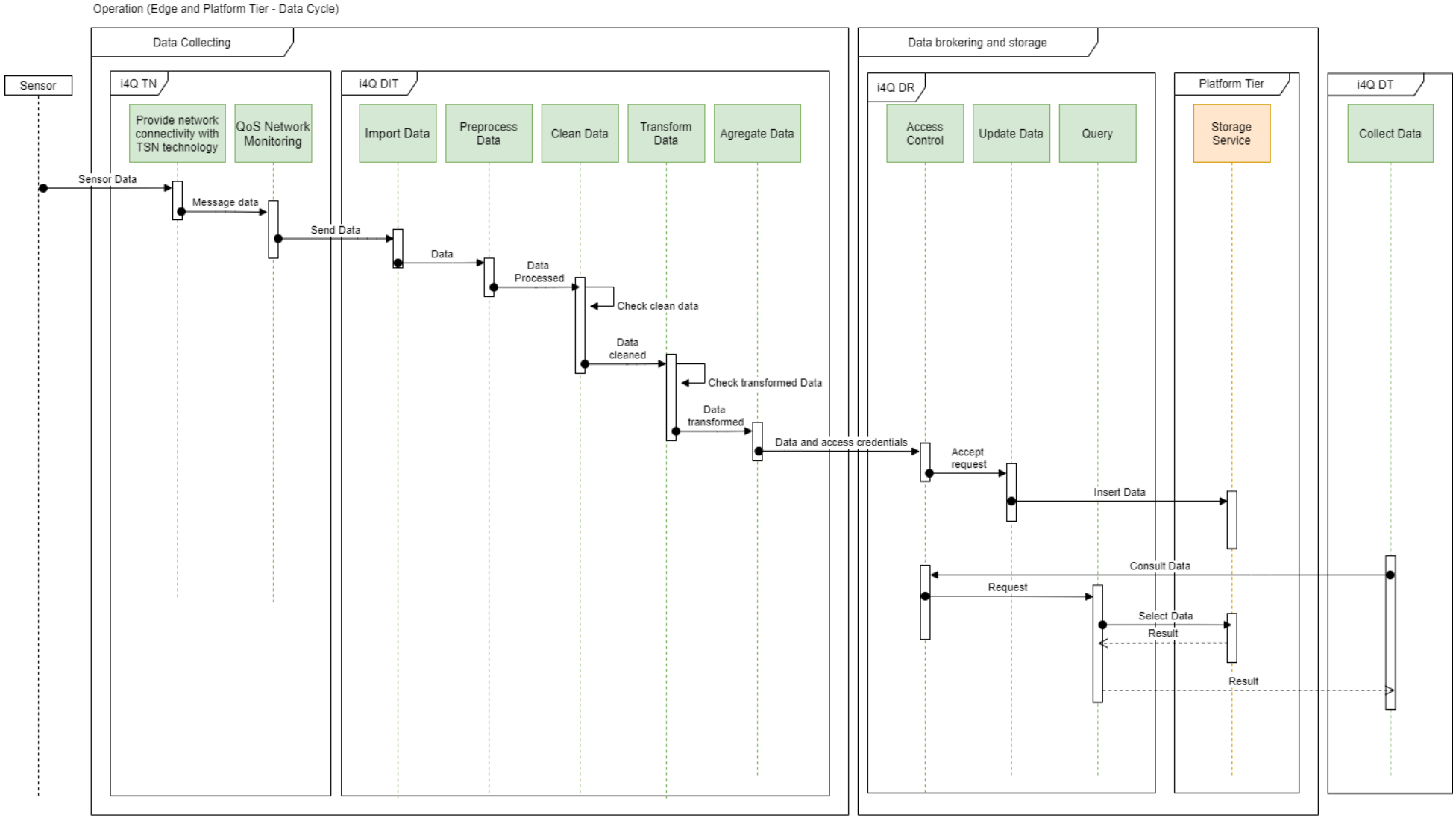

Figure 32. Data model operation 1

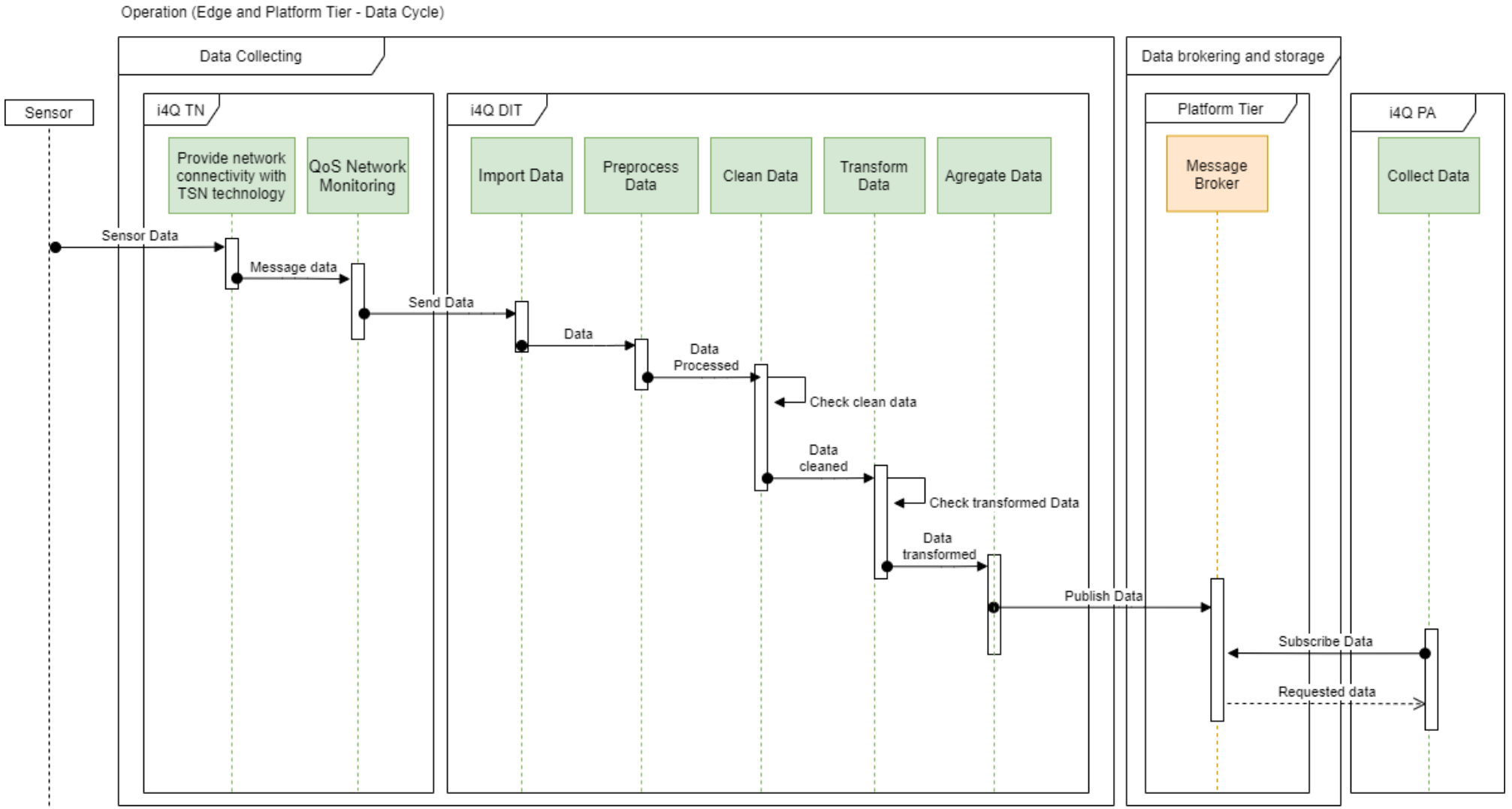

Figure 33. Data model operation 2

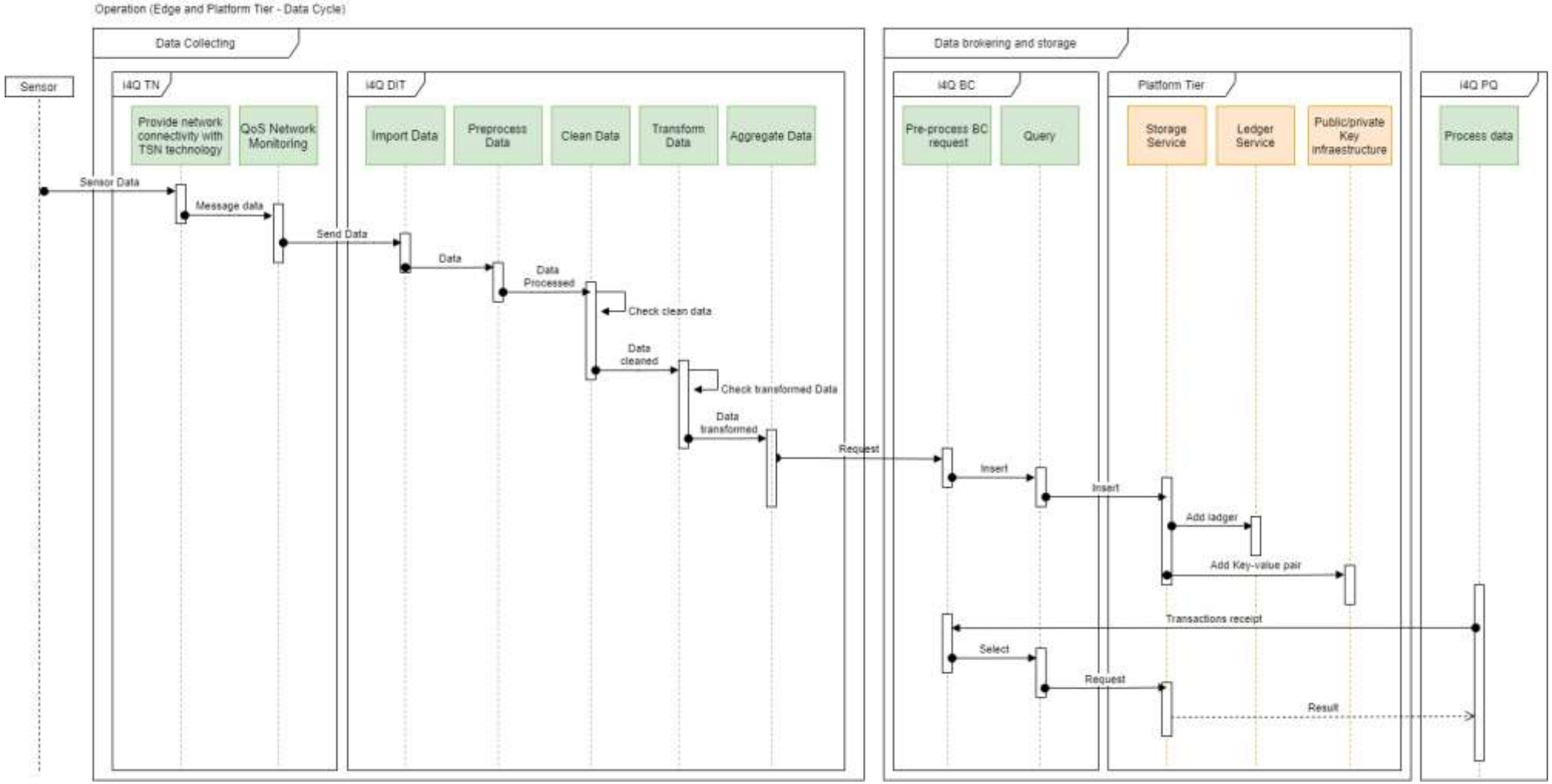

Figure 34. Data model operation 3

6. Conclusion

This document is the fifth of the six deliverables on the design of the WP2 framework and focuses on the functional viewpoint, following the usage viewpoint and preceding the implementation viewpoint. The deliverable provides an in-depth definition of the different functional components of the 17 software i4Q solutions, focusing on their interactions and their capability to provide the functionality expected in the different use cases as a whole.

Based on the information provided in the development of the functional specification, solution providers received questionnaires with questions that helped to better define the boundaries of their functional components, what do they provide and what do they expect from other i4Q solutions and external services or systems, and when are the functionalities used according to the life-cycle stages defined for i4Q solutions. Based on this information, this document provides a detailed definition of the different functional domains (control, operations, information, application and business) of the use cases, functional component maps of the different user tasks identified in the usage viewpoint, detailed definitions of the life-cycle of i4Q Solutions, standard data models, ontologies and formats that can be used to implement the different communication workflows, and sequence diagrams describing relevant aspects of the life-cycle of i4Q solutions.

References

ICC. (2018). *The Industrial Internet of Things Volume G2: Key System Concerns*. https://www.iiconsortium.org/pdf/Industrial_Internet_of_Things_Volume_G2-Key_System_Concerns_2018_08_07.pdf

ICC. (2019). *The Industrial Internet of Things Volume G1: Reference Architecture*. https://www.iiconsortium.org/pdf/IIRA-v1.9.pdf

ISO 42010. (2011). *ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description*. https://www.iso.org/standard/50508.html

OMG SYSML. (2003). *SysML Open Source Project* . https://sysml.org/